Implementation of Gaussian Process models for Nonlinear System Identification

A thesis submitted for the degree of
Doctor of Philosophy

Department of Electronics and Electrical Engineering

University of Glasgow

Keith Russell Thompson

 November 2009

Abstract

This thesis is concerned with investigating the use of Gaussian Process (GP) models for the identification of nonlinear dynamic systems. The Gaussian Process model is a non-parametric approach to system identification where the model of the underlying system is to be identified through the application of Bayesian analysis to empirical data. The GP modelling approach has been proposed as an alternative to more conventional methods of system identification due to a number of attractive features. In particular, the Bayesian probabilistic framework employed by the GP model has been shown to have potential in tackling the problems found in the optimisation of complex nonlinear models such as those based on multiple model or neural network structures. Furthermore, due to this probabilistic framework, the predictions made by the GP model are probability distributions composed of mean and variance components. This is in contrast to more conventional methods where a predictive point estimate is typically the output of the model. This additional variance component of the model output has been shown to be of potential use in model-predictive or adaptive control implementations. A further property that is of potential interest to those working on system identification problems is that the GP model has been shown to be particularly effective in identifying models from sparse datasets. Therefore, the GP model has been proposed for the identification of models in off-equilibrium regions of operating space, where more established methods might struggle due to a lack of data.

The majority of the existing research into modelling with GPs has concentrated on detailing the mathematical methodology and theoretical possibilities of the approach. Furthermore, much of this research has focused on the application of the method toward statistics and machine learning problems. This thesis investigates the use of the GP model for identifying nonlinear dynamic systems from an engineering perspective. In particular, it is the implementation aspects of the GP model that are the main focus of this work. Due to its non-parametric nature, the GP model may also be considered a ‘black-box’ method as the identification process relies almost exclusively on empirical data, and not on prior knowledge of the system. As a result, the methods used to collect and process this data are of great importance, and the experimental design and data pre-processing aspects of the system identification procedure are investigated in detail.

Therefore, in the research presented here the inclusion of prior system knowledge into the overall modelling procedure is shown to be an invaluable asset in improving the overall performance of the GP model.

In previous research, the computational implementation of the GP modelling approach has been shown to become problematic for applications where the size of training dataset is large (i.e. one thousand or more points). This is due to the requirement in the GP modelling approach for repeated inversion of a covariance matrix whose size is dictated by the number of points included in the training dataset. Therefore, in order to maintain the computational viability of the approach, a number of different strategies have been proposed to lessen the computational burden. Many of these methods seek to make the covariance matrix sparse through the selection of a subset of existing training data. However, instead of operating on an existing training dataset, in this thesis an alternative approach is proposed where the training dataset is specifically designed to be as small as possible whilst still containing as much information. In order to achieve this goal of improving the 'efficiency' of the training dataset, the basis of the experimental design involves adopting a more deterministic approach to exciting the system, rather than the more common random excitation approach used for the identification of black-box models. This strategy is made possible through the active use of prior knowledge of the system.

The implementation of the GP modelling approach has been demonstrated on a range of simulated and real-world examples. The simulated examples investigated include both static and dynamic systems. The GP model is then applied to two laboratory-scale nonlinear systems: a Coupled Tanks system where the volume of liquid in the second tank must be predicted, and a Heat Transfer system where the temperature of the airflow along a tube must be predicted. Further extensions to the GP model are also investigated including the propagation of uncertainty from one prediction to the next, the application of sparse matrix methods, and also the use of derivative observations. A feature of the application of GP modelling approach to nonlinear system identification problems is the reliance on the squared exponential covariance function. In this thesis the benefits and limitations of this particular covariance function are made clear, and the use of alternative covariance functions and 'mixed-model' implementations is also discussed.

Acknowledgements

The completion of this thesis would not have been possible but for the support and guidance of a number of individuals, and for that I would like to express my most sincere gratitude. Firstly, I'd like to thank my supervisor Professor David Murray-Smith, both for giving me the opportunity to study, and more importantly for remaining positive and having faith in me when things became desperately difficult. I will be eternally grateful for his expertise and untold levels of patience. Others who have made a great impact on my understanding of the subject and provided a great deal of technical know-how, especially at the beginning of my studies, have been those in the Computing Science Department of Glasgow University, namely Professor Roderick Murray-Smith and Dr. Gary Gray. Furthermore, it also important to acknowledge the use of software made available by Dr. Carl Rasmussen and Dr. Chris Williams.

From the Electronics Department, a special mention must be given to Tom O'Hara whose technical expertise and friendship made working in the control lab, whether performing data collection experiments or lab demonstrating, not just possible but also great fun. From my fellow PhD students, I'd especially like to thank Alistair Grant, Jill Walker, Alisdair Mitchell and David McGeoch for their friendship and camaraderie over the last few years. I'm also grateful for the enjoyable company of others in Room 507, including Jin, Gregory, Fang and Wei.

Just as important has been the support of all my friends outside the department; especially during those difficult days where I'm sure my morbid demeanour became almost intolerable! A special mention must therefore go out to those who have had to put up with me on a daily basis during the last few years, especially my brother Neil and his flatmates Calum and Russell for letting me stay rent-free when they didn't have to, Murray for his generosity and extra-curricular golf tuition, and also Hema and Sarah for giving me a warm welcome at the beginning of my years of study. Finally, the greatest of thanks must go to my parents, who without the emotional and not inconsiderable financial support, I would never have had the opportunity or even the ambition to attempt this work, let alone stubbornness to see it through to the end.

Contents

1) Introduction

The field of system identification is concerned with the development of mathematical models of real systems or processes using prior knowledge of the system and empirical data. However the problems encountered in forming an accurate representation of a system can be seen to have parallels with other forms of empirical analysis where information must be gleaned from available data. The broad topic of mathematical modelling can be seen to exist across almost all technical research disciplines with many different approaches having been developed. Most notably, ideas and techniques from the fields of statistics and computing have been embraced into the more engineering-based discipline of system identification. In many cases methodologies originating from different research fields can be seen to have similarities with one another despite being developed independently. In particular, the learning task associated with the field of artificial intelligence or adaptive systems has been a research topic for both the machine learning community as well as those from an automatic control background. Research into artificial neural networks has led to collaborative efforts between markedly different fields, such as those from a background in biological sciences and researchers from engineering and computing science.

From the fields of mathematics and statistics, the analysis of probability and error has given other research disciplines the tools with which to identify the most likely or optimal solution, such as regression algorithms, and ultimately the means to assess and validate the performance of an identified model. The use of probability theory and methods is relevant as it formally introduces the analysis of uncertainty into the modelling procedure. As the purpose of system identification is to investigate systems where knowledge is limited and of uncertain accuracy, it is therefore sensible that probabilistic methods are employed. The Gaussian Process (GP) modelling approach investigated in this thesis can be seen to originate from research into the statistics of spatial data, and in recent years has received considerable interest in the machine learning research community as a tool for nonlinear regression and classification. In the machine learning setting, the GP method has been demonstrated as a viable alternative to more established learning systems such as the neural-network approach.

The driving factor behind the continued research into alternative system identification methods is the ever-increasing demands of new and existing applications. Mathematical models of real systems are often required to assist in the design process of a system (e.g. by simulating performance, cost effectiveness etc.), and also used as the basis for the design of automatic control systems. In both these cases, the quality of the identified model will play a large role in determining the quality of the final solution. For example, in order to design a control system that maximises a systems potential performance, the mathematical model must represent the true system as closely as possible. The increase in model prediction accuracy provided by a precise mathematical model, can allow the design of a control system to be performed with a greater amount of confidence in how the system will behave when subjected to control inputs. The further development of mathematical models through the expansion of the operating range accurately represented can also facilitate the design of control systems that allow more demanding performance requirements to be realised. An example of this would be the design of modern aircraft where the development of accurate mathematical models has allowed the design of more sophisticated fly-by-wire controllers, leading to more agile fighter aircraft that can be controlled whilst operating in unstable conditions (e.g. Eurofighter Typhoon). Overall, a strong demand will always remain for methods that can improve the accuracy of a mathematical description.

The GP modelling approach is of great potential interest in the field of system identification due to a number of desirable features. A primary motive for the original surge of interest in GP models in the machine learning community is that through the model's application of Bayesian methods some of the difficulties associated with optimising complex learning systems can be bypassed through the adoption of this probabilistic approach. Such difficulties can also be seen to present themselves within the field of nonlinear system identification as more complex models have been adopted for use in representing more complex systems. As a result, these alternative GP methods have now been proposed towards problems found in system identification.

Another feature of the GP modelling approach is that through the probabilistic analysis, a predictive probability distribution rather than a single predictive estimate is the output from the model. As a result, the GP model can be seen to provide predictions of nonlinear system behaviour together with a measure of the uncertainty over each prediction. This

uncertainty or variance term has been shown to be of potential value in the design of adaptive or predictive controllers where the behaviour of a control system may be modified to reflect the uncertainty associated with the prediction. The variance output of the GP model can also be utilised to help identify regions of operating space that are not well described by the empirical data. This is a useful feature for what may be considered as a 'black-box' method of identification.

A further important feature of the GP modelling approach is that the method has been found to outperform alternative learning systems where the amount of empirical data is limited. In the identification of real systems, the amount of available data that can be used to train a model may be limited due to a number of factors. Therefore a modelling approach that can provide useful predictions in situations where little data or prior knowledge is present is something worthy of consideration. For example, in the identification of many real systems a significant problem is presented by the lack of available data in certain (typically off-equilibrium) regions of operating space. Without sufficient data, the identification of an accurate model in these regions can become impossible, and this deficiency is also passed on to any corresponding control system. However, as the GP model has been shown to perform well on problems where data is limited, it has been proposed as a potentially useful method for identification in off-equilibrium regions of operating space where more conventional methods can struggle.

1.1) Original Contributions

The application of the GP modelling approach towards the task of identifying nonlinear dynamic systems can be seen to be in its early stages with relatively few dedicated resources currently available. The majority of the existing research into GP models has concentrated more upon defining the mathematical methodology and theoretical possibilities of the approach. Furthermore, much of this existing research has been focused towards problems found in machine learning and statistics. As a result many of the examples investigated in the existing literature have utilised simulated and benchmark datasets that are not particularly demonstrative of the types of problems found in the field of nonlinear system identification. In particular, static nonlinearities remain the most popular example applications for much of the existing theory.

The primary contribution of this work is in the investigation of the GP modelling approach as a method for nonlinear system identification. Whilst in existing research the GP method has been proposed as an alternative to more established methods, a very limited amount of research has been devoted to the implementation of the approach from an engineering perspective. As a result, a particular emphasis has been placed on the practical implementation of the GP model through the identification of real laboratory systems from empirical data. In this way the usability of the approach toward general system identification problems can be made clear. Therefore, the main original contribution of this thesis has been to provide a general guide to the implementation of the GP modelling approach for system identification problems. Other original contributions made in this thesis are:

- As the implementation of the GP model is the prime objective of this thesis, a detailed discussion of the most important issues is provided. The properties of various different covariance functions, and the techniques used to optimise the GP model are discussed in detail. Furthermore, through experimental results a detailed review of the relative strengths and weaknesses of the most popular (Squared Exponential) covariance function is performed. The potential use of the Matèrn covariance function to represent less smoothly varying data is also demonstrated.
- The computational implementation of the GP model is discussed in detail. Both the size and conditioning aspects of the covariance matrix are discussed, and then related to the training data pre-processing and experimental design aspects of the system identification procedure (e.g. excitation signals, sampling rate etc.).
- The design of the training dataset used to identify the GP modelling approach is examined closely. In order to meet the size and conditioning constraints of the method, the training set must be pre-processed carefully. The most likely source of ill conditioning in the covariance matrix was identified as the presence of steady-state data. A random excitation signal that is sufficiently excited is first shown to provide a good strategy for the identification of a GP model. However, a more deterministic strategy for the design of the excitation signal consisting of a number of small-step inputs is shown to be an attractive alternative that allows more information to be included in the training dataset in a smaller space.
- The inclusion of previous inputs/outputs as additional model inputs is discussed in detail and then demonstrated. Due to the limitations placed on the size of the

training dataset, a potential discrepancy between the sample intervals of the training and test data can occur. Therefore, the process of including previous inputs/outputs is not as straightforward as in other modelling approaches.

- Alternative ‘mixed-model’ implementations of the GP model where the methods are combined with other existing methods are discussed and demonstrated using the Coupled Tanks system. These proposals are aimed at retaining the advantages of the GP modelling approach whilst overcoming some of the disadvantages.

1.2) Thesis Outline

Chapter 2

In this chapter the overall process of system identification is reviewed. Important aspects including the choice of model architecture, the role of prior knowledge, experimental design, pre-processing of training data, model optimisation and validation are discussed with references provided. This review is to provide an overview of the field rather than an in-depth discussion of all the available methods.

Chapter 3

In this chapter the theoretical background and literature of the GP modelling approach is presented in detail. This chapter begins with an in-depth discussion of the motivation behind GP models with reference to some of the model architectures discussed in the previous chapter. The concepts behind Bayesian learning are then introduced and the potential benefits in terms of dealing with model complexity issues are made clear. Next, the process and difficulties of applying a Bayesian learning framework are presented, and the mathematical peculiarities of the Gaussian process are shown to provide a solution to some of these difficulties. Finally, the task of using Gaussian processes and Bayesian learning for the purposes of regression is discussed.

Chapter 4

In this chapter the implementation of the GP modelling approach is discussed in detail. A review of the role played by the covariance function is first provided, followed by a discussion of various alternative covariance functions. Next, the optimisation of the GP model is discussed in detail. The computational implementation of the GP model is then

tackled with the challenges posed by large datasets and matrix ill-conditioning made clear. From this discussion the implementation of the GP model is described using both direct and approximate methods.

Chapter 5

In this chapter the specific challenge of implementing GP models for dynamic system identification purposes is investigated and a variety of simulated and real nonlinear dynamic systems are identified. Notable extensions to the GP model are first described, including the propagation of uncertainty and derivative observations, together with the implications for control system design using GP models. After a brief discussion of the experimental objectives (based on the research presented in the previous two chapters), the GP modelling approach is then applied to a number of simulated examples. The purpose of these simulated examples is to demonstrate the overall process of implementing the GP model and its ability to identify nonlinearities using relatively few training observations. Both static and dynamic examples are tackled. Following on from these simulated examples, two real laboratory-scale nonlinear systems are investigated: a Coupled Tank system, and a Heat Transfer system. Through these examples the identification process using real empirical data is demonstrated with problems regarding the size and conditioning of the covariance matrix tackled through experimental design, model structure definition and training data pre-processing. Utilising these methods, the two real systems are then identified to a good degree of accuracy. Further extensions investigated include the incorporation of derivative observations and an alternative ‘mixed-model’ implementation which combines the use of an analytical model of the Coupled Tanks system with a GP model. A summary of the experimental results is then provided at the end of this chapter.

Chapter 6

In this chapter the thesis is concluded by discussing some of the main points raised through the course of the preceding chapters. A general guide to the implementation of the GP modelling approach is then presented. Finally, a number of possible strategies for improving the GP modelling approach are discussed; together with recommendations for the most important areas that should be targeted in future research.

2) Nonlinear System Identification

System Identification can be seen to be the determination of a mathematical model through the use of empirical data together with prior knowledge of a particular system's characteristics. In this chapter the overall process of system identification is to be discussed where the various choices involved in developing models are examined and the practical implications are made clear. Furthermore, a background review of the various types of linear and nonlinear models and optimisation techniques currently being employed toward the task of system identification is included. This discussion is worthwhile as it provides an insight into where the GP modelling approach is to fit in amongst its alternatives.

2.1) The System Identification Process

The overall objective of the system identification process is to provide an accurate and robust approximation to the behaviour of a given system. In the identification of a suitable model of system behaviour, an iterative development process is normally undertaken where a number of design choices must be made and subsequently refined if through evaluation they are found to be unsatisfactory.

The system identification procedure may be seen to follow the loop detailed in Figure (2.1) where the process begins with the examination of any available 'a priori' knowledge of the system. This initial or prior knowledge may take the form of a detailed understanding of system characteristics, such as an analytical or physical model derived from first principles, or merely knowledge as to the availability and nature of any experimental data. Before undertaking the modelling task, it also of great importance that the intentions for any identified model are carefully considered so that any performance requirements such as accuracy, robustness or level of complexity can be met.

From a prior (or 'a priori') understanding of system components and behaviour, the human operator or modeller can then make informed decisions as to the level of data required to successfully capture the dynamics of the system. Utilising this information can then lead to the creation of any number of experimental conditions which may be

used to record empirical data. The level of prior information and availability of data will then dictate what kind of modelling strategy and therefore what kind of model structure will be the most suitable to formulate our description. It is common for a number of possible models to be proposed which offer different levels of description or performance. Through a 'criterion of fit' each possible model must be evaluated and the most suitable chosen.

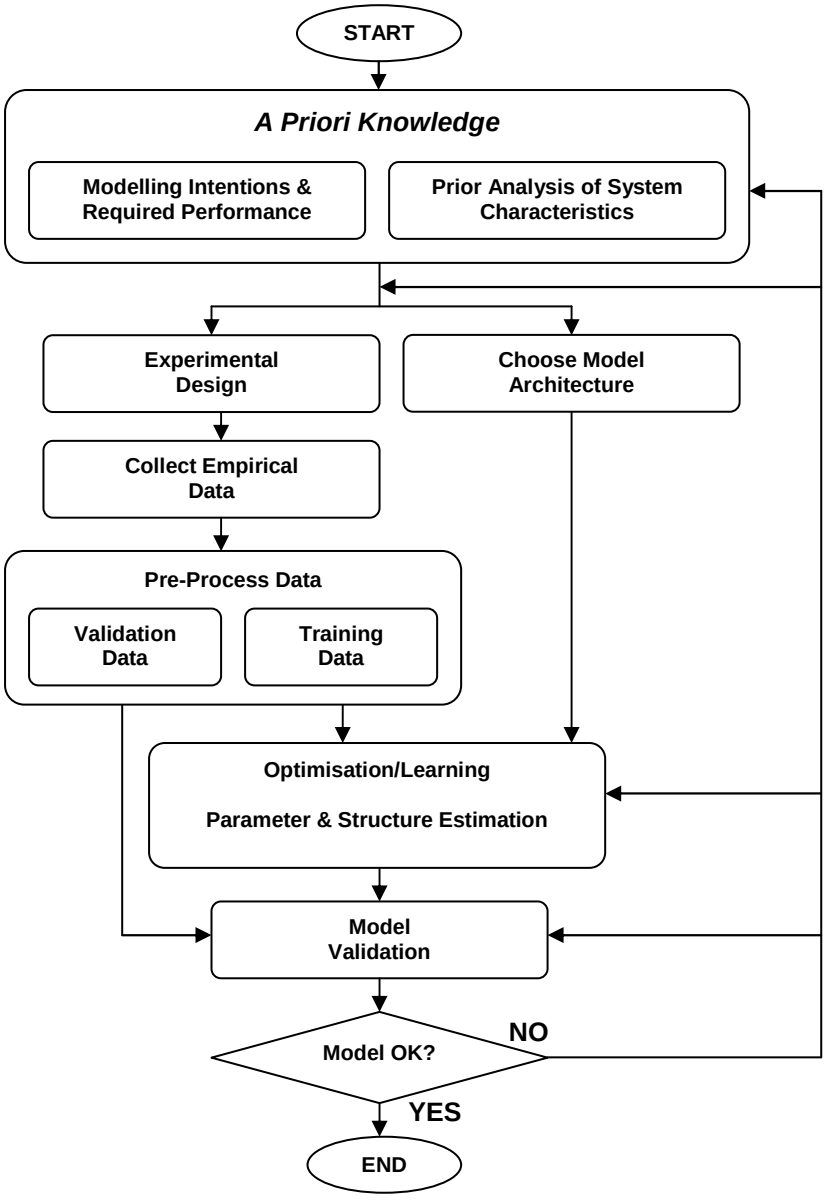


Figure (2.1) - System Identification Loop

The choice of model type or architecture together with the level of empirical data available will then influence the decision of how best the model may be fitted to the data through a process of optimisation. This is a learning process where optimum parameters

and structure are to be identified through the application of machine-learning procedures to the empirical data (e.g. Least-Squares regression). It is often the case that raw empirical data collected from an experimental set-up must be first pre-processed before it may be used successfully in any optimisation regime. This pre-processed data is then referred to as training data as it is this information with which we will train our model.

Finally, any model identified must be validated successfully before it may be employed through comparison with the observed system response. A separate set or subset of data, known as a validation or test set, is normally generated under similar but not necessarily identical operating conditions to that of the training data set. Statistical measurements of model error and likelihood can then be calculated from the comparison of the model output to the measured system response. The outcome of the validation procedure will then determine whether or not the identified model meets its required criteria. If the model is found to be lacking in some aspect, the modeller must return to the previous design conclusions and potentially modify any aspects of the adopted model structure, optimisation or experimental procedure that requires refinement.

Through each step and iteration of the System Identification procedure it is normal that the prior knowledge associated with the system under investigation is enhanced. This improved understanding of the system is most desirable and is often a fundamental objective of the modelling process. If the model is to have practical implementations such as for the basis of an automatic control design, it is useful to have as much knowledge over the system's behaviour as possible. In the following sections of this chapter we will look at different aspects of the system identification process in more detail. Good sources of information on the general topic of System Identification are the books by Ljung (1999), Söderström and Stoica (1989), and Juang (1994). The book by Unbehauen and Rao (1987) provides very useful information on the identification of continuous systems using more classical methods. The book by Nelles (2001) details the system identification field with specific emphasis on identifying nonlinear systems using modern techniques.

2.2) Role of Prior Knowledge

The level of prior knowledge available is of fundamental importance in determining how any modelling procedure should be approached and plays a vital role throughout the development of a successful model. For complex nonlinear systems where a physical analysis of the system is difficult and little or no physical insight or system knowledge is available, models must be learned solely through the use of experimental data. The application of learning systems can be seen to reduce the requirement for a detailed physical knowledge and the use of such methods has led to the following categorisation of models based upon the level prior knowledge available or employed:

White-Box Models

The model has been constructed entirely from prior knowledge and physical insight. Typically, models derived from first-principles such as a nonlinear differential equation model. White-box models often appear to have an advantage in overall interpretability, however for complex systems a resultant model can also be seen to be very complex.

Grey-Box Models

Some physical insight is available or used, with certain aspects such as model structure and parameters being directly estimated from experimental data using optimisation or learning techniques.

Black-Box Models

No physical insight is available or used; models are constructed solely from experimental data together with the application of learning systems and a chosen model structure. Black-box modelling can also be referred to as empirical modelling.

In practice, a white-box model derived from first principles, such as a nonlinear differential equation model, will rarely fully replicate a real system's behaviour due to environmental effects, and experimental data will often be used to tune parameters or coefficients of a model. Similarly it is rare that absolutely no knowledge of the system is available. Nevertheless, in tackling problems where prior knowledge is seen to be lacking, the use of learning systems can be seen to overcome this deficiency through the

continuous use of system information in the form of current and previous inputs and outputs.

In the selection of a white-box or black-box modelling strategy a further consideration is the time and cost associated with adopting various alternatives. The development of a white-box form of model of a complex physical process can be seen to be a particularly demanding task. As a result, an alternative approach based on the application of learning algorithms to empirical data may be easier and therefore more cost-effective to implement. Therefore the role of prior knowledge in most cases is to provide a basis for the design of an appropriate model structure, and consequently to dictate the level of empirical data required and subsequent experimental policy. In this regard, the use of prior knowledge allows the modeller to potentially reduce the learning task through extending a tangible influence upon what may otherwise become a interpreted as a prescribed machine learning or function approximation algorithm where data is submitted without any regard to validity. This prior or initial knowledge may take many forms and is discussed below.

2.2.1) Overall Modelling Objectives

The role of prior knowledge begins with an understanding of the problem to be solved. The ultimate purpose of the model will have a great influence on the kind of model that will be required. Is the goal of the model merely to provide a basis for simulation/prediction as an estimator, or is the modelling process to provide further physical insight and reflect the workings of a process? Is the model to be used as the basis for the design of a control system? What level of model complexity, interpretability, accuracy and robustness is desirable? Another important factor that should be borne in mind is that if the resultant model is to be used as the basis of an automatic controller, this controller may be able to compensate for less than perfect model accuracy. As a result, it is the robustness rather than merely the accuracy of the description that will dictate the quality of the solution.

2.2.2) Knowledge of System Characteristics

Any knowledge regarding the characteristics of the system can prove invaluable when deciding upon which modelling approach to adopt. Information regarding the type of nonlinearities involved, such as the dynamic order of the system, or the number of parameters or interactions between variables can allow the complexity of any potential model to be reduced and thereby limit the dimensionality of the problem. For example, in most fixed-wing aircraft the coupling between longitudinal and lateral directional variables can be neglected, therefore allowing these subsystems to be modelled independently. By contrast, in helicopters, such decoupling is generally not appropriate in flight mechanics models, except possibly under some particular flight conditions. Knowledge concerning any environmental influences such as disturbances and noise effects can also be incorporated into the model design. Another source of prior knowledge could be any existing models of the system that have been developed. Such models can act as a basis for the design of a new improved model or may even form an integral part of a new solution. A previous model may also be employed as a performance benchmark for any new development.

2.2.3) Knowledge of Empirical Data or Experimental Conditions

Prior knowledge of the system characteristics may also extend to information about the operating range of the system and the availability of empirical data. Many systems primarily operate within limited regions of the potential operating range of that system and this may be reflected in the overall constitution of the available data. Empirical data may be scarce in regions where it is difficult to perform data collection experiments due to limitations in the experimental set-up. Constraints on the physical operation of the process, such as operating conditions that can lead to damaging the system itself, can also lead to an uneven distribution of data across the full operating range of the system. Knowledge of such conditions can impact on subsequent design conclusions, as modelling approaches that are predominantly data-reliant may not be possible to pursue due to the sparsity of available data.

2.3) Experimental Design

The experimental design stage of the system identification process is of critical importance if the chosen modelling approach is to depend significantly on empirical data. As well as selection of which variables are be measured, the experimental design process must also consider what kind of excitation signals will be necessary in order to gather as much information about the underlying system as possible. More practical considerations will concern the actual experimental equipment set-up (e.g. sensors, calibration, etc.), and the sampling rate used to record the data. It is important to remember that whilst some design choices can be proposed and examined at length whilst working away from the actual system, the experimental design choices made can only be changed through conducting new experiments. Therefore, careful consideration and planning of the experiment in advance is necessary if potentially costly redesign and repetition are to be avoided. The previously mentioned textbooks on system identification by Ljung (1999) and Söderström and Stoica (1989) are good starting points for general information regarding experimental design, and further more dedicated sources of information are Godfrey (1993), Goodwin and Payne (1977), Goodwin (1987), and Mehra (1981).

Through the design of a suitable experimental procedure, a **Training** dataset can then be created to support the learning or optimisation process from the resultant empirical data. For the identification of complex systems where prior information is limited (black-box modelling) the accuracy of any model will be entirely dependant on the quality and scope of the training data supplied to the learning system. Consequently the design and construction of the training dataset where important data maybe extracted and represented appropriately is not a trivial task. The overall design objectives of any model and subsequently developed control system must be kept into consideration when the experimental methods are chosen. Experiments must be tailored to the system under investigation with the resultant data covering all the relevant areas of operating space. If data is not recorded over the full range of operation we wish to model and ultimately control, the resulting model accuracy in regions where data was not recorded is likely to be unsatisfactory. Important considerations in the design of an experiment include:

2.3.1) Which measurements?

Ideally, open-loop system response data should be used in the modelling process so that the behaviour of the system can be examined directly. However, in some cases this is not possible due to operational constraints and a closed-loop identification approach must be adopted. An example of this would be a system that proves to be unstable under open-loop operating conditions and through prolonged operation can compromise the safety of the operator or the system itself. In tackling closed-loop identification, a variety of methods have been proposed, with the most simple being 'Direct' methods where the closed-loop system is treated as if it where open-loop and the same overall system identification methods applied. However, a particular problem in closed-loop identification is that the process input u is typically correlated with the output noise m. The result of this is that some identification methods are not well suited to this direct approach, and a number of 'indirect' methods have been proposed where external signals measured between the controller and the plant are incorporated. For more information on closed-loop identification, the previously mentioned textbook by Ljung (1999) provides a good overall account of the problem, and a review of closed-loop identification issues has been completed by Van den Hof (1997). In this thesis we are focusing on systems where open-loop response data is available, and are not investigating closed-loop identification.

Another potential problem in the design of the experiment is the availability, placement, accuracy and flexibility of any measurement equipment. Measurement devices must not introduce further disturbances or noise to the system response that would otherwise not be present. Noise and prevalent disturbances in the system must also be investigated. If a system is susceptible to particular disturbances as part of its normal operation, this behaviour cannot readily be separated from the underlying system response and its influence should not be ignored or removed from the Training dataset.

There are also potential commercial considerations involved in conducting a number of experimental procedures. The investment of time and money into equipment and experienced personnel may be limited. An extreme example would be the prolonged flight-testing program necessary in the development of a new aircraft. Expensive prototypes fitted with measurement devices must be developed as well as the experienced

pilots and ground crew needed to support the operation. In this example further costs would be incurred by extensive wind-tunnel work and the design and support involved in ground based experimental-rig set-ups. A more straightforward constraint would be the potential downtime involved in halting a manufacturing or chemical processes in order to perform modelling analysis.

If the modelling procedure is to adopt an inclusive operating regime outlook this also introduces a number of important challenges. Regions of the operating range where the model must be most accurate will require sufficient data. These will be regions of operating space where the model will be required to operate for most of the time. Such operating regions may display off equilibrium or significant nonlinearities where large amounts of data may be necessary to represent such complexities or regions of critical importance. The relative importance of individual data samples must also be examined to allow the frequency of different situations, noise levels, or definitions between operating regions, to be incorporated into the model design. Under certain operating conditions the system may also likely be damaged or become dangerous, and the acquisition of data and successful modelling in these regions may prove difficult. Awareness of the limitations of the system must be employed in the experimental design.

2.3.2) Excitation Signals

The choice of excitation signals will ultimately determine the nature of the system response data included within the Training set and therefore plays a crucial role in how good a representation can be achieved. The selection of excitation signals is predominantly specific to the particular application and places a great deal of dependence upon the expertise of the engineer. As system noise and underlying disturbances are beyond the influence of the modeller, the input signal is the only avenue open for manipulation.

For nonlinear systems with complex dynamics it is first necessary to cover the whole operating range ($u_{\min} \rightarrow u_{\max}$) of the system by varying the amplitude of the input signal. Furthermore, the dynamics of any system may only reveal themselves under excitations at certain frequencies. Which frequency dynamics are excited and therefore likely to be

represented by our resultant model is determined by the spectrum of our input signal. Choices of excitation signal include:

Constant

Not readily suitable for identification as no dynamics are excited. Only one parameter, such as the static gain, may be identified.

Impulse

Not readily suitable for identification purposes. A possible indication of the overall transient response may be forthcoming, but gain may not be estimated with any great accuracy.

Step

Popular and well suited for identification purposes. Transient response can be fully appreciated and good estimations of the static gain and low frequency response can be obtained. Related to the step input is the 'Doublet' excitation signal consisting of an initial positive step quickly followed by a negative step.

Rectangular

In essence this can be scene to have the same qualities as that of the Step input but also introduces a frequency component allowing a particular frequency range to be emphasized. However this information as to the frequency response is limited to one frequency, as would any sinusoidal input of a particular wavelength. If a model is to replicate the behaviour of a system known to operate at particular frequencies, it is important that the design of any excitation signal take this into account (e.g. a mixture of sine waves or rectangles).

Filtered Gaussian White Noise

If little is known of the process's frequency dynamics or even the models intended use, a good choice for the excitation signal may be a Gaussian white noise signal put through a filter so that particular frequencies may be emphasized to tailor an overall signal spectrum, and also to curtail signal amplitude within predefined limits.

Pseudo-Random Binary Signal (PRBS)

Another popular general choice for linear systems is the use of the Pseudo-Random Binary signal that can be seen to be a periodic deterministic signal of constant amplitude that displays properties in keeping with the white-noise alternative. A range of frequencies may be emphasized accurately. However, for the identification of nonlinear systems this signal must be adapted to include changes in signal amplitude as otherwise we would have empirical output data restricted to the upper and lower limits dictated by the original input binary signal. For such reasons, amplitude modulated PRBS (APRBS) can be adopted to allow both the amplitude and frequency ranges of the input space to be investigated.

2.3.2.1) Active Learning

As with the other aspects of modelling process the selection of an appropriate array of excitation signals can be seen to be dependant on prior knowledge of the system and can involve a significant level of iterative design, perhaps resorting to heuristics or trial and error, in order to produce a suitable training dataset. In the absence of such prior knowledge, such as for complex systems being identified with black-box methods, a common strategy is to endeavour to make the distribution of training data as uniform as possible across the operating space or even to use all available training examples. However, a non-selective approach where data is submitted to the learning system without proper consideration can lead to problems regarding the conditioning of the training set. Repeated or redundant data examples may be included that unnecessarily increase the size of the training set, or examples concentrated within non-essential areas of operating space due to local complexities may be incorporated.

A modern approach to the design of excitation signals in the machine-learning field has been the concept of Active Learning. Instead of first collecting and pre-processing empirical data and then employing a suitable learning algorithm to identify a model, the two tasks are brought together by actively acquiring new information about the system (by searching for an optimal training set by exploring input space for the appropriate excitation signals) whilst the process is in operation. In essence, the learning system is to interact with the system directly in order to obtain and enhance the required training data.

A conventional Active Learning scheme would endeavour to optimise the amount of new information that can be garnered from each subsequent measurement. Therefore, in order to determine the most informative data measurement, a method to determine the model error associated with any previous measurements must be employed. By examining the model error associated with each sample we assign a level of curiosity to the Active Learning algorithm as the excitation signals will be targeted toward the goal of finding new information close to operating points at which the model is struggling. As can be readily appreciated, a potential problem encountered by the active learning approach is that although we are seeking to optimise the entire training set, the searching impetus of the algorithm can focus too closely and remain confined to one local region of operating space. Such an increase in potential local complexity can in turn require an increase in data with which to identify parameters. Another potential problem associated with this approach is that the searching algorithm can be slow and computationally expensive. This is especially true if an undirected random search for data is employed.

In processes where active learning has been implemented whilst remaining under normal day-to-day operations, the effort demonstrated by the searching algorithm (termed curiosity component) must be constrained so as to not fully disrupt the performance of the system whilst still seeking new information with which to improve the model. The research presented by (Cohn et al., 1990), (Cohn, 1994), (Thrun, 1992) and (Plutowski, 1994) explores the different details of Active Learning, the work presented by (Murray-Smith, 1994) explores the use of Active Learning with regard to the Local Model Network learning system, and (Cohn et al., 1997) shows how Active Learning may be incorporated with the mixture of Gaussians model framework.

2.4) Pre-processing Data – Creating the Training Data Set

Once the empirical data has been collected, a degree of pre-processing is normally required in order to a construct a suitable set of training data that may be used with the selected learning system. Through the pre-processing stage the learning task can be made easier and therefore can allow a greater level of model performance to be attained. Factors such as the size of the training data set, the distribution of data examples with

regard to the whole of the available operating space and the validity of individual data examples must be examined before a final Training set may be determined. The sampling rate of the data acquisition must be fast enough to allow system dynamics to be accurately reflected, but also not lead to excessive amounts of data and therefore increase level of data pre-processing required. Different Learning systems or optimisation algorithms may involve complex iterative mathematics which can place limits on the size or length of the Training set due to the computational effort required to find a solution.

Through pre-processing and subsequent analysis of the resultant model error it may become apparent that certain modifications to the excitation signals and sampling rate may be required. Furthermore, certain regions of data or individual samples may have to be omitted as they may cause poor conditioning within the training set. This may have implications with regard to the amplitude range explored by our excitation signals. A further consideration when designing a suitable training set is the minimum hold time (shortest period of time that the excitation signal remains constant) associated with the excitation signal and response data. If an excitation signal is of a given length, the minimum hold time will dictate the number of steps or transitions within the signal and therefore influence the frequency characteristics of the system. For a linear system the minimum hold time is normally selected to be equal to that of the sampling time. However for nonlinear system identification a design trade-off is introduced. Too small a minimum hold time can prevent the system reaching a settled or equilibrium state, this leads to the recorded output data being restricted with regard to the potentially available output amplitude range with the data examples being concentrated within the middle of the operating range. Too large a time will restrict the number of transitions within the signal and therefore potentially restrict the number of operating points that may be excited by the signal and can overemphasize the importance of low frequencies. A heuristic often employed would be to select the minimum hold time to be close to that of the dominant time constant, see Nelles (2001).

2.5) Choice of Model Architecture

Perhaps the most fundamental part of the system identification process is the selection of a suitable model architecture or structure with which to build a representation. A great range of alternative model structures have been proposed and successfully implemented over many years. This section is to provide an overall guide to the various types of models that are available, rather than a full account of the precise details of each alternative.

2.5.1) Linear and Nonlinear Models

The most fundamental distinction made between various types of model is whether it can be said to be either linear or nonlinear. Systems are often categorised as being either linear or nonlinear, however most real dynamic systems can be seen to display a level of non-linearity (e.g. noise). From a logical perspective it would seem that a nonlinear system would require a nonlinear model to fully exhibit its characteristics. However, it is common practice that a linear model will be the first choice structure with which to identify a model of a nonlinear system, and that this course of action often leads to satisfactory model fit for its purpose.

One of the primary reason for adopting a linear approach is the very well understood and widely adopted methodology of defining a linear structure and utilising a comparatively simple linear optimisation technique (such as linear least squares) with which to identify parameters from data. Nonlinear modelling approaches typically require significantly more effort due to an increase in complexity and optimisation. Linear modelling techniques are therefore still used successfully when considering nonlinear systems and can often be seen to form the building blocks of a nonlinear description.

Many system plants are also designed to behave as linearly as possible within certain operating ranges so that they may be operated more easily. Moreover, a well-designed feedback controller will also act to contain the effects of the nonlinearities in the system. It has also been shown that linear theory can be applied to model nonlinear systems operating at equilibrium points. The mathematician Lyapunov showed that the local

stability of a system in equilibrium where nonlinearities are smooth, and therefore differentiable, could be predicted through the application of linear theory. This has particular relevance to models identified from first principles in the form of ordinary differential equations. Such nonlinear descriptions may facilitate linear models to be identified through the linearisation of these equations at particular equilibrium operating points.

The process of designing a control system for a system plant is also made significantly easier if a Linear Time-Invariant (LTI) model can sufficiently represent the plant. A well-established and straightforward methodology for designing controllers for LTI systems has been in existence for many years, with most introductory control system design books covering the basic principles, see Dorf and Bishop (2004) and Nise (2003). This is not the case when dealing with systems that exhibit significant nonlinearities, where to describe the system with a single linear model would result in an inadequate representation of the system's behaviour. In cases where significant nonlinearities are present, a linear model will not accurately describe the real system behaviour away from the equilibrium region at which it was linearised.

No standard or generic response from a nonlinear system will exist as such systems can behave in very different ways. For example, nonlinear systems can display random or indeterministic behaviour (where the behaviour of a system cannot readily be predicted), periodic and aperiodic (e.g. chaotic oscillations) oscillations, and multi-stability (i.e. alternating between two or more exclusive states). Consequently, no generic all-purpose modelling methodology and control design procedure has been established. Therefore nonlinear modelling and control remains an extremely active area of research where various possible solutions have been proposed each with their particular strengths and weaknesses and associated level of complexity. A further important categorisation of models is whether or not they are to operate within a Time or Frequency domain. Obviously a Time-domain model will involve the analysis of a system or unknown function with respect to time, and a Frequency-domain will operate with respect to frequency. In this thesis only problems within the Time-domain (i.e. time-series data) are considered.

2.5.2) Parametric and Nonparametric Models

A classical distinction is often made between parametric and nonparametric models. A parametric model will consist of an assumed functional form constructed from a limited number of variable parameters. The function supposed by this approach will then be optimised through its parameters to fit any recorded empirical data as closely as possible. Parametric models are the more widely adopted approach, as due to the limited number of parameters a more interpretable model is often the result. In the presence of prior knowledge of the system characteristics, it is often possible for a parametric model to be constructed that directly reflects the relationships between particular physical quantities. A non-parametric modelling approach will not assume or impose a functional form on the function to be identified. Nonparametric methods are often seen to offer a more flexible approach to the identification task, as no prior structure is adopted an infinite number of parameters may be used to represent the process exactly. As a predefined structure is not to be imposed on the unknown function, a greater degree of freedom over the form of final model is possible. As a result, non-parametric methods are often seen as ideal tools for the identification of systems where a priori knowledge is limited such as in black-box modelling problems.

Although in theory a non-parametric approach may offer an infinite dimension parameter vector, in practice a limitation on the number of parameters will ultimately be encountered due to the restrictions imposed by complexity and computational constraints. Furthermore, as prior knowledge is either unavailable or not employed in the modelling process, non-parametric approaches are often said to be more dependant on the quality and quantity (or relative sparsity) of empirical data. Classical approaches to constructing non-parametric models include Transient Analysis, Frequency Analysis, Correlation Analysis and Spectral Analysis. Further information on these classic methods can be found in the aforementioned system identification textbooks by Ljung (1999), Söderström and Stoica (1989) and Unbehauen and Rao (1987).

The GP modelling approach investigated in this thesis may also be categorised as a nonparametric method as instead of specifying a parametric structure to form the basis of a description, a prior probability space over functions is to be specified instead. In order to specify this space over functions, a kernel-based nonparametric regression method is

utilised. Further methods of nonparametric regression include kernel smoothing estimators and Spline-smoothing techniques. Useful reviews of these smoothing methods include Hardle (1990), Eubank (1999) and Wahba (1990). These nonparametric methods are not often deployed toward the task of system identification, and instead are more typically utilised toward statistical problems. However, the GP model can be seen to have much in common with these alternative nonparametric regression methods and a full discussion of the similarities can be found in Rasmussen and Williams (2006). A detailed discussion of the GP modelling approach itself is to form the basis of the next chapter.

2.5.3) Linear Dynamic Models

A general framework for the description of different linear dynamic models can be found in most system identification textbooks, including Ljung (1999), Söderström and Stoica (1989), and Nelles (2001). A further useful resource is the survey paper by Leontartis & Billings (1985). The framework allows different linear dynamic models to be described through the combination of various transfer function elements. A general form of the problem can be seen to describe the unknown output through the function

$$y(t) = f(\varphi(t)) + e(t) \tag{2.1}$$

where *y(t)* is the output (measured data), *f(*)* the function we wish to model, $\varphi(t)$ a vector of adjustable parameters and *e(t)* representing noise present in the system. Note, that in describing the elements present in the following general model structure, the time t has been substituted for k, and we make use of the time shift operator q (equivalent to writing $(k-1)$).

A general linear model structure was introduced by Ljung (1999) and is formed from decomposing the influences on the model into deterministic and stochastic components. The deterministic model component operates on the principal that at some given input a corresponding output response will be generated, and that all actions are determined through preceding events to the exclusion of random elements. The stochastic model component can be seen to introduce randomness to the model structure and therefore allow unknown or external disturbances (noise) to be included. A solely deterministic

model structure can then be realised through employing a linear filter G(q) between the input) and output y(k). In the same manner, a linear filter H(q) can be introduced to filter white noise v(k) and therefore allow noise frequency components to be modelled. The term *input transfer function* is used to describe the filter G(q), and *noise transfer function* adopted to describe H(q). Both of these transfer functions can then be further decomposed into a numerator and denominator. A general model structure can then be constructed through the combination of the deterministic and stochastic components:

$$y(k) = G(q)u(k) + H(q)v(k) = \frac{\tilde{B}(q)}{\tilde{A}(q)}u(k) + \frac{\tilde{C}(q)}{\tilde{D}(q)}v(k) \tag{2.2}$$

A further decomposition is normally adopted in the general framework where common denominator dynamics are identified from G(q) and H(q), and given the signifier A(q). $\tilde{A}(q) = F(q)A(q)$ and $\tilde{D}(q) = D(q)A(q)$ with common denominator A(q). The general model structure can then be written as:

$$A(q)y(k) = \frac{\tilde{B}(q)}{\tilde{F}(q)}u(k) + \frac{\tilde{C}(q)}{\tilde{D}(q)}v(k) \tag{2.3}$$

With the transfer functions composed of polynomials as:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na} \\ B(q^{-1}) &= b_1 q^{-1} + \ldots + b_{nb} q^{-nb} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc} \\ &etc.. \end{aligned} \tag{2.4}$$

and a parameter vector can then be written as

$$\Theta = (a_1 \ldots a_{na} \quad b_1 \ldots b_{nb} \quad c1 \ldots c_{nc} \quad etc...) \tag{2.5}$$

From this general model structure a number of different linear dynamic models can then be defined through the combination or omission of certain elements of the general structure. The simplest model structure would be a model consisting of either solely deterministic or stochastic components. However, when examining real systems it is highly improbable that uncertainty in the form of noise would not extend some influence

upon the output response. Therefore, a wholly deterministic model (e.g. $y(k) = G(q)u(k)$) is not a common structure to employ within the system identification field.

With the help of any available prior knowledge regarding the nature of the system, the relevant inputs of any identified model can be outlined and incorporated into our chosen model structure. The inclusion of one or more input variables u(k), known as an **exogenous** input (X), allows the deterministic nature of a system to be incorporated in that an input will have a determinable influence on the system output. These input/output models can be further categorised by the way the noise component is incorporated into the structure. A distinction can be made between **Equation Error** models (ARX, ARMAX, ARARX) and **Output Error** models (OE, BJ, FIR) where for Equation Error models a common denominator polynomial 1/A(q) can be adopted to demonstrate shared dynamics between the input and output noise. From this general framework, the modelling process would involve the selection of the model structure most suitable to identify a particular system. For the sake of thesis brevity, I refer the reader to the aforementioned system identification textbooks (Ljung (1999), Nelles (2001), Söderström and Stoica (1989)) for a detailed account of where the various model types may be best employed. However, the general approach taken is to first utilise a simple model structure for the identification task before considering a more complex one.

2.5.3.1) Linear to Nonlinear Dynamic Models

As in this thesis we are expressly concerned with the identification of nonlinear systems, therefore it might seem tempting to dismiss the various linear dynamic models discussed previously. However, the nomenclature used for the description of linear dynamic models (e.g. ARX) crops up frequently in the literature devoted to nonlinear system identification. Furthermore, methods have been developed to extend these linear models for the purposes of nonlinear modelling. For the extension to nonlinear model structures, the ARX model is particularly important (due to it's linear in the parameters structure) and forms the basis of the Nonlinear ARX (NARX) model. The NARX model extends the ARX structure through the replacement of the linear relationship with some unknown nonlinear function $f(\cdot)$. Therefore, assuming that the model is to be implemented on a digital computer, the discrete-time nonlinear model can be stated as:

$$y(k) = f(u(k-1),\ldots u(k-m), y(k-1),\ldots, y(k-m)) \tag{2.6}$$

Where y(k) is the output, and u(k) the inputs, with *m* representing a time delay, and thus assuming a multiple input single output (MISO) form. Model structures that include more advanced noise components (e.g. ARMAX) are rarely applied for nonlinear system identification due to the resultant increase in complexity. Thus, simple dynamics representations that result in input-output mappings are more common for nonlinear models. For a detailed review of this extension of linear system identification methods toward nonlinear problems, see Leontartis and Billings (1985) and Nelles (2001). The NARX model structure can also be interpreted as a tapped-delay-line as depicted in Figure (2.2).

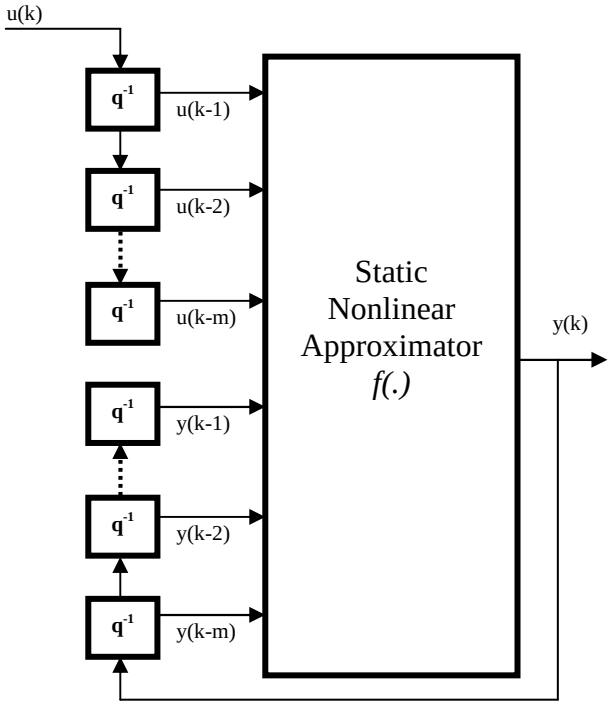


Figure (2.2):- NARX model as Tapped-Delay Line

Furthermore, this approach to nonlinear modelling is also known as the 'external dynamics' approach due to the separation of the model structure into a nonlinear static approximator and an external dynamic filter bank, see Nelles (2001) for more discussion. Due to this separation, any nonlinear model architecture can then be chosen for the approximator, such as a nonlinear polynomial or neural network architecture. A drawback of the external dynamics approach is that typically a large number of inputs are required by the approximator, leading to the requirement for the chosen approximator to

be able to manage high-dimensional mappings for complex systems that involve large numbers of delayed inputs and outputs. Furthermore, this can lead to matrix conditioning problems in some learning systems as delayed inputs/outputs will be highly correlated with their immediate successors if a high sampling rate is chosen. Consequently, the overall input space of the model may become impossible to cover completely with training data observations resulting in certain operating regions that are difficult to identify due to a lack of empirical data. This is something that may prove problematic for modelling approaches that rely upon the partitioning of the input space. However, this restriction of the input space is also something that can become advantageous as the overall learning task may be reduced.

Whilst the external dynamics approach to nonlinear system identification is the most widely used method, an alternative method termed '**internal dynamics'** is also possible. In contrast to the external approach where the dynamics are handled by a separate tapped-delay line and then employed as inputs to the nonlinear approximator, the internal approach does away with external feedback and uses internal memory and feedback instead. Thus, the dynamics are to be learned by the network itself. The internal dynamics approach is common within the field of neural networks where they known as recurrent networks with notable implementations being the Hopfield network discussed in Hopfield (1982), and the Boltzmann machine discussed in Hinton and Sejnowski (1986). The internal dynamics approach can be seen to be an attractive alternative as a reduction in the dimensionality of the input space (a problem with the external approach) may be realisable. However, they remain less popular due to the increase in the complexity of the network, and the lack of interpretability of the internal model states. Further alternatives for tackling nonlinear problems include the inclusion of derivative information into the model, and parameter scheduling approaches.

2.5.4) Nonlinear Dynamic Models

In the previous section the extension of linear to nonlinear dynamic models was discussed. However, in the implementation of the NARX method suitable nonlinear static model or approximator must also be selected. This nonlinear approximator is then to define a mapping between inputs and outputs.

Classic parametric methods of modelling nonlinearities include the identification of **polynomial** representations of the system characteristics. These methods can be seen to be an increase of model order over the linear model (a 1st degree polynomial) and are often employed for interpolation or curve-fitting problems of lower dimensions. A discussion of the Kolmogorov-Gabor (K-G) polynomial and Volterra-series modelling approaches that represent a nonlinear model with output feedback (as in NARX) can be found in Nelles (2001). However, this method is only suitable for low-dimensional problems as the number of regressors present in these models grows very quickly with the chosen degree of the polynomial. Furthermore, high-degree polynomial approaches have a tendency toward oscillatory interpolation behaviour and unreliable extrapolation behaviour. Other classic methods of nonlinear models include the **Hammerstein** and **Wiener** approaches. These methods are widely adopted in industry and rely upon an assumption that a separation exists between the dynamics and the nonlinearity of the system. The Hammerstein model structure implements a static nonlinear model (typically a polynomial but any model is possible) followed in series with a dynamic linear model, and the Wiener model structure is in the reverse order. The implementation of both these methods relies on prior knowledge of the system that facilitates the inherent structural assumptions. As a result, they cannot be regarded as general purpose modelling approaches for black-box problems. For more information on Hammerstein and Wiener models see Ljung (1999).

At this point it is worth pointing out a particular obstacle that is inherent to all modelling approaches, the '**curse of dimensionality'**. This phrase is often used in describing the effect of including more input dimensions (and parameters) into the chosen model architecture. As the number of variables or parameters necessary to represent a particular function increases, so will the likelihood of oscillatory interpolation (an increase in variance error). Furthermore, the computational demand of optimising the parameters of the chosen model structure will also increase. Therefore, model structures that quickly grow in complexity as the number of included variables increases are not suitable for high-dimension problems, or for black-box problems where little is known about the underlying modelling problem. Model structures that can be seen to suffer particularly from this 'curse' include polynomial models, and grid or lattice based approaches that seek to partition the operating space in a uniform manner such as 'look-up' tables.

Overcoming or bypassing this curse of dimensionality is one of the most important driving forces behind some of the modern approaches to nonlinear system identification (e.g. using non-uniform partitions based on prior knowledge, mapping inputs onto different function spaces). Furthermore, in the identification of real systems, the curse of dimensionality may not manifest itself too detrimentally due to the peculiarities of the system under investigation. For example, the model may be able to be simplified due to the presence of inputs that are correlated or redundant, smooth regions of operating space may allow simple models that require less data to represent, and the presence unreachable regions of operating space (e.g. due to correlated data or operational constraints) may reduce the overall operating space to be identified.

In the identification of black-box models where prior knowledge of the system is limited, the model must therefore be identified from empirical data. This is also sometimes referred to as 'empirical modelling', and a fundamental aspect of this approach is that the chosen model architecture must facilitate this learning process. Furthermore, without prior knowledge of the system characteristics, the chosen model structure must be flexible enough to allow a wide range of nonlinear behaviours to be approximated. A general framework that a large number of different model architectures can be seen to follow is the formulation of a network of **basis functions**:

$$y = \sum_{i=1}^{M} \theta_i^l \phi(\theta_i^{nl}) \tag{2.7}$$

In this formulation, the mapping $f(\cdot)$ is to be modelled as a weighted sum of M basis functions, where θ^l are the weighting linear parameters, and θ^{nl} are the nonlinear parameters of the basis functions $\phi(\cdot)$. Therefore, for nonlinear models the basis functions must be nonlinear, and it is also worth noting that linear and polynomial models can also be interpreted under this basis function formulation. Furthermore, basis functions can be described as being either global or local. Global basis functions can be seen to contribute to overall model output across the operating range, whereas local basis functions only contribute to the model output in small 'local' regions of the operating range.

2.5.5) Neural Networks

The artificial neural network (ANN) is a black-box model architecture that consists of a large number of interconnected neurons (simple nonlinear processing units) that act as a parallel information processor. The origins of the artificial neural network approach come from research into the operation of human brains where the concept of neurons as structural elements of the brain was proposed. If the human brain is viewed as a computational device (performing tasks such as movement, pattern recognition and perception etc.) it becomes clear that even the most complex man-made machines are vastly less capable. Therefore, researchers working in the fields of machine learning and artificial intelligence have sought to emulate the vast processing power of the brain. Although the history of artificial neural networks can be seen to stretch back to the work of McCulloch and Pitts (1943), a great catalyst for the modern surge in interest in neural networks was the work of Rumelhart et al. (1986) that popularised the backpropagation algorithm used for training Multilayer Perceptron (MLP) networks. A good general resource that details many different types of neural networks is the book by Haykin (1994), where a full and interesting account of the historical advancements made in neural networks is also provided. Other good neural network textbooks include Bishop (1995) and Ripley (1996), however much of the discussion is focused towards pattern recognition or classification tasks. Good resources on neural networks from an engineering or system identification perspective are the books by Brown and Harris (1994) and Nelles (2001), and the paper by Sjöberg et al. (1995).

For the purposes of nonlinear system identification, the most widely adopted neural network structures are **feed-forward** networks. In this arrangement the information is to travel in one direction, from the network inputs to the network outputs. This flow of information is depicted in Figure (2.3), and the neural network structure can be readily interpreted as a static nonlinear mapping between the input and outputs, and thus compatible with the previously discussed ‘external dynamics’ NARX dynamical modelling approach.

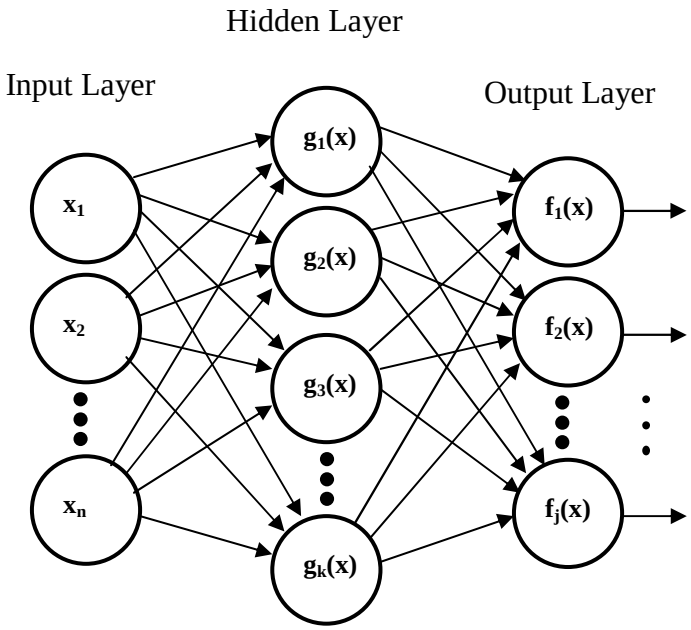


Figure (2.3) – Feed-forward Neural Network

Alternative network structures where delayed information is passed between neurons are known as **recurrent** networks. As discussed in the previous section, recurrent networks have the potential to reduce the impact of the curse of dimensionality, but this increase in complexity makes the learning task more difficult. Furthermore, the presence of feedback in the network brings the possibility for instability, as discussed in Braham (1998). The neural network architecture can also be understood as a network of basis functions where each basis function (or hidden layer neuron) is of the same type. The two most widely adopted feed-forward neural network architectures in the field of system identification are the Multilayer Perceptron (MLP) network and the Radial Basis Function (RBF) network. In this section a brief description of the properties of these two networks is provided.

2.5.5.1) Multilayer Perceptron (MLP) Network

The Multilayer Perceptron network is the most widely known neural network architecture and has become synonymous with what is generally understood to be a neural network. This feed-forward network utilises a ridge construction mechanism in order to project the input vector **u** onto a nonlinear parameter vector **x**:

$$\mathbf{x} = \boldsymbol{\theta}^{nl}\mathbf{u} = \theta_0^{nl}u_o + \theta_1^{nl}u_1 + \ldots + \theta_p^{nl}u_p \tag{2.8}$$

The nonlinear parameters are also known as the 'weights' of the hidden layer of the neural network $\theta_i^{nl} = [w_{i0} \quad w_{i2} \quad \cdots \quad w_{ip}]^T$. An activation function exhibiting saturation behaviour (typically a sigmoid function, e.g. $g(\mathbf{x}) = \tanh(\mathbf{x})$) is then applied to the parameter vector **x**. This combination of the construction mechanism and the activation function may then be termed as a '**perceptron**'. This individual hidden layer neuron can then be combined in parallel with other hidden layer neurons of the network through connecting the outputs of each neuron to an 'output layer neuron'. This output layer neuron is most commonly a linear combination of the hidden layer outputs that are each weighted by a set of parameters known as the output layer weights w_i. The overall network structure can then be written using the basis function formulation (where M is the number of hidden layer neurons and p is the number of inputs) as:

$$\hat{y} = \sum_{i=0}^{M} w_i \Phi_i \left(\sum_{j=0}^{p} w_{ij} u_j \right) \tag{2.9}$$

This combination of a single hidden layer of neurons and a linear output neuron is the most simple and common implementation of the MLP network. Further complexity can be achieved by incorporating additional hidden layers, or employing a nonlinear output neuron. Overall, we can control the number of parameters in the model by modifying the number of hidden layer neurons included. However, by increasing the complexity of the model structure, the optimisation procedure may become more demanding. An alternative to including more hidden layers is to include more neurons in a single hidden layer. As discussed in Nelles (2001), a general preference for one strategy over the other is difficult to substantiate and is dependent on the problem at hand.

As the MLP network contains nonlinear parameters in the hidden layers (and potentially the output layer), a nonlinear optimisation procedure will be required if these hidden layer weights are to be made optimal. The general problem of model optimisation and various nonlinear optimisation strategies are discussed in more depth in Section (2.6) of this chapter. However, it is worth stating here that the task of optimising the parameters and structure of any neural network can become a significant challenge. The need for

nonlinear optimisation techniques to be employed can result in very long training times. However, some of the resultant computational burden may be reduced if the output layer weights of the MLP network are chosen to be linear, therefore allowing more efficient linear optimisation algorithms to be implemented.

Furthermore, in addition to the optimisation of the linear and nonlinear model parameters, the overall model structure (i.e. number of neurons, number of hidden layers etc.) must also be optimised. Traditionally, the network structure is fixed in advance and then the model parameters optimised and the performance validated (i.e. we choose the number of neurons/layers 'a priori'). Such an approach can obviously become frustrating as if the model is found to be lacking in some way. As a result of this, methods that seek to regulate the complexity of the network structure by 'growing' or 'pruning' the number of hidden layer neurons have been developed. The difficulty of selecting an appropriate model structure for the MLP network is also compounded by the fact that the component parts of the model structure are not readily interpretable. In particular, the individual hidden layer neurons of the MLP network cannot be interpreted as active in only certain local regions of operating space (i.e. by changing or eliminating one hidden layer neuron, the whole network is affected and model output at all input regions will possibly change).

So far in this section we have focused on the difficulties of training a MLP network without detailing the major advantages of the model architecture. The main advantage of the MLP network architecture is that it can be seen to be a '**universal function approximator'**, which means that the MLP can approximate any smooth function to an arbitrary degree of accuracy as the number of hidden layer neurons is increased. Furthermore, this facility holds true for MLP networks composed of only one hidden layer for certain classes of activation functions (i.e. sigmoidal) as proven in Cybenko (1989) and Hornik et al. (1989). As a result, the MLP network can be seen to be a good general purpose modelling approach that exhibits a very high flexibility that allows a great range of different function shapes to be represented, and thereby may be applied to any function approximation problem. It is however worth noting that this universal approximation feature is not exclusive to the MLP network as other modelling approaches (such as polynomials) and types of neural network also demonstrate this property As the ridge construction mechanism acts to project the input space onto a lower dimensional hidden layer space, this allows the effects of the 'curse of dimensionality' to

be reduced, making the MLP network suitable for higher dimensional problems. A further property of the MLP network that makes it an attractive method for system identification and control is the fact that the number of neurons present in an MLP network is typically smaller than other neural network approaches (e.g. RBF networks). The result of this is that the evaluation speed associated with making predictions is typically lower than the alternatives. In summary, the MLP network is a very powerful and flexible method of function approximation. However, this flexibility comes at a cost due to the need for potentially time consuming and computationally expensive nonlinear optimisation methods to be employed. Furthermore, the more subtle nature of model structure optimisation and general lack of interpretability regarding individual neurons/weights makes the identification process difficult. As a result, the training of MLP networks can descend into a less than rigorous process of applying various heuristics or even trial and error.

2.5.5.2) Radial Basis Function (RBF) Network

Radial Basis Functions were originally developed as a method of multivariate interpolation (as discussed in Powell (1985)) in isolation to the development of MLP or other neural networks. The integration of the RBF methodology into the wider field of neural networks took place after the surge in interest in MLP networks, with notable papers being Broomhead and Lowe (1988), Moody and Darken (1989), and Poggio and Girosi (1990). Further papers that investigate the use of the RBF network for modelling purposes are Barnes et al. (1991), Murray-Smith (1992), and Pantaleón-Prieto at al. (1993). The RBF network is a feed-forward architecture where a radial construction mechanism is first used to calculate the scalar distance x between the input vector **u** and a centre vector $\mathbf{c} = [c_1 \quad c_2 \quad \cdots \quad c_p]^T$, with respect to a norm matrix Σ_i used to scale and rotate the input axes.

$$\mathbf{x} = \left\| \mathbf{u} - \mathbf{c}_i \right\|_{\Sigma_i} = \sqrt{\left(\mathbf{u} - \mathbf{c}_i\right)^T \Sigma_i \left(\mathbf{u} - \mathbf{c}_i\right)} \tag{2.10}$$

As in the MLP network, an activation function is then applied to this new parameter x. This activation function is normally selected to exhibit some kind of local character around a maximum at $x = 0$, with the most popular choice being the Gaussian function

$g(x) = \exp(-\frac{1}{2}x^2)$.In the same manner as the MLP network, this individual hidden layer neuron can then be combined in parallel with other hidden layer neurons of the network through connecting the outputs of each neuron to an output neuron. This output layer neuron is again most commonly a linear combination of the hidden layer outputs that are each weighted by a set of parameters known as the output layer weights w_i. The overall network structure (composed of *M* hidden layer neurons) can then be written using the basis function formulation as:

$$\hat{y} = \sum_{i=0}^{M} w_i \Phi_i \left(\left\| \mathbf{u} - c_i \right\|_{\Sigma_i} \right) \tag{2.11}$$

The RBF network can be seen to consist of three different components or types of parameter: output layer weights (which are linear parameters that determine the height of the basis functions and the offset value), Centres (which are nonlinear parameters of the hidden layer neurons that determine the position of the basis functions), and the Norm matrix (which are nonlinear parameters of the hidden layer neurons in the form of standard deviations that determine the widths and rotations of the basis functions).

The RBF network has also been proven to be a 'universal function approximator', see Park and Sandberg (1991) for details, but unlike the MLP network the prospect of combining multiple hidden layers in the RBF network is not thought to be particularly useful. As a result the RBF network is normally only employed with one hidden layer. One of the attractions of the RBF network over the MLP network is that through the radial construction mechanism and local activation function, the hidden layer neurons of the RBF network can be more readily interpretable. As the basis functions are local, the effect of changing the parameters of one neuron has only a small effect for input values that are far away from the designated centre of the neuron. Therefore, each neuron is predominantly active in a specific region of operating space, and the network as a whole can be interpreted more as a combination of local sub-models or multiple model. As a result, employing multiple layers of RBF neurons is likely to diminish this interpretable aspect as the outputs of the first hidden layer already span the input space, leaving subsequent layers to span the space of some less interpretable intermediary input space.

The RBF network can be interpreted as a two-layer network that is linear in the parameters if the nonlinearities and RBF centres are first fixed in the hidden layer. As the output layer weights of the RBF network are linear, the optimisation of these parameters can be performed through the use of efficient linear (least squares) regression. However, the nonlinear parameters that determine the position and character of the basis function must also be optimised. It is therefore common to first determine the hidden layer parameters of the RBF network (i.e. place the basis functions and determine the standard deviations) before optimising the linear output layer weights (thus determining the heights of the basis functions). The task of optimising the hidden layer parameters of the RBF network is also termed '**centre placement**' and it is normal to attempt to exploit the more interpretable 'local' nature of the hidden layer parameters so that the use of demanding nonlinear optimisation algorithms can be minimised. A variety of different approaches to the problem of training the hidden layer parameters of the RBF network have been proposed, see Nelles (2001) for a good review. These include simple approaches such as Random and Grid Based Centre Placement, where the basis functions are centred at random or in a uniform manner. More sophisticated Clustering Methods have also been developed where unsupervised methods (e.g. K-means algorithm, Kohonen's Self-Organizing Map) can be used to determine basis function centres that reflect the nature of the training data distribution,(i.e. many RBFs can be placed in regions of dense data, and few RBFS can be placed in regions of sparse data.

A further alternative to the problem of training the hidden layer of the RBF network are constructive methods such as the Orthogonal Least Squares (OLS) method proposed in Chen et al. (1991). This forward regression method can be understood as a form of Subset Selection, where a subset of suitable centres (regressors) is selected from a large set of candidate or potential basis functions. Unlike other methods, this subset selection uses supervised learning as the OLS algorithm only selects basis functions on the basis that they are effective at reducing the model error. Therefore, a key advantage of the OLS approach is that the RBF network is trained incrementally. The main disadvantage of adopting this incremental or constructive approach is the increased computational demand that may result in long training times. Furthermore, the OLS algorithm is still heuristic in nature, and is unlikely to outperform a RBF network trained with even more computationally expensive nonlinear optimisation methods, see Wettschereck and Dietterich (1992) for information on the application of nonlinear optimisation to RBF

training. Nevertheless, the OLS method has become a popular method for training RBF Networks and is the standard method used in the MATLAB Neural Network Toolbox, see Demuth and Beale (1998).

2.5.5.2.1) Normalised RBF Networks

A problem associated with the application of clustering techniques (and RBF networks in general) is the potential existence of '**dips**' in the interpolation behaviour. Such behaviour is normally the result of regions of operating space where either no basis function is present or the standard deviations of the neighbouring basis functions is too small. As clustering attempts to place the RBFs in accordance with the distribution of the training data, it is not unreasonable to expect that some sparser (but potentially important) regions of operating space are not sufficiently covered by the defined basis functions. Therefore, when the model is asked to predict within such a region, the model output is likely to be highly inaccurate. For high-dimensional input spaces the problem can become almost unavoidable and these 'dips' can lead to unexpected and undesirable behaviour in the output. A further potentially undesirable property of RBF networks is that the extrapolation behaviour tends to zero due to the local activation functions. However, through the **normalisation** of the RBF network these drawbacks can be overcome. The normalisation process results in the sum of all the basis functions being equal to 1, and this property is known as a **partition of unity**. Therefore, the partition of unity ensures that an equal weighting is given to every point in the input space, so that any variation in the output of the network is due to the weighting parameters of the basis functions (i.e. no unexpected 'dips'). As a result, the Normalised RBF network is less sensitive to poorly chosen basis functions, and an overall output level can be fixed without any explicit offset value (unlike MLP and RBF networks that normally employ a separate offset or bias weight (w_0, ϕ_0)). Further advantages of the Normalised RBF network are outlined in Werntges (1993).

However, the normalisation of the RBF network can present some less than desirable side-effects as the normalisation introduces interactions between the basis functions. A detailed discussion of these side-effects can be found in Shorten and Murray-Smith (1994). Most fundamentally, the basis functions may lose their uniform shape resulting in the maximums of basis functions being shifted from their centres, and the monotonic

decrease in the basis function as the distance from the centre may also be affected. Furthermore, the basis functions may reactivate in different regions of operating space. Overall, these side-effects are not enough to fully diminish the advantages of normalisation, but the interaction between basis functions is undesirable as it may result in basis functions that are multi-modal and non-local. These aspects are perhaps the defining qualities of the RBF network, so without care the normalisation of the RBF network can potentially diminish the local interpretability of the approach.

2.5.6) Multiple Model Networks

In the previous section the advantages of the RBF network were discussed. In particular, the locally active basis functions can be seen to offer an advantage in terms of interpretability and ease of training over the MLP neural network. However, whilst the RBF network may be more interpretable than the MLP alternative, the model still does not offer much insight into the underlying system or make it particularly straightforward to incorporate prior knowledge of the system into the identification process. Overall, the RBF network can be interpreted as a large number of locally accurate piece-wise constant (zero-order) models that are placed across the operating space. As a result, these simple local constant models are not going to offer much physical insight into the underlying system. Nevertheless, the RBF network and the Basis function formulation in general, can be seen to offer a methodology that allows a network of multiple local models to be defined.

The concept of developing a multiple model network can be seen to be an attractive prospect as the identification of complex systems may become more manageable if the overall problem can be reduced into a number of smaller problems. Such an approach is typically known as '**divide and conquer**', and in the field of system identification this can understood as dividing or partitioning the operating space into a number of local regions or 'regimes', and then identifying a local model that is accurate for each region. A global model may then be constructed through the combination of these local models, as depicted in Figure (2.4) on the next page.

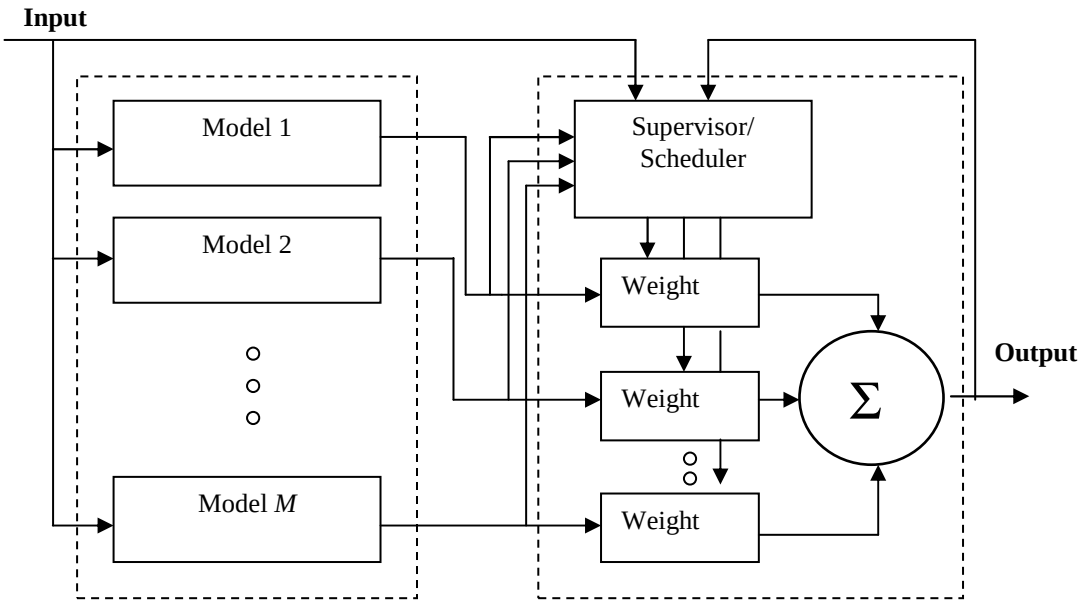


Figure (2.4): General Multiple Model Structure (from Murray-Smith and Johansen (1997))

A number of different multiple modelling approaches have been proposed to help solve the problem of nonlinear system identification. One notable approach is the application of **fuzzy logic** (see Zadeh (1965)) to the problem of partitioning the operating range where a number of rules must be defined. Through such an approach, any prior knowledge of the system (especially qualitative knowledge) can be incorporated directly into the model through the definition of membership functions. An important development in the application of fuzzy rules for system identification problems was the framework introduced by Takagi and Sugeno (1985), known as the TS model. Unlike linguistic or singleton fuzzy models, the outputs of TS model are functions (normally linear models) of the system inputs. Therefore, local models based on expert qualitative knowledge can be defined.

However, it is unlikely that qualitative knowledge will provide enough information for a successful model to be identified, and the inclusion of empirical data and learning methods is typically required. Such combined methods are often referred to as Neuro-Fuzzy models. In comparison to the neural network approach, Fuzzy networks can be seen to have an advantage in interpretability, however the development if good models may require the meticulous modification of the logical rules that are to define the

membership functions. Further explorations of this approach can be found in Jang and Sun (1995), Pfeiffer and Isermann (1994), and Babuška and Verbruggen (2003).

An alternative approach to the problem of defining the multiple model networks is the '**Operating Regime'** based methods developed in Johansen and Foss (1992, 1993, 1995a, and 1997). In this work the **Local Model Network** (LMN) architecture was proposed and further examined in Murray-Smith (1994), Murray-Smith and Gollee (1994), and Murray-Smith and Hunt (1995). There are close links and equivalences between LMNs, RBF Networks, Takagi-Sugeno fuzzy models and other approaches, and a good overall review can be found in Murray-Smith and Johansen (1997). A further related local linear approach to system identification is the Local Linear Model Tree (LOLIMOT) developed in Nelles et al. (2000) and expanded on in Nelles (2001). In this section we are to briefly focus on the Local Model Network approach as it can be further linked to the Gaussian Process modelling approach that is to be investigated in this thesis, as discussed in Gregorčič and Lightbody (2008).

2.5.6.1) Local Model Networks

The Local Model Network can be interpreted as an extension or generalisation of the Normalised RBF network where instead of the simple weights (constant or zero-order models) used in the output layer, more complex local models are to be employed. In theory, these local models can be of any type, but local linear models are normally employed for ease of implementation and interpretation. An advantage of using more complex local models is that in comparison to the zero-order weights of the RBF network, each local model can cover a larger portion of the operating space. Therefore, an LMN network of equivalent accuracy can normally be defined using a smaller number of basis functions (or validity functions) than the RBF network, thus improving computational efficiency and interpretability. Furthermore, engineers are well used to using linear models, and engineering systems are often designed and operated near to equilibrium operating conditions. Therefore, the collection of sufficient empirical data is likely to be achievable therefore allowing efficient linear regression methods to be applied. The local model network can be described by:

$$\hat{y} = \sum_{i=1}^{M} \Phi_i(d(x; c_i \sigma_i)) f(x_i; w_i) \tag{2.12}$$

Where $\hat{y}$ is the output prediction, Φ is the validity function (equivalent to the basis function of the RBF network, and similar to a membership function of a fuzzy network) constructed from an adjustable distance function d, and f is a function consisting of the inputs to the local model and a weighting function w. As in the case of RBF networks, the LMN can be normalised so that a partition of unity can be guaranteed (along with the potential for undesirable side-effects). The contribution from each local model is therefore defined by the activation of the corresponding validity function. Furthermore, the validity function is also sometimes stated as being a function of a scheduling vector, i.e. $\Phi_i(\phi(t))$, rather than expressly stated using a distance function akin to that of the RBF network. The scheduling vector $\phi(t)$ must be chosen carefully as it is to represent the nonlinear properties of the underlying system, and therefore help to define the operating point of the system so that the correct local model can be used at any one time. The scheduling vector is typically chosen from a part of the entire data vector, and in making this selection the use of prior knowledge can prove to be invaluable (i.e. certain current or delayed inputs or outputs of the system should provide a good indication of the current operating point of the system). A useful discussion regarding the choice of scheduling vector can be found in Gollee (1994).

The training of the individual components of the Local Model Network can be treated in a similar manner to that of other basis function approaches. Firstly, the parameters of the local linear models can be optimised through linear least squares. However, these parameters can either be learned globally or locally. In global learning, the process of optimising the parameters is to be performed simultaneously for all local models, and can be interpreted as similar to the optimisation of the output layer weights of the RBF network. In local learning, the parameters of each local model are to be optimised independently. Overall, a trade-off can be seen to exist between achieving an accurate global model and retaining the local nature of the models. The parameters of one local model estimated using the global learning approach are not independent of neighbouring local models. Therefore, the local models are not accurate linearisations of the system at the centre of the validity functions. As a result, these local models cannot truly be

interpreted as locally accurate models. Furthermore, the global learning method is computationally expensive in comparison to the local learning method and is less robust to over-parameterised or poorly structured local model networks. Therefore, local learning methods will tend to perform better when there is insufficient or noisy training data. However, global learning methods cannot be fully discounted, as they will tend to result in more accurate global models when the model structure is well chosen, the training dataset is well populated and the underlying nonlinearities are smooth. Further information on the relative merits of local and global learning can be found in Murray-Smith (1994), Murray-Smith and Johansen (1995), Cleveland et al. (1996) and Nelles (2001). The problem of structure optimisation, where the validity (basis) functions are to be defined, can be tackled in a similar manner to that of identifying the centres and standard deviations of the basis functions of an RBF network, e.g. uniform grid-based methods and clustering techniques such as the k-means algorithm. Furthermore, constructive forward regression methods such as those defined in Murray-Smith (1994) and Nelles et al. (2000) can also be used to determine the structural optimisation of the Local Model Network. In addition, backward regression or 'pruning' methods (see Reed (1999) and Jutton and Fambon (1995)) can also be employed to reduce model complexity by identifying and removing useless parameters, or similar and therefore redundant local models. As with the RBF network, popular activation functions include the Gaussian function and B-splines.

Once the validity functions and parameters of the local model network have been identified, the next problem to be tackled is how a global model is to be constructed from these local models. This problem can be understood as interpolating between the local models or 'blending' them together. Two different methods that have been proposed to tackle this problem are '**blending the outputs**' and '**blending the parameters**'. The first 'blending the outputs' method implements a simple weighted sum of the local model outputs in a similar manner to the linear combination of hidden layer outputs of the RBF network. The 'blending the parameters' method is an attractive alternative for cases where the local models can be seen to share the same structure across the operating space. In such an implementation, as the structure of the global model remain consistent across the operating range; it is the parameters of the global model that will change in accordance with the scheduling vector.

2.5.6.1.1) Off-Equilibrium Dynamics

The Local Model Network architecture offers a particularly good approach to modelling of nonlinear dynamic systems where both prior system knowledge and empirical data is available. In particular, the use of local linear models can be seen to be especially appropriate for systems where prolonged periods of operation (therefore providing an abundance of data) are to occur near to equilibrium or steady-state operating points. As many engineering systems are designed to operate primarily at certain stable operating points for ease of operation, this means that a great proportion of the available empirical data has a tendency to be centred around such operating points. However, in the investigations of Shorten et al (1999) and Murray-Smith et al. (1999), this reliance on local linear models has been found to compromise the validity of the LMN architecture when the off-equilibrium dynamics of the underlying system are considered.

As the validity of each local model is restricted to representing the system close to a defined operating point, in transient regions between such operating points the LMN model will not usually provide an accurate representation of the underlying system. Therefore, each local model will only provide an insight into the full model behaviour in a very small region of operating space. This problem does not tend to explicitly manifest itself when the operating point and therefore the scheduling vector change slowly. However, for faster or more violent transients (e.g. quickly driving the input across the full operating range) between operating regimes, the operating point of the system can be model can be forced far away from the equilibrium regions where the local models were identified. This may result in unexpected and undesirable transient effects in the output that may compromise the stability of the model and therefore prove problematic from a control perspective. In tackling this problem of off-equilibrium dynamics, local models may be placed in the off-equilibrium regions. Indeed, such a strategy can be seen to be in keeping with the overall proposal for a multiple model approach. However, as discussed in Shorten et al. (1999), it is possible for non-unique parameterisations of the model behaviour to exist (i.e. any identified off-equilibrium model may only partially represent the off-equilibrium region). Furthermore, the model structure of any identified off-equilibrium models may end up being significantly different from that of the existing equilibrium local models, thus impacting on the overall interpretability and transparency of the global model.

The problem of retaining transparency in off-equilibrium local models has been directly tackled through the use of **velocity-based descriptions** (see Leith and Leithead (1999)) where an analytical framework for relating global dynamic behaviour to local models was proposed. The original proposal for the application of the velocity-based framework focused on the linearisation of known nonlinear systems (Leith and Leithead (1999)), but further investigations in McLoone et al. (2001) have demonstrated the construction of a velocity-based Local Model Network using empirical data. However, a problem demonstrated in McLoone (2000) is that whilst the velocity-based description may provide a more accurate representation of the nonlinear dynamics of the system, the steady-state response of the underlying system was less accurately modelled. Furthermore, the velocity-based framework requires that the derivative of the input be available. This is something that may prove to be problematic to obtain due to noise and discontinuities in the input signal (i.e. sharp transients in the input signal will have near-infinite gradient and therefore reduced differentiability). However, alternative implementations of the model may allow differentiation of the input to be avoided.

The problem of identifying off-equilibrium models is also compounded by more practical operational constraints that often lead to a general lack of available off-equilibrium empirical data. In the identification of real systems it is often not possible or even unwise to excite the system in such a way to initiate an off-equilibrium response due to the potential damage to the system or even the operator. Therefore, if an off-equilibrium identification strategy is to be implemented, the experimental design and data collection process must be considered carefully. Due to the difficulty of identifying off-equilibrium models (i.e. lack of interpretability and lack of empirical data), alternative methods of identification have been proposed. One such method is the GP modelling approach discussed in this thesis. The GP model is non-parametric modelling approach where the model is identified almost exclusively from empirical data (overcoming the lack of interpretability problem). Furthermore, a number of specific properties of the GP model make it a good candidate for identification of models where empirical data is sparse. The use of GP models in tackling off-equilibrium identification problems was proposed in Murray-Smith et al (1999), Leith et al (2000), and Leithead et al (2000), and further discussion of this aspect is provided in Section (5.1).

2.6) Model Optimisation

A crucial part of the system identification process is the optimisation or learning task that must be undertaken in order to fit the chosen model structure to the available empirical data. Within the system identification community, model optimisation has also traditionally been referred to as ‘parameter estimation’ due to the popularity of parametric models over nonparametric alternatives. Furthermore, as applications have become more demanding, leading to the requirement for more sophisticated models, the learning task has also grown accordingly. As a result, the optimisation procedures required to identify the overall structure and parameters of these more complex models has also become more sophisticated. Therefore, in the selection of a suitable model structure the level of optimisation required is an important consideration.

2.6.1) Types of Learning

Techniques for optimisation can be categorised into three different approaches that are distinguished by the amount of information or data that would be required by the chosen model architecture.

2.6.1.1) Supervised Learning

Supervised learning methods require that both input data and output data of the process is available. Typically this would involve empirical data consisting of matching pairs of input and output data. The objective of supervised learning techniques is to identify an optimal solution through the minimisation of a measurement of the error between the model and that of the observed process. In order to provide this measurement of the error, a loss function is employed to analyse the difference between each possible model solution and the target output against some criteria. From a machine learning perspective, the use of output data can be seen to perform the role of a ‘teacher’ and therefore provide supervision for the learning system. The supplied output data can therefore be interpreted as examples of a correct response that allows a comparison to be made with the learning system’s current solution. Most optimisation problems in system identification can be seen to fall within the domain of a supervised learning algorithm, as we would typically

expect the output of real systems to be available. Therefore, it is supervised learning that this thesis will focus on.

2.6.1.2) Reinforcement Learning

In reinforcement learning, a degree of information about the quality of the model is available, but no desired output value is known for each input. This type of learning is normally employed toward evaluating the quality of different strategies or long-term goals, rather than evaluating the error of individual test (input-output) cases. This kind of learning has particular relevance in a number of applications (e.g. robotic control, dynamic programming and gaming strategies) where it is difficult to assess the quality of individual manoeuvres or events, as it is the final outcome that will determine success or failure. For more information on this subject, see Sutton and Barto (1998) and Kaebling et al. (1996).

2.6.1.3) Unsupervised Learning

In unsupervised learning, only the input data is typically available or used. Unsupervised learning techniques are therefore used to extract any compressed information about the input data distribution. Furthermore, as no output information is utilised, unsupervised learning methods are typically employed in conjunction with supervised learning methods in order to obtain an optimal model solution. As a result, unsupervised learning techniques are predominantly used as tools for data pre-processing. In Nelles (2001), two main categories of unsupervised learning are discussed, namely **Principal Component Analysis** (PCA) and **Clustering** techniques.

The goal of PCA methods is to simplify the overall learning problem through transforming the input axes. In particular, PCA methods are used to reduce the dimensionality of the problem through eliminating any input axes that are uninformative about the data. Therefore, the relative significance of each input axes must be evaluated, and this is done through assessing the degree of variance (i.e. high-variance indicating high significance and vice versa). Such methods have been shown to be particularly valuable for high-dimensional problems, where computational demands and overall model complexity can be reduced. However, it is important to remember that this

dimensional reduction is performed through the analysis of only the input distribution, and the loss of important information is still possible (i.e. there is no reason why a low input variance should automatically imply a low significance of that particular input).

The goal of clustering techniques is to find groups of similar data samples. A similarity measure must therefore be defined, where the shape of the cluster is to be determined (e.g. circle, spherical, elliptical etc.). The number of clusters can be chosen initially or determined automatically in more advanced methods. Notable clustering techniques include the K-Means algorithm, Gustafson-Kessel Algorithm and Kohonen's Self-Organising Map. For a more detailed discussion on clustering techniques, see Ripley (1996), Kohonen (1990) and Bezdek (1981).

2.6.2) Parameter Optimisation

The overall task of model optimisation can be split into two parts: **parameter optimisation** and **structure optimisation**, in this section we focus on the former. The process of optimising the parameters is also known as parameter estimation and begins with the identification of a suitable criterion that defines the exact mathematical measure that is to be optimised. This criterion is also commonly termed a **Loss function** and is typically a measure of the error between the measured output of the system $y(i)$, and the corresponding output of the model $\hat{y}(i)$ for a defined training dataset, i.e. $e(i) = y(i) - \hat{y}(i)$.

In the general system identification literature, such as Ljung (1999) and Söderström and Stoica (1989), three different loss functions are commonly discussed: the **Least Squares** method (or sum of squared errors), **Maximum Likelihood** method, and the **Maximum A-Posteriori** (MAP) estimate. The Least squares method is the most widely adopted approach and forms the basis for linear optimisation methods to be discussed briefly below. The remaining two methods can be categorised as probabilistic approaches to parameter estimation. A probabilistic approach to modelling is relevant as it introduces the concept of uncertainty into the modelling procedure. The maximum likelihood method is to be employed for the optimisation of the GP model and is discussed in

chapter 4. The MAP estimate can be seen to be a form of Bayesian analysis that is to be discussed in the next chapter.

2.6.2.1) Linear Optimisation

Depending on the nature of the chosen model architecture, the parameter optimisation procedure can be termed as either linear or nonlinear optimisation. A linear optimisation problem can be said to exist if the model error can be categorised as being linear in the parameters $\boldsymbol{\theta}$ and if the sum of squares error (least squares) loss function is employed. Therefore, a model $\hat{y}$ of the dependent variable y, composed of n independent variables (regressors) x_n can written as:

$$\hat{y} = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i \tag{2.13}$$

In matrix form can be written as $\hat{\boldsymbol{y}} = \boldsymbol{X\theta}$, where $\boldsymbol{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ is the regression matrix and $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_n \end{bmatrix}^T$ is the parameter matrix.

Linear optimisation techniques are the most widely adopted parameter estimation methods due to their interpretability and ease of application. It is also due to these qualities that linear model architectures remain the preferred method in the field of system identification as a whole. Further desirable qualities of linear optimisation include the ability to provide optimum parameters that are unique (global), and the speed and robustness of the optimisation algorithm relative to nonlinear optimisation techniques. At this point it is worth briefly detailing the linear least-squares algorithm, further details of which can be found in the majority of system identification and statistical texts. The model error can be written as $\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X\theta}$, and the sum of square errors loss function can then be stated as $V(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e}$. As this loss function is quadratic in θ, the minimum value can be easily computed by setting the derivative of this function to zero. The least squares estimate can then be stated as $\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.

Utilising modern numerical software, such as Matlab, it is then straightforward to compute the least-squares estimate of the model parameters and then assess the performance of the identified model. However, it is important to point out that the required matrix inversion of the Hessian $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ can prove problematic if the matrix is ill-conditioned. Such matrix conditioning problems can result if the empirical data collected from the system and then used to generate the regression matrix is not sufficiently excited. This aspect is mentioned in most system identification texts such as Söderström and Stoica (1989). Furthermore, the direct inversion of these matrices is not normally carried out due to likelihood for mathematical difficulties and a number of alternative approaches, such as Gaussian elimination, Cholesky decomposition or singular value decomposition, are normally employed instead. Useful resources on various matrix methods include Barnett (1979) and Golub and Van Loan (1987). These matrix computation and conditioning aspects are particularly relevant as the same kind of problems can be seen to present themselves in the implementation of GP models. As a result a more detailed discussion of matrix conditioning aspects is to follow in Chapter 4. However, for linear optimisation implementations, regularisation techniques such as ridge regression have been developed to tackle matrix conditioning problems, see Tikhonov and Arsenin (1977) and Johansen (1997) for more details.

A further feature of linear least-squares optimisation is that using the regression matrix it is also possible to generate covariance matrices of the parameter estimate $\boldsymbol{\theta}$ and model output $\hat{y}$. Utilising the covariance matrix of $\hat{y}$ it is therefore possible to generate measures of variance and therefore errorbars, if some information regarding the noise distribution is known or assumed. More detailed information on this can be found in Nelles (2001), however it is mentioned here due to the similarities found in the GP modelling approach which involves the specification of a covariance matrix. Therefore, a more detailed discussion of covariance and covariance matrices is found in the next chapter.

Extensions to the linear least-squares algorithm also include the weighted least-squares implementation, where the contribution of each squared error can be weighted with a factor. This facility allows knowledge of the relevance or confidence in each data sample to be incorporated. A further extension is the recursive least squares algorithm that allows the parameter vector to be updated whilst online. This is a useful feature for real-

time implementations where the model must be updated. More general information on linear optimisation techniques can be found in Draper and Smith (1998) and Wolberg (2005).

2.6.2.2) Nonlinear Optimisation

If the model function is nonlinear in the parameters (i.e. the parameters appear as functions), a nonlinear optimisation technique must be applied to search for optimal parameters. However, the general goal of nonlinear optimisation techniques remains the same, to find the minimum of a given loss function with respect to the parameters. A wide range of nonlinear optimisation techniques have been developed and good general sources of information and algorithms include Scales (1985), Reklaitis et al. (1983), Vanderplaats (1984) and Press et al. (1992). Nonlinear optimisation can prove to be a challenging endeavour due to the potential presence of multiple local optima. Therefore, more than one set of 'optimal' parameters may be identified from data and care must be taken to find the most appropriate solution (i.e. some optimal parameter values will lead to better models than others). Furthermore, as more than one possible solution can exist, in contrast to the computationally desirable 'one-shot' solution typical of linear optimisation, nonlinear optimisation techniques are iterative in nature and require algorithms that search for and then converge on local optima. As a result, nonlinear optimisation methods are not typically suited for online application.

Due to the iterative nature of nonlinear optimisation methods, in order to identify a good local optimum and speed up the algorithm's convergence to such a solution, an important consideration is the choice of initial parameters. Whilst a random or arbitrary choice of initial parameters may result in the convergence toward a suitable optimum, the selection of a good set of initial parameters (through the use of prior knowledge) can increase the chances of a good result and speed up the process considerably. Furthermore, nonlinear optimisation methods can also be categorised into Local and Global methods, as in Nelles (2001).

Although both methods will converge on local optima, local optimisation methods tend to converge on local optima close to the supplied initial conditions as search directions are obtained from neighbourhood information such as first and second order derivatives.

As a result, local algorithms may become stuck in poor local minima and more suitable optima in other regions of parameter space may not be considered. Global nonlinear optimisation methods are aimed at overcoming this problem, and typically rely on the inclusion of random or stochastic components that allow the algorithm to escape from local optima. Notable global optimisation techniques include Simulated Annealing, described in Kirkpatrick et al. (1983) and Laarhoven and Aarts (1987), and a range of Evolutionary algorithms such as the Genetic Algorithm, see Holland (1975) and Goldberg (1989) for details. However, as global methods are to search the whole parameter space (potentially for multiple parameters) a significant disadvantage is the high computational demand and slow convergence speed of these algorithms. As a result, nonlinear local optimisation methods that are generally faster to converge remain more popular. Furthermore, using local methods, it is possible to obtain a more global optimum through applying a 'multi-start' approach where a number of local optimisations are performed using different initial parameters, and the best solution then chosen. A further option is to combine the use of global and local methods, e.g. using global methods to locate the region around suitable local optima, and then deploying a faster converging local optimisation method to provide a more precise local estimate.

The simplest general-purpose nonlinear local optimisation techniques are termed Direct Search methods. These include Simplex Search and Hookes-Jeeves methods, and utilise only loss function values in their search for local optima. These methods are typically slow to converge and only used when the derivatives of the loss function are not available or require significant computation time to compute. As a result, local optimisation approaches that make specific use of gradient information are amongst the most widely adopted methods. Notable gradient-based methods include Steepest Descent, Newton's Method, Quasi-Newton, and Conjugate-Gradient methods. These methods are reviewed in depth in Scales (1985), but the general concept of these methods is given by $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta_{k-1}\boldsymbol{p}_{k-1}$ with $\boldsymbol{p}_{k-1} = \boldsymbol{R}_{k-1}\boldsymbol{g}_{k-1}$. Where the parameter vector $\boldsymbol{\theta}_k$ is to be updated by the quantity $\eta_{k-1}\boldsymbol{p}_{k-1}$, where η_{k-1} is a step-size (typically determined by 'line search' methods) that fixes the proportionality of the search direction $\boldsymbol{p}_{k-1}$, which is defined by the gradient direction $\boldsymbol{g}_{k-1}$ that is rotated and scaled by a direction (or rotation) matrix $\boldsymbol{R}_{k-1}$. The different gradient-based methods of local optimisation algorithms can then be defined by different choices of step-size and rotation matrix.

The simplest gradient-based method is the Steepest Descent method where the rotation matrix is taken to be an identity I matrix. This method is notable for its non-requirement for second-order derivatives of the loss function, but is typically slow to converge and has been somewhat made redundant by more sophisticated methods. In Nelles (2001), the equivalence between the Steepest Descent method and the famous backpropagation algorithm used in the training of Neural Networks is discussed. The Newton's method employs the inverse of the Hessian matrix for use as the rotation matrix, and therefore brings a demand for second-order derivatives which may be computationally expensive to compute if unavailable analytically. Furthermore, Newton's method is computationally demanding due to the need for matrix inversion and is therefore recommended only for small optimisation problems. The Quasi-Newton method is therefore aimed at reducing the computational complexity through replacing the inverse Hessian used as the rotation matrix, with an approximation to this inverse (so the computational demands of inversion can be avoided). A popular formula for defining this approximation is the Broyden-Fletcher-Goldfarb-Shanno (BFGS), as described in Scales (1985).

Both Newton and Quasi-Newton methods are typically described as having very good convergence properties (i.e. fast convergence in terms of number of required iterations rather than computational demand), but for large problems these methods are still found to suffer from excessive computational demands. The Conjugate-Gradient method is a further alternative local optimisation method that can be seen to be less computationally intensive. Rather than attempt to directly approximate the Hessian, the conjugate-gradients method employs a rougher approximation where the search direction is computed in a more direct manner as $\boldsymbol{p}_{k-1} = \boldsymbol{g}_{k-1} - \beta \boldsymbol{p}_{k-2}$. Different conjugate-gradients methods can then be distinguished through the choice of the scalar β, with popular choices being the Fletcher-Reeves and Polak-Ribiere methods, see Fletcher (1993) for more precise details. Conjugate gradient methods are typically found to require a higher number of iterations than the Quasi-Newton and Newton methods to converge upon an optimum, however due to their less demanding computational nature, the overall speed of the algorithm is found to be superior. Therefore, conjugate-gradient methods are the preferred choice for larger optimisation problems, however they typically require a more accurate line search to be performed.

Whilst these general gradient-based methods of local optimisation are well established, further alternative methods can be considered if the loss function to be minimised is of the sum of squares type. These methods are known as nonlinear least squares methods and are typically recommended over the previously discussed choices if the loss function is of the required type. The two most common nonlinear least-squares methods are the Gauss-Newton method and the Levenberg-Marquandt method. The nonlinear least-squares loss function can be formulated as $V(\boldsymbol{\theta}) = \boldsymbol{f}^T \boldsymbol{f}$ with $\boldsymbol{f} = [\, f(1,\boldsymbol{\theta}) \quad \cdots \quad f(N,\boldsymbol{\theta})]$, and the gradient as $\boldsymbol{g} = 2\boldsymbol{J}^T \boldsymbol{f}$ where J is the Jacobian matrix (first-order derivatives). For both the Gauss-Newton and Levenberg-Marquandt methods, an approximation can then be introduced where the Hessian matrix can be approximated as $\boldsymbol{H} \approx \boldsymbol{J}^T \boldsymbol{J}$. This allows the both methods to approximate the second-order derivatives of the Hessian through the first-order derivatives of the Jacobian, which results in a computational saving. For more information on this approximation and its assumptions, see Scales (1985). The Gauss-Newton method can then be described by $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta_{k-1}(\boldsymbol{J}_{k-1}^T \boldsymbol{J}_{k-1})^{-1} \boldsymbol{J}_{k-1}^T \boldsymbol{f}_{k-1}$ and the Levenberg-Marquandt method can be described by $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta_{k-1}(\boldsymbol{J}_{k-1}^T \boldsymbol{J}_{k-1} + \alpha_{k-1}\boldsymbol{I})^{-1} \boldsymbol{J}_{k-1}^T \boldsymbol{f}_{k-1}$.

The Levenberg-Marquandt method can therefore be seen to be an extension of the Gauss-Newton method where the quantity α is introduced. This feature can be seen to be equivalent to the ridge-regression regularisation method employed in linear regression and acts to overcome matrix-conditioning problems associated with the Gauss-Newton method. Furthermore, as with the linear regression case, the inversion of these matrices would again not be undertaken directly and less mathematically problematic methods such as Cholesky decomposition would be employed instead.

2.6.3) Model Structure/Complexity Optimisation

The task of optimising the structure of a chosen model can also be interpreted as the optimisation of the model's complexity. Whilst there are many different characteristics that could be used to describe the complexity of a model, the term 'model complexity' is usually generally related to the number of parameters present. Therefore, a model is said to increase in complexity if parameters are added, and vice-versa. Furthermore, with more parameters a model is said to increase in flexibility, where the variety of possible

functions that could be described by the model is subsequently increased. Fundamentally, a model that is too simple will potentially fail to sufficiently capture the behaviour of the underlying system, and thus lead to inaccurate predictions. However, a more complex model constructed from a large number of parameters can also perform poorly if the amount of available training data is insufficient.

Therefore, in order to identify a good model, the complexity of the model must be appropriate for the task. The system should be over-determined (especially in complex regions of the input space) in that more training data than model parameters should exist. The process of optimising the complexity of the model is normally performed in conjunction with the model validation stage (to be discussed in the next section) where the performance of the model is assessed before any modifications are made. An important feature of the model validation process is that the model's performance is examined on a separate 'test' dataset that is different from the training dataset. The importance of this strategy is that in this way the **generalisation** ability of the identified model is examined. Generalisation is the ability of model to provide an accurate prediction of the system output when presented with inputs on which it has never been trained. This is a important objective of the modelling process, as we are interested in obtaining a model that is robust and performs well on a range of new (test) data, rather than models which perform well on limited range of data or indeed a memorisation of the training dataset. In describing the generalisation ability of the model, it is common for the terms **underfitting** or **overfitting** to be used to describe models that perform poorly. In cases where the test data is estimated poorly by a model that would appear to be insufficiently flexible (i.e. too simple), this is generally known as underfitting. For cases where a complex/flexible model is employed and the training set appears to be learned to a reasonable extent, but the generalisation remains poor, this may be evidence of overfitting. The model has potentially learned the noise present in the training data, or has correctly learned the data but the interpolation between datapoints is incorrect.

2.6.3.1) Bias/Variance Dilemma

In analysing the generalisation ability of the model, a useful strategy is to decompose the model error into two components, the **bias error** and the **variance error**. In this section

this decomposition is briefly discussed but a full account of this can be found in Nelles (2001) and Geman et al. (1992). The model error initially be formulated as

$$\mathrm{E}\{(y-\hat{y})^2\} = \mathrm{E}\{(y_u-\hat{y})^2\} + \mathrm{E}\{n^2\} \tag{2.14}$$

Where $\hat{y}$ is the model output, y is the measured system output, and y_u is the true output that is uncorrupted by noise. The first part of this expression is the model error between the true system output and the model output, and the second part of this expression represents the noise variance. Therefore, assuming the model does not represent the system exactly (where the first part would disappear), and the model does not influence the noise variance, the model error can be decomposed into bias and variance parts accordingly:

$$\begin{aligned} &\text{(model error)}^2 = \text{(bias error)}^2 + \text{variance error} \\ &\mathrm{E}\{(y_u-\hat{y})^2\} = [y_u - \mathrm{E}\{\hat{y}\}]^2 + \mathrm{E}\{[\hat{y} - E\{\hat{y}\}]^2\} \end{aligned} \tag{2.15}$$

The bias error can be interpreted as the proportion of the model error that is due to the fundamental difference between the model structure and the system or process. In real systems the process may be significantly more complex than the class of models that are to be considered for application. This lack of flexibility in the model means that an exact representation of the system is impossible, and this deviation is known as the bias error. Therefore in order to reduce the bias error, it is necessary to make the model more flexible which leads to a growth in the number of model parameters. In theory this leads to a strategy of employing as complex a model as computationally feasible, however this is not normally realisable due to the variance error component of the model error.

The variance error can be interpreted as the proportion of the model error that is due to differences between the estimated model parameters and their optimal values, resulting from inadequate optimisation. As the identification of real systems requires that the model parameters must be estimated from noisy training data that is of limited size, a difference between the estimated parameters and the truly optimal parameters is likely to be present.

Therefore, for cases where a high variance error is present, the optimisation of the model parameters must be improved. One strategy would be to reduce the number of parameters, as models containing fewer parameters will typically be easier to optimise using the same amount training data. Alternatively, larger training sets containing more information with which to estimate these parameters is a method of reducing the variance error. Fundamentally, the number of parameters in the model should not exceed the number of training data samples, as if they are equal the model will be fit exactly to the noisy training data and then generalise poorly on different noisy test data. Furthermore, higher levels of noise present in the data will result in higher variance error, and require larger training datasets in order to compensate. For very complex or flexible models, the variance error will dominate the model error and the bias error can become negligible.

Overall, it can be seen that in order to reduce the bias error, a model should be made more complex thus increasing number of parameters. However, a model composed of more parameters will result in a larger variance error unless the training dataset used to estimate these parameters is also increased. For many implementations, the data available for training is limited and the inclusion of more data is not something that is feasible. Therefore, the bias and variance components of the model error can be seen to be in conflict (hence the 'dilemma') and must be traded off against one another in order to achieve optimal model complexity. The difficulty in the implementation of this trade-off is that the bias/variance error components are typically unknown.

2.6.3.2) Model Complexity Optimisation Strategies

Given that the chosen model structure should only be as complex as necessary, a large number of different strategies for dealing with complexity have been proposed. In addition to the potential overfitting problems discussed previously, more complex structures will typically require a more computationally demanding parameter optimisation stage to be completed. Furthermore, a more complex model may have a detrimental impact on the overall interpretability of the model.

If the chosen model architecture is found to 'underfit' the test data it is obvious that the model in its current guise is not sufficiently well equipped to handle the learning task at hand, and the addition of further parameters or the consideration of alternative structures

may be required. However, before eliminating potential candidate models, it must be clear that the problem lies with the lack of flexibility of the model, rather than with insufficient training data or a poorly executed parameter optimisation strategy. For the case of an overly complex 'overfit' model, a number of strategies exist to help regulate the complexity of the chosen model architecture. Furthermore, other approaches have been developed that seek to reconstitute the learning problem allowing different and possibly less complex model architectures to be utilised.

Through the course of this chapter a number of ideas for dealing with complexity have already been mentioned, however a fundamental concept worth reiterating is the 'curse of dimensionality', where the learning task becomes harder solve as the dimensionality of the input space, and therefore the number of parameters, increases. In trying to solve harder learning problems, more complex models are therefore required and the potential for overfitting increases. Therefore, adopting modelling strategies that seek to mitigate this 'curse of dimensionality' may ease the identification of more optimally complex models. In particular, the various network structures discussed previously typically do not suffer from the same rate of growth in the number of parameters with the number of input dimensions, as more classical nonlinear methods. Furthermore, many of the unsupervised learning methods such as clustering and principal component analysis are expressly concerned with helping to organise and simplify the learning problem.

For linear models composed of a number of regressors, well established subset selection methods have been developed to help determine which of these regressors are the most important, therefore allowing the number of parameters to be optimised. Important subset selection techniques include forward selection, backward elimination and stepwise selection; see Nelles (2001) for a full discussion. These methods can also be adopted in the optimisation of Neural Network based structures where basis functions or neurons can be added to or eliminated from the model (i.e. the network may be 'grown' or 'pruned' accordingly). Furthermore, the adoption of the operating regime approach where a divide and conquer approach is applied to the operating range of the system can help facilitate the inclusion of prior knowledge into the identification process, e.g. local linear models identified at known equilibrium regions. Another possible approach to decomposing the input space is the classification and regression tree (CART) methods proposed in Breiman et al. (1984).

Further alternative methods include the use of 'Hybrid' model structures where more than one type of model are combined together. Through such an approach, the identification process may be further broken down into more manageable components where a number of simpler model structures can then be more easily identified and then combined in some arrangement (e.g. additive or hierarchical models). In this way, models that are well suited to representing certain aspects of a system's behaviour (e.g. an existing model derived from first principles) can be combined with other methods (e.g. a sophisticated black-box model derived from empirical data). Another possibility is the adoption of a 'projection-based' approach where instead of eliminating or partitioning regions of the input space, the input space is projected onto a different set of axes and the learning task redefined. This is the basis of various kernel methods and also the GP modelling approach examined in this thesis, where a probabilistic 'function space' is defined and Bayesian learning is employed towards finding optimal model complexity.

So far, these complexity optimisation strategies have tackled the problem through the modification of the model structure and are in essence attempting to reduce the number of model parameters to be included. However, an alternative strategy is offered by various **regularisation** methods where instead attempting to reduce the number of parameters included by elimination, the goal is to restrict the overall influence of the parameters. Therefore, the effect of regularisation is to compel the model behave as though the model is composed of fewer parameters than it actually has. As a result, regularisation can be seen to reduce the number of 'effective' model parameters and has a smoothing effect (reducing variance) on the model output. For linear regression problems the previously mentioned technique of ridge regression is a particularly prominent method of regularisation, and the same principle is employed for neural network type problems under the guise of 'Weight Decay', as discussed in Nelles (2001). Further methods of regularisation include the inclusion of constraints on the parameters values (e.g. certain parameters are given a fixed value or restricted range, or must be positive or negative), and 'staggered' optimisation where instead of attempting to optimise all model parameters simultaneously, the learning task is split up with subsets of parameters being optimised in turn. Another notable regularisation strategy is 'Early Stopping' where instead of allowing the iterative nonlinear optimisation algorithm to converge to a minimum using solely the training data, a set of test or validation data is also employed and the iterative optimisation algorithm is concluded when the model

error on this validation set reaches its minimum. This is a good strategy to adopt for very flexible models where the convergence of all parameters may not be desirable.

2.7) Model Validation

Once the selection and optimisation of the model has been performed, a process of model validation should be completed before the model may be declared as ready for use. This process of model validation is an important but often overlooked stage of the overall modelling process as a whole. In the context of the system identification loop discussed previously, model validation involves the careful evaluation of the model against some performance criteria. The discussion of these criteria is often restricted to the application of statistical testing of model accuracy (e.g. Mean-square error), but the application of more subjective reasoning (i.e. using prior knowledge) should also be incorporated. In this way, the overall suitability of an identified model can be assessed (e.g. is the model interpretable?). It is normal to find a trade-off exists between more complex and therefore flexible models, and the overall interpretability of the approximation. If the model is found to perform adequately then the overall identification process can be said to be complete, however if some aspects of the model's performance are seen to be deficient, the practitioner should return to previous stages of the modelling process and consider modifications. Using the information gained from the model validation stage, various strengths and weaknesses of the identified model can become clearer (e.g. particular operating regions where accuracy is poor), thus facilitating any necessary modifications.

Once a model has been identified from a set of training data, the most straightforward method of evaluating the performance of the model is to then test the model on a different set of data. This concept of splitting the overall set of empirical data into separate training and test datasets is generally known as **cross-validation**. For cases where an abundance of empirical data is available, this validation procedure would not prove to be problematic. However, for cases where the amount of empirical data is limited, the requirement for separate training and test datasets can be difficult to meet. As discussed previously, in order for a successful model to be identified, the training data must be representative of the unknown system function. Therefore, through experimental

design and data pre-processing, the training dataset must attempt to include sufficient datapoints from the all regions of operating space that are to be considered. However by the same rational, if we are to fully evaluate the performance of the model, then the test dataset must also be representative of as much of the operating range as possible.

Fundamentally, a significant restriction on the size of the training dataset in order to provide large amounts of test data can be seen to be counter-productive (even wasteful of data), as the modeller is of course charged with identifying the best possible model. Therefore, in cases where empirical data is limited (a common problem in the identification of real systems), it is normal for pressure to be put on restricting the amount of available test data in order to boost the amount of training data. As a result, instead of arbitrarily splitting the available data, more sophisticated validation methods have been proposed that seek to maximise the exploitation of the available data. One such method is ***n*-fold cross-validation** where the available empirical data is partitioned into n sub-samples. Each sub-sample is then employed in turn as a test dataset for a model trained on the other (n-1) sub-samples, with the overall error rate being taken as the average of these n sub-sample tests. A further alternative is **Leave-One-Out-Validation**, where a single observation of the overall data is to be left out and used as a test example, and the remaining data used for training. As before, this process is then repeated until each member of the training set has also been used as a test example. This can be seen to be an extreme case of cross-validation that is only computationally feasible for small datasets.

As cross-validation schemes can prove to be computationally expensive, the overall process of model validation can become frustrating for complex models. The overall process involved in testing the model, then potentially modifying and retraining the model, and then retesting it, can prove to be a time-consuming one. Therefore, alternative methods of evaluating test error that are less computationally expensive have been developed. These include the use of various **'information criteria'** methods, such as Akaike's Information Criterion (AIC), and Final Prediction Error (FPE), see Akaike (1974). For more information on various model validation strategies, see the aforementioned system identification texts by Ljung (1999) and Nelles (2001), and also Leontaritis and Billings (1987).

3) Gaussian Process Models

The Gaussian Process (GP) model may be regarded as a nonparametric method of nonlinear system identification where new predictions of system behaviour are computed through the use of Bayesian inference techniques applied to empirical data. The GP model may also loosely be considered as a 'black-box' method as the identification process relies heavily upon experimental data. However, as with other methods of identification, a priori knowledge such as physical insight can prove to be invaluable in the design of the model and experimental procedures. In this chapter we introduce the theoretical background and literature of the Gaussian Process modelling approach, together with a discussion of the motivation behind the methods.

3.1) What is a Gaussian Process Model?

Through the course of this chapter the mathematical framework of the GP modelling approach is to be presented in detail. However, before beginning an exploration of the methods it is first necessary to place the GP modelling approach in context with alternative modelling approaches. From this point, the motivation behind the proposed adoption of these methods can then be discussed.

The GP modelling approach is typically described as a nonparametric 'black box' method that employs Bayesian learning with which to identify a model of system behaviour. Therefore, in order to characterise the GP model, it is first important to realise that the GP approach has much in common with alternative nonparametric methods where identification is performed through the application of learning techniques to empirical data. In the previous chapter, the Neural Network approach was briefly described and it is this machine learning approach that the GP modelling approach is most often compared with. However, an important distinction between the two methods is that unlike the Neural Network's adaptive basis functions, the GP modelling approach can be thought to employ a fixed basis function or kernel. As will be discussed through the course of this chapter, this move from adaptive to fixed basis functions has advantages in dealing with the complexity issues (parameter and structural optimisation) raised in the previous chapter. Furthermore, it is through the use of Bayesian inference

coupled with the mathematical properties of the Gaussian process that the GP model will be shown to be a powerful method for nonlinear regression. Therefore, as with other nonlinear regression techniques the GP modelling approach can be understood as a method of interpolation, where a curve is to be fitted to data.

In order to illustrate the workings of the Gaussian Process modelling approach it is first necessary to outline the supervised learning problem to be tackled. Namely, we are seeking to identify an unknown function y, by constructing a model from noisy data with which we can use to make predictions given new input data. A model for our noisy data example can be formulated as:

$$y_n = f(x_n) + \upsilon_n \tag{3.1}$$

The input is denoted as **x**, with the output or target denoted as y. The input is a vector **x** dependant on the number of input variables, and the target is continuous data for this regression case. In essence, the problem can be understood as a multiple-input, single-output (MISO) arrangement. The noisy training dataset *D* consists of N observations $D=\{(x_i,y_i)|\ i = 1,\ldots,N\}$, i.e. N pairs of L-dimensional input vectors $\{\mathbf{x}_i\}$ and scalar outputs $\{y_i\}$ for i = 1…N.

Given the observed behaviour present in the set of training data, we now wish to make predictions from the model for new inputs **x*** not seen in the training set. Therefore the problem is to make predictions for all possible inputs based on information given a finite set of training data examples. In mathematical terms this is known as inductive reasoning where a general conclusion may be drawn from a series of premises based on experience or experimental evidence.

In order to successfully identify the unknown function from a set of training data, the more popular and well-established methods of system identification, such as a parametric model structure or Neural Network, would seek to make assumptions or use prior knowledge with which to simplify the learning task. Furthermore, the modeller must ensure that any model considered suitably represents the underlying function, rather than just providing a valid fit to the observed data. Considerations such as the likely order of the underlying function would be used to gauge the necessary level of complexity of the

chosen model structure (e.g. would a linear or lower order polynomial model be sufficient?).

By utilising this parametric approach where a finite number of variables must be determined through optimisation, we run the risk of selecting a model structure insufficient in its flexibility to be able to satisfy the given goal in terms of providing an accurate representation of the function to be approximated. Consequently, the complexity of the chosen model structure may need to be increased so that the function characteristics are better recreated. As discussed in the previous chapter, the risk associated with such an endeavour is the possibility of overfitting the model to the training data. Furthermore, as the number of parameters increases, the need for a greater amount of training data may also become apparent.

In seeking to solve this problem, the Gaussian Process model differs from more conventional system identification methods by adopting a Bayesian approach to the learning task. The GP model does not seek to assume a functional form with which to compute new predictions. Instead, a prior probability is given to *every* possible function with the most likely function identified through the use of Bayesian inference. This approach would seem to present a significant problem as if we are looking at every possible function, and therefore not discounting any particular category of function as in the parametric case, the task of computing likely functions appears to be almost infinite! However, it is through the specific use of the mathematical properties of the Gaussian Process that this computing dilemma can be overcome.

3.2) Motivation for GP models

Before going into a detailed description of the theory and methods associated with the GP model approach it is worth discussing the motivation behind the recent interest in Gaussian Process methods. Furthermore, the GP model is mostly defined in terms of its relationship to other methods and its application of Bayesian inference towards the problem of nonlinear regression. Therefore, some background discussion is necessary to provide much of the rationale that lies behind the adoption of the approach.

In the previous chapter, models based around a Neural Network type approach were described as being of immense potential in providing a tool for universal approximation that could be applied across a wide range of problems. The central tenet that a group of adaptive basis functions, or hidden layers/structures, could be learned from data, allowed the user the flexibility to approximate complex nonlinearities and discover patterns in data that were previously hidden. Through the 1980s and early 1990s Neural Network techniques became very popular topics of study across many research fields and the underlying understanding of methods and their relation to existing statistical principles became apparent. Furthermore, the problems associated with employing such complex model structures and the general lack of a stringent unifying framework for implementation became clear.

The problem of optimising the complexity of these multiple models remains the biggest challenge presented to those who employ these methods. In tackling this problem, the principle that a model should only be as complex as completely necessary for the intended application is almost fundamental to the field of system identification. In terms of interpretability, a simple model may often be preferable to a complex one. Furthermore, the idea that complexity should be minimised also becomes a practical necessity where the computational demand of identifying parameters becomes unviable. This principle of economy also relates to the philosophical concept of Occam's Razor, which states that assumptions should not be needlessly multiplied.

As discussed previously in section (2.6.3), the potential for 'overfitting' is significant when considering a model of high complexity. The Bias/Variance trade-off dictates that although a more complex model may be successful in reducing the bias error by more closely approximating the underlying process or function, there may be an increase in the variance error due to a potential tendency to approximate the function to any random variation in the training data. Conversely a simpler model may have a higher bias (dependant on whether the underlying process or function is comparably simple), but a lower variance. The consequence of this trade-off is that in order for a complex model to be identified successfully, without being hamstrung by poor variance error, a large amount of training data is required. Therefore, from an overall perspective, we are restricted in our choice of model complexity by the amount of training data available.

3.3) Dealing with Complexity

For many applications, large quantities of training data may not be available. Therefore, if we are to employ large complex networks to identify the underlying nonlinear function, the problem of promoting accuracy whilst restricting model complexity must be tackled. The various possible choices in architecture, activation functions and optimisation procedures discussed in the previous chapter show that whilst solutions to the complexity problem have been proposed, a great deal of uncertainty remains over which method would be most appropriate given a certain set of conditions. The consequence of this doubt has therefore led to the suspicion that perhaps the Neural Network was not the great 'catch-all' solution to supervised learning problems.

Within the System Identification community, the development of Neuro-Fuzzy and Local Linear methods can be seen as a direct response to the ambiguous principles of the Neural Network approach, and indeed as an integration of classical methods of identification (local linear models) and prior system knowledge into the powerful adaptive basis function network methodology. Other methods such as the inclusion of penalty functions to the optimisation process have also been successful in applying a specific upper limit on the complexity of the resultant description.

In tandem with this effort to simplify or make more methodical the Neural Network identification process, research has continued within the Statistical and Machine Learning communities on alternative 'kernel' methods. Rather than attempt to approximate a function through the use of large numbers of adaptive basis functions, these kernel methods approach the learning problem through the use of fixed basis functions or 'kernels'. Whilst this movement from adaptive to fixed basis functions may be seen to be somewhat of a backward step, it has been shown that if enough of these fixed basis functions are used, the problems associated with complexity/overfitting can be mitigated, and therefore the perceived limitation in flexibility may be overcome. Furthermore, as there is only one fixed basis function to optimise, the resultant model structure can be seen to have an advantage in overall simplicity.

Many of these kernel methods have been developed toward the problem of classification, rather than as a tool for nonlinear regression, with the most well known approach being the Support Vector Machine (SVM), see Vapnik (1995). These kernel methods rely upon the implementation of the 'Kernel Trick', Aizerman (1964), where original observations may be mapped onto a higher dimensional feature space. By performing this mapping process, computational savings can be made as the nonlinear algorithms can be transformed into linear algorithms. The Support Vector Machine has been further extended to tackle regression problems as in Drucker et al. (1997). Furthermore, the Gaussian Process model can be seen to be an example of a Kernel method or machine, as it relies on the use of a single optimised kernel rather than adaptive basis functions. However, the GP model can be distinguished from the majority of kernel methods due to its implementation of Bayesian methods.

3.4) The Bayesian Alternative

An alternative approach to overcoming the problem of overfitting in complex network models is to adopt a Bayesian framework. In the early 1990s there was significant progress in the field of adapting Bayesian methods to the field of machine learning in an effort to solve learning problems, see Mackay (1991). These Bayesian methods attempted to address the problems associated with employing complex learning systems through the use of a probabilistic framework. The importance of adopting a probabilistic approach, where a prior distribution is defined and then a posterior distribution is inferred, is that through these distributions, information may be gleaned about both the overall error of the approximation, and the uncertainty or likelihood associated with this error. This new information may then be redeployed toward the goal of improving the approximation. In the analysis of identification arising from the use of a non-Bayesian approach, only information regarding the size of the error may be forthcoming.

The field of Bayesian modelling originates from within the statistics community where the probabilities of various events or outcomes must be calculated. The term Bayesian refers to the use of Bayesian inference, an interpretation of probability that allows the degree of belief in a hypotheses or event to be the basis of an estimate of its probability.

Through the use of Bayes' Theorem, this prior 'degree of belief' can then be revised or updated upon the discovery of new information, giving a posterior estimate.

The Bayesian interpretation of probability is often quoted as being in keeping with the scientific method, where a rule or hypothesis is first proposed, and revised upon any new discoveries. However, the concepts of Bayesian probability remain somewhat controversial within the statistical community as they are in contrast to that of the classical Frequency probability interpretation. In the Frequency interpretation, the probability associated with an event is defined as the limit of its relative frequency after observation in a large number of trials (e.g. the observed frequency of 'heads' when tossing a 'fair' coin should indicate that the probability of the event equals ½, given a large enough number of tosses). In contrast, a Bayesian will use a probability distribution over possible values for an unknown probability to express this uncertainty, and will then update this distribution as the outcome of each toss becomes known using probability theory.

The introduction of the Prior probability is the fundamental step that allows us to move from a likelihood function to a posterior probability distribution through the application of Bayes' Theorem. The adoption of this prior is also the main source of contention between Frequentist and Bayesian theorists, as the choice of prior can be viewed as arbitrary in many respects as the decision is made in the absence of experimental evidence. The counter argument is that the choices made are often done so on the basis of some kind of knowledge and are therefore not truly arbitrary. Furthermore, it could be argued that in the Bayesian approach our prior beliefs are at least stated explicitly, rather than employed tacitly as in other methods of probabilistic analysis.

To support this discussion on Bayesian modelling and the details of the GP model, Appendix A contains a brief overview of the most relevant probability definitions. Useful introductions to the topic of Bayesian statistics are Box & Taio (1973), Press (1989) and Lee (2004). More advanced methods including the application of Bayesian methods to regression and classification problems are examined in Gull (1988), Gelman et al. (2004), Congdon (2003) and Denison et al. (2002). However, the growth of interest in GP models can be seen to predominantly originate from investigations into the use of Bayesian learning in Neural Network implementations. Important sources of information

on this particular aspect are the texts by Mackay (1991, 1992a, 1992b) and Neal (1996), and much of the forthcoming discussion can be seen to have its origins in this research.

3.5) Bayesian Learning

The Bayesian approach to the modelling of data is based upon the expression of knowledge in terms of probability distributions. Whereas more conventional parametric modelling approaches would seek to optimise the parameters of a model structure so that any model error is minimised, a Bayesian approach would seek to maximise the probability of a model given some data. Therefore, rather than dealing directly with the error in the model, we are to operate upon the probability of the model given the data. Consequently, if the goal of the modelling process is to obtain a prediction estimate, rather than directly computing the value of a new prediction y^{N+1}, we must first find the probability of this new prediction $P(y^{N+1})$.

The Bayesian approach begins in a similar fashion to that of a more conventional parametric modelling approach. From examination of any 'a priori' knowledge of the unknown function, we can speculate upon a number of initial model structures or hypotheses H_i {i.e. H_1, H_2 … H_L} that we believe may offer the level of flexibility or sophistication (e.g. would a linear model suffice?) needed to form an accurate representation. This set of models can be termed the 'hypothesis space' with each model said to be characterised by a set of parameters $\mathbf{w}_i$, which are to be identified through some empirical data D.

This collection of models can be thought to be competing with one another to account for the data we have obtained, with each model H_i aiming to maximise the plausibility of the data. In this sense, the Bayesian approach adopted here differs from other probabilistic interpretations, as it is the inverse probability (rather than forward probability) that is to be employed through the use of the Likelihood principle. By doing so, the relative plausibility of these alternative models are to be computed based on the information present in the single data set that is to be observed. In the Bayesian framework our initial model or hypotheses H_i would be termed as a 'prior belief', and expressed as a **Prior** distribution over all possible models $P(H_i)$. These initial beliefs (prior to the arrival of

any data) about the relative plausibility of these models can be thought to be a quantified list of probabilities $P(H_1)$, $P(H_2)$, …, $P(H_L)$ which sum up to 1 (a certain event). Furthermore, we can define a prior distribution over the parameters $\mathbf{w}_i$ that is conditional on this initial model $P(\mathbf{w}_i|H_i)$.

3.5.1) Levels of Inference

After deciding upon our possible models and observing experimental data, the Bayesian modelling approach has two stages or '*levels of inference*' as described in Mackay (1991).

3.5.1.1) 1st Level of Inference

In the first level of inference, the task is to fit each model to the observed data through applying Bayes' theorem to infer the parameters $\mathbf{w}_i$ of each model. Therefore, this stage of the Bayesian approach is fundamentally similar to other modelling approaches where the parameters of a proposed model are to be optimised using information gained from the empirical data. Therefore, our goal is to infer a probability distribution over the parameters that is conditional on the data and the model/hypothesis $P(\mathbf{w}_i|D,H_i)$.

This first level of inference is performed through the application of Bayes' theorem, where a **Posterior** distribution is inferred from combining the information present in the **Prior** $P(\mathbf{w}_i|H_i)$ with that of the information gained from the data. The information gained from the data is known as the **Likelihood**, $P(D|\mathbf{w}_i,H_i)$, which gives the probability of the observed data as a function of the unknown model parameters. This probability can be said to be conditional on the initial model structure H_i and parameters $\mathbf{w}_i$, and is often expressed as the likelihood function $L(\mathbf{w}_i)$:

Bayes' Theorem: Posterior = Likelihood x Prior / Evidence

$$P(w_i \mid D, H_i) = \frac{P(D \mid w_i, H_i)P(w_i \mid H_i)}{P(D \mid H_i)} \tag{3.2}$$

If our set of training data D is, for example $\{(x^{(1)},y^{(1)}),\ldots, (x^{(n)},y^{(n)})\}$, the **likelihood function** $L(\mathbf{w}_i)$ can be written as:

$$\begin{aligned} \mathrm{L}(w) &= L(w \mid (x^{(1)}, y^{(1)}),...,(x^{(n)}, y^{(n)})) \\ \mathrm{L}(w) &\propto P((x^{(1)}, y^{(1)}),...,(x^{(n)}, y^{(n)}) \mid w) = \prod_{i=1}^{n} P((x^{(i)}, y^{(i)}) \mid w) \end{aligned} \tag{3.3}$$

The unknown parameters of the model can now be identified through the optimisation of this likelihood function, with methods such as **maximum likelihood** or maximum-penalised likelihood.

The **Posterior** distribution can therefore be seen to be the product of the **Likelihood** and the **Prior**, with the **Evidence** $P(D|H_i)$ (also known as the marginal likelihood) acting as a normalising constant. This marginal likelihood or evidence quantity can be somewhat ignored in the first level of inference. However, it is this evidence quantity that allows comparisons to be made between the likelihood of different models in the '**second level of inference'** discussed below.

Note that for more complex models we can adopt a hierarchical approach to reflect the hierarchy of the proposed description. For example, Neural Networks are often characterised by a set of parameters to control the individual weights of the hidden layer. A further set of hyperparameters that control the distribution of these lower level parameters may then be defined. To implement this for the first level of inference, the inference process can be repeated for the hyperparameter-level through defining a hyper-prior distribution over the hyperparameters.

3.5.1.1.1) Getting a predictive distribution

In the first level of inference, a posterior distribution over the model parameters has been achieved $P(\boldsymbol{w}|(x^{(1)},y^{(1)}),\ldots, (x^{(n)},y^{(n)}))$. However, it is the probability of the new output that we are ultimately interested in. Therefore, in order to obtain a **predictive** distribution for the probability of a new output, $P(y^{(n+1)})$, the Bayesian inference must be completed. This

is achieved through the **integration** of the model (i.e. $y^{(n+1)}$ from parameters w) with respect to the posterior distribution of the parameters

$$P(y^{(n+1)} \mid x^{(n+1)}, (x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})) = \int P(y^{(n+1)} \mid x^{(n+1)}, w) P(w \mid (x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})) dw \quad (3.4)$$

3.5.1.1.2) From predictive distribution to single-value prediction

In most modelling tasks, we wish to approximate the output of underlying system or process. So far we have formulated a method to provide us with a predictive distribution over the output, i.e. the probability of the unknown value $y^{(n+1)}$. For a single-value prediction, we must estimate the output y from this predictive distribution. The **mean** of such a probability distribution is therefore taken as the most probable estimate. Note that the precise choice of point estimate is dependant on the assessment criteria we are using to compare model error. For a squared error loss function the mean is appropriate, but the median of the distribution may prove to be a better choice if the model error is being analysed as an absolute error, see Neal (1996).

3.5.1.2) 2nd Level of Inference

In the first level of inference the parameters $\mathbf{w}_i$ of a particular model or hypothesis H_i were inferred from the observed data D using Bayes' theorem, resulting in the conditional $P(\mathbf{w}_i|D,H_i)$. Therefore, this process of model fitting is not radically different from other non-Bayesian approaches to the problem where a model is proposed and then optimised to reflect the available data. In the second level of inference, the objective is to compare a number of different models to find the most likely or plausible model given the data. As a result, this process has comparable goals to that of model validation, where a number of proposed models may be assessed and ranked accordingly.

The second level of inference once again employs Bayes' theorem to find the posterior probability of these different models given the data $P(H_i|D)$. The existing Prior probability distribution $P(H_i)$ (independent of parameters) over all possible models is employed in this inference. The likelihood $P(D|H_i)$ represents what the data is telling us

about the plausibility of each of the models H_i, and corresponds to the evidence or marginal likelihood quantity of the first level of inference. A normalising constant $P(D)$ is once again employed to ensure the probability sums to unity.

$$P(H_i \mid D) = \frac{P(D \mid H_i)P(H_i)}{P(D)} \tag{3.5}$$

The goal of this second level of inference is not to replace or replicate the process of validation where different models are compared in terms of accuracy or error. The Bayesian approach merely provides a further level of information regarding the probability of the model. This may then be used to distinguish between the suitability of competing models.

3.5.2) Evaluating Integrals

In order to implement the Bayesian approach a number of integrals must be evaluated in order to compute the posterior distributions of interest. Specifically, in the first level of inference, in order to obtain the posterior distribution of any new output y^{n+1}, we must integrate over the parameters. Furthermore, in order to infer the most likely parameters, we must evaluate the marginal likelihood or evidence that is itself an integral:

$$P(D \mid H_i) = \int P(D \mid w_i, H_i)P(w_i \mid H_i)dw_i \tag{3.6}$$

The evaluation of this integral over the parameter space is also important to any subsequent model comparison undertaken in the second level of inference. Therefore, the evaluation of the marginal likelihood is of fundamental importance to the implementation of the Bayesian inference as a whole, and is perhaps the most distinguishing feature of the Bayesian approach over more conventional methods of model optimisation and selection.

The constituent parts of this integral are the likelihood and prior distributions detailed in the application of Bayes' Theorem to the first level of inference. Both of these probability distributions can generally be seen to be nonlinear functions of the

parameters. The likelihood may often be expressed as a sum of a squared error term, and the prior may be of any form that we have deemed necessary to express our beliefs about the parameter values before any data has been observed. Taken together, this integral can prove to be analytically intractable and therefore impossible to evaluate directly. This is a significant problem with the Bayesian modelling approach and requires the use of approximation methods in many implementations.

The most general purpose and powerful methods of evaluating these intractable integrals rely upon the use of Markov Chain Monte Carlo (MCMC) methods. Other possibilities include methods based on the use of Gaussian approximations to the modes (peak values) of the posterior distribution. These Gaussian approximations rely upon the assumption that one or more modes of the posterior distribution can be initially located and that the most of the important information contained within the distribution is to be found close to these modes. Various methods of implementing Gaussian approximations have been described in Mackay (1991,1992b, 1992c), Thodberg (1996) and Hinton and van Camp (1993).

The approximation schemes based around the use of MCMC methods make no assumptions about the form of the posterior distribution under investigation, such as whether or not it might be approximated by a Gaussian. Therefore these methods are potentially more powerful and may be used to find multiple modes of the posterior distribution. However, the main disadvantage of MCMC methods is the computational demand many implementations require to converge to an adequate solution. The approach taken by Neal (1992a, 1993b, 1996) uses the Hybrid MCMC method for the implementation of Bayesian inference in Neural Networks.

For the sake of thesis brevity, a full discussion of MCMC methods is not included in this thesis, but general resources detailing the vagaries of Monte Carlo methods are Gilks et al. (1996), Mackay (1998a), and Smith and Roberts (1993). Furthermore, due to the reliance of many Bayesian inference implementations upon such statistical sampling algorithms, good introductions to MCMC methods can be found in the more general Bayesian analysis textbooks mentioned before, such as Gelman et al. (2004)

3.5.3) What Prior?

Another challenging aspect of the Bayesian approach is the determination of suitable prior probabilities with which to begin the inference process. These prior probabilities must embody our initial beliefs about the model before we have access to any data. For examples where a complex initial model is proposed, the task of expressing our prior beliefs over the model through assigning probability distributions, perhaps over different levels of parameter that are in themselves not readily interpretable, can be seen to be particularly demanding.

As a consequence of the relative difficulty of this task, combined with the mathematical complexities detailed previously, the temptation is to adopt a prior for mathematical convenience rather than truly expressing our beliefs about the underlying function. This course of action can be seen to be inconsistent with the philosophy of the Bayesian interpretation of probability, as the choice of prior should be made irrespective of mathematical convenience. To proceed otherwise would therefore invite questions as to whether it the methods used may be truly described as Bayesian, therefore placing doubt over the validity of the approach. To remain consistent with such formalism, objectivity must be maintained. A prior placed on an object should be determined through prior knowledge, and to meet the requirements for objectivity, those working with the same prior knowledge should reach the same conclusions and therefore propose the same priors.

Therefore, a balance must be struck between finding priors that are interpretable and readily applicable, whilst still reflecting our knowledge of the system. In practice, it is therefore common to apply prior distributions that are wide in terms of scope so as not to inherently rule too much in or out (e.g. a uniform or flat prior over models would not favour one model over another). For Bayesian Neural Network implementations independent Gaussian distributions are often used as the prior distributions over the parameters, see Mackay (1992b).

A final consideration to be made regarding the choice of priors, particularly priors over models to be subsequently compared in the second level of inference, is that if the ‘true’ model is not included within our set of possible models then it obviously cannot be

included within the comparison. This has been termed the ‘closed hypothesis space’; in that we can only compare models that have been distinctly specified, see Gibbs (1997). As a result, if no model has been specified that comes close to the ideal, then the subsequent inference process will not compensate for this inherent flaw, as there is no Bayesian criterion for assessing the suitability of the defined hypothesis space. This is in keeping with any other modelling paradigm where if a candidate model is inherently unsuitable for approximating the data, then no amount of optimisation will overcome this fundamental error in model selection.

3.5.4) Relating Back To Complexity

In the previous section we have discussed the Bayesian approach to nonlinear regression and then gone on to discuss some of the difficulties of implementation. However, we introduced the Bayesian approach as an alternative strategy for dealing with the problems of implementing complex models found in more conventional methods of model selection and optimisation. In these more established methods we found that a candidate model’s complexity may be limited by the quantity of training data available to the optimisation process.

By contrast, if a Bayesian approach is taken toward the goal of identifying a complex model, at no point in this procedure (Priors → Collect Data → Infer Posterior → Predictions → Comparison) is the complexity of the model modified to meet restrictions imposed by the amount of training data. Such a course of action can be seen to be inconsistent with the Bayesian perspective, as if a model or prior are deemed to be correct for cases where a certain number of data points are observed, they should theoretically remain correct for cases where more data points are available.

Unfortunately, although the theoretical principles of the Bayesian approach may preclude the influence of the size of the dataset over the model complexity realisable, limitations may still be imposed by more practical considerations. For example, if data is sparse for a given application, a simpler model may be deemed more suitable if the advantages of a more complex description cannot be realised without considerable computational expense. Furthermore, a moderately inaccurate prior might prove to be a more significant handicap to the identification process where the data is insufficient. Therefore, for

applications where a simple model is not likely to provide an adequate description, an appropriate Bayesian approach would be to implement the most complex solution that is computationally viable, disregarding the size of the training dataset.

3.5.4.1) Occam's Razor

In the identification of a model of suitable complexity, the principle of Occam's Razor is often cited as being relevant. The maxim dictates that in the presence of several compatible models or hypotheses, no more assumptions should be made than are necessary. This leads to a preference for simpler solutions over more complex alternatives, as they are often founded upon fewer assumptions. In some circumstances, this preference for a simple solution may be aesthetically motivated by the desire for a mathematically elegant solution, but it may also be construed as an excuse for reductionism or mathematical convenience. Nonetheless, invoking this concept towards the goal of regulating the complexity of a model has practical benefits in terms of computational demand and model interpretability.

Whilst we can employ this preference for simple models over more complex alternatives in any situation where competing models have been identified, by simply choosing the least complex model that still meets our accuracy requirements. However, the Bayesian setting outlined in this chapter offers a further level of information with which to employ this preference against complexity. Indeed, the Bayesian approach can be used to automatically apply Occam's Razor, allowing a simpler solution to become our preferred choice (rather than purposely having to implement some method of regularisation into our optimisation procedure that places an arbitrary upper limit on complexity). This automatic implementation of Occam's Razor relies upon the examination of the marginal likelihood or evidence, i.e. the probability of the data given the model.

To illustrate this feature, suppose that we have two competing models (H_1 and H_2) of different complexities, with H_2 being much more complex. Fundamentally, a simple model may only offer a limited variety of possible target values for a given set of inputs, whereas a more complex model has the scope to offer a wider range of possible targets due to the greater flexibility on offer. This relationship is visualised in Figure (3.1), which shows the behaviour of the marginal likelihood for the two different model

complexities, $P(D|H_1)$and $P(D|H_2)$. This figure and further discussion can be found in many of the aforementioned sources, including Mackay (1991, 2003) and Rasmussen and Williams (2006).

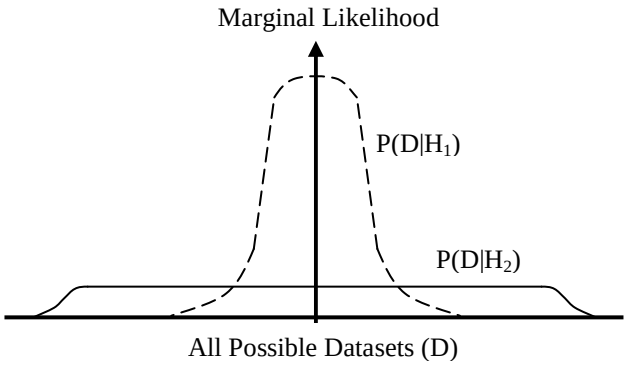


Figure (3.1): Occam's Razor from Marginal Likelihood

Interpreting this figure in terms of marginal likelihood (a probability distribution over the data given the model that is normalised to unity), a simple model will have a large value of marginal likelihood where the model does accurately account for the data, i.e. a precise fit to the data, but this distribution will be narrow (low variance) due to the limited potential of the simple model. Conversely, a more complex model offers the possibility of accounting for a wider range of data (high variance), but the value of the marginal likelihood for any given model will not reach the same magnitudes as would be seen for a simple model.

For cases where both models can be seen to be compatible with the data, the simpler model H_1 will have a larger marginal likelihood and therefore may be interpreted as more probable. Therefore, we may express a preference for a simpler solution through assessing the marginal likelihood of each model and selecting the most probable model. Furthermore, we have expressed our predilection towards simpler models without applying any arbitrary or subjective prejudice against more complex solutions, such as an external parameter to govern the trade-off between model complexity and model accuracy. Consequently, we can utilise the marginal likelihood as a tool for selecting an appropriate model complexity that is well suited to the observed data.

3.6) Gaussian Process Modelling

In this chapter the primary focus has been on building a case to support the consideration of a Bayesian approach to the supervised learning problem. The importance of this discussion is that by adopting this Bayesian formalism the overfitting problem normally associated with complex networks of a large size (Neural Networks or Kernel Machines) may be potentially overcome. In addition to outlining the potential of the Bayesian approach, the previous discussion highlighted potential problems regarding the identification of suitable priors, and the possible occurrence of intractable integrals that may require the use of time-consuming Markov Chain Monte Carlo (MCMC) methods for solution.

Fortunately, the main advantage of the Gaussian Process (GP) modelling approach is that we can remove some of the mathematical complexity associated with implementing a Bayesian framework, whilst retaining its features regarding its approach to dealing with complexity. Specifically, the mathematical properties of the Gaussian Process allow the problematic integrals associated with the evaluation of the marginal likelihood to become tractable, and therefore directly calculable thus forgoing the requirement for MCMC methods. Furthermore, much of the flexibility and power of approximation associated with the complex adaptive basis function methods such as the Neural Network can also be retained through the use of the GP model. Instead of a neural network composed of a finite number of adaptive basis functions, the GP model will be seen to correspond to an infinite network of fixed basis functions (or a kernel-based method), thus allowing significant computational savings to be made.

Much of the recent work on Gaussian Process models originates from the work of Mackay (1991, 1992a, 1992b) and Neal (1993) who applied the Bayesian approach toward the problems of learning with complex Neural Networks. In particular it was the research carried out by Neal (1996) into the possibilities of implementing infinite networks using Bayesian inference that lead to the realisation that, under some circumstances, a network composed of infinite fixed basis functions corresponds to a Gaussian process. Specifically, Neal (1996) demonstrated that under the Bayesian framework, neural networks with one hidden-layer converge to a Gaussian process as the

number of hidden layer neurons increases towards infinity, assuming suitable (zero-mean) priors.

The key outcome of this observation is that instead of struggling to define different priors to represent our initial beliefs about the proposed network (as would be the procedure in the Bayesian approach outlined previously), we can directly define a Gaussian Process as the prior over the possible functions. Due to the mathematical properties of the Gaussian process this is a significantly more straightforward task. As a result, the explicit use of Gaussian processes to express our prior beliefs was proposed as a potentially simpler approach to implementing Bayesian analysis.

This proposal of using Gaussian Processes as the basis for Bayesian nonlinear regression was taken up by Rasmussen and Williams (1996) who showed that the Gaussian Process models compared favourably to other approaches including Neural Networks and the Multivariate Adaptive Regression Splines (MARS) of Friedman (1991). In Rasmussen (1996) the GP method was also contrasted with a Bayesian Neural Network structure based upon the methods developed by Neal (1996) and was found to be significantly less computationally intensive as MCMC methods were not required.

The use of GPs gained further momentum through the work of Mackay (1997) where GP models were even postulated as a potential replacement for Neural Networks. Later versions of this paper with further information and discussion can also be found in Mackay (1998b, 2003). This work by Mackay was the first general review of the methods involved in undertaking supervised learning methods using Gaussian Processes. The reviews also contain discussion about how the approach can be seen to have notable equivalents and parallels across other learning methods and indeed in other areas of research. Important links between Gaussian Processes and other methods such as Kalman filters, Splines, and generalised radial basis functions are made apparent. Furthermore, the work by Mackay makes the observation that the use of Gaussian Processes for the purposes of regression can even be stretched back to work of O'Hagan (1978), Wiener (1948) and of the astronomer Thiele working in the 19^{th} century as described in Lauritzen (1981). However, the most interesting was the parallel made between Gaussian Processes and the well-established Geostatistics technique of Kriging, which uses the probabilistic analysis of data for the identification mineral deposits. The methods of Kriging have

been found to be identical to Gaussian Process regression, with the original research being conducted by Matheron (1963) and named after a mining engineer D.G. Krige. A review of these methods can be found in the text by Cressie (1994) devoted to statistical techniques for dealing with spatial data.

Further reviews of GP regression were completed by Gibbs (1997), Williams (1998) and Seeger (2004). The latest research into the GP model has focused upon overcoming some of the limitations of the approach and adapting it toward the task of nonlinear system identification. Precise details of these advancements and references are to be discussed in later sections of this chapter and the next. Much of this previous research has now been reviewed and brought together into a single volume by Rasmussen and Williams (2006), which builds on previous reviews and provides an excellent grounding in the theory and methods required for successfully adopting the GP approach. Furthermore, the text by Rasmussen and Williams (2006) provides a detailed comparison of the GP method with other machine learning architectures, such as the Support Vector Machine and Spline smoothing techniques. However, only a very limited amount of this research into the GP model has been aimed toward meeting the specific demands of system identification for engineering problems that is the main focus of this thesis.

3.6.1) What exactly is a Gaussian Process?

Given that the mathematical properties of a Gaussian Process are key to the mechanics of the Gaussian Process model, it is pertinent to discuss the peculiarities of this complex mathematical object. Put simply, a Gaussian Process is a stochastic process. But what exactly is a stochastic process?

The concept of a mathematical stochastic process was inspired by the need to model physical stochastic processes, which are processes in which the measured variable is governed by probabilistic laws. The most famous example of a physical stochastic process would be the Brownian motion of particles suspended in a liquid. A mathematical stochastic process can be defined as a collection or family of random variables. More loosely, a stochastic process may be thought of as a random function, with each function value being a random variable. Indeed, a random variable can be thought to have been created by a random or stochastic process. A further distinction can

be made between a stochastic process and a random field. Whereas a stochastic process can be thought to be evolving and therefore indexed with time (at time *t*, a random variable X_t is specified), a random field can be seen to be a collection of random numbers whose values are mapped onto any defined space of *n* dimensions.

Another important definition to consider is the fact that a random variable is actually defined as a mathematical function itself, rather than a true mathematical variable of assignable value. A random variable will map the possible outcomes of an experiment rather than describe the actual outcome. With each random variable, a probability distribution can be defined to describe its qualities, i.e. if X is a random variable, the corresponding probability distribution assigns to an interval $[x_1, x_2]$ the probability $P[x_1 \leq X \leq x_2]$. A probability distribution is often further characterised by a probability density function (PDF) where integrals are defined over an interval to calculate the precise probability.

Given these definitions, we can more precisely define a Gaussian Process to be a generalisation of the Gaussian (or normal) probability distribution (i.e. random variable $X \sim \text{Normal}(\mu, \sigma^2)$), where sample functions generated over time $\{X_t\}$ have the property that any linear combination will be normally distributed (i.e. the process is Gaussian if all joint distributions are multivariate normal). Put into more mathematically explicit terms, for any given set of inputs $\{\mathbf{x}_1,\ldots, \mathbf{x}_n\}$, the resultant random variables $\{f(\mathbf{x}_1),\ldots, f(\mathbf{x}_n)\}$ have an n-dimensional Gaussian distribution:

$$p(f(\mathbf{x}_1),\ldots, f(\mathbf{x}_n)|\mathbf{x}_1,\ldots, \mathbf{x}_n) = \text{Normal}(\mathbf{m},\boldsymbol{\Sigma}) \tag{3.7}$$

where $\mathbf{m}$ is the *n* x 1 vector of expected values (or means) and $\boldsymbol{\Sigma}$ is the *n* x *n* matrix of covariances between all pairs of points. The covariance can be interpreted as a measure of how much two variables vary together (since variance is a measure of how much a single variable varies). Two independent variables are defined as having a covariance of zero; therefore two random variables whose covariance is zero are defined as uncorrelated. For two variables that show a tendency to vary together (i.e. they can be seen to display a degree of correlation such that both variables are found to be above an expected value), a positive covariance will result. Alternatively, if two variables are found to vary in an opposing trend, the covariance should be negative.

The Gaussian or Normal distribution can be characterised by the following Gaussian 'Bell' probability density function:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.8}$$

with mean μ corresponding to the expected value of the random variable, and variance σ^2 describing the statistical dispersion around the expected value, and commonly interpreted as the width of the probability distribution. Variance measures are also commonly converted to standard deviations and presented as error bars, with one standard deviation (σ) of a standard Gaussian distribution corresponding to a 68% confidence interval, and 2σ as a 95% confidence interval.

Given the properties of the Gaussian or normal distribution, one can fully specify a Gaussian process solely through its mean and covariance function:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), C(\mathbf{x}_i,\mathbf{x}_j)) \tag{3.9}$$

with mean function $m(\mathbf{x}) = E[f(\mathbf{x})]$, and covariance function $C(\mathbf{x}_i,\mathbf{x}_j) = \mathrm{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)]$. In a probabilistic framework this may also be written as

$$\begin{aligned} m_i &= E[f(\mathbf{x}_i)|\mathbf{x}_i] \\ \Sigma_{ij} &= \mathrm{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)|\, \mathbf{x}_i,\mathbf{x}_j] = E[f(\mathbf{x}_i), f(\mathbf{x}_j)|\, \mathbf{x}_i,\mathbf{x}_j] - E[f(\mathbf{x}_i)|\mathbf{x}_j]E[f(\mathbf{x}_j)|\mathbf{x}_j] \end{aligned} \tag{3.10}$$

A covariance matrix Σ_{ij} can then be generated from evaluating the covariance function given all the pairs of recorded data. Note also that given the assumed form of $y = f(x)$ between inputs x and outputs y, the covariance between inputs $\Sigma_{ij} = \mathrm{Cov}(x_i,x_j) = C(x_i,x_j)$, is also equal to the covariance evaluated at the corresponding outputs $C(y_i,y_j)$, where C(.,.) is some covariance function. The covariance matrix Σ_{ij} can then be defined as:

$$\Sigma_{ij} = \begin{bmatrix} C(x_1,x_1) & \dots & C(x_1,x_n) \\ \dots & \dots & \dots \\ C(x_n,x_1) & \dots & C(x_n,x_n) \end{bmatrix} \tag{3.11}$$

The overall result of adopting these mathematical constructs is that we can calculate the properties of the Gaussian Process at any finite number of points, and receive the same answer as if we were to calculate all the points of the process. This quality of the Gaussian process is sometimes known as the **marginalisation property**, and it allows us to overcome the perceived computational impossibilities associated with handling infinite dimensional random objects. This marginalisation property is also interpreted as a consistency requirement that must be fulfilled through the use of a covariance function to specify each entry of the covariance matrix, (i.e. if we are to employ a function that specifies the entries of an inverse covariance matrix then this property is no longer satisfied).

Furthermore, from this description of the Gaussian Process, we can see that the covariance function plays a fundamental role in how the resultant Gaussian Process will be specified. It is this function that generates the covariance matrix, and therefore influences how inputs and outputs are to be correlated with each other. The user must select the covariance function used by the GP model and therefore the choice represents a significant design control over the resultant model. More information on different covariance functions and how a suitable function may be identified will be given in the next chapter. However, from a purely mathematical perspective, any function that results in a positive semi-definite covariance matrix may be seen to be a valid choice of covariance function.

In conclusion, due to the complex random mathematical objects that make up the building blocks of the GP model, the descriptive terms used to categorise the approach (e.g. nonparametric, random variables, stochastic processes, …etc.) can seem rather ambiguous, impenetrable and by their very nature imprecise, especially for those uninitiated with probabilistic analysis. However, it is important to remember that whilst the components parts of the GP model may be seen to be complex random mathematical objects, they may be precisely expressed with interpretable mathematical functions such as Normal distributions and covariance functions.

3.6.2.) From Infinite Networks to Gaussian Processes

As mentioned previously, much of the interest in Gaussian Process models was initiated by the demonstration of a level of equivalence between Gaussian processes and Bayesian neural networks composed of an infinite number of fixed basis functions, assuming suitable priors. In the previous section, we have closely defined the mathematical particulars of the Gaussian process, but the relationship to infinite neural networks is worth exploring in order to complete the picture. From this point, the task of performing regression with Gaussian process models becomes more interpretable. For this section the work of Neal (1996) is referred to again and use is also made of much of the same theory and notation found in Gibbs (1997) and Mackay (1998b).

3.6.2.1) Defining Fixed-Basis Function Model

Restating the learning problem to be solved, we are given a set of N training data-points $(\mathbf{X}_N, \mathbf{t}_N) = \{\mathbf{x}^{(n)}, t_n\}$, composed of inputs $\mathbf{x}$ that are vectors of some fixed input dimension $\boldsymbol{I}$, and corresponding outputs or targets t_N. Our task is to infer an unknown function $y(x)$ assumed to be well represented by the data, and then seek to calculate predictions of new targets t^{N+1} given a new observed input $\mathbf{x}^{N+1}$.

Adopting a parametric approach to the modelling task we aim to approximate the unknown function $y(\mathbf{x})$, by a nonlinear function $y(\mathbf{x}; \mathbf{w})$ that may be characterised by parameters $\mathbf{w}$. If we now choose to adopt a network of H fixed basis functions $\{\phi_h(\mathbf{x})\}_{h=1}^{H}$ as our model structure, then we can specify the model as

$$y(\mathbf{x};\mathbf{w}) = \sum_{h=1}^{H} w_h \phi_h(\mathbf{x}) \tag{3.12}$$

Notice that by adopting this structure, the dependence between the output y and the parameters w is linear. This is an important point, as we are seeking to identify a nonlinear function $y(\mathbf{x}; \mathbf{w})$, but we have specified that the relationship between the unknown function y and the unknown parameters $\mathbf{w}$ is linear. However, if we specify basis functions that are nonlinear functions of $\mathbf{x}$, then the overall model $y(\mathbf{x}; \mathbf{w})$ is said to

be nonlinear. If we therefore select the nonlinear radial basis function centred at fixed points $\{c_h\}_{h=1}^{H}$, we can define

$$\phi_h(\mathbf{x}) = \exp\left[-\frac{(\mathbf{x}-\boldsymbol{c}_h)^2}{2r^2}\right] \tag{3.13}$$

For notational convenience, we can define a matrix **R** (*N* x *H*) to represent the values of the basis functions $\{\phi_h(x)\}_{h=1}^{H}$ at the points $\{\mathrm{x}^{(\mathrm{n})}\}$. We can then define the vector $\mathbf{y}_\mathrm{N}$ to be the vector of values of *y*(**x**) at the *N* points.

$$\begin{aligned} R_{nh} &\equiv \phi_h(\mathbf{x}^{(n)}) \\ y_n &\equiv \sum_h R_{nh} w_h \end{aligned} \tag{3.14}$$

From an overall perspective we can therefore interpret this model as being a multilayer network where only the output weights **w** are adaptive. The inner connections between the input layer and the hidden layer are fixed.

3.6.2.2) Define (Zero-Mean) Prior

In keeping with the Bayesian approach, after defining this initial model structure, we must now define a prior probability distribution over the unknown parameters **w** of this model. In the absence of any data, a possible choice of prior could be a Gaussian distribution of zero mean.

$$P(\mathbf{w}) = \mathrm{Normal}(\mathbf{0}, \sigma_w^2 \boldsymbol{I}) \tag{3.15}$$

As we have defined *y* as being a linear function of **w**, we can therefore deduce that *y* will also be Gaussian distributed, with a mean, E[**y**], of zero. The covariance matrix **Q** of y can then be defined as

$$\begin{aligned}\mathbf{Q} &= \mathrm{E}[(\mathbf{y} - \mathrm{E}[\mathbf{y}])(\mathbf{y} - \mathrm{E}[\mathbf{y}])^{\mathbf{T}}] \\ \mathbf{Q} &= \left\langle \boldsymbol{y}\boldsymbol{y}^{\boldsymbol{T}} \right\rangle = \left\langle \mathbf{R}\mathbf{w}\mathbf{w}^{\mathbf{T}}\mathbf{R}^{\mathbf{T}} \right\rangle = \mathbf{R}\left\langle \mathbf{w}\mathbf{w}^{\mathbf{T}} \right\rangle \mathbf{R}^{\mathbf{T}} \\ \mathbf{Q} &= \sigma_{\mathrm{w}}^{2}\mathbf{R}\mathbf{R}^{\mathbf{T}}\end{aligned} \tag{3.16}$$

Giving the prior distribution of **y** as:

$$P(\mathbf{y}) = \mathrm{Normal}(\mathbf{0}, \mathbf{Q}) = \mathrm{Normal}(\mathbf{0}, \sigma_w^2 \mathbf{R}\mathbf{R}^{\mathrm{T}}) \tag{3.17}$$

Therefore, for any selected number of points $\mathbf{X}_{\mathrm{N}}$, the vector of function values **y** will always have a Gaussian distribution. As a result of assuming a Gaussian zero-mean prior, we have therefore recreated the defining property of the Gaussian process in that a probability distribution of a function y(x) is a Gaussian process for any finite selection of points $\{\mathbf{x}^{(\mathrm{n})}\}$, the probability density $\mathrm{P}(y(x^{(1)}), y(x^{(2)}), \ldots, y(x^{(\mathrm{n})}))$ is also Gaussian.

Looking more closely at the covariance matrix **Q**, the individual (*n*, *n'*) element of **Q** is

$$Q_{nn'} = \left[\sigma_w^2 \mathrm{R}\mathrm{R}^{\mathrm{T}} \right]_{nn'} = \sigma_w^2 \sum_h \phi_h(\boldsymbol{x}^{(n)}) \phi_h(\boldsymbol{x}^{(n')}) \tag{3.18}$$

The covariance matrix **Q** describes the covariances of the function values at locations $\mathbf{X}_{\mathrm{N}}$, but we must also describe the covariance at the output or target values $\mathrm{t_n}$. Assuming each target $\mathrm{t_n}$ differs from the corresponding function value by Gaussian distributed additive noise σ_v^2, then the targets will also have a Gaussian prior distribution

$$P(\mathbf{t}) = \mathrm{Normal}(\mathbf{0}, \mathbf{Q} + \sigma_v^2 \boldsymbol{I}) \tag{3.19}$$

Therefore, denoting the covariance matrix of the targets t by C:

$$\mathbf{C} = \mathbf{Q} + \sigma_v^2 \boldsymbol{I} = \sigma_w^2 \mathbf{R}\mathbf{R}^{\mathbf{T}} + \sigma_v^2 \boldsymbol{I} \tag{3.20}$$

Looking more closely at the covariance matrix **C**, the individual (*n*, *n'*) element of **C** may be shown to be

$$Q_{nn'} = \sigma_w^2 \sum_h \phi_h(\boldsymbol{x}^{(n)})\phi_h(\boldsymbol{x}^{(n')}) + \delta_{nn'} \tag{3.21}$$

3.6.2.3) Move to Infinite Basis Functions

If we now consider the model based upon an infinite rather than finite number of basis functions ($H \to \infty$), the summation over the basis function at (*n*, *n'*) becomes an integral. We can simplify the form of $\mathbf{Q}_{nn'}$ by assuming that the basis functions are to be uniformly spaced, with each basis function *h* centred on the point $x = h$. Additionally, the variance component σ_w^2 can be scaled so as not to diverge with the increasing *H*, by redefining it as a constant *S* dependent on the number of basis functions per unit length of the *x*-axis.

$$Q_{nn'} = S \int_{h_{\min}}^{h_{\max}} \phi_h(\boldsymbol{x}^{(n)})\phi_h(\boldsymbol{x}^{(n')})dh \tag{3.22}$$

$$Q_{nn'} = S \int_{h_{\min}}^{h_{\max}} \exp\left[\frac{(x^{(n)} - h)}{2r^2}\right] \exp\left[\frac{(x^{(n')} - h)}{2r^2}\right] dh \tag{3.23}$$

Setting the limits of integration to ± ∞, this integral becomes:

$$Q_{nn'} = \sqrt{\pi r^2}\, S \exp\left[\frac{(x^{(n')} - x^{(n)})}{4r^2}\right] \tag{3.24}$$

From this expression describing the individual entry (*n*,*n'*) of covariance matrix ***Q***, we can generalise to form a covariance function describing all entries with the constant terms grouped together to form the hyperparameter θ_1.

$$C(x^{(n)}, x^{(n')}) \equiv \theta_1 \exp\left[\frac{(x^{(n')} - x^{(n)})}{4r^2}\right] \tag{3.25}$$

Therefore, for any valid covariance function, we can define the covariance matrix **Q** for *N* function values at locations $\mathbf{X}_N$ to be:

$$\boldsymbol{Q} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) \tag{3.26}$$

And assuming Gaussian additive noise, the covariance matrix **C** for the corresponding *N* target values is:

$$\mathbf{C} = C(\mathbf{x}^{(n)}, \mathbf{x}^{(n')}) + \sigma_v^2 \delta_{nn'} \tag{3.27}$$

3.6.2.4) Where does this leave us?

The consequence of increasing the number of fixed basis functions of the model towards infinity is that effectively a prior may now be defined using a covariance function that is equivalent to the prior specified in terms of basis functions and priors over the parameters. In essence we have simplified the process of defining a suitable prior as instead of defining individual priors over the type of model or function, parameter values, and noise beliefs, we can combine everything through defining a **Gaussian Process Prior** that may be specified through the choice of covariance function C.

Furthermore, as well as making the task of defining priors easier, the subsequent steps required to complete the inference and obtain predictions also become easier to implement. Previously, the presence of intractable integrals made the evaluation of different components (particularly the marginal likelihood) of the Bayesian framework difficult to achieve without the use of iterative methods. Through the use of the Gaussian process, these integrals become tractable and can therefore be treated analytically.

3.7) Regression with Gaussian Processes

The previous sections have provided an account of how we can use a Gaussian process to represent our prior beliefs for Bayesian learning. By using a Gaussian process prior, we can alleviate much of the difficulty of identifying suitable priors over the type of functions and parameters if a more conventional parametric structure were to be employed. Therefore, to employ a Gaussian process as our prior we must specify its defining characteristics, namely the mean of the process, and a covariance matrix **C** that reflects the correlations found in the training data set.

A common choice for GP models is to specify a zero-mean Gaussian process as our prior. In reality, the choice of a zero-mean Gaussian process may not be particularly representative of the underlying data, but this is not seen as a drastic limitation as our posterior process is not confined to zero. For the specification of the covariance matrix **C**, a suitable covariance function C(**x**, **x**') must be applied to the training data. Various choices of covariance functions are possible (see Section 4.3), and the most appropriate candidate must be selected on the basis of prior knowledge. As a result, the choice of covariance function is fundamental to the GP modelling approach as it dictates how the data is to be transformed into a matrix that reflects their correlations.

After selecting an appropriate covariance function, the parameters $\boldsymbol{\theta}$ of this function (known as hyperparameters due to the similar role played by the upper-level parameters of neural network approaches) must be identified from the training data in order to optimise this covariance function. This process is to be discussed in the next chapter, but for the moment we can assume that an optimised covariance function has been identified and that our Gaussian process prior has been specified. At this point, we wish to discuss how predictions can be made using this Gaussian process prior.

3.7.1) Making Predictions

To initiate Bayesian learning, after defining the prior of our Bayesian model, the next stage is to infer the posterior and ultimately make predictions given a new input $\mathbf{x}_{N+1}$. However, due to the nature of our prior, instead of following the procedure of the Bayesian approach outlined previously, we can also bypass some of these steps. In the Bayesian framework outlined, the inference of a posterior distribution with which to make predictions relied upon the application of Bayes' theorem. However due to the nature of our prior we have in effect explicitly stated the probability of the data (i.e the marginal likelihood) in one step. Therefore, we can obtain the predictive distribution of t_{N+1} from the straightforward application of **conditional probability** instead of applying Bayes theorem. As a result, it could be said that the GP modelling approach is not truly Bayesian due to the absence of Bayes' theorem. In fact, the prior placed upon the space of functions comes from the very probabilistic nature of the model (a Gaussian process being a random function) instead through the explicit use of Bayes' formula.

Restating the regression problem, we are given a set of *N* training data points $\boldsymbol{D} = (\mathbf{X}_N, \mathbf{t}_N)$, composed of inputs **x** that are vectors of some fixed input dimension (i.e. multiple inputs), and their corresponding scalar outputs or target values **t** (single output):

Inputs: $$\mathbf{X}_N = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \tag{3.28}$$

Targets: $$\mathbf{t}_N = [t(\mathbf{x}_1), t(\mathbf{x}_2), \ldots, t(\mathbf{x}_N)] \tag{3.29}$$

The regression task at hand is to predict a new output or target t_{N+1}, given the new input $\mathbf{x}_{N+1}$. However, due to the probabilistic nature of the GP modelling approach, the regression process will involve the computation of a posterior probability distribution over t_{N+1}, and subsequent determination of a singular prediction estimate $\hat{t}_{N+1}$ based upon the mean of this distribution.

Utilising the GP modelling approach, we are to specify a Gaussian process prior distribution over the space of functions. As discussed previously, the GP prior is a collection of random variables that are assumed to have a **joint** multivariate Gaussian distribution, thus allowing it to be fully specified by its mean **μ** and covariance matrix **C**. As the Gaussian process is a collection of random variables, we can explicitly state the probability of the target data as the joint distribution

$$P(\mathbf{t} \mid \mathbf{C}, \mathbf{X}_N) = \prod_{i=1}^{n} P(t_i \mid C(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta}), \{\mathbf{x}_n\}) \tag{3.30}$$

The probability of the target data is conditional on the covariance matrix and input data, $P(\mathbf{t} \mid \mathbf{C}, \mathbf{X}_N)$. This joint distribution can then be rewritten as

$$P(\mathbf{t} \mid \mathbf{C}, \mathbf{X}_N) = \frac{1}{Z} \exp\left(-\frac{1}{2} (\mathbf{t} - \mu)^T \mathbf{C}^{-1} (\mathbf{t} - \mu) \right) \tag{3.31}$$

where **C** is the covariance matrix defined by the parameterised covariance function $C(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta})$ applied to the input data, μ is the mean of the process, and **Z** is a normalising constant. Furthermore, we can realise the prior belief regarding the Gaussian process having zero-mean, $\mu_N = 0$, allowing us to rewrite the Gaussian process prior as

$$P(\mathbf{t}_N \mid \mathbf{C}_N, \mathbf{X}_N) = \frac{1}{Z_N} \exp\left(-\frac{1}{2}(\mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N)\right) \tag{3.32}$$

The regression task is now to infer the prediction t_{N+1}, using this Gaussian process prior and a new input observation $\mathbf{x}_{N+1}$. Therefore the predictive distribution we desire can be interpreted as a conditional probability distribution over t_{N+1}, written as $P(t_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1})$, where $\boldsymbol{D} = (\mathbf{X}_N, \mathbf{t}_N)$.

If we briefly adopt a more concise notational form, (where our prior $P(\mathbf{t}_N \mid \mathbf{C}_N, \mathbf{X}_N)$ is denoted as $P(\mathbf{t}_N)$, and the desired posterior $P(t_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1})$ is denoted $P(t_{N+1})$) we can interpret this conditional probability as

$$P(t_{N+1} \mid \mathbf{t}_N) = \frac{P(t_{N+1}, \mathbf{t}_N)}{P(\mathbf{t}_N)} \tag{3.33}$$

Therefore we must find the joint probability $P(t_{N+1}, t_N)$ of the new input and the existing prior probability. Reverting back to the previous notation, we can restate this conditional probability as

$$P(t_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1}) = \frac{P(\mathbf{t}_{N+1} \mid \mathbf{C}_N, \mathbf{X}_N, \mathbf{x}_{N+1})}{P(\mathbf{t}_N \mid \mathbf{C}_N, \mathbf{X}_N)} \tag{3.34}$$

where the conditional probability $P(\mathbf{t}_{N+1} \mid \mathbf{C}_N, \mathbf{X}_N, \mathbf{x}_{N+1})$ in the numerator of this expression is equivalent to the joint distribution $P(t_{N+1}, t_N)$. Therefore, before finding the conditional distribution $P(\mathbf{t}_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1})$, we must first define the joint distribution $P(\mathbf{t}_{N+1} \mid \mathbf{C}_N, \mathbf{X}_N, \mathbf{x}_{N+1})$. This can be done by treating the new input observation as a continuation of our Gaussian process, where we can apply the covariance function to the new observation and therefore update the covariance matrix from $\mathbf{C}_N$ to $\mathbf{C}_{N+1}$, giving the joint distribution:

$$P(\mathbf{t}_{N+1} \mid \mathbf{C}_N, \mathbf{X}_N, \mathbf{x}_{N+1}) = \frac{1}{Z_{N+1}} \exp\left(-\frac{1}{2}(\mathbf{t}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{t}_{N+1})\right) \tag{3.35}$$

At this stage we can introduce the mechanics of how the covariance matrix $\mathbf{C}_{N+1}$, is to be updated through the introduction of a new input. From the training set $\mathbf{X}_N$, the covariance matrix $\mathbf{C}_N$ was determined. For predictions, a new covariance matrix (incorporating the new input data) $\mathbf{C}_{N+1}$ of size $(N+1) \times (N+1)$ is obtained from

$$\mathbf{C}_{N+1} = \begin{bmatrix} [\mathbf{C}_N] & [\mathbf{k}] \\ [\mathbf{k}^T] & [\kappa] \end{bmatrix} \tag{3.36}$$

Where the sub-matrix $\mathbf{k} = \left[\mathrm{C}\left(\mathbf{x}_{1,}\mathbf{x}_{N+1};\boldsymbol{\theta}\right),\ldots,\mathrm{C}\left(\mathbf{x}_{N,}\mathbf{x}_{N+1};\boldsymbol{\theta}\right) \right]$ is the vector of covariances between the new test point and existing training cases, and $\kappa = \mathrm{C}\left(\mathbf{x}_{N+1,}\mathbf{x}_{N+1};\boldsymbol{\theta}\right)$ is the variance of the individual test case.

Returning to the task of obtaining the conditional probability $P(\mathbf{t}_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1})$, remembering the relationship that $\frac{e^{x_1}}{e^{x_2}} = e^{x_1 - x_2}$, we can write

$$P(\mathbf{t}_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1}) = \left(\frac{1}{Z_{N+1}} . \frac{Z_N}{1} \right) \exp\left(\left(-\frac{1}{2}(\mathbf{t}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{t}_{N+1}) \right) - \left(-\frac{1}{2}(\mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N) \right) \right) \tag{3.37}$$

$$P(\mathbf{t}_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1}) = \frac{Z_N}{Z_{N+1}} \exp\left(-\frac{1}{2}(\mathbf{t}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{t}_{N+1} - \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N) \right) \tag{3.38}$$

At this point we can see that in order to compute this posterior distribution to find the probability of the new target t_{N+1}, we would need to already have observed t_{N+1}, i.e. the very quantity which we are hoping to predict. Thankfully, through the use of some mathematical substitution made possible by using partitioned inverse equations, see Bartnett (1979), we can move forward. These partitioned inverse equations allow the specification of $\mathbf{C}_{N+1}^{-1}$ in terms of $\mathbf{C}_N$ and $\mathbf{C}_N^{-1}$, and thereby allow us to effectively implement a model where the number of basis functions may be much larger than the number of observed data points N. This can be seen to be a computational saving (the matrix inversion is $N \times N$, rather than $(N+1) \times (N+1)$) as we can in effect fix the computational demand to that of $O(N^3)$.

The precise details of this mathematical manipulation and the derivation of the GP predictive equations have been omitted here for the sake of thesis brevity, but may be found in Appendix B. Utilising these expressions we can specify the posterior probability of in terms of $\mathbf{C}_N$ and $\mathbf{k}$, thereby allowing computation. The result of this process is that we can define the posterior distribution in a readily interpretable Gaussian form:

$$P(t_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, x_{N+1}) = \frac{1}{Z} \exp\left(-\frac{(t_{N+1} - \hat{t}_{N+1})}{2\sigma^2_{\hat{t}_{N+1}}} \right) \tag{3.39}$$

where the mean and variance are defined as:

$$\hat{t}_{N+1} = \mathbf{k}^T_{N+1} \mathbf{C}_N^{-1} \mathbf{t}_N \tag{3.40}$$

$$\sigma^2_{N+1} = \kappa - \mathbf{k}^T_{N+1} \mathbf{C}_N^{-1} \mathbf{k}_{N+1} \tag{3.41}$$

Therefore, we have arrived at the point where a prediction of the output $\hat{t}_{N+1}$ has been obtained, together with the uncertainty over that prediction that may be interpreted using error bars.

In the next section a more visual interpretation of the GP modelling approach is provided, however it is worth pointing that an alternative '**weight-space**' mathematical interpretation of the GP framework has also been developed, see Rasmussen and Williams (2006). In this chapter the 'function-space' interpretation of the GP method has been presented. Through the weight-space-view, the GP modelling approach can be seen to be equivalent to the Bayesian linear regression. As the two interpretations can be seen to result in the same predictive framework, the function-space viewpoint has been preferred solely due to being more intuitive in my experience.

3.8) Demonstration of Gaussian Process Modelling

To provide a better understanding of the GP modelling approach, a simple one-dimensional regression problem is considered, with input x and output f(x). This example is aimed at providing a more visual interpretation of the method.

3.8.1) Defining a Gaussian Process Prior

The first step in any application of Bayesian inference is to define a Prior distribution over the kinds of function we expect to observe before any data is presented. For a Gaussian Process model, we employ a Gaussian Process to define our Prior to represent our prior beliefs over the underlying function. For this example, a zero mean Gaussian process with a point-wise variance $N(0,1)$ has been taken as our Prior. In Figure (3.2), we can see the assumed space where we believe the function is to exist on a chart of the output y (= f(x)) versus input x.

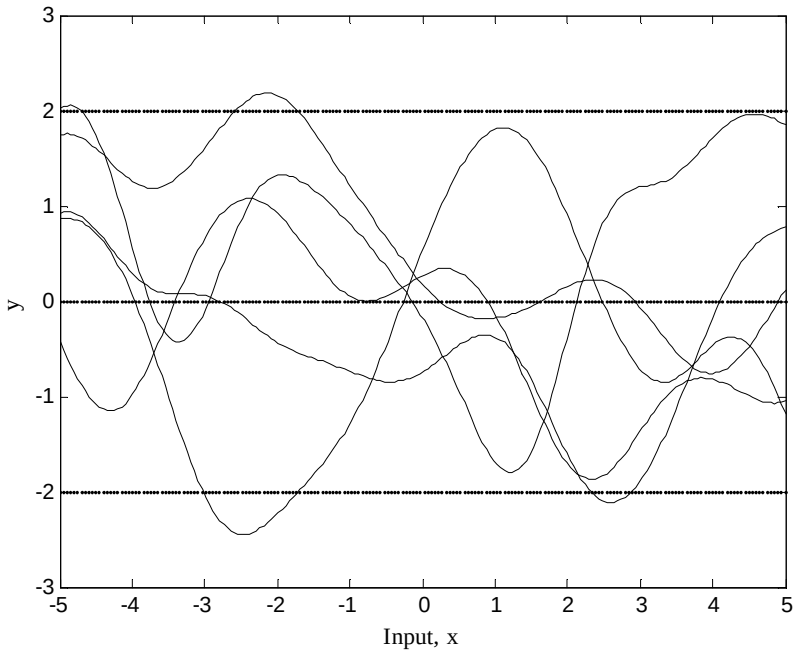


Figure (3.2): Prior Distribution over functions with 5 sample functions

A number of sample functions from the Gaussian Process have also been drawn onto Figure (3.2) to show a number of functions that could possibly be close to what we are searching for. Note that these sample functions are just possibilities drawn at random,

and do not involve prior knowledge or represent particularly likely candidates that are to be corrected in the next step. These sample functions merely provide a visualisation as to the variety of possible functions that exist over our function space. The dotted lines at y = -2, 0, and +2, can be seen to represent the Prior where the average value of the sample functions at each x is zero (at this point in the process we have no data to assume otherwise), together with 2 times the point-wise variance (i.e. 2σ = +/- 2) that we have used as an indicator as to the variability of the sample functions. Furthermore the Gaussian process used has specified a prior variance that is independent of the input x.

In Figure (3.2), the sample functions are drawn at random from the Prior distribution over functions. A further assumption has been introduced that implies that the underlying function will vary in a smooth manner. The samples shown in Figure (3.2) are all characteristically similar in that they have been drawn from a Gaussian process defined from the same Covariance function with identical hyperparameters. A random element has been introduced to show a few different possible functions based on the same Covariance function and hyperparameters. More information on different covariance functions and the influence of their hyperparameters is to follow in the next section. Again, these are sample functions drawn from the Prior distribution over functions for the purposes of visualisation, not the Prior itself. Normally, we are not interested in generating random samples from the prior, but in generating a posterior and then making predictions.

3.8.2) Compute Posterior

Following the template for Bayesian inference, the next stage is to compute the Posterior distribution. In order to make this happen, we must have some observed data to combine with our Prior distribution over functions. Therefore consider that we are now given a small dataset comprised of 4 pairs of input-output data $D = \{(x_1,y_1), \ldots, (x_4,y_4)\}$, as shown in Figure (3.3).

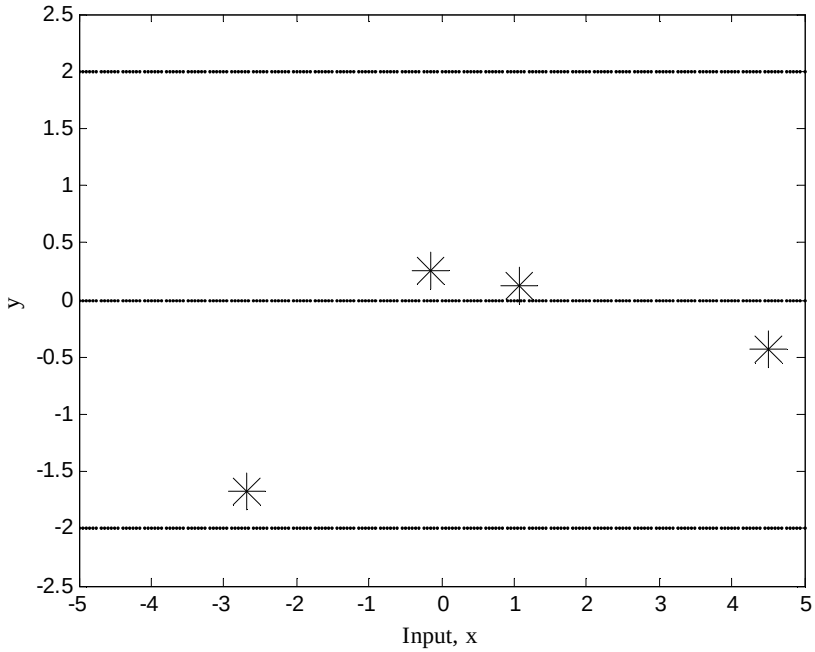


Figure (3.3): Observed Data (4 data points)

Visually, the idea is to move from our Prior space over every possible function, to now considering functions that pass through or close to the observed data. We are not 'correcting' the previous sample functions of Figure (3.2), but identifying new functions from our function space that are consistent with the observed data.

In Figure (3.3) the observed dataset is displayed as point values, and in Figure (3.4) the Posterior distribution over functions is displayed. In Figure (3.4), a number of possible functions (in dashed lines) consistent with the observed data are displayed, together with the predicted mean of the posterior distribution (solid line) that is normally taken as the overall prediction estimate of the GP model. Figure (3.4) also displays error bars showing twice the standard deviation (2σ) that provide an estimate as to the uncertainty of the predicted mean relative to the input x. From these error bars we can see that the variance of the prediction is markedly reduced close to the observed values, indicating that we are more certain in these regions and therefore more confident in our model's accuracy.

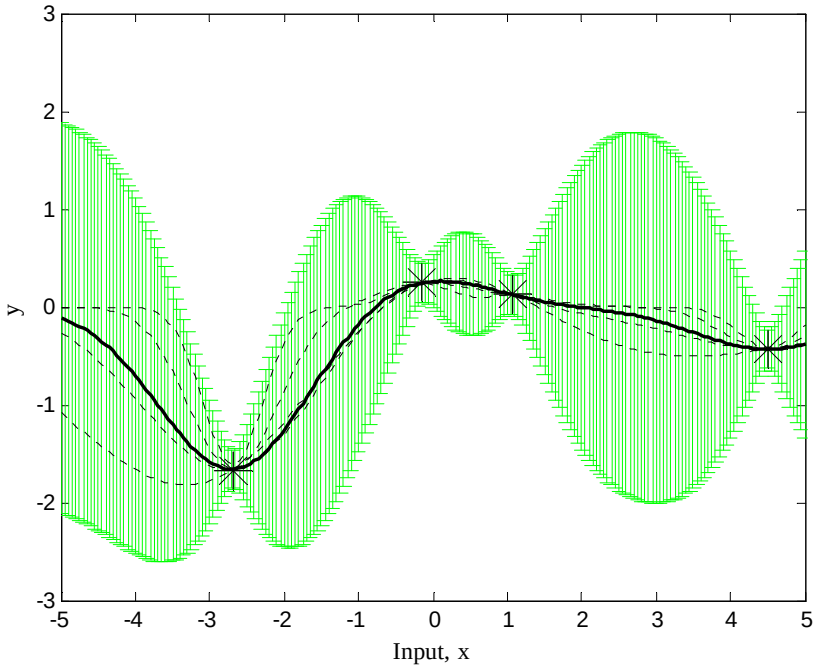


Figure (3.4): Posterior Distribution with 4 sample functions (dashed lines) and Predicted Mean (solid line) & 2σ Errorbars

If the size of observed dataset is increased, as in Figure (3.5), we can see the mean of the posterior is further adjusted to remain consistent with the data, together with a further decrease in variance close to the observed values. Due to the increase in the number of data points, we have more evidence with which to analyse the correlations between the different data points. As a result, we can be more certain of our model's accuracy over a greater range of the input space. Furthermore, the regions where data is sparser become ever more pronounced through the analysis of the uncertainty/errorbars that is made readily possible in the GP modelling approach.

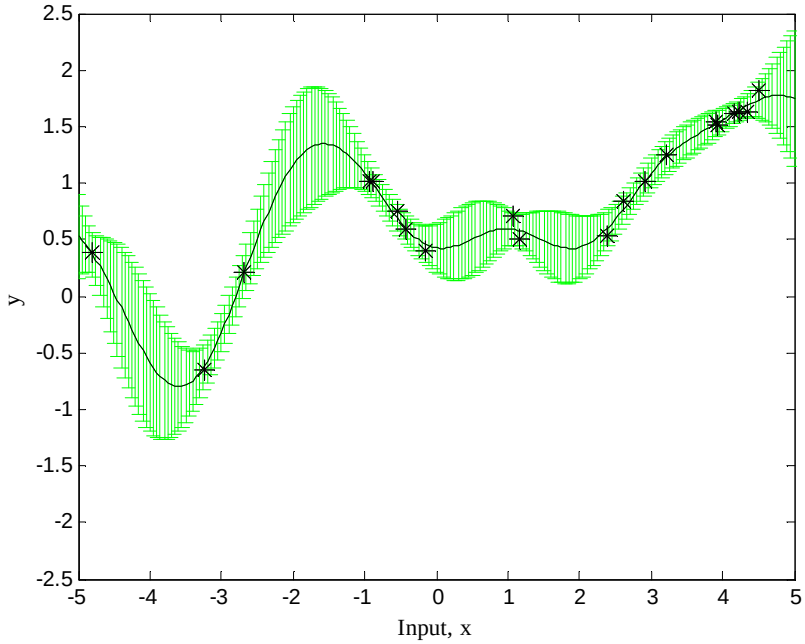


Figure (3.5): Posterior Distribution (20 data points) with Predicted Mean (solid line) & 2σ Errorbars

Note that in this demonstrative example, the choice of covariance function or hyperparameters has not been discussed. A reasonable fit to the underlying function was achieved primarily because the observed data points were random points that were themselves generated from a Gaussian process of closely matching hyperparameters. For the identification of unknown functions, a suitable covariance function composed of identified hyperparameters must be determined from the available training data. The details of this identification process are to be discussed in the next chapter.

4) Implementation of GP models

In the previous chapter the theoretical background to the GP modelling approach was discussed together with the mathematical framework necessary to facilitate nonlinear regression. In this chapter, we are to look more closely at the details of exactly how the GP method is to be implemented. Of fundamental importance to the GP modelling approach is the specification of a covariance function that allows us to represent successfully the correlations between different training data observations. Furthermore, in order to provide accurate predictions of system behaviour, the parameters of this covariance function must be optimised using the available observations. In practice, both of these objectives are subject to mathematical and computational difficulties relating to the size and conditioning of the covariance matrix. Methods aimed at tackling these implementation difficulties are then discussed in detail.

4.1) Role of the Covariance Function

As a Gaussian Process prior is specified by its mean and covariance matrix, the covariance function used to generate this covariance matrix will therefore play a fundamental role in the GP modelling approach. Consequently, the covariance function must be chosen to reflect our prior assumptions about the function or system we wish to identify, and ultimately instil these assumptions into the covariance matrix. Through the selection of the covariance function, we are attempting to fix the properties of the functions that are to be considered for inference. As a result, this stage of selecting a covariance function is somewhat analogous to the selection of a parametric model structure. For an example, referring back to the previous demonstrative example, where the sample functions in Figure (3.2) are all smooth and stationary (informally, stationary means that the functions will look similar at all $\mathbf{x}$ locations). These are properties that are defined by the chosen covariance function of the GP; other covariance functions that exhibit other properties are possible.

Typically, a covariance function will be constructed out of a number of ‘free’ parameters $\boldsymbol{\theta}$ that may be used to adjust the properties of the Gaussian process prior. The parameters of the covariance function are more often referred to as **hyperparameters** due to the

similar role they play to that of the upper-level parameters of neural network structures. Therefore, the problem of supervised learning in GP models can be seen to be to the problem of identifying suitable hyperparameters for the covariance function. The hyperparameters of the covariance function $\boldsymbol{\theta}$ must therefore be identified from the observed training data. This learning process is examined in the next section, but before hyperparameters can be determined a suitable covariance function must first be selected. Furthermore, before discussing the nature of various covariance functions it is first worthwhile underlining a number of assumptions inherent in the underlying regression problem:

- As the covariance function is directly applied to the training data, in order for the resultant covariance matrix to reflect the nature of the correlations between inputs and outputs of the underlying function, the training data collected must therefore adequately reflect the characteristics of the underlying function. This is of course common sense, but the GP approach presents particular mathematical requirements that may cause difficulties in adhering to this principle.

- For most nonlinear regression problems, and especially those found in the identification of real systems, the observed data used for training is likely to have been corrupted by noise. As a result, a noise model would be an appropriate feature to incorporate into our chosen covariance function.

- An important assumption made in most supervised learning problems is that similar inputs should result in similar outputs. This assumption further manifests itself as an expectation that two data points close together in input space are likely to have a greater correlation than two points that are distant. Furthermore, as we assume that similar inputs are likely to result in similar target values, we can assume that training points (input and output pairs) near to a test point (input) should be informative about the desired prediction (output). Therefore this concept of nearness or similarity is something that all regression methods are founded upon. In the context of GP models, it is the covariance function that is to define the nearness or similarity of the individual data points.

Taking this final assumption concerning the nearness or similarity of individual data points, it is now useful to reiterate this concept in terms of covariance. Remember that the covariance measures the similarity between two random variables, where a high covariance is representative of two random variables that are closely related, and a low covariance is representative of those that are weakly related. Furthermore, leaving aside the mathematics used to define different covariance functions and the GP model as a whole, it is worthwhile first attempting provide a more descriptive interpretation of how the covariance function is to define the characteristics of the functions that are to be considered.

From a purely mechanical point of view, in order to identify our model we have a set of training cases of input and output data, but for prediction we will only have an input. Therefore, we must build our GP model so that given an input we can generate an appropriate output. The result of this is that we are not interested in the covariance between inputs and outputs, nor are we expressly concerned with the covariance between different inputs. Instead we are interested in relating the covariance between the inputs to that of the outputs, and this is achieved through the covariance function *C*. As a result, the covariance between the outputs or targets can be written as a function of the inputs.

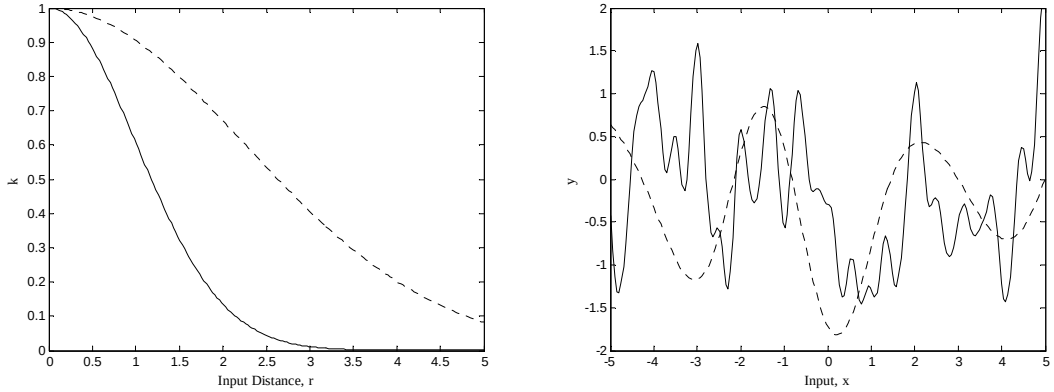
$$\mathrm{cov}(f(\mathbf{x}_m), f(\mathbf{x}_n)) = C(\mathbf{x}_m, \mathbf{x}_n) \tag{4.1}$$

Under our initial assumption, two data points that are close together in input space are to be informative about each other's respective targets, thus reflected in a high covariance. Similarly, for two distant points thought to be uninformative, the covariance is to be low. Note that this initial assumption may not necessarily be the case for all problems (e.g. periodic functions where relationships between relatively distant datapoints must be considered). Nevertheless, we can more closely define the role of the covariance function to be that of a model of how the covariance is to change as the distance between different inputs changes.

To demonstrate this visually we can display the relationship between the covariance *k* and the distance between inputs, $r = |\mathbf{x} - \mathbf{x}'|$, for two different covariance functions as in Figure 4.1(a). For both these covariance functions, the covariance approaches unity

between variables whose corresponding inputs are very close, and decreases as the input distance is increased. However, for the dashed-line example, the rate of decay in the covariance as the input distance is increased is much slower than that of solid-line example. Note that both of these covariance functions are in fact of the same squared-exponential form, see Section (4.3.1.1), but with different hyperparameters.

In combination with an assumed a zero-mean, the two defined covariance functions of Figure 4.1(a) can be thought to have each defined a Gaussian process prior. Therefore, we can gain a further understanding of the role played by the covariance function through drawing sample functions from each Gaussian process prior. In Figure 4.1(b) two sample functions have been drawn from the GP priors of the dashed and solid-line covariance functions of Figure 4.1(a). Immediately we can see that the more rapidly decaying covariance function of the solid line example results in a sample function that varies far more rapidly, thus resulting in a less smooth process.



(a) **(b)**

Figure (4.1): Chart (a) shows two different covariance functions where the covariance k varies against the input distance r. Chart (b) shows random functions drawn from the Gaussian process with the same covariance functions as Chart (a).

Relating the smoothness characteristics of the samples shown in Figure 4.1(b) to the covariance functions displayed in Figure 4.1(a), we can understand that for the covariance function that decays more rapidly as the distance in input space is increased, the degree of similarity between nearby inputs is to reduce more quickly. As a result, the random sample function generated from the prior will have the capacity (remember that we are specifying a space over functions rather than an actual function) to vary more

quickly, as each generated point of the sample function will have a weaker relationship to those points immediately preceding them. This relationship between the distance between inputs and the potential for the sample function to vary more rapidly (or become more 'wiggly') can be understood as the **characteristic length scale** of the process, and is one of the possible properties of the GP prior that we can control through the hyperparameters of a chosen covariance function.

Returning to our system identification remit, we have seen that through the manipulation of the covariance function, the length-scale property of the resultant functions can be modified. As a result, for an underlying system that is known to vary in a smooth manner, the covariance function and accompanying hyperparameters can be chosen to reflect this prior knowledge through manipulation of the length-scale property of the Gaussian process prior. Similarly, for systems that are known to vary in a less smooth or more abrupt manner, the covariance function and its hyperparameters can be altered to reflect this prior knowledge. Overall, in this simple example (where we have restricted the discussion to the length-scale property) we can see how the properties of our Gaussian process prior can be fixed through the choice of covariance function and its hyperparameters to suit the identification task at hand.

4.2) Choice of Covariance Functions

In order to identify a GP model, a suitable covariance function that reflects our prior knowledge of the underlying system must be selected. Furthermore, the process of selecting or defining a suitable covariance function can also be seen to be the process of constructing a valid Gaussian (stochastic) process. As a result, any arbitrary function cannot be chosen for use as a covariance function as the construction of stochastic processes places particular demands on the nature of this function.

4.2.1) Validity of Covariance Functions

In the selection of an appropriate covariance function, an important constraint exists over the validity of any possible function. This constraint states that the covariance function must generate a **positive semi-definite** (or non-negative definite) covariance matrix.

Therefore, any arbitrary function of input pairs x and x' will not in general be a valid covariance function. To more accurately define what is meant by a positive semi-definite matrix, consider a set of input points $\mathbf{x}_n$. We can apply our chosen covariance function, or more generally a 'kernel' to this data to generate the matrix $\mathbf{K} = k(\mathbf{x},\mathbf{x}')$. The matrix generated from the application of a kernel is known as the *Gram* matrix, where if k is a valid covariance function the matrix K can be termed a covariance matrix. Whilst a Gram matrix generated from a kernel function need not be positive semi-definite, a covariance matrix must adhere to this constraint.

Mathematically, a real $n \times n$ matrix K is said to be positive semi-definite (PSD) if it satisfies the condition $Q(\mathbf{v}) = \mathbf{v}^{\mathbf{T}}\mathbf{K}\mathbf{v} \geq 0$ for all vectors where $\mathbf{v} \in \mathbb{R}^n$, and Q is a quadratic form. This can possibly be better understood by stating that for a PSD matrix, the eigenvalues of the matrix must be non-negative. The positive definiteness of a matrix can also be described in terms of the sign of the determinant of the matrix. As the determinant is a scalar function of the matrix, where matrices are symmetric (as is the case for covariance matrices), the positive definiteness of the matrix will only remain if the matrix and every principal submatrix (formed by removing row-column pairs) have a positive determinant. A matrix that does not meet this condition is not positive definite.

4.2.1.1) Why does Positive-Definiteness matter?

The mathematical descriptions of positive definiteness do not provide any great deal of information as to why this constraint exists upon the choice of covariance function. Therefore, perhaps further comment on the reasoning behind this constraint would be worthwhile, as it offers further insight into how the GP modelling approach actually works.

Fundamentally, the requirement for positive semi-definiteness originates from the mathematics employed in the construction of stochastic processes. Remember that we are attempting to construct a stochastic (Gaussian) process with which to apply Bayesian inference. In general, the properties of a stochastic process or random field may be described by a set of finite-dimensional distributions. For a Gaussian process or Gaussian random field, these finite-dimensional distributions are multivariate normal distributions. This property allows them to be specified by a mean and covariance as described in the

previous chapter. For the specification of non-Gaussian random fields, the definition of finite-dimensional distributions is not as straightforward, hence the appeal of Gaussian processes.

These finite-dimensional distributions $F_{\mathbf{t}_1,\ldots,\mathbf{t}_k}$ can therefore be defined at any available training data points or coordinates $\{\mathbf{t}_1,\ldots,\mathbf{t}_k\}$, and then utilised to construct our stochastic (Gaussian) process. Note that these finite-dimensional distributions are cumulative distributions, $F_{\mathbf{t}_1,\ldots,\mathbf{t}_k}(x_1,\ldots,x_k) = \mathrm{P}\{X_{\mathbf{t}_1} \leq x_1,\ldots,X_{\mathbf{t}k} \leq x_k\}$, and must therefore be right-continuous and non-decreasing. From this system of finite-dimensional distributions, a valid random field or stochastic process is said to exist if certain symmetric and compatibility conditions are met. This is known as Kolmogorov's Existence theorem (also known as Kolmogorov's Extension). Taken together, the conditions concerning symmetry and compatibility can be regarded as requirements for consistency over the finite-dimensional distributions. Therefore, the question is how do we ensure this consistency over the finite-dimensional distribution and therefore create a valid stochastic process. This is where the constraint for positive-definiteness comes into it:- a positive-definite covariance function will ensure a positive-definite covariance matrix, which in turn safeguards the existence of a valid Gaussian process.

4.2.2) Types of Covariance Function

In this chapter we are focusing on describing the properties of a number of existing covariance functions. Therefore, a full and precise account of the methodology involved in the construction of stochastic processes, and how covariance functions can be derived has not been included. This is a complicated area of probability/statistics that more detailed resources on this particular subject are better placed to cover. Therefore, I refer to the texts by Adler (1981), Billingsey (1986), Doob (1953), and Papoulis (1991). Fortunately a number of valid covariance functions have been already defined in the existing literature and identified as being particularly suitable for use in the GP modelling approach. In particular, reviews of different covariance functions can be found in Abrahamsen (1997), Stein (1999), Mackay (1998b), and Rasmussen and Williams (2006). These references also include detailed information as to how various covariance

functions can be derived. In the forthcoming sections, a number of these different covariance functions are to be discussed.

4.2.2.1) Stationary & Non-stationary Covariance Functions

In describing the properties of different covariance functions the most important distinction is whether or not the function may be described as stationary or non-stationary. Stationary covariance functions can be seen to be functions of $\mathbf{x}-\mathbf{x}'$ and are therefore said to be invariant to translations in the input space. Loosely, this means that sample functions drawn from a stationary GP prior will look or behave similarly at all **x** locations (i.e. the process does not depend on the location of the observer). For non-stationary covariance functions, this is not the case and sample functions may vary wildly in terms of variable smoothness over the input space. Furthermore, if a covariance function is a function of $|\mathbf{x}-\mathbf{x}'|$ then it may be described as **isotropic** and therefore invariant to all rigid motions. Therefore, for stationary isotropic covariance functions, the quantity *r* introduced as the 'input distance' at the start of this chapter can be more specifically defined as the Euclidean distance, $r=|\mathbf{x}-\mathbf{x}'|$.

At this point a possible parallel between stationary and non-stationary covariance functions and static and dynamic systems may become apparent. For dynamic systems where the output response is to vary significantly over the defined input space, it might be thought that a non-stationary covariance function would seem most appropriate. Nevertheless, stationary covariance functions are more commonly used for implementation and interpretability reasons. Furthermore, existing research has demonstrated that excellent models of dynamic systems may be identified using stationary GPs. However, one issue to consider arises if a stationary covariance function is adopted for a case in which the underlying system is prone to change its behaviour during operation (e.g. some systems may heat-up or cool-down influencing the response). As a result, in such a case the operating response of the system may not be seen to behave consistently across the input range and a non-stationary covariance function might be a better choice.

4.2.2.2) Smoothness Properties

A further consideration in the selection of a suitable covariance function is the resultant smoothness properties of the defined Gaussian process prior. In the opening section of this chapter, the role of the covariance function in influencing the characteristics of the sample functions drawn from the resultant GP prior was introduced. In particular, it is the smoothness characteristics of the resultant sample functions that can be seen to be a fundamental differentiator between different covariance functions. In order to describe the relative smoothness of functions, mathematicians often employ the terms 'continuity' and 'differentiability'. In simple terms, if a function that approaches an infinite gradient (i.e. vertical) it can be thought of as being discontinuous and non-differentiable at that point. The occurrence of such sample function characteristics can therefore be seen to be symptomatic of a rough or non-smooth Gaussian process.

Therefore, through the selection of a suitable covariance function we are endeavouring to select the appropriate smoothness properties characterised by the relative continuity/differentiability of the sample functions. However, relating the smoothness properties of sample functions to a chosen covariance function is not mathematically straightforward, and different properties known as the 'mean-square' (MS) continuity/differentiability are normally employed. MS properties are more easily derived and are directly related to the derivatives of the covariance function and moments of spectral distribution. Unfortunately, these MS properties are less interpretable than the sample function properties as we can more readily judge the nature of the sample function continuity visually, as in Figure (4.1b).

The difference between sample function continuity/differentiability and their MS counterparts is the level of continuity displayed. Sample function continuity is a much stronger property than mean-square continuity, as discontinuities can be allowed under the weaker MS properties. Therefore, in general, mean-square continuity does not imply sample path continuity, and vice versa. However, for Gaussian random fields such as the GP we are defining, mean-square continuity is a necessary and almost sufficient condition for continuous sample paths. Furthermore, a random field can be seen to be continuous in mean square at $\mathbf{x}*$, if and only if its covariance function $k(\mathbf{x},\mathbf{x}')$ is continuous at the point $\mathbf{x} = \mathbf{x}' = \mathbf{x}*$. For stationary covariance functions this can be

reduced to checking continuity at $k(\mathbf{0})$, where **0** signifies a vector of all zeros. As a result, it is the properties of the kernel around **0** that determine the smoothness properties of a stationary process. For a more in-depth discussion of the geometrical properties of stochastic processes, see Adler (1981), Stein (1999) and Abrahamsen (1997). The thesis by Paciorek (2003) also contains useful information with specific regard to smoothness properties in terms of sample function continuity rather than mean-square continuity

4.3) Examples of Covariance Functions

General Form of Covariance Function

In the application of the GP modelling approach to practical system identification problems, a degree of noise is likely to be present on the empirical data. Therefore, only noisy function values are typically available, i.e. $y = f(\mathbf{x}) + \varepsilon$. If the assumption that this noise is additive independent identically distributed Gaussian noise ε with variance σ_n^2, a general form for the covariance function can be stated as

$$\boldsymbol{C}_{mn} = C(\mathbf{x}_m, \mathbf{x}_n; \boldsymbol{\theta}) + \sigma_n^2 \delta_{mn} \tag{4.2}$$

where δ_{mn} is a Kronecker delta which is one if and only if $m = n$ and zero otherwise. Due to this independent noise assumption, in comparison to a noise-free implementation, a diagonal matrix is added (i.e. K(x,x) + $\sigma_n^2 I$). Other noise models where independence from the input is not assumed are also possible.

4.3.1) Stationary Covariance Functions

In Table (4.1) a number of stationary non-degenerate covariance functions have been given. Note that instead of displaying the general form (covariance function C with or without noise), we are to concentrate on discussing the properties of the different kernels *k*. Furthermore, the variable $r = |\mathbf{x} - \mathbf{x}'|$ is the input distance measure, and ℓ is the characteristic length-scale hyperparameter. The properties of these covariance functions are now to be discussed below.

Covariance Function	Expression
Squared Exponential	$k_{SE}(r) = \exp\left(-\frac{r^2}{2\ell^2}\right)$
Matérn	$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}r}{\ell}\right)^{\nu} K_{\nu}(\frac{\sqrt{2\nu}r}{\ell})$
Exponential	$k(r) = \exp\left(-\frac{r}{\ell}\right)$
γ-Exponential	$k(r) = \exp\left(-\left(\frac{r}{\ell}\right)^{\gamma}\right)$
Rational Quadratic	$k_{RQ}(r) = \left(1+\frac{r^2}{2\alpha\ell^2}\right)^{-\alpha}$

Table (4.1): Table of stationary covariance functions

4.3.1.1) Squared Exponential Covariance Function

The most widely adopted choice of covariance function found in the GP literature is the squared exponential shown in Table (4.1). This function generates a Gaussian distribution shape and can be seen to be equivalent to the radial basis functions used in other modelling approaches. The squared exponential covariance function is infinitely differentiable and therefore has mean-square derivatives of all orders. As a result, a GP defined by this covariance function will be very smooth in terms of the sample functions drawn from it. The squared exponential covariance function is often implemented in an anisotropic form where each input dimension (D) can be assigned a different hyperparameter $\{\ell_d\}$ to control the characteristic length-scale. This form of the squared exponential covariance function may be written as

$$C(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \theta_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{\ell_d^2}\right] + \theta_2 \qquad (4.3)$$

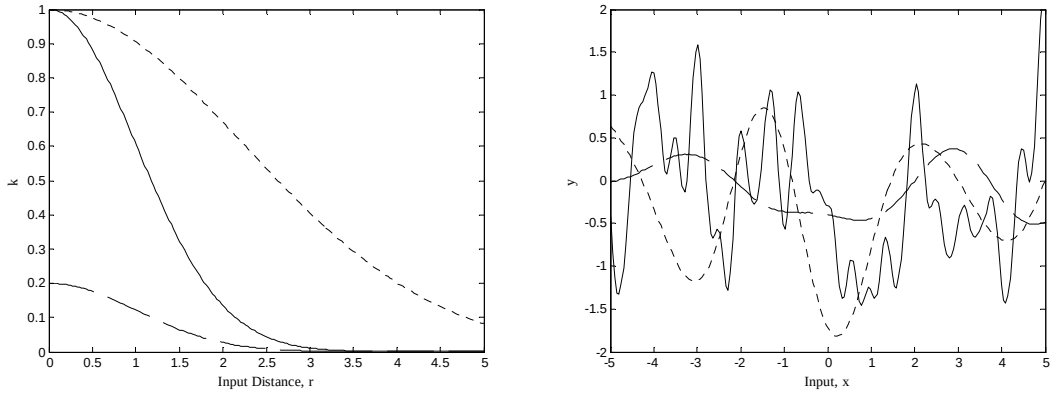
where x_d is the d^{th} component of $\mathbf{x}$, a D-dimensional vector, and hyperparameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \{\ell_d\})$. As this is the most popular covariance function, it is worth discussing in more

depth the role of these different hyperparameters. The θ_1 hyperparameter can be seen to define the vertical scale of the possible variations of the defined function, the θ_2, hyperparameter can be seen to be a bias term that allows the whole function to be offset from zero by some unknown amount. As mentioned above, a separate length-scale hyperparameter $\{\ell_d\}$ has been defined for each input dimension, and can be thought of as determining the distance in that particular direction over which the output *y* is expected to vary significantly. Therefore, if a particular input is to be given a very large length-scale hyperparameter, this input can be thought of as being irrelevant (or at least non-contributory) to the output y, as the output is expected to be a constant function of this input.

At this point it is pertinent to introduce a feature of the GP modelling approach known as **Automatic Relevance Detection (ARD)**. This feature was first introduced in Mackay (1994) and Neal (1996) in the context of Bayesian neural network implementations. The ARD facility utilises the anisotropic format of the squared exponential covariance function (or indeed any stationary isotropic covariance function) to assess the relative importance of contributions made by each input through the comparison of their length-scale hyperparameters. Therefore, during the optimisation of the GP model (to be discussed in section (4.2)) where the hyperparameters of the chosen covariance function are to be identified, we can also employ the ARD facility to help optimise the structure of the model. As a result, the ARD feature can be seen to be of particular value for system identification purposes where there is a lack of prior knowledge regarding the nature of suitable inputs. This is one of the attractive features of the GP modelling approach, as due to the probabilistic optimisation we can develop a greater understanding of how different inputs can influence the model. On a practical level we can see that this facility can be utilised to tune the overall model structure employed, where unimportant inputs can be eliminated from the model structure and thereby improve computational efficiency and ultimately model interpretability.

The squared exponential function embodies the property that points that are close together in input space are strongly correlated and hence give rise to similar values of target **t**. The strong smoothness properties can be demonstrated by drawing some sample functions from the defined GP prior. In Figure (4.2a) a number of different squared exponential covariance functions have been defined using different hyperparameter

values, and the impact on the sample functions drawn from the prior can be seen in Figure (4.2b).



(a) **(b)**

Figure (4.2): Chart (a) shows three different Squared Exponential covariance functions where the covariance k varies against the input distance r. The Chart (b) shows random functions drawn from the Gaussian process with the same covariance functions as Chart (a).

As expected, through increasing the size of the length-scale hyperparameter ℓ (comparing the solid line versus the dotted line) we can see that the sample functions have a tendency to vary much more slowly. Furthermore, through reducing the size of the θ_1 hyperparameter we can be seen to restrict the vertical scale of the variations of the sample functions (comparing dashed line versus solid and dotted lines). Overall, the use of the squared exponential covariance function implies an assumption that the function to be identified exhibits smooth and continuous behaviour with a high correlation between outputs and inputs in close proximity.

4.3.1.2) Matérn Class of Covariance Functions

Whilst the squared exponential function can be seen to be the most widely adopted covariance function, due to the infinitely differentiable nature of this function there is an implicit assumption that the underlying function is to be smoothly varying. This is a strong assumption that must be substantiated from prior knowledge of the system or from empirical data. Therefore, a facility that allows a less stringent prior assumption over the smoothness or differentiability of the underlying function can be seen to be an attractive

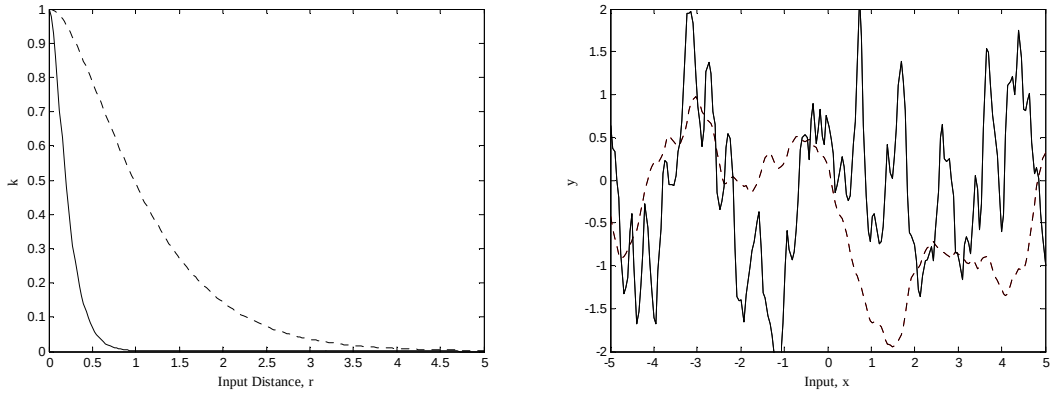
possibility. A control over the relative smoothness or differentiability of the GP prior probability is a feature of the Matérn class of covariance functions, sees Table (4.1).

The Matérn class of covariance functions is given by the expression in Table (4.1) where ν and ℓ are positive parameters, and K_ν is a modified Bessel function. The parameter ν can be seen to control the differentiability of the sample functions. As $\nu \to \infty$ the Matérn form approaches the squared exponential (infinitely differentiable) covariance function discussed above. In the text by Rasmussen and Williams (2006) the most interesting cases for machine learning purposes are stated as $\nu = 3/2$ and $\nu = 5/2$:

$$k_{\nu=3/2}(r) = \left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right) \tag{4.4}$$

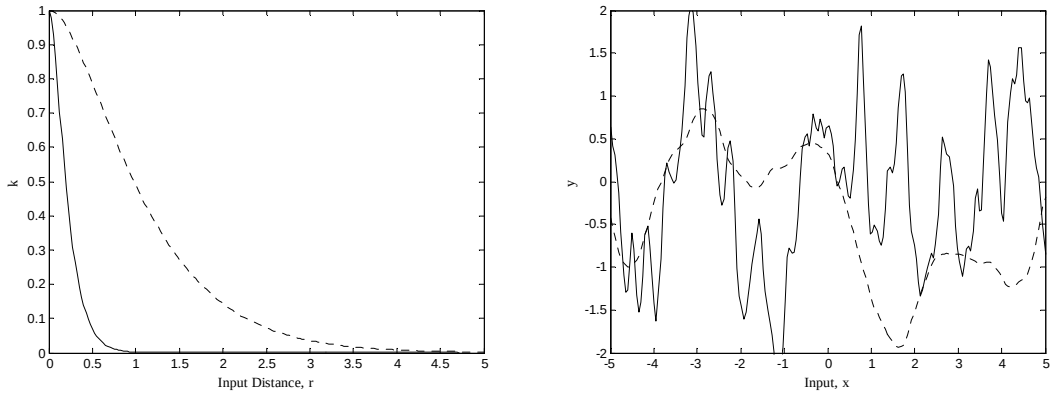
$$k_{\nu=5/2}(r) = \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right) \tag{4.5}$$

In Figures (4.3) and (4.4), both of these covariance functions (for $\nu = 3/2$ and $\nu = 5/2$) are displayed with two different length-scale hyperparameters, together with sample functions drawn from their respective priors. Again, the manipulation of the characteristic length-scale hyperparameter can be seen to have a great effect on the overall smoothness of the resultant sample functions. In comparison to the squared exponential covariance function, the covariance can be made to decay much more rapidly resulting in sample functions that can become significantly less smooth and therefore less differentiable. Furthermore, as the ν hyperparameter is increased from $\nu = 3/2$ to $\nu = 5/2$ there is a slight reduction in the roughness of the sample functions. This is in keeping with the earlier statement that as $\nu \to \infty$ the Matern form will become equivalent to the squared exponential. In Rasmussen and Williams (2006) it is stated that for cases where $\nu \geq 7/2$ the processes will be difficult to distinguish from one another, and indeed that of the squared exponential.



(a) **(b)**

Figure (4.3): Chart (a) shows two different Matérn (ν=3/2) covariance functions where the covariance k varies against the input distance r for two different length-scales. The Chart (b) shows random functions drawn from the Gaussian process defined by the same covariance functions as Chart (a).



(a) **(b)**

Figure (4.4): Chart (a) shows two different Matérn (ν=5/2) covariance functions where the covariance k varies against the input distance r for two different length-scales. The Chart (b) shows random functions drawn from the Gaussian process defined by the same covariance functions as Chart (a).

Thus, it can be seen that through the use of the Matérn covariance function we can express a lack of prior knowledge about the sample function differentiability. This proposal for more control over the relative differentiability of the covariance functions is supported in the text by Stein (1999), where the strong smoothness assumptions embodied by the squared exponential covariance function are questioned from a practical and asymptotic perspective.

4.3.1.3.) Exponential, γ-Exponential, and Rational-Quadratic Covariance Functions

In the case where $\nu = 1/2$, the Matérn form can be seen to be equivalent to the **exponential** covariance function also given in Table (4.1). As a result, the sample functions drawn from a process defined by the exponential function can be made to be highly non-smooth and therefore non-differentiable. From the exponential class of covariance functions the Ornstein-Uhlenbeck process used to model the velocity of a particle in Brownian motion can also be defined. A further class of covariance functions given in Table (41) is the **γ-exponential**. This class of function is equivalent to the previously discussed Squared-exponential covariance function when γ=2, but is not MS differentiable for $\gamma < 2$. As a result, Rasmussen and Williams (2006) state that this family of covariance functions is less flexible than the Matérn class.

The Rational Quadratic (RQ) covariance function given in Table (4.1) is an interesting covariance function as it can accommodate several characteristic length-scales. Due to this property, the RQ covariance function can be interpreted as a scale mixture or infinite sum of squared exponential covariance functions with different characteristic length-scales. The RQ covariance function exhibits the same infinitely MS differentiable properties of the squared exponential covariance function.

4.3.2) Non-stationary Covariance Functions

The most simple non-stationary covariance function discussed in Mackay (1998b) is the one corresponding to a **linear** trend.

$$k_{lin}(\mathbf{x}, \mathbf{x}'; \{\sigma_w, \sigma_c\}) = \sum_{d=1}^{D} \sigma_w^2 x_d x'_d + \sigma_c^2 \tag{4.6}$$

This linear covariance function can also be generalised into **Dot Product** covariance functions as discussed in Rasmussen and Williams (2006), where polynomial covariance functions can then be defined:

$$k(\mathbf{x},\mathbf{x}') = (\mathbf{x}\bullet\mathbf{x}') = \left(\sum_{d=1}^{D} x_d x'_d\right)^p \tag{4.7}$$

However, this polynomial is not thought to be particularly useful for regression problems, as the prior variance will become very large with $|\mathbf{x}|$ as $|\mathbf{x}|>1$.

One of the inherent assumptions of the previously discussed stationary covariance functions is that the length-scale is to be fixed in all directions. This is obviously not going to be the case for all systems and a non-stationary covariance function with the ability vary the length-scale as a function of **x** has been proposed in Gibbs (1997). This **spatially varying length-scale** covariance function defines an arbitrary positive function $\ell_d(x)$ of the input:

$$k(\mathbf{x},\mathbf{x}') = \prod_{d=1}^{D}\left(\frac{2\ell_d(x)\ell_d(x')}{\ell_d^2(x)+\ell_d^2(x')}\right)^{1/2} \exp\left(-\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{\ell_d^2(x)+\ell_d^2(x')}\right) \tag{4.8}$$

A further alternative non-stationary covariance function is the **neural network** covariance function discussed in Williams (1998) and Rasmussen and Williams (2006), but based on the Bayesian neural network research found in Neal (1996). In defining this function, the input vector is augmented as $\tilde{\mathbf{x}} = (1, x_1, \ldots, x_d)^T$, and the hidden layer transfer function used is the error function $h(z) = erf(z)$, rather than a more common sigmoid neural network function such as *tanh(z)* as this is not found to be positive definite,

$$k_{NN}(\mathbf{x},\mathbf{x}') = \frac{2}{\pi}\sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^T\Sigma\tilde{\mathbf{x}}'}{(1+2\tilde{\mathbf{x}}^T\Sigma\tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^T\Sigma\tilde{\mathbf{x}}')}\right) \tag{4.9}$$

This covariance function can also be of use in tackling problems where the length-scale is to vary across the input space. In Rasmussen and Williams (2006) this function is successfully applied to a static step data problem (i.e. slow varying steady-state followed by a rapid input transition).

In the research of Paciorek and Schervish (2004) further non-stationary covariance functions are proposed that are generalisations of Gibbs' spatially varying covariance function. Furthermore, a non-stationary version of the Matérn covariance function was outlined, thus allowing control over sample function differentiability to be combined with control over length-scale variance.

4.3.3) Combining Covariance Functions

Due to the properties of the Gaussian process it is possible to combine different covariance functions in order to define new stationary and non-stationary covariance functions. As a result, different aspects of the nonlinearity of the underlying function can be treated by individual kernels and combined into a global covariance function. This facility is discussed in both Mackay (1998b) and Rasmussen and Williams (2006). Furthermore, an informative example is provided in Rasmussen and Williams (2006) where a number of covariance functions are combined towards the identification of a complex nonlinearity composed of a number of different contributing nonlinearities.

4.3.3.1) Sum of Covariance Functions

Fundamentally, a sum of kernels can be seen to be a kernel itself. If a random process $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$ where $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are independent, then the kernels that generate them can also be combined $k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')$ and be considered a valid covariance function. This construction can be seen to be particularly useful for application in nonlinear problems where a number of different characteristic length-scales can be observed, and therefore has similarities to the Rational Quadratic covariance function.

4.3.3.2) Product of Covariance Functions

Similarly, a product of two kernels can be seen to be kernel. If $k_1(\mathbf{x},\mathbf{x}')$ and $k_2(\mathbf{x},\mathbf{x}')$ are covariance functions on the same input space then they can be combined as $k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$. For covariance function over different spaces, $k_1(\mathbf{x},\mathbf{x}')$ and

$k_2(\mathbf{y},\mathbf{y}')$, a product space can be defined as $\mathbf{z}=(x,y)$, and the covariance functions $C_1(\mathbf{z},\mathbf{z}')=k_1(\mathbf{x},\mathbf{x}')+k_2(\mathbf{y},\mathbf{y}')$ and $C_2(\mathbf{z},\mathbf{z}')=k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{y},\mathbf{y}')$ may then also be defined.

Further discussion of these possibilities can be found in Rasmussen and Williams (2006), but from an overall perspective we can see that if faced with complex nonlinearities, the possibility exists to break the problem down into constituent nonlinear contributions and devote an individual covariance function to tackle each component. From this point an additive model may be defined through utilising individual covariance functions as building blocks for a more global representation. As a result, this feature of the GP modelling approach can be seen to be particularly in keeping with the divide-and-conquer approach that has been adopted in other methods of nonlinear system identification.

4.3.3.3) Vertical Rescaling and Convolution

A straightforward method of transforming a given stationary covariance function into a non-stationary version is to introduce another function $a(\mathbf{x})$, giving $C(\mathbf{x},\mathbf{x}')\equiv a(\mathbf{x})k(\mathbf{x},\mathbf{x}')a(\mathbf{x}')$. This method can also be used to normalise kernels. Furthermore, Mackay (1998b) discusses the potential exists to convolve (or 'blur') an existing covariance function to generate a new one, through integration with an arbitrary kernel h, i.e. $C(x,x')=\int dy\,dy'\,h(x-y)k(y,y')h(y'-x')$.

4.3.3.4) Nonlinear Mapping (Warping)

A further alternative method of implementing a non-stationary solution is to employ an arbitrary nonlinear mapping (also known as warping) of the input $\mathbf{u}(\mathbf{x})$ to handle the non-stationary nonlinearity of the function in tandem with a stationary covariance function to operate in **u**-space.

$$C(\mathbf{x},\mathbf{x}')\equiv k(\mathbf{u}(\mathbf{x}),\mathbf{u}(\mathbf{x})') \tag{4.10}$$

As the original input space **x** need not exhibit the same dimensionality as that of the new **u**-space, we are free to use whatever input mapping is most conducive to identifying a

satisfactory model. This facility is demonstrated in Mackay (1998b) to define a periodic random covariance function. Examples of this strategy can be found in the paper by Sampson and Guttorp (1992), and in Gibbs (1997) where the nonlinear mapping strategy is contrasted with the previously mentioned spatially varying length-scale covariance function. A more in-depth analysis of 'Warping' can also be found in Snelson et al. (2004) Snelson (2007). Furthermore, in Girard (2004) empirical data collected from a Ph Neutralisation plant is first modelled using a linear model, with the subsequent residual is then modelled by a stationary GP model defined with the squared exponential covariance function. In this example, the strategy taken is not to actively 'warp' the covariance function to identify a new covariance function, but to modify the input space data in a manner that allows the easier implementation of a subsequent GP model (i.e. to define a set of latent input variables).

Through the use of this nonlinear mapping strategy, the potential exists for the GP modelling approach in its most common form (i.e. a stationary GP defined using a Squared Exponential) to be combined with other modelling strategies to form a hybrid representation. Such a hybrid approach may be particularly useful for problems where an existing but somewhat inaccurate model (such as an analytical model derived from first principles) can be combined with the powerful data-driven approach of the GP model. As a result, the overall interpretability of the original description may be somewhat retained and then combined with the simplest and most interpretable form of the GP model acting as a corrective device. This may be an attractive alternative to the use of more complex non-stationary covariance functions, which may be less interpretable and more difficult to train due to the potential for a greater number of hyperparameters.

4.4) GP Model Optimisation

Due to the probabilistic nature of the GP model, the popular model optimisation approach where model parameters, and possibly also the model structure, are optimised through the minimisation of a loss function defined in terms of model error (e.g. mean square error), is not readily applicable. Furthermore, as GP modelling has also been described as a Bayesian probabilistic method, a probabilistic approach to the optimisation of the model would seem appropriate. Fortunately, we have already discussed the

framework for Bayesian learning in the previous chapter. Essentially, instead of minimising model error, it is the probability of the model that is to be maximised. Therefore, after the selection of an appropriate covariance function, it is the hyperparameters of this function that must be optimised to accurately reflect the correlations present in our observed training data set.

4.4.1) Optimising Hyperparameters

The overall problem of learning unknown parameters from data can be seen to correspond to the first level of Bayesian inference discussed previously. The overall goal of this first level of Bayesian inference was to obtain the predictive distribution $P(t_{N+1} \mid t_N, X_N, x_{N+1})$ of the new target t_{N+1} given the training data (t, X) and a new input x_{N+1}. In order to realise this posterior distribution, a prior distribution over the hyperparameters can first be defined $P(\theta \mid t_N, X_N)$, followed by the integration of the model over the hyperparameters

$$P(t_{N+1} \mid t_N, X_N, x_{N+1}) = \int P(t_{N+1} \mid \theta, t_N, X_N, x_{N+1}) P(\theta \mid t_N, X_N) d\theta \qquad (4.11)$$

As discussed in the previous section on Bayesian modelling, the computation of such integrals can prove difficult due to the intractable nature of the nonlinear functions. One solution to the problem of intractable integrals is to adopt numerical integration methods such as the Monte-Carlo approach. These numerical methods offer considerable flexibility and accuracy of approximation. Unfortunately, significant computational expense may be required in order to achieve a sufficiently accurate approximation. An alternative approach based on the Maximum Likelihood optimisation method has also been developed and is applied to maximise the marginal likelihood or evidence. Therefore, by searching for hyperparameters that maximise the probability of the training data, we are optimising the properties of the Gaussian process prior that is to be used to generate new predictive distributions. Both of these methods are discussed below, beginning with the Marginal Likelihood maximisation approach.

4.4.2) Marginal Likelihood (Evidence) Maximisation

This method of optimisation is based upon the application of Bayesian inference and is commonly referred to as Marginal Likelihood or 'Evidence' maximisation, see Mackay (1992c), Rasmussen (1996) and Gibbs (1997). This optimisation strategy dispenses with the need for potentially time-consuming or computationally intensive MCMC methods of numerical integration with the computational burden scaling linearly with the number of hyperparameters. Instead, an approximation to the integral is made through the use of the **most probable** values of hyperparameters $\boldsymbol{\theta}_{\text{MP}}$.

$$P(t_{N+1} \mid \mathbf{t}_N, \mathbf{X}_N, \mathbf{x}_{N+1}) \simeq P(t_{N+1} \mid \boldsymbol{\theta}_{MP}, \mathbf{t}_N, \mathbf{X}_N, \mathbf{x}_{N+1}) \tag{4.12}$$

The basis for the approximation is the assumption that the posterior distribution $P(\theta \mid t_N, X_N)$ is sharply peaked around $\boldsymbol{\theta}_{\text{MP}}$ relative to the variation in the predictive distribution $P(t_{N+1} \mid t_N, X_N, x_{N+1}, \theta)$. Therefore, this optimisation strategy relies upon the identification of the most probable hyperparameters from the training data, signified by the posterior distribution $P(\theta \mid t_N, X_N)$. The inference of this posterior probability is performed through the straightforward application of Bayes' theorem:

$$P(\boldsymbol{\theta} \mid \mathbf{t}_N, \mathbf{X}_N) \propto P(\mathbf{t}_N \mid \mathbf{X}_N, \boldsymbol{\theta}) P(\boldsymbol{\theta}) \tag{4.13}$$

where $P(\mathbf{t}_N \mid \mathbf{X}_N, \boldsymbol{\theta})$ is the marginal likelihood or evidence (or probability of the data), and $P(\boldsymbol{\theta})$ is a prior over the hyperparameters. Note that this posterior probability has been expressed as proportionality, rather than as a function due to the omission of the denominator that is independent of the hyperparameters.

Overall, this Bayesian method of determining the hyperparameters can be seen to offer significant advantages over other model selection and optimisation methods utilised by alternative modelling approaches. In particular, by performing optimisation through the analysis of the marginal likelihood, the automatic implementation of Occam's Razor detailed in the previous chapter can be used to regulate complexity of the model and thereby curtail overfitting.

4.4.2.1) Marginal Likelihood Loss Function

The marginal likelihood component $P(\mathbf{t}_N \mid \mathbf{X}_N, \boldsymbol{\theta})$ of the previous proportionality has been discussed in the previous chapter on Bayesian learning, and (assuming a zero-mean) may be stated as:

$$P(\mathbf{t}_N \mid \mathbf{C}_N, \{x_n\}) = \frac{1}{Z_N} \exp\left(-\frac{1}{2} (\mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N) \right) \tag{4.14}$$

If we ignore the Prior over the hyperparameters $P(\boldsymbol{\theta})$ for the moment, we can restate the marginal likelihood as a loss function that is to be maximised. The log of the marginal likelihood is first taken for numerical scaling purposes (note that the negative log transforms this into a minimisation), resulting in the loss function

$$L(\boldsymbol{\theta}) = -\frac{1}{2} \log(|\mathbf{C}_N|) - \frac{1}{2} \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N - \frac{N}{2} \log(2\pi) \tag{4.15}$$

The three components of the log marginal likelihood function have interpretable roles as described in Rasmussen and Williams (2006). The only component that includes the observed target data is $-\frac{1}{2} \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N$ and can be interpreted as a 'data-fit' term. The term $-\frac{1}{2} \log(|\mathbf{C}_N|)$ is dependent only on the choice of covariance function and the input data, and may be interpreted as 'complexity' penalty. The final component $-\frac{N}{2} \log(2\pi)$ acts a normalisation constant. The discussion example in Rasmussen & Williams (2006) shows that the 'data-fit' term can be seen to decrease monotonically as the length-scale increases. This is what would be expected, as an increase in length-scale would be symptomatic with a loss of flexibility in the model. By contrast, the negative 'complexity' term will be seen to increase with an increase in length-scale due to the model becoming ever less complex.

Furthermore, the marginal likelihood itself will become more peaked as the number of included training data points is increased. This is in agreement with what would be

expected, as with more available data a better insight into the underlying function will be forthcoming, resulting in a more likely approximation. Relating the marginal likelihood to the length-scale, for problems where few points are available, the slope of the log marginal likelihood is shallow as both short and intermediate values of the length scale can be considered as consistent with the data. If the length-scale increased to be larger than 1, the marginal likelihood can be seen to decrease rapidly as the model no longer provides a good approximation to the data. With larger amounts of data, the 'complexity' term of the loss function becomes more severe and therefore acts to discourage adoption of length-scales that are too short, and therefore guards against overfitting.

4.4.2.2) Gradient Calculations

The next stage is to find the maximum/minimum of this loss function and therefore locate the most probable hyperparameters. As we are endeavouring to locate the maximum of the log marginal likelihood, this process is therefore equivalent to locating the Maximum A-Posteriori (MAP) estimate of this distribution. Therefore, given the nonlinear loss function $L(\boldsymbol{\theta})$, we can analytically express the partial derivatives of the log marginal likelihood with respect to hyperparameters $\boldsymbol{\theta}$ as follows:

$$\frac{\partial L}{\partial \theta} = -\frac{1}{2} trace(\mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta}) + \frac{1}{2} \mathbf{t}_N^T \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta} \mathbf{C}_N^{-1} \mathbf{t}_N \tag{4.16}$$

In order to perform these gradient calculations, a nonlinear (local) optimisation algorithm is required. Furthermore, it is important to note that the calculation of the derivative of the likelihood again relies upon the efficient calculation of the inverse covariance matrix. As stated previously, the inversion of large matrices is a computationally expensive process of the order $O(N^3)$. Therefore, to ensure that a viable optimisation is achieved, the size of the covariance matrix to be inverted (and therefore the size of the training set) must not be unfeasibly large. Once the inversion has been computed, the remaining components of the log likelihood and its derivatives are less computationally demanding being of the order $O(N^2)$.

To perform the nonlinear optimisation, a conjugate gradients approach to the problem has been successfully implemented to locate a local maximum of the log marginal

likelihood. In this thesis we follow the same implementation as Rasmussen (1996) where a Polack-Ribiere version of the conjugate gradients methods is utilised in tandem with the Wolfe-Powell stopping conditions. Further information on various different types of conjugate gradients methods can be found in Fletcher (1987, 1993). Compared with methods relying upon MCMC integration, this conjugate gradients approach can find a reasonable approximation to a local maximum after relatively few function and gradient evaluations.

4.4.2.3) Multiple Local Maxima

As with any nonlinear local optimisation based on the identification of local maxima, the MAP estimation through the conjugate gradients method may be subject to problems where the marginal likelihood is multi-modal. In such cases the algorithm may become stuck in bad local maxima that ultimately results in a poor estimation of the most probable hyperparameters. Fortunately, other researchers into the GP model, (Rasmussen (1996), Gibbs (1997)) have found that for simple covariance functions, the scale of the problem presented by multiple local maxima is not something that cannot be overcome if a degree of care is taken over the optimisation procedure.

Furthermore, it is worth remembering that these alternative local maxima are merely different interpretations of the data. Therefore, for applications where the training dataset is relatively small, a large number of potential interpretations of the data (and therefore local maxima) will be possible as relatively little information has been presented to the optimisation procedure. However, given sufficient data, a more acute or obvious interpretation of the data should begin to emerge, resulting in a local maximum that is significantly larger than alternative modes. As a result, we can then dismiss other local maxima (interpretable as alternative models) as being less likely. This can be seen to relate back to the previous discussion regarding the automatic Occam's Razor effect that is inherent with optimisation through the marginal likelihood.

In addition, further links between the type of covariance function being optimised and the propensity for encountering problematic multiple local maxima can be made. As more complex covariance functions (consisting of more hyperparameters) are likely to offer greater model flexibility, and therefore more possible interpretations of the training data,

the potential for more local maxima should increase. Therefore, this might lead to further problems finding the 'most probable' hyperparameters using the marginal likelihood maximisation technique. Thus, an alternative optimisation strategy, such as MCMC methods detailed below, may be required for determining the hyperparameters of more complex covariance functions.

Nevertheless, we can outline strategies aimed at minimising problematic multiple local maxima. One possible strategy is to attempt multiple runs of the optimisation algorithm utilising different initial values for the hyperparameters. As the algorithm is dependant on the evaluation of the partial derivatives of the likelihood, unsuitable initial hyperparameter values that result in very small derivative values may cause computational difficulties. Therefore, different initial conditions could be either be selected at random, or deterministically where we may purposely avoid initial points of the training data (or regions of operating space) that have proven to be problematic. Performing multiple runs of the optimisation procedure may also prove useful if more than one set of training data is available. In cases where a number of viable alternative sets of hyperparameters have been obtained, the final selection of hyperparameters can then be performed through analysing the performance of each model through model validation.

A further option for improving the optimisation procedure is make use of the Prior component of the Bayesian inference. So far, the prior over the hyperparameters $P(\boldsymbol{\theta})$ has been ignored, and the marginal likelihood used as in Maximum Likelihood optimisation. Therefore, the role of the prior over hyperparameters had been relegated to merely being a set of initial values for $\boldsymbol{\theta}_{MP}$. From a certain perspective, this dismissal of the Prior distribution from the optimisation procedure can be seen to be somewhat contrary to the spirit of Bayesian inference, as without the inclusion of prior knowledge, possible values for the most probable hyperparameters $\boldsymbol{\theta}_{MP}$ that are incorrect or inconsistent with the data or covariance function may be allowed. However, as discussed in the previous chapter, the determination of suitable prior distributions based on 'a priori' system knowledge is a difficult task. One possibility discussed in the research by Neal (1996) and Gibbs (1997) is to employ Gamma distributions as priors over $\boldsymbol{\theta}_{MP}$.

4.4.3) Monte-Carlo Alternative

Another possible optimisation strategy is to perform the integration over $\boldsymbol{\theta}$ is through using numerical MCMC methods as described in Williams and Rasmussen (1996), and Neal (1997). The MCMC approach employs a Markov chain to approximate the integral together with sampling methods to calculate an approximation to the overall predictive distribution, as described by:

$$P(t_{N+1} \mid \mathbf{x}_{N+1}, \mathbf{X}_N, \mathbf{t}_N, C(.)) \simeq \frac{1}{T} \sum_{t=1}^{T} P(t_{N+1} \mid \mathbf{x}_{N+1}, \mathbf{X}_N, \mathbf{t}_N, C(.), \boldsymbol{\theta}_t) \tag{4.17}$$

where the $\boldsymbol{\theta}_t$ are samples drawn from the posterior distribution over $\boldsymbol{\theta}$, $P(\boldsymbol{\theta} \mid \mathbf{X}_N, \mathbf{t}_N, C(.))$. The resultant accuracy of the MCMC approximation is dependent on the number of samples taken from the posterior distribution over $\boldsymbol{\theta}$, and as each term in this summation is a Gaussian distribution, the MCMC approximation to the desired predictive distribution can be termed a mixture of Gaussians. Furthermore, as we are sampling from the posterior over $\boldsymbol{\theta}$, prior distributions over the hyperparameters $P(\boldsymbol{\theta})$ will be required as in the previous strategy based on the maximisation of the marginal likelihood.

Typically, in order to facilitate a good approximation to the integral, a large number of samples must be taken from the posterior. Therefore, the adoption of the MCMC approach carries a potentially high computational cost, especially for problems where large numbers of observations are included in the training set and must be stored as the algorithm proceeds. As a result, methods to improve the efficiency of the algorithm may be required. In particular, the method used to sample from the posterior distribution over $\boldsymbol{\theta}$ will influence the efficiency of the approach, as the samples taken from the posterior must adequately represent the underlying distribution. For example, if a particular region of $\boldsymbol{\theta}$ space is not adequately sampled, the overall approximation to the integral will suffer, especially if this region has a high associated probability.

A further step toward improving the efficiency of the MCMC method for use in Gaussian processes found in the work presented by Williams and Rasmussen (1996) and Neal (1997), is the adoption of the Hybrid Monte-Carlo method developed by Duane et al.

(1987). The key objective of adopting this particular method of MCMC is to avoid random walk behaviour of the more popular Gibbs and Metropolis sampling approaches. The Hybrid MCMC is a stochastic dynamics sampling algorithm that introduces an auxiliary momentum vector (gradient information) with which to move across the sample space in larger steps and thereby sample from the posterior more efficiently and converge more rapidly to the target distribution.

4.4.4) Which Optimisation Method?

In the existing literature on GP models, a preference for the optimisation through marginal likelihood maximisation has been indicated. As a result, it is the method adopted in this thesis. Although this method fundamentally relies upon an approximation, and may be prone to difficulties associated with multiple local maxima, given a decent set of training data this method has been proven to provide very good estimates for optimal hyperparameters. Furthermore, as well as providing sufficient training data conducive to obtaining a good estimation of the most probable hyperparameters, a number of possible strategies (multiple restarts, different initial conditions or express priors) exist so that the problems associated with multiple local maxima may be mitigated. Furthermore, the fundamental result of optimisation through the marginal likelihood, where a set of most probable hyperparameters are obtained, is an attractive result in itself, as it can provide the user with an insight into the data being modelled.

The marginal likelihood maximisation has a further advantage over the MCMC alternative when examined in computational terms. Upon the completion of the algorithm only one set of final hyperparameters exist, leaving a single covariance matrix that must be stored and subsequently inverted in order to make predictions. If further training data points are to be included at a later date, only one inverse covariance matrix must therefore be updated using the partitioned inverse equations detailed previously. This is not the case for the MCMC method where all the inverse matrices must be stored if new points are to be included, thus adding to the potential computational expense of the MCMC approach.

However, in terms of overall accuracy and flexibility, the MCMC method has potential advantages. As the MCMC method gradually builds a more exact approximation of the

integral, for problems where more complex covariance functions are being employed the MCMC may provide a better result. Furthermore, for problems where the size of the training set is relatively small and therefore the amount of matrices to be stored does not prove to be infeasible, the performance of the MCMC method may be preferable to that of the marginal likelihood maximisation method.

4.5) Mathematical & Computational Implementation

In the identification of a GP model, we wish to construct a covariance matrix that defines a Gaussian process with which to infer new test predictions. This covariance matrix is to be specified through the application of a covariance function to a set of training observations. In the discussion so far, we have mentioned the various choices of covariance function that are available to us, together with strategies for the determination of optimum hyperparameters from our set of training data. What has not been discussed are issues relating to the actual training dataset itself. Therefore, in this section the requirements for this training dataset are to be discussed together with the computational implementation of the GP model.

4.5.1) Size of the Covariance Matrix

As can be seen from the predictive equations (3.40) and (3.41), the identification of a GP model revolves around the specification and manipulation of the covariance matrix. As a result, the GP modelling approach can be seen to be particularly susceptible to any mathematical or computational difficulties found when performing matrix manipulation. In particular, it is the inversion of the covariance matrix that has been proven to be the main source of difficulty in the GP modelling approach. The inversion of large matrices is a well-established computational problem encountered across many research fields where large amounts of data must be analysed and manipulated. Consequently, the potential size of the covariance matrix to be inverted is something that any researcher must be conscious of. Furthermore, as the size of the training data set (N) dictates the size of the resultant covariance matrix (N×N), the number of observed data points included in the training set must therefore be kept within reasonable limits if the GP model (as prescribed by the predictive equations) is to be implemented directly.

From a more fundamental perspective, we can also see that by placing limits on the size of the covariance matrix, we are also potentially restricting the amount of information we can include in the training dataset. Therefore, this mathematical constraint can be seen to impact the potential flexibility of the GP modelling approach, e.g. for complex nonlinear problems where thousands of data-points are necessary to adequately characterise the system. Therefore, in recent years a variety of methods aimed at overcoming this constraint on the size of the covariance matrix have been proposed. These strategies are often referred to as Approximate Methods, and are to be discussed in Section (4.5.5).

4.5.2) Conditioning of the Covariance Matrix

Another implication of the mathematical framework of the GP modelling approach is that in order for the inversion of the covariance matrix to remain accurate and computationally viable, the matrix to be inverted must not be 'ill-conditioned'. The term '**ill-conditioned**' matrix relates to the condition number of a matrix, which provides a measure of stability or sensitivity of a matrix to certain numerical operations (i.e. how numerically 'well-posed' is the problem?). A low condition number is indicative of a problem that is 'well-conditioned', 'too large' a condition number is indicative of 'ill-conditioning', and an infinite condition number is indicative of a matrix that is singular and therefore does not have an inverse. A further aspect to the conditioning requirements placed upon the covariance matrix has been outlined in the previous section detailing the theory of various covariance functions. In order for a valid Gaussian process to be defined and subsequently used as a Prior with which to infer predictions, the covariance matrix must be Positive Semi-Definite (PSD) in order to ensure consistency. In the simplest terms the constraint for PSD can be most easily interpreted as the requirement that the eigenvalues of the covariance function must be non-negative.

Therefore, we have a constraint fundamental to the GP theory that the covariance matrix must be conditioned to be PSD, and a practical constraint on the conditioning of the covariance matrix that it may be suitably conditioned for numerical manipulation. Furthermore, these two conditioning requirements can be seen to be somewhat complimentary, and therefore difficult to decouple from one another upon examination of the matrix. However, by ensuring positive semi-definiteness we should also go some way to ensuring a well-conditioned and therefore invertible matrix.

One aspect of ensuring that the covariance matrix is appropriately conditioned to meet the PSD requirement is to employ a valid covariance function as described in the opening sections of this chapter. However, the characteristics of the training data to be utilised by the chosen covariance function also play an important role in determining whether or not the resultant covariance matrix is suitably conditioned. Furthermore, as well as ensuring that the type of data included in the training dataset is conducive to the construction of a valid GP model, the training data collected must also meet the demands of the system identification task to be undertaken. Therefore, gaining an appreciation of the potential problems associated with the inclusion of particular types of data (e.g. steady-state, rough, smooth, oscillatory etc.) in the training dataset would be a useful step in ensuring the successful implementation of a GP model.

4.5.2.1) Dealing with Non-Positive Definite Matrices

As discussed previously, the covariance matrix to be generated from the application of the covariance function to the training data must be Positive Semi-Definite in order to meet the consistency requirements of the GP model. In other words, the eigenvalues of the covariance matrix must be non-negative. Therefore, if we can identify the cause of negative eigenvalues in the covariance matrix we can hopefully take steps to eliminate their presence and ensure a suitably conditioned matrix. A good resource for dealing with this problem of Non-positive definite matrices is Wothke (1993).

4.5.2.1.1) Negative Eigenvalues from Problematic Data

One of the principal causes of negative eigenvalues, and therefore matrices that are not positive definite, is the presence of unsuitable data in the training set. In particular, the presence of equilibrium or constant data in the training dataset can be a major contributory factor in the definition of a not positive definite matrix. If a variable can be seen to remain almost constant (such as when recording the steady-state response of the system under identification), it will exhibit zero variance and result in a covariance matrix that may be non-positive-definite. Further covariance matrix problems may also be encountered where a near perfect linear dependency (or correlation) exists between two variables. Therefore, from an overall perspective, we can see that the conditioning requirements placed upon the covariance matrix have potential implications for the

nature of the types of system response data we can readily utilise. This in turn will have implications for the experimental design aspects of the system identification process.

4.5.2.1.2) Eigenvalue Decomposition

A possible strategy for diagnosing the cause of matrix conditioning problems is to perform an eigenvalues decomposition of the covariance matrix. Through this analysis, we may be able to locate problematic training cases and therefore make adjustments to the training dataset more easily. Furthermore, performing such eigenvalue decomposition may also be of interpretable benefit if the GP model is to be viewed through the weight-space interpretation of Rasmussen and Williams (2006). Software development environments such as Matlab, should allow the straightforward computation of the eigenvalues of a given matrix. Furthermore, the condition number of a matrix may also be computed and thus establish whether or not a matrix is indeed ill-conditioned. However, the size and dimensions of the matrix to be computed may impact on the viability of conducting this kind of analysis repeatedly.

4.5.2.1.3) Training Data Pre-processing

As an alternative strategy to performing eigenvalue decomposition, it may be possible to identify problems in the training dataset by simply maintaining an awareness of the empirical data that is to be included. Therefore, it may be possible to tackle conditioning problems that result from problematic data (such as prolonged steady-state response data) directly without resorting to further computational manipulation. However, for more complex implementations where the nature of a multitude of inputs must be accounted for, it may not be particularly straightforward to detect exactly the cause of conditioning problems, or determine possible remedies. One possible strategy would be to add each input dimension to the covariance matrix in turn, whilst maintaining an awareness of the conditioning of the matrix at each stage.

Moving forward, once a problematic region of data has been identified, the next stage to be tackled is to determine what course of action should be taken. The most straightforward approach would be simply to remove any data points from the training dataset that are causing conditioning difficulties. Furthermore, as we also have

constraints on the overall size of the training dataset to meet, this may prove to be a sensible approach. In the case of equilibrium data such as that resulting from a prolonged steady state response, these training cases can be interpreted as repeated and therefore redundant data, and therefore prime candidates for elimination. However, before removing any data from the training dataset the overall principle that we are also potentially removing relevant information from the modelling process must also be appreciated. For the case where we wish to eliminate equilibrium data, the potential exists to be too aggressive with the removal process resulting in the loss of important information in the transition between transient and steady-state operating regions.

A further option in attempting to improve the conditioning of the covariance matrix is to gain an appreciation of the noise level of the underlying system or function. For a system response that exhibits a high level of noise, prolonged periods of constant or repeated data would seem to be improbable. As a result, the resultant conditioning of the covariance matrix may not be as adversely affected by periods close to equilibrium. This is something that can be used to our advantage through the introduction of a random element or 'jitter' to the raw empirical data where the level of noise present in the data can be increased so as to combat any conditioning errors encountered. The addition of a small 'jitter' term or ridge adjustment to the diagonal elements of the covariance matrix acts to attenuate the estimated dependency between variables, and has been shown to improve the overall conditioning of the covariance matrix in Neal (1996). Furthermore, this strategy can be seen to be equivalent to the ridge regression regularisation methods that are often used in other modelling approaches. Of course by introducing noise, we are also potentially introducing error into the model, therefore such a 'jitter' term should not be inappropriately large and be in keeping with the relative magnitude of transitions observed in the system under investigation.

Overall, we can see that the specification of a good training dataset can prove to be a challenging aspect of the GP modelling approach that may involve significant pre-processing of the empirical data. Not only must the training dataset be of a reasonable size, but also appropriately conditioned so as to meet the requirements of the mathematical framework of the GP model. Furthermore, we can see that such requirements have significant implications for the experimental design procedure utilised to collect the training data in the first place.

4.5.3) Implications for Experimental Design

At this point we have established the impact that the peculiarities of the mathematical framework of GP models can have on the size and condition of the covariance matrix. Furthermore, such requirements can be seen to extend their influence into the experimental design process that is to be adopted in order to generate the training data. In much of the current literature devoted to the GP modelling approach this is an aspect that has not been discussed in great detail. This is probably due to the statistics and machine learning origins of the method where the design of the training data set may not be something that the researcher has complete control over. However, the objective of this project has been to provide guidance for the implementation of GP models towards system identification tasks, and as a result the design of experimental procedures to collect data is something that is of fundamental importance.

In the previous discussion, the size of the covariance matrix has been identified as a potential source of implementation problems due to the algorithm's need for repeated matrix inversion. As the size of the covariance matrix is dictated by the size of the training dataset, this constraint can be seen to have a direct influence on the choice of sampling rate employed in the collection of data from a system. Through the existence of an upper limit on the size of the covariance matrix, we may be forced into choosing a sample rate that would be lower than normally recommended by standard system identification procedures (e.g. rules of thumb based on the limits associated with Nyquist sampling theory) so as not to include excessive data. A further pressure on the choice of sampling rate comes from the knowledge that large quantities of equilibrium or repeated data can have a detrimental effect on the conditioning of the resultant covariance matrix. This result can be due to an overly high sampling rate where a large number of points are collected resulting in data points so close together that any variance is diminished, therefore affecting the conditioning of the covariance matrix.

Nevertheless, the choice of sampling rate must also be adequate to meet the demands of the system identification task. Fundamental in this task is that we retain enough information within the sampled data so as to adequately represent the underlying function, i.e. we can capture the dominant nonlinearities exhibited. Furthermore, another important facet to the development of a good mathematical model is that, through the

experimental design process, we attempt to gather as much information about the system as possible. In essence, we would wish the training dataset to cover as much of the operating range of the system as possible. Therefore we must balance the demands for a reasonably sized training dataset with the requirements that as much of the operating range be included, and indeed sampled in a manner so as to adequately represent the behaviour of the system.

We can now move on to consider the impact of the conditioning requirements of the GP approach on the experimental design component of the system identification process. Previously we have pointed to the potential for equilibrium or steady-state data to cause conditioning problems in the covariance matrix. For the identification of engineering systems this would appear to be a serious problem, as the empirical data gathered from such applications routinely includes both equilibrium and transient behaviour as the system is moved through various operating points. Furthermore, many systems are explicitly designed to remain in relatively stable operating regions so as to facilitate manual or even automatic control. These operational constraints therefore present further challenges to the design of the training dataset.

Furthermore, the types of excitation signals that are readily employed to gather response data are also very likely to include periods of constant or equilibrium data as, in seeking to identify nonlinear dynamic systems, researchers often design inputs that elicit a response that takes a significant time to develop. For example, in response to a step input, a system may have an initial transient or oscillatory behaviour, leading eventually to a steady-state response. For the identification of this system, all of this information will need to be captured in order to fully characterise the system response. Similarly, many engineering systems may exhibit a delay in the response to an input (dead-time), or a saturation of the output in response to an input. In all of these cases, the potential for the inclusion of steady-state or constant data in the training set is great, especially if previous inputs or outputs are to be utilised as regressors. Thus, whilst the presence of equilibrium data in the training dataset and the impact on the conditioning of the covariance matrix may be easy to identify, strategies to overcome this problem whilst holding true to the demands of the system identification task are needed. This problem is not something that is unique to identification using GP models as many other modelling approaches (including linear regression) are also subject to matrix conditioning problems that result

from incompatible training data. The main solution to this problem is to ensure that the experimental procedure employed endeavours to excite the system sufficiently so as to ensure the empirical data collected does not contain prolonged periods of steady-state data. In the next chapter the problems associated with the pre-processing of training data and its impact on experimental design procedures are to be investigated with a number of practical examples. However, at this point, the computational implementation of the GP model is to be discussed.

4.5.4) Direct Implementation of the GP model

If we now assume that a well-conditioned covariance matrix has been constructed we can look into the direct implementation of the GP model. In the direct implementation of the GP model the goal is to compute the predictive equations (3.40) and (3.41) exactly. However, as discussed previously, the size of the covariance matrix as dictated by the number of included training points (N) can prove to be computationally challenging. This is due to the repeated multiplication and inversion of the potentially large covariance matrix that is required not only by the predictive equations, but also by the optimisation techniques discussed previously. A number of different 'approximate' methods have been proposed that deal directly with the size constraints of the GP modelling approach and these are discussed in the next section.

Before discussing the precise details of the direct implementation it is also first useful to discuss the computational limits that have been established for this direct implementation of the GP model. For the GP modelling approach, the overall computational burden for the direct implementation has been estimated as $O(N^3)$ by Rasmussen and Williams (2006). This has led to the recommendation that for large problems (N>10000), further approximation methods (described in the next section) should be adopted. Earlier texts by Mackay (1998b) and Gibbs (1997) put a feasible limit of (N<1000) points upon the size of the covariance matrix. In my own experience working with data sets for the identification of dynamic engineering applications, the lower limit of (N<1000) data points is more realistic for those working with average desktop PC computational facilities.

An exact implementation of the GP model predictive equations outlined in Gibbs and Mackay (1997) is as follows:

The predictive equations to be computed are

$$\hat{t}_{N+1} = \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{t}_N$$
$$\sigma_{N+1}^2 = \kappa - \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}$$

Given a new test input $\mathbf{x}_{N+1}$, to calculate a single prediction $\hat{t}_{N+1}$, the following procedure can be followed:

1) Construct the vector, $\mathbf{k}_{N+1} = \left[C\left(\mathbf{x}_1, \mathbf{x}_{N+1}; \boldsymbol{\theta}\right), \ldots, C\left(\mathbf{x}_N, \mathbf{x}_{N+1}; \boldsymbol{\theta}\right) \right]$

2) Invert covariance matrix, $\mathbf{C}_N^{-1}$

3) Calculate the vector, $\mathbf{v} = \mathbf{C}_N^{-1} \mathbf{t}_N$

4) For the mean prediction, find the dot product, $\hat{t}_{N+1} = \mathbf{k}_{N+1}^T \mathbf{v}$

5) Evaluate covariance of test input, $\kappa = C\left(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}; \boldsymbol{\theta}\right)$

6) Calculate the scalar, $\mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}$

7) Subtract for variance prediction, $\sigma_{N+1}^2 = \kappa - \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1}$

For subsequent test inputs, $\mathbf{x}_{N+2}$ and so on, to calculate new predictions only the new vector $\mathbf{k}_{N+2}$ need be constructed, as the vector $\mathbf{v}$ will not have changed. Therefore, after the initial test input (where the matrix must first be inverted $\mathbf{C}_N^{-1}$ and applied to a vector $\mathbf{t}_N$), the calculation of the remaining prediction horizon requires only the evaluation of the dot product, i.e. $\hat{t}_{N+2} = \mathbf{k}_{N+2}^T \mathbf{v}$, to be repeated, therefore reducing the remaining computational demand to around that of $O(N)$.

With regard to the implementation of the Marginal Likelihood maximisation algorithm used to find the most probable hyperparameters θ_{MP}, we can also see that that each calculation of the gradient of the log likelihood requires the inversion of the covariance matrix. Gibbs and Mackay (1997) break down each evaluation so that they require 4 matrix to vector applications, and 1 dot product calculation. However the evaluation of

$trace(\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta})$ can be seen to avoid a full calculation of $\mathbf{C}_N^{-1}$ as the trace operator only requires diagonal elements.

4.5.4.1) Using Matrix Decomposition

Whilst this direct method of implementing the GP model has significant computational disadvantages due to the need for the repeated explicit inversion of the covariance matrix, a further disadvantage is that through this inversion and the subsequent application and dot product computations involving this matrix the overall accuracy and computational stability of this method can become compromised. This is due to potential for ill-conditioning in the covariance matrix, as discussed earlier. As a result alternative methods for the direct implementation of the predictive equations have been developed so as to avoid the need for the explicit inversion of the covariance matrix. Rather than attempt the direct inversion of matrices (involving the definition of an adjugate matrix divided by the determinant), alternative methods that rely upon the decomposition of the covariance matrix can be adopted to help mitigate the potential for numerical inaccuracies and ultimately lessen the computational expense.

The LU matrix decomposition method applied in Gibbs and Mackay (1997) allows a square matrix to be decomposed into upper and lower triangular matrices (of identical size), allowing more rapid inversion of these smaller matrices and subsequent multiplication (i.e. $C^{-1} = U^{-1}L^{-1}$). Furthermore, the authors reported that numerical errors were reduced through the adoption of this decomposition. Nevertheless, such decomposition still employs inversion techniques that can prove time-consuming for large datasets. A further alternative is to take advantage of the requirement that the covariance matrix must be symmetric and positive semi-definite, and implement Cholesky decomposition of the covariance matrix instead. The Cholesky decomposition is a special case of LU decomposition that allows the decomposition of a square positive Hermitian matrix into the product of the lower triangular matrix and its transpose (i.e. $C=LL^T$). The use of Cholesky decomposition has been recommended in Rasmussen and Williams (2006) as it has been shown to be both faster and more computationally stable. Furthermore, through the implementation of Cholesky decomposition in development environments such as Matlab, a check on the conditioning of the covariance matrix can

be performed. Therefore, the underlying software may generate a potentially informative error message when the conditioning of the covariance matrix has deteriorated.

The text by Rasmussen and Williams (2006) has provided the following useful guide for implementing the GP predictive equations using Cholesky decomposition:

Applying Cholesky decomposition, we can generate L = Cholesky(**K**). For the calculation of the predictive mean ($\hat{t}_{N+1} = \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{t}_N$) we first simplify this equation into the form $\hat{t}_{N+1} = \mathbf{k}_{N+1}^T \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = \mathbf{C}_N^{-1} \mathbf{t}_N$. Then through substituting the Cholesky decomposition ($\mathbf{C}_N = LL^T$) and solving for $\boldsymbol{\alpha}$, we find $\boldsymbol{\alpha} = L^T \backslash (L \backslash \mathbf{t}_N)$. We can then express the predictive mean as $\hat{t}_{N+1} = \mathbf{k}_{N+1}^T \boldsymbol{\alpha}$. For the predicted variance, we can make further use of the Cholesky decomposition and define $\mathbf{v} = L \backslash \mathbf{k}_{N+1}$, and then express the variance as $\sigma_{N+1}^2 = \kappa - \mathbf{v}^T \mathbf{v}$.

4.5.5) Approximate Implementations of the GP model

In the direct implementation the GP model the overall computational demand can be seen to scale with the size of the training dataset to the order of $O(N^3)$. As a result, the implementation of this direct approach presents a significant difficulty for those working on problems that involve large quantities of data. Furthermore, the computational load required by the method may prove to be beyond that of users with access only to average desk-top computing facilities. Therefore the development of methods aimed at reducing the computational demands of the GP modelling approach have received a great deal of attention and remain a focus of ongoing research.

As discussed previously, the predictive equations of the GP model involve both the storage and inversion of a potentially large covariance matrix. Furthermore, the implementation of the direct method can be seen to revolve around the problem of solving the linear system $(K + \sigma_n^2 I)\mathbf{v} = \mathbf{y}$ for **v**, (Note that for consistency with existing literature we have substituted the $(K + \sigma_n^2 I)$ for the previously used $\mathbf{C}_N$ which assumes Gaussian independent noise, and **y** for targets $\mathbf{t}_N$).

In tackling this problem of reducing the computational demand, a number of different approaches have been developed. Useful reviews of these approximate methods can be found in Seeger (2003), Rasmussen and Williams (2006) and Quinonero-Candela et al. (2007). The review paper by Quinonero-Candela et al. (2007) builds upon that found in Rasmussen and Williams (2006) in providing a unifying view of the techniques used to approximate the GP model for regression. The former paper states that the various options can be generally categorised into two different approaches to the problem:

1) Using Fast matrix-vector multiplication methods (**MVM**) to approximate the direct implementation of the GP model.
2) Using **Sparse** matrix methods to approximate the covariance matrix.

In this section an overview of the main ideas behind these approaches has been provided, rather than a full mathematical exploration.

4.5.5.1) Fast Matrix Vector Multiplications (MVM)

The primary cause of the demanding computational requirements of the direct implementation of the GP model has been isolated as the need for the repeated inversion of the potentially large covariance matrix (or the solution to the linear system $(K+\sigma_n^2 I)\mathbf{v}=\mathbf{y}$ for $\mathbf{v}$). As a result, efficient computational methods aimed at solving this problem and therefore speeding up GP regression have been proposed. The fast MVM methods proposed in Wahba (1995) and Gibbs and Mackay (1997) tackle this problem through the use of iterative methods such as conjugate gradients. The paper by Gibbs and Mackay (1997) takes its inspiration from the methods proposed by Skilling (1993), and provides a detailed resource for the reconfiguration of the GP predictive equations and the maximum likelihood optimisation loss function into expressions that avoid the explicit inversion of the covariance matrix. Overall, the number of iterations of the conjugate gradients method completed can be seen to dictate the computational demand of the fast MVM method. Every iteration of the conjugate gradients method has a computational demand of the order of $O(n^2)$, however an approximate solution can be arrived at if the algorithm is terminated after k iterations, giving an overall computational demand of $O(kn^2)$.

With regard to the overall task of improving the computational efficiency of the GP model, the reviews of Quinonero-Candela et al. (2007) and Rasmussen and Williams (2006) are somewhat dismissive of this option. This is primarily due to the overall computational demand incurred by the deployment of these methods still being seen to scale nonlinearly as $O(n^2)$, and thus not offering the computational savings that are desired. As a result, the fast MVM methods have been stated as being of the most potential benefit for problems where the number of input dimensions is relatively small.

4.5.5.2) Sparse Matrix Methods

The most straightforward method of reducing the computational burden of the GP approach is to restrict the size of the training dataset and therefore reduce the size of the covariance matrix. This most obvious of strategies has been named the Subset of Data (SOD) approach and entails the definition of a subset of the training dataset for use in the construction of the covariance matrix. This approach is to be discussed below, but suffers from the fundamental drawback that through the elimination of training data from the training dataset we are of course potentially throwing away valuable information about the underlying system and therefore compromising the performance of the resultant model.

Therefore alternative methods to "sparsify" the covariance matrix have been proposed. The idea here is somehow to retain the bulk of the information contained in the full training dataset, but reduce the rank (i.e. the number of linearly independent rows) of the resultant covariance matrix so as to facilitate a less computationally demanding implementation of the GP model. These sparse methods are to approximate the full posterior and therefore the predictive equations of the GP model through the use of expressions that involve matrices of lower rank $m < n$ (where m is the rank of the sparse covariance matrix, and n is the rank of the full GP covariance matrix).

In Rasmussen and Williams (2006) the discussion of approximate methods for large datasets begins with a proposal for a method for improving computational efficiency through the eigendecomposition of the GP model kernel. As a result of this eigendecomposition, a method of reducing the rank covariance matrix may then be forthcoming. However the discussion points out that the problem of approximating the

kernel in terms of eigenvalues and eigenvectors is a computationally demanding one in itself, and may therefore negate any resulting benefit in the subsequent implementation of the GP model prediction framework. Therefore, methods aimed at reducing the computational demand of eigendecomposition are proposed as a possible way forward. One such method is the Nyström approximation described below, but before examining the details of the various methods, a discussion of the subset selection procedures used to define each sparse method is required.

4.5.5.3) Subset Selection

In constructing a reduced rank or sparse version of the covariance matrix *K* (note that due to the potential modification of the covariance matrix it may be more correctly termed as the Gram matrix) the first step is the selection of a subset of datapoints. The selection of this 'included' or 'active' subset of data can be seen to be something common to all Sparse methods, where the included latent variables are to be treated exactly by the GP model framework and the remaining variables are to be approximated by a less computationally demanding method. This means that, unlike the previously mentioned Subset of Data (SOD) method, the data not included in the subset is not going to be completely eliminated from the approximation.

This subset of data is to be of size $m < n$, where n is the size of the overall training dataset, and is denoted as *I* (as in 'included' datapoints) in Rasmussen and Williams (2006), with the 'remaining' $n-m$ datapoints then said to form the set *R*. If the training datapoints are then assumed to be ordered in a manner so that the subset *I* appears first, the matrix *K* can be partitioned without loss of generality as the following:

$$K = \begin{pmatrix} K_{mm} & K_{m(n-m)} \\ K_{(n-m)m} & K_{(n-m)(n-m)} \end{pmatrix} \tag{4.18}$$

where the top $m \times n$ block can also be denoted to as K_{mn} and its transpose as K_{nm}.

In the review by Quinonero-Candela et al. (2007) a slightly different perspective is taken where the active set is known as a set of 'inducing' variables. This review paper builds on the previous account by Quinonero-Candela and Rasmussen (2005) that sought to

provide a unified view of the various sparse matrix methods that had been developed. This is achieved through the reinterpretation of the various sparse methods as “exact inference with an approximate prior”, rather than the more common interpretation of “approximate inference with the exact prior”. As a result the ‘effective’ prior being employed by each algorithm can be computed from the analysis of the posterior. The overall objective of this reinterpretation is to provide a means of direct comparison between the various sparse matrix methods.

As only the active set is to be treated fully in the sparse model, the process of determining which datapoints are to be included is critical to the success of the approximation. One possible strategy is to carefully build the subset of data through the manual selection of datapoints based on ‘a priori’ knowledge of the underlying system characteristics. This method of selecting an optimum training data subset can be seen to be particularly in keeping with the system identification process where the pre-processing of empirical data is often an important stage. However, for implementations where ‘a priori’ knowledge is limited, or for complex nonlinearities composed of multiple dimensions, the determination of a suitable subset may become a challenging problem. As a result, a simple strategy such as the random selection of datapoints may be a suitable course of action.

As an alternative, more iterative approaches to the selection of the active set have also been proposed. In particular, ‘Greedy Approximation’ methods have been shown to be of great potential where the active set is selected and updated according to some criterion. Such an algorithm would initiate with an empty active set I with the remaining set R containing all indexed training observations. Then, using an iterative method, each indexed training example is added to the active set in turn and the selection criterion evaluated. If the criterion is met, and the active set can be seen to be further optimised, the training example under review will be included in the active set. As a result, many of these algorithms can be seen to have significant parallels with the ‘active learning’ methods briefly mentioned in Section (2.3.2). Note that the computational expense of considering all training examples with respect to the criterion in one sitting may prove to be prohibitive and therefore working subsets of data may also need to be defined, see Rasmussen and Williams (2006) for a general description of the Greedy Approximation algorithm.

The next question that arises is what kind of selection criteria should be used to determine the active subset of data. Various methods have been proposed including the 'Informative Vector Machine' (IVM) of Lawrence et al. (2003), the 'Informative Gain' criterion of Seeger et al. (2003), the online learning algorithm of Csato and Opper (2002), minimisation of the residual sum of squares as in Luo and Wahba (1997), and maximising the effective posterior instead of the effective marginal likelihood as in Smola and Bartlett (2001). A further method could be the maximisation of the marginal likelihood (i.e. the same optimisation used to identify hyperparameters) with respect to the inducing inputs as described by Snelson and Ghahramani (2006). In addition, the selection of the active set of data can also be incorporated into the existing optimisation of the hyperparameters as in Seeger et al. (2003).

A final aspect to consider in the determination of a suitable active set is that there is no fundamental reason why the subset has to be chosen from the training dataset itself. The review by Quinonero-Candela et al. (2007) states that subset selection from a disjoint of the training dataset may be a viable alternative, and points to the paper by Snelson and Ghahramani (2006) where the discrete selection of training/test cases has been replaced by an algorithm more in keeping with continuous optimisation. After determining a suitable subset of data, we can now turn our attention toward describing some of the various sparse methods that have been proposed.

4.5.5.4) Subset of Data (SoD)

The Subset of Data (SoD) method can be seen to be the most straightforward method of sparse matrix approximation, where an active subset of data m is to be selected from the whole training dataset n. The existing predictive equations and optimisation expressions remain unchanged by this method, resulting in an overall computational demand of $O(m^3)$, where $m < n$. As discussed previously, the Subset of Data approximation method would seem to be fundamentally handicapped in comparison to alternative sparse methods, as training data that is not included in the active subset is simply discarded rather than approximated. However, in comparison to the more sophisticated sparse methods, the computational demand of the SoD method is independent of n. Furthermore, through the use of carefully selected data or the greedy selection methods discussed previously, the resultant active subset to be used by the SoD method can

become more optimised. As a result, the SoD method may still provide a good approximation to a 'full' GP model and should not be wholly discounted in favour of more sophisticated sparse methods.

4.5.5.5) Nyström Approximation

The Nyström approximation method involves the analysis and approximation of the eigenfunctions and eigenvectors of the kernel. The Nyström method is described in Press et al. (1992), and has been proposed as a sparse method for GP regression in Williams and Seeger (2001). This method allows the covariance (or more generally the Gram) matrix K to be approximated by a reduced rank or sparse version $\tilde{K}$ that can then be substituted into the GP predictive equations. By then choosing the number of eigenvalues/vectors to be included in the approximation to be the same as the size of our defined subset I, the Nyström approximation of K can be written as:

$$\tilde{K} = K_{nm} K_{mm}^{-1} K_{mn} \tag{4.19}$$

This approximation $\tilde{K}$ can then be substituted for K in the main GP predictive equations. Note that it is only the matrix K that is to be substituted, the covariance function k is not going to be to be substituted by $\tilde{k}$. The computation demand associated with the method is $O(m^2 n)$ for the required matrix computations, and $O(n)$ and $O(mn)$ for the evaluation of the predictive mean and variance respectively. In the paper by Williams et al. (2002), the experimental results point out that the Nyström method performs poorly relative to other methods when the size of the active set m is small. Furthermore, due to the fact that the covariance function is not completely replaced by an approximation $\tilde{k}$, numerical errors may be encountered.

4.5.5.6) Subset of Regressors (SoR)

The Subset of Regressors (SoR) method takes advantage of an equivalence between the GP model's mean predictor and that of a finite-dimensional generalised linear regression model. This method originates from Wahba (1990) and Poggio and Girosi (1990), and has been adapted for use in Sparse GP models by Smola and Bartlett (2001). Therefore

the SoR model is a finite linear-in-the-parameters model with a particular prior on the weights. For any input x*, the corresponding function value f* is given by:

$$f(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i) \text{ with a prior } \boldsymbol{\alpha} \sim \text{Normal}(\mathbf{0}, K^{-1}) \tag{4.20}$$

In order to formulate an approximation to this model only a subset of regressors are considered so that:

$$f_{SR}(\mathbf{x}_*) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_*, \mathbf{x}_i) \text{ with a prior } \boldsymbol{\alpha}_m \sim \text{Normal}(\mathbf{0}, K_{mm}^{-1}) \tag{4.21}$$

We can then formulate the predictive distribution in the same manner as described in the 'weight-space' interpretation of the GP model (Rasmussen and Williams, 2006), to find the mean and variance:

$$\widehat{f}_{SR}(\mathbf{x}_*) = \mathbf{k}_m(\mathbf{x}_*)^T (K_{mn}K_{nm} + \sigma_n^2 K_{mm})^{-1} K_{mn}\mathbf{y} \tag{4.22}$$

$$Var[\, f_{SR}(\mathbf{x}_*)] = \sigma_n^2 \mathbf{k}_m(\mathbf{x}_*)^T (K_{mn}K_{nm} + \sigma_n^2 K_{mm})^{-1} \mathbf{k}_m(\mathbf{x}_*) \tag{4.23}$$

From these predictive equations we can see that in contrast to the SoD method, the SoR method is to employ all *n* datapoints of the training set in the approximation. However, a major disadvantage of the SoR method is that due to its basis upon a linear-in-the-parameters model, the GP model becomes degenerate. Under the unifying view of Quinonero-Candela et al. (2007) where methods are described in terms of approximate priors, the degenerate nature of the SoR model can be seen to restrict the variety of possible functions that will be plausible under the posterior. Furthermore, as the SoR model can be seen to have only *m* degrees of freedom, this implies the restriction that we can only draw *m* linearly independent functions from the prior, with subsequent $m+1$ functions being a linear combination of the previous functions.

The main consequence of this degeneracy is that the resultant predictive distributions can become unreasonable. For covariance functions that decay as the distance between inputs

increases, we would expect that the predictive variance to increase as the distance between inputs is increased, thus indicating an increase in uncertainty. Unfortunately, the due to the restrictions placed by the SoR model on the approximate prior over functions, the predictive distribution can in some cases have no prior variance. This can result in predictive distributions in which the variance tends to zero for far apart inputs (i.e. the opposite of what would be desired). Overall, whilst the SoR method can be seen to be a useful approach in terms of approximating the GP mean prediction, the accompanying predictive variances can at best be described as overconfident, and at worst absurd. The computation demand associated with the SoR method is $O(m^2n)$ for the initial matrix computations, and $O(m)$ and $O(m^2)$ for the evaluation of the predictive mean and variance respectively.

4.5.5.7) Further Sparse Methods

A number of other sparse matrix methods have been proposed in recent years that can be seen to overcome the main weakness of the SoR approximate method, i.e. the greatly reduced scale and therefore usefulness of the variance output. Notable methods discussed in the review by Quinonero-Candela et al. (2007) include the Deterministic Training Conditional (DTC) Approximation (that is equivalent to the method of Projected Latent Variables (PLV) proposed in Seeger (2003) and also discussed in Rasmussen and Williams (2006) as Projected Process Approximation (PPA)), and the Partially Independent Training Conditional (PITC) and Fully Independent Training Conditional (FITC) (also known as the Sparse Pseudo-input Gaussian Process (SPGP) which was proposed by Snelson and Ghahramani (2006) and further discussed in Snelson (2007)).

These different sparse methods are more complex and without fully exploring the unified view of sparse methods proposed in Quinonero-Candela et al. (2007), where each sparse method is interpreted through their effective priors, it is difficult to provide a detailed explanation of these methods. In essence, the DTC and SoR sparse approximations can both be seen to impose a deterministic relationship between the training and inducing latent variables that results in inducing training conditionals where the covariance matrix has been set to zero. For the PITC and FITC sparse methods, the approximation to the training conditional is to include a portion of the true covariance matrix (a block-diagonal in the case of PITC) and set the remaining elements of the matrix to zero.

Overall, the computational demand of these different methods has been found to be similar at $O(m^2n)$. As a result, it is difficult to draw conclusions as to which method would be the preferred choice for a given application.

4.5.6) Which Approximation Method?

As many of the sparse methods have been estimated as having a similar computational demand, it does not appear immediately obvious as to which method should be preferred. Furthermore, the reviews of Quinonero-Candela et al. (2007) and Rasmussen and Williams (2006) also present final conclusions that are somewhat inconclusive and recommend further empirical investigations. However, such an evaluation of the various methods on offer can be seen to be a challenging task where many contributing factors can determine the suitability of a particular approximate method, together with numerous possible measures of approximation accuracy and computational efficiency. Issues such as the complexity of the underlying function, dimensionality of the model input, and the degree of noise present on the targets have been identified as being potentially influential in determining the suitability of a chosen approximation (e.g. the SoR method is degenerate which may limit flexibility). For assessing the predictive performance of an approximation, measures such as such as mean square error or negative log likelihood may be utilised, and for computational performance the time taken for testing, pre-computation (i.e. operations required before test predictions are made), and hyperparameter learning may all be useful measures of efficiency. Overall, some approximate algorithms may be seen to perform better under some criteria, and comparatively worse under others.

For the more complex varieties of sparse matrix methods proposed (i.e. discounting the SoD method), as the computational demand would seem to be similar, the most important distinguishing feature between them would appear to be how the predictive variance is to be treated. Therefore, the selection of an appropriate approximate method could be based upon the need for an accurate representation of the variance of the prediction. In some applications, this measure of prediction uncertainty may be considered superfluous (see Seeger (2004) for a discussion as to the worth of the variance output from a machine learning perspective), and it is the predictive mean that we are interested in. As a result, the SoR method may become a strong candidate due to the

relative straightforward nature of the approximation. If the predictive variance is to be of great importance, the more sophisticated approximate methods of the PP/DTC, PITC and FITC should be considered. However, before making any concrete recommendations it must be pointed out that due to the vagaries presented by current empirical evidence, a more simple SoD method may still prove to be competitive. Referring to the results of the empirical example presented in Rasmussen and Williams (2006), the measure of model error (for a fixed subset size) can be seen to be reduced if more complex sparse methods (SoR and PP/DTC) are adopted in comparison to the SoD method. However, the results point to the fact that the overall 'mean runtime' (indicative of the computational load) associated with the more complex sparse methods is substantially higher than the equivalent SoD method. As a result, a more simple SoD approximate constructed from a larger subset of data can remain competitive in both predictive accuracy and computational load, than a more complex sparse method composed of a smaller subset of data. Note that the authors do not present this evidence as a definitive result, but merely as indicative of the problem of assessing the relative performance of competing approaches.

In the selection of an appropriate approximation a further consideration is the intended implementation or application of the GP model. In particular a significant proportion of the computational demand of the GP model can be seen to be take place before predictions may actually be computed (i.e. the pre-computation involving the inversion of the covariance matrix and application to the target vector, and the optimisation of the hyperparameters). In some applications it is reasonable to assume that the time taken for pre-computation and training is not of great concern, and it is the speed of prediction that is of overriding importance (e.g. real-time implementations involving a trained GP model). In contrast, other applications may require that the identified GP model must be adaptable, where the speedy re-training of the hyperparameters takes precedence over the speed of prediction (e.g. implementations where relatively few predictions are computed, but the model must be modified or updated repeatedly). As a result, it is the practical demands of the problem under investigation that must dictate whether or not any approximate methods are required, and ultimately which method would be preferable.

4.5.6.1) Implications for System Identification

Returning to the specific demands of system identification, further comment on the overall usefulness of these approximate methods is worthwhile. The primary objective of these methods are to reduce the computational demand of the GP model, either through optimising the covariance matrix through sparse matrix methods, or through reconfiguring the mathematical implementation of the model through fast (MVM) methods. However, none of these methods are aimed at tackling the matrix conditioning problems also discussed. Therefore, regular system identification issues such as the design of experiments and collection of data that can be seen to directly influence the conditioning of the covariance matrix must also be considered before any approximate methods are employed.

In the discussion concerning the matrix conditioning aspects of the GP model the need for significant pre-processing of the training set was identified as being likely for system identification problems. Furthermore, we can see that through the pre-processing of the training set and potentially removing problematic data we are in fact adopting a Subset of Data (SoD) sparse approximate method even before actively considering the computational load. As a result it may be the case that through the pre-processing of the training data, the final subset of m data to be employed may be of a size that is computationally feasible. This in turn may diminish the need for more exotic approximate methods to be employed. In essence, the implementation strategy should first endeavour to pre-process the training data to tackle any conditioning problems, before then considering whether or not any further approximate methods are required.

The potential need for the pre-processing of the training data to meet the conditioning requirements of the GP model can also be seen to suggest that a more manual or “hands-on” approach to subset selection may be preferable to the use of more random or iterative selection methods where datapoints are greedily added upon evaluation against some criteria. In this way, any prior knowledge of the underlying system can be utilised in avoiding conditioning problems as well as meeting the demands of the system identification task and regulating the overall size of the subset of data. Furthermore, through the manual pre-processing of the training dataset any specific issues such as ensuring the inclusion of particularly important regions of operating space can be tackled

directly. After all, the most important issue to be borne in mind is whether or not any sparse method employed can be seen to impact detrimentally on the overall integrity of the training data to be employed in the construction of a GP model. Therefore, in the first stages of pre-processing, a more manual approach to subset selection of the training dataset would seem to be a sensible strategy. After this stage, a more iterative selection approach could then be employed to further improve the subset. For example, by carefully adding more points to increase the accuracy of the model, or through carefully removing points to increase the computational efficiency.

Nevertheless, for some applications such a detailed manual pre-processing of the training dataset may not be viable. In cases where the size of the initial training dataset is very large, or where prior knowledge is limited, a random or iterative approach to subset selection should be considered. However for the relatively simple nonlinear dynamic systems considered in the next chapter, this thesis is to adopt the subset of data (SoD) approximation where the selection or pre-processing of training data is to be performed by hand. As a result, the general guideline that for the direct implementation of the GP model the training dataset should contain (n<1000) is to be followed.

4.5.6.2) Further Possibilities

As has been discussed previously, the direct implementation of the GP model has been shown to be problematic due to the need for the inversion of a potentially large covariance matrix. To tackle this problem, the sparse matrix methods discussed in the previous section attempt to reduce the computational footprint of the covariance matrix through approximation, whilst endeavouring to retain as much information as possible in the modified covariance matrix. In tackling the implementation difficulties of the GP model a further two possibilities have been proposed and are worthy of discussion.

4.5.6.2.1) Multiple GP models

An obvious alternative to this problem of squeezing as much information into as small a space as possible is to split up the information into number of smaller spaces. In essence, the idea would be to follow the precedent laid out in other forms of mathematical modelling where a ‘divide and conquer’ approach is taken, and a multiple model or

network of models are specified in place of a single global model. Such a scheme could follow the previously discussed Operating Regime approach, see Murray-Smith and Johanson (1997), where the operating range of the system under investigation is partitioning into a number of local regions. Using empirical data collected from each local operating region a local GP model could then be identified. As each individual GP model would be identified from a subset of the overall set of training data, the resultant size of each covariance matrix will then be significantly smaller and therefore easier to implement.

The next stage to consider is how this group of local GP models are to be combined into a global representation of the system. In particular, the switching between individual GP models is something that must be considered carefully, as in addition to the prediction estimate each GP model is to provide a predictive variance. The paper by Rasmussen and Ghahramani (2002) proposes a scheduler or manager that probabilistically assigns points to each local expert model. The paper by Shi et al. (2003) utilises a probabilistic approach with a hierarchical arrangement to structure a mixture of GP models. In both of these papers, the inference required the use of MCMC methods to maintain the validity of the Bayesian framework.

A further example that specifically aims to implement a ‘local model network‘ comprised of local GP models is the paper by Gregorčič and Lightbody (2007). In this paper a global GP utilising the squared exponential covariance function is first identified to divide the operating range of the system into local regimes composed of clusters of training data. This is achieved by examining the variance output of the global GP model where a low variance is taken to be indicative of a suitable region for the identification of a local model due to the presence of sufficient data, thus providing the centres for local validity functions. Subsequently, a linear covariance function is then employed to identify a local linear model in each operating regime. The problem of ensuring that the variance output of this multiple GP model remains indicative of model uncertainty is solved through blending the different local models through their parameters, rather than blending the multiple models through their respective outputs. Overall, this prospect of employing the GP model within the well developed multiple modelling strategy would appear to be a promising avenue for further research.

4.5.6.2.2) Derivative Observations

A further alternative to the previously discussed methods of approximation is to incorporate derivative observations into the GP model, as in Leith et al. (2002) and Solak et al. (2003). The idea behind this approach is to summarise any training data that is close to an equilibrium point by defining a local linear model to cover this region of operating space. Therefore, standard linear regression solutions (i.e. linear least-squares) can be used to estimate a derivative observation from perturbation data in the vicinity of an equilibrium point. After obtaining a set of derivative observations, these can then be combined with the existing training data (function observations) in the covariance matrix.

Therefore, through the identification of a number of derivative observations we can potentially summarise a significant proportion of the training data very concisely, and utilise the remaining space (i.e. the majority of the covariance matrix) for the inclusion of function observations collected in off-equilibrium regions. Therefore, this strategy of utilising derivative observations for equilibrium regions and function observations for off-equilibrium regions, can also be thought of as being in keeping with the divide and conquer methodology of the multiple model approach, where the derivative observations can be interpreted as local linear models. This has particular relevance in the identification of nonlinear dynamic systems where real applications can often be seen to exhibit prolonged periods of operation near to equilibrium points, with off-equilibrium data being comparatively sparse. As a result, the benefits of incorporating derivative information can be seen to be particularly relevant to the system identification and engineering applications considered in this thesis, and are therefore to be discussed in greater detail in the next chapter, see section (5.4).

5) Nonlinear Dynamic System Identification with GP models

In the previous two chapters, the mathematical background and computational implementation of the GP modelling approach has been discussed. In this chapter, we are to investigate the application of the GP modelling approach for the specific task of identifying nonlinear dynamic systems. Notable extensions to the GP framework that are particularly relevant for nonlinear system identification (Uncertainty Propagation and Derivative Observations) are also discussed. To support this investigation, a number of simulated example systems are first identified with the GP modelling approach. Finally, a number of real laboratory-scale nonlinear systems are to be identified from empirical data. Through utilising these different examples, the implementation of the GP model from a practical engineering perspective can therefore be discussed. Furthermore, rather than focussing solely on judging the accuracy of the identified GP models (as is common in machine learning applications of the method), the robustness qualities of the identified models are also to be assessed. As the main overall objective of the system identification process is to provide both an accurate and robust approximation to the underlying system (especially if the model is to be ultimately used for control purposes, see Section (2.2.1)), the ability of the identified GP models to represent the full range of behaviour exhibited by the example applications must be judged carefully.

5.1) Background of GP models in System Identification

The recent interest in the GP modelling approach as a method for nonlinear system identification can be seen to originate from the ideas presented in Murray-Smith et al (1999), Leith et al (2000), and Leithead et al (2000). In these papers, a non-parametric modelling approach was proposed for use in the identification of local models (as part of a multiple model type structure) in off-equilibrium regions. In such operating regions, the popular strategy of identifying local linear models has been shown to be fundamentally limited, as discussed in Shorten et al (1999). Much of the difficulty associated with representing such off-equilibrium regions is due to the potential absence of prior knowledge, coupled with a lack of sufficient data with which to identify local models. To combat this lack of prior knowledge, non-parametric empirical modelling methods have

therefore been proposed as a solution. However, due to the scarcity of the available data, such data-based modelling approaches may also struggle to identify a meaningful description.

The reason why the GP model offers a viable alternative is the fact that due to the Bayesian probabilistic nature of the approach, the relative scarcity of the empirical data (or density of data) can be reflected in the approximation through the variance output. In addition, the GP model is to directly use whatever data is available in computing each prediction (i.e. the covariance matrix is directly defined from applying the covariance function to the training data), rather than rely solely on parameters defined through optimisation. As a result, the GP modelling approach has been shown to perform well in identifying models from small datasets as the number of structural parameters (hyperparameters of the covariance function) to be identified is typically less than that of other complex learning systems (see Kocijan et al (2003a) for a practical comparison of the GP modelling approach with a Neural Network alternative). Further good general sources of information on applying the GP modelling approach toward dynamic system identification problems are Gregorčič and Lightbody (2002), Murray-Smith et al. (2002), Kocijan et al. (2003b), Wang et al (2005), Ažman and Kocijan (2007), Kocijan and Ažman (2007), and Kocijan and Likar (2007). In Gray et al (2003), and Thompson and Murray-Smith (2006), some of the more practical implementation issues associated with the GP model are also discussed, with some of this research forming the basis of this thesis.

A further motivating factor behind the GP modelling approach are the possibilities that exist for the incorporation of the GP methods into the well established ‘divide and conquer’ multiple modelling strategy discussed previously. As stated above, the paper by Murray-Smith et al. (1999) proposed the use of local GP models to identify off-equilibrium operating regions, and then combining these local GP models with local linear models used to identify equilibrium regions. A further strategy in keeping with this desire to retain the local linear modelling approach and combine it with the GP model is the incorporation of derivative observations into the GP modelling approach (as mentioned in Section (4.5.6.2.2)). This extension to the GP modelling approach takes advantage of the fact that as differentiation is a linear operation, the derivative of a GP remains a GP. As a result, as long as the derivative of the covariance function is employed, these derivative observations can be handled by the same predictive framework as the normal functional observations.

As derivative observations can be thought of as equivalent to linearisations about an equilibrium operating point, through incorporating derivative observations we can develop a global model built out of multiple local linear models that can be blended almost seamlessly with any off-equilibrium functional observations. Whilst in certain applications direct access to derivative observations may be available (e.g. sensors measuring speed and acceleration), it is also possible to generate them through the application of standard linear regression techniques to any available equilibrium functional observations. An important outcome of adopting this approach is that significant improvements in the computational efficiency of the overall GP modelling approach may be realised. By using computationally efficient linear regression techniques to identify derivative observations from (commonly abundant) equilibrium data, the more computationally expensive standard GP methods can then be reserved for the more scarce off-equilibrium data. This divide and conquer strategy based on the combination of functional and derivative observations is discussed in more detail in a forthcoming section, and previous detailed sources include Leith et al (2002), Solak et al (2003) and Kocijan et al (2003c).

This overall synergy between the methods employed in the multiple model approach and that of the GP modelling approach is further described in the review by Gregorčič and Lightbody (2004, 2008). Furthermore, as discussed in section (4.5.6.2.1), another paper by Gregorčič and Lightbody (2007) proposed another interesting but related alternative where local linear GP models (in this case linear covariance functions are used) in the development of a local model network model structure. Further notable contributions to the field of GP modelling for dynamic system identification include the consideration of non-Gaussian noise models as discussed in Murray-Smith and Girard (2001). In the previous section discussing covariance functions (Section (4.3)), the general form of covariance function assumed additive independent identically distributed Gaussian noise. However, in the identification of real systems it is not unlikely that noise is dependent on other variables. A further important development in GP models for system identification purposes is the development of more a complex multi-step ahead prediction method where the uncertainty over one prediction can be propagated to the next prediction. This ‘Uncertainty Propagation’ or ‘Prediction with Uncertain Inputs’ was first proposed in Girard et al (2002) and expanded on in Girard (2004), and will be discussed in more detail in a forthcoming section.

5.1.1) Control with GP models

In this thesis the problem of applying automatic control to nonlinear dynamic systems identified by GP models has not been investigated. However, it is worthwhile to provide a brief overview of the existing research into this aspect, as one of the primary drivers behind the development and maturation of any system identification approach is whether or not the methods are well suited to solving existing control problems or if new control design strategies are made possible.

One application of the GP model used in the context of control is the development of Nonlinear Model Predictive Control (NMPC) strategies, as described in Kocijan and Murray-Smith (2004) for a Ph Neutralisation process. The general idea behind MPC strategies is to employ an explicit model of the process to predict the future behaviour of the process up to a chosen prediction horizon, and then optimise the manipulated variable against some cost function to obtain an optimal future process response. This input information is then directed to the process, and the control horizon is then completed before the whole sequence is repeated again. For more general information on NMPC, see the reviews by Henson (1998), Qin and Badgwell (2000) and Allgöwer and Zheng (2000). In the paper by Kocijan and Murray-Smith (2004), the interesting development is that the NMPC algorithm is implemented with constraints placed on the variance output of the GP model. Therefore, the process can be controlled in a robust manner that prohibits the operation in regions of operating space that the GP model deems ‘unsafe’ as designated by a high variance output. This exploitation of the variance output of the GP model for control purposes is one of the main attractive features of the GP modelling approach, as variance information is not normally so readily available. Further papers that have also investigated control using GP models include Murray-Smith et al (2003) where the variance output is used to implement ‘cautious’ control, Murray-Smith and Sbarbaro (2002), Sbarbaro and Murray-Smith (2005), and Likar and Kocijan (2007). Furthermore, the incorporation of derivative observations into the control system is discussed in Kocijan and Leith (2004), and the application of ‘Fault Detection’ using GP models is discussed in Jurirčič and Kocijan (2006).

5.2) Applying the GP Model

In applying the GP modelling approach, we are to assume a Multiple-Input-Single-Output (MISO) model structure, where the inputs $\mathbf{x}$ are to be mapped to a single output y. For the identification of dynamic systems, we are interested in utilising information from previous states to provide information about future states. Therefore, regressors such as previous inputs and outputs are important quantities that we must build into our model. As a result, the simple NARX (Nonlinear ARX) model structure discussed in Section (2.5.3.1) can be seen to be an appropriate choice for the overall structure of the GP model:

$$y(k) = f(y(k-1), y(k-2), \ldots, y(k-L), \quad u(k-1), u(k-2), \ldots, u(k-L)) + \varepsilon \qquad (5.1)$$

Where ε is white noise, and k is used to denote a consecutive number of data samples.

The selection of appropriate regressors is a key stage of the optimisation of any model structure. In order to make this selection prior knowledge of the system can prove to be an invaluable resource in tackling this problem. Furthermore, through a model testing and validation stage it may become clear which inputs are most important. However, a further facility of the GP modelling approach that can be used in the selection of inputs is the Automatic Relevance Detection (ARD) feature of certain covariance functions. This feature was briefly discussed in Section (4.3.1.1) in relation to the most popular Squared Exponential covariance function. The ARD facility allows the relative importance of each input dimension to be assessed through the relative size of the corresponding trained hyperparameter, and therefore allows any redundant or non-contributing inputs to be identified and then eliminated from the model structure.

5.3) Multi-Step Ahead Prediction

After the identification of the GP model hyperparameters, the next stage to consider is how the model is to be employed for prediction. A standard approach to implementing multi-step (or *k*-step) ahead prediction is to make repeated one-step ahead predictions up to the desired prediction horizon, whilst all the time feeding back the predictive mean (model output) as part of the model input. This ‘iterative’ approach can be contrasted with a

'direct' approach where the model is designed to predict a certain number (l) of steps into the future. The direct approach suffers from the requirement that the model must include all the required inputs to begin with (i.e. $u(k), u(k+1), \ldots, u(k+l-1)$), thus resulting in an increase in the dimensionality of the input space. Furthermore, such a model can only predict exactly l steps ahead and cannot be readily employed in applications where previous outputs are required, thus restricting the flexibility of such an implementation. However, an important drawback of the iterative one-step ahead prediction method is that it is an approximation, where the prediction relies on previous predictions that may result in an accumulation of prediction error. Nevertheless, this prediction method is the standard approach used in most modelling problems.

5.3.1) Uncertainty Propagation

An alternative method of iterative multi-step ahead prediction has been proposed for use in the GP modelling approach where the uncertainty or variance over each prediction is fed back along with the predictive mean at each time step. In this method the input at which we wish to calculate the prediction becomes a normally distributed random variable, therefore allowing the uncertainty over each prediction to be propagated onto subsequent predictions by updating this input random variable. The result of adopting this strategy is that the variance over each prediction can potentially be made more informative, resulting in less constrained (or wider) error bars where the model has been asked to repeatedly predict in regions where the amount of training data is limited (i.e. a previous prediction with a high corresponding variance (high uncertainty) can be taken into account when calculating subsequent predictions). This method is discussed in more detail below and was first proposed in Girard et al (2002) and is expanded on in Girard (2004). In this section a summary of the overall method has been provided. A full mathematical derivation of this extension can be found in Girard (2004), and a slightly more concise version is also given in Kocijan et al. (2003c).

Firstly, given a set of training data $D = \{\mathbf{x}_i, t_i\}_{i=1}^{N}$, we are to employ a zero-mean GP with covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$ to model the input/output relationship $t_i = y_i + \varepsilon_i$, where $y_i = f(\mathbf{x}_i)$, ε_i is white noise of zero-mean and variance v_t and inputs $\mathbf{x}_i$ are noise free. As discussed previously, given a new test input $\mathbf{x}$ (note that the previously used notation of

$\mathbf{x}*$ is replaced for simplicity), the predictive distribution of the corresponding output $y = f(\mathbf{x})$ can be readily obtained using the previous mean and variance predictive equations (3.40) and (3.41). Rewriting these equations in the same form as Girard (2004), and adopting $\beta = \mathbf{K}^{-1}\mathbf{t}$, the predictive equations become:

$$\mu(\mathbf{x}) = \sum_{i=1}^{N} \beta_i C(\mathbf{x}, \mathbf{x}_i) \tag{5.2}$$

$$\sigma^2(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - \sum_{i=1}^{N} K_{ij}^{-1} C(\mathbf{x}, \mathbf{x}_i) C(\mathbf{x}, \mathbf{x}_j) \tag{5.3}$$

This predictive distribution can also be described by: $p(y|D, \mathbf{x}) = N_y(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$

At this point we wish to consider a new input that is corrupted by noise (i.e. an uncertain input), such that $\mathbf{x} = \mathbf{u} + \boldsymbol{\varepsilon}_{\mathbf{x}}$ where $\boldsymbol{\varepsilon}_{\mathbf{x}} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{x}})$. Therefore, the input can now be considered a random variable that is normally distributed. In order to make a new prediction at this random input $\mathbf{x} \sim N(\mathbf{u}, \boldsymbol{\Sigma}_{\mathbf{x}})$, the existing predictive distribution must be integrated over this new input distribution:

$$p(y|D, \mathrm{u}, \boldsymbol{\Sigma}_x) = \int p(y|D, \mathbf{x})\, p(\mathbf{x}|\mathbf{u}, \boldsymbol{\Sigma}_x) d\mathbf{x} \tag{5.4}$$

As $p(y|D, \mathbf{x})$ is a nonlinear function (as given by equation (3.8)) of $\mathbf{x}$, the new predictive distribution $p(y|D, \mathrm{u}, \boldsymbol{\Sigma}_x)$ is not Gaussian and cannot be readily integrated without resorting to methods of approximation. In Girard (2004), a number of different approximation strategies are discussed and can be broadly categorised into either **numerical** or **analytical** approximations. The proposed numerical approximation relies on the use of Markov-Chain Monte Carlo (MCMC) techniques with which to sample $\mathbf{x}^t$ from the Gaussian input distribution, $\mathbf{x} \sim N(\mathbf{u}, \boldsymbol{\Sigma}_{\mathbf{x}})$, whilst the analytical approximations are dependent on the choice of covariance function The Gaussian (Squared Exponential) and linear covariance functions are shown to result in integrals that may be computed exactly, thus allowing the **exact** mean and variance to be calculated. In other cases, an approximation to these integrals is proposed where a Taylor approximation to the selected

covariance function is utilised. From this approximation to the covariance function, an **approximate** mean and variance of the predictive distribution can be obtained.

5.3.2) When to use Uncertainty Propagation?

The uncertainty propagation extension to the GP modelling approach has been primarily discussed and implemented for the identification of nonlinear dynamic systems, and especially for the purposes of control. In the literature dedicated to machine learning, the application of uncertainty propagation for GP models is not given much consideration. Therefore, as this proposed extension can be seen to add a further level of complexity it is worth discussing when uncertainty propagation is best employed.

The main outcome of including the uncertainty propagation extension is an overall 'flattening' effect on the output predictive distribution. Therefore, in comparison to the standard or 'naïve' implementation of the GP model, the predictive distribution is wider (increasing the variance) and the mean value can become less pronounced. As the predictive distribution is wider (and may also be less uniform Gaussian shape if the numerical Monte-Carlo approximation is calculated), the location of the mean of the predictive distribution can be found to be slightly different to that found with the 'naïve' implementation. In the examples presented in Girard (2004), this discrepancy between the means found in the naïve and non-naïve cases is not normally huge, and at some points in the prediction horizon either implementation may be more accurate (i.e. closer the recorded output). Therefore, it is difficult to argue for the inclusion of uncertainty propagation purely in terms of improving the quality of the mean prediction.

Where the uncertainty propagation extension does offer potential advantages is the effect on the variance prediction. As the predictive distribution is wider, the variance output is therefore boosted and is thus less constrained to the mean. For systems where the model is found to represent the data very accurately, increasing the scale of the variance output is perhaps not something that appears to be particularly worthwhile. However, for systems that are less accurately modelled, if the model is to be tested on a long prediction horizon and previous outputs are to be used as future inputs, it is clear that a growth in uncertainty over the predictions is to be expected. Furthermore, despite the best intentions, the potential remains for certain areas of operating space or test conditions not to be reflected in the

training data of the model (especially in a method where pressure exists to reduce the size of the training set). In addition, for the identification of real-systems the potential exists for unexpected disturbances or some low-level non-stationary behaviour to introduce some differences between previously recorded and test behaviour. Therefore, it is perfectly plausible for the model to make mean predictions based on evidence in the training dataset that are not as accurate as the variance output would suggest (i.e. the variance output (depicted as error-bars) does not fully envelope the real recorded response). Therefore, through the use of uncertainty propagation, the boosted variance output will have a far greater chance of enveloping the real response.

Fundamentally, this potential discrepancy between test and training data is something that all data-driven modelling approaches have to deal with and is known as the generalisation ability of the model. However, whereas most modelling approaches have only an output prediction to consider, the GP modelling approach also has a variance output that should ideally reflect the potential error in the model. Therefore, reflecting this growth in uncertainty through the variance output of the model allows the GP model to become more informative. Furthermore, it is worth noting that this growth in the uncertainty as the prediction horizon extends is not unbounded or exponential, the capability exists to 'catch' the system (reducing the variance) at later test points as reported in Girard (2004).

Another aspect to consider regarding the use of uncertainty propagation is that the method adds further level of complexity to the predictive framework of the approach, and thus potentially adds further computational expense. Overall, whilst the expressions for mean and variance are slightly more complex, and a further input covariance matrix (defined by the size of the input vector) must be computed, as the size of the main covariance matrix is not increased and that the repeated inversion of this potentially large matrix is the main computational bottleneck, the additional computational expense of implementing uncertainty propagation does not appear to be too much of a problem. Nevertheless, whilst no additional large-scale matrix inversion is required, it is likely that by including the propagation of uncertainty the evaluation speed of the GP model will be further slowed. This has implications for the on-line application of the GP model, and with the exception of a small number of papers outlining model-predictive control with GP models (see Section (5.1.1)), there have been little dedicated investigations into the real-time implementation of the GP model (with and without uncertainty propagation) and how it compares to other

black-box methods. Such a comparison would be a worthwhile future direction for research as it may more clearly define the types of problem in which the GP model is most appropriate (e.g. RBF networks are typically found to be slower to evaluate than MLP networks, a significant disadvantage in certain applications).

Overall, the concept of taking into account the uncertainty of the input (and propagating output uncertainty to subsequent predictions) would seem to be eminently sensible for applications where time-series data is to be modelled. Furthermore, due to the probabilistic nature of the GP model, implementing this consideration of input uncertainty is something that is more feasible than in other modelling approaches. However, most modelling approaches or implementations of multi-step ahead prediction do not seek to include the uncertainty over the input, and it is merely understood that the accuracy of the model may reduce as the prediction horizon is extended. Therefore, the uncertainty propagation can be seen to be a useful but perhaps unnecessary extension if only the mean prediction is to be used. But if the variance is to be actively employed in some manner, such as in the design of control systems (see Section (5.1.1)), the uncertainty propagation may prove to be an important addition. As this thesis is investigating the use of GP models for identification purposes, rather than actively seeking to employ the variance output, we are more interested in the accuracy of the mean predictions. As a result, the application of this propagation of uncertainty extension has not been a priority, and the examples investigated have employed the standard or ‘naïve’ implementation of the GP model.

5.4) Derivative Observations

The concept of incorporating derivative observations into the GP model framework has been previously discussed in Section (5.1). Overall, the proposal is particularly interesting for system identification purposes as it offers a method to efficiently include derivative information (either directly available from data, or generated from identified linearisations) into the GP modelling approach. The main advantage of this proposal is the compatibility with the divide-and-conquer strategy of other multiple model approaches, where local linear models (in the form of derivative observations) can be combined with functional observations to form a global representation.

Furthermore, in the identification of real nonlinear systems, it is often the case that much of the available empirical data is found close to various equilibrium operating points, with the availability of off-equilibrium transient data being typically scarce (see Section (2.5.6.1.2) for more discussion on this point). As one of the main difficulties of the GP modelling approach is the heavy computational demand associated with inverting the covariance matrix (the size of which is dictated by the size the training dataset), any method that can reduce this demand is worthy of consideration (e.g. the sparse matrix methods discussed in Section (4.5.5)). The incorporation of derivative observations is an attractive extension as the typically abundant equilibrium empirical data can be summarised using derivative observations identified from computationally efficient linear optimisation, thus leaving the remaining off-equilibrium data to be treated as normal function observations by the GP model.

In addition, as discussed in Section (4.5.2), the presence of prolonged periods of steady-state data can have a negative effect on the conditioning of the covariance matrix. Therefore, as steady-state response data can often result from operating near to equilibrium points, summarising such data in the form of a derivative observation through local linearisation would appear to be an attractive alternative to deleting this problematic data all together.

5.4.1) Identifying Derivative Observations from Data

Whilst in certain applications derivative observations may be directly available from empirical data, it is also possible to identify them from applying simple linear regression techniques to the training data. Therefore, derivative observations can be generated for any system using linearisation around suitable operating points. In order to perform linearisation, small signal or perturbation data close to this operating point is required as local linearity is only guaranteed near to the defined operating point of continuous systems. Although it is possible to identify linearisations at any point, as with other modelling approaches based on local linear models (e.g. Local Model Networks) it is normal to identify linearisations at equilibrium points. Equilibrium operating points are important when considering the stability of the system, which therefore has implications for control purposes.

The linearisation at an equilibrium operating point can be achieved through applying the Taylor series approximation (i.e. a function at $x = a$ can be approximated by $y = f(a) + f'(a)(x - a)$, for a 1st order Taylor approximation where the higher order terms can be ignored by ensuring small scale perturbations from a). Therefore, the linearisation involves the calculation of the slope or gradient (derivative) of the linear model $\hat{\boldsymbol{y}} = \boldsymbol{X\theta}$, through applying standard linear regression (i.e. gradient can be found by $\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, variance from $\sigma^2 = \frac{1}{N}(\hat{\boldsymbol{y}} - \boldsymbol{X}\hat{\boldsymbol{\theta}}^T)^2$, and local linear covariance matrix from $\Sigma = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$).

5.4.2) Gaussian Process Derivatives

Fundamentally, as differentiation is a linear operation, the derivative of a Gaussian Process remains a Gaussian Process. Therefore, in order to incorporate derivative observations, the covariance function to be employed for function observations must be differentiated, and this derivative covariance function used to handle the derivative observations. For the most popular Gaussian (Squared Exponential) covariance function, the existing covariance function relating any two data points in the case of two **functional** observations is:

$$C(\mathbf{x}_i, \mathbf{x}_j) = v \exp\left[-\frac{1}{2} \sum_{d=1}^{D} w_d (x_{i\,d}^d - x_j^d)^2 \right] \tag{5.5}$$

For the case of **mixed** set of derivative and functional observations:

$$C(\frac{\partial \mathbf{x}_i}{\partial x_i}, \mathbf{x}_j) = -v w_d (x_{i\,d}^d - x_j^d) \exp\left[-\frac{1}{2} \sum_{d=1}^{D} w_d (x_{i\,d}^d - x_j^d)^2 \right] \tag{5.6}$$

In the case of two **derivative** observations, (where $\delta_{e,d}$ is a Kroneckor operator between the e^{th} component derivative in $\mathbf{x}_i$ and the d^{th} component derivative in vector $\mathbf{x}_j$):

$$C(\frac{\partial \mathbf{x}_i}{\partial x_i}, \frac{\partial \mathbf{x}_j}{\partial x_j}) = v w_e (\delta_{e,d} - w_d (x_{i\,d}^e - x_j^e)(x_{i\,d}^d - x_j^d)) \exp\left[-\frac{1}{2} \sum_{d=1}^{D} w_d (x_{i\,d}^d - x_j^d)^2 \right] \tag{5.7}$$

In the existing literature, only the Gaussian covariance function has been considered for this extension to the GP model. However, it is clear that any covariance function that is differentiable may offer a suitable alternative. It is also worth noting that although the covariance function is altered through differentiation, no new hyperparameters are defined so the optimisation procedure is unchanged by this extension. As a result, the dataset used for training need not include the derivative observations.

Furthermore, although we are to employ a derivative covariance function to handle the derivative observations, the overall covariance matrix is still to be populated in a manner that allows the existing GP predictive equations to be applied for output predictions. In addition, the previously discussed proposal for uncertainty propagation is something that can also be incorporated with the derivative observations extension. In this next section we will briefly cover the incorporation of derivative observations in the standard or 'naïve' multi-step ahead implementation of the GP model, but full details of incorporating combining uncertainty propagation and derivative observations can be found in Kocijan et al. (2003c) and Girard (2004).

5.4.3) Incorporating Derivative Observations

Using the same notation as Kocijan et al. (2003c), the incorporation of derivative observations can be achieved in the following manner. A possible for the grouping of the data in the input matrix **X** and the target vector **t** is:

$$\mathbf{X} = \begin{pmatrix} \mathbf{Y}_{oeq} & \mathbf{U}_{oeq} \\ \mathbf{Y}_{eq} & \mathbf{U}_{eq} \\ \mathbf{Y}_{eq} & \mathbf{U}_{eq} \\ \vdots & \vdots \\ \mathbf{Y}_{eq} & \mathbf{U}_{eq} \\ \vdots & \vdots \end{pmatrix} \quad \text{and} \quad \mathbf{t} = \begin{pmatrix} \mathbf{Y}_{oeq1} \\ \mathbf{Y}_{eq} \\ \left[\frac{\partial f}{\partial y(k)}\right] \\ \vdots \\ \left[\frac{\partial f}{\partial u(k)}\right] \\ \vdots \end{pmatrix} \tag{5.8}$$

Where:

$\mathbf{Y}_{oeq1}$ is a vector of target response points out of equilibria

$\mathbf{Y}_{oeq}$ is a vector of input response points out of equilibria

$\mathbf{U}_{oeq}$ is a matrix of input points out of equilibria

$\mathbf{U}_{eq}$ is a matrix of equilibria input points

$\mathbf{Y}_{eq}$ is a vector of equilibria response points

$\left[\frac{\partial f}{\partial y(k)}\right]$ is a vector of derivative observations of response component (vector of a linear model coefficient in different points).

$\left[\frac{\partial f}{\partial u(k)}\right]$ is a vector of derivative observations of input component (vector of a linear model coefficient in different points).

Therefore, the target vector **t** now contains derivative observations rather than just output measurements, and the input matrix **X** has also been extended to include the values of the regressors associated with each derivative observation. Furthermore, in this model structure, a derivative observation vector ($\left[\frac{\partial f}{\partial y(k)}\right]$ and $\left[\frac{\partial f}{\partial u(k)}\right]$), exists for each

component of the input matrix. Therefore, the dimension of the input space is $(n+D\cdot n_D)\times D$ and the dimension of the target vector is $(n+D\cdot n_D)\times 1$, where D is the number of derivative observation vectors, n is the number of function observations (input-output training data), and n_D is the number of derivative observations (input-output equilibrium training data).

From this organisation of the input matrix and target vector, a corresponding organisation of the covariance matrix is given by:

$$\mathbf{K}=\begin{pmatrix} \left[C(\mathbf{x}_i,\mathbf{x}_j)\right] & \left[C(\mathbf{x}_i,\frac{\partial \mathbf{x}_j}{\partial x_j})\right]_{d=1} & \cdots & \left[C(\mathbf{x}_i,\frac{\partial \mathbf{x}_j}{\partial x_j})\right]_{d=D} \\ \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\mathbf{x}_j)\right]_{d=1} & \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\frac{\partial \mathbf{x}_j}{\partial x_j})\right]_{e=1,d=1} & \cdots & \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\frac{\partial \mathbf{x}_j}{\partial x_j})\right]_{e=1,d=D} \\ \vdots & \vdots & \vdots & \vdots \\ \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\mathbf{x}_j)\right]_{d=D} & \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\frac{\partial \mathbf{x}_j}{\partial x_j})\right]_{e=D,d=1} & \cdots & \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\frac{\partial \mathbf{x}_j}{\partial x_j})\right]_{e=D,d=D} \end{pmatrix} \quad (5.9)$$

$$\mathbf{k}(\mathbf{x})=\begin{pmatrix} \left[C(\mathbf{x}_i,\mathbf{x})\right] \\ \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\mathbf{x})\right]_{d=1} \\ \vdots \\ \left[C(\frac{\partial \mathbf{x}_i}{\partial x_i},\mathbf{x})\right]_{d=D} \end{pmatrix} \quad (5.10)$$

$$k(\mathbf{x})=\left[C(\mathbf{x},\mathbf{x})\right]=v \quad (5.11)$$

Therefore, utilising these matrices, the standard GP predictive equations can be computed to give the mean and variance components of the output predictive distribution.

5.5) Experimental Methods and Objectives

In the remaining parts of this chapter, a number of simulated and experimental example systems are to be identified using the GP modelling approach. Therefore, before delving fully into the specific characteristics of the systems to be identified, a brief discussion of what precise aspects of the GP modelling approach are to be investigated is worthwhile. Furthermore, this section is to provide a means to reiterate some of the main findings of the previous sections, and therefore justify some of the implementation methods and model design choices that are to be made in the process of identifying the example systems investigated.

5.5.1) Implementation of GP Models

One of the more overlooked aspects within the literature of GP models is how best to implement the approach for a given application. This is partly due to the application specific nature of any system identification task, i.e. what will work on one problem, may not work on another. Nevertheless, a more practical guide to the implementation of GP models to for a variety of nonlinear systems would be a useful addition to the field. In Chapter 4, a detailed discussion of the implementation aspects of the GP modelling approach was presented. In this section we focus on some of the main points of this discussion that are to be investigated in the forthcoming examples.

In the development of a suitable mathematical model one of the key stages of the system identification process is to select an appropriate model order. However, as the GP model is a nonparametric method, this component of the identification process is not required. Instead the GP model is to be completely defined through the selection of an appropriate covariance function together with a suitable set of training data. The choice of covariance function will impact heavily on the types of nonlinearity that the resultant GP model will be able to represent effectively, and the design of the training dataset will have a profound bearing on the relative accuracy of any identified GP model.

With regard to the design of the training dataset, as this component will contain the information with which to identify the GP model, the selection of an appropriate set of inputs or regressors will play an important role in the development of an accurate model.

Therefore, for complex systems where a large number of regressors will be required to characterise the system, the identification of these regressors is an important part of the identification process. For the relatively simple example applications examined in this thesis the identification of suitable regressors was found to be quite straightforward as the number of regressors required was quite low (typically used only a single input variable together with a few delayed input and output variables). However, for more complex systems the task of identifying suitable regressors may become more challenging and the use of delay-embedding theory (see Takens (1981)) may be of particular relevance in order to identify the nonlinear mapping through dynamic reconstruction of the observed time-series data.

5.5.1.1) Choice of Covariance Function

For the choice of covariance function, any covariance function that results in a positive semi-definite covariance matrix may be employed. However, whilst a number of different covariance functions and even combinations of covariance functions have been proposed, see Section (4.3), a limited amount of practical research is available with which to select an appropriate covariance function. As a result, the most popular squared exponential or Gaussian covariance function has become almost uniformly adopted in the GP modelling approach. The use of this stationary function imposes the assumption that the input-output data to be approximated varies in a smooth and consistent manner. In the case of real engineering systems such qualities are common, as most real systems are designed to operate smoothly for ease of use.

However, it is also clear that many systems will exhibit responses that fail to meet this assumption of smooth and stationary behaviour. Therefore this reliance on the squared exponential function can be seen to be limiting the potential flexibility of the GP model. Nevertheless, the squared exponential covariance function has been used successfully in the identification of a variety of systems, so a further investigation into the flexibility and ultimate limitations of the Squared Exponential function is worthwhile. Therefore, in the experimental results presented in this thesis the choice of covariance function has been initially restricted to the popular Squared Exponential function. From this point we can then move onto identifying the potential limitations that this choice imposes, and then seek to offer alternatives.

A further reasoning behind the selection of the Squared Exponential covariance function is that it is well known from other approaches (e.g. RBF Networks), and the hyperparameters can be interpreted more easily than for other covariance functions. In addition, some of the extensions (uncertainty propagation and derivative observations) to the standard GP modelling approach have been proposed with this covariance function specifically in mind. Whilst other covariance functions can be used with these extensions (i.e. an approximation based upon a Taylor-series expansion for uncertainty propagation), as yet very few further developments or experimental investigations have been presented.

5.5.1.2) Design of Training Dataset

The design of a suitable training dataset is paramount in the successful identification of any model developed primarily from empirical data. In Section (4.5), the size and conditioning aspects of the covariance matrix and its implications for the design of the training dataset were discussed in detail. In the forthcoming examples, we are to investigate some of the issues raised in this discussion.

In particular, as the size of the training dataset dictates the size of the covariance matrix, in order to ensure that the identified GP model remains computationally viable, the number of datapoints included in the training set should not be too large. In Section (4.5.4) an upper limit of ~1000 datapoints was proposed as being suitable for the direct implementation of the GP model for system identification purposes on average desktop PC facilities and a number of approximate methods have also been proposed for larger datasets. In the examples presented here, we are to stick to this limit of ~1000 datapoints, and therefore investigate the identification of GP models using relatively small datasets. Therefore, important issues such as the choice of sampling rate and the design of the input signal are to be discussed. Furthermore, any aspects where prior knowledge of the system can be utilised are to be made clear.

Given this constraint over the size of training dataset, the task of creating a dataset that captures the essence of the underlying process becomes a significant challenge to ensure that we make the most of the space available. In addition, the choice of covariance function can be seen to extend an influence toward the suitability of any training dataset, i.e. the squared exponential function’s requirement for smoothly varying data. Furthermore, during

the optimisation of the hyperparameters and ultimately the computation of new predictions, the covariance matrix built from this combination of covariance function and training data must be inverted many times. Consequently, not only is the size of the training set important, but the conditioning of the resultant matrix must also be appropriate in order to allow efficient and accurate inversion. Furthermore, as discussed in Section (4.5.3), the requirement for a well conditioned covariance matrix has implications for the design of the data collection experiments.

In section (4.5.2) one of the most likely causes of covariance matrix ill-conditioning was identified as the presence of large amounts of steady-state data in the training dataset. Therefore, one of the main experimental design strategies that can be employed is to endeavour to keep the system excited (i.e. not operating under equilibrium conditions) for the duration of the experiment. Such an approach can be undertaken through the use of random excitation signals. However, as will be discussed in the forthcoming examples it is also necessary to allow the system to approach steady-state in order to include this information in the training dataset. As a result, the excitation signal used to collect the training data must be considered carefully. Furthermore, in some of the examples presented in this thesis it has also been necessary to manually remove prolonged periods of steady-state data from the training dataset. Through this removal of problematic steady state datapoints it could therefore be construed that an asynchronous approach to sampling is being taken. However, the approach taken was to remove certain sections of data and then reconstruct a complete training dataset from the portions of the overall dataset that are to be kept. As a result, a uniform sample interval (synchronous sampling) was maintained in the training dataset.

An alternative approach to tackling the problem of ill-conditioning caused by steady-state data is to employ some method of regularisation, as mentioned in Section (4.5.2.1.3). Such an approach would add a further level of noise or ‘jitter’ to the data so that prolonged periods of steady state data can be made more variable, see Tikhonov and Arsenin (1977). However, for the examples presented in this thesis we are to concentrate on the pre-processing of the training dataset without the use of such regularisation techniques. In this way, the problems encountered with the data can be stated more clearly and therefore tackled directly, rather than masked through the addition of an artificial noise component.

5.5.1.3) Further Developments

The initial experimental results presented are focused on presenting the capabilities of the most straightforward implementation of the GP model when applied to relatively simple nonlinear static and dynamic systems. However, through the selection of a particular covariance function (Squared Exponential) and the restriction of the number of training points included in the covariance matrix, the potential flexibility of the approach has been compromised somewhat. As a result, through the course of this chapter a number of potential strategies have been investigated with a view to providing a solution to some of the problems encountered.

In order to tackle the constraints over the size of the training dataset, in Section (4.5.5) a number of approximate methods were discussed. Furthermore, is Section (5.4) an alternative method based on the use of derivative observations was discussed. In the forthcoming results, some of these methods are to be investigated. In the final part of this chapter, the proposals for 'mixed model' implementations of the GP model are also to be demonstrated. Overall, these methods are aimed at overcoming some of the weaknesses in the GP modelling approach that have been encountered.

5.5.2) Examining Performance of the GP model

In order to support the proposal for considering GP models as a tool for nonlinear system identification, it is necessary to demonstrate that accurate models of system behaviour can be obtained. From a fundamental perspective, there would be not much point in persevering with the GP modelling approach, let alone recommending it, if the resultant predictive performance is poor. Consequently, careful validation procedures conducted on separate test datasets (i.e. cross-validation), must be adopted in order to provide evidence of the accuracy of the identified models. This is especially important to provide a feel as to the performance of the GP model, as no direct numerical comparison of the GP models with other different modelling procedures is to be presented in this thesis. A detailed numerical comparison between the GP modelling approach and other machine learning methods is provided in Rasmussen (1996), and a more practical system identification comparison can be found in Kocijan et al (2003a).

For the evaluation of the GP modelling approach a number of different measures of performance can be utilised. These include the standard quantitive measures of model accuracy that can be used to evaluate the mean prediction, such as Mean Square Error (MSE) and Mean Relative Square Error (MRSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \tag{5.12}$$

$$\text{MRSE} = \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{\sum_{i=1}^{N} y_i^2}} \tag{5.13}$$

Where y_i is the system output, $\hat{y}_i$ is the model output and $e = \hat{y}_i - y_i$ is the prediction or model error at the *i*th case of the test dataset (of size *N*).

Furthermore, as the GP model is a probabilistic approach, where an output predictive distribution is provided, it is also possible to evaluate the performance of the model using more probabilistic measures. For the GP model, the negative Log Predictive Density (LPD) and negative Log-Likelihood (LL) have been used as valuable indicators of model performance:

$$\text{LPD} = \frac{1}{2}\log(2\pi) + \frac{1}{2N}\sum_{i=1}^{N}\left(\log(\sigma_i^2) + \frac{e_i^2}{\sigma_i^2}\right) \tag{5.14}$$

$$\text{LL} = \frac{1}{2}\log|\mathbf{K}| + \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} + \frac{N}{2}\log(2\pi) \tag{5.15}$$

The LPD estimate (smaller or ‘more negative’ the better) accounts for the model uncertainty (variance output), and trades it off against the accuracy of the model. As a result, this measure of performance is to especially indicate when the predictions are ‘overconfident’ (high model error & low variance), rather than predictions that are ‘good’ (low model error & low variance) or ‘bad’ (high model error & high variance). The negative LL estimate (smaller or ‘more negative’ the better) is the same loss function used to train the hyperparameters of the covariance function, and provides an overall indication as to the probability of the model. As a result, the LL measure is less useful for assessing

model performance as it will be modified every time the training dataset and model setup are changed.

Along with these quantitative measures of model performance, of further importance is to assess the model in more subjective or qualitative terms. In this way, the model validation stage can ensure that the identified model is fit for its intended purpose. Therefore, important aspects such as the overall plausibility and interpretability of the model can be assessed. Furthermore, together with close scrutiny of the prediction error, the variance output of the GP model can provide some valuable insight as to the model performance in local operating regions (e.g. a high variance is likely to be due to a lack of training data in a particular region).

As the variance output of the GP model is one of the most potentially attractive features of the approach, the characteristics of this extra output information must also be investigated. In an ideal situation, the location and magnitude of the variance output would exactly mirror that of the prediction error between the model and the process. In this way, we would then have a measure of the model error that is available at all times or 'online', rather than only when validation tests are being completed. As discussed earlier, this extra information could then be integrated into some form of model-based predictive control or fault detection implementation. However, fundamentally model error is not what the variance output signifies; instead it is a measurement of uncertainty over each prediction, not a measure of the error itself. In other words, the level of agreement between the test data and the information found within the training dataset is what governs the resultant variance output. Therefore, test data presented to the GP model that does not fall within the boundaries of the training set, or shows significantly different characteristics, should result in a potentially inaccurate prediction estimate together with a marked increase in the magnitude of the variance output. Therefore, something that should be closely examined for each example application is the relationship between the level of the variance output and the degree of model error.

Of further interest in the examination of the variance output, is the proposal for uncertainty propagation from one prediction to the next, see Section (5.3). Overall, the idea that the uncertainty surrounding one prediction should then be reflected in subsequent predictions appears to be sensible. However, as discussed in Section (5.3.3), the uncertainty

propagation algorithm is primarily of use in applications where the variance output is to be actively employed in some manner. As this thesis is primarily concerned with identification and not control, the uncertainty propagation extension has not been implemented. Furthermore, as the outcome of adopting this extension is well understood to be an overall 'flattening effect' on the output predictive distribution, rather than an increase in the accuracy of the GP mean prediction, there is no great reason to repeat this demonstration.

5.6) Simulated Examples

Before tackling the identification of the real laboratory based nonlinear systems from experimental data, the GP modelling approach is first to be applied to some initial simulated examples. These examples are aimed at demonstrating the power of the GP modelling approach when confronted with the task of identifying strongly nonlinear mathematical functions from a small number of training observations. Furthermore, these examples are intended to demonstrate the full process of applying the method. The previous demonstrative example, see Section (3.8), was aimed at conveying the theoretical procedure, where the hyperparameters were not identified from the training data. Instead, the posterior was generated using the same random process as the datapoints. In these simulated examples and for the experimental results to come, the hyperparameters of the covariance function are to be identified from the training data. A further intention of these simulated examples is to highlight the differences between implementations of the GP modelling approach where static nonlinearities are to be identified, such as those found in the regression or interpolation problems found in machine learning and statistics, and the more typical dynamic time-series problems found in engineering which are the focus of this research.

5.6.1) 'Smooth' Data - Static Nonlinear Example

In this opening example we are to consider one-dimensional static nonlinearities. Consider the following smoothly varying nonlinear mathematical function:

$$y = \frac{\sin(x)}{x} \tag{5.16}$$

In this example we are to attempt to identify the unknown y from a number of observed inputs and outputs {x, y} and therefore build a 1 dimensional mapping from x to y so that when presented with a different set of inputs x*, we may be able to predict the resultant outputs. For this static nonlinear problem, we can simply define an input range, $\mathbf{x} \in [0 \quad 20]$ and calculate the resultant outputs. To make things slightly more realistic, some random noise can be added to the function. For the initial part of this opening example the noise is to be kept very low, however it will be increased in a later example. A number of training observations can then be selected through sampling of the original function calculations (every 2 seconds, resulting in 10 datapoints), as in Figure (5.1).

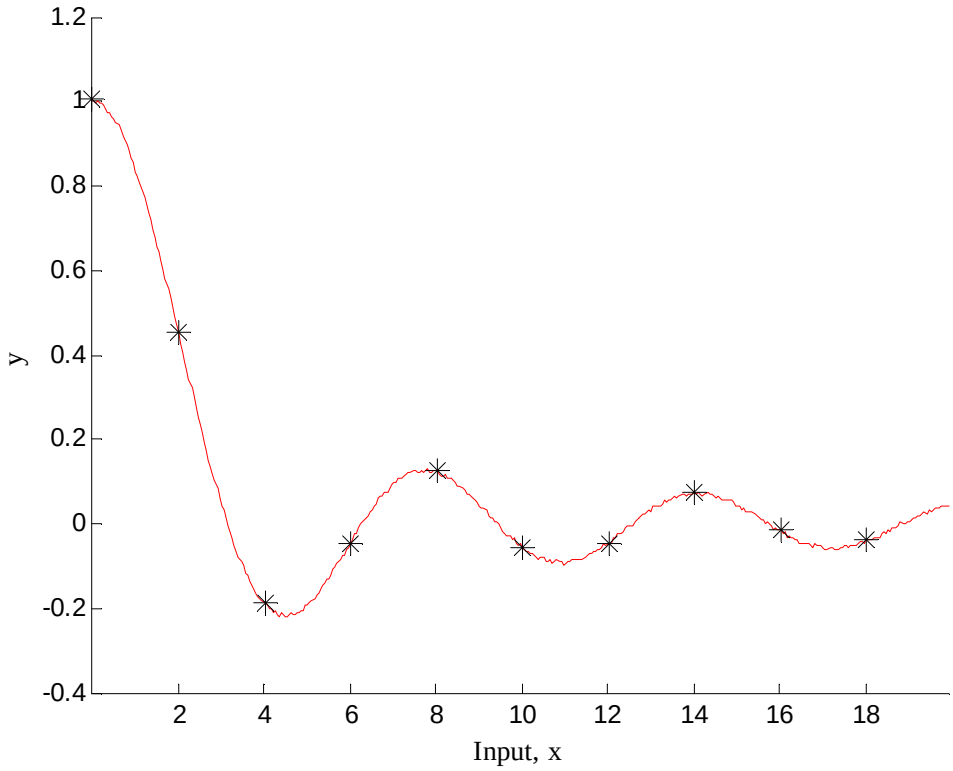


Figure (5.1): Shows one-dimensional simulated example function, with 10 evenly spaced training observations marked.

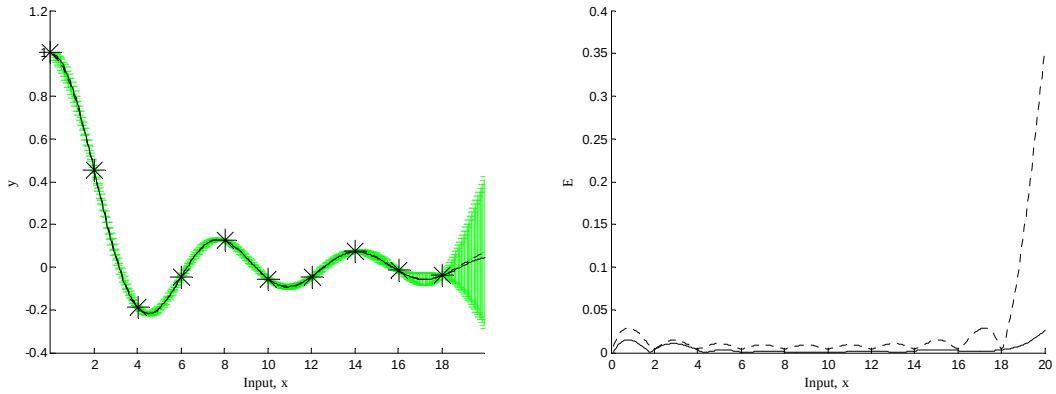
The next stage of the GP modelling process is to select an appropriate covariance function with which to generate the covariance matrix that specifies the Gaussian process prior. As indicated earlier, the most popular choice of covariance function is the Squared Exponential function and it is this one that we are to employ for this problem. Furthermore, as the choice of this function implies a smoothly varying function, it would seem appropriate for this simulated example. Lastly, this covariance function is to be combined with the simple noise model as described in section (4.3).

The task is now to perform maximum likelihood optimisation to identify a set of optimal hyperparameters for the covariance function. In this simple one-dimensional example we are to find three hyperparameters of the Squared Exponential covariance function: a vertical variance or amplitude hyperparameter θ_1, a length-scale parameter θ_2, and a noise parameter θ_3. For this example, a set of default initial values of these hyperparameters (log(-1) for all 3) were selected, equivalent to uniform priors over each hyperparameter. By applying the methods detailed in Section (4.4.2), the hyperparameters were calculated as $\boldsymbol{\theta}_{MP}$.= (θ_1 = 2.7677 θ_2 = 0.4566, θ_3 = 0.0023) Following the identification of a set of optimal hyperparameters, the covariance matrix K may then be fully specified. The next stage is then to employ the predictive equations (3.40) and (3.41) toward the goal of predicting new output target values given a series of test inputs x*.

Regarding the choice of data to be employed to test the GP model, it is at this point that we can define the major difference between the static nonlinear mapping problems considered here, and the more conventional dynamic time-series data examples found in engineering applications. As we have collected a set of training data that evenly covers the whole of our defined input range ($0 < x < 20$), any test data subsequently collected will also lie in close proximity to these training points. In effect, the test data will closely match the training data and therefore lead us to the expectation that the resultant GP model predictions should closely match the underlying function (assuming that we have included sufficient training points). This point may seem an obvious one, but it is worth stating as a lot of the GP modelling literature demonstrates the approach with such static one-dimensional examples of this kind. Therefore, it is important to point out that in such cases, the training and test data are often quite similar and a good model should not be an unexpected outcome. Furthermore, it is also important to make clear that as no previous output information is to be used as additional inputs, this example should be termed as a simulation rather than a prediction.

In Figure (5.2a) we can see that the GP model predictions do achieve a good fit to the underlying function, with a Mean-Square Error (MSE) of 3.67e-005, Log Predictive Density (LPD) of 3.9632, and log likelihood (LL) of 0.9607. Furthermore, due to the even spread of the training points, even though there are relatively few training data points (10 in this case), the variance output of the GP model is relatively low and consistent across most of the defined operating range of the input. However, we can see a marked increase in the

variance at $x > 18$ and also the beginnings of model error in the mean prediction. This growth in the model error and variance is further indicated in Figure (5.2b), and is due to the lack of training points in this region of input space. Notice also the slight growth of the variance for test points occurring between the evenly spaced training points.



(a) **(b)**

Figure (5.2): One-dimensional simulated example function. Chart (a) shows GP mean predictions (dotted line) vs Underlying function (solid line) together with GP model variance shown as 2σ errorbars (95% confidence interval). Chart (b) shows GP Model Error (solid line) and Variance (2σ) output (dashed line).

The decay in the prediction accuracy in regions where training data is lacking is something that all empirically based modelling approaches will be subject to. Therefore, the process of collecting a suitable empirical dataset that covers the entire operating range of interest is of fundamental importance. Furthermore, it is worth reiterating that, fundamentally, the GP modelling approach is a method of interpolation, where a curve is to be fitted between observed values. This means that outside of the input range that is populated with training observations, the GP model will not provide any reliable estimates of the underlying function (i.e. the extrapolation ability of the GP model is poor). However, due to the existence of the variance output (that should increase substantially in regions where training data is limited), we can at least display a lack of confidence in any predictions made in these sparse regions.

As would be expected, through further reduction of the number of included training points the accuracy of the model diminishes and the variance output increases in regions of input space where data are sparse. Furthermore, a reduction in the number of training points can lead to optimisation problems (in this example, $N<8$ results in a failure of the optimisation algorithm), as there is simply insufficient information available with which to identify the

hyperparameters. By contrast, an increase in the number of training points included will result in greater prediction accuracy at the expense of an increased computational burden.

Therefore, for online applications a temptation may exist to train the model offline using a large dataset to obtain suitable hyperparameters, then for reasons of computational efficiency use a smaller dataset with which to make predictions. However, such an approach is untenable as the GP model is defined by the information present in its covariance matrix (i.e. hyperparameters such as the length-scale are defined by the precise spatial relationship between training points), rather than as a parametric form where parameters may be interchanged or manipulated. This is perhaps an obvious but important point, the hyperparameters are not transferable between different models, and although their individual roles may be interpretable, the interdependency of the hyperparameters leads to an overall lack of interpretability that prevents any meaningful manual adjustment.

5.6.2) ‘Sparse’ Data Region - Static Nonlinear Example

As discussed previously, one of the primary drivers behind the research into the GP modelling approach was the method’s potential use in the identification of models in off-equilibrium operating regimes. The main difficulty associated with identifying models of such operating regimes using empirical methods is in obtaining enough empirical data. In the previous example, the general impact of the number of included training points on the GP model’s predictive accuracy was discussed. Furthermore, the growth of the variance output in operating regions where training data is sparse was also made clear.

In this example, we are to build on this second point and demonstrate why the GP model is a good choice for tackling identification in sparse regions of operating space. Furthermore, rather than employ an arbitrarily chosen nonlinear function to generate empirical data, a simple Simulink model, see Figure (5.3), is instead utilised to generate nonlinear data.

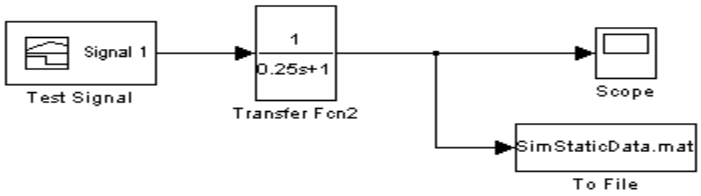


Figure (5.3): Simple 1-dimensional Static Nonlinear System

Through the course of this chapter, a number of different Simulink models are to be utilised for the generation of example training and test datasets, as a greater control over the type of nonlinearity and scale of the input range is possible. Therefore, whilst such models are no substitute for the real system applications, they do offer the possibility to easily try out different strategies and demonstrate different aspects of the GP modelling approach. In this example, the 'Signal Builder' block is used to generate an input signal composed of a number of positive and negative step inputs. This data is then fed through a simple 1st order transfer function block that has the effect of slowing down this transient behaviour in order to allow a set of smoothly varying data to be collected. As this smoothly varying data is one-dimensional (i.e. not input and output data), it is to be interpreted in a similar manner to that of the previous static nonlinear example. In this example, the Squared exponential covariance function is again used to define a Gaussian Process prior.

As this example is to demonstrate how the GP model is to tackle the identification task in regions of sparse data, the data signal was designed in a manner to reflect 3 different local operating regimes. At small and large values of x the output y is to vary smoothly and have relatively small amplitude variation (i.e. regions $x < 20$ and $x > 40$). In the middle of these regions of input space (i.e. $20 > x < 40$), the scale of the amplitude variation of the data signal is to be considerably larger. The data collected from this system was initially sampled every 0.2 seconds to provide 300 training points (note that in terms of sampling the input x is interpreted as a timescale). Then the middle section of the data was further sampled by a factor of 5 so that the training data in the region $20 > x < 40$ is observed every 1 second, resulting in an overall training dataset of 220 points. This training dataset is shown in Figure (5.4).

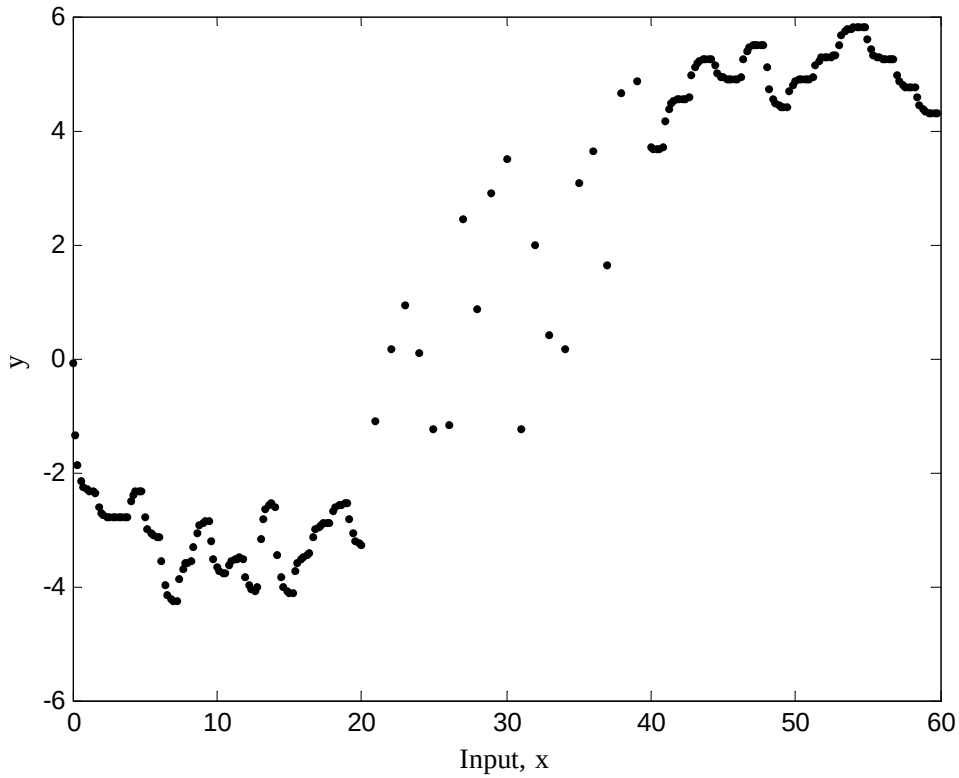


Figure (5.4): Training Data of Sparse Data Example

The next stage is to identify the hyperparameters of the covariance function through the optimisation of the marginal likelihood. As the covariance function is the same as that used in the previous example, the same 3 hyperparameters need to be identified. Using the same initial values (log(-1) for all 3), the hyperparameters were calculated as $\boldsymbol{\theta}_{MP}$.= (θ_1 = 0.9851 θ_2 = 2.9812, θ_3 = 0.0757). As before, due to the static nature of the problem, the test data will invariably be of a similar constitution to that of the training data. However, in order to investigate the quality of the model identified in the sparse middle region of the data, the full complement (sampled every 0.01s) of the generated x data is to be used as the test input. Once again, this example can be understood as a simulation rather than a prediction. The GP model's mean predictions are shown in Figure (5.5) and compared with the underlying test data.

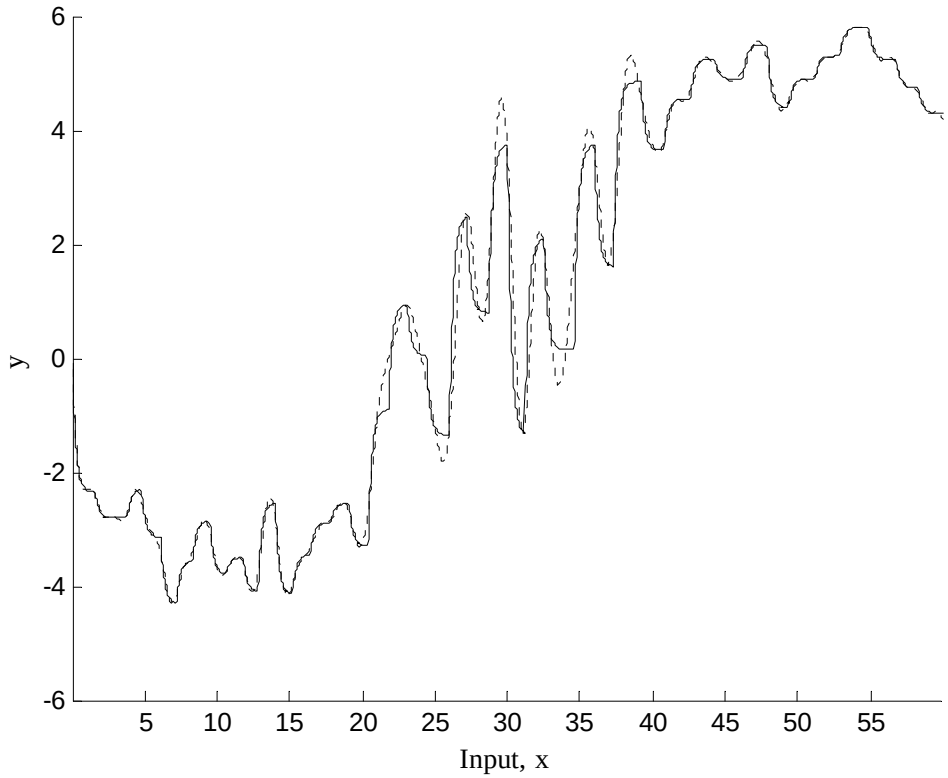


Figure (5.5): GP Mean predictions (dotted line) vs. Underlying function (solid line)

Overall, we can see that the GP model achieves a good level of approximation in the upper and lower regions of the input space where training data is plentiful. As would be expected, the quality of the model predictions is significantly reduced in the middle region where training data is more sparse. Using the same measures of model performance as before gives a Mean-Square Error (MSE) of 0.0918, Log Predictive Density (LPD) of -0.2169, and log likelihood (LL) of -7.7237. The variance output of the GP model can again be plotted on the same axis and compared with the model error as shown in Figure (5.6).

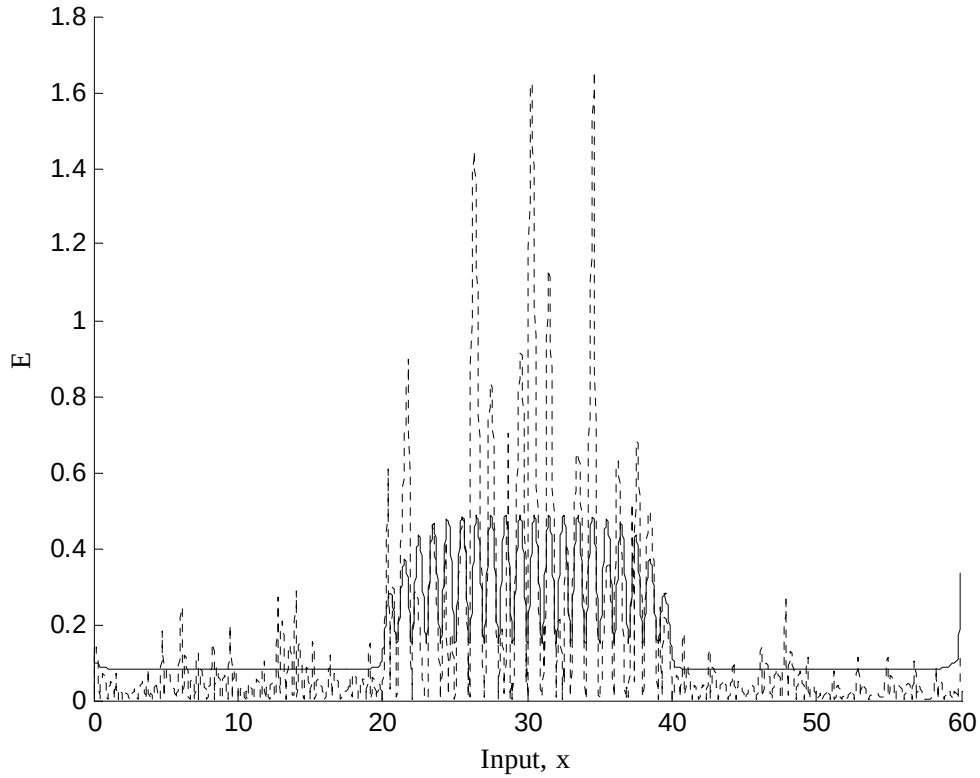


Figure (5.6): GP Model Error (dashed line) and Variance (2σ) output (solid line)

In Figure (5.6) we can see that, as well as a substantial increase in the model error present in the middle region of the input space, the variance output of the GP model is also significantly larger. It is this facility to indicate the confidence over each prediction that makes a probabilistic approach such as the GP model an attractive alternative to other modelling approaches.

Fundamentally, any empirical modelling approach will struggle to identify operating regions where data is limited; therefore the reduction of the model accuracy in this example is not something that should be unexpected. Nevertheless, the GP model does provide a reasonable attempt at identifying this sparsely populated region of operating space with the identified function at least bisecting the included training points. This is the other main advantage of utilising the GP method to tackle this kind of problem, as whatever data that is available will be used directly in making predictions in this region, rather than just employing a function whose characteristics are primarily identified from data in other operating regions (i.e. the GP model 'constrains' the identified function to at least touch the few included training points).

5.6.3) 'Noisy' Data - Static Nonlinear Example

In the previous two examples, the level of noise present on the data was minimal. In this example, the effect of a larger level of noise on the identification process of the GP model is to be discussed. Returning to the simple nonlinear example used in the first example (equation (5.16)), a significant level of random noise is to be added to the data, and the computed values are to be sampled in the same manner as before to obtain 10 training points, as depicted in Figure (5.7).

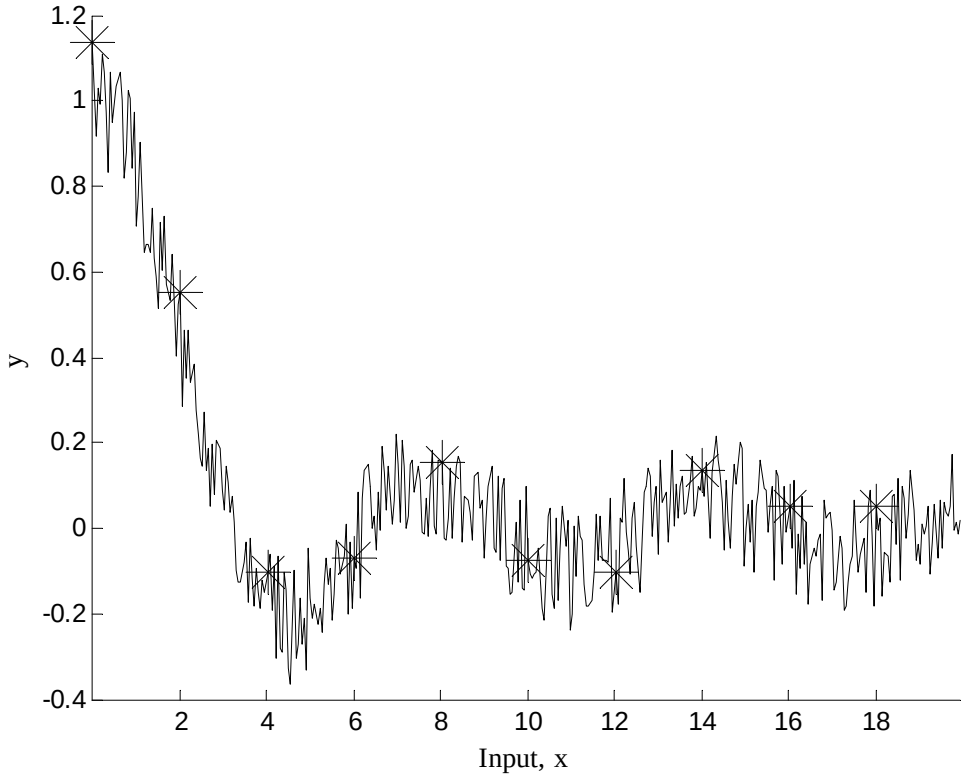


Figure (5.7): Shows Noisy simulated example function, with 10 evenly spaced training observations marked.

The effect that this additional noise has had on the training observations can be readily understood by plotting the previous noise-free training points and underlying function on the same chart as the noisy training observations, as in Figure (5.8).

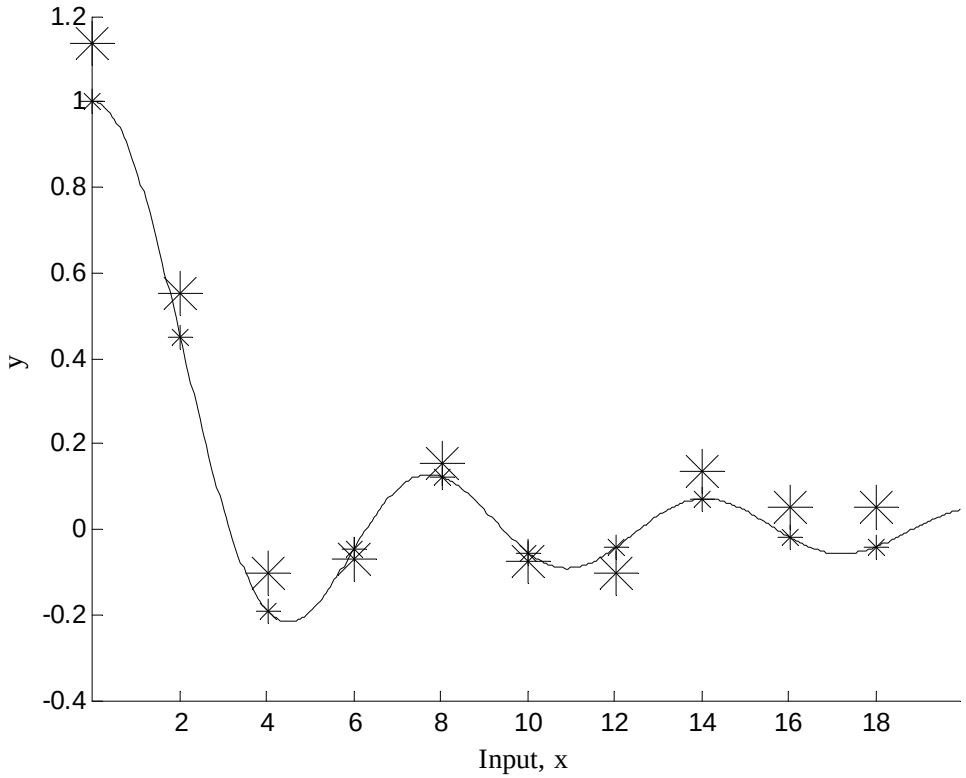


Figure (5.8): Shows Noisy simulated example function, with noisy (large markers) and noise-free (small markers) training observations marked.

As would be expected, the added noise has significantly modified the position of the training observations. Therefore, it is likely that any identified GP model will also be less accurate in approximating the underlying function. This is indeed the case as can be seen in Figure (5.9), where a significant error between the identified GP model (dotted line) and the underlying function (dashed line). Furthermore, as the GP model has provided a smooth estimate of behaviour, the model prediction has completely failed to capture the noise present in the data. This is again to be expected, as by including only 10 training points, it is impossible to capture the higher frequency oscillatory behaviour present in the noise. In order to approximate such behaviour, it would appear that the model would have to include a far larger number of training observations.

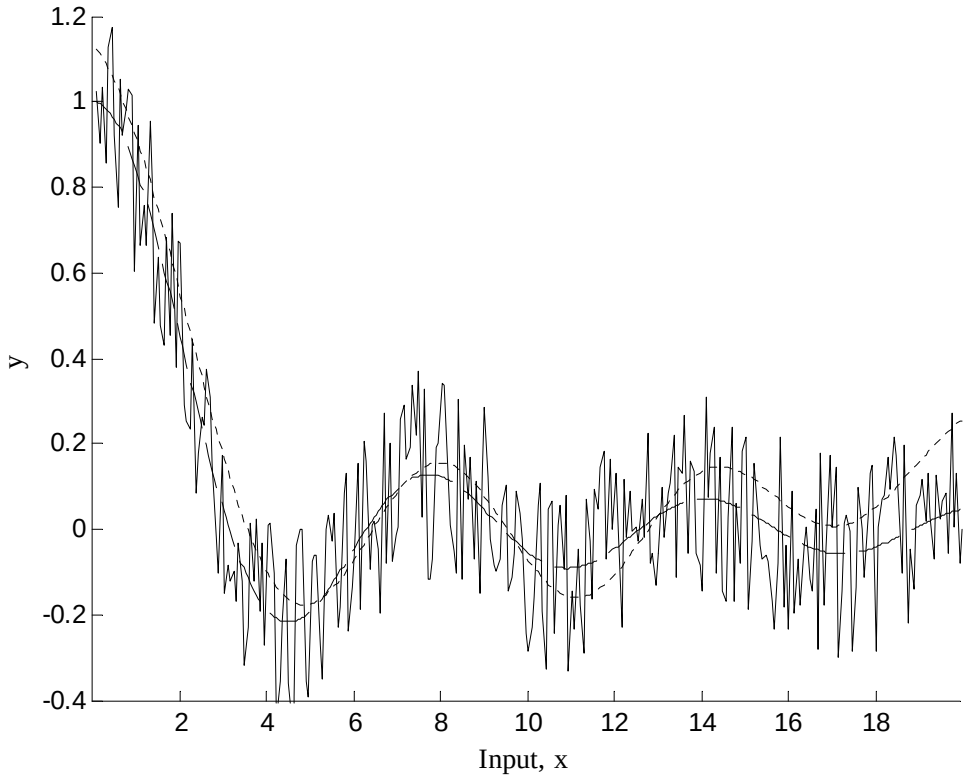


Figure (5.9): GP Mean predictions (dotted line) vs. Noisy Underlying Function (solid line) vs. Underlying function (dashed line) for 10 training observations.

In many modelling approaches, in order to successfully model systems where a large amount of noise is present, it is common practice to employ larger amounts of empirical data and perhaps even more complex model structures. Of course by doing so, the risk of 'overfitting' the data becomes more pronounced, where the model has begun to identify the noise rather than just the underlying function. In Chapter 3, the potential benefits of using the Bayesian approach of the GP model to tackle the problem of model complexity where made clear. Therefore, in this example it is worth retraining the GP model to include a far larger number of training points in order to demonstrate whether or not overfitting is to become a significant problem.

Using the same noisy nonlinear data, instead of sampling to provide 10 training points, this new implementation is to employ 400 training points. The effect of including more training points on the quality of the GP mean predictions is depicted in Figure (5.10). Overall, we can see that the new predictions of the GP model are very close to that of the underlying function, whilst the model has still not approximated the noise present in the data.

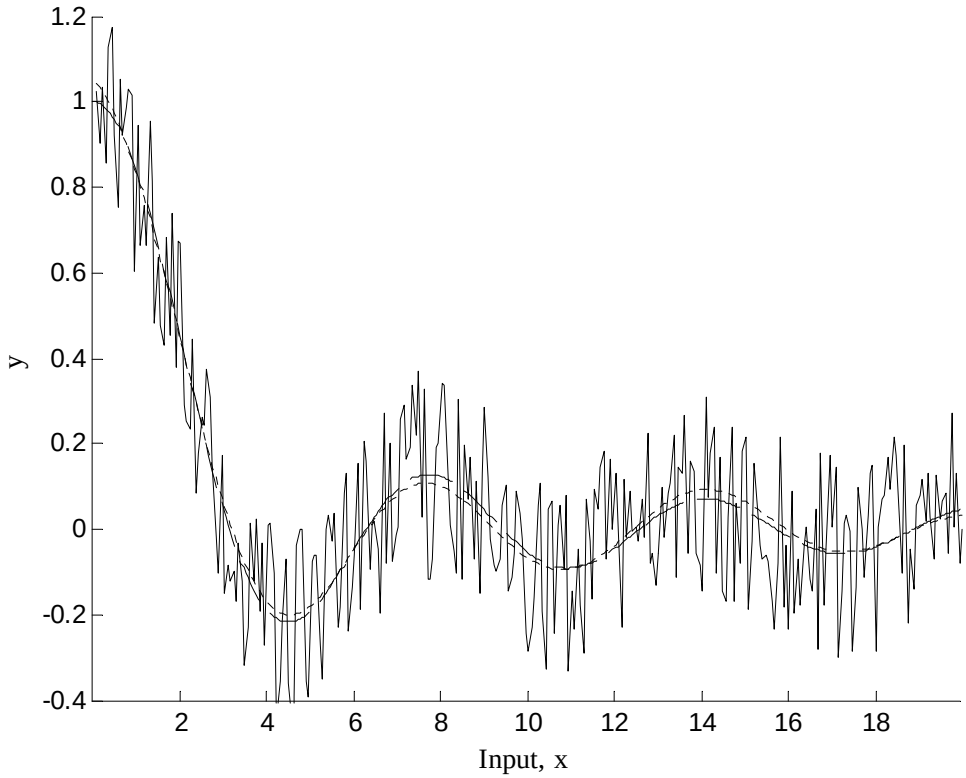


Figure (5.10): GP Mean predictions (dotted line) vs. Noisy Underlying Function (solid line) vs. Underlying function (dashed line) for 400 training observations.

Therefore, it is clear that as with other modelling approaches, the identification of noisy systems will require a significantly larger training dataset in order to provide accurate predictions of the underlying function. Furthermore, in contrast with other complex model architectures used for empirical modelling, the GP model does not tend to overfit the data. In some respects this reluctance of the GP model to begin modelling the noise present in the data is due to the automatic implementation of the Occam's Razor principle which implements a preference for the simplest solution (see Section (3.5.4.1)). However, a further aspect in play is that the specified Squared Exponential covariance function is only capable of providing smooth (infinitely differentiable) posterior functions. Therefore, any GP model specified with such a covariance function will fail to model such a sharply varying almost discontinuous (or non-differentiable) function.

5.6.4) 'Spiky' Data - Static Nonlinear Example

In the previous example, the GP model specified with a Squared Exponential was demonstrated as being unable to identify a sharply varying or 'spiky' example function. Whilst in the previous example this was beneficial in that the smoothly varying underlying function could be successfully identified given enough training points, despite a large noise component present in the training data. On the other hand, this inability to approximate more sharply varying data can be seen to be a fundamental limitation of the GP modelling approach when this most popular Squared Exponential covariance function is employed. For many real applications (including the two examples considered later in this chapter), this limitation to smoothly varying systems is not something that is unduly troublesome. However, for systems that exhibit a less smooth response, alternative methods must be pursued.

A possible strategy is to employ a different covariance function that has less strong assumptions over the smoothness properties of the underlying function. In Section (4.3.1.2), the Matérn class of covariance functions was identified as being suitable for such a task. The Matérn class of covariance functions allows the relative smoothness or differentiability of the GP prior to be controlled through the parameter ν. In this example, a Matérn covariance function is to be compared with the more popular Squared Exponential covariance function for the approximation of a 'spiky' static dataset.

The spiky dataset to be approximated was generated using a simple Simulink model as in Figure (5.3), however the denominator of the transfer function block was changed from (0.25s+1) to (3s+1), and the 'signal builder' block was used to define a number of positive and negative steps that vary in and around zero (rather than gradually increasing as in Figure (5.3). The 'spiky' dataset generated is depicted in Figure (5.11) and can be seen to vary considerably less smoothly than in the opening two examples of this section, but is also not extremely discontinuous as in the noisy previous example. As in the previous examples this dataset is to be used for both training and testing the GP model, and this example can be understood as a simulation rather than prediction. For the purposes of comparison, two training different sized datasets were sampled from this data, one containing 100 points and the other containing 34 points. The location of the smaller set of training observations is marked on Figure (5.11).

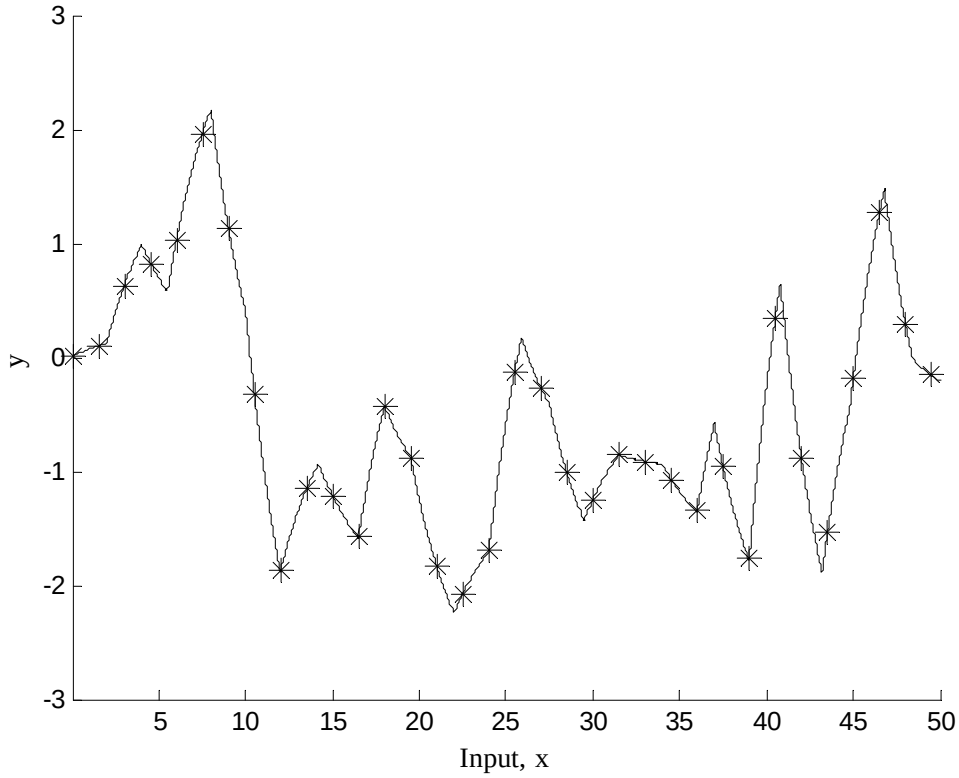


Figure (5.11): Shows 'Spiky' example function, with 34 training observations marked.

The Squared Exponential covariance function is first used to approximate the data using the 100-point training dataset (see Test 1 below) and then the 34-point training dataset (see Test 2). This process is then repeated for the Matérn covariance function (see Test 3 and 4). For the Matérn case, as discussed previously at higher values of the differentiability parameter, $\nu \geq 7/2$, sample functions taken from prior defined by the Matérn covariance function become almost indistinguishable from those defined by the Squared Exponential. Therefore, for this example the differentiability parameter is chosen to be $\nu = 3/2$, and the covariance function given by equation (4.4).

Test 1 – Squared Exponential Covariance Function(100 training points)

In Figure (5.12) the GP predictions are compared with the underlying data for the model trained on 100 training points. The hyperparameters were calculated as $\boldsymbol{\theta}_{\text{MP}}$.= ($\theta_1$ = 1.3877 θ_2 = 0.9999, θ_3 = 0.0783) and the validation measures calculated as Mean-Square Error (MSE) of 0.0028, Log Predictive Density (LPD) of 1.5058, and log likelihood (LL) of

7.2224. The variance output of this GP model can again be plotted on the same axis and compared with the model error as shown in Figure (5.13).

Overall, we can see that the GP model identified using the Squared Exponential covariance function provides a good approximation to the 'spiky' dataset. However, looking more closely at the sharp peaks of this dataset, there is a noticeable error between the model and the underlying data. This deficiency is due to the smoothness assumptions inherent in the choice of this particular covariance function.

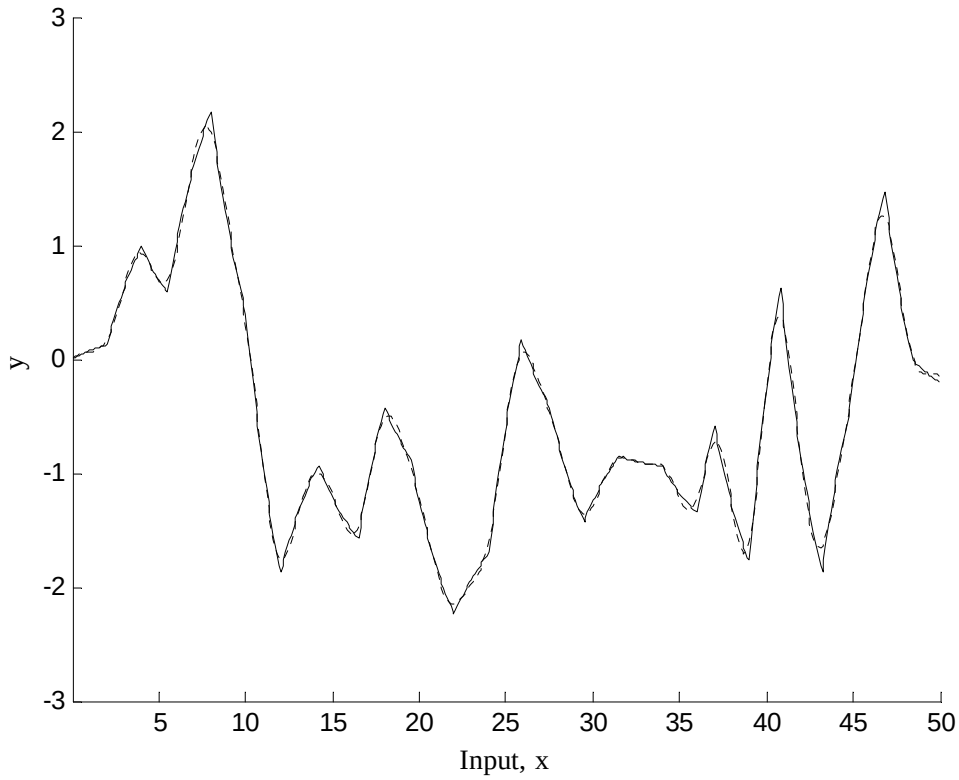


Figure (5.12): 'Spiky' Data Example (100 training points, Sq. Exp Cov. Function) - GP Mean predictions (dotted line) vs. Underlying function (solid line)

This mistaken assumption over the smoothness properties is further demonstrated in the complete lack of correlation between the model error and predictive variance as shown in Figure (5.13). In this example, the optimisation procedure has failed to find an optimal set of hyperparameters that would allow a smoothly varying function (as defined by the chosen prior) to accommodate the included training points. As a result, the variance associated with each prediction remains consistently large across the entire defined input range (i.e. the variance does not even tend to zero when test and training points are the same).

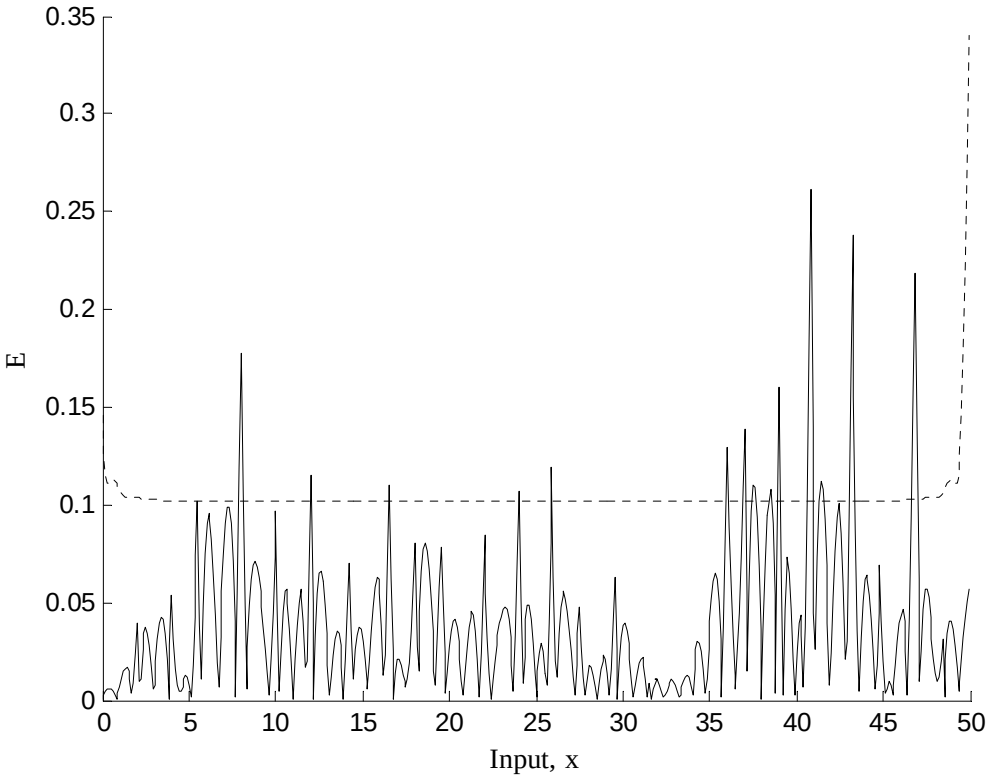


Figure (5.13): 'Spiky' Data Example (100 training points, Sq. Exp Cov. Function) - GP Model Error (solid line) and Variance (2σ) output (dashed line)

The next question that arises is that if the optimisation procedure has not exactly been a success, why does the GP model's mean predictions still provide a good approximation? The primary reason behind this outcome is the large quantity of training data, which significantly reduces the difficulty of this or indeed any interpolation task. Therefore, by including large quantities of training data, a poor choice of covariance function or failure of the optimisation process may be possibly overcome. Of course, by reverting to this kind of 'brute-force' strategy of including more and more training data, the computational expense and efficiency of the model may become unrealistic, and the risk of compromising the conditioning of the covariance matrix also becomes more pronounced (i.e. a low sampling interval may result in repetition in the data).

Test 2 – Squared Exponential Covariance Function (34 training points)

In this next test the size of the training set is reduced to 34 evenly space training points and the Squared Exponential covariance function retrained to obtain the following hyperparameters $\boldsymbol{\theta}_{MP}$.= (θ_1 = 1.8898, θ_2 = 1.0182, θ_3 = 0.3845). As before, the GP mean

predictions are compared with the underlying data in Figure (5.14), and the variance output of the GP model is compared with model error in Figure (5.15). Using the same model validation measures as before, we calculated a Mean-Square Error (MSE) of 0.0444, Log Predictive Density (LPD) of 0.0389, and log likelihood (LL) of -41.2297. Overall, the performance of the GP model trained on a smaller number of training datapoints is considerably poorer. Furthermore, the scale of the variance output is considerably larger than that of the previous test case. Neither of these outcomes can be seen to be surprising, as if the optimisation procedure is not capable of identifying suitable hyperparameters for the covariance function when a larger quantity of training data is available, it will naturally struggle even more when much of this information is taken away.

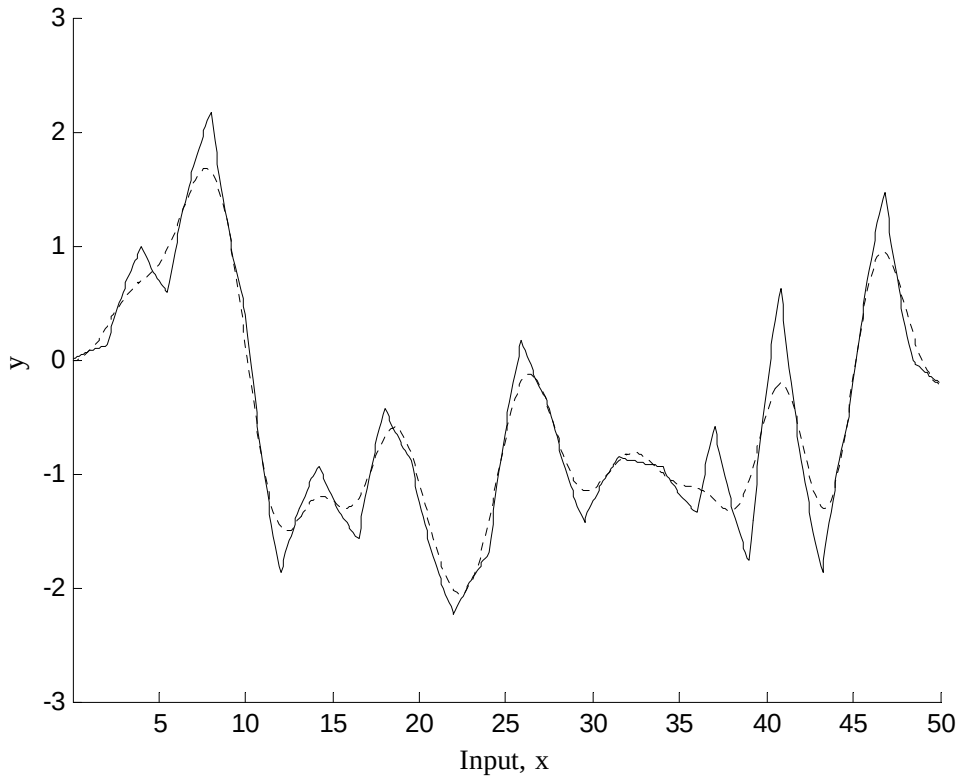


Figure (5.14): 'Spiky' Data Example (34 training points, Sq. Exp Cov. Function) - GP Mean predictions (dotted line) vs. Underlying function (solid line)

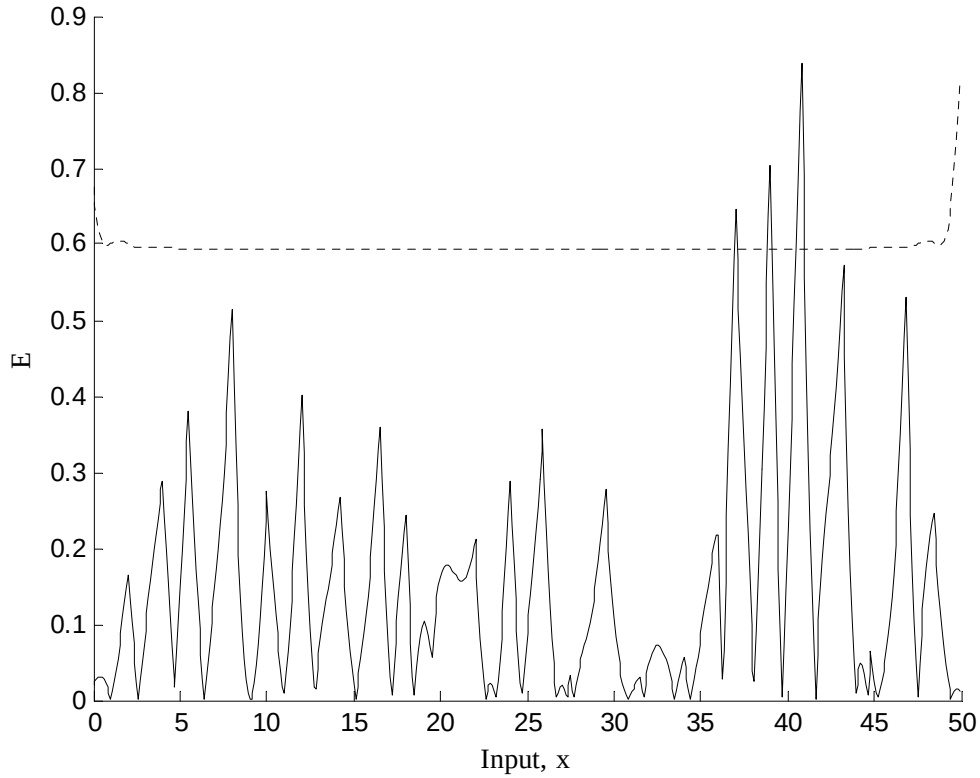


Figure (5.15): 'Spiky' Data Example (34 training points, Sq. Exp Cov. Function) - GP Model Error (solid line) and Variance (2σ) output (dashed line)

Furthermore, through the reduction of the size of the training dataset the previous 'safety-net' of a large dataset and therefore an easier interpolation task is taken away, and we can see that the GP model fails to even reach some of the observed training values. This is better illustrated in Figure (5.16) where the training observations are plotted on to the same chart as in Figure (5.15). Therefore, in this example where the training dataset has been reduced in size we can better demonstrate that the Squared Exponential covariance function is an unsuitable choice for problems involving sharply varying or 'spiky' data.

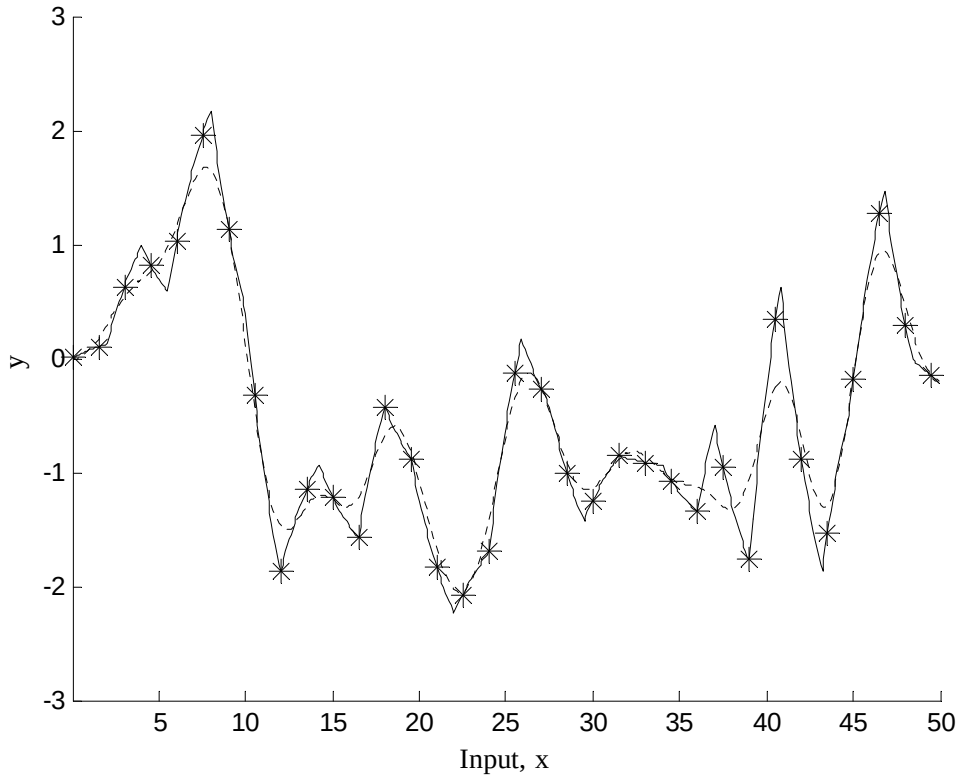


Figure (5.16): 'Spiky' Data Example (34 training points (Marked), Sq. Exp Cov. Function) - GP Mean predictions (dotted line) vs. Underlying function (solid line)

Test 3 – Matérn Covariance Function (100 training points)

Using the same training datasets, the Matérn covariance function (with $\nu = 3/2$) combined with a simple noise model is now to be employed to approximate the data. In Figure (5.17) the GP predictions are compared with the underlying data for the model trained on 100 training points. The hyperparameters were calculated as $\boldsymbol{\theta}_{MP}$.= (θ_1 = 2.8529, θ_2 = 1.1249, θ_3 = 0.0001), where these 3 hyperparameters play a similar role to that of the previous Squared Exponential covariance function. Using the same performance measures, the following were calculated, Mean-Square Error (MSE) of 0.00064, Log Predictive Density (LPD) of 2.1252, and log likelihood (LL) of 6.9822. The variance output of this GP model can again be plotted on the same axis and compared with the model error as shown in Figure (5.18).

Overall, the predictive accuracy of the identified GP model can be seen to be superior to the model identified with the Squared Exponential covariance function in Test 1, using the same data. In fact, the mean predictions of the GP model can be seen to be almost

indistinguishable from the underlying function, with the sharper peak regions of the data being better approximated by this Matérn GP model than the Squared Exponential GP model. This improvement in the predictive performance can be put down to the less constraining smoothness assumptions that are implied by using the Matérn covariance function. As the choice of this covariance function results in a GP prior that is capable of generating less differentiable sample functions, the sharper regions of data can be better approximated.

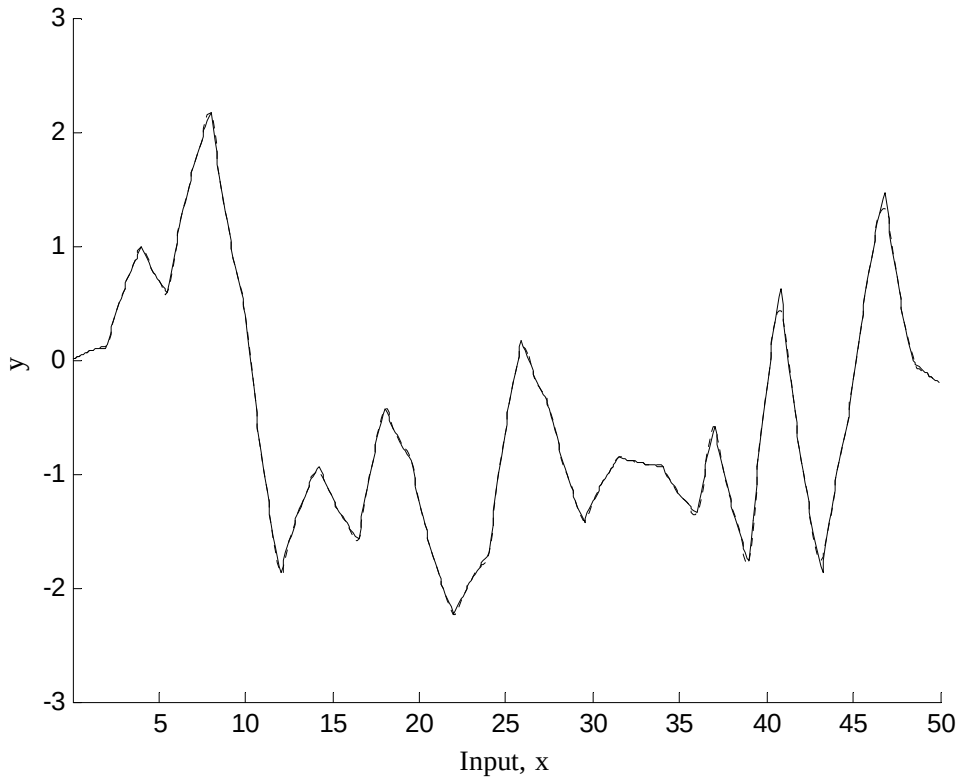


Figure (5.17): 'Spiky' Data Example (100 training points, Matérn Cov. Function) - GP Mean predictions (dotted line) vs. Underlying function (solid line).

Regarding the variance output of the GP model, in Figure (5.18) the variance can be seen to be near zero at test points that are co-incidental with training points (as would be desirable), however the growth and decay of the variance output can be seen to be very rapid in the small intervals between the test points.

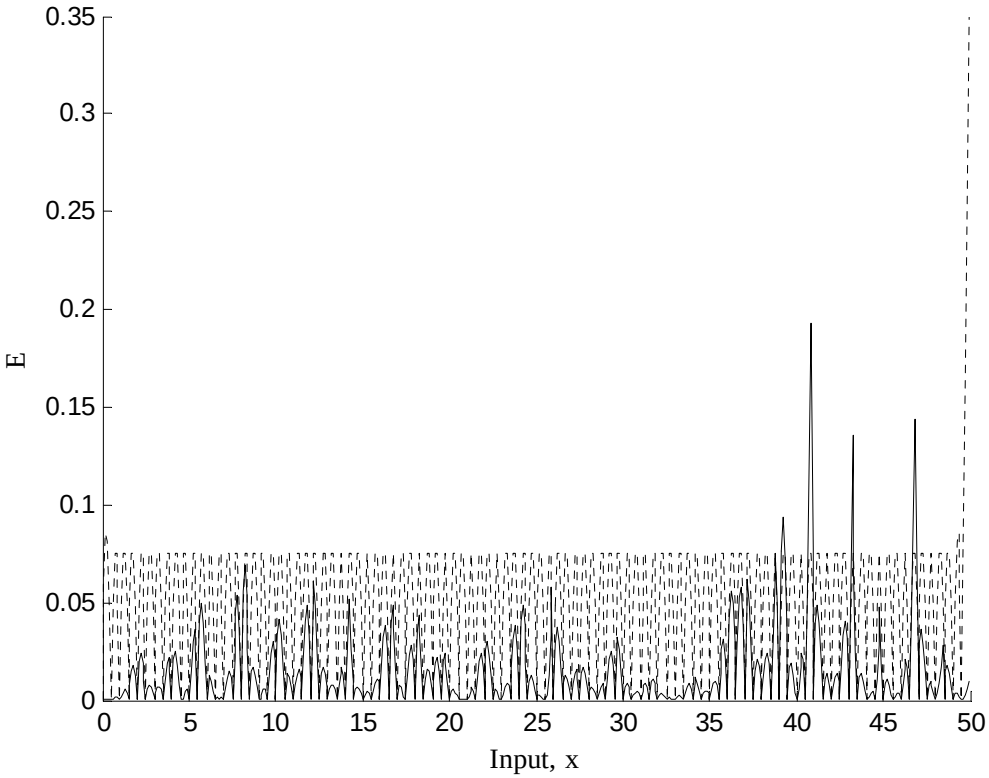


Figure (5.18): 'Spiky' Data Example (100 training points, Matérn Cov. Function) - GP Model Error (solid line) and Variance (2σ) output (dashed line).

Test 4 – Matérn Covariance Function (34 training points)

The Matérn covariance function is now to be applied to the problem of approximating the underlying function using the smaller training (34 point) dataset. Training the hyperparameters resulted in $\boldsymbol{\theta}_{MP}$.= (θ_1 = 2.3542, θ_2 = 1.0800, θ_3 = 0.2114). In Figure (5.19) the GP predictions are compared with the underlying data for the model trained on 34 training points, and the variance output of this GP model is again plotted on the same axis as the model error, as shown in Figure (5.20). Using the same performance validation measures the following were calculated, Mean-Square Error (MSE) of 0.0124, Log Predictive Density (LPD) of 0.3731, and log likelihood (LL) of –40.901. Overall, the predictive accuracy of the identified GP model is certainly less than that of the previous example that was trained on a larger number of training points. A more interesting comparison is the performance of this Matérn GP model with that of the Squared Exponential GP model identified with the same 34 training points. Therefore, in the Figure (5.19), the training points have been plotted as well as the mean predictions to allow easier comparison with Figure (5.16).

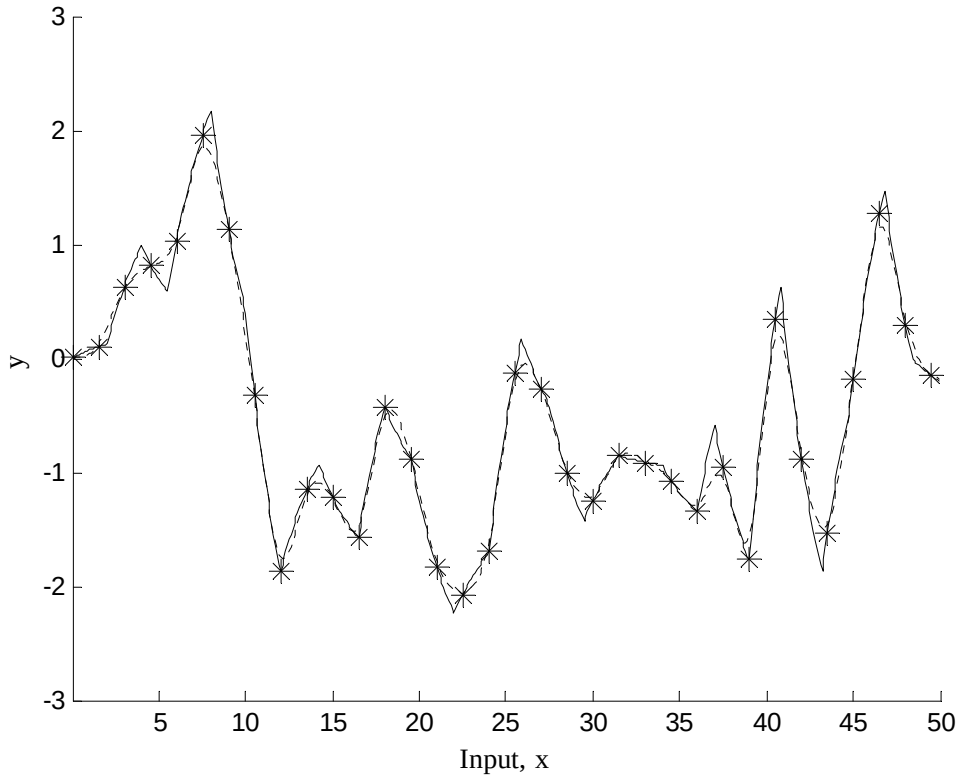


Figure (5.19): 'Spiky' Data Example (34 training points (Marked), Matérn Cov. Function) - GP Mean predictions (dotted line) vs. Underlying function (solid line)

In comparing the Matérn GP model with the Squared Exponential model, we can see that the predictive performance of the former is considerably better than that of the latter (i.e. MSE drops from 0.0440 to 0.0124). Most notably the mean predictions also manage to cope better with the sharper peaks/troughs in the data. Furthermore, unlike the Squared Exponential GP model, the mean predictions of the Matérn GP model successfully bisect the included training points.

Turning our attention to the variance output of this Matérn GP model as depicted in Figure (5.20), we can see that the overall level of the variance output is considerably higher than that of the previous example that included more training data. Furthermore, the variance output fails to reach zero even at test points that are equal to observed training points. Nevertheless, in comparison to the Squared Exponential Model, the level of the variance does reflect the location of the training data rather than just remain constant across the input space. Taken together, the reasonable mean predictive performance and the rather high (but still reactive) variance output of this Matérn GP model suggests that the size of the training set (34 points) is approaching the lower limit of including enough information with which

to identify suitable hyperparameters. This is indeed to found to be the case when a further few training points are removed, and the GP mean predictions become highly inaccurate.

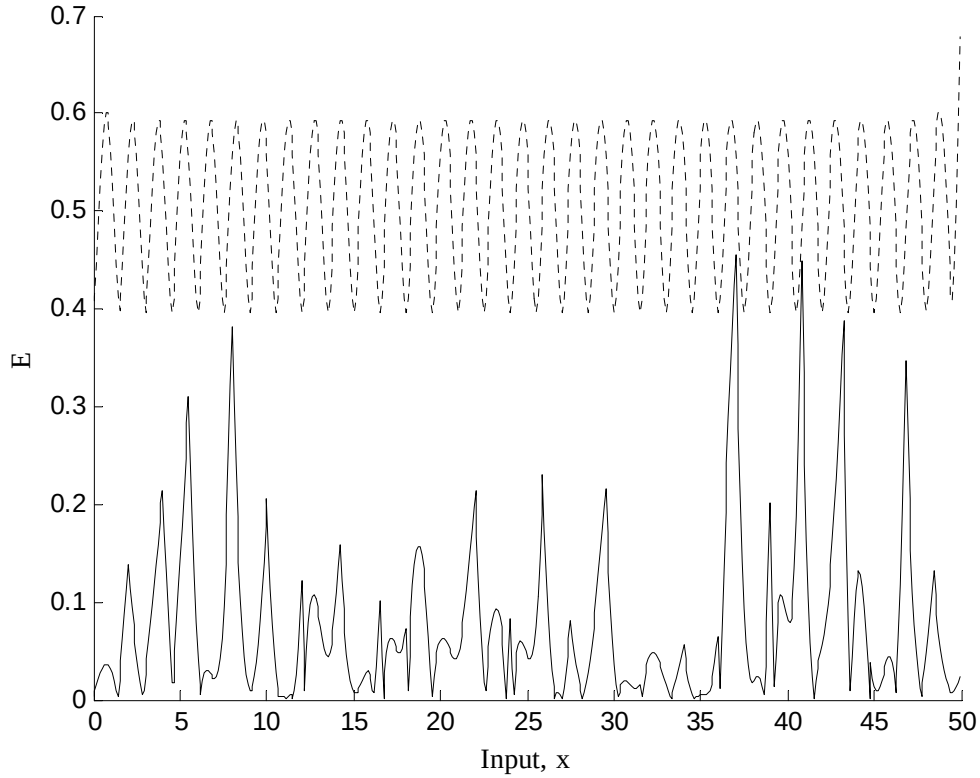


Figure (5.20): 'Spiky' Data Example (34 training points, Matérn Cov. Function) - GP Model Error (solid line) and Variance (2σ) output (dashed line).

Furthermore, up until this point we have not commented on the inconsistencies present in the validation performance measures of these 4 test examples. Whilst the MSE measure of model accuracy can be seen to reflect what may be visually interpreted in the accompanying figures (i.e. the MSE of the larger training datasets outperform the smaller datasets, and the Matérn GP models outperform their comparably sized Squared Exponential GP models). The more probabilistic measures of performance do not concur with the MSE measures, as the LPD and LL of the Squared Exponential models are smaller and therefore 'better' than those of the Matérn examples. This is slightly troubling but as the near constant variance output of both Squared Exponential models seems to indicate that the optimisation process has not been entirely successful, the variance output cannot be seen to be particularly reliable. Therefore, this highlights the need to employ a number of different measures of model performance, and also to ensure that such validation measures also concur with what is visually interpretable or subjectively plausible.

Overall, this ‘spiky’ data example has attempted to show (over the 4 tests) the limitations of the Squared Exponential covariance function when used to tackle sharply varying data. This is important due to the fact that the Squared Exponential covariance function has become almost ubiquitous in its selection for GP model implementations (especially for system identification purposes). In this example a Matérn covariance function was found to be more suitable for this kind of problem, and this is to be expected due to its less stringent prior assumption over the smoothness or differentiability of the underlying function. In the forthcoming sections devoted to identifying real experimental systems, the smoothness properties of these systems are found to be compatible with that of the popular Squared Exponential covariance function. Nevertheless, for some applications this will not be the case and the use of alternative covariance functions (such as the Matérn covariance function used here) is something that should be considered.

5.6.5) Lorenz Attractor – A Dynamic Nonlinear Example

The Lorenz attractor is a 3 dimensional model structure defined by 3 differential equations that was developed by Edward Lorenz in 1961 from a simplified analytical model of thermal convection in a layer of fluid. The Lorenz attractor is perhaps the most widely known example of a **chaotic** system, and played an important role in the general development of chaos theory. Chaotic systems can be described generally as nonlinear deterministic systems that are very sensitive to initial conditions, are highly periodic, and exhibit behaviour where phase space (trajectory) overlaps occur in different regions of the operating space (termed topological mixing). The deterministic characteristic is important as unlike systems that exhibit random behaviour, chaotic systems can be described exactly through analytical or parametric models. However, despite this characteristic, the prediction of future behaviour is difficult due to the particular properties of chaotic behaviour. Of paramount importance in the analysis of chaotic behaviour is the sensitivity to initial conditions where even miniscule changes in the initial conditions can lead to drastically different responses as the system behaviour evolves over time. This feature of chaotic systems is popularly known as the 'Butterfly effect'. This terminology originates from the influential paper by Lorenz (1972) titled '*Predictability: Does the flap of a butterfly's wings set off a tornado in Texas?*' and also relates to the shape of trajectory of the Lorenz attractor as depicted in Figure (5.22). The differential equations that describe the Lorenz attractor are:

$$\frac{dx}{dt} = \sigma(y - x) \tag{5.17}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{5.18}$$

$$\frac{dz}{dt} = xy - \beta z \tag{5.19}$$

Where $\sigma, \rho, \beta > 0$, and σ is called the Prandtl number, and ρ is called the Rayleigh number. Typically, these values are constants, and for chaotic behaviour given the values $\sigma = 10$, $\rho = 28$, $\beta = 8/3$. Given a set of initial conditions for *x*, *y* and *z*, the behaviour of the system can then be simulated through applying numerical methods. For this example the initial conditions where $x = -2$, $y = -3.5$, $z = 21$ and the 'ode45' Runge-Kutta solver in Matlab was utilised to generate the data. The three differential equations are displayed in

Figure (5.21) for a 40 second time-scale, and the phase plane or trajectory of the system displayed in Figure (5.22).

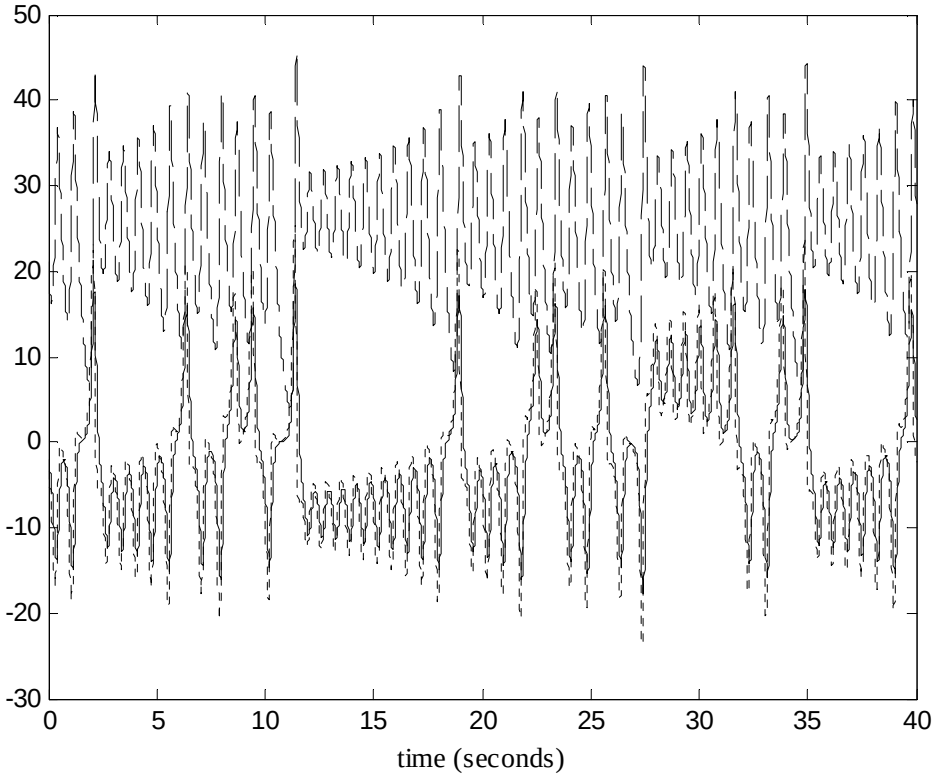
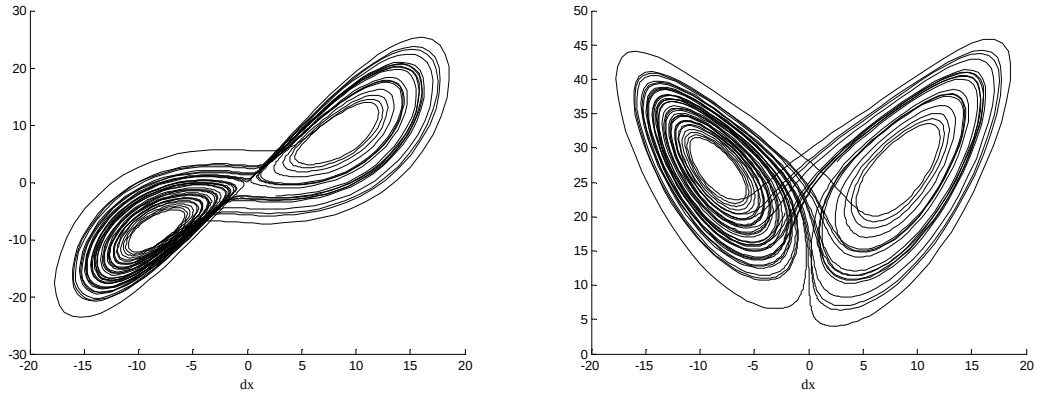


Figure (5.21): Lorenz Attractor - dx/dt (solid line), dy/dt (dotted line) and dz/dt (dashed line).



(a) **(b)**

Figure (5.22): 'Butterfly Effect' of Lorenz Attractor - Chart (a) shows trajectory dy/dt vs. dx/dt. Chart (b) shows trajectory dz/dt vs. dx/dt.

In this example, the simulated time-series data of the Lorenz attractor is to be used to demonstrate the process of identifying nonlinear dynamic systems using the GP modelling approach. Therefore, a detailed investigation into the Lorenz attractor or chaotic behaviour

in general is not conducted. For more detailed information on chaos theory, the textbooks by Alligood (1997) and Gollub and Baker (1996) are good resources. As chaotic systems are highly nonlinear whilst remaining deterministic, they are often used in the comparison and benchmarking of alternative prediction methods. In Girard (2004) the Mackey-Glass series (generated from a model of blood cell count in leukaemia patients) was used as an example application for the GP model. However, as the Mackey-Glass time-series is described by only one variable and is therefore not particularly comparable with the typical problems of system identification where both input and output data exists.

As the Lorenz attractor is a 3 dimensional dynamic model, it is first necessary to decide which quantity we wish to predict (i.e. model output), and which quantities are to be used as model inputs. Furthermore, as this example is dynamic in nature, previous inputs and outputs may also be used as inputs to the model. Firstly, in order to avoid confusion and inconsistencies in the nomenclature used, we are to rename the previous characteristic equations as:

$$A(k) = \frac{dx}{dt} = \sigma(y - x) \tag{5.20}$$

$$B(k) = \frac{dy}{dt} = x(\rho - z) - y \tag{5.21}$$

$$C(k) = \frac{dz}{dt} = xy - \beta z \tag{5.22}$$

To begin with the identification problem was defined as trying to identify $C(k)$ using $A(k)$ and $B(k)$ as the model inputs (see Test 1 below). A second implementation is then tackled where the identification problem was defined as trying to identify $C(k)$ using $A(k)$ and $B(k)$ together with the previous output $C(k\text{-}1)$ (i.e. one-step back) as the model inputs, resulting in an ARX model structure (see Test 2 below). Therefore, as previous output information is to be used as an additional input, this example application can therefore be termed as one-step ahead prediction rather than simulation. However, before tackling these models, a couple of important issues regarding the practical implementation of the approach to dynamic problems must first be discussed.

5.6.5.1) Incorporating Delayed or Regressed Inputs/Outputs

Firstly, it is the construction of the model structure that is to be an important point of discussion in this example implementation of the GP modelling approach. As discussed previously, one of the main difficulties of the GP modelling approach is that it is computationally expensive to include large quantities of data in the training dataset. Consequently, this puts pressure on the amount of training data that can be readily included, which may result in the choice of a sampling rate that is slower (i.e. larger intervals between points) than would normally be employed for identification purposes. However, it is perhaps unreasonable to place such a constraint on the size and sampling rate of any test dataset that the model is to be applied to (e.g. we may have to sample every 0.1 seconds in order to reduce the size of the training data, but we might want to predict every 0.05 seconds). This issue is particularly important when dynamic systems are considered where previous inputs and outputs are routinely employed as model inputs. In effect, if a previous output y(k-1) is to be used as an input, the training data may be sampled in such a manner that y(k-1) corresponds to *X* seconds previous, whereas the test dataset may be sampled in a manner that y(k-1) corresponds to *Y* seconds previous (e.g. for training data sampled every 0.1 seconds, one-step back corresponds to 2-steps back if the test data is sampled every 0.05 seconds).

Therefore, an important aspect in the implementation of the GP modelling approach for dynamic problems is to ensure that the training and test data are pre-processed in such a manner that allows the desired model structure to be maintained. As a result, unless the training data and test data are to be sampled at the same rate, once the training dataset has been pre-processed, the test data must also be processed in order to ensure consistency. Furthermore, the process of creating a training dataset must also keep in mind any subsequent requirements over the ultimate use of the model (e.g. the need to test at a certain interval). Otherwise, it is possible to create a training dataset that has discarded information that may be needed to make test predictions. As an example, if the training data is sampled every 0.5 seconds and the test data sampled every 0.1 seconds, a model trained using a previous output y(k-1) as the input, should be tested with a model that employs the 5^{th} previous output y(k-5). Therefore, the test data must be processed to start predictions (k=1) at the 5^{th} point in the dataset, whilst holding onto the previous 5 data points for prediction (and updating this variable as the prediction horizon proceeds). As a result, for problems

that are to include multiple inputs composed of different delayed inputs/outputs, the task of ensuring the consistency of the training and test data sets becomes more challenging.

This need to process the test data in order to remain compatible with the model structure is certainly not something that is unique to the GP modelling approach. However, due to the possible need to minimise the number of included training points, the potential for different sampling rates to be employed for training and test data is perhaps greater than in alternative approaches. Furthermore, this need for careful processing of the test data can be seen to be a notable drawback of utilising this nonparametric approach where the data is directly included in the model (i.e. the covariance matrix can be interpreted as a precise spatial mapping). For parametric models, where the training data is used only to optimise a number of parameters, careful processing of test data is not something that is normally required if the test and training data are similarly sampled (as would normally be the case).

5.6.5.2) Normalising and Rescaling Data

Another important aspect with regard to the implementation of the GP modelling approach is the potential need for the rescaling and normalisation of the training data. Firstly, in order to remain consistent with the Bayesian framework of the approach where a zero-mean prior is defined, the output or target training data should also have a zero mean (note that in the previous simulated examples the functions varied in and around zero). This can be easily achieved through calculating the mean value of the target data, and then subtracting this value (or offset) from the target data. A further potential source of problems is the scaling of the input variables, where if large differences in the relative scaling of different inputs exist, the optimisation of the hyperparameters can become difficult. Therefore, by calculating the standard deviation of the different inputs, the scaling of each input dimension can be checked and then re-scaled if considerable differences exist.

At this point the potential need for the normalisation of the training target data and re-scaling of the training input data has been discussed. However, for models that are to employ previous outputs as model inputs, this normalisation and rescaling of the training data must be treated with care. As discussed before, it is fundamental that the input training and input test data are consistent with one another from a timing perspective, but also now from a scaling perspective. Therefore, employing previous output/target data (that has been

normalised) as a model input, and subsequently training hyperparameters under this condition, means that the corresponding test input data (i.e. the previous output) must also be normalised accordingly. This means that when performing multiple-step ahead prediction, the calculated output prediction must be normalised before being fed back for use as a model input.

Overall, whilst these implementation issues regarding the incorporation of previous inputs/outputs and the normalisation and rescaling of the data are important, it is also worth noting that it is often the case that a reasonable model can be identified even if these considerations are not implemented perfectly. As the processing of the training and test data can become quite complex, and time-consuming if a number of iterations of training and testing procedure are required, it is sometimes easy to overlook some of the more subtle aspects and assume that everything is correct as a reasonable model performance has been achieved. Nevertheless, if these strategies are correctly employed, a greater level of model performance should be possible.

Test 1 – Predicting $C(k)$ using $A(k)$ and $B(k)$ as inputs

In this example the following model structure is employed: $C(k)$ is the model output, and use $A(k)$ and $B(k)$ are the model inputs. The differential equations were then employed to generate the dataset displayed in Figure (5.21). In order to adhere to the basic principles of cross-validation, this data is then split into separate training and test datasets. Therefore, it is important to be clear that the data used to train the model will not be used as a test dataset. For this example, the data was partitioned evenly (at 20 seconds) into test and training datasets. The original empirical data generated from the differential equations was done so at an interval of 0.01 seconds for 40 seconds, resulting in 2000 points each in the test and training datasets. Therefore, in order to reduce the computational burden, the training dataset was then sampled by a factor of 4 to give 500 points, and the test data was sampled by a factor of 2 to give 1000 points. This difference in the sampling rate of the training and test datasets will not be important for this test as no previous output information is to be fed back as an additional input.

After this initial processing of the data, the next stage to consider is the normalisation and scaling of the data. Examining the mean and standard deviation of the three quantities the

following values were calculated**:** $mean(A(k)) = -3.91$, $std(A(k)) = 6.94$**),** $mean(B(k)) = -3.92$, $std(B(k)) = 8.03$**,** and $mean(C(k)) = 23.80$, $std(C(k)) = 8.37$. Overall, we can see that the standard deviation of the three variables is quite similar, so rescaling of the data to be employed as inputs is not required. However, as the mean of target $C(k)$ is significantly different from the prior zero-mean assumption inherent in the GP modelling approach, the target values should be offset by this mean value. As no previous regressive outputs are to be used in this implementation as model inputs, this normalisation of the target data will not affect the input data. However, this offset value must be retained and added to the computed predictions.

The next stage to consider is the selection of an appropriate covariance function along with the optimisation of suitable hyperparameters. For this example, the Squared Exponential covariance function was employed with the same initial values chosen for hyperparameters as before (-1 for all). As this example model is to employ two inputs, a second length-scale hyperparameter is required (i.e. vertical variance or amplitude hyperparameter θ_1, a length-scale parameter θ_2 (for $A(k)$ input dimension), a length-scale parameter θ_3 (for $B(k)$ input dimension), and a noise parameter θ_4.). Applying the same marginal likelihood maximisation optimisation scheme as before results in $\boldsymbol{\theta}_{MP}$.= (θ_1 = 2.0439 θ_2 = 5.1122, θ_3 = 10.0623, θ_4 = 1.9969). Now that the GP prior has been defined, the predictive mean of the posterior can then be calculated for all test inputs and compared with the real function data as in Figure (5.23) on the next page.

Overall, the GP model has provided a reasonable representation of the behaviour of target function but significant error can be seen to exist. This is reflected in the validation measures of performance where the following were calculated: Mean-Square Error (MSE) of 6.4987, Log Predictive Density (LPD) of -10.2683, and log likelihood (LL) of -1.1795e+003. As before, it is informative to plot the model error on the same chart as the variance output of the GP model as in Figure (5.24).

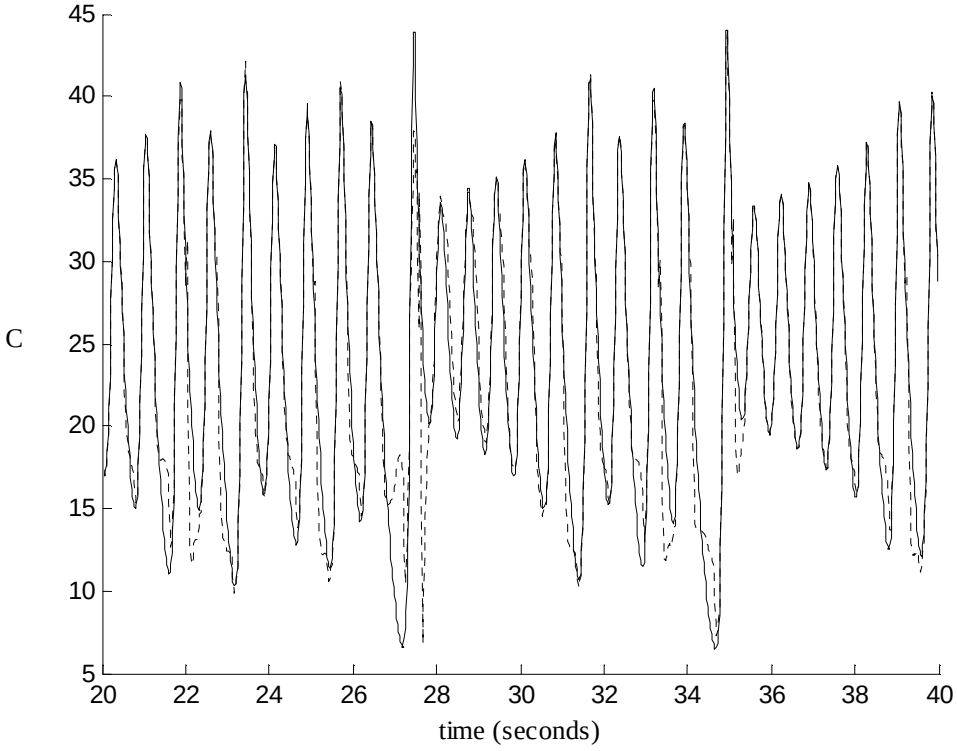


Figure (5.23): Lorenz Example - GP Mean predictions (dotted line) vs. Underlying function (solid line).

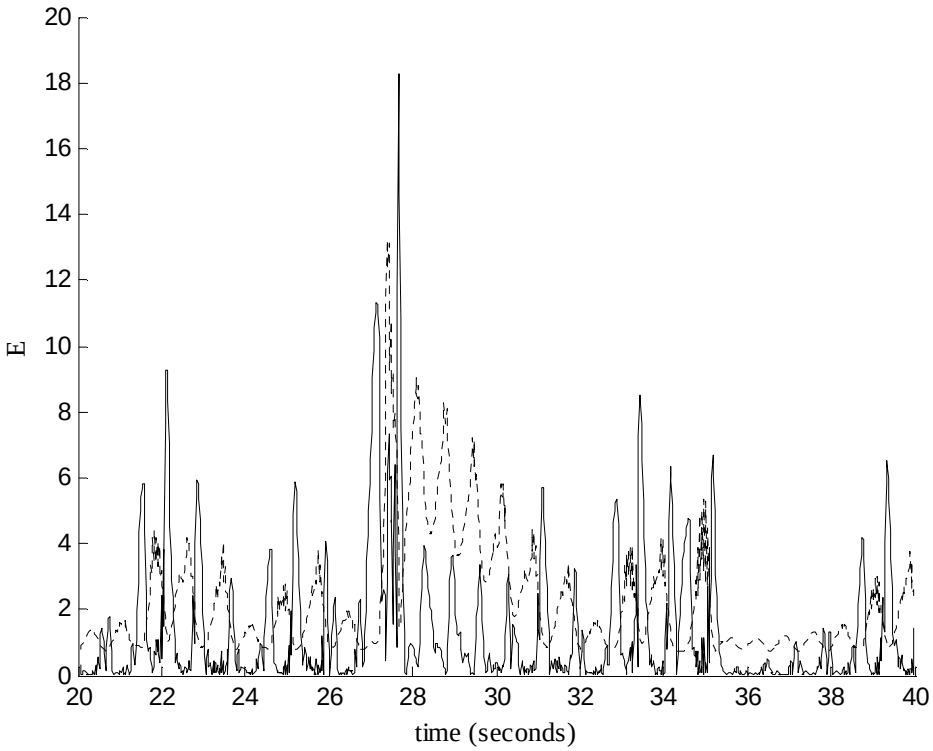


Figure (5.24): Lorenz Example - GP Model Error (solid line) and Variance (2σ) output (dashed line).

In Figure (5.24), the peaks in the model error can be seen to mostly correspond with the peaks in the variance output. Furthermore, a notable increase in the overall level of the variance output can be observed in the middle portion of the test data between t = ~27 seconds and t= ~31 seconds. We can interpret this increase in the variance output, and sharp increase in the model error at the beginning of this region of test data, by referring to the training data included in the model (first 20 seconds of data in Figure 5.22). In the region ($27 < t < 31$) of the test data, the model inputs $A(k)$ and $B(k)$ can be seen to shift upward to a higher value and continue oscillating. However, in the first 20 seconds of this dataset that has been used for training, this upward shift in the observed data is not present. As a result, the training dataset does not contain sufficient information in the form of observed data with which to make accurate predictions at these input values, thus leading to more uncertain predictions that have a higher variance. This can be further understood by referring to the trajectory chart of Figure (5.22b), where the less frequent oscillations that occur at higher values can be interpreted as the right-hand 'wing' of the 'butterfly'. In this example, we have not included enough observations from this right-hand 'wing' in order to make accurate predictions there.

Overall, this example demonstrates the fundamental dependency that the GP modelling approach has on the quality of the training dataset. Of course by including more observations in the training dataset the quality of the model may be improved, but this may lead to considerable computational expense. Therefore, if strict controls are to be placed on the size of the training dataset, it is clear that the quality of the training dataset must be improved using some other strategy. It is at this point that prior knowledge of the system and available data can prove to be of significant importance. Rather than just employing an arbitrarily chosen block of the available data, if the training dataset can be pre-processed more carefully to cover a greater range of the operating space more evenly, a better overall model can often result. Conversely, if there are specific regions of operating space that are more common or more important, the training data should be concentrated into covering those areas. Either way, using prior knowledge of the system together with the overall objectives of the model can play an important role in developing the most suitable model.

In order to demonstrate this aspect of choosing the training dataset more carefully, the same analytical model and initial conditions were again used to generate data, but the time-scale of the collected data was extended to 100 seconds in order to gain a better appreciation of

the long-term behaviour of the system, see Figure (5.25). Overall, the input oscillations can be seen to repeatedly alternate about zero in an aperiodic manner, and this is indeed the case if alternative initial conditions are employed. Therefore, the initial model trained on the first 20 seconds of data can be seen to be particularly lacking in sufficient observations of the input data where the oscillations occur above zero. As a result, if the model is retrained to include more observations in this region of input space, the performance of the model should be improved. To demonstrate this, the model is now to be trained on the data present in the region $20 < t < 40$ where input oscillations occur above and below zero, and the performance of the model compared with the original model trained on the data contained in the region $0 < t < 20$.

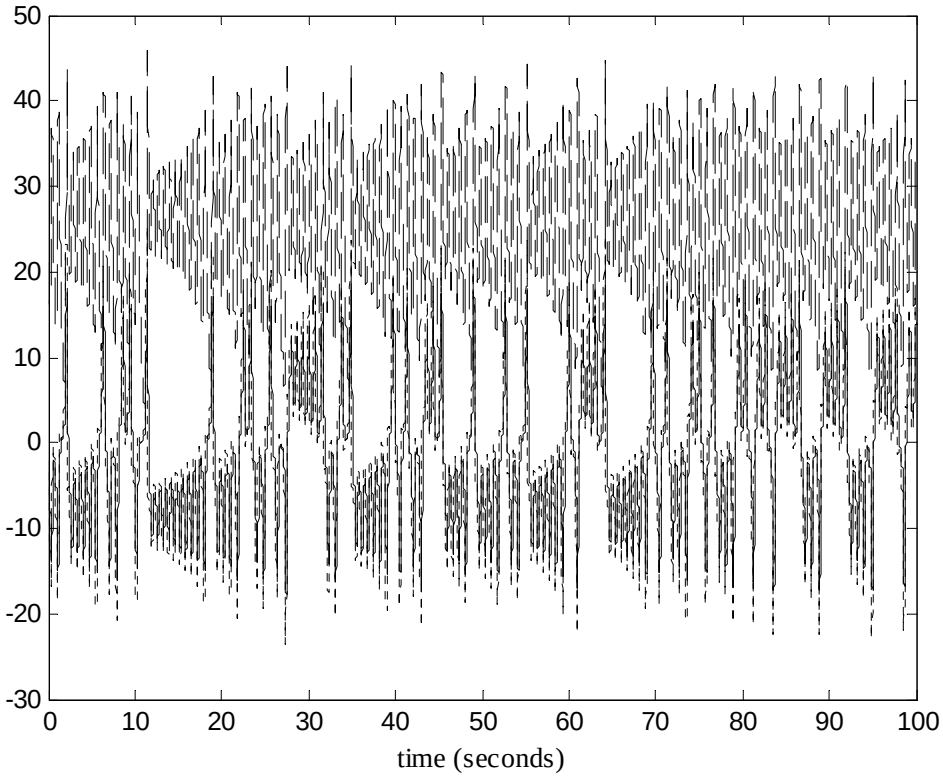
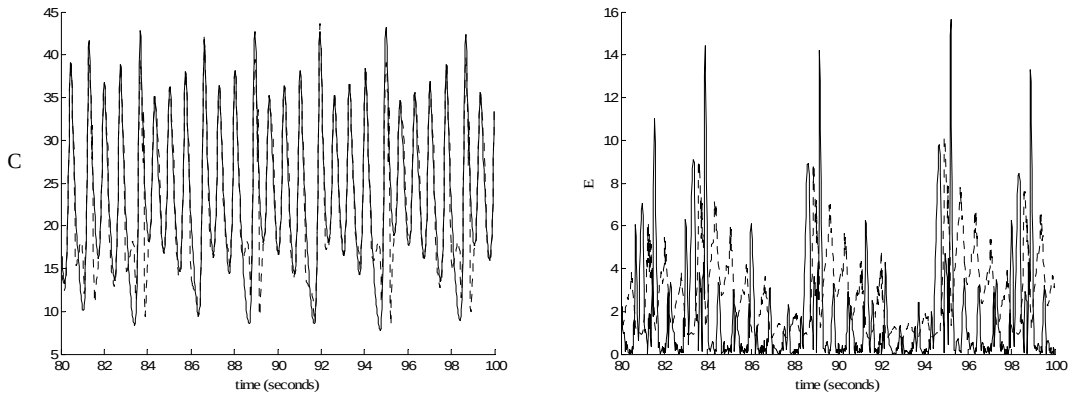


Figure (5.25): Lorenz Attractor - *A*(*k*) (solid line), *B*(*k*) (dotted line) and *C*(*k*) (dashed line).

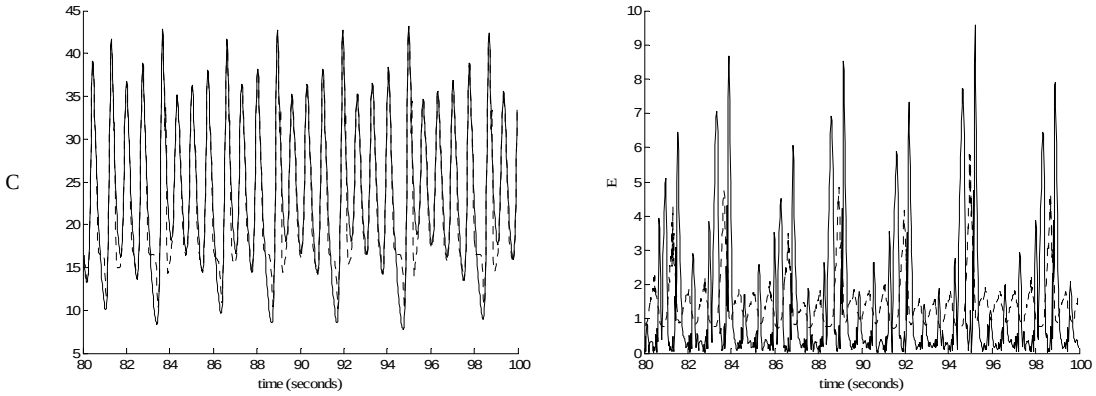
As before, the training datasets of both models are to be sampled so as to include 500 data points, and the same optimisation procedure followed. In order to test these two models, the data contained in the region $80 < t < 100$ is to be used, and the model performance can seen in Figure (5.26) and Figure (5.27). Overall, we can see that the level of error present in the model trained on the data $20 < t < 40$ is mostly superior to that of the model trained on the data $0 < t < 20$ with the large peaks in the model error corresponding to the larger

amplitude oscillations. The variance output of this newly trained model is also typically lower and more consistent between these peaks in model error. For the model trained on the data on the original $0 < t < 20$ data, the validation measures where Mean-Square Error (MSE) of 9.6090, Log Predictive Density (LPD) of -12.6828, and log likelihood (LL) of -1.1795e+003. For the model trained on the data on the new $20 < t < 40$ data, the validation measures where Mean-Square Error (MSE) of 5.3996, Log Predictive Density (LPD) of -19.4800, and log likelihood (LL) of -1.2148e+003.



(a) **(b)**

Figure (5.26): Lorenz Example (trained on $0 < t < 20$) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dashed line).



(a) **(b)**

Figure (5.27): Lorenz Example (trained on $20 < t < 40$) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dashed line).

From these validation measures we can see that MSE accuracy of the model has improved as expected. In contrast, the LPD measure that relates to the variance output of the models shows that the first model has less uncertainty about its predictions. However as a

significant level of error still exists in both models, this is not something that is unduly concerning.

It is also worth pointing out that whilst this new GP model trained on the data contained in $20 < t < 40$ provides a slightly better performance than that trained with the data contained in $0 < t < 20$, this is partly due to the fact that the test region $80 < t < 100$ contains considerable amounts of input data above and below zero. If the test data is to primarily include input response data below zero the first model identified with data $0 < t < 20$ is likely to perform better as it has a greater concentration of training observations that are similar to the test points, whereas the second model has training data that is spread more evenly across input space. This is indeed the case in the data region $40 < t < 60$, where the first model outperforms the second. This is important in relation to this particular example application as due to the chaotic nature of the response and sensitivity to initial conditions, it is often possible to generate input data that oscillates above or below zero for prolonged periods of time before switching. Therefore, in order to identify a robust model that performs well across the input range, it is necessary to select the training data carefully in order to include as much information as possible in the limited space available.

Furthermore, it is necessary to employ test data that examines the performance of the model across as much of the input space as possible. Therefore, it is worth testing both of these models on a larger dataset $50 < t < 100$ in order to confirm which model is superior. For the model trained on the data on the original $0 < t < 20$ data, the validation measures where Mean-Square Error (MSE) of 6.3394, Log Predictive Density (LPD) of -10.1268, and log likelihood (LL) of -1.1795e+003. For the model trained on the data on the new $20 < t < 40$ data, the validation measures where Mean-Square Error (MSE) of 4.9244, Log Predictive Density (LPD) of -12.0030, and log likelihood (LL) of -1.2149e+003. From the MSE validation measures we can see that accuracy of the model trained using the data $20 < t < 40$ is remains slightly better than the model trained using the data $0 < t < 20$, and the LPD measure now also indicates that this model is less uncertainty associated with it. Note that for this example, the model performance is not plotted as the charts are difficult to read due to the frequency of the oscillations and length of the timescale.

Overall, the significant error in the model demonstrates the difficulty of predicting the chaotic behaviour of the Lorenz attractor using the chosen inputs: $A(k)$ and $B(k)$. This is due to the fact that these input variables are not particularly informative of the desired output

$C(k)$ (i.e. a relationship between the input and output oscillations is not easy to interpret). Therefore, in order to improve this model $C(k)$ additional or alternative inputs would be required.

Test 2 – Predicting $C(k)$ using $C(k\text{-}1)$, $A(k)$ and $B(k)$ as inputs

As this example application is dynamic in nature, the previous states of the inputs and outputs are available for use in predicting future behaviour. In this example the previously used inputs, $A(k)$ and $B(k)$, are to be augmented with previous or delayed output information $C(k\text{-}1)$ as an additional input to the model. Therefore, this GP model can now be understood as an implementation of one-step ahead prediction This example is to employ the same initial conditions and therefore use the same data as in the previous model implementation. As before, employing an additional input in the model structure means that another length-scale hyperparameter must be added to the squared exponential covariance function. Furthermore, as this additional input is to be the previous output fed back, the normalisation that was performed in the original pre-processing of the training data must also be applied to this input data $C(k\text{-}1)$. As before, after the test predictions are computed, the offset that results from this normalisation can then be added to the output predictions.

In addition, as the training data has been sampled to include 500 points resulting in a 0.04 second interval between data points, and the test data has been sampled so that a 0.02 second interval between data points exists, the previous output must be incorporated in a manner that ensures consistency in the timing. Therefore, the delayed or previous output (one-step back) of the training data is equivalent to the previous output (two-steps back) of the test data, i.e. 2×0.02s = 0.04s. As a result, the output must be stored in a variable after each prediction so it can then be used to calculate the appropriate subsequent prediction (2 steps in advance in this case). This process can be better understood by writing out the form of the inputs/outputs as below:

At the first test case, k=1:

Model Input: $\begin{bmatrix} C(k-2) & A(k) & B(k) \end{bmatrix}$

Model Output: $\begin{bmatrix} C(k) \end{bmatrix}$

Previous Output (stored for next prediction): $\begin{bmatrix} C(k-1) \end{bmatrix}$

At the second test case, k+1:

Model Input: $[C(k-1) \quad A(k+1) \quad B(k+1)]$

Model Output: $[C(k+1)]$

Previous Output (stored for next prediction): $[C(k)]$

At the third test case, k+2:

Model Input: $[C(k) \quad A(k+2) \quad B(k+2)]$

Model Output: $[C(k+2)]$

Previous Output (stored for next prediction): $[C(k+1)]$

And so on….

An obvious problem of including previous or regressed outputs is that for the initial test predictions, this input information would not appear to be available (i.e. for the first test case we need $C(k-2)$ and $C(k-1)$). In some applications we can employ knowledge over the initial conditions of the system in order to provide this input information. Furthermore, in this simulated example, where test data has been selected from a larger set of empirical data, it is straightforward to include some observations of the output that immediately precede the chosen start of the test data (i.e. we can include observations $C(k-2)$ and $C(k-1)$ for use as input information, with the remaining observations of $C(k)$ being used for comparison with the model predictions).

However, in other cases it is possible that such initial conditions are not available (e.g. applying the model online to fresh datasets) and we must make the first few predictions using whatever input information is available (i.e. only $A(k)$ and $B(k)$). A problem with adopting such an approach is that the hyperparameters have been identified using the full complement of inputs in the training data, and therefore do not remain optimised if one or more of the inputs and accompanying hyperparameters are removed from the model set-up. This is due to the coupling that exists between the identified hyperparameters, and means that if the model set-up is altered, the hyperparameters must be retrained on similarly re-configured training data in order to remain optimal. As the optimisation of the hyperparameters can become a computationally demanding process if a large quantity of data is included in the training dataset, the retraining of the hyperparameters to identify a GP model that is only going to be used to predict a few initial test cases would seem to be computationally expensive from an overall perspective. Furthermore, for this particular

application, the GP model identified using only $A(k)$ and $B(k)$ as the model inputs has been shown to be of limited accuracy in Test 1. Therefore, the resulting overall model will provide a poor level of predictive accuracy for the initial test predictions, but is likely to improve significantly dramatically when the previous outputs become available and can be used as model inputs, as will be demonstrated in this example. Therefore, the strategy involving the inclusion of initial conditions in the form of recorded previous output data has been adopted for the applications investigated in this thesis.

Firstly, we are to employ the same training and test dataset as in the first part of Test 1 where the model is trained on the data included in the region $0 < t < 20$ and tested on data in the region $20 < t < 40$. The hyperparameters where identified as before, resulting in $\boldsymbol{\theta}_{MP}$.= (θ_1 = 51.2158, θ_2 = 9.4911, θ_3 = 59.5256, θ_4 = 28.2600, θ_5 = 0.0039), and the model performance is compared with the underlying data in Figure (5.28), and the model error and variance output of the GP model shown in Figure (5.29).

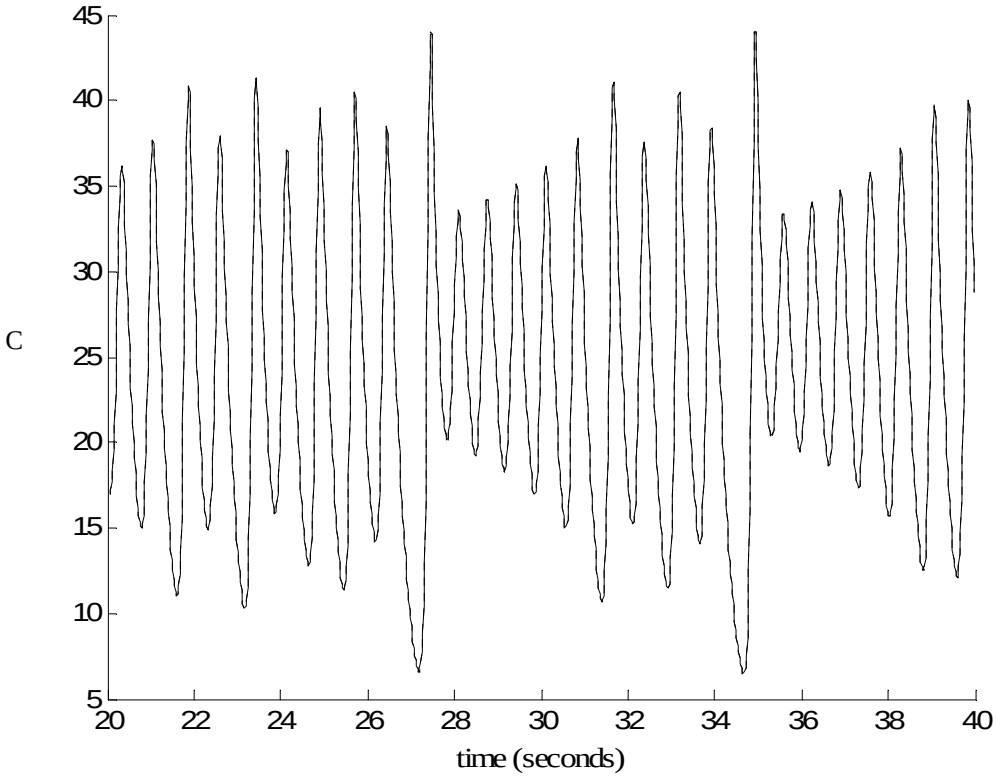


Figure (5.28): Lorenz Example - GP Mean predictions (dotted line) vs. Underlying function (solid line).

Overall, we can see that the accuracy of the model has improved dramatically by including the previous output as a model input, with the GP mean predictions being practically indistinguishable from the underlying test data, with validation measures Mean-Square Error (MSE) of 1.7524e-004, Log Predictive Density (LPD) of -9.2260, and log likelihood (LL) of 1.7919e+003. This great improvement in the model performance is not something that should be unexpected as the inclusion of the previous output information provides an input that is likely to be highly correlated with the desired output. Examining the model error and variance output of the GP model in Figure (5.29), we can clearly see that as in the previous example (shown in Figure (5.24)) both quantities grow in the middle portion of the test dataset as the training dataset includes a smaller number of observations in this region of input space.

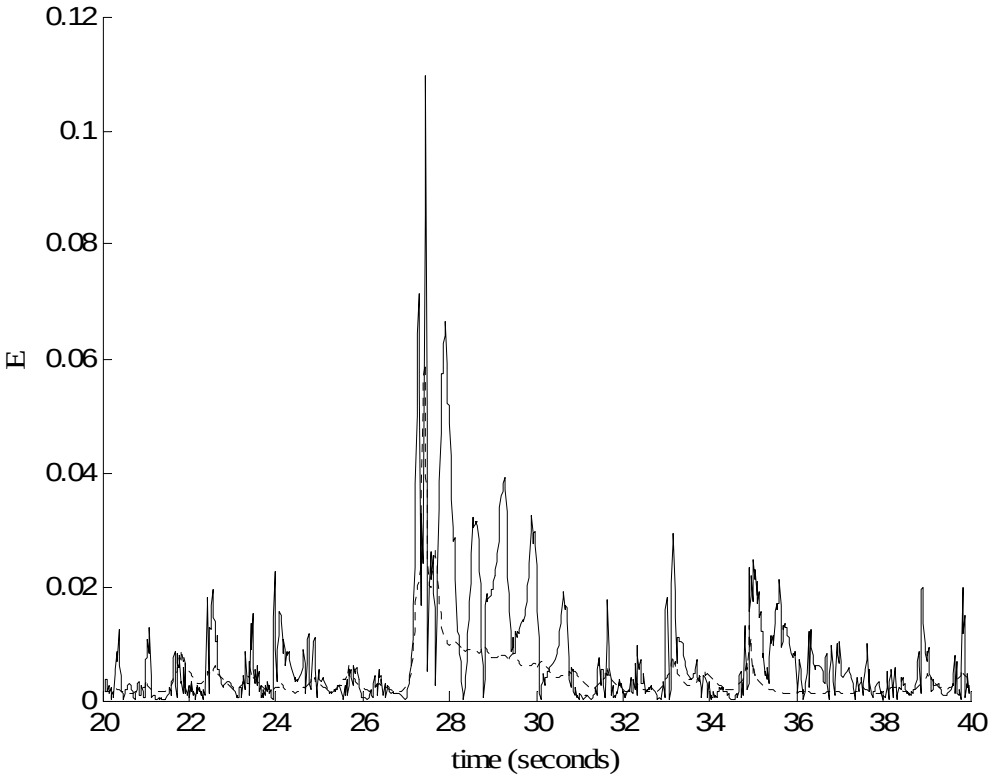
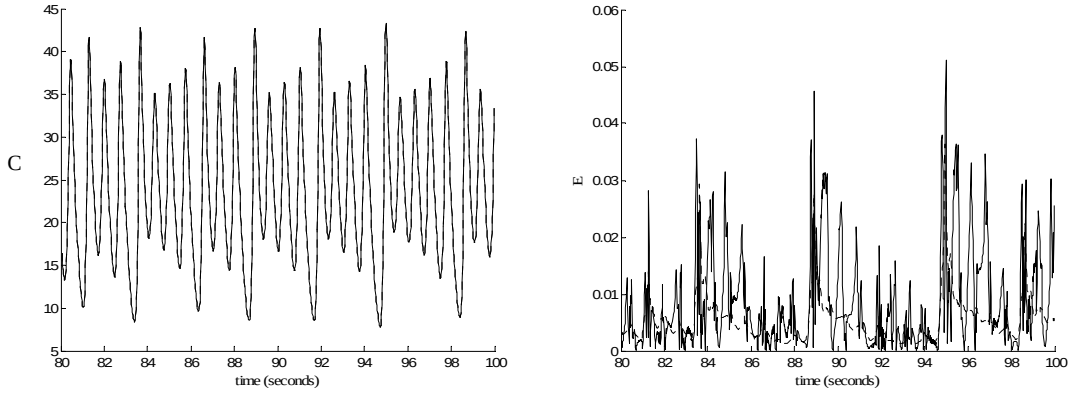


Figure (5.29): Lorenz Example - GP Model Error (solid line) and Variance (2σ) output (dashed line).

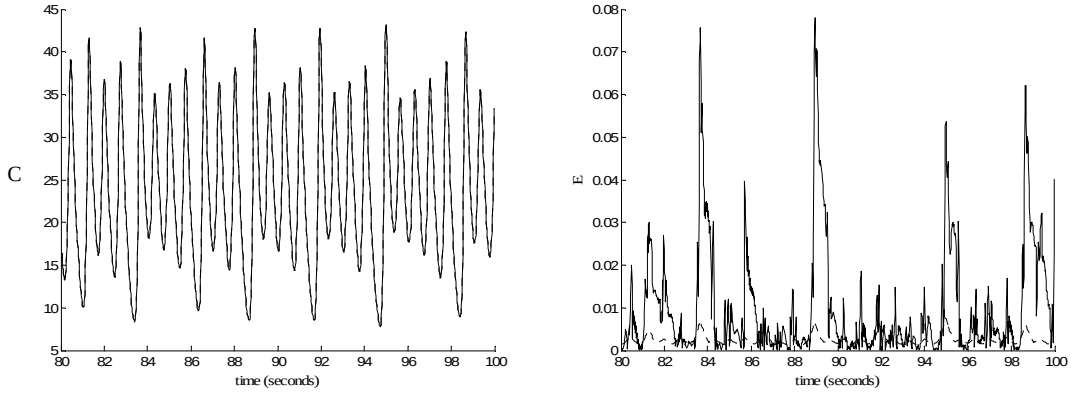
As in the previous example, we can attempt to modify the training dataset to better cover the whole of the input range. Therefore, the model identified using data contained in the region $0 < t < 20$ is to be compared with a model identified using data contained in the

region $20 < t < 40$, with the data contained in the region $80 < t < 100$ being used as the test data. The performance of the two models can be seen in Figure (5.30) and Figure (5.31).



(a) **(b)**

Figure (5.30): Lorenz Example using previous output (trained on $0 < t < 20$) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dashed line).



(a) **(b)**

Figure (5.31): Lorenz Example using previous output (trained on $20 < t < 40$) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dashed line).

Overall, we can see that both models provide an accurate representation of the underlying data. For the model trained on the data on the original $0 < t < 20$ dataset, the validation measures where Mean-Square Error (MSE) of 1.4252e-004, Log Predictive Density (LPD) of -3.7094, and log likelihood (LL) of 1.7919e+003. For the model trained on the data on the new $20 < t < 40$ dataset, the validation measures where Mean-Square Error (MSE) of 2.7576e-004, Log Predictive Density (LPD) of -43.3763, and log likelihood (LL) of 1.7377e+003. Therefore, in the measure of model error (MSE) the first model is actually

slightly superior to the second model, whilst in the measure of variance (LPD) it is the other way round. As in the previous example, the model error of the first model can be seen to be increase at test cases where the inputs $A(k)$ and $B(k)$ are above zero, where there is fewer included training observations. For the second model, the model error can be seen to be slightly lower than in the first model for some of the test cases, but increases sharply at test cases that coincide with the transition of the oscillating inputs $A(k)$ and $B(k)$ from below zero to above zero. Furthermore, through closer inspection of the model error and predictive variance of each model we can gain a greater appreciation of their different characteristics. Due to the more even spread of training data (in relation to the input space $A(k)$ and $B(k)$) of the second model trained on the data region $20 < t < 40$, the variance level of this model is consistently lower than that of the model trained on the data region $0 < t < 20$. This means that the second GP model is more confident over the predictions it has made, and is perhaps overconfident at certain test points where the model error is actually larger than that found in the first model where the model error and variance are more in tune with one another.

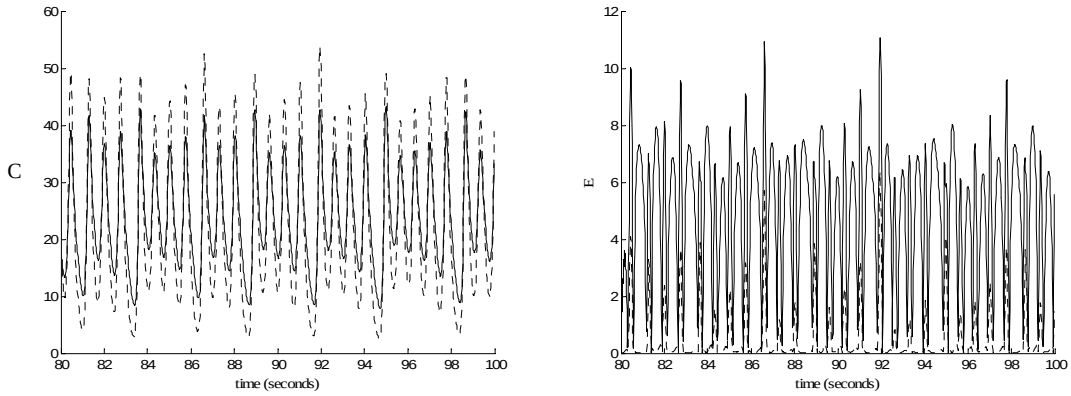
From inspecting the training datasets of both models, it is difficult to see exactly why the second model performs slightly worse than the first model with regard to modelling the output when this transition in the inputs occurs. It is clear that, unlike the first model, the training dataset of this second model is not particularly deficient in covering the available input space of this example, as the variance output of the model remains quite small. Instead, the slight advantage of the first model in representing this transition is due to this model's training dataset being slightly more informative at these particular test cases. This is an interesting result, as it shows that as long as some training data is included from across the operating range, and informative inputs are employed, a good model can be identified. Furthermore, this model is competitive with the alternative model where the training data is more evenly spread.

The informative nature of the training data is not something that has been given much consideration in the example applications so far. This is primarily due to the static nature of the nonlinearities considered, where the problem can almost be reduced to including as much data in the training dataset as is computationally feasible. However, for the identification of nonlinear dynamic systems, the design of the training dataset must also take into account the characteristics of the system under investigation.

One of the most important aspects of creating a training dataset is the selection of a suitable sampling rate for the experimental data collection and data pre-processing stages. In this example so far, the training data has been sampled using a 0.04 second interval between data points to provide a training dataset of 500 points, and the test data sampled so that 0.02 second interval between data points exists. Using this data along with previous output information has allowed us to identify some models of decent accuracy. However, one thing that has not been made clear so far is that it is considerably more computationally expensive and therefore slower to compute predictions where previous outputs are to be included as model inputs. This is due to the need to evaluate the predictive equations at each individual step in time in order to feed back the newly computed output. In the previous static examples, the predictive equations could be computed in a single iteration as the full compliments of test inputs are immediately available. Therefore, in order to speed up the identification of hyperparameters and evaluation of predictions it is worthwhile attempting to carefully reduce the size of the training dataset whilst retaining sufficient model performance, i.e. trade-off the computational efficiency of the model against model accuracy.

The problem with attempting to reduce the size of the training dataset is that we are potentially eliminating important information. Obviously, if the sampling rate is kept constant, by reducing the number of observations included in the training dataset, we are reducing the size of the time-scale that is to be included. Therefore, less information is likely to be included in the training dataset as the time-series may be too short to exhibit the full characteristics and operating range of the system. An alternative strategy is to reduce the sampling rate used in processing the training data. This will obviously allow us to include a longer time-series, thus potentially increasing the amount of operating space covered by the training dataset. The downside to reducing the sampling rate used for the training dataset is that we run the risk of failing to capture the some of the more subtle characteristics of the system, especially regions of the response that vary quickly. As the Lorenz attractor system is characterised by its highly oscillatory behaviour, if the sampling rate is reduced significantly these rapid changes and higher frequency characteristics are not going to be represented as well by using a smaller numbers of points. This is demonstrated below where the training data $0 < t < 20$ was sampled by a factor of 2 resulting in a training dataset of 250 points, with the data contained in the region $80 < t < 100$ again being used as the test data. The hyperparameters where identified as

before, resulting in $\boldsymbol{\theta}_{MP}$.= (θ_1 = 40.5500, θ_2 = 10.2231, θ_3 = 48.8862, θ_4 = 29.4780, θ_5 = 0.0049), and the and the model performance is compared with the underlying data in Figure (5.32). The validation measures where Mean-Square Error (MSE) of 30.2199, Log Predictive Density (LPD) of 1.7260, and log likelihood (LL) of 717.5159.



(a) **(b)**

Figure (5.32): Lorenz Example using previous output (trained on $0 < t < 20$, Smaller 250 point dataset) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dashed line).

The performance of this model can be seen to be substantially poorer than that of the previous model identified with twice the number of training examples. This is to be expected, but as discussed above, it is not purely the reduction in the number of points that is to blame for this decrease in model performance. It is the fact that the sampling rate employed is too slow and fails to capture the nature of the system dynamics. The effect of reducing the sampling rate is to concentrate the training data around the middle of the operating range, resulting in a model that fails to represent the underlying data at the extremities of the operating range. Looking more closely at the variance output of the model in comparison to the model error, it would appear that that these quantities are not well correlated at all. The variance output can be seen to grow substantially where the output transient is fastest (i.e. as it passes through the middle of the oscillation), and drops to a low level when the output transient slows down near the peak/trough of the oscillation. This is in contrast to the model error which is at its greatest when the output nears these peaks/troughs. This reinforces the point that we cannot use the variance output (or the LPD measure) as a definitive guide to the level of model error in the identified model. The

training data has failed to capture the dynamics at the extremities, so a model trained with this deficiency cannot hope to indicate this uncertainty.

Overall, this example has demonstrated the process of applying the GP modelling approach to a complex nonlinear system. As the GP modelling approach is dependent on the quality of the training data, in this example we have outlined the fundamental requirements behind identifying a suitable model. Firstly, that the training dataset must include data from across the operating range of the system in order to provide a robust model, and secondly that the training data must be sampled at an appropriate rate so that the dynamics of the system may be well represented. However, as the models where identified using portions of training data collected from a generated time-series, it is perhaps not truly demonstrative of the system identification process as no control over the input signals was possible. Therefore, in the next examples to follow, we are to take control over the input signals and apply the GP modelling approach to real laboratory-scale nonlinear systems.

5.7) Coupled Tank System

The GP modelling approach is now to be applied to a real laboratory-scale nonlinear dynamic system. The Coupled or 'Twin' Tank System, shown in Figure (5.33), is comprised of two water tanks identical in size and shape, coupled together through a small hole of known diameter between the dividing wall of the tanks. The system's control input is a variable flow rate into the first tank controlled through a variable speed water pump, with the height of the water present in both tanks used as measured variables. An output pipe is present in the second tank to prevent overflow. The system displays nonlinearity throughout the operating range due to orifice flow behaviour between the tanks and at the output flow pipe. The system is open-loop stable under all operating conditions with relatively long time constants (2-5 minutes) being observed.

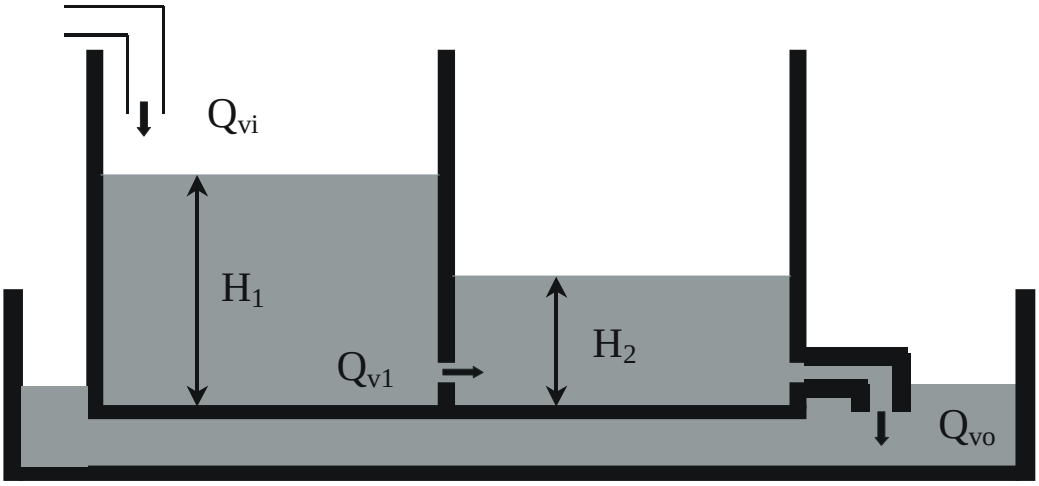


Figure (5.33): Coupled Tank System

This system can be seen to exhibit a first order nonlinear response between H1 and H2 across the entire operating range, where output data varies with excitation input data in a smooth and continuous manner. Consequently, the behaviour of this system would seem to be highly appropriate for the smoothly varying requirements imposed by the squared exponential covariance function employed in the most common GP model architecture. Further studies into this system can be found in Gong and Murray-Smith (1998), Chong and Li (2000), and Chan (2003). In this example application a number of different GP models are to be identified. In the next section, the process of developing a suitable training dataset for the identification of a GP model is to be examined using a simulated version of the Coupled Tanks system. Following on from this discussion, the experimental methods

applied to the collection of empirical data are described and an existing analytical model developed from first principles is also examined. Utilising the information gained from these initial investigations the GP modelling approach is then applied to empirical data collected from the real system.

5.7.1) Simulated Coupled Tanks System

In order to identify a GP model, a training dataset must first be created from any available empirical data. In contrast to the previous examples, in this application we are to take control over the input or excitation signals so that the full system identification process can be demonstrated. To provide a more in-depth investigation into the development of a suitable training dataset a number simulated models have also been employed, where a number of simple first order system models were developed to generate training and test datasets. The responses of these simulated models are designed to resemble the overall characteristics of the two real systems rather than provide an exact approximation. Using these simulated examples allowed different sets of training data to be collected more easily, and facilitated a greater degree of experimentation (e.g. model set-up, system dynamics, excitation signals, sampling rate etc.) than would be possible given the operational constraints present in performing data collection experiments on the real systems. The simple model structures were constructed using Simulink component of the Matlab environment as shown in Figure (5.34).

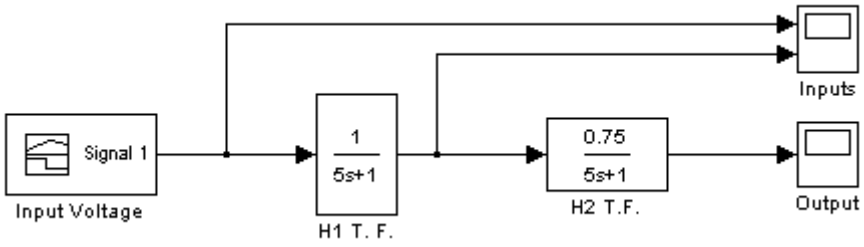


Figure 5.34: - Simulink Diagram of Simulated System

For the excitation signal “Input Voltage”, the various options within Simulink’s ‘Sources’ menus can be employed. Particularly useful are the random number generators to allow Pseudo-Random signals for inputs, and the ‘Signal Builder’ block that allows the user to construct a specific input from various transition steps. This input signal is to represent the input voltage used to control the water pump and therefore the inflow into Tank 1. The

initial 1st Order transfer function block 'H1 T.F.' represents the dynamics of the height of water in the first tank, and the second transfer function block 'H2 T.F.' represents the dynamics of the height of water in the second tank. The values for the numerators and denominators of both transfer functions were arbitrarily chosen, however the second transfer block was designed to vary along a similar timescale and with a lower amplitude than the first transfer block, thereby mimicking the overall nature of the real system where the height of water in the second tank (H2) varies smoothly in tandem with that of the height of water in the first tank (H1). Furthermore, the time constants associated with these transfer functions are much lower than for the real system (~20 seconds rather than ~5 minutes).

In previous sections of this thesis (see Sections (4.5.2) and (4.5.3)), important aspects regarding the size and conditioning of the covariance matrix and the implications for experimental design have been discussed. Of particular relevance to the process of experimental design is the fact that including prolonged periods of equilibrium or steady-state data in the training dataset can adversely affect the conditioning of the covariance matrix. As a result, in order to avoid this problem of matrix ill-conditioning, the input signals used must adequately excite the system so that the response data does not remain in steady-state for long periods. However, a significant problem can be seen to exist where the inputs and outputs can be seen to vary at different rates. To demonstrate this problem we can plot the response of both transfer function blocks to a step input signal, see Figure (5.35). In this chart we can see that if the full transient responses (H1 and H2 in the real system) are to be included, the corresponding input signal (pump input voltage in the real system) that remains constant must also be included, thus potentially degrading the conditioning of the training data. In contrast, we can see that as the transients associated with the H1 and H2 vary along a comparable time-scale, the problem of avoiding the inclusion of constant or equilibrium data would appear to be more easily dealt with.

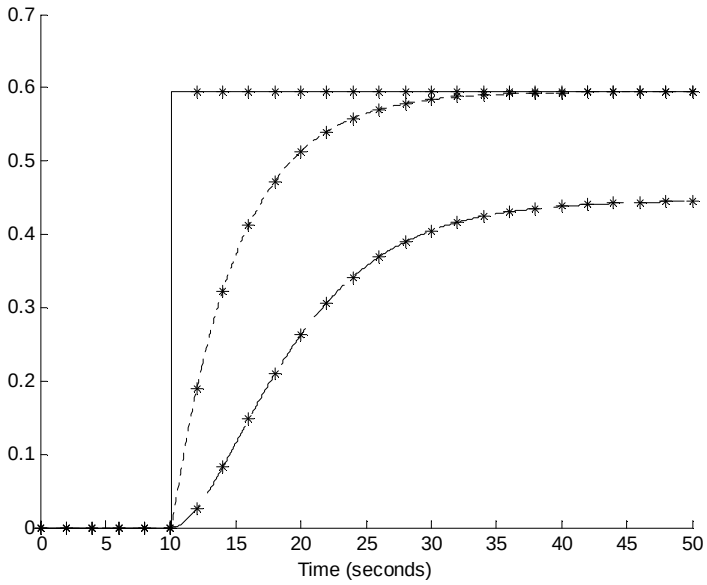


Figure (5.35): Step Response of Simulated System – Input Voltage (solid line), Height (H1) of water in Tank 1 (dotted line), Height (H2) of water in Tank 2 (dashed line).

5.7.1.1) Random Noise Excitation Signal

Therefore, if we are to model the relationship between the input voltage and the level of water in either tank, it would appear that the use of step excitation inputs is not a viable option. An alternative course of action would be to use an input signal that is constantly excited across the timescale and is therefore unlikely to result in prolonged periods of steady-state data being included in the training dataset. In Figure (5.36) a random (sampled Gaussian noise) excitation signal has been generated and applied to the simulated system. Random excitation inputs are especially popular in the development of black box models as they offer a suitable method of manipulating the system in an unbiased manner that reflects the lack of prior knowledge of the system that is often inherent.

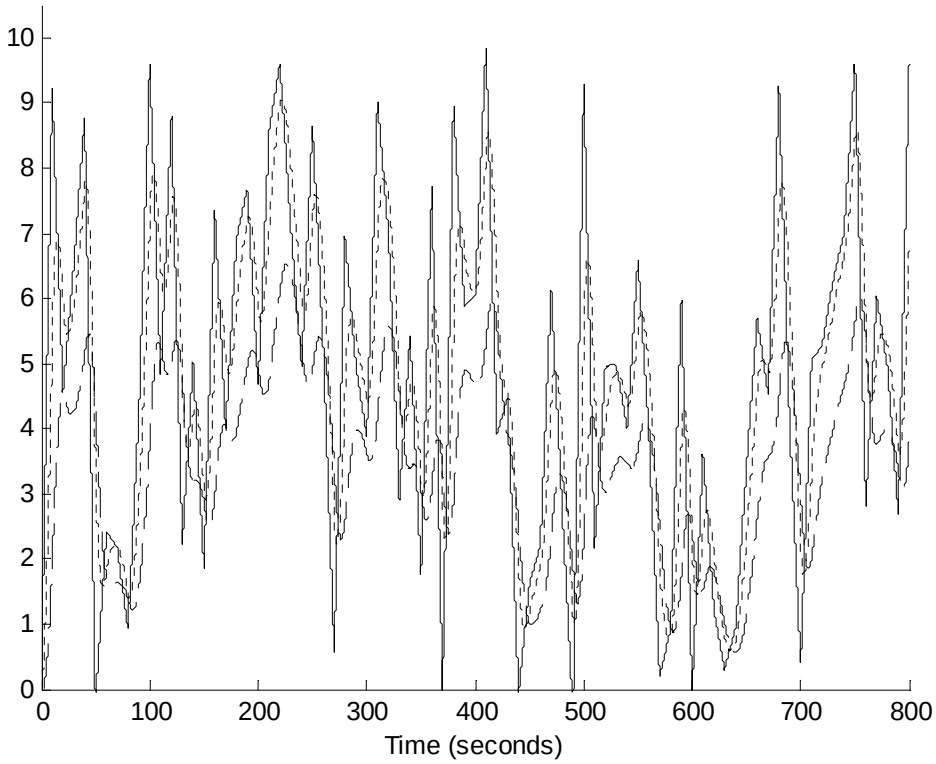
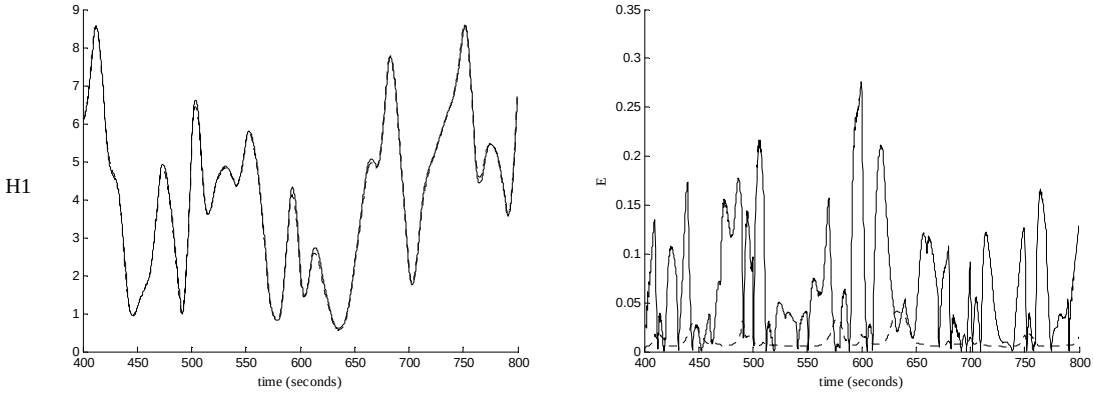


Figure (5.36): Coupled Tanks Simulated System (Random Noise Excitation Signal)– Input Voltage (solid line), Height (H1) of water in Tank 1 (dotted line), Height (H2) of water in Tank 2 (dashed line).

As in the previous Lorenz attractor example, in order to adhere to the basic principles of cross-validation, this data set is to be split into separate training and test datasets. For this example, the data set was partitioned evenly (at 400 seconds) into test and training datasets. The original empirical data generated from the simulated system was done so at an interval of 0.05 seconds for 400 seconds, resulting in 8000 points each in the test and training datasets. Therefore, in order to reduce the computational burden, the training dataset was then re-sampled using a factor of 20 to give 400 points, and the test data were re-sampled by a factor of 4 to give 2000 points.

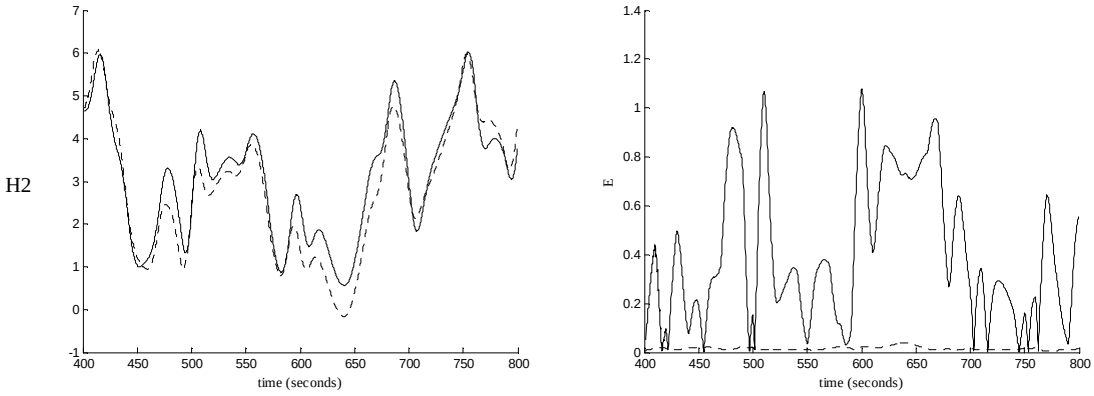
Using this random excited training data we can train three different models, firstly we model the relationship between the Input Voltage and H1, secondly the relationship between the Input Voltage and H2, and thirdly the relationship between H1 and H2. As in the previous dynamic example, the previous output is to be used as a second input for these models (i.e. one-step ahead prediction). Therefore, due to the difference in sampling rates between the test and training data sets (a factor of 5), the test data is to employ the output

point 5 steps before the current time (i.e. (k-5)) in order to remain consistent with the training data. In addition, the target data is again normalised in accordance with the zero-mean prior assumption, whilst the inputs are not rescaled, as their standard deviations are not hugely different. The squared exponential covariance function is again chosen, and the hyperparameters found using the same maximum likelihood maximisation scheme as before. These three model implementations (M1, M2, M3) were then tested with the model performance and variance output displayed in Figure (5.37), Figure (5.38) and Figure (5.39), with the validation measures given in Table (5.1).



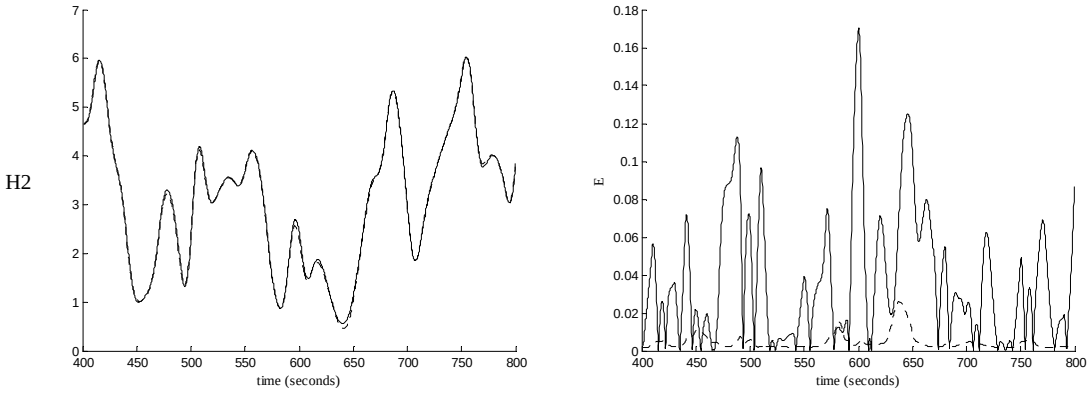
(a) **(b)**

Figure (5.37): Coupled Tanks Simulated Example (Model M1: Input V→H1) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).



(a) **(b)**

Figure (5.38): Coupled Tanks Simulated Example (Model M2: Input V→H2) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).



(a) **(b)**

Figure (5.39): Coupled Tanks Simulated Example (Model M3: H1→H2)- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Validation Measure	M1: Input V→H1	M2: Input V→H2	M3: H1→H2
MSE	0.0080	0.2412	0.0025
LPD	-273.7931	-3.0388e+003	-432.9226
LL	880.8169	512.6326	1.2552e+003

Table (5.1): Validation measures of Coupled Tank Simulated Model

Overall, we can see that the GP modelling approach achieves a good level of model performance for models M1 and M3, whilst the model M2 is significantly less accurate. For the model M3, as the input H1 and output H2 can be seen to vary smoothly with one another, it is not surprising that this model implementation has been successful. As a result, it is this model implementation that we will focus on when applying the GP modelling approach to the real empirical data. For the model M1, as the input (Input Voltage) and the output H1 are quite closely correlated (as shown in Figure (5.36)), the GP model can be seen to achieve a decent approximation. This is not the case for the model M2 where the input and output are not closely correlated, and a substantially poorer level of accuracy is found. This lack of model accuracy is due to the training data not providing sufficient information with which to make predictions, as the output varies at a far slower rate to that of the input. Therefore, the full response of the output to a change in the input is never captured by the training data, leading to error in the model. Indeed, it important to point out that the relationship between the Input Voltage and H2 can be understood as a second rather

than a first order system. As a result, the current ARX structure of model M2 is insufficiently complex (i.e. not enough inputs), leading to a poor description of this second order relationship.

Furthermore, in all of 3 of these models it is worth drawing attention to the fact that the variance output of each GP model is not particularly informative of the presence of model error. Whilst the LPD measure is better (more negative) for model M3 over M1, therefore in keeping with the MSE measure, the LPD measure of M2 is better than either despite attaining a substantially poorer accuracy. This again reinforces the fact that the variance output of the GP model is predominantly only really informative of where training data is sparse. In this example, the random excitation signal is sufficiently long enough that the whole operating range is covered relatively evenly by the training data. Therefore, the error in model M2 is predominantly due to the unsuitability of this model implementation, where the relationship between fast input transients to slower output transients is to be mapped. This aspect is something that we will return to in Section (5.7.6) where a mixed-model implementation is discussed, but for the moment we will concentrate on modelling the relationship between H1 and H2.

A problem with conducting system identification tests using such a random excitation signal (shown in Figure (5.36)) is that this kind of operating response is not particularly representative of how the real coupled tanks system is normally operated. In operating the real coupled tanks system, the system can be seen to settle into steady-state operating points very readily, and the recorded system response consists typically of relatively slow transitions between steady-state operating points. Therefore, the random excitation signal shown in Figure (5.36) is slightly artificial and whilst it offers a straightforward method for avoiding the inclusion of problematic steady-state data, it may be difficult to implement in practice. Another important aspect that has not yet been considered is the performance of these GP models for data that actually reaches steady-state. As the system is likely to spend a significant proportion of time under such operating conditions, the identified model must be able to provide accurate predictions of such steady-state behaviour.

5.7.1.2) Random Step Excitation Signal

In Figure (5.40) a different excitation signal is now applied to the simulated coupled tanks system where the input consists of a number of step transitions of different magnitudes. The magnitude of these input steps have been chosen arbitrarily to appear somewhat random in nature, and the time between steps was chosen to be ~25 seconds so as to let the full transients of H1 and H2 be captured. This time between steps was selected from inspecting the step response shown in Figure (5.35), where at ~25 seconds after the input step, both H1 and H2 both approach equilibrium.

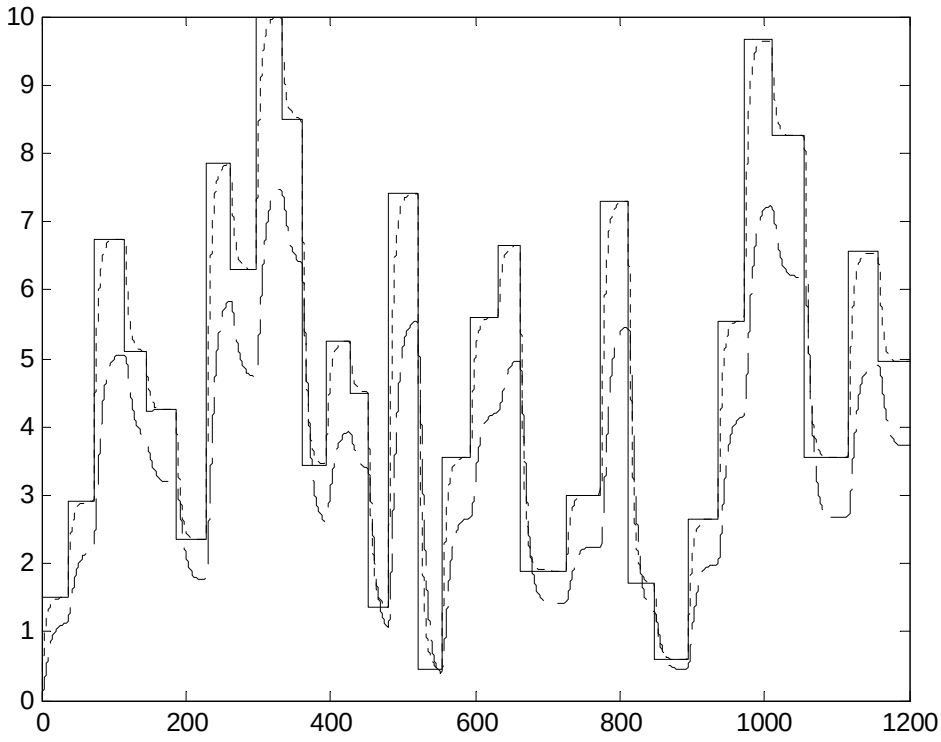
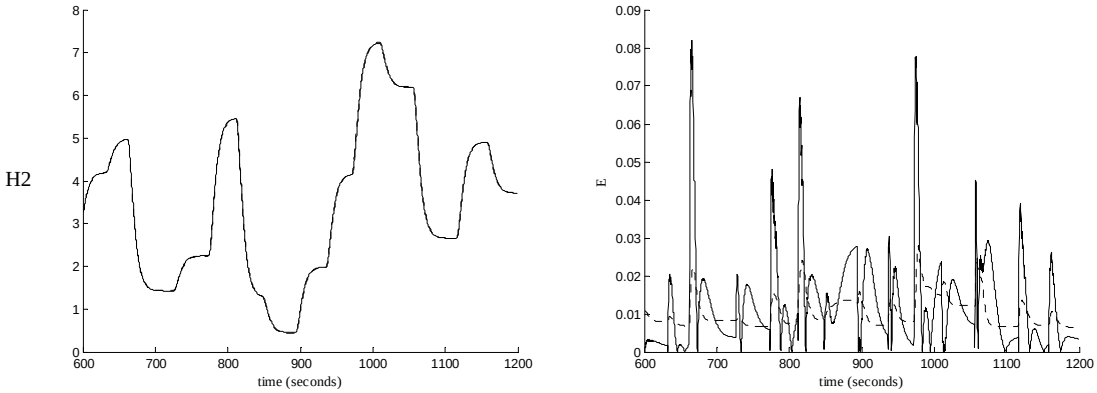


Figure (5.40): Coupled Tanks Simulated System (Random Step Excitation Signal) – Input Voltage (solid line), Height (H1) of water in Tank 1 (dotted line), Height (H2) of water in Tank 2 (dashed line).

Therefore, by ensuring the time between steps is kept within such a margin, we can avoid including large quantities of problematic equilibrium data. Note that it is important to ensure that neither the inputs (H1 and H2(k-1)) or the outputs reach steady-state, so this upper limit on the time between steps should be enforced rigorously, otherwise the conditioning of the covariance matrix can begin to degrade quickly. This problem becomes

most apparent if the optimisation process ends in an abrupt manner, resulting in poorly identified hyperparameters. As before, this data set is then split into training and test datasets. The first 600 seconds of the time history is to be used as training data, and the second 600 seconds used as a test dataset. For this example, H2 is to be identified using H1 and the previous H2 as the model inputs. The squared exponential covariance function is again utilised and the hyperparameters optimised as before. The sampling rates chosen for the training and test data remain the same as in the previous example, resulting in a training dataset of 600 points and a test dataset of 3000 points. Note that the timescale of the whole dataset for this example has been extended from 1000 to 1200 seconds to allow more of these slower transitions to be included. The predictive performance of the model can be seen in Figure (5.41), and the validation measures were calculated as: Mean-Square Error (MSE) of 3.2688e-004, Log Predictive Density (LPD) of -0.0140, and log likelihood (LL) of 1.4375e+003. Overall, the identified model can be seen to provide accurate predictions of the underlying function, with the model error and model variance remaining low with the odd spike coinciding with the larger input transitions.

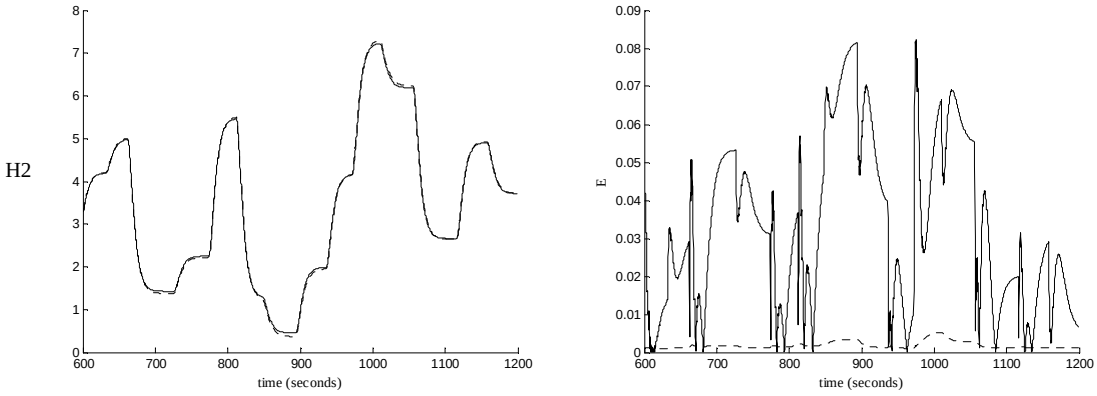


(a) **(b)**

Figure (5.41): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 600 training points from first 600 seconds of Figure (5.40))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

We are now to contrast the performance of this model with the previous model identified using the more 'noisy' random excitation signal shown in Figure (5.36). In order to provide a fair comparison, the training data selected from this dataset was extended to include 600 points from the first 600 seconds. The model performance can be seen in Figure (5.42), and the validation measures where calculated as: Mean-Square Error (MSE) of 0.0016, Log

Predictive Density (LPD) of -870.8319, and log likelihood (LL) of 1.8728e+003. Overall, we can see that the GP model identified from the more 'noisy' random training data provides a good approximation to the underlying system, but does not quite match the MSE performance of the model identified using the more slowly varying step inputs. However, the variance output of the model trained on the noisy data is lower than that of the model trained on the random step data (indicating a more successful optimisation, probably due to better training set conditioning of the noisy data as it includes no steady-state data). Of particular importance is the that accuracy of this 'noisy' data trained model where the output begins to reach steady-state is noticeably poorer than that of the step input trained model, especially at the lower and higher ends of the output operating range. This is to be expected as the 'noisy' training data does not include as much information near to steady-state as the step training dataset. A further by-product of the faster transitions included in the 'noisy' training dataset is that the data has a tendency to concentrate in the middle of the input range leaving the extremities of the operating range more sparsely populated by training observations. It is this tendency that is reflected in the model's poorer predictive performance at transitions that take place in the lower and higher regions of the output operating range.



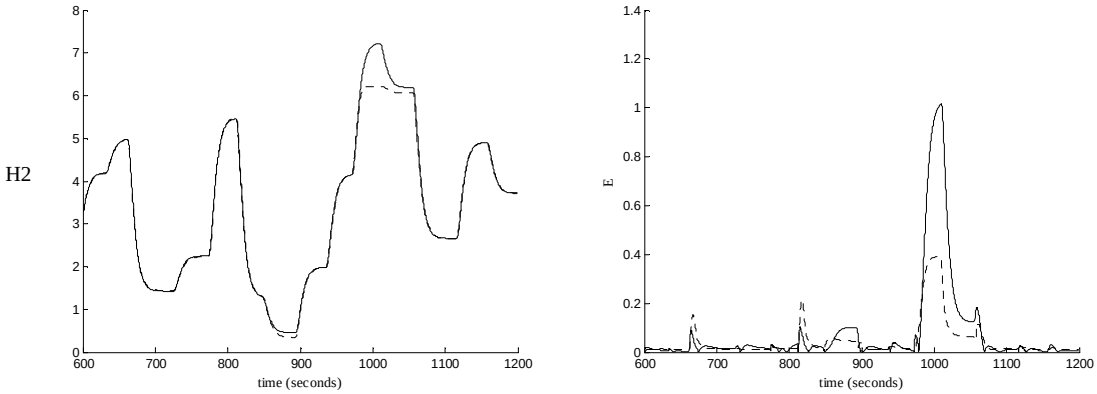
(a) **(b)**

Figure (5.42): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 600 training points from first 600 seconds of Figure (5.36))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

So far in this example application, it is the conditioning of the covariance matrix that has driven the process of identifying a suitable training dataset. However, in the implementation of the GP modelling approach, the size of this training dataset must also be

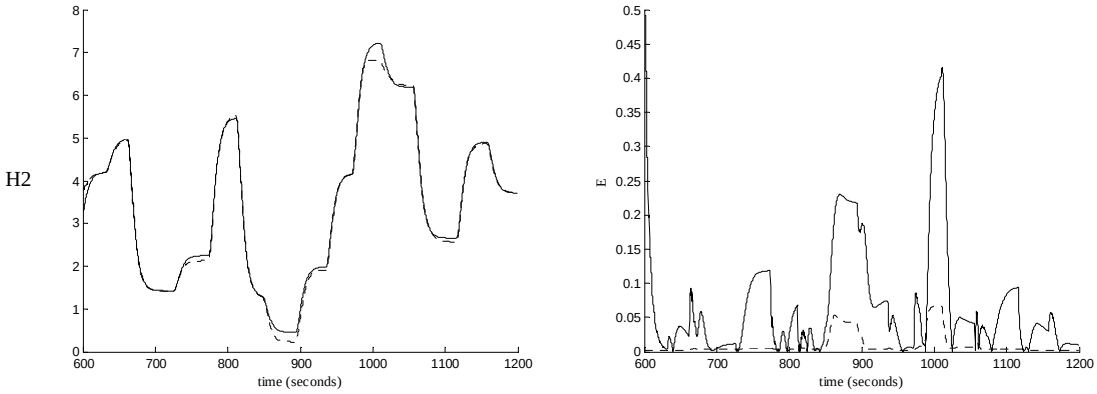
considered carefully, due to the large computational expense of including large quantities of training data. In the examples so far, the training datasets have included 600 points, which could not readily be described as very large. However, it is at around this size that the computation time associated with optimising hyperparameters and then predicting 1000+ test cases becomes slightly prolonged (~15 minutes on my PC: 1.6 GHz, single-core, 1Gb RAM). Therefore, in order to speed this process up it is worth discussing methods to reduce the computational expense of the GP model. If the two existing excitation signals are to be retained, there are two different strategies that can be applied to reducing the size of the training dataset: 1) Reduce the length of the time-scale included, or 2) Decrease the sampling rate used to collect the data. Both approaches are to be investigated for the two random excitation signals shown in Figure (5.36) and Figure (5.40).

Firstly, to reduce the size of the training dataset, the timescale of both excitation signals was reduced from 600 seconds to 300 seconds resulting in a training dataset of 300 points with the existing sampling rate. The model performance of both models is shown in Figure (5.43) and Figure (5.44), and the validation measures of the model in Figure (5.43) where calculated as: Mean-Square Error (MSE) of 0.0443, Log Predictive Density (LPD) of -1.2257, and log likelihood (LL) of 704.7914, and for the model in Figure (5.44) the validation measures where calculated as: Mean-Square Error (MSE) of 0.0119, Log Predictive Density (LPD) of -693.2591, and log likelihood (LL) of 948.7873.



(a) **(b)**

Figure (5.43): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 300 training points from first 300 seconds of Figure (5.40))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).



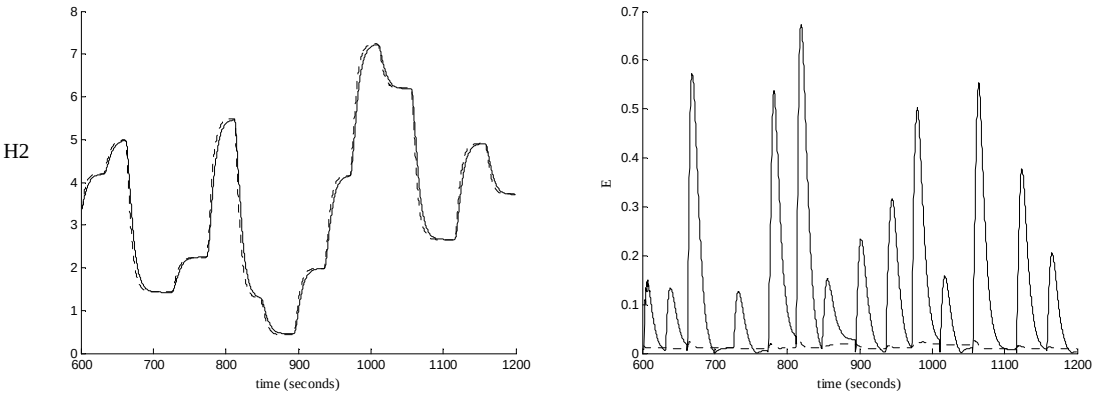
(a) **(b)**

Figure (5.44): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 300 training points from first 300 seconds of Figure (5.36))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, we can see that the two models offer a good approximation to the underlying function over the majority of the operating space. However, the model identified using the random step input data can be seen to have a particular deficiency in predicting at the higher levels of the output H2. This is due to the removal of important training data observations from the training dataset by reducing the included timescale (i.e. transients at this higher region are included in the original data (Figure (5.38) but at t > 300 seconds). The model identified by the 'noisy' random data is slightly less affected by this curtailment of the training dataset at the higher end of the operating range, but is more affected at the lower end of the operating range. This is again evidence of the tendency for data collected from more rapidly varying signals to become concentrated toward the middle of the operating range (i.e. the minimum hold time is not of sufficient length).

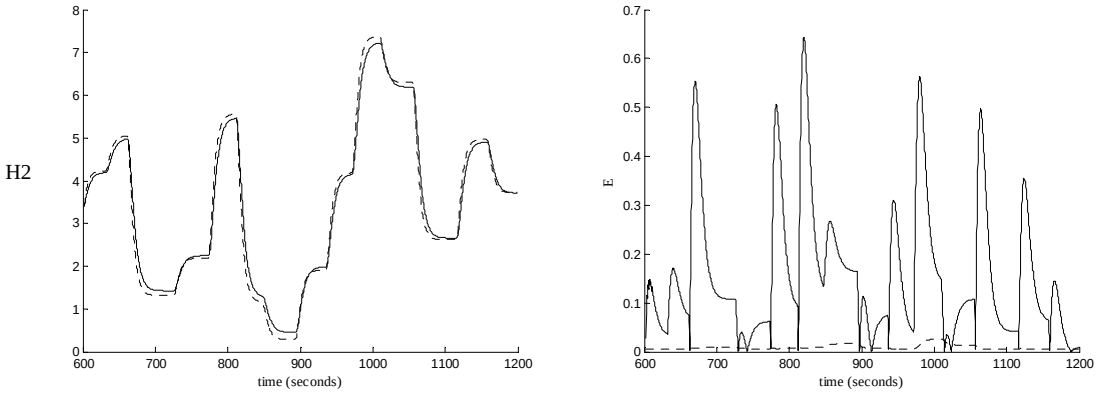
This potential omission of observations from important regions of operating space is a significant problem with employing random excitation signals for the collection of training data. In order to ensure that the full input range of the system is covered by the training data, it is therefore necessary to design excitation signals that are of sufficient duration to make up for this lack of precise control over the input. Furthermore, it is worth reiterating the point that the GP modelling approach can be thought of as a method of interpolation. As a result, any predictions carried out on test points that lie out with the training data are very likely to be inaccurate with a correspondingly large variance output.

Instead of shortening the length of the included excitation signal, an alternative strategy based on reducing the sampling rate (or increasing the interval between datapoints) can be pursued. Unlike the previous example application (Lorenz attractor), as this simulated coupled tank system (and especially the real version) varies more slowly, the potential deterioration in model accuracy when the sampling rate is reduced should be less rapid. Using the same two excitation signals, the sampling rate of the training data was increased by a factor of two (from 1 second to 2 seconds, resulting in 300 datapoints collected from the 600 second duration excitation signals), with the test data sampled as before (0.2 seconds) thus requiring an adjustment in the implementation of the input H2(k-1) from 5 steps previous to 10 steps previous. The model performance of both models is shown in Figure (5.45) and Figure (5.46), and the validation measures of the model in Figure (5.45) where calculated as: Mean-Square Error (MSE) of 0.0367, Log Predictive Density (LPD) of -349.8891, and log likelihood (LL) of 626.7114, and for the model in Figure (5.46) the validation measures where calculated as: Mean-Square Error (MSE) of 0.0447, Log Predictive Density (LPD) of -1.6761e+003, and log likelihood (LL) of 556.7409.



(a) **(b)**

Figure (5.45): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 300 training points from first 600 seconds of Figure (5.40))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).



(a) **(b)**

Figure (5.46): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 300 training points from first 600 seconds of Figure (5.36))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, both of these models can be seen to offer a reasonable approximation to the underlying function, however a significant level of model error is demonstrable across the whole of the test dataset, and particularly at test cases where large transients occur. This is to be expected, as by reducing the sampling rate and therefore the size of the training dataset from 600 to 300 points, the more subtle characteristics of the system response have been lost from the training dataset. This is in contrast to the previous models identified with data from a shortened excitation signal, where the model error is specific to certain regions of the operating range that are not covered by the training data.

5.7.1.3) Small Step Excitation Signal

In the previous implementations of the GP modelling approach, a random excitation signal was employed to collect the training data. Whilst such an approach has been shown to provide suitable training data and therefore good models, a disadvantage of using random signals is that the length of the signal must be of sufficient duration so that training data can be collected from the full operating range of the system. Furthermore, as the sampling rate used to collect the training data cannot be reduced without potentially losing important information, in order to reduce the size of the training set alternative methods to decrease the size of the training dataset should be considered. Assuming the sampling rate of the

training data is kept at a minimum that remains viable to capture the dynamics of the system, it is therefore the length of the excitation signal that could be modified.

Therefore, instead of employing a random excitation signal, a more deterministic approach to the design of the excitation signal and therefore the training dataset could be adopted. Fundamentally, due to the interpolation characteristics of the GP model, the greatest source of model error is normally due to the lack of training data in certain regions of the operating space. Therefore, any excitation signal must attempt cover the entire operating range of the system (i.e. input range is 0 to 10). Taking this prior knowledge and combining it with the other important knowledge over the duration of individual transients, it is therefore possible to design an excitation signal that excites the system across the whole of the operating range whilst remaining shorter in terms of timescale than the previously employed random signals. In essence, the idea is to cram as much information into as small a training dataset as possible. As a result, the computational expense of the GP model may then be reduced without resorting to further complex methods (e.g. sparse matrix methods).

In Figure (5.47) an excitation signal composed of a number of small step transitions that covers the full range of the operating space was developed. As before, it is important to avoid including steady-state data, and the size and number of transitions were chosen so that the input could be stepped up and down through the range within a reasonably short timescale. It is also worth pointing out that although the response of this simulated system should remain symmetrical, in that an input step should result in the same length of transient whether it is positive or negative, the real coupled tank system does not display such perfect symmetry. Therefore, this has influenced the inclusion of ‘downsteps’ as well as ‘upsteps’. Furthermore, including the ‘downsteps’ allows the training dataset to be populated by the full range of data a second time, thus improving model performance.

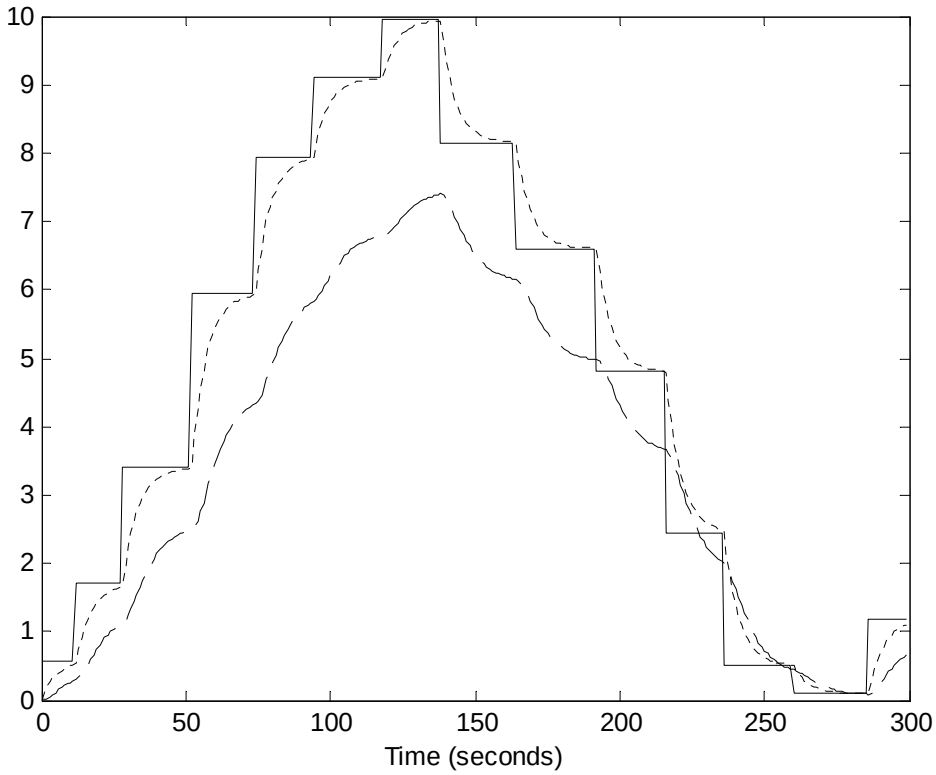
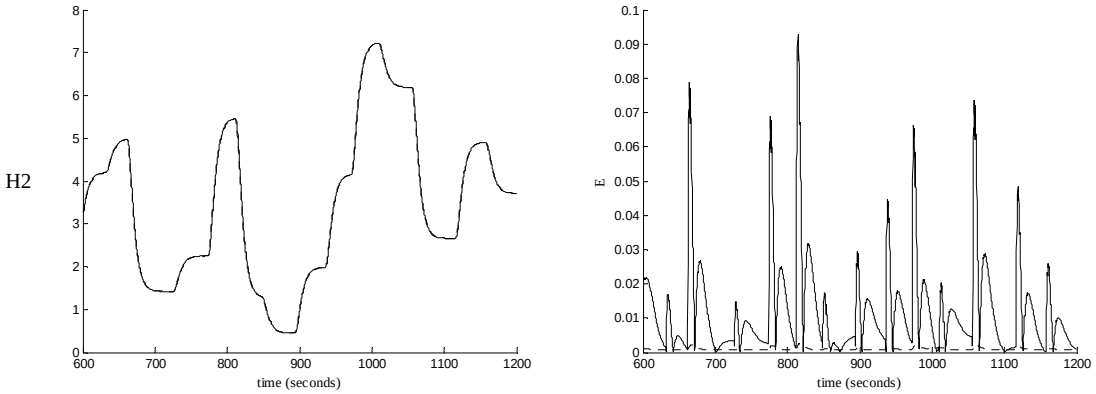


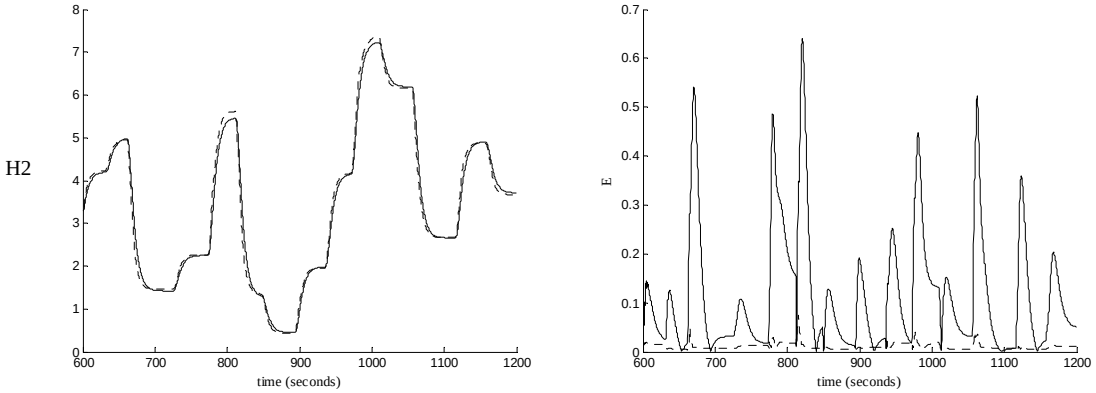
Figure (5.47): Coupled Tanks Simulated System (Small Step Excitation Signal) – Input Voltage (solid line), Height (H1) of water in Tank 1 (dotted line), Height (H2) of water in Tank 2 (dashed line).

Utilising the same sampling rates as before, two training datasets where collected from this excitation data and used to identify two GP models. The first of these models is to use the original sampling rate of 1 second resulting in a training dataset of 300 datapoints, and the second model is to use a 2 second sampling rate thus resulting in a training dataset of 150 datapoints. The performance of these two models can be seen in Figure (5.48) and Figure(5.491), and the validation measures of the model in Figure (5.48) where calculated as: Mean-Square Error (MSE) of 3.6230e-004, Log Predictive Density (LPD) of -455.8357, and log likelihood (LL) of 1.1774e+003, and for the model in Figure (5.49) the validation measures where calculated as: Mean-Square Error (MSE) of 0.0300, Log Predictive Density (LPD) of -465.9639, and log likelihood (LL) of 410.8031.



(a) **(b)**

Figure (5.48): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 300 training points from 300 seconds of Figure (5.47))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).



(a) **(b)**

Figure (5.49): Coupled Tanks Simulated Example (Model: H1, H2(k-1) → H2, 150 training points from 300 seconds of Figure (5.47))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

The performance of the model in Figure (5.48), where 300 training datapoints are included in the training dataset, can be seen to be very competitive with the previous models identified with 600 datapoints. Therefore, as this model performs significantly better than the previous models identified from the same size of training dataset (300 datapoints), this can be seen to be a substantial reduction in the computational expense of the model without much sacrifice in the predictive performance of the model. This is further demonstrated in the model shown in Figure (5.49) where a model identified from 150 datapoints

outperforms in both MSE and LPD the previous models identified from twice the amount of data.

Overall, in this example we have demonstrated that it may be possible to employ prior knowledge of the system in the design of the training dataset used to identify the GP model. Instead of relying on random excitation signals that are routinely used to identify black-box models, by taking more control over the exact nature of the empirical data included, there can be significant reductions in the size of the training dataset required. This can therefore lead to a reduction in the computational burden of the approach. However, it is worth pointing out that the design of the excitation signal and therefore the training set must always take into account the specific characteristics of the application. For the simulated Coupled Tank system adopting an experimental approach where the input is stepped up and down the operating range quickly is a viable method as the system varies in a consistent manner across the whole of the operating range. This may not be the case for other applications. Furthermore, this 'small step' approach has practical benefits when considering the real Coupled Tank system where the transients are very slow and the length of time taken to collect data can become very long. Therefore, conducting an experimental approach that stimulates the system across the whole operating range in a straightforward and systematic manner is an attractive proposition, as moving from one steady-state operating point to another and then back again is a slow process.

5.7.2) Experimental Methods for Coupled Tank System

The Coupled Tanks (CE5) system manufactured by TecQuipment has been equipped with differential pressure sensors (Sensym SX01D) that measure the difference between atmospheric pressure and the pressure at the base of each tank and return an electrical signal. The resultant signals are then amplified to provide a suitable range of measurement of the depth of water in both tanks. Measurements from the Coupled Tank system were automatically recorded through a PC data acquisition device (National Instruments 6024E) controlled with National Instruments LabVIEW software. Through this software the height of water in both tanks (H_1 and H_2), together with the input voltage applied to the water pump, was recorded over the duration of the experiments.

To identify nonlinear systems it is desirable to collect empirical data across as much of the operating range as possible. The system was subjected to a series of small and large step inputs and the response recorded. The system response to large inputs is often difficult to record, as operational restrictions must be enforced (e.g. to avoid water overflow). As a result, the restrictions existing when operating close to the boundaries of the operating envelope are reflected in the smaller quantities of data collected close to the system's limits (Heights near maximum of 0.3m). Additionally, the water pump response to input voltage can be seen to be linear only within a certain range, where the input voltage can be directly transformed to input flow rate through an identified constant. Outside of this range (close to minimum and maximum flow rate), flow rates tend to be related to the pump input voltage in a nonlinear fashion. The Two-Tank system is a stable system that displays high damping, as overshoots and oscillatory behaviour are not observed in its current configuration. The system tends to settle into equilibrium readily, with equilibrium points existing across the operating range. This makes the system suitable for modelling using local linear techniques and small perturbation theory.

The Coupled Tanks system response is relatively slow with time constants typically being several minutes. Consequently, the sampling rate chosen to acquire the data must not be too high that an excessive amount of slow varying data is produced, but also not so low as to inadequately represent the system response. Therefore, a sampling rate of 10 seconds was implemented. For this system, it is worth noting that negative input transients tend to result in faster output transient responses than equivalently sized positive transients.

5.7.3) Analytical Model of the Coupled Tanks System

The conventional method for analysing dynamic systems similar to the two-tank system is to apply the principle of continuity of mass and energy (Bernoulli's theorem). This results in a nonlinear model of the system described by the following equations:

Tank 1 (Height of water H_1, Cross-sectional area A_1, Inflow Q_{vi} and Outflow Q_{v1})

$$A_1 \frac{dH_1}{dt} = Q_{vi} - Q_{v1} \tag{5.23}$$

Tank 2 (Height of water H_2, Cross-sectional area A_2, Inflow Q_{v1} and Outflow Q_{vo})

$$A_2 \frac{dH_2}{dt} = Q_{v1} - Q_{vo} \tag{5.24}$$

The rate of change of volume (cross-sectional area *A,* multiplied by height *H,* for rectangular shaped tanks) of liquid in each tank must be equal to the difference between the flow rate into and out of each tank. If the hole which links the two tanks and the outflow pipe are treated as simple orifices, the flow rate through each hole can be related to the fluid heights through the following equations

$$Q_{v1} = C_{d1} a_1 \sqrt{2g(H_1 - H_2)} \tag{5.25}$$

$$Q_{vo} = C_{d2} a_2 \sqrt{2g(H_2 - H_3)} \tag{5.26}$$

where a_1 and a_2 are the cross sectional areas of the orifice and outlet pipe, H_3 is the height of the outlet pipe above the base of the tanks, g is the gravitational constant, and C_{d1} and C_{d2} are discharge coefficients of constant value. The system can be seen to be nonlinear due to the presence of the square root covering the difference in water levels of the two tanks. From these expressions it is possible to formulate ordinary differential equations that describe the system at different operating or initial conditions. When the system is operating in it's natural operating state of $H_1 > H_2$ with $H_2 > H_3$, the system can be described by

$$f_1(H_1, H_2, Q_o, t) = \frac{dH_1}{dt} = \frac{Q_o}{A} - \frac{C_{d1} a_1}{A} \sqrt{2g(H_1 - H_2)} \tag{5.27}$$

$$f_2(H_1, H_2, Q_o, t) = \frac{dH_2}{dt} = \frac{C_{d1} a_1}{A} \sqrt{2g(H_1 - H_2)} - \frac{C_{d2} a_2}{A} \sqrt{2g(H_1 - H_3)} \tag{5.28}$$

Utilising the same coefficients as in Gong and Murray-Smith (1998), the simulated response of the analytical model was then compared with experimental data collected from the real system when subjected to the same input, see Figure (5.50).

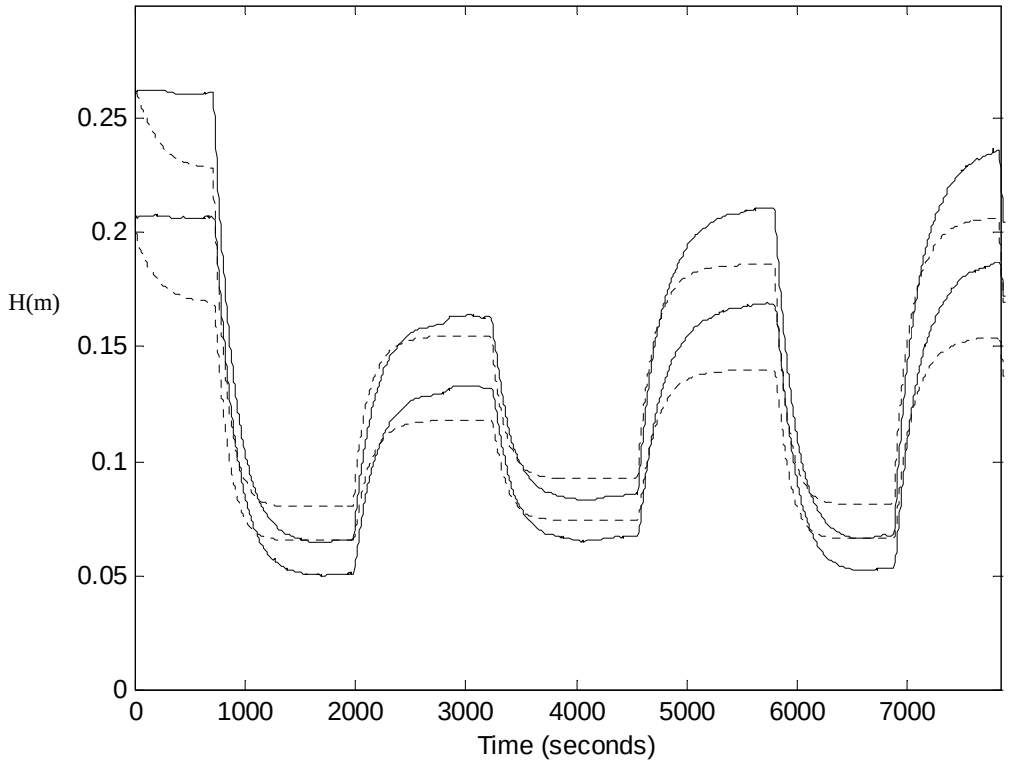


Figure (5.50): Nonlinear Analytical Model (dotted lines) vs. Real Data (sold lines), with the water level in Tank 1 (H1) greater than the water level in Tank 2 (H2).

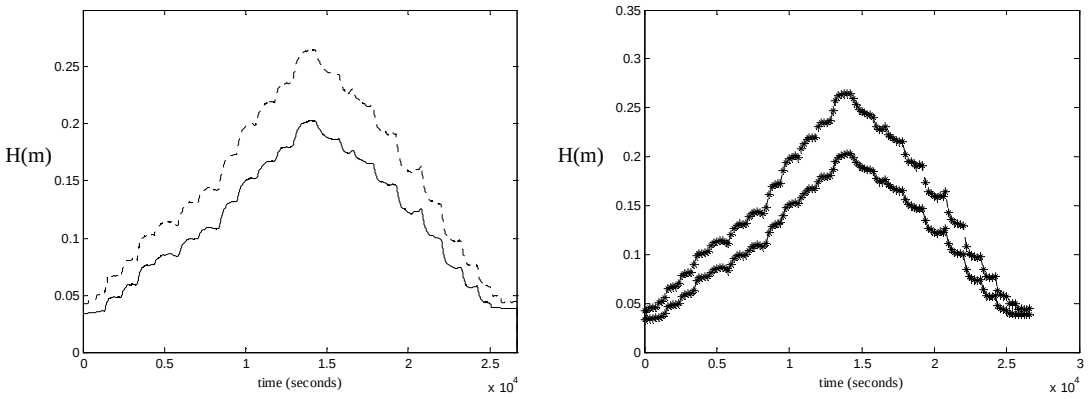
Overall, the analytical model can be seen to approximate the behaviour of the recorded response from the system, but the model has significant errors present. The time constants associated with the transients of the simulated system are slightly faster than in reality, and the steady state values are significantly different. As certain quantities identified in the model (H_1, H_2, A_1, A_2, and Q_{vi}) are well known, they are therefore unlikely to introduce significant error into the model. Instead, the paper by Gong and Murray-Smith (1998) has pointed to a deficiency in the model structure through this assumption for orifice type flow behaviour. As the outlet of the second tank is not strictly an orifice (rather a short section of pipe with a drain tap), other flow properties, such as turbulent or laminar flow, that are not well represented by the model of orifice flow will be present. A possible strategy indicated in Gong and Murray-Smith (1998) for reducing this model error is to introduce a variable discharge coefficient into the model that is dependent on the depth of water in the second tank. Nevertheless, whilst it would be possible to improve the predictive performance of this existing analytical model, this research is focused on the application of the nonparametric GP model alternative.

5.7.4) Application to the Real System

In this section, we are now to tackle the identification of the real Coupled Tanks system using empirical data collected from the system. To collect the training data, the previously discussed experimental approach where the system is excited by a number of small positive and negative step inputs was adopted. In this way, we can attempt to collect empirical data from the whole operating range of the system in a more concise manner than would be achievable if random excitation signals were to be adopted. A further reasoning behind the adoption of the small step excitation approach instead of the random excitation approach is that for many real world applications it is not practically possible to employ large rapidly changing input signals due to operational circumstances and constraints. In many applications, such large and abrupt input transients cannot be performed repeatedly without potentially damaging the system. For the Coupled Tanks system in particular, the use of very large input transients was discouraged for fear of water overflowing from the first tank. Consequently, it is often the case that small perturbation data, collected from the open loop response at various equilibrium points, is used to identify models. Nevertheless, we would want to investigate the performance of the identified GP model for large input transients. This is because of the fact that, for any test data very similar to the training data used, an identification method that fails to find an accurate description could certainly not be recommended. Therefore, training on small step data and testing on larger input responses is a useful method of assessing the flexibility of the identified model and the suitability of the identification method used.

The small step excitation input and output response can be seen in Figure (5.51a) where the data was collected using a sampling rate of 10 seconds leading to a dataset of 2672 datapoints. As discussed previously, in order to avoid considerable computational expense, it is recommended that the size of the training set be reduced to a more manageable level (i.e. below 1000 datapoints). Furthermore, as the system response is smooth and slowly varying, this empirical dataset can be re-sampled without fear of losing valuable information. Therefore, the excitation data shown in Figure (5.51a) was re-sampled by a factor of 20 to provide a training dataset of 134 datapoints with a sample interval of 200 seconds. A further important consideration in the pre-processing of the training dataset is the conditioning of the data itself. Most importantly, the existence of large amounts of steady-state data in the training dataset can lead to ill-conditioning in the covariance matrix

of the GP model. For the excitation data shown in Figure (5.51a), both the input and output signals can be seen to remain sufficiently excited (i.e. the period between the step transitions is relatively small) so as to remove the need to eliminate any steady-state data by hand. Therefore, the only pre-processing employed for this training dataset was the re-sampling of the empirical data. The effect of this pre-processing can be seen in Figure (5.51b).



(a) **(b)**

Figure (5.51) Coupled Tanks System – Small Step Training Data. Chart (a) shows recorded data with input (dotted line) and output (solid line). Chart (b) shows pre-processed Training Data (134 datapoints).

In order to validate the GP models identified from this training dataset, four further test datasets where collected from the Coupled Tanks system. These are shown in Figure (5.52), where the first test dataset is composed of a number of large positive and negative transitions or 'pulses', the second test dataset is composed of a single large positive 'pulse' where the steady-state performance of the model can also be assessed, the third dataset is composed of a number of small positive step transitions designed to test the model on data that is more similar to that present in the training dataset, and the fourth test dataset is a mixture of different size and more rapidly varying transitions. The test datasets shown in charts (b), (c), and (d) of Figure (5.52) where collected using the same sampling interval (every 10 seconds) as that used to collect the empirical data used for training. However, the test dataset shown in chart (a) of Figure (5.52) was collected using a different sampling interval of 4 seconds. This difference is the result of a lack of experimental consistency rather than for any other practical reason. For the purposes of testing the GP models identified, these test datasets are not going to be re-sampled. Therefore, for any previous or regressed inputs/outputs that are to be incorporated into the model structure, an awareness

of the different sampling rates is required. For the test datasets shown in charts (b), (c), and (d) of Figure (5.52), 1-step back of the training data will be equivalent to 20 steps back of the test data, and for chart (a), 1-step back of the training data will be equivalent to 50 steps back of the test data.

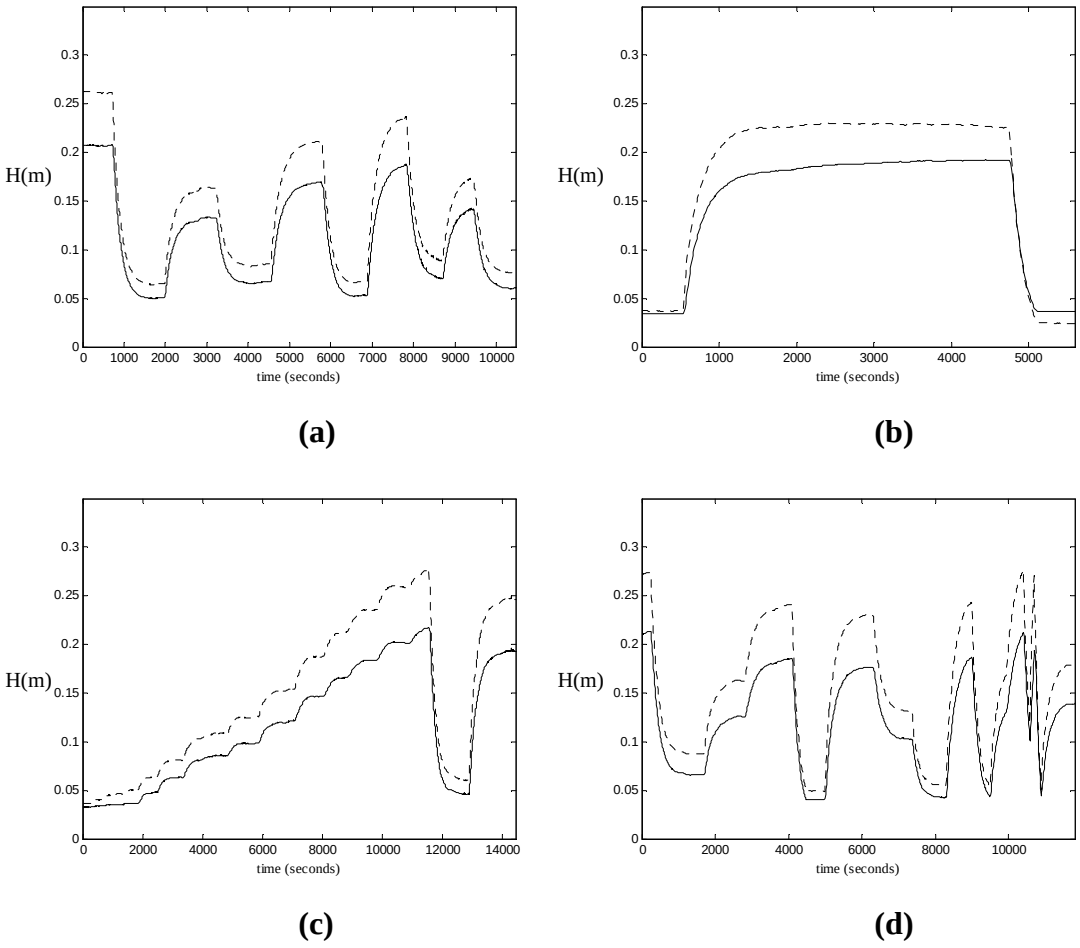


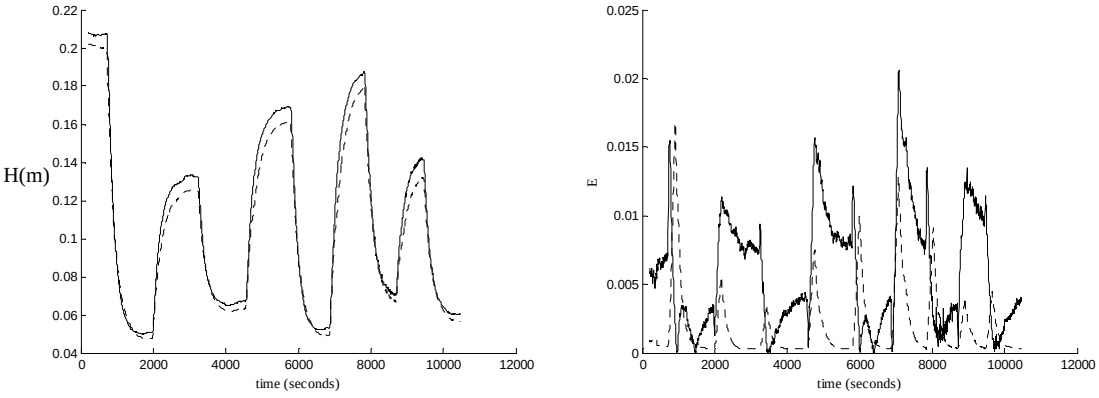
Figure (5.52): Coupled Tanks System – Test Datasets. Chart (a) shows a number of large positive and negative 'pulse' transitions. Chart (b) shows a single large positive step transition with a long period in 'steady-state'. Chart (c) shows a number of small positive steps inputs followed by a large negative transient. Chart (d) shows mixture of slow and fast transitions. All charts show system input (dotted line) and output (solid line).

Before the hyperparameters of the Squared Exponential covariance function are optimised it is first necessary to define the precise model structure that we wish to employ. As in the simulated version, a simple ARX model structure is to be used where the model inputs are to be the current input H1(k) and previous output H2(k-1), with the model output being H2(k). Next, the training target data must be normalised in accordance with the zero-mean

GP prior assumption, and the scaling of the input variables is checked to ensure no large differences are found. Applying the same marginal likelihood maximisation optimisation scheme as before results in θ_{MP}.= (θ_1 = 0.1965, θ_2 = 0.0796, θ_3 = 0.0696, θ_4 = 0.0007). Now that the GP prior has been defined, the predictive mean of the posterior can then be calculated for all test inputs and compared with the real system data. The four test datasets are then to be identified using the GP prior with the results presented below followed by a discussion of the model's performance.

Test 1 – Large 'Pulses' Test

In this test the performance of the GP model identified using the small step training data is tested on the large positive and negative 'pulse' transitions shown in Figure (5.52a). The GP mean predictions can be seen in Figure (5.53) and the validation measures were calculated as Mean-Square Error (MSE) of 5.2375e-005, Log Predictive Density (LPD) of -249.1887, and log likelihood (LL) of 742.1701.



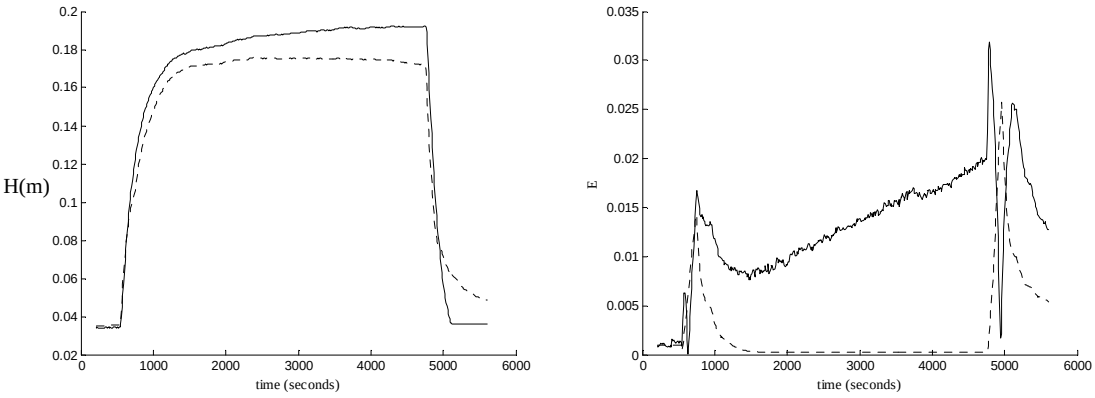
(a) **(b)**

Figure (5.53): Coupled Tanks System – Large 'Pulses' Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Test 2 – Large Step and 'Steady-State' Test

In this test the performance of the GP model identified using the small step training data is tested on the large positive step transition followed by a prolonged period of near steady-state behaviour shown in Figure (5.52b). The GP mean predictions can be seen in Figure (5.54) and the validation measures were calculated as Mean-Square Error (MSE) of

2.0191e-004, Log Predictive Density (LPD) of -3.1304e+003, and log likelihood (LL) of 742.1701.

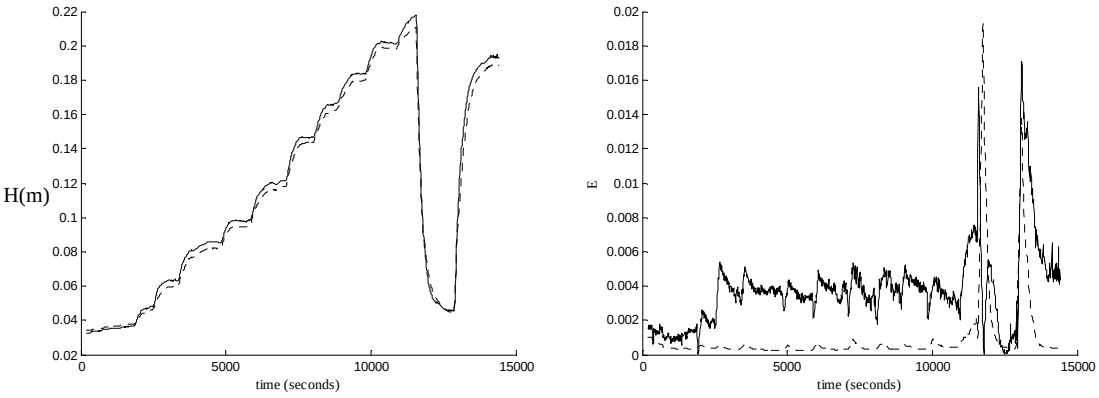


(a) **(b)**

Figure (5.54): Coupled Tanks System – Large Step and 'Steady-State' Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Test 3 – Small Steps Test

In this test the performance of the GP model identified using the small step training data is tested on the small positive step transitions shown in Figure (5.52c). The GP mean predictions can be seen in Figure (5.55) and the validation measures were calculated as Mean-Square Error (MSE) of 1.9965e-005, Log Predictive Density (LPD) of -145.4783, and log likelihood (LL) of 742.1701.

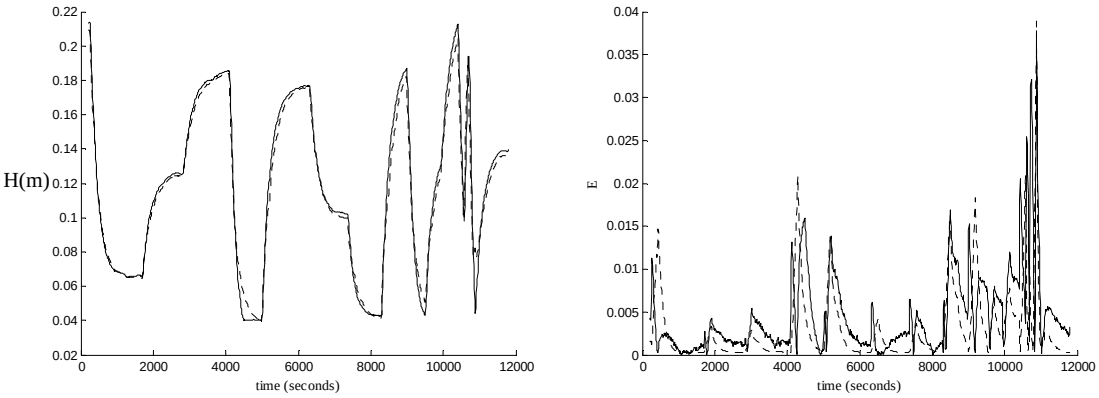


(a) **(b)**

Figure (5.55): Coupled Tanks System – Small Step Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Test 4 – Mixture of Slow and Fast Transitions Test

In this test the performance of the GP model identified using the small step training data is tested on the mixture of slow and fast transitions shown in Figure (5.52d). The GP mean predictions can be seen in Figure (5.56) and the validation measures were calculated as Mean-Square Error (MSE) of 4.3327e-005, Log Predictive Density (LPD) of -45.5146, and log likelihood (LL) of 742.1701.



(a) **(b)**

Figure (5.56): Coupled Tanks System – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, the performance of the GP model in each of the four test cases can be seen to provide a reasonable representation of the system behaviour, with the whole operating range being successfully covered by the training dataset. However, a significant level of error can be seen in the GP mean predictions of Tests 1, 2 and 3 in particular. Firstly, it is important to point out that although the predictive performance of this GP model does not initially appear to be particularly impressive, it is worth remembering that the data employed has been collected from a real system. All the previous examples examined in this thesis have employed simulated data where the datasets used for training and validation are not affected by inconsistencies such as noise, model uncertainties or disturbances. As a result, obtaining a perfect representation of the underlying system from imperfect data is more challenging problem. Nevertheless, it is important to closely analyse the performance of the model to see if any improvements can be made.

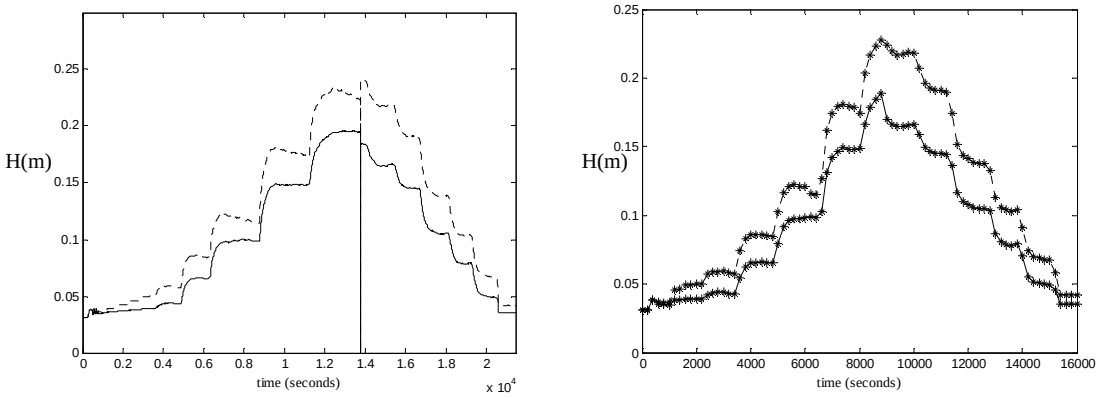
In analysing the performance of the GP model in Test 1 and Test 2, a conclusion that could be drawn is that the training data, which consists of only small step transitions, has not provided enough information with which to allow accurate predictions of larger step transitions to be made. However, in the case of Test 4, which is also mainly made up of large transitions, the GP model can be seen to perform much better. A further troubling result is that the GP model's performance in Test 3 is slightly disappointing given that the test data is pretty similar in nature to that of the training data. Therefore, from an overall perspective the performance of this GP model would appear to be somewhat inconsistent or unreliable. As a result, such a mixed outcome is perhaps more difficult to contend with than a model that is simply poor, and can therefore be disregarded. One aspect that can be clearly understood is that the GP model has failed to represent the very slow positive drift in the level of the output (between the large positive and negative transitions) seen in Test 2. This kind of subtle behaviour is something that the GP modelling approach will fail to capture, as despite this drift in the test output, the level of the test input remains constant. Furthermore, as the training data has been pre-processed to reduce the presence of data near steady state, this more subtle nonlinear behaviour is not included in the training dataset with the resultant growth in model error not being reflected in a growth in the variance output of the GP model. In addition, the variance output from the model in Test 2 is not at all correlated with the model error, and an unfeasibly good LPD measure is the result.

One possible strategy for improving the performance of the model is to include more data in the training set through reducing the sampling interval between observations. Through adopting such an approach, more rapidly varying and subtler nonlinearities can be better represented in the training data. However, for the case of the Coupled Tanks system, as the system response is slow and readily settles into steady-state conditions across the operating range, including more data in the training dataset does not offer any great improvement in the performance of the model. In fact, for the same reasons, reducing the amount of data in the model by a factor of two (giving a training set of 67 points) has also been found to have only a small negative effect on the accuracy of the GP model. Furthermore, due to the smoothly varying nature of the data, the reduction of the training set can also lead to an improvement in the model performance as a greater interval between observations can further reduce the potential inclusion of repeated data, therefore improving the conditioning of the training dataset and leading to better hyperparameters being identified.

Therefore, if the sampling rate chosen for pre-processing and the size of the training dataset are not found to be the deficient, it is perhaps the nature of the training dataset that is the limiting factor in the performance of this GP model. In Tests 1 and 2 the performance of the model was found to be somewhat lacking when attempting to predict large transitions. As the training dataset is composed of only small step transitions it is plausible that if larger transitions are included in the training dataset the accuracy of the model could be improved. Furthermore, such a strategy would seem to be appropriate if a closer analysis of the system characteristics is performed. In particular, the transient response of the system to different size input transitions is not likely to vary in the same consistent manner as that displayed in the simulated version of the system. As the simulated version of the system employed constant transfer function parameters, the amplitude of the output transient response varies in a consistent manner directly dependent on the size of the input transient. For the case of the real Coupled Tank system, we already know that this same consistency is not upheld, as previous analytical models based on constant discharge coefficients have found to be inadequate. In essence, the behaviour of the output response will vary in a less predictable manner than that of the simulated version, where the real response is dependent on the size of the input transient as well as on the current region of operating space. Nevertheless, it is still uncertain whether or not this variation is the major cause of model error, as we have seen in Test 4 that the GP model has performed much better on similarly large transients.

Therefore, to investigate whether or not the small step training data is the problem, an alternative training dataset is to be employed to identify a second GP model, which is to be validated using the same four test datasets. This new training dataset is to contain larger step transitions than in the previous model, whilst still hopefully maintaining coverage of the whole operating range. In order to create this new training dataset, two existing empirical datasets were combined and then pre-processed in accordance with the previous guidelines where the size and conditioning aspects of the data must be considered. Therefore, the overall size of the training dataset was restricted through employing the same sampling rate as before (resulting in a 200 second interval between observations). Furthermore, in order to reduce the problem of ill-conditioning in the covariance matrix, it was necessary to manually remove certain regions of steady state data from the empirical dataset. As discussed in Section (5.5), it also possible to employ a form of regularisation to the data through adding noise to the diagonal of the covariance matrix. However, in this example we are to concentrate on removing problem data as it more informative of the

problem than simply masking it with noise. The effect of this pre-processing on the training data can be seen in Figure (5.57b) where 81 datapoints are included.



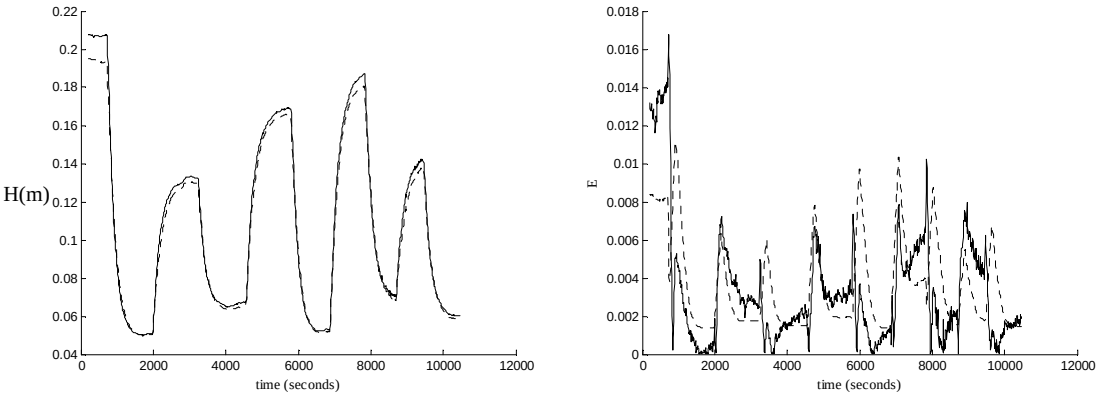
(a) **(b)**

Figure (5.57) Coupled Tanks System – Larger Step Training Data. Chart (a) shows recorded data with input (dotted line) and output (solid line). Chart (b) shows pre-processed Training Data (81 datapoints).

With regard to the process of combining or piecing together different datasets into a more substantial training dataset, this is something that should be done with care and avoided if possible. As the recorded data is a dynamic time-series, it is clear that by simply splicing two datasets together the potential exists to introduce behaviour into the time-series that is uncharacteristic of the system. As a result, if previous inputs and outputs are to be used in the model structure, in the region where data overlaps, this previous information can prove to be noticeably inconsistent with other regions of data. Therefore, as this data set is to be used first to train hyperparameters and then retained in the covariance matrix, it is possible to introduce significant error into the model. Consequently, in order to minimise this risk an effort should be made to combine these different training datasets at points in the time-series where the data is in a similar operating region. In the training dataset shown in Figure (5.58), the two datasets were combined where the output level had reached a similarly high value. Applying the same marginal likelihood maximisation optimisation scheme as before results in θ_{MP}.= ($\theta_1 = 0.7879$, $\theta_2 = 0.2174$, $\theta_3 = 0.0921$, $\theta_4 = 0.0042$). Now that the GP prior has been defined, the predictive mean of the posterior can then be calculated for all test inputs and compared with the real system data. The same four test datasets are now to be identified using the GP prior with the results presented below followed by a discussion of the model's performance.

Test 5 – Large 'Pulses' Test

In this test the performance of the GP model identified using the 'larger' step training data is tested on the large positive and negative 'pulse' transitions shown in Figure (5.52a). The GP mean predictions can be seen in Figure (5.58) and the validation measures were calculated as Mean-Square Error (MSE) of 2.1095e-005, Log Predictive Density (LPD) of 2.6727, and log likelihood (LL) of 311.9477.

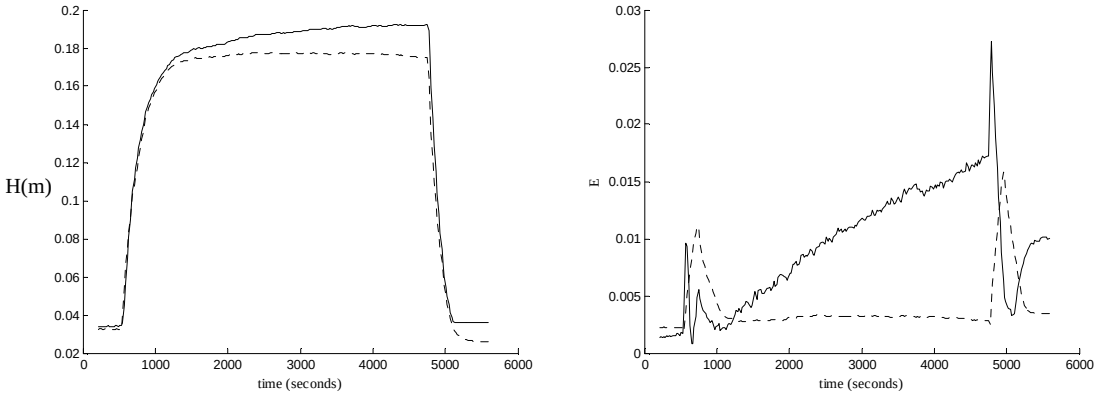


(a) **(b)**

Figure (5.58): Coupled Tanks System (larger training steps) – Large 'Pulses' Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Test 6 – Large Step and 'Steady-State' Test

In this test the performance of the GP model identified using the 'larger' step training data is tested on the large positive step transition followed by a prolonged period of near steady-state behaviour shown in Figure (5.52b). The GP mean predictions can be seen in Figure (5.59) and the validation measures were calculated as Mean-Square Error (MSE) of 1.1253e-004, Log Predictive Density (LPD) of -15.8432, and log likelihood (LL) of 311.9477.

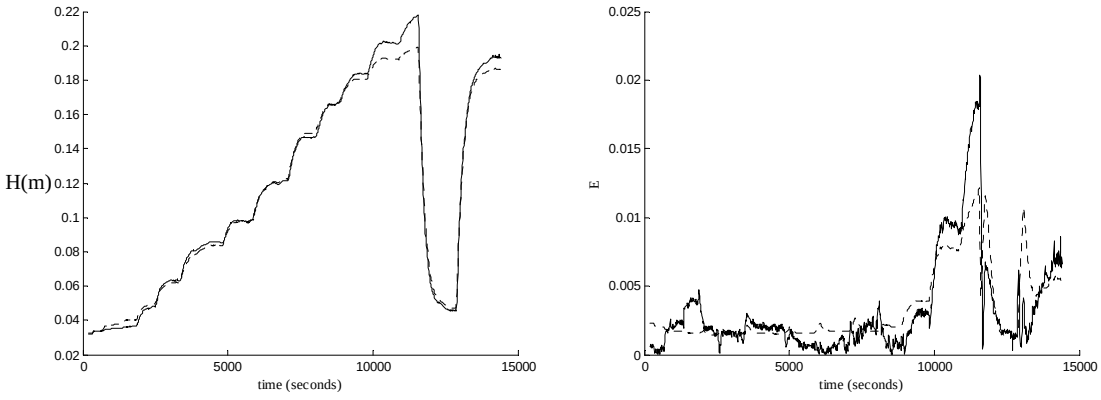


(a) **(b)**

Figure (5.59): Coupled Tanks System (larger training steps) – Large Step and 'Steady-State' Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Test 7 – Small Steps Test

In this test the performance of the GP model identified using the 'larger' step training data is tested on the small positive step transitions shown in Figure (5.52c). The GP mean predictions can be seen in Figure (5.60) and the validation measures were calculated as Mean-Square Error (MSE) of 2.3790e-005, Log Predictive Density (LPD) of 3.2935, and log likelihood (LL) of 311.9477.

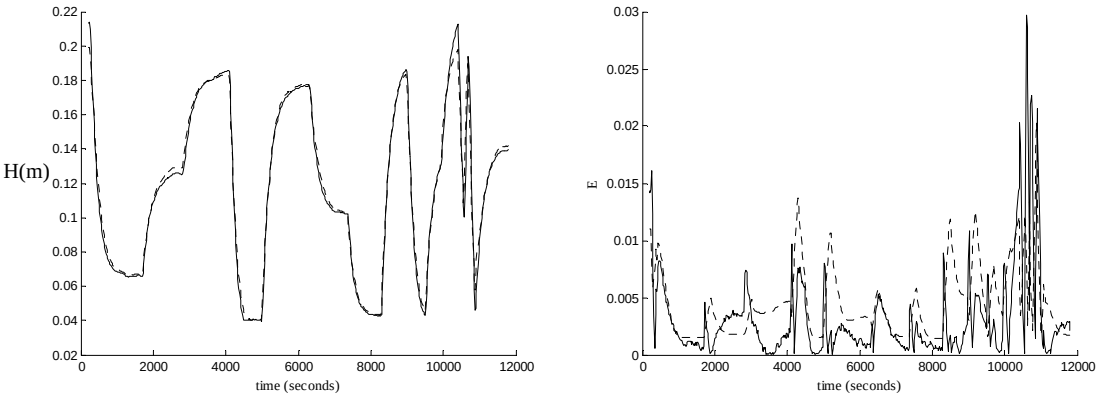


(a) **(b)**

Figure (5.60): Coupled Tanks System (larger training steps) – Small Step Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Test 8 – Mixture of Slow and Fast Transitions Test

In this test the performance of the GP model identified using the 'larger' step training data is tested on the mixture of slow and fast transitions shown in Figure (5.52d). The GP mean predictions can be seen in Figure (5.61) and the validation measures were calculated as Mean-Square Error (MSE) of 2.5033e-005, Log Predictive Density (LPD) of -2.3593, and log likelihood (LL) of 311.9477.



(a) **(b)**

Figure (5.61): Coupled Tanks System (larger training steps) – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Inspecting the GP mean predictions of the 4 tests, we can see a significant overall improvement in the accuracy of the GP mean predictions across the majority of the operating range. However, in both Test 5 and Test 7 a significant level of error is present in model at high output levels. This is due to the excitation data included in the training dataset not quite reaching the same high output levels, therefore resulting in a loss of model accuracy and growth in the corresponding variance output. In Test 6, the new GP model still does not provide a better representation of the low level positive drift discussed earlier, but as this new training dataset is not radically different to the previous training dataset this problem is not likely to be solved. In Test 8, the new GP model does not offer a significant improvement over the previous model, with both models providing a decent level of approximation of the slower transitions, and a slightly weaker performance on the faster transitions. Such an outcome is to be expected as the training data is composed of slower transitions and is sampled in such a manner to capture these slower dynamics in relatively few points.

In order to ease comparison of this new model with that of the previous model trained on the original training dataset the validation measures are tabulated below (remember that test data used for Test 1 is the same as that in Test 5, Test 2 is the same as Test 6, etc.).

Original Training Data	MSE	LPD	LL
Test 1	5.2375e-005	-249.1887	742.1701
Test 2	2.0191e-004	-3.1304e+003	742.1701
Test 3	1.9965e-005	-145.4783	742.1701
Test 4	4.3327e-005	-45.5146	742.1701

Table (5.2): Validation measures from Tests 1 to 4.

New Training Data	MSE	LPD	LL
Test 5	2.1095e-005	2.6727	311.9477
Test 6	1.1253e-004	-15.8432	311.9477
Test 7	2.3790e-005	3.2935	311.9477
Test 8	2.5033e-005	-2.3593	311.9477

Table (5.3): Validation measures from Tests 5 to 8.

Overall, the use of this second training dataset (where larger step transitions are employed) has improved the MSE accuracy of the model in three of the four test cases. In Test 7, the MSE performance of the second GP model is worse than that of the original model due to the fact that the second training set fails to fully cover the whole operating range. Therefore, the suspicion that the original small step training dataset does not excite the system enough to gather information with which to predict large transitions accurately would appear to be true. Also worth noting is the fact that the LPD performance of the second model is worse than that of the original model, despite the model error being mostly improved. The reason for this apparent inconsistency is that the original training dataset was better conditioned (less steady-state data), therefore allowing a more successful optimisation stage to be completed. This is also indicated by the fact that the Log Likelihood (LL) measure being higher for the first GP model.

As discussed previously, the use of such small step training data has imposed an assumption that the scale of variation or flow rates remains consistent for large and small step inputs. Furthermore, such a deficiency would appear to be similar to that built into the analytical model where the discharge coefficients were defined as constants. In addition, this lack of consistency in the system response over the operating range can be understood

as an example of non-stationary behaviour, and of course the GP model has been defined using a stationary covariance function. Nevertheless, it is likely that the GP model could be improved through the use of a longer excitation signal composed of both larger and smaller steps, such as a randomly generated input signal. However, the key objective of using the more deterministic small-step excitation approach was to ensure that the whole of the input range would be covered in a relatively short time-scale. Therefore, any random excitation signal must also be designed to be sufficiently long in order to ensure that the whole of the input range is covered. Furthermore, due to the slow time-constants associated with the system, the sampling rate used to collect the data maybe large enough to avoid including large quantities of data in the training dataset. In addition, further methods of reducing the computational demand could also be adopted such as including derivative observations or sparse matrix methods. A potential problem of adopting a more random excitation approach is the time that may be necessary to collect the data. For example, the small step training dataset shown in Figure (5.51) took ~7 hours to collect, and in order to ensure that the full operating range of the system is covered a similar or perhaps even longer period may be necessary. Whilst this is certainly not an impossible task, and something that I believe should have been pursued, unfortunately it has not been possible to undertake further experimental work. In many ways this is reflective of some of the more practical limitations that are routinely encountered in the identification of real systems, where large quantities of suitable empirical data are not always available.

A further practical aspect that has not yet been discussed is the reliability or consistency of the empirical data. Whilst the inclusion of larger step transitions in the training data has proven to improve the predictive accuracy of the GP model, another possible reason why the first GP model identified using the small step training data does not perform as well as expected is due to subtle variations in the system's properties. It is difficult to substantiate this proposal, but both the small step training data shown in Figure (5.51) and the test data shown in Figure (5.52d) where both recorded ~2-3 months after the other datasets used for training and testing. In addition to the presence of intangible inconsistencies such as noise, it is possible that the properties of the system can have changed by a small amount. This is especially plausible as the measurement devices employed in the experimental apparatus were recalibrated on more than one occasion. It is this potential drift in the measurements that is perhaps the main reason why the GP model trained on the small step training data performs much better on the test data collected at around the same time (in Test 4) than in

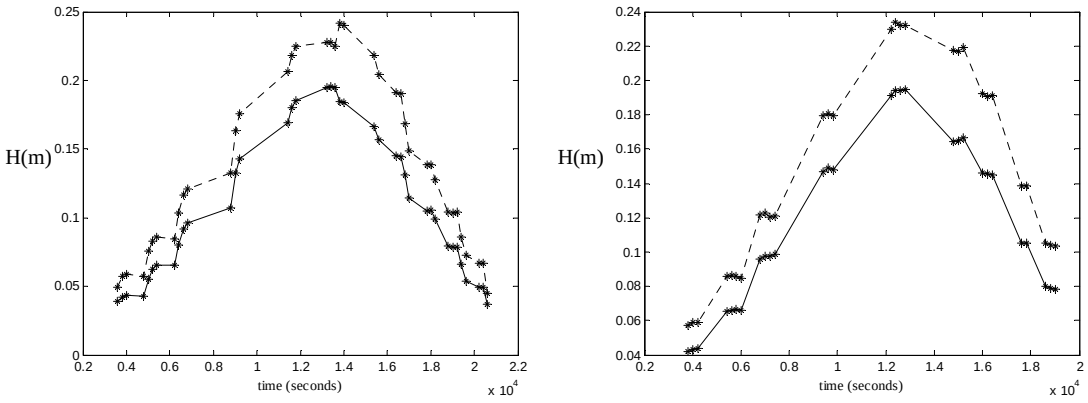
the other test cases. Overall, as with other empirically based modelling approaches, the GP modelling approach will be especially vulnerable to changes in the properties of the system. As the GP model can be interpreted as a precise mapping between input and output, any changes in the system that have not been reflected in the training data are not going to be predicted with any great accuracy by the model.

5.7.5) Incorporating Derivative Observations

In Section (5.4) the use of derivative observations was discussed as potentially useful extension to the GP modelling approach. Through this extension empirical data points (function observations) that are close to equilibrium or have reached steady-state can be summarised into derivative observations. As a result, the computational demand associated with inverting the covariance matrix (whose size is dictated by the number of function observations included in the training dataset) may be reduced. Incorporating derivative observations is often described as being in keeping with the divide-and-conquer strategy prevalent in many multiple model approaches, as the identification of derivative observations at equilibrium operating points is somewhat equivalent to identifying local linear models. In this section the process of identifying derivative observations and then incorporating them into the GP modelling approach using data collected from the Coupled Tank system is to be demonstrated. As the second training dataset composed of slightly larger input transitions (see Figure (5.57)) was found to provide a slightly better model accuracy, it is this training dataset that is now to be used to identify a number of derivative observations. Furthermore, the previously adopted model structure of including the previous output as a second model input is to be retained.

In the previous GP model of the Coupled Tanks system, the training data was sampled using a sampling interval of 200 seconds resulting in a dataset of 81 datapoints. Furthermore, before this final re-sampling was performed, the majority of the steady-state data was removed from the training dataset in order to improve the conditioning of the subsequent covariance matrix. In this example, instead of simply discarding this steady state empirical data it is to be used to identify derivative observations at these equilibrium operating points using the linearisation methods described in Section (5.4.1). Turning our attention to the function observations that are to be included, as the steady-state parts of the excitation data are to be used to form the derivative observations, we can therefore focus on

including only off-equilibrium or transient data as function observations. In the previous GP model, the manual pre-processing performed was aimed at minimising the inclusion of steady-state data, whilst still retaining enough information so that the transition between the input transient and steady state behaviour could be captured. In this implementation, an effort has been made to be more ruthless in removing steady-state data, leaving only transient behaviour included as function observations. This division of the empirical data into separate datasets that are to be used either as function or derivative observations can be seen in Figure (5.62). The result of this process is a training dataset consisting of 39 function observations and 9 derivative observations.

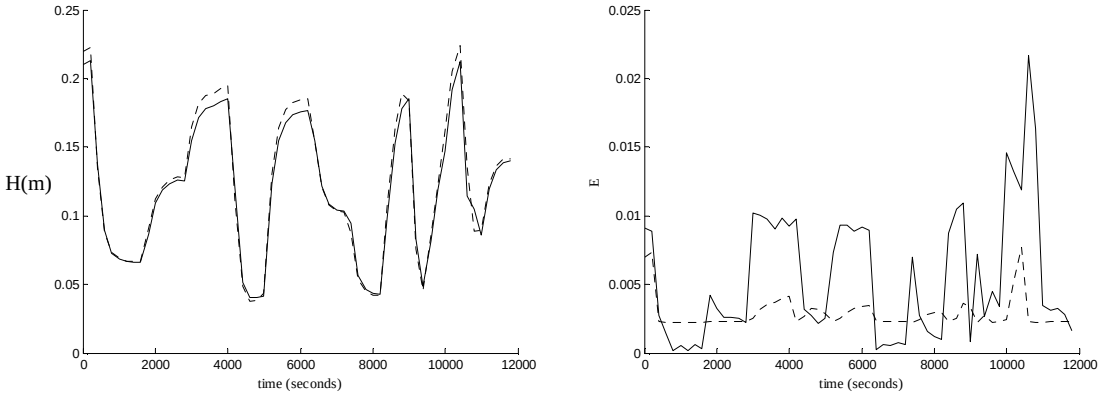


(a) **(b)**

Figure (5.62) Coupled Tanks System – Larger Step Training Data. Chart (a) shows recorded data used for function observation (39 datapoints) with input (dotted line) and output (solid line). Chart (b) shows recorded data used to identify 9 derivative observation with input (dotted line) and output (solid line).

In order to validate the performance of this model the same test datasets (shown in Figure (5.52)) that were used previously can again be employed. Furthermore, in order to simplify the model implementation, these test datasets are to be re-sampled so that the sample interval of 200 seconds is consistent for both test and training datasets. As a result, the delayed or previous output H2(k-1) that is to be used as an additional model input will be indexed one-step back in both the test and training datasets. After performing the same normalisation and scaling checks as before, the hyperparameters of this new GP model can then be identified and model predictions computed. Note that the optimisation and predictive procedures are largely unaffected by including derivative observations when uncertainty propagation is not also performed, see Section (5.4.2). If we now apply this new GP model (consisting of 39 function observations and 9 derivative observations) to the test

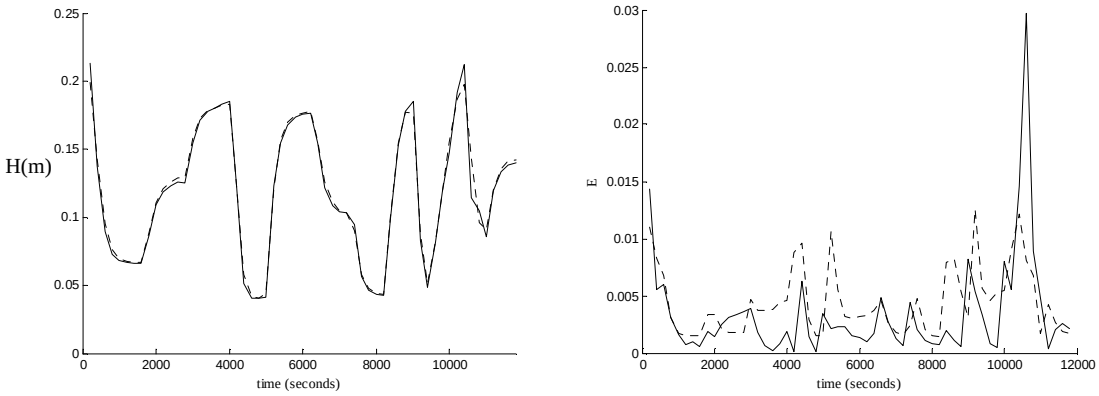
dataset shown in Figure (5.52d), where a mixture of slow and fast transitions are present, the GP mean predictions can be seen in Figure (5.63) with the validation measures calculated as Mean-Square Error (MSE) of 5.2375e-005 and Log Predictive Density (LPD) of -7.8533, and log likelihood (LL) of 133.5611.



(a) **(b)**

Figure (5.63): Coupled Tanks System (39 function observations, 9 derivative observations) – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

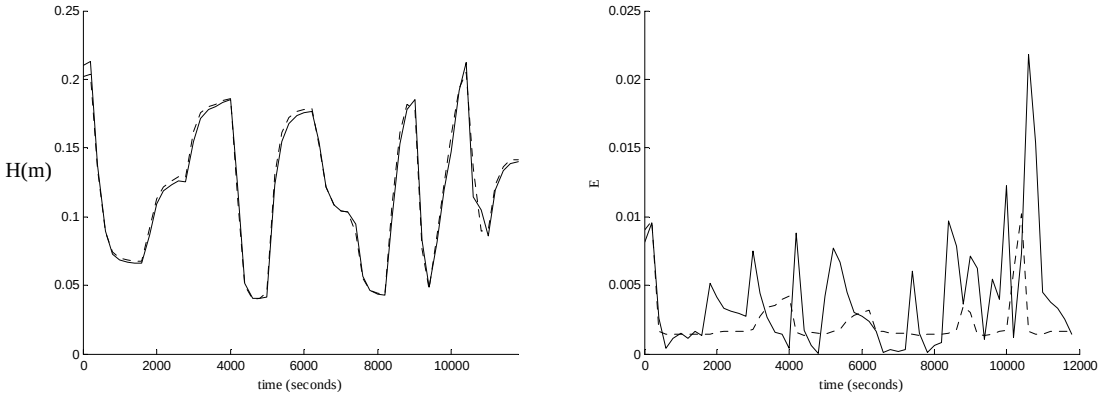
Overall, this new implementation of the GP model (39 function observations, 9 derivative observations) has provided a reasonable approximation to the test dataset. The model does seem to perform less well at higher levels of the output, and it is also worth pointing out that as the test data has been re-sampled to match the sampling interval of the training data, the faster transitions included in this test dataset have been affected. Nevertheless, this same test dataset is now going to be used to validate the performance of a number of different model implementations so that the effect of incorporating these derivative observations can be analysed. Firstly, in order to provide a comparison for this new GP model (39 function observations, 9 derivative observations), the performance of the previous GP model (81 function observations, 0 derivative observations) on the same test data is now to be plotted, see Figure (5.64). The validation measures were calculated as Mean-Square Error (MSE) of 3.2537e-005 and Log Predictive Density (LPD) of 2.9138, and log likelihood (LL) of 311.9477. Overall, the MSE performance of this existing GP model is superior as would be expected from including twice as many datapoints (function observations) in the training dataset. However, the LPD performance of the model including derivative observations is slightly superior.



(a) **(b)**

Figure (5.64): Coupled Tanks System (81 function observations, 0 derivative observations) – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

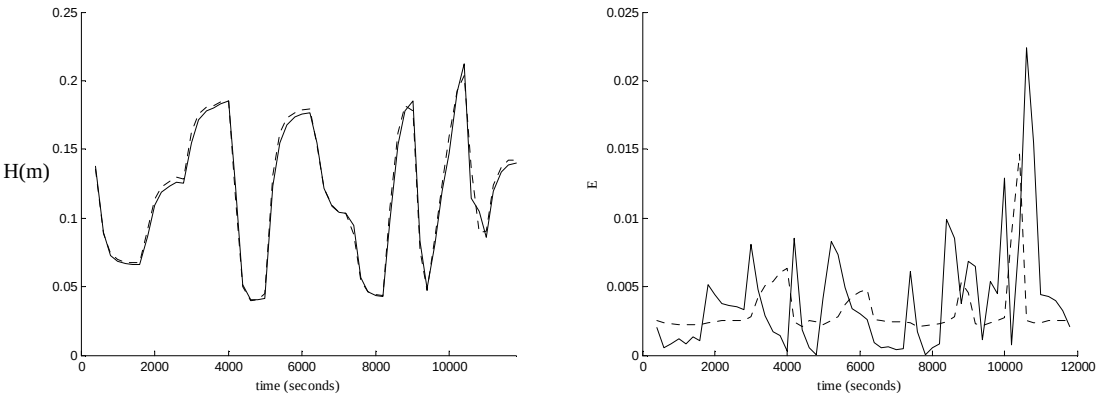
If we now include the identified derivative observations with this original training dataset of 81 function observations, we can investigate whether adding such derivative information improves the accuracy of the model. This performance of this implementation of the model can be seen in Figure (5.65), and the validation measures were calculated as Mean-Square Error (MSE) of 3.1775e-005 and Log Predictive Density (LPD) of -15.0374, and log likelihood (LL) of 311.9477.



(a) **(b)**

Figure (5.65): Coupled Tanks System (81 function observations, 9 derivative observations) – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, the effect of incorporating the derivative observations has been minimal with only a small increase in the accuracy of the description as compared to the previous case where no derivative observations. However, it is worth stating that the proposal for including derivative observations is not expressly aimed at improving the accuracy of models that have already been found to offer good predictions. Instead, the use of derivative observations is aimed at reducing the computational demand of the GP modelling approach through allowing empirical data near to equilibrium operating points to be summarised. Therefore, in order to provide some comparison to the previous model shown in Figure (5.63), where 39 function observations and 9 derivatives were employed, the existing training dataset pre-processed to include 81 training datapoints is to be re-sampled by a factor of two, providing a training dataset of 41 function observations (with a 400 second sample interval). GP model predictions using this training dataset (41 function observations, 0 derivative observations) were then computed and plotted in Figure (5.66), and the validation measures were calculated as Mean-Square Error (MSE) of 3.2790e-005 and Log Predictive Density (LPD) of -3.7751, and log likelihood (LL) of 147.5693.

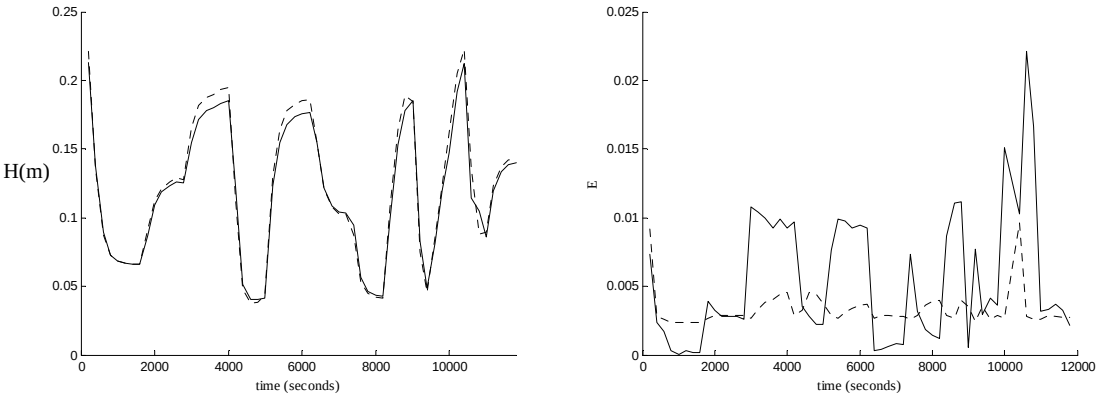


(a) **(b)**

Figure (5.66): Coupled Tanks System (41 function observations, 0 derivative observations) – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

From the result shown in Figure (5.66) we can see that the performance of the GP model defined using the alternative training dataset (39 function observations, 9 derivative observations) shown in Figure (5.63) is perhaps not as great as first thought. As both training sets contain a similar amount of function observations, it can be deduced that the slightly more aggressive data selection process adopted in the pre-processing shown in

Figure (5.62) has not provided the model with same level of information with which to make accurate predictions (especially at higher levels of the output). This can be confirmed by implementing a GP model that uses only the 39 function observations without the accompanying derivative observations, see Figure (5.67), where the validation measures were calculated as Mean-Square Error (MSE) of 5.3451e-005 and Log Predictive Density (LPD) of -5.4487, and log likelihood (LL) of 133.5611. The result shown in Figure (5.67) is very similar to that shown in Figure (5.63), and confirms the previous case that the inclusion of derivative observations only has a small positive effect on the accuracy of the resultant model.



(a) **(b)**

Figure (5.67): Coupled Tanks System (39 function observations, 0 derivative observations) – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, in this application the incorporation of derivative observations has not been found to offer any large benefit in terms of improving the model accuracy, or allowed the number of functional observations to be reduced greatly. In terms of improving the model accuracy, this can perhaps be expected, as defining a linearisation at an equilibrium point is not going to provide any great deal of information with which to make predictions away from each equilibrium point. More importantly, the lack of benefit shown in a computational sense must also be appreciated in the context of the example system. As the Coupled Tanks system exhibits a slow and smoothly varying response, it is therefore possible to capture the dynamics of this response with relatively few datapoints. In other cases the system response may be more complex or less smooth, thus requiring a larger number of functional observations to be included with which to capture the dynamics. In such as case, the

potential benefits of summarising any data that is close to equilibrium in a small number of derivative observations, so that the computational expense may be better spent on including more functional observations, may become more apparent. As mentioned in Section (5.4), the abundance of data close to equilibrium is a feature of many engineering applications. Nevertheless, for the practical applications investigated in this thesis, the benefits of applying this extension to the GP modelling approach are not massive. This is due to the fact that the great majority of data close to equilibrium has instead been eliminated from the training dataset in order to improve the conditioning of the covariance matrix. Therefore, incorporating derivative observations may be seen as an alternative to simply removing equilibrium data, and something that does offer a slight increase in model accuracy.

It is also worth pointing out that in this implementation the number of datapoints used to identify the derivative observations was quite low (e.g. 4 or 5 points). Therefore, it is likely that if a larger number of datapoints were included in each linearization a more accurate derivative observation could be obtained. Furthermore, as each derivative observation can be interpreted as a local linear model, a further improvement to the overall description could be obtained if the performance of each local model was validated individually. Therefore, if the inclusion of derivative observations in the GP model is thought to be important to gaining a good overall description, it is perhaps wise to retain access to a significant portion of steady-state data. In the implementation presented here, the design of the training dataset and the pre-processing employed has acted to remove most of the steady-state data, significantly reducing the potential for accurate local linear models to be identified.

5.7.6) Mixed Model Implementation

So far in this example application, the objective of the modelling process has been to identify a model of the relationship between the input H1 and the output H2. As both measurements can be seen to vary along a similar timescale, matrix conditioning problems that result from the inclusion of steady-state data can be mostly avoided, as discussed in Section (5.7.1). However, whilst in this application it has been possible to select input and output variables that vary smoothly in tandem with one another; this is certainly not going to be the case for all applications. As a result, in the current implementation of the GP modelling approach we would seem to be restricted to problems where the input and output

data are constantly varying in a smooth manner. As discussed earlier, this restriction has been imposed through the selection of the most popular Squared Exponential covariance function.

Therefore, in order to tackle problems where the data do not vary along a similar timescales one possible strategy is to employ more sophisticated non-stationary covariance functions. In Section (4.3.2) a number of non-stationary covariance functions were outlined, and in Section (4.3.3) methods of combining covariance functions were also discussed. However, the use of more complex covariance functions has not been the subject of much practical investigation. Therefore, instead of changing the covariance function, an alternative strategy would be to change the nature of the data in order to make the situation more compatible with the existing methodology that employs the squared exponential covariance function. This proposal for changing the nature of the data has also already discussed in Section (4.3.3.4) where it was referred to as 'Nonlinear Mapping' or 'Warping'. However, this concept can also be interpreted as a 'mixed' or 'hybrid' model implementation where the identification problem is divided into two parts, where the input space data is first mapped or 'warped' on to a intermediate or latent function space using some sort of initial model, followed by the application of a GP model that is to model the relationship or residuals between this new intermediate value and the desired output.

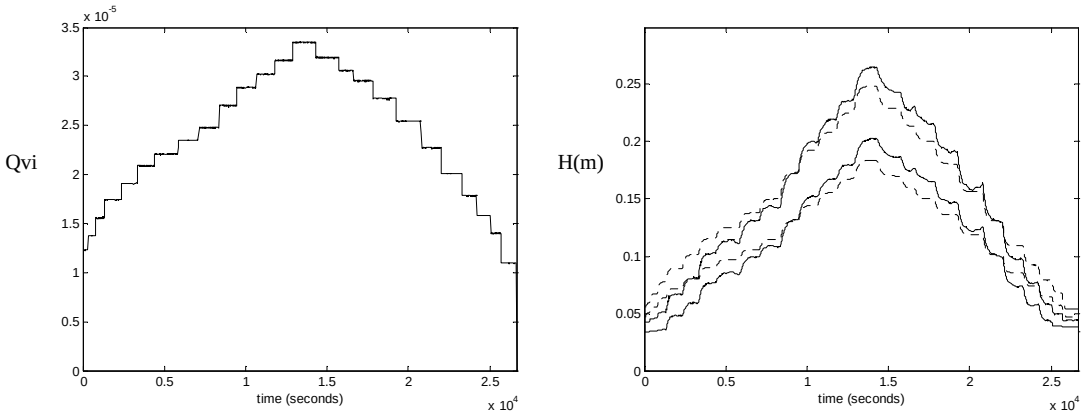
Utilising a 'mixed' or 'hybrid' model implementation that is composed of more than one type of model for the purposes of system identification is certainly not without precedent, and can be interpreted as a further form of model complexity optimisation as the identification task is divided into more manageable components. A general discussion of the various possible combinations of different models can be found in Nelles (2001), and particularly well known cases of mixed models are the Hammerstein and Wiener model structures which utilise linear dynamic and nonlinear static components in combination. Mixed model implementations can often involve the combination of analytical models developed from first principles with models developed from empirical data. As a result, such a mixed model may have an advantage over models developed solely from data in that they make use of any existing model that is available. This means that any prior knowledge and interpretability associated with the existing model can be retained, rather than disregarded in the creation of a completely new description. Further advantages of adopting a mixed model approach are that the extrapolation and robustness properties of the model

can be improved, as a greater level of prior knowledge is retained the model it is less likely to be dumbfounded by missing or suspect data.

In this example, we are to demonstrate the potential use of the GP model in such a mixed model implementation, and also attempt to identify a model of the Coupled Tanks system that describes the relationship between the input voltage and the height of water in the second tank (H2). Therefore, the recorded height of water in the first tank (H1) is not going to be employed by this description. As a result, we must first find some method of modelling (or mapping) the recorded input voltage data onto an intermediate or latent input space, where this data is found to vary on a similar timescale to that of the recorded H2 output data. After this initial model or mapping is performed, the GP modelling approach is then to model the residual to provide a prediction of the system output. For the initial model or nonlinear map, the analytical model of the Coupled Tanks system outlined in Section (5.7.3) would seem to be useful. Although this model was not found to be particularly accurate, it does offer a reasonable overall depiction of the system response and as it has been developed from first principles, such a model contains a level of interpretability that is desirable.

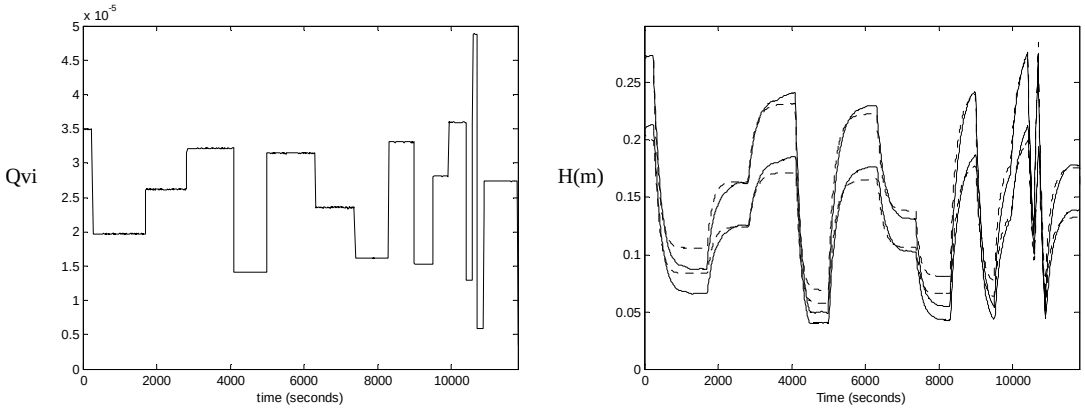
In order to identify the GP model we must first create a suitable training dataset in order to identify hyperparameters. In this example, we are to make use of the previous small step training dataset, and then test the model on one of the 4 previously used test datasets. In order to make use of the analytical model, the input voltage recorded must first be converted into an input flow rate Q_{vi} through multiplying the voltage by the constant 7.6e-6. This input flow rate of the training dataset can be seen in Figure (5.68a), and the recorded H1 and H2 values shown as solid lines in Figure (5.68b). The input flow rate signal shown in Figure (5.68a) was then fed through the analytical model using the ‘ode45’ Matlab solver in order to simulate predictions of H1 and H2. These simulated values of H1 and H2 are shown as dotted lines in Figure (5.68b) and as expected can be seen to differ from the real recorded values by a significant amount. This whole process was then repeated for the test dataset shown in Figure (5.69). Therefore, through applying the input flow rate signal to the analytical model, we have identified values of H1 and H2 that can now be thought of as a nonlinear mapping of the original input space. Furthermore, as both of these simulated values can be seen to vary along a similar timescale to that of the recorded H2 output, either quantity could now be employed in the GP model. As the GP component of this mixed

model is to be interpreted as a corrective device, the simulated output H2 is therefore going to be used as the input to the GP model.



(a) **(b)**

Figure (5.68) Coupled Tanks System – Mixed Model – Small Step Training Data. Chart (a) shows recorded Input Flow data. Chart (b) shows simulated H1 and H2 values (dotted lines with H1>H2) and recorded H1 and H2 values (solid lines with H1>H2).

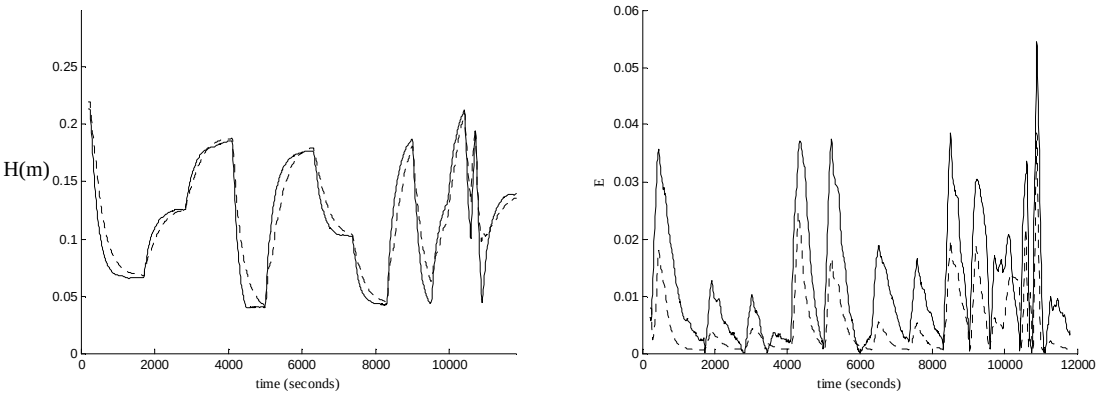


(a) **(b)**

Figure (5.69) Coupled Tanks System – Mixed Model – Mixture of Slow and Fast Transitions Test Data. Chart (a) shows recorded Input Flow data. Chart (b) shows simulated H1 and H2 values (dotted lines with H1>H2) and recorded H1 and H2 values (solid lines with H1>H2).

Before optimisation of the hyperparameters of the GP model can take place it is first necessary to complete the definition of the model structure, and utilising the same ARX structure of the previous GP models, the previous model output H2(k-1) is again going to be employed as an additional model input. Furthermore, it is also necessary to pre-process

the training data in order to reduce computational expense and avoid ill-conditioning of the covariance matrix. Therefore, the same procedure that was applied for the previous GP model identified from this training dataset was applied here, where the data was re-sampled to provide a sampling interval of 200 seconds resulting in a training dataset of 134 datapoints. After performing the previously outlined normalisation of the training targets, and ensuring that the scaling of the two model inputs is not significantly different, the hyperparameters of the GP model can then be optimised and the predictive mean and variance of the test data computed. The GP mean predictions can be seen in Figure (5.70) and the validation measures were calculated as Mean-Square Error (MSE) of 2.5148e-004, Log Predictive Density (LPD) of -21.1800, and log likelihood (LL) of 634.4323.



(a) **(b)**

Figure (5.70): Coupled Tanks System – Mixed Model – Mixture of Slow and Fast Transitions Test Data - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, the accuracy of the mixed-model is not as good as that found previously where the GP model was used to model the relationship between H1 and H2, see Test 4. However, such an outcome is unlikely, as the H2 output from the initial analytical model is not as well correlated and therefore less informative than the recorded H1 values that were previously used as the input to the GP model. Nevertheless, we would hope that the mixed model significantly outperforms the simulated model, as without such an outcome the effect of adding the GP model stage would be pointless. Comparing the mean predictions in Figure (5.70a) with the simulated values of H2 (the GP model input) in Figure (5.69b) this is certainly the case. Looking closely at the model error in Figure (5.70b), the greatest differences between the GP mean predictions and the underlying data occur during the

transitions between steady-state operating points. This is unsurprising as the simulated H2 values used as the GP model input do not reflect the more subtle characteristics of the transient response. With regard to the variance output of the GP model, a good level of correlation between the variance and the model error can be seen.

In this implementation, the potential use of the GP model as part of a mixed or hybrid modelling approach has been demonstrated. In this way the interpretability of the existing analytical description of the system can be retained, and then combined with the powerful empirically based identification methods of the GP approach. This approach has been shown to somewhat overcome the problem of the modelling the relationship between an input variable that is not smoothly varying and an output variable that is smoothly varying. Therefore, adopting such an approach may allow the GP modelling approach to be used in implementations where the nature of the empirical data may not seem to be particularly suitable. For this particular application, the mixed model could be further improved by attempting to improve the performance of the initial analytical model. Currently, the initial analytical model employs the same parameter values that have already been found to be less than optimal. Therefore, some further analysis into the characteristics of the system could yield a significant improvement in the quality of this initial model, thereby increasing the accuracy of the mixed model. Furthermore, whilst it is desirable to maintain the interpretability of the initial analytical model, there is no reason why a more empirically based method could be employed instead to create a suitable mapping between the input voltage and some intermediate or latent function space. Of course in selecting an initial model structure or nonlinear mapping that is to be combined with the GP modelling approach, it is necessary to maintain an awareness of the overall complexity of the resultant description. If two complex sub-models are to be employed in series, the computational demand and therefore speed of evaluation may become prohibitive.

5.8) Heat Transfer System

The Heat Transfer System, shown in Figure (5.71), measures the temperature of air blown through a heating element toward a heat detector placed a certain distance along a plastic insulated tube from the heating element. The system input is a desired temperature value converted into a voltage input to the heater element power supply. A fan built into the system in close proximity to the heating element provides the system with a constant airflow through the insulated tube. The level of airflow can however be adjusted through the altering the angle of the fan intake (blower angle), therefore allowing the amount of ambient air accessible to the fan to be limited. The system output is a measured temperature provided by a thermister device that may be placed in one of three different positions along the insulated tube of constant diameter. The small output voltage from the thermister is then scaled to the same 10V range as the input voltage range to allow easy comparison.

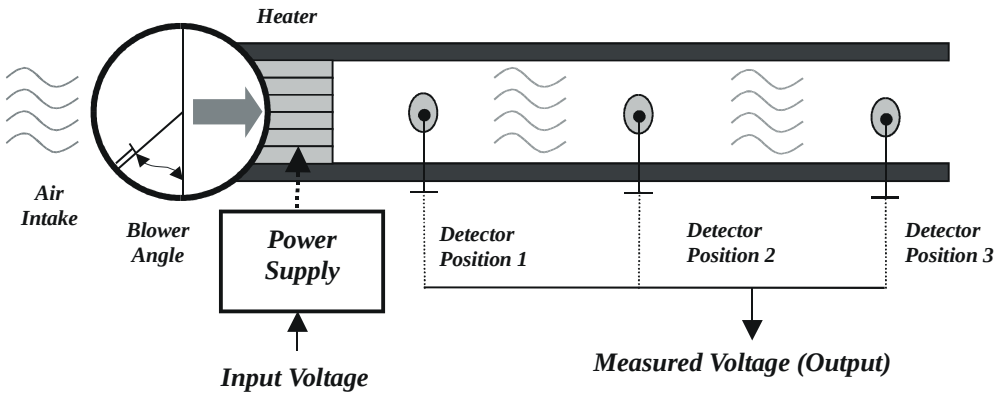


Figure (5.71): Heat Transfer System

The Heat Transfer system exhibits nonlinear behaviour throughout its operating range and has a stable open-loop response. The system response can be typically described as that of a first order system with a time delay which is approximately ‘pure’. The system transient response is significantly faster than that of the Coupled Tanks system with rise times of a few seconds only. The position of the output temperature sensor dictates the magnitude of the pure time delay (or deadtime), as a change in input temperature will take some time to propagate through the tube to the detector. As would be expected, the greater the distance between heating element and temperature sensor, the longer the delay. The angle of the fan air intake (Blower Angle) may also be varied to alter the transient behaviour of the system.

A larger blower angle will allow more air to pass through the heating element resulting in a cooling effect on the system. The transient behaviour of the system will be slightly slower as the heater power would have to be similarly increased to maintain the same temperature. A small blower angle can be seen to reduce the operating range of the system as heat is retained in the system, resulting in the base temperature of system operating range being significantly higher than that observed for large blower angles. Other significant nonlinearities that are present in the system include heat losses from the heating element and insulated tube, and turbulent airflow through the tube as a result of fan dynamics and boundary effects.

Due to the presence of such nonlinearities, no accurate analytical mathematical model of the Heat Transfer system has been developed from first principles. As a result, the application of black-box non-parametric identification methods would seem appropriate. Previous detailed investigations into the Heat Transfer System can be found in Gollee (1994) and Johansen and Foss (1995b), and the system is also used as a tutorial example in the appendix of Ljung (1999). As in the previous Coupled Tanks system application, an initial simulated version of the Heat Transfer system is investigated to demonstrate the specific characteristics of this application. After this initial investigation, the experimental methods used to collect empirical data are then described. The GP modelling process is then implemented to identify models from data collected at two different sensor positions.

5.8.1) Simulated Heat Transfer System (1st Order + Delay)

Before tackling the identification of the Heat Transfer system using real empirical data, a simple simulated version of this application is first investigated where the precise problems of dealing with the pure delay or deadtime in the Heat Transfer system are discussed. In the literature provided by the system's manufacturer, TecQuipment, the general structure of the Heat Transfer system is identified using the transfer function model (5.29).

$$G(s) = \frac{A}{1+s\tau} e^{-T_d s} \tag{5.29}$$

Although it may be possible to estimate suitable values for these parameters we are instead focusing on the development of a GP model of the system. However, we can use the fact

that the system is thought to demonstrate a first order response as the basis for developing a simulated version of the system. Overall, as the Heat Transfer system displays a first order response, the greatest distinction between this system and the previously tackled Coupled Tanks system is the presence of the pure delay or deadtime between an input transition and the output response. To investigate this aspect further a simulated model, see Figure (5.72) that loosely replicates the behaviour of the Heat Transfer system was employed.

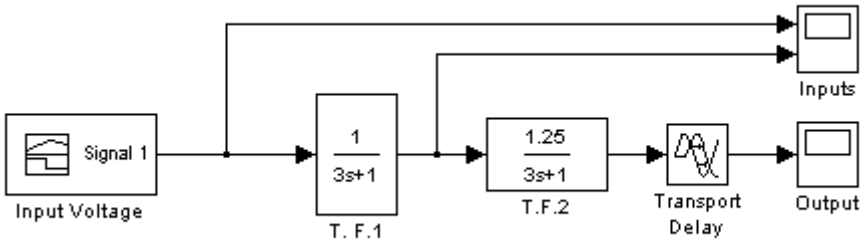


Figure (5.72): - Simulink Diagram of 1st Order + Delay System

In this example, the first transfer function block 'T.F.1' is employed to smooth the sharp step transitions of the original input so that the input information will be vary in a similar manner to that of the output, thereby reducing the potential conditioning problems associated with including steady-state data. The output of this first transfer function therefore symbolises the input voltage that would be applied to the real system, and the output from the second transfer function block 'T.F.2' to be used as the output of the system. The 'Transport Delay' block of the Simulink model then implements a delay of 4 seconds between the output of the second transfer function block and the measured output. In Figure (5.73) the delay between the input transient and output response can be seen clearly. Using this simulated model structure, the previously employed identification approach was performed where a set of training data composed of small positive and negative input steps were collected, together with a test dataset composed of larger input transitions. Both the training and test data were sampled at the same rate for simplicity with a sample interval of 1 second, resulting in a training dataset of 116 points.

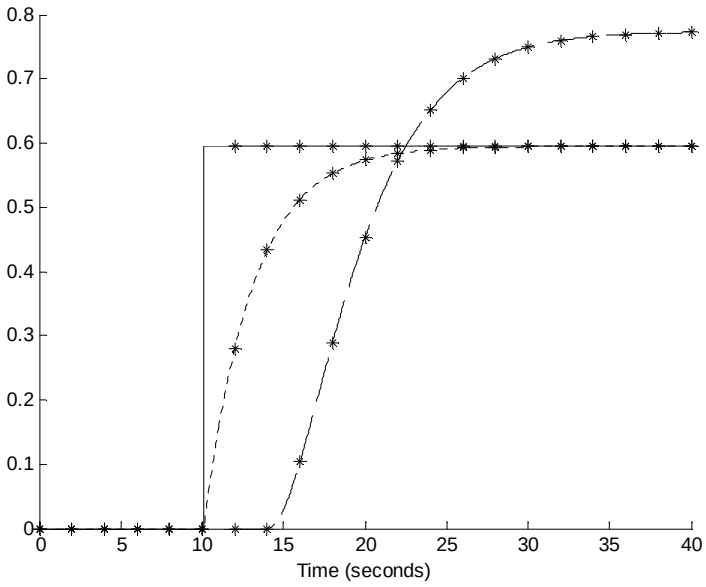
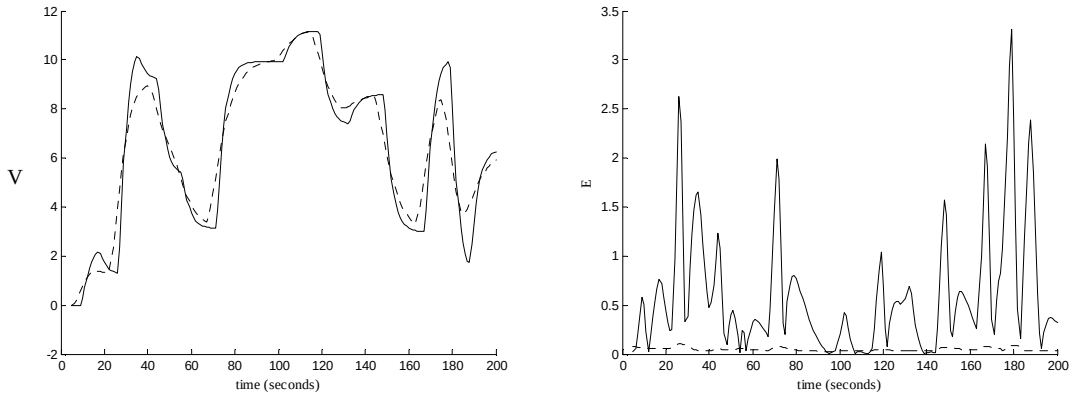


Figure (5.73): Step Response of (1st Order + Delay) Simulated System – Step Input (solid line), Smoothed Input (dotted line), Model Output (dashed line).

The problem that arises from the presence of this delay or 'deadtime' between the input transient and the output response is that if we wish to capture the full character of the output response we have to include the corresponding input data that has already reached steady state. Furthermore, if we wish to capture the precise nature of the input transition, we need to include the corresponding output data that remains at steady-state for the duration of the delay. Therefore, as this delay is increased the link between the current input value and its corresponding output value will become less significant. Furthermore, if a number of these transitions between transient and steady-state behaviour are to be included in the training dataset, this can result in including a significant amount of steady-state data in the training dataset. As a result, the conditioning of the training dataset can begin to suffer which can lead to problems identifying suitable hyperparameters. If both the delay between input and output and the difference in length of the input and output transient can be seen to be minimal, through the careful selection and sampling of transient data, conditioning problems associated with inverting the covariance matrix may be minimised. However, when confronted with a noticeable delay, the set-up of the GP model must also be considered.

Fundamentally, the presence of this delay or deadtime between the input transient and corresponding output response has meant that the relationship between the input and output has diminished. Therefore, the previously used model structure where the current input and

previous output where employed as model inputs (one-step ahead prediction) is not likely to provide an accurate description of the system. This is indeed shown to be the case in Figure (5.74), where significant error exists between the model predictions and test data, with validation measures: Mean-Square Error (MSE) of 0.7532, Log Predictive Density (LPD) of -512.1717, and log likelihood (LL) of 54.5609.

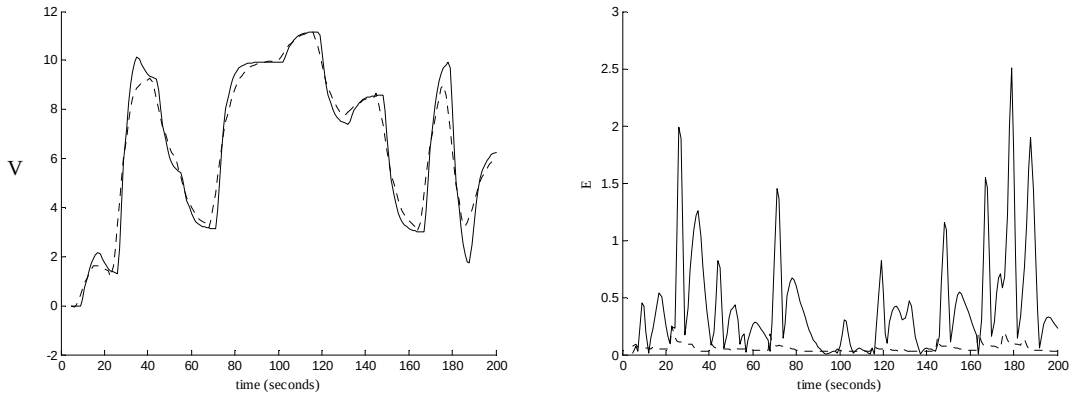


(a) **(b)**

Figure (5.74): Heat Transfer Simulated Example (Model Inputs: Input (k), Output (k-1))- Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Therefore, to tackle this problem we must attempt to modify the structure of the model so that a greater correlation between the output and certain inputs can be established. As the relationship between the 'current' input and the 'current' output has been diminished by the presence of the delay, a suitable strategy to adopt would be to include previous inputs into the model structure. By including previous inputs that are delayed an appropriate length of time (i.e. equivalent to the delay in the system) we can therefore hopefully provide more relevant information with which to predict the current output. Initially, the previous input (one-step back or (k-1)) was added to the model structure, requiring that the training dataset be re-configured, the hyperparameters retrained, and the test data also re-configured to accommodate this alternative structure. In Figure (5.75) the performance of this new model can be seen to be slightly better than the previous model structure where no previous input information was utilised, with validation measures: Mean-Square Error (MSE) of 0.3832, Log Predictive Density (LPD) of -218.3926, and log likelihood (LL) of 63.7730. Whilst this model is slightly better than the previous one, significant model error still remains. This

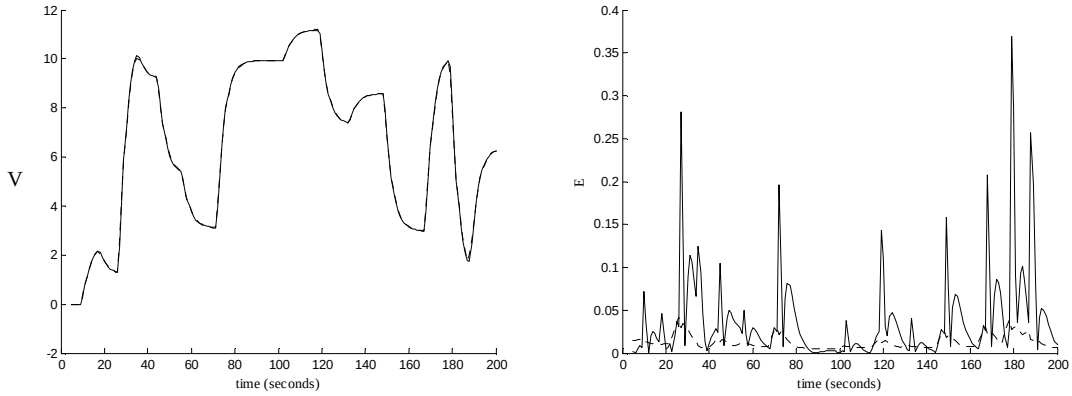
is due to the fact that the additional input (1-step back) still does not contain the information needed to account for the delay in the system.



(a) **(b)**

Figure (5.75): Heat Transfer Simulated Example (Model Inputs: Input (k), Input (k-1), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

This can be addressed by considering the properties of the simulated model where a 4 second delay was implemented. In addition as the training and test data have both been sampled so as to provide a 1 second interval between observations, we can easily deduce that the previous input (4-steps back) would likely provide more meaningful information regarding the start point of the output transition. Therefore, by replacing the previously introduced input (1-step back) with this alternative input (4-steps back) we should see an increase in model performance. This is shown to be the case in Figure (5.76) where the model predictions become almost indistinguishable from the underlying test data, with validation measures: Mean-Square Error (MSE) of 0.0044, Log Predictive Density (LPD) of -28.8988, and log likelihood (LL) of 240.4052. However, it is again worth noting that the LPD measure of this improved model has become slightly worse (i.e. higher variance) indicating that by altering the model structure we may have slightly diminished the conditioning of the resultant covariance matrix.



(a) **(b)**

Figure (5.76): Heat Transfer Simulated Example (Model Inputs: Input (k), Input (k-4), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, this simulated example has attempted to show how the presence of delay or deadtime in a system can cause conditioning problems in the covariance matrix due to the potential inclusion of steady-state data, as transients in the output will develop across a potentially longer time-scale than that of the input. Furthermore, it has also demonstrated how this delay property can be tackled through the modification of the structure of the GP model to include previous input information. This is an example of how prior knowledge can be incorporated into the identification process of the GP model. However, it is very easy to take into account prior knowledge of user defined simulated systems, and such information may not be so forthcoming in the application of the methods to the real Heat Transfer system.

5.8.2) Experimental Methods for Heat Transfer System

The same experimental approach taken toward the Coupled Tanks system was applied to the Heat Transfer system (manufactured by TecQuipment) with the open loop response being recorded under different operating conditions using National Instruments (6024E) data acquisition hardware and LabVIEW software. The blower angle was kept at a constant 70 degrees in order to investigate the effect of the pure delay (sensor position) on the accuracy of the GP model. The 70-degree blower angle was selected to allow the full operating of the system to be investigated, as larger and smaller blower angles act to restrict

the overall range of the system. The choice of sampling rate for the collection of data was an important consideration as the transient behaviour of the system is very rapid in comparison to that of the Coupled Tank system. In addition, the recorded data must accurately portray the pure delay time present in the system, or the resultant model will not reflect the true system characteristics. A sampling rate of 0.05s was initially found to be sufficient to correctly recreate the input and output voltage traces at the different operating conditions. As before, the precise sampling rate used to define the training datasets is likely to be modified to meet the previously outlined requirements over the size and conditioning of the covariance matrix.

A further important practical observation is that the Heat Transfer system has a tendency to heat up significantly as time elapses. The result of this is the base level of the input and output measurements can be seen to drift as time goes on, and the operating range of the system can begin to be restricted as lower output levels cannot be reached. Overall, this can be seen to be an example of non-stationary behaviour and as we are employing a stationary GP model, an effort was made to maintain a consistent approach toward data collection so that the effect of this underlying behaviour can be minimised in the data. In practical terms, this meant switching the machine on and leaving it to settle for a short period (~20 minutes) before conducting experiments, and switching the machine off to cool down after short period (~20 minutes) once completing a number of experiments.

As in the previous example application, the approach taken to the design of the excitation signal, and therefore the design of the training dataset, was to vary the input voltage (applied to the heating element) in a series of small upward and downward steps across the full operating range of the system. Subsequently, larger step responses were also recorded with which to validate the models identified. In addition, a number of training and test datasets were collected where the system was excited in a more arbitrary or random fashion. The squared exponential covariance function was again utilised as the system can be seen to operate in a smoothly varying manner.

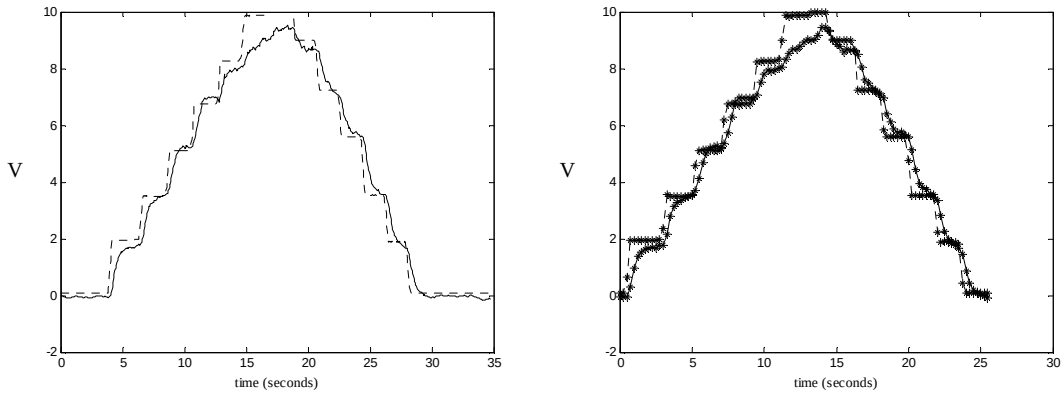
5.8.3) Application to the Real System

In this section, we are to tackle the identification of the Heat Transfer system using empirical data collected from the system. Firstly, a GP model is to be identified using data

collected at sensor position 1 where the delay between the input transient and output response is thought to be relatively minor. This process is then to be repeated for data collected at sensor position 3 where the delay is thought to be considerably larger.

5.8.3.1) GP Model at Sensor Position 1

As in the previous example, a excitation signal composed of a number of small positive and negative step inputs was utilised to generate a training dataset that would allow the whole of the input range to be covered by observations within a reasonable timescale. This dataset is shown in Figure (5.77a) where the data was collected using a sampling rate of 0.05 seconds. In order to reduce the size of the resultant covariance matrix, this training dataset was then re-sampled by a factor of 5 resulting in a sample interval of 0.25 seconds and a training dataset of 98 datapoints. Furthermore, in order to limit the potential for ill-conditioning caused by the inclusion of steady-state data, certain parts of the training dataset were carefully removed. These parts included the initial period of inactivity at the start of the training dataset, the middle portion where the input has reached its maximum, and the final period of inactivity at the end of the training dataset. The effect of this pre-processing can be seen in Figure (5.77b).

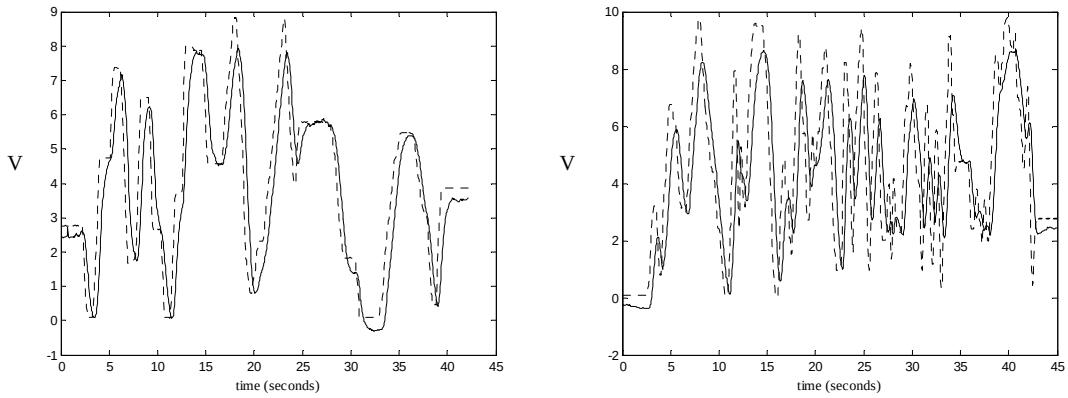


(a) **(b)**

Figure (5.77): Heat Transfer System (Sensor Position 1) – Small Step Training Data. Chart (a) shows recorded data with input (dotted line) and output (solid line). Chart (b) shows pre-processed Training Data (98 datapoints).

In order to validate the GP models identified from this training dataset, two further test datasets where collected from the Heat Transfer system whilst operating under the same conditions. These are shown in Figure (5.78), where the first test dataset is composed of a

mixture of large and small input transitions, and the second test dataset is composed of more rapidly varying transitions that are designed to appear somewhat random in nature. After identifying a GP model using the small step training dataset this 'random' test dataset shown in Figure (5.78b) is also going to be used to train a GP model in order to provide some comparison between different training datasets. The test data shown was collected at the same rate as the original sampling rate used to collect the training data (0.05 seconds), and this data is not going to be re-sampled. Therefore, for any previous or regressed inputs/outputs that are to be implemented, 1-step back of the training data will be equivalent to 5 steps back of the test data.



(a) **(b)**

Figure (5.78): Heat Transfer System (Sensor Position 1) – Test Datasets. Chart (a) shows mixture of large and small transitions with input (dotted line) and output (solid line). Chart (b) shows more rapidly varying 'random' transitions with input (dotted line) and output (solid line).

In the datasets shown in Figure (5.77) and (5.78) the delay present between input transient and output response would appear to be quite consistent across the whole of the operating range at around 0.5 seconds. Therefore, as in the previous simulated example, it may be possible to employ this prior knowledge into the development of a suitable model structure. However, in more complex systems where large numbers of variables may be present, it may not be possible to formulate any strong conclusions about the system's characteristics. Therefore, a more automated approach to the selection of model inputs that does not rely on the availability of prior knowledge would be a desirable feature for this black box modelling approach. Fortunately, such a feature has previously been discussed in Section (4.3.1.1), where the Automatic Relevance Detection (ARD) facility of the anisotropic form of the squared exponential function was introduced. Using this ARD feature, the question

of which inputs to employ in the model structure can be addressed by examining the optimised length-scale hyperparameters w_d of the squared exponential covariance function for a number of different model set-ups.

Therefore, from the optimised covariance function a length-scale hyperparameter w_d exists for each delayed input that indicates the relative importance of one delayed variable over another, as the covariance function contains a summation of distances between the training points of each input. Consequently, a large w_d may be interpreted as a significant input for prediction as it dictates a large contribution of the input in direction d to the covariance. Additionally, as w_d is inversely proportional to the horizontal scale of variation in direction d, a large w_d can also be interpreted as fast variation in the same direction. A further important consideration to note is that the relative weights between different variables should not be compared directly (e.g. previous outputs should not be compared with previous inputs). From performing the hyperparameter optimisation the relative weights of each delayed input variable can therefore be assessed and an impression of the optimal model set-up can be obtained. However, in order to be certain of the best model the predictive performance of each model must also be assessed, and the balance between model performance and model complexity (computational effort) must be traded off.

In this example five different GP model set-ups (M1 to M5) were considered where only the input is ‘stepped-back’ more than once in order to limit complexity. After performing optimisation and then prediction using the test data shown in Figure (5.78a), the hyperparameters and resultant validation measures are shown in Table (5.4). Looking at the table, as expected the introduction of previous inputs to the model set-up has increased the MSE accuracy of the model substantially (i.e. the performance of model M1 is significantly poorer than the others). However, as in the simulated case, this improvement in model accuracy is not also accompanied by an improvement in the LPD measurement. It is somewhat unclear why this is the case, but through employing additional inputs, the potential exists to damage the conditioning of the covariance matrix. From closer inspection of the length-scale hyperparameters of the more complex model structures, the previous input one-step back (k-1) appears to have the least significance in terms of contributing to the prediction. Furthermore, from inspecting the hyperparameters of model M5, the most important previous inputs appear to be those at (k-2) and (k-3), which correspond to delays of 0.5 and 0.75 seconds in accordance with the sampling rate of the

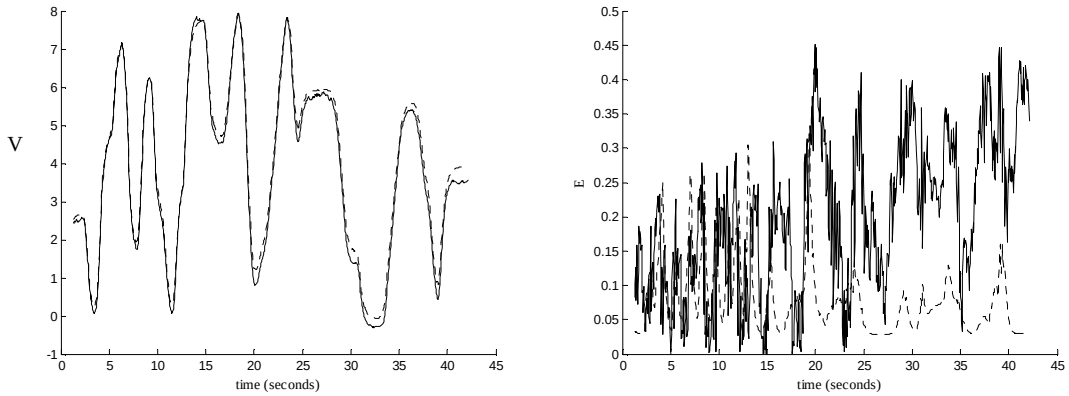
training data. This second figure of 0.75 seconds is slightly higher than would be expected from visually inspecting the data where a delay of ~0.5 seconds was observed. Furthermore, the model M3 has slightly less model error associated with it, so it may be worth attempting to re-configure the model set-up to use less inputs than in model M5, but use the more inputs deemed more 'important' by this ARD facility of the GP model.

Included **Inputs/Hyps**	**Model M1**	**Model M2**	**Model M3**	**Model M4**	**Model M5**
W_1 Output(k-1)	28.895	11.109	10.568	10.669	10.216
W_2 Input(k)	33.065	59.930	54.336	55.312	55.570
W_3 Input(k-1)	-	11.978	11.549	11.983	12.573
W_4 Input(k-2)	-	-	95.269	93.160	251.307
W_5 Input(k-3)	-	-	-	151.936	343.462
W_6 Input(k-4)	-	-	-	-	124.974
MSE	0.1933	0.0573	0.0523	0.0534	0.0547
LPD	-92.769	-44.183	-39.667	-38.335	-40.412
LL	44.806	85.724	86.425	86.695	87.367

Table (5.4) – Table of 5 different model structures (M1 to M5) with optimised hyperparameter values.

Nevertheless, after some experimentation with different combinations of inputs, the overall model performance could not be improved beyond that displayed in models M3 and M4. Furthermore, it is worth pointing out that due to the interaction or coupling between different hyperparameters, it is sometimes the case that removing seemingly less important inputs from the model structure can have a large negative effect on the overall model performance. Therefore, it is important to view the ARD facility of the GP model as a potentially useful guide to optimising the model structure, but not to view any such guidance as unquestionable evidence. Furthermore, it is important to understand that usefulness of the ARD facility is also dependent on the GP model hyperparameters being successfully optimised. If the model structure has been modified in such a manner that reduces the overall likelihood of the model, the weightings of individual hyperparameters

will not be particularly informative as they may not have been optimised very successfully. As a result, any model structures found to be potentially suitable candidates should be validated through the use of further test datasets, with the model complexity then traded off against model performance. The predictive performance of the model M3 is shown in Figure (5.79).

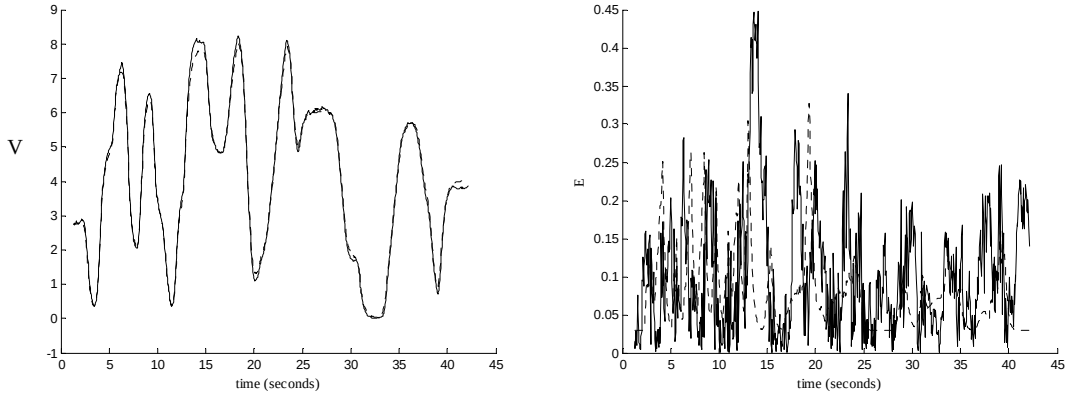


(a) **(b)**

Figure (5.79: Heat Transfer System (Sensor Position 1) – (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

From Figure (5.79) we can see that the GP predictions of model M3 do provide a reasonable level of accuracy when compared with the underlying data. However, the GP model predictions can be seen to be significantly worse in the latter half of the test dataset, and it is worth noting that the value of the output dips below zero between around 31 to 34 seconds where the output level reaches ~-0.3. Within this particular region the test data is outwith the range covered by the training dataset of which the smallest value of the output is ~-0.1. In an ideal situation both the training and test datasets would vary in a range between 0 and 10, however due to the previously mentioned propensity for the system to retain heat in the system, the base level or calibration of the input and output measurements can drift significantly. As this is an important characteristic of the system it is perhaps unwise to compensate for this drift, however as the GP model is fundamentally a precise mapping of input to output based on empirical data, in some ways we are not providing a fair test of the methodology. This again demonstrates the potential for practical problems to influence the overall viability of empirical modelling techniques such as the GP model.

If the 'base' or zero level of the output measurement of both the training and test data are reset by adding an offset, (training data shifted up by 0.1, test data shifted up by 0.3), we can see that the model performance (using the same M3 model structure) is improved significantly over the latter half of the test dataset. The increased accuracy of the model can be seen in Figure (5.80), and the validation measures were calculated as Mean-Square Error (MSE) of 0.0178, Log Predictive Density (LPD) of -10.3213, and log likelihood (LL) of 86.4249. However, it is worth noting that the predictive performance in certain parts of the first half of the test dataset (at ~14 seconds) is slightly worse.

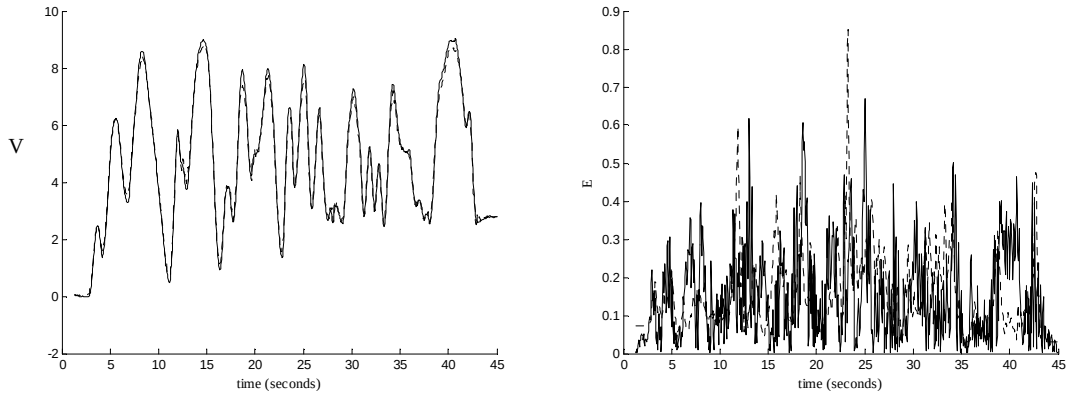


(a) **(b)**

Figure (5.80): Heat Transfer System (Sensor Position 1) with Re-set Output Scale - (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Returning to the problem of identifying the most suitable model structures, in order to gain further assurance over the slight superiority of model M3 over model M4, both models were further tested using a different test dataset composed of more rapidly varying data as shown in Figure (5.78b). This dataset was generated by manual operation of the input, but done so in an effort to appear almost random or arbitrary. As in the previous test dataset, the output of this rapidly varying test data can be seen to dip below zero in the initial part of the dataset where the input is zero, thus indicating a calibration problem or fluctuation in the base level of the output range. As we are more interested in the performance of the model on the rapidly varying aspects of this test data, rather than re-iterating the point that test data outwith the range of the training data will result in significant error, the base level of the test data was reset to zero by adding an offset of 0.35 (identified from looking closely

at the initial part of Figure (5.78b) where the input is at zero and the output is at -0.35). In order to remain consistent the training dataset is also offset by 0.1. From inspecting the validation results of the two different models, the model structure M3 was found to still maintain a very small advantage over M4 in terms of predictive performance, with an MSE of 0.0372 compared to 0.0375. The full validation measures for model M3 were calculated as Mean-Square Error (MSE) of 0.0372, Log Predictive Density (LPD) of -4.1627, and log likelihood (LL) of 86.4249. The predictive performance of the model M3 on this rapidly varying or 'random' test dataset can be seen in Figure (5.81). Overall, the identified model provides a good representation of the underlying system, however, the model does struggle to cope with the sharp peaks of the test dataset. This can be put down to the fact that only slower step variations are included in the training dataset. Nevertheless, this test dataset has shown that for this application the training dataset composed of small step excitation data has provided a good platform with which to identify a GP model using only ~100 training data observations.



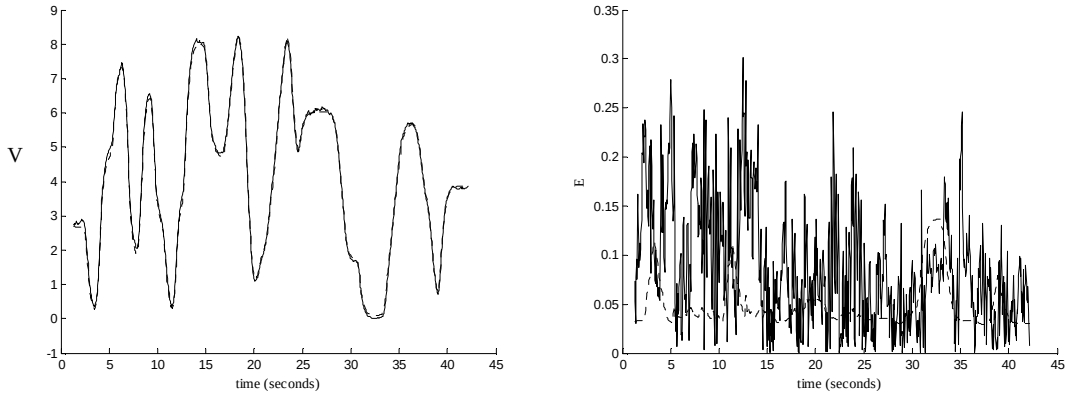
(a) **(b)**

Figure (5.81): Heat Transfer System (Sensor Position 1) – 'Random' Test Data - (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Before moving on to consider the identification of GP models from data collected at position 3, it is also worth investigating the use of random excitation data as the source of the training dataset. In the previous Coupled Tanks example application, the small step training data was found to be somewhat limited, as certain characteristics of the system were not excited. In particular, the small step transitions included in the training dataset did not provide sufficient information with which to predict larger transitions very accurately.

In the tests so far, the GP models identified from small step excitation data of the Heat Transfer system at sensor position 1 do not appear to be display the same limitations. As a result, we can understand that the Heat Transfer system at sensor position 1 does not exhibit the same kind of variation in transient response across the operating range as was seen in the Coupled Tanks system. Nevertheless, performing identification with data collected using random excitation signals may provide a useful comparison. Therefore, the rapidly varying dataset that was previously used to test the model, see Figure (5.78b), is now to be employed as a training dataset.

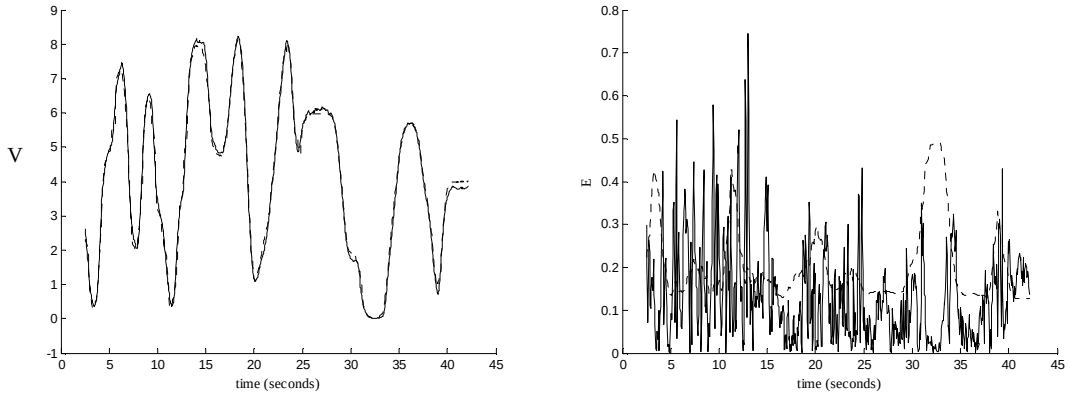
As before, the first stage to consider is the pre-processing of this rapidly varying dataset. Overall, it is unlikely that any great conditioning problems associated with presence of steady-state data will be encountered due to the nature of the excitation signal. As can be seen in Figure (5.78b), in the initial and final few seconds of the data there is some data where little excitation is present, so it is worthwhile removing these regions. Of greater concern is the sampling rate used to further process the training dataset. As this dataset is of greater length, if the same sampling rate as before is employed (0.25 seconds) the overall size of the resultant training dataset is 167 points. Whilst this is well within the range of what is computationally feasible, it is still significantly larger than the 98 datapoints included in the previous small step training dataset. After selecting the same model structure as before (Model M3), the hyperparameters are then optimised and test predictions are then calculated on the same test data as before (shown in Figure 5.78a). The GP model predictions and variance can be seen in Figure (5.82) with the validation measures calculated as Mean-Square Error (MSE) of 0.0113, Log Predictive Density (LPD) of -10.5118, and log likelihood (LL) of 112.3860.



(a) **(b)**

Figure (5.82): Heat Transfer System (Sensor Position 1) – 'Random' Training Data (167 datapoints) - (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, the performance of the GP model identified using this rapidly varying 'random' training dataset can be seen to be slightly superior to that of the previous small step training datasets. However, as we have included more points in this training dataset, it is worthwhile reducing the size of this training dataset in order to confirm any superiority. As discussed previously, it would be possible to either re-sample the training dataset to reduce the number of points included, or shorten the length of the included excitation signal. As adopting the latter strategy can lead to the removal of datapoints from potentially important regions of the operating range, the re-sampling of the training dataset would be the preferred course of action as long as the dynamics of the underlying can still be captured. For this example, the training data was re-sampled by a factor of two resulting in a sample interval of 0.5 seconds between training observations and a dataset of 85 points. If the same model structure is employed, as the sample interval has been doubled it is important to note that the delayed information present in the previous inputs will have also changed (i.e. previous inputs (k-1) and (k-2) are now delayed 0.5 and 1 seconds, instead of 0.25 and 0.5 seconds). The performance of the model trained using this smaller dataset can be seen in Figure (5.83) and the validation measures were calculated as Mean-Square Error (MSE) of 0.0311, Log Predictive Density (LPD) of -0.4813, and log likelihood (LL) of -51.3729.



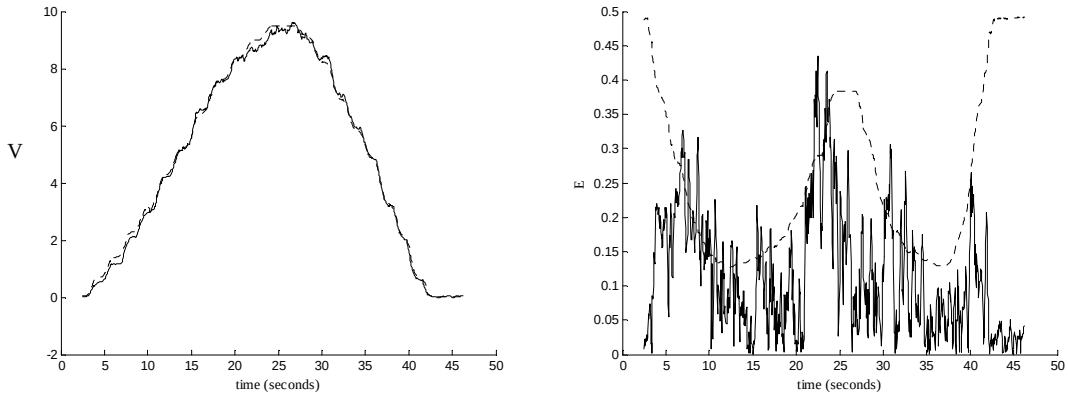
(a) **(b)**

Figure (5.83): Heat Transfer System (Sensor Position 1) – 'Random' Training Data (85 datapoints) - (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

As would be expected, the reduction of the size of the training dataset has led to a decrease in the accuracy of the model. Of greater interest is the fact that the validation measures from this test are worse than those found with the model trained with small step training data shown in Figure (5.80), where the MSE was calculated as 0.0178. Therefore, for training datasets of comparable size (98 and 85 datapoints) this example shows that models trained on the small step training data would appear to perform better than those trained on the rapidly varying 'random' data. Furthermore, it is worth pointing out that the performance of the model trained on the larger 'random' training dataset did not outperform this small step training data by a great deal.

Another important aspect of using rapidly varying 'random' excitation data is the tendency for the training data to be concentrated around the middle of the operating range. As a result, due to the sparsity of training data at the extremities of the operating range, the GP model's predictions can become inaccurate in these regions with the variance increasing substantially as confidence in the prediction has diminished. This characteristic can be seen in Figure (5.83) where as the test output reaches close to zero at around t= 33 seconds, there is a marked increase in the variance output of the model. Furthermore, as the 'random' training data and test data are not completely dissimilar in nature, in that they both consist mainly of large rapid transitions, in some ways this validation test is not providing a stern test of the model performance in the extremities of the operating range. To investigate this

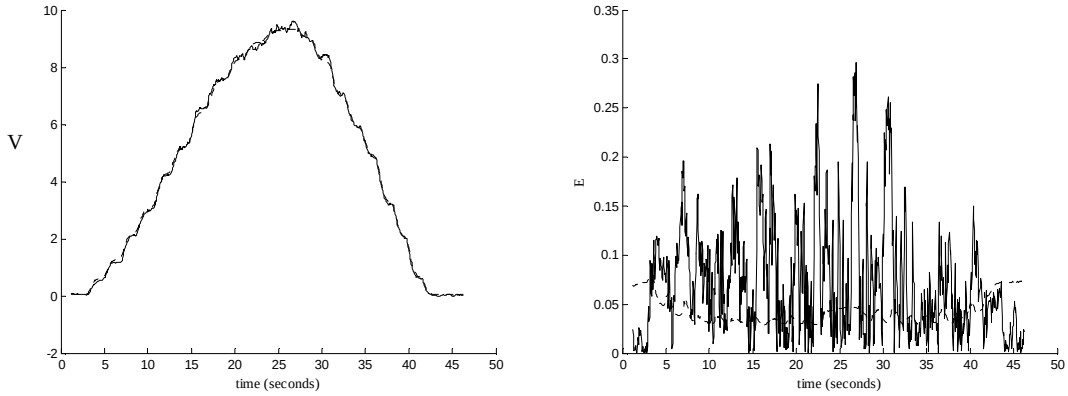
aspect, the GP model trained on the 'random' training data was then tested on data that closely resembles that of the original small step training dataset. This new small-step test dataset is different from the original small-step training dataset so that both models can be tested fairly. Firstly the GP model trained on the 'random' dataset composed of 85 points was used for prediction, with the performance shown in Figure (5.84), and the validation measures calculated as Mean-Square Error (MSE) of 0.0214, Log Predictive Density (LPD) of 0.3886, and log likelihood (LL) of -51.3729.



(a) **(b)**

Figure (5.84): Heat Transfer System (Sensor Position 1) – 'Random' Training Data (85 datapoints) – Small Step Test - (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Next, the previous GP model identified using small step training data composed of 98 points was applied to the same test data, with the performance shown in Figure (5.85), and the validation measures calculated as as Mean-Square Error (MSE) of 0.0085, Log Predictive Density (LPD) of -9.7277, and log likelihood (LL) of 86.4249. Overall, the GP model trained on the small step training data can be seen to be superior to the GP model trained on the more rapidly varying 'random' training data. Again, this is perhaps to be expected, as the training data employed by this model closely resembles that of the test data.



(a) **(b)**

Figure (5.85): Heat Transfer System (Sensor Position 1) – 'Small Step' Training Data (98 datapoints) – Small Step Test - (Model M3 with inputs: Input (k), Input (k-1), Input (k-2), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

More interesting is the variance output of the two GP models. In particular, as the variance level of the model trained on the 'random' excitation data grows substantially in the extremities of the operating range (i.e. close to maximum and minimum values); this example clearly demonstrates the tendency for the training data to become concentrated in the middle portion of the operating range, leading to problems in other regions of operating range. For the GP model trained on the small step training data this tendency is greatly reduced. By including more datapoints in the 'random' training dataset, such as reverting to the previous model containing 167 points, the level of variance can be reduced and the model accuracy improved, however both quantities remain comparatively worse than that of the GP model trained on the small step data.

5.8.3.2) GP Model at Sensor Position 3

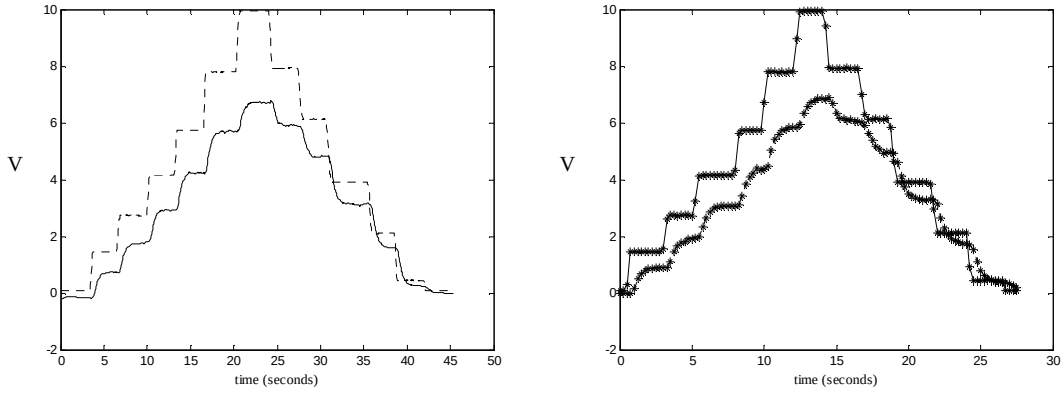
In this section we are now going to apply the GP modelling approach to identifying the Heat Transfer system where the output voltage is to be measured at sensor position 3. As sensor position 3 is located significantly further away from the heating element than sensor position 1, the delay between changes in the input being reflected in the response of the output is thought to be greater. As in the previous case, the squared exponential covariance function is to be employed and the identification process is to first investigate the use of

small step excitation data as the source of training data. The small step excitation input and output response can be seen in Figure (5.86a) where the data was collected using a sampling rate of 0.05 seconds. Before pre-processing this empirical data shown in Figure (5.86a), it is first important to examine the behaviour of the output response before attempting to reduce the size of the training dataset. Firstly, in comparison to the situation at sensor position 1, the overall scale of the variation of the output has been reduced significantly (max value of ~6.5V rather than ~9V). This is understandable as the distance between the heating element and sensor has increased meaning that less heat will reach the sensor due to losses. Secondly, the output transient response can be seen to be slightly slower than that found at sensor position 1. As the distance between the heating element and sensor is increased, any changes in the input will take longer to propagate through the insulation tube to the output and also be subject to the loss of heat reducing the magnitude of any transition. Lastly, as expected a very slight increase in the 'pure' delay or deadtime between input transient and output response can be seen in the upward steps of the excitation data, but this delay seems to diminish significantly when downward steps of the data are considered. Therefore, unlike the previous case, the pure delay in the system is not consistent across the operating range.

Overall, these changes to the properties of the system can be seen to effect the requirements for pre-processing of the empirical data into a viable training dataset. As before, in order to reduce the size of the resultant covariance matrix, the empirical dataset was re-sampled by a factor of 5 resulting in a sample interval of 0.25 seconds. However, even after re-sampling, the data is still found to include a significant proportion of steady-state data. As in the previous example at sensor position 1, as the delay between input transient and output response is increased, the relationship between the current input value and its corresponding output value will decrease. Therefore, if we wish to capture the full character of the output response this means that we have to include the corresponding input data that has already reached steady state. However, at sensor position 3 this situation is further compounded by the fact that the output transient response is slower than at sensor position 1, meaning that even more of the corresponding steady-state input data must be included.

Therefore, in order to limit the ill-conditioning caused by the inclusion of steady-state data, certain parts of the training dataset were to be carefully removed. As was the case for sensor position 1, these parts included the initial period of inactivity at the start of the

training dataset and the final period of inactivity at the end of the training dataset. However, for the case of sensor position 3, this strategy was extended to remove steady-state data from each step transition. This process was done manually by selecting valid regions of data using the time-scale as an index for the sake of simplicity, however more automatic approaches designed to include or remove data based upon meeting some criteria could also be considered. It is also important to ensure that the specific data values where each transient begins and finishes are not removed. The effect of this pre-processing can be seen in Figure (5.86b) where the training dataset includes 104 datapoints.

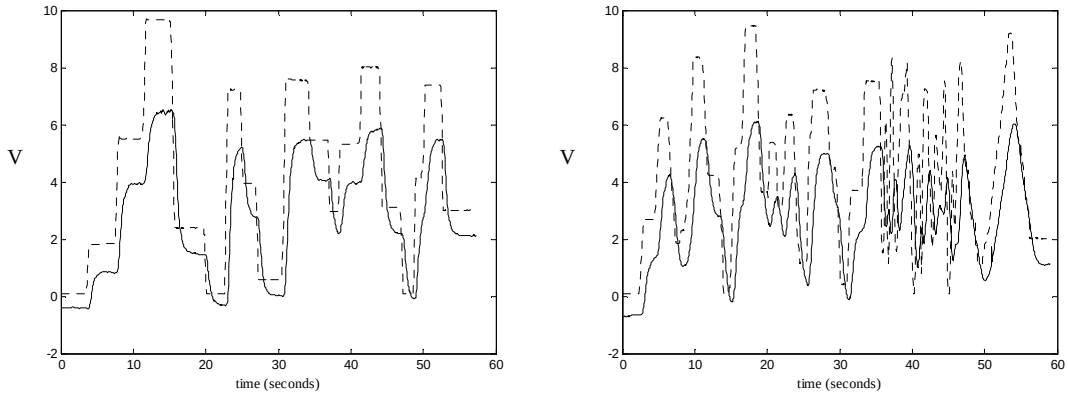


(a) **(b)**

Figure (5.86) Heat Transfer System (Sensor Position 3) – Small Step Training Data. Chart (a) shows recorded data with input (dotted line) and output (solid line). Chart (b) shows pre-processed Training Data (104 datapoints).

After identifying GP models from this training datasets, two further test datasets were collected from the Heat Transfer system using sensor position 3. These are shown in Figure (5.87), where the first test dataset is composed of a mixture of large and small input transitions, and the second test dataset is composed of more rapidly varying transitions that are designed to appear somewhat random in nature. As in the previous example, after identifying a GP model using the small step training dataset this 'random' test dataset shown in Figure (5.87b) is also going to be used to train a GP model in order to provide some comparison between different training datasets. As the pre-processing involved in the development of the small step training dataset of this example has become slightly awkward, a comparison with the 'random' dataset where pre-processing is likely to be limited is worthwhile. The test data shown was collected at the same rate as the original sampling rate used to collect the training data (0.05 seconds), and this data is not going to be re-sampled. Therefore, for any previous or regressed inputs/outputs that are to be

implemented, 1-step back of the training data will be equivalent to 5 steps back of the test data. Of further importance is the fact that in both test datasets shown in Figure (5.87), the base level of the output is again inconsistent due to variations in the plant. As in the previous example, before attempting to predict with such data, this base or zero level of the output is to be reset to zero. The level of offset required for each dataset is obtained from inspecting the initial few datapoints of each dataset where the input is yet to be excited.



(a) **(b)**

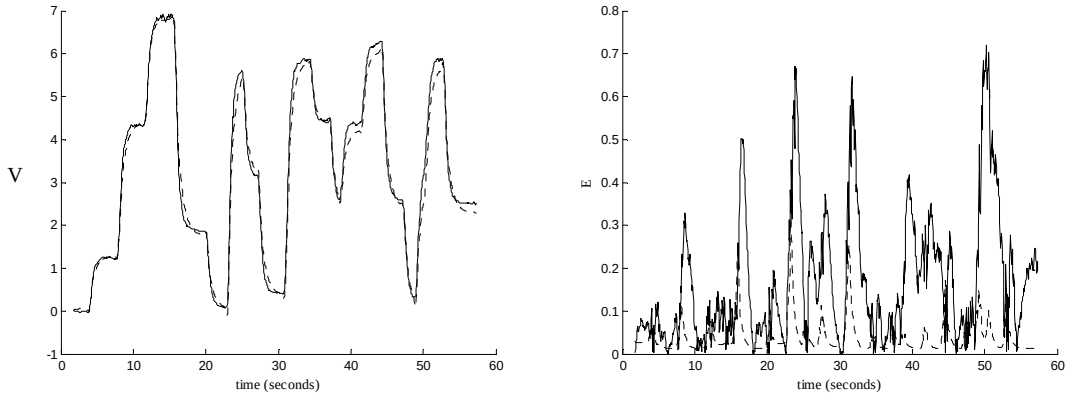
Figure (5.87): Heat Transfer System (Sensor Position 3) – Test Datasets. Chart (a) shows mixture of large and small transitions with input (dotted line) and output (solid line). Chart (b) shows more rapidly varying 'random' transitions with input (dotted line) and output (solid line).

The next stage to consider in the identification process is the optimal structure of the GP model. Firstly, from inspecting the data visually, in comparison to the situation at sensor position 1 the increase in the delay between the input transient and output response has been found to be only marginal for the upward transitions, and this delay can be seen to almost disappear in the downward transitions. Therefore, the process of inspecting the hyperparameters of different model structures where multiple delayed inputs have been included is not going to yield any significantly different result. The delay present in the system is still around ~ 0.5 seconds for the upward transitions. A further difficulty encountered is that despite the effort made to remove steady-state data from the training dataset, the overall conditioning of this data is far from perfect. The result of this is that if more and more inputs are added to the model structure, in order to repeat the previous process of examining hyperparameters of more complex structures, the overall conditioning of the covariance matrix degrades to the point where the optimisation process stops early having failed to optimise the hyperparameters.

One way round this would be to increase the sampling interval between training observations to improve the overall conditioning and thus allow more inputs to be added to the structure. The problem with this is by increasing the sample interval the previous inputs become more widely dispersed, resulting in previous inputs that are actually of little use. For example, at a sample interval of 0.25 seconds we might want to include Input(k-3), equivalent to 0.75 seconds previous, but are prevented from doing so by conditioning problems. If we re-sample the data to provide a sample interval of 0.5 seconds, not only is Input(k-3) now equivalent to 1.5 seconds previously, we have also lost the resolution in the data needed to include the desired information at 0.75 seconds previous. Therefore, in order to maintain this resolution, instead of re-sampling the data to alleviate conditioning problems it is perhaps wise to first find an upper limit on the amount of inputs that can be included without compromising the optimisation process, and then investigate the use of different combinations of the available inputs. This is an area where prior knowledge of the system can play an important role in providing a good model structure and therefore a good representation.

As the delay between the input transient and output response can be seen to have increased only marginally, the first model structure employed was that found to work best in the previous example identified at sensor position 1. This model structure was previously termed M3 (see Table (5.4)) where the model consisted of four inputs (Output(k-1), Input(k), Input(k-1), Input(k-2)). Applying this model structure to the data collected at sensor position 3 and testing on the dataset shown in Figure (5.87a) resulted in a model with the validation measures: Mean-Square Error (MSE) of 0.0534, Log Predictive Density (LPD) of -117.9385, and log likelihood (LL) of 158.7032. This result was then regarded as a benchmark to be beaten through making use of the prior knowledge obtained from analysing the change in system properties that result from moving the sensor from position 1 to position 3. After some experimentation with various combinations of inputs, the largest number of model inputs that could be included in the model structure without encountering conditioning problems was found to be five. Therefore, if we include an additional delayed input we can still optimise the hyperparameters effectively. From testing a number of different combinations, the best model structure was found through adding a further delayed input (Input(k-3)) to the previous model structure (therefore equivalent to model structure M4) which resulted in the following validation measures: Mean-Square Error (MSE) of 0.0522, Log Predictive Density (LPD) of -132.5880, and log likelihood (LL) of

166.5851. The addition of this further delayed input information makes sense in a practical sense as the delay has been seen to increase slightly in certain areas of the training data. The performance of this model can be seen in Figure (5.88).

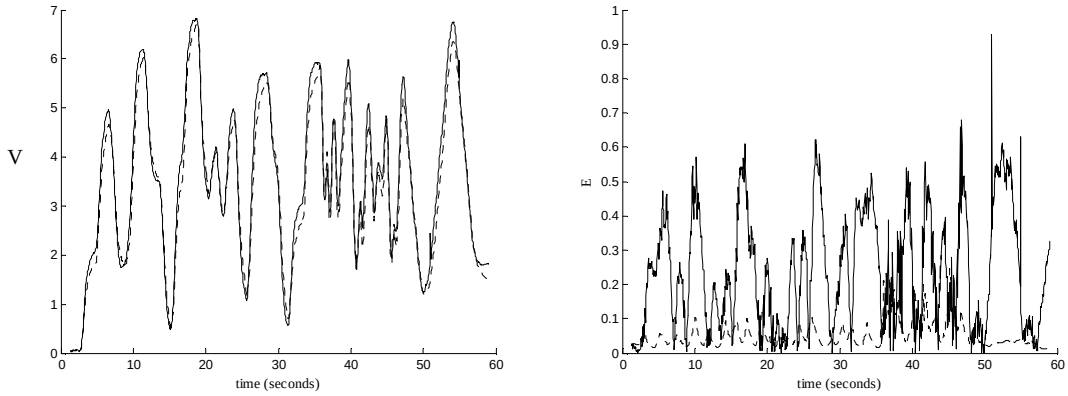


(a) **(b)**

Figure (5.88): Heat Transfer System (Sensor Position 3) – (Model M4 with inputs: Input (k), Input (k-1), Input (k-2), Input (k-3), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, the model can be seen to provide a reasonable approximation to the underlying data, however significant discrepancies remain. Furthermore, this model is considerably poorer than that of the model found at sensor position 1 using similar training data and similarly rescaled to account for base level drift (see Figure (5.80) where an MSE of 0.0178 was obtained).

In order to provide further validation, the more rapidly varying 'random' test data was then applied to the identified GP model, see Figure (5.89), resulting in the validation measures: Mean-Square Error (MSE) of 0.0860, Log Predictive Density (LPD) of -152.2902, and log likelihood (LL) of 170.8908. In both these test cases, the identified GP model can be seen to provide a decent if not outstanding approximation to the underlying data. Given that the system is prone to drift in terms of its output measurement in particular, and that the empirical data was offset in a somewhat heuristic manner, it is perhaps not surprising that a more exact match was not forthcoming. However, it is clear that the GP model identified through the use of the small step training data has not been as successful as that in the previous example where the output data was collected at sensor position 1.

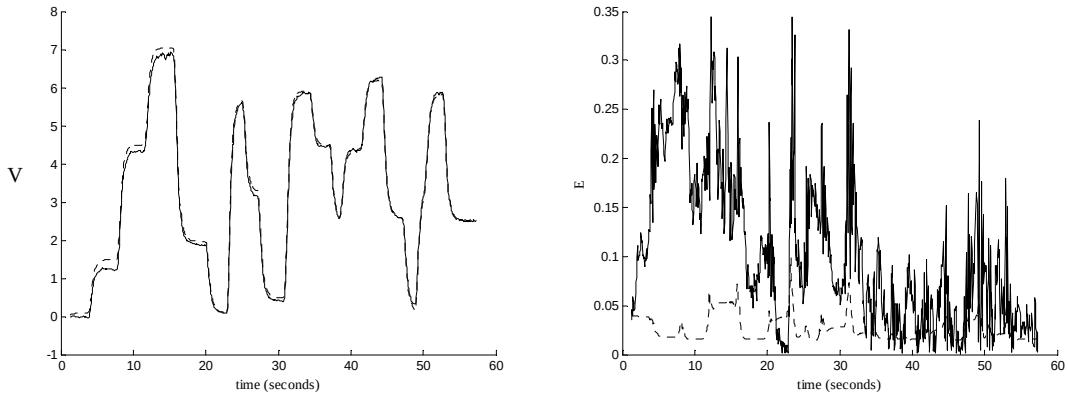


(a) **(b)**

Figure (5.89): Heat Transfer System (Sensor Position 3) – 'Random' Test Data - (Model M4 with inputs: Input (k), Input (k-1), Input (k-2), Input (k-3), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Fundamentally, this change in the location of measurement has added further complexity to the system that has not been fully grasped by the training data employed. Therefore, it would seem appropriate to consider alternative excitation signals with which to design a suitable training dataset. As in the previous 'sensor position 1' example, the more rapidly varying 'random' test data shown in Figure (5.87b) is now to be employed as the source of a set of training data, with the other test data shown in Figure (5.87a) used for validation. From inspecting the excitation of this 'random' data, problems associated with steady-state data would not appear to be likely; however the previously implemented offset on this 'random' excitation data is to be retained. Of greater consideration is the size of the training dataset. Whilst it is good practice to compare similarly sized training datasets in order to compare the performance of different excitation signals, in the case of the small step training data, the conditioning problems encountered were such that the size of the dataset, and by implication the sample interval, was not a wholly free choice. A larger amount of data resulted in a failure of the optimisation scheme due to matrix ill-conditioning, and a smaller amount of data resulted in poorly optimised hyperparameters due to a lack of information. Therefore, as these specific circumstances are not present in the case of the 'random' excitation data, the opportunity exists to employ different sampling rates more effectively. This is important, as we have seen that by changing the position of the output sensor, we have added complexity to the system. Therefore, in order to tackle this additional complexity, the potential to include more training data observations is valuable.

Initially, the sampling rate chosen to apply to the 'random' excitation data is the same as that implemented on the small step training data, resulting in a sample interval of 0.25 seconds and a training set of 232 observations. The same model structure M4 was then employed in order to compute predictions. The predictive performance of this model can be seen in Figure (5.90) and the validation measures were calculated as Mean-Square Error (MSE) of 0.0172, Log Predictive Density (LPD) of -61.4704, and log likelihood (LL) of 267.1939.



(a) **(b)**

Figure (5.90): Heat Transfer System (Sensor Position 3) – 'Random' Training Data (232 datapoints) - (Model M4 with inputs: Input (k), Input (k-1), Input (k-2), Input (k-3), Output (k-1)) - Chart (a) shows GP Mean predictions (dotted line) vs. Underlying function (solid line). Chart (b) shows GP model error (solid line) and Variance (2σ) output (dotted line).

Overall, the performance of this model trained on the 'random' excitation data can be seen to be superior to that found using the small step training data. However, the accuracy of the model is noticeably poorer for the initial step transitions in the test dataset, despite the fact that the model employs more than twice the amount of training observations. If the number of observations included in the training set is reduced from 232 points through doubling the sample interval to 0.5 seconds, resulting in 114 points after organising initial conditions, the model performance is actually significantly worse than that found using the small step training data of similar size and on the same test dataset, with validation measures: Mean-Square Error (MSE) of 0.1235, Log Predictive Density (LPD) of -0.6448, and log likelihood (LL) of -50.0829. Through increasing the sample interval between training observations, the full character of the faster transitions present in this 'random' excitation data becomes impossible to capture. Furthermore, by changing the interval between points,

the model structure is also modified as discussed previously, with the three delayed inputs at (k-1), (k-2), and (k-3), now corresponding to 0.5, 1, and 1.5 seconds rather than 0.25, 0.5 and 0.75 seconds respectively. This means that the model structure is now failing to include important information that was found to correspond to 0.75 seconds previous.

If we now consider including more training datapoints, the model structure will again have to be modified in order to maintain consistency with what is thought to be optimal. However, in this case the problem of loosing resolution and therefore important delayed information is not encountered. Nevertheless, after some experimentation, the performance of the model shown in Figure (5.90) could not be improved significantly despite the inclusion of significantly more datapoints. Therefore, as was the case for the model identified from small step data, the accuracy of the model is again limited by the quality of the excitation data present in the training dataset. As a result, whilst it may be possible to identify an improved GP model using a different 'random' excitation signal, the shortcomings of the employed 'random' excitation signal are not immediately obvious. Furthermore, it is also important to be aware that by changing the position of the output sensor, we have added a further level of complexity to the system that is perhaps not easily identifiable using only two measured variables. In essence, it is perhaps unrealistic to expect the GP modelling approach using a stationary covariance function to obtain as good a representation as that found at sensor position 1, let alone a perfect representation of the system response at sensor position 3. This is especially true due to the inherent inconsistency of the measurements that result from the equipment heating up and cooling down through operation. Therefore, to have obtained a reasonably accurate description of the system using a relatively small number of training observations cannot be deemed as a disappointment.

5.9) Summary of Experimental Results

As this project has focused on the implementation of the GP modelling approach a number of different example applications have been investigated. Therefore, rather than concentrating on a single application where the goal is to improve the final accuracy of a particular model, the experimental work has been presented in a manner that aims to illustrate the identification process. As a result, to conclude this chapter a short summary of all the experimental work completed in this thesis is provided.

Simulated Static Nonlinear Examples

In Section (5.6) a number of simulated static nonlinear examples were first investigated in order to demonstrate some of the specific properties of the GP modelling approach. Most notable amongst these properties is the growth in the variance output at test predictions made away from training data observations, and the performance of the model in operating regions where data is sparse. Furthermore, the ability of the GP modelling approach to avoid overfitting noisy data was demonstrated. A further simulated static example was used investigate the ability of the GP modelling approach to identify a less smooth or more 'spiky' nonlinear function. As expected, a GP model defined using the Squared Exponential covariance function was found to struggle in terms of identifying the sharper, less continuous transitions of the data, especially when the number of training observations included was reduced. Therefore, an alternative GP model defined using the Matérn covariance function was identified and found to perform significantly better. Through utilising the Matérn covariance function, a less stringent prior assumption over the smoothness or differentiability of the underlying function can be made. This example aims to demonstrate that through use of alternative covariance functions (rather than exclusively relying on the Squared Exponential function) better quality GP models may be identified.

Simulated Dynamic Nonlinear Examples

In Section (5.6.5) a further simulated example was then employed to demonstrate the application of the GP modelling approach to dynamic nonlinear problems. In this example a more complex Lorenz attractor that exhibits chaotic nonlinear behaviour was successfully identified. Through this example the inclusion of previous output behaviour as an

additional model input was discussed, as well as the normalisation and re-scaling of empirical data. Furthermore, the need to ensure that the intended structure of the model is preserved through maintaining awareness over the pre-processing applied to both training and test datasets is also made clear. Finally, the creation of the training dataset from selecting a subset of data from the empirical time series was discussed. Important choices are required in order to reduce the potential computational demand of the GP approach and the subset selection of data was performed through both limiting the length of the time-scale included, and altering the sampling rate used to capture the data. In either case, it is important to cover the entire operating range with training data, and also to ensure that the sampling rate allows the dynamics of the response to be accurately captured by the data.

Coupled Tanks System Example

In Section (5.7) the GP modelling approach was used to identify a model from empirical data collected from a real laboratory nonlinear system. As discussed earlier, one of the challenges in implementing the GP modelling approach is that the input data must not remain at steady state for any prolonged period of time so that the covariance matrix does not become ill-conditioned. As a result, the GP modelling approach was shown to be most suited to modelling the relationship between the H1 and H2 variables, as both variables vary smoothly along similar timescales.

Through the use of a simulated version of the Coupled Tanks system, random excitation signals were first shown to be suitable experimental strategies, as the inclusion of steady state data can be mostly avoided. However, in order to ensure that the entire operating range is covered, the length of the excitation signal must be sufficient for the data to be captured from the whole of the input range. Furthermore, such a random excitation signal must be designed carefully in order to allow the full output response to fast input transitions to be captured. Otherwise the data may become concentrated within the middle of the operating range. An alternative strategy where a number of small positive and negative step transitions where used as excitation signals was then shown to be a promising experimental approach for this simulated version of the system. Through this more deterministic strategy, prior knowledge regarding the system dynamics and overall size of the operating range could be taken into account. As a result, a training dataset that captures of the dynamics of

the system whilst covering the whole operating range could be created that is significantly smaller than would be readily found through employing random excitation signals.

Once applied to the problem of identifying a GP model from the real empirical data, the 'small step' excitation approach was found to provide a good overall description of the underlying system. The performance of this model was then improved by using an alternative training dataset where the step transitions included were slightly larger. The model found using this alternative dataset was found to be superior due to the subtle variations in the transient response of the Coupled Tanks system when different sized input transitions are considered. Furthermore, inconsistencies in the empirical data were also found to be present, leading to inconsistencies in the performance of the identified GP models. It is worth re-stating that any empirically based modelling approach will struggle to predict accurately if the behaviour present in the test data has changed from that present in the training data. Furthermore, it is important to highlight the fact that the variance output of the GP model is not likely to indicate that such an error in the model exists. In the examples presented in this thesis, the variance output was found only to be reliable in indicating regions where training data is sparse. Whilst in such regions the model prediction is more likely to be erroneous, it is important to remember that if the model is tasked with making predictions from regions that are well covered by training data, but the training data is not particularly indicative of the behaviour in these regions, the GP mean predictions are likely to be inaccurate (high model error) whilst the corresponding variance output may remain small (i.e. indicating over-confidence).

The incorporation of derivative observations was then demonstrated for the Coupled Tanks system. This extension is primarily aimed at reducing the computational demand of the GP modelling approach through the approximation of any empirical data (function observations) found near equilibrium. This data was used to identify a number of derivative observations and then combined with the remaining function observations. Overall, this extension was found to offer a slight increase in the predictive performance of the GP model when the derivative observations where combined with the previously used training dataset created from small step excitation data. However, the inclusion of derivative observations was not found to offer much benefit in terms of reducing the computational demand of the GP model. This outcome was thought to be partly due to the particular properties of the example application where small training datasets could already be

realised due to the slowly varying nature of the output response. In addition, if larger quantities of steady-state data are used to first identify and then validate these local linear models, an improved the result may be obtained.

Finally, a GP model was combined with an existing analytical model of the Coupled Tanks system in order to demonstrate the potential use of the approach in the development of 'mixed-model' implementations. Through this approach, the problem of identifying a model of the second-order relationship between input voltage (input flow rate) and output H2 was tackled. The existing analytical model was employed as an initial model or nonlinear map of the input voltage to an intermediate or latent function space, and the GP model then used to identify the residual. Overall, the mixed-model identified was found to offer a reasonable approximation of the underlying system, but significant model error was present. Nevertheless, such a mixed-model or nonlinear mapping approach could prove useful in adapting the GP modelling approach for use in applications where the data does not seem initially compatible.

Heat Transfer System Example

In section (5.8) the GP modelling approach was applied to the problem of identifying a Heat Transfer system using real empirical data. An important feature of this application is the delay or 'deadtime' between transitions in the input being reflected in the output response. Such a characteristic was found to exacerbate the problem of the input and output data not varying along similar timescales and therefore increasing the potential for including steady-state data in the training dataset. In order to tackle this problem, the model structure of the GP model was changed so that previous input information was used as current model inputs. The use of the Automatic Relevance Determination (ARD) facility of the GP model was found to offer a useful source of information with which to aid the determination of an optimal model structure. Overall, the GP model was found to provide excellent predictions of the system response at sensor position 1 (close to the heating element), and slightly less good predictions at sensor position 3 (further away from the heating element). This was put down to the added complexity of the system when operating under this configuration.

6) Conclusions

The objective of this thesis was to investigate the use of Gaussian Process (GP) models for the identification of nonlinear systems. This was carried out through attempting to model a number of different simulated and real nonlinear systems. As the majority of the existing research into the GP modelling approach has focused primarily upon the mathematics and potential of the framework for use in machine learning problems, this thesis has concentrated on addressing the implementation issues associated with the approach. Therefore, a concerted effort was made to use as much empirical data drawn from real laboratory systems as possible so that the practical implementation issues associated with system identification could be identified. From an overall perspective, the GP model proved capable of successfully representing the behaviour of the variety of nonlinear systems investigated. However, significant effort combined with the use of prior knowledge was required in the process of designing a suitable training data set with which to successfully identify the hyperparameters of the GP model. In this final chapter a summary of the major discussion points and outcomes from this project are presented, together with some recommendations for future research. Furthermore, this chapter is intended to act as a guide to the implementation of GP models for system identification purposes.

6.1) Summary of GP Modelling Approach

One of the more challenging aspects of the GP modelling approach is the theoretical background of the methodology. As the origins of the GP modelling approach lie in the fields of statistics and machine learning, some of the mathematical concepts employed by the method are not immediately interpretable by those from a more engineering-based background. In particular, the application of Bayesian probability theory is not something that is often encountered by those working on typical system identification or automatic control problems. Furthermore, it is difficult to provide a more visual interpretation of the GP methodology where diagrams can be used to aid description (e.g. Neural Networks are often described using diagrams of the neuron structure). A further problem in gaining a foothold into the theory of the GP methodology is that the properties of the approach are often described or defined in terms of how they compare with alternative

methods of machine learning, such as Neural Networks or kernel based methods. Whilst the fields of machine learning and system identification are closely linked, significant differences exist in the application of methods and the types of problem investigated. As a result, a significant proportion of the existing literature devoted to GP models is not easily digestible by those unfamiliar with machine learning or probabilistic analysis.

Therefore, before discussing the specific methodology of the GP model, in Chapter 2 a review of the field system identification was completed where some of the more popular methods used to identify nonlinear systems from empirical data were discussed. One of the main intentions of providing this background discussion is to draw attention to the fact that some of the more complex modelling approaches based on Neural Networks and other multiple model implementations have notable disadvantages. In particular, a significant challenge routinely encountered in identifying complex multiple model descriptions is the optimisation of the model parameters and structure. Furthermore, multiple model structures based on the identification of local linear models have been found to be of questionable accuracy when tasked with predicting outside of equilibrium operating regions. In addition, in many cases the complexity of the model structures requires that a large amount of training data be available so that optimal model parameters and structure can be identified, thus avoiding underfitting/overfitting through trading-off Bias/Variance model error.

The primary motivation for the use of GP models is that through the application of Bayesian probability, some of the problems associated with optimising complex model structures can be avoided. Furthermore, through the specific use of Gaussian processes, some of the mathematical difficulties associated with employing Bayesian analysis can also be avoided. Both of these aspects are discussed in detail in Chapter 3. The result of this discussion is that the GP modelling approach can be seen to be highly suitable for empirical modelling problems where the amount of data is limited, as the optimisation through Bayesian probability automatically implements a preference for simpler model descriptions (Automatic Occam's Razor) therefore reducing the potential for overfitting. Furthermore, as the training data is retained within the covariance matrix, what training data is available ultimately forms part of the model structure. As stated at the beginning of this section, the disadvantages of the GP modelling approach are that the model is not particularly interpretable, and due to the need to retain training data information in the

form of a covariance matrix (rather than retain information in the form of a number of identified parameters), the modelling approach can become computationally demanding if a large number of training observations are required to identify an accurate description. A further interesting feature of the GP modelling approach is the fact that the output from the model is a probability distribution. This feature means that the uncertainty or variance of each prediction is readily available, something that is not the case for the majority of modelling approaches.

6.2) Guide to GP Model Implementation

In this section a guide to the implementation of the GP modelling approach is to be provided. Whilst a great deal of this information has already been presented in earlier sections of this thesis, especially Section (5.5), it is worth reiterating the main outcomes. Overall, the system identification process of the GP modelling approach can be described by the following:

1) Examine Prior Knowledge of System
2) Collect Experimental Data
3) Select Covariance Function
4) Create Training Dataset
5) Optimise Hyperparameters
6) Compute Predictions
7) Validate Results

As with any system identification process, the GP modelling approach is iterative where modifications to any stage may be required. However, the identification of a GP model can be generally decomposed into two components: 1) Select a suitable covariance function, and 2) Design a ‘good’ training dataset. From this point optimal hyperparameters can be identified and predictions can be made.

6.2.1) Choice of Covariance function

In the implementation of the GP modelling approach, one of the key stages is the selection of a suitable covariance function. This covariance function is to be used to generate the prior distribution (defined as a Gaussian process that is specified by a zero-mean assumption, together with a covariance matrix generated from applying the optimised covariance function to the training data observations) with which the predictive distributions are to be inferred through Bayesian inference. The overall role played by the covariance function is described in Section (4.1), together with a description of a number of existing covariance functions in Section (4.2). The Squared exponential covariance function is the most popular covariance function and its use implies an assumption of stationary and smoothly varying nonlinear behaviour. Whilst other covariance functions have been proposed, there is relatively little existing research into the practicalities of adopting them. Furthermore, the smoothness assumptions of the Squared Exponential covariance function make it suitable for identifying many real systems as such applications are normally designed to be operated in a smooth or consistent manner in order to facilitate manual or automatic control. Therefore, it is this covariance function that has been primarily used in the experimental investigations undertaken. Nevertheless, in some circumstances the smoothness assumptions inherent in this choice of covariance function may be unrealistic, and an alternative function should be considered. In particular, the Matèrn covariance function may be useful for systems that exhibit a rougher response. Overall, we can see that prior knowledge of the systems characteristics can prove to be important in selecting an appropriate covariance function.

6.2.2) Design of Training Dataset

For the purposes of system identification, the training dataset must be created through the combination of excitation signal design and pre-processing of empirical data. Furthermore, in the design of a suitable training dataset, the size and conditioning of this dataset must also be considered carefully.

6.2.2.1) Size of Training Dataset

In Chapter 4 the mathematical and computational implementation of the GP modelling approach was discussed. As stated previously, one of the disadvantages of the GP modelling approach is that the method can become computationally demanding if the training dataset is to include a large number of observations. As a result, reasonable upper limits on the size of this training dataset were discussed, and for the direct implementation of the GP model using average desktop PC facilities this limit was set at $N<1000$. For larger scale problems a number of approximate methods were also discussed including sparse matrix methods, fast matrix vector multiplications and derivative observations. Overall, the existing literature has not provided conclusive evidence as to which of these methods is preferable. Furthermore, the use of simple Subset of Data methods where excess data is simply discarded has been found to remain competitive in many cases. In addition, in the experimental work carried out in this thesis, through careful pre-processing of the training data, the size of the training dataset could be kept under $N<1000$, therefore making the use of such approximate methods unnecessary. Nevertheless, these extensions to the GP modelling approach should be considered for more complex systems.

Overall, in acting to constrain the size of the training dataset, one of the main tools at our disposal is in modifying the sampling rate chosen to collect the data. However, it is important to remember that whilst it may be tempting to reduce the sampling rate (increase sample interval) in order to include a longer time-series that may provide observations across a broader operating range, this will lead to a loss in the resolution of the data. As a result, some of the more subtle aspects of the system's response may not be adequately described by the training data. This can therefore lead to a poor model accuracy (GP mean predications), and also unrealistically low variance predictions. Therefore, it is important to choose the sampling rate carefully so as to retain the ability to include as much information as possible.

6.2.2.2) Conditioning of Dataset

A further important aspect of the computational implementation of the GP modelling approach is the conditioning of the covariance matrix. In order for the hyperparameters to

be optimised and predictions to be computed various matrix inversion and multiplication operations must be performed accurately, therefore the covariance matrix must not be ill-conditioned. Whereas the definition of a valid covariance function has a role to play in the specification of a well-conditioned (positive semi-definite) covariance matrix, another aspect that is not given much coverage in the existing literature is that the training dataset must also be carefully constructed. This aspect is especially important for system identification purposes, et as one of the key stages in the development of an empirical model is the design of the experimental approach and the collection of excitation data. This is perhaps different from typical problems found in statistics and machine learning where the data to be analysed may not be within the control of the modeller.

In the examples investigated in this thesis one of the main causes of ill-conditioning in the covariance matrix was the presence of steady-state data in the training dataset. Therefore in order to ensure a well conditioned covariance matrix the presence of steady-state data must be minimised through the use of an appropriate experimental design strategy (discussed below) and careful pre-processing of the training dataset. With regard to pre-processing of the training data, the main strategy employed in the work presented in this thesis was to manually eliminate portions of the training data that were found to contain prolonged periods of steady-state data. Through adopting this strategy, it was possible to ensure a well conditioned covariance matrix. An alternative strategy that can also be used in addition to the manual editing of the training dataset is to employ some method regularisation. Through adding a small level of noise to the diagonal of the covariance matrix, it is possible to improve the conditioning of the covariance matrix. However, it is important to remember that through the addition of noise, the accuracy of the model may be diminished if care is not taken.

6.2.2.3) Experimental Design

The excitation signal must be designed so that the dynamics of the system are sufficiently excited across as much of the operating range as possible. As the GP model can be understood as an interpolation method, the extrapolation properties (i.e. making predictions outside of the range covered by training data) of an identified model have been found to be poor. Of further importance is that the excitation signal must be

sufficiently excited, as one of the key practical outcomes from this work is that the presence of even small quantities of data near steady state is likely to lead to a significant deterioration in the conditioning of the covariance matrix. However, for system identification problems the potential for encountering steady-state regions of data is great, as input and output variables may vary in a quite deliberate manner (e.g. step inputs, delayed responses etc.). Therefore, we are unlikely to be always dealing with two or more variables that vary smoothly or even roughly in tandem with one another in some arbitrary manner.

One straightforward strategy for the design of the excitation signal is to employ randomly varying inputs. Through the use of random inputs, the potential to record long periods of steady-state data is greatly reduced. However, a downside with such an approach is that the slower dynamics of the system being investigated can fail to be accurately conveyed by such training data. Furthermore, the extremities of the operating range can be sparsely populated by observations as the recorded data becomes concentrated within the middle of the operating range. Therefore, in some applications a more deterministic approach to the design of the excitation signal may prove to be a preferable strategy. This was the case for the Coupled Tanks system investigated in this thesis where the slow transient response could be captured by the training data more successfully using a number of step transitions. A further benefit of utilising this strategy was that function observations could be captured from the entire operating range of the system in a smaller timescale. This is important as it allowed a smaller training dataset to be created, therefore decreasing the computational burden of the GP modelling approach.

6.2.2.4) Model Structure Selection

Another important stage in the modelling process is in the selection of suitable inputs or regressors. This is a stage where prior knowledge of the system can prove invaluable as any characteristics of the system (e.g. delayed output response) can be accommodated into the structure of the GP model. Furthermore, it is possible that utilising certain variables as inputs or outputs may appear to cause problems in adhering to the demands of the chosen covariance function (e.g. mapping very fast input transitions to very slow output transitions). In such a case alternative covariance functions should be considered, and the ‘mixed-model’ approach detailed in section (5.7.6) may also be worthy of

investigation. Furthermore, if previous inputs/outputs are to be employed as additional model inputs, it is important to ensure these delayed inputs are incorporated in a manner consistent with the desired model structure. As the size of the training dataset must be kept within reasonable limits, a discrepancy between the sample intervals of the training and test datasets can result. Therefore, in order to maintain the integrity of the model structure it is important to ensure that these delayed variables are incorporated correctly. Further adjustments to regressors may also be required if any variables require normalisation.

The Automatic Relevance Detection (ARD) feature of the Squared Exponential covariance function is another feature that can be used to identify an optimal model structure. This feature allows the relative importance of different inputs to be assessed. However it is worth reiterating that the ARD feature is not foolproof and is dependent on the model structure selected being sufficiently viable so that the hyperparameters can be successfully optimised. Furthermore, more complex systems than that examined in this thesis may require a more sophisticated approach to selecting optimal regressors.

6.2.3) Training Hyperparameters

In order to compute accurate predictions of system behaviour, the hyperparameters of the selected covariance function must be optimised using training data collected from the system. In Chapter 4, the optimisation of hyperparameters using marginal likelihood maximisation and Monte-Carlo methods is described A clear preference in the literature has been shown for the marginal likelihood maximisation, and this method has been used in the experimental work carried out. Furthermore, the optimisation of hyperparameters through this method has been found to perform robustly as long as a suitably conditioned training dataset has been created. Therefore, through attempting optimisation of the hyperparameters, the conditioning of the training dataset can be assessed.

6.2.4) GP Model Validation

As with any modelling approach it is important to validate the identified model through comparing the predictions from the model with the observed output response. Separate

test or validation datasets must therefore collected from the experimental set-up so that the generalisation ability of the model can be assessed through cross-validation. Standard measures of prediction accuracy (e.g. mean square error) can then be used to examine the performance of the model. In addition, the variance output of the model should also be examined closely to provide a further indicator of the performance of the GP model. The variance at individual test cases can be easily understood through plotting error bars (2σ) on prediction charts, and an overall indication of the uncertainty in the model can be given through computing the log predictive density. If the predictive accuracy of the model is found to be deficient in a particular area of the operating space it is common that the variance output of the model is also higher in this region. Therefore, this information can the be used to modify the training dataset in order to improve the accuracy of the model in this particular region. Nevertheless, it is also important to remember that it is also possible to obtain poor model predictions in combination with a low variance output. Such a result is likely due to the training dataset not accurately representing some aspect of the system's behaviour. Once again, it may be necessary to modify the training dataset to address any particular problems encountered. Finally, it is also important to reiterate that if the variance output of the model is to be utilised directly, such as in the basis of model predictive control algorithm, it may be worthwhile implementing the 'propagation of uncertainty' extension to the GP model.

6.2.5) Final Thoughts

The overall outcome from this discussion is that in the creation of a suitable training dataset a number of different practical considerations (overall matrix size, overall matrix conditioning, coverage of operating range, dynamics appropriately sampled, etc.) must be borne in mind. As a result, the use of prior knowledge of the system has been found to be very important in the creation of this training dataset. This means that the methods applied here cannot truly be described as a 'black box' approach. Furthermore, it is worth returning to the question over the overall interpretability of the GP modelling approach. As the training data is retained in the model (in the form of the covariance matrix), from a certain perspective, the GP model can be thought of as highly interpretable. This is due to the fact that the quality of the model can be directly linked to the location or relative sparsity of the training data (assuming the dynamics are captured well, and the covariance matrix is appropriately conditioned). This is contrast to alternative multiple

modelling approaches where the structure and parameters of individual sub-models or neurons must be identified from data and prior knowledge. In essence, the overall robustness of the GP method means that the problem of identifying an accurate model is almost reduced to creating a suitable training dataset. Therefore, whilst the actual theoretical background of the GP model and the identified hyperparameters of the covariance function may not be especially easy to interpret, the actual process of identifying GP models can easily be understood as it is the training dataset that must be 'optimised' through pre-processing rather than a large number of parameters. Nevertheless, for applications where large numbers of input dimensions are present, the problem of pre-processing a suitable training dataset may become challenging.

6.3) Future Work

Throughout the course of this thesis a number of potential extensions to the GP modelling approach have been discussed. In particular, the following extensions could prove to be worthwhile avenues for future research:

More Complex Applications

As this thesis has been concerned with discussing the general implementation of a relatively unproven method of nonlinear system identification, the experimental applications investigated have been relatively simple (i.e. only a few input dimensions, smoothly varying system responses). Therefore, the next obvious stage to consider is the application of the GP methodology to more demanding multivariate applications (e.g. multiple output problems). Through applying the GP modelling approach to more complex applications, some of the methods and extensions presented here could be developed further.

Alternative Covariance Functions

One of the most important future directions for research should be into the use of alternative covariance functions. Whilst in the experimental applications investigated in this thesis have been successfully tackled using the most popular Squared Exponential

covariance function, this is due to the smoothly varying characteristics of those systems. Other applications where less smooth responses are to be identified will not be as well approximated by this ‘standard’ GP model implementation. Furthermore, the fact that the Squared Exponential covariance function is stationary is another notable limitation. Therefore, further investigations into the use of alternative covariance functions should be a priority if the full potential of the GP modelling approach is to be realised. Fortunately, a number of different covariance function have already been proposed, see Section (4.2) together with methods to combine different covariance functions. However, these proposed alternative covariance functions have not been the subject of much experimental investigation. Therefore, in order to address this current lack of information, an investigation into the suitability of these different covariance functions for different nonlinear problems would be a valuable addition to the field.

For engineers charged with identifying real nonlinear systems, this would ideally involve the use of empirical data collected from real systems. If the use of different covariance functions remains exclusively researched within the machine learning and computing science communities, it is possible that standardised or benchmark test data sets, or simple simulated static nonlinearities, will continue as the prime investigative tools. Such investigations can prove to be valuable resources, but the application of methods to real world dynamic systems may prove to be more enlightening to those working on system identification problems. Overall, it would be desirable to establish better links between types of observed behaviour and the covariance functions best suited to identify them, as unlike Neural Network approaches that can act as universal function approximators, the properties of different covariance functions can sometimes be made more interpretable and applicable to specific problems. However, it is worth pointing out that the adoption of alternative covariance functions has implications for training data selection and overall interpretability. One of the advantages of the Squared Exponential function is that the hyperparameters have interpretable roles, and that we can optimise them using the Bayesian marginal likelihood maximisation method. With more complex covariance functions the potential exists for greater optimisation difficulties associated with multiple local optima. Hence, the potential need to use MCMC methods may become apparent. The overall consequence may be that the more complex GP model may require a more complex and time-consuming model optimisation strategy. Therefore, some of the

original advantages of the GP modelling approach may be lost, as the added complexity may result in alternative non-Bayesian modelling approaches becoming more attractive.

Approximate Methods

In Section (4.5.5) a number of alternative approximate methods were discussed, where the objective is to reduce the computational demand of the GP modelling approach. Although various methods have been proposed and empirical investigations have been conducted, as yet there is no definitive answer as to which would be the preferred method in a particular situation. This is a difficult thing to analyse, as many contributing factors exist. Nevertheless, further analysis of these methods from a practical perspective would be a valuable addition to the field. As a result, the trade-off between model accuracy and computational expense could be better implemented. Furthermore, the current literature devoted to exploring the majority of these approximate methods (especially sparse matrix methods) has been developed primarily for use in machine learning cases where static nonlinearities are commonly used as example implementations. As a result, the use of many of these approximate methods has not been explored towards system identification problems. In addition, as discussed in Section (4.5.6), many of the approximate methods do not initially appear to be particularly in keeping with the demands of the system identification problem, where careful pre-processing of the training data has been required. This is at odds with some of the random subset selection methods typically used to sample training data from empirical data.

As part of the experimental work carried out in this thesis, an attempt was made to adapt the Subset of Regressors sparse matrix method for use in forming an approximation to the GP model of the Heat Transfer system. Unfortunately, computational problems where encountered where the approximated covariance matrix was routinely found to be ill-conditioned, thus leading to poor quality predictions. This is despite the fact that in some cases the same covariance matrix was found to be adequately conditioned for the standard GP predictive methods to be employed. It is certainly possible that some mistakes in the implementation of the method were made, but it is also the case that existing research is not very clear as to whether or not such an approximate method is well suited for application to GP models of dynamic systems where previous input and output information are to be used as model inputs. Nevertheless, further research into

these methods would be very useful as although the systems investigated in this thesis are relatively simple, and therefore allow accurate models to be identified from small training datasets, more complex applications may necessitate the use of approximate methods.

Derivative Observations

Although this extension to the GP modelling approach was investigated using the empirical data from the Coupled Tanks system, any overall benefit in terms of reducing the computational expense of the resultant model was not clearly evident. As discussed in Section (5.7.5), it is possible that the specific characteristics of the example system made it an unsuitable candidate for demonstrating the full ability of this extension. Nonetheless, this is an interesting extension to the GP modelling approach that is straightforward to implement. Therefore, further investigations into the incorporation of derivative observations may prove to be valuable.

Mixed-Model Implementations

Another extension to the GP modelling approach that was explored through the experimental work carried out on the Coupled Tanks system was the use of a 'Mixed' model or Nonlinear Mapping approach. Through this extension, some of the implementation difficulties associated with the GP approach can be bypassed through employing an initial model structure or nonlinear mapping that can be used to generate intermediate or latent function values that are more compatible with the standard GP modelling approach (e.g. Squared Exponential covariance function). Such an approach can be seen to have considerable appeal in that existing descriptions of a particular system can potentially be retained and then improved through the use of a secondary GP model stage. As this extension can perhaps extend the usefulness of the GP modelling approach, further investigations into the use of 'Mixed' models would be useful.

Multiple GP Models

A further potential research avenue is the use of multiple GP models, as discussed previously in Section (4.5.6.2.1). Through this proposal, some of the implementation

problems associated with the GP modelling approach could be tackled through applying a ‘divide and conquer’ approach. Therefore, instead of attempting to include large quantities of training data into a single covariance matrix in order to specify a ‘global’ GP model of the system, the operating range could be partitioned into local operating regimes as in alternative multiple modelling approaches. At present, there has been only a limited amount of research into this aspect, as detailed in Section (4.5.6.2.1), but further investigations could prove to be useful. However, it is worth bearing in mind some of the initial motivation behind the GP modelling approach was that the problems associated with optimising complex networks could be avoided. Therefore, in some respects there may be limited appeal in moving in the reverse direction.

Control System Design with GP models

The need for better or more informative models is often motivated by the desire to obtain better or more precise control systems. Therefore, perhaps one of the most important avenues for future research would be the continuation of research into automatic control systems that make use GP models. In existing research, see Section (5.1.1), the variance output has shown promise with regard to implementing model predictive or adaptive control. However, as yet the GP model has not been proven to be particularly useful or compatible with other control algorithms. Therefore, great potential research avenues exist with regard to implementing controllers based on GP models.

Real-Time Implementation

Related to the application of GP models to control system design is the question of how well the methodology is suited to real-time implementation. As a significant amount of the existing literature, and indeed this thesis, has been concerned with managing the computational demands of the GP model, this is an important topic. Once the training data has been finalised and the hyperparameters identified, the computational requirement for the prediction at a new test point still requires the inversion of the covariance matrix. Therefore, for control applications that require real-time or near instantaneous updates as to the model’s predicted output, this burden of calculation might prove to be problematic. As a result, a practical investigation using real hardware would be valuable in order to assess the potential of the GP model as an ‘online’ estimator of

system behaviour. Such an investigation may also prove to be informative with regard to the suitability of some of the extensions to the GP model that have been discussed. Further considerations that could be investigated include the potential of the GP model to be adapted to allow new data to be added online. Through adding data to the training dataset whilst the system is online, or modifying it in some other way, the potential exists to tune the performance of the model if required. For example, if through the operation of the system we encounter new or off equilibrium operating regions, we may wish to learn this behaviour and incorporate it into our model. Such a feature is also comparable with the Active Learning strategies discussed in Section (2.3.2.1). Initially, such a feature would appear to be computationally demanding, as hyperparameters may need to be re-optimised if the training dataset is modified. Nevertheless, such a feature is worthy of investigation as it may also prove worthwhile in the application of approximate methods where the goal is to optimise the efficiency of the training dataset.

Appendix A – Probability Definitions and Background

This section provides information regarding the axioms of probability and background definitions most relevant to Bayesian analysis. Most of this information is readily available in many standard mathematics or statistics textbooks and has been gathered together for inclusion here to provide a convenient source using common notation throughout.

Probability of an Event

In the mathematical treatment of probability theory, the probability of an event *A* is represented by a real number in the range from 0 (an **impossible** event) to 1 (a **certain** event), and written as *P*(*A*) with interval [0, 1].

Probability of an Opposite Event

The probability of an event opposite to *A*, is defined as the complement rule:

$$P(\text{not } A) = P(\overline{A}) = 1 - P(A)$$

Joint Probability

Joint probability is the probability of two events in conjunction. That is, it is the probability of both events together. The joint probability of *A* and *B* is written as $P(A, B)$ or $P(A \cap B)$.

If two events are **independent**, the joint probability is:

$$\text{P}(A \text{ and } B) = \text{P(A,B)} = P(A \cap B) = P(A)P(B)$$

Addition Rule

If two events are **mutually exclusive** (i.e. two events that may not occur at the same time), then the probability of **either** occurring is:

$$\mathrm{P}(A \text{ or } B) = P(A \cup B) = P(A) + P(B)$$

If two events are **NOT mutually exclusive** (i.e. two events that may not occur at the same time), then the probability of **either** occurring is:

$$\mathrm{P}(A \text{ or } B) = P(A) + P(B) - \mathrm{P}(\mathrm{A \text{ and } B})$$

Conditional Probability

Conditional probability is the probability of some event A, given the occurrence of some other event B. Conditional probability is written $P(A \mid B)$ and is read as 'the probability of *A*, given *B*.

The conditional probability of *A* given *B* is defined as:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A, B)}{P(B)}$$

Rewritten: $P(A, B) = P(A \mid B)P(B)$

This relationship is termed the **Product Rule** and is the fundamental rule of probability calculus as it allows us how to combine conditional probabilities for individual variables, to define joint probabilities for sets of variables.

Marginal Probability

Marginal probability or prior probability is the probability of one event, **regardless** of the other event. Marginal probability is obtained by summing (or integrating, more generally) the joint probability over the unrequired event. This is called marginalisation. The marginal probability of *A* is written *P*(*A*), and the marginal probability of *B* is written *P*(*B*).

Bayes' Theorem

The simplest form of Bayes theorem relates the joint probability $P(A,B)$of two events (A and B) in terms of marginal and conditional probabilities. This can be expressed as:

$$P(A,B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

By rearrangement, we obtain Bayes' theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

The different components of Bayes' theorem are often defined using the terminology:

$$\text{Posterior} = \frac{\text{Likelihood x Prior}}{\text{Marginal Likelihood}}$$

The Prior P(A) is the prior or marginal probability of the event A as it does not take into account any information about the event B. The posterior $P(A \mid B)$ is the conditional probability of A given B, and $P(B \mid A)$ is the conditional probability of B given A and is termed the likelihood. $P(B)$ is the marginal probability of B (also termed the 'evidence') and acts as a normalising constant.

Bayes' theorem is often further expressed as proportionality as the posterior probability is proportional to the product of the prior probability and the likelihood:

$$P(A \mid B) \propto L(B \mid A)P(A)$$

What is Likelihood?

In popular usage, the term 'likelihood' is often used interchangeably for probability, however in probability theory a separate technical definition exists. In an informal sense, likelihood works in reverse to probability. For example, if probability allows us to predict unknown outcomes based on known parameters, then the likelihood allows us to

determine unknown parameters based on known outcomes. The likelihood function remains a conditional probability function, but is considered a function of its second argument with the first argument held fixed.

Given B, we use conditional probability P(A|B) to reason about A.

Similarly, given A, we use the likelihood function L(B|A) to reason about B.

Appendix B – Deriving GP Predictive Equations

This section is to provide details of how the mean and variance predictions of the GP predictive posterior distribution can be derived, see Section (3.7.1) for the background. This derivation comes from Gibbs (1997).

The conditional probability to be solved in Section (3.7.1) was stated as:

$$P(\mathrm{t}_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1}) = \frac{P(\mathbf{t}_{N+1} \mid \mathbf{C}_N, \mathbf{X}_N, \mathbf{x}_{N+1})}{P(\mathbf{t}_N \mid \mathbf{C}_N, \mathbf{X}_N)} \tag{B.1}$$

where $\boldsymbol{D} = (\mathbf{X}_N, \mathbf{t}_N)$ is a set of N training data points, and $\mathbf{C}_N$, $\mathbf{C}_{N+1}$, $\mathbf{t}_{N+1}$, $\mathbf{x}_{N+1}$ where defined as before. Using the Gaussian process model the following was then derived:

$$P(\mathrm{t}_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1}) = \frac{Z_N}{Z_{N+1}} \exp\left(-\frac{1}{2} (\mathbf{t}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{t}_{N+1} - \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N) \right) \tag{B.2}$$

This distribution is Gaussian with respect to t_{N+1}, and in order to find the mean and variance a partitioned inverse form of $\mathbf{C}_{N+1}$ is defined, see Bartnett (1979).

The covariance matrix is defined as:

$$\mathbf{C}_{N+1} = \begin{bmatrix} [\mathbf{C}_N] & [\mathbf{k}_{N+1}] \\ [\mathbf{k}_{N+1}^T] & [\kappa] \end{bmatrix} \tag{B.3}$$

where the sub-matrix $\mathbf{k}_{N+1} = \left[\mathrm{C}(\mathbf{x}_{1,} \mathbf{x}_{\mathrm{N}+1}; \boldsymbol{\theta}), \ldots, \mathrm{C}(\mathbf{x}_{\mathrm{N},} \mathbf{x}_{\mathrm{N}+1}; \boldsymbol{\theta}) \right]$ is the vector of covariances between the new test point and existing training cases, and $\kappa = \mathrm{C}(\mathbf{x}_{\mathrm{N}+1,} \mathbf{x}_{\mathrm{N}+1}; \boldsymbol{\theta})$ is the variance of the individual test case.

The inverse is then defined as:

$$\mathbf{C}_{N+1}^{-1} \equiv \begin{bmatrix} \mathbf{M}_N & \mathbf{m}_{N+1} \\ \mathbf{m}_{N+1}^T & \mu \end{bmatrix} \tag{B.4}$$

Using the fact that $\mathbf{C}_{N+1}\mathbf{C}_{N+1}^{-1} = \mathbf{I}_{N+1}$, the following can be written:

$$\mathbf{C}_N\mathbf{M}_N + \mathbf{k}_{N+1}\mathbf{m}_{N+1}^T = \mathbf{I}_N \tag{B.5}$$

$$\mathbf{C}_N\mathbf{m}_{N+1} + m\mathbf{k}_{N+1} = \mathbf{0} \tag{B.6}$$

$$\mathbf{k}_{N+1}^T\mathbf{M}_N + \kappa\mathbf{m}_{N+1}^T = \mathbf{0}^T \tag{B.7}$$

$$\mathbf{k}_{N+1}^T\mathbf{m}_{N+1} + \kappa\mu = 1 \tag{B.8}$$

Through multiplying equation (B.5) by $\mathbf{C}_N^{-1}$, and then substituting the resultant expression into equation (B.7) we obtain:

$$\mathbf{m}_{N+1} = \frac{-\mathbf{C}_{N+1}^{-1}\mathbf{k}_{N+1}}{\left(\kappa - \mathbf{k}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{k}_{N+1}\right)} \tag{B.9}$$

Substituting this into equation (B.6) then gives us:

$$\mu = \left(\kappa - \mathbf{k}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{k}_{N+1}\right)^{-1} \tag{B.10}$$

With the remaining components of (B.4) can be defined as:

$$\mathbf{m}_{N+1} = -\mu\mathbf{C}_N^{-1}\mathbf{k}_{N+1} \tag{B.11}$$

$$\mathbf{M}_N = \mathbf{C}_N^{-1} + \frac{1}{\mu}\mathbf{m}_{N+1}\mathbf{m}_{N+1}^T \tag{B.12}$$

Returning to predictive posterior distribution in equation (B.2), we can write down the dependence of the exponent on t_{N+1} in terms of the elements of $\mathbf{C}_N^{-1}$ as defined in (B.4):

$$\mathbf{t}_{N+1}^T \mathbf{C}_{N+1}^{-1} \mathbf{t}_{N+1} - \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N = \mu t_{N+1}^2 + 2(\mu \mathbf{m}_{N+1}^T \mathbf{t}_N) t_{N+1} + \text{const.} \tag{B.13}$$

Using the expressions derived above for μ and $\mathbf{m}_{N+1}$, it is then straightforward to calculate the mean and variance of the Gaussian predictive distribution $P(t_{N+1} \mid \boldsymbol{D}, \mathbf{C}_N, \mathbf{x}_{N+1})$ as:

$$\hat{t}_{N+1} = \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{t}_N \tag{B.14}$$

$$\sigma_{N+1}^2 = \kappa - \mathbf{k}_{N+1}^T \mathbf{C}_N^{-1} \mathbf{k}_{N+1} \tag{B.15}$$

References

Abrahamsen, P. (1997). A Review of Gaussian Random Fields and Correlation Functions. *Technical Report 917,* Norwegian Computing Center, Oslo, Norway.

Adler, R. J. (1981). *The Geometry of Random Fields*. Wiley, Chichester.

Aizerman, M. A., Braverman, E. M., and Rozoner, L. I. (1964) Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control, 25, 821-837.*

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716-723.

Alligood, K. T. (1997). *Chaos: an introduction to dynamical systems*. Springer-Verlag, New York.

Allgöwer F., Zheng A., (eds) (2000) Nonlinear Model Predictive Control, *Progress in system and control theory*, Vol. 26, Birkhäuser Verlag, Basel.

Ažman, K., and Kocijan, J. (2007). Application of Gaussian processes for black-box modelling of biosystems. *ISA Transactions*, 46(4), pages 443-457.

Babuška, R., and Verbruggen, H. (2003). Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control* 27, 73-85.

Barnes, C., Brown, S., Flake, G., Jones, R., O'Rourke, M., and Lee, Y. C. (1991). Applications of neural networks to process control and modelling. In *Artificial Neural Networks, Proceedings of 1991 Internat. Conf. Artif. Neur. Nets*, volume 1, pp 321-326.

Bartnett S., (1979). *Matrix Methods for Engineers and Scientists*, McGraw-Hill.

Bezdek. J. C. (1981). *Pattern Recognition with Fuzzy Objective Function.* Plenum Press, New York.

Billingsey, P., (1986). *Probability and Measure*, Second edition, John Wiley & Sons, New York.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford.

Box, G. E. P., and Taio, G. C. (1973). *Bayesian inference in statistical analysis*. Addison-Wesley

Braham, R. (1998). Incorporation of Long-Range Feedback in Neural Networks Under Stability Conditions. Neural Networks, 11, (1)141-144.

Breiman**,** L., Friedman, J., Olshen, R. and Stone, C. **(**1984**),** *Classification and Regression Trees*. Chapman & Hall, New York.

Broomhead, D. S.,and Lowe, D. (1988). Multivariable functional interopolation and adaptive networks. *Complex Systems*, 2:321-355.

Brown, M., and Harris, C. J. (1994). *Neurofuzzy Adaptive Modelling and Control*. Prentice Hall, New York.

Chan, L. (2003). *Time-Series Prediction Using Evolutionary Lateral-Delay Neural Networks*. PhD Thesis, University of Glasgow.

Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). Orthogonal least-squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2).

Chong, G. and Li, Y. (2000). Trajectory Controller Network and Its Design Automation Through Evolutionary Computing. *Lecture Notes in Computer Science: Real-World Applications of Evolutionary Computation*, 1803, Eds.: Cagnoni, S. et al., Springer-Verlag, 139-146.

Cleveland, W. S., Devlin, S. J., and Grosse, E. (1996). Regression by local fitting: Methods, properties and computational algorithms. *Journal of Econometrics*, 37:87-114.

Cohn, D., Altas, L., and Ladner, R. (1990). Training connectionist networks with queries and selective sampling. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann Publishers, San Mateo, CA.

Cohn, D. (1994). Network exploration using optimal experiment design. In Cowan, J.D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, San Francisco, CA. Morgan Kaufmann Publishers.

Cohn, D. A., Ghahramani, Z and Jordan M. I., (1997). Active Learning with Mixture Models. In *Multiple Model Approaches to Modelling and Control*, edited by R Murray-Smith and T. A. Johansen.

Congdon, P. (2003) *Applied Bayesian Modelling*. Wiley, New York:

Cressie, N. (1993), *Statistics for Spatial Data*, Wiley.

Csat´o, L. and Opper, M. (2002). *Sparse On-Line Gaussian Processes. Neural Computation*, 14(3):641–668.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2, 303-314.

Demuth, H., and Beale, M. (1998). *MATLAB Neural Network Toolbox User's Guide, Version 3.0*. The MATHWORKS Inc., Natick, MA.

Denison D. G. T., Holmes, D. C., Mallick, B. K., and Smith, A. F. M. (2002).

Bayesian Methods for Nonlinear Classification and Regression. Wiley.

Doob, J. L. (1953) *Stochastic Processes*, John Wiley & Sons, New York.

Dorf, R. C., and Bishop, R. H. (2004). *Modern Control Systems* Pearson Education; (Tenth edition).

Draper, N. R., and Smith, H. (1998). *Applied Regression Analysis*. Probability and Mathematical Statistics. John Wiley & Sons, New York. (Third Edition).

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support Vector Regression Machines. *Advances in Neural Information Processing Systems* 9, NIPS 1996, 155-161, MIT Press.

Duane, S., Kennedy, A. D., Pendleton, B.J., and Roweth, D. (1987) Hybrid Monte Carlo, *Physics Letters B*, vol. 195, pp. 216-222.

Eubank, R. L. (1999). *Nonparametric Regression and Spline Smoothing*. (Second Edition) Marcel Dekker.

Fletcher, R. (1987) *Practical Methods for optimization*. John Wiley and Sons inc., Second edition.

Fletcher, R. (1993). An Overview of Unconstrained Optimization. *NATO ASI on Algorithms for Continuous Optimization*, Il Ciocco, Italy.

Friedman, J. H. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics.* 19:1-141.

Gelman, A., Carlin, J. B., Stern, H. S, and Rubin, D. B. (2004) *Bayesian Data Analysis*, Second edition. Chapman and Hall.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the Bias/Variance Dilemma. *Neural Computation* 4, 1-58.

Gibbs, M. N. (1997). *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, Department of Physics, University of Cambridge.

Gibbs, M. N. and MacKay, D. J. C. (1997). *Efficient Implementation of Gaussian Processes*. Unpublished manuscript. Cavendish Laboratory, Cambridge, UK.

Gilks, W. R., Richardson, S., and Spiegelhalter, D.J. (1996) *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

Girard, A., Rasmussen, C. E., and Murray-Smith, R. (2002). Gaussian process priors with uncertain inputs: multiple-step-ahead prediction. Technical Report DCS TR-2002-119, University of Glasgow, Glasgow.

Girard, A. (2004). *Approximate methods for propagation of uncertainty with Gaussian process models*. PhD thesis, University of Glasgow, Glasgow.

Godfrey, K. R. (1993). *Perturbation Signals for System Identification*. Prentice Hall International, New York.

Goldberg, D. (1989). *Genetic Algorithms in Searching, Optimisation and Machine Learning*. Addison Wesley, Reading, MA.

Gollee, H. (1994). *A Non-Linear Approach to Modelling and Control of Electrically Stimulated Skeletal Muscle*. PhD Thesis, University of Glasgow.

Gollub, J. P., Baker, G. L. (1996). *Chaotic Dynamics*. Cambridge University Press

Golub, G. H. and Van Loan, C, F. (1987). *Matrix Computations*. Mathematical Sciences. The Johns Hopkins University Press, Baltimore.

Gong, M. and Murray-Smith, D. J. (1998) A Practical Exercise in Simulation Model Validation. *Mathematical and Computer Modelling of Dynamical Systems*. Vol. 4, No. 1, pp. 100-117.

Goodwin, G. C. and Payne, R. L. (1977*), Dynamic System Identification: Experiment Design and Data Analysis.* Academic Press, New York.

Goodwin, G. C. (1987). Experiment design for system identification. In *Encyclopedia of Systems and Control* (M. Singh, ed.). Pergamon Press, Oxford.

Gray, G., Murray-Smith, R., Thompson, K., and D. J. Murray-Smith, D. J. (2003) Tutorial example of Gaussian process prior modelling applied to twin-tank system. Technical Report DCS TR-2003-151, University of Glasgow, Glasgow.

Gregorčič, G., and Lightbody, G. (2002) Gaussian processes for modelling of dynamic non-linear systems. In *Proceedings of Irish Signals and Systems Conference, Cork*, pages 141-147, Cork.

Gregorčič, G., and Lightbody, G. (2003) From multiple model networks to the Gaussian processes prior model. In *Proceedings of IFAC ICONS conference*, pages 149-154, Faro.

Gregorčič G., and Lightbody, G. (2007) Local model identification with Gaussian processes. *IEEE Transactions on neural networks*, 18(5), pages 1404-1423.

Gregorčič, G., and Lightbody, G. (2008). Nonlinear system identification: From multiple-model networks to Gaussian processes. In *Engineering Applications of Artificial Intelligence* 21:1035-1055.

Gull S. F. (1988). Bayesian inductive inference and maximum entropy. In *Maximum Entropy and Bayesian Methods in Science and Engineering, vol. 1; Foundations,* ed. By G. Erickson and C. Smith. Pp. 53-71. Kluwer.

Hardle, W. (1990). *Applied Nonparametric Regression*. Cambridge

Haykin, S. (1994). *Neural Networks. A Comprehensive Foundation*. Macmillan, New York.

Henson, M. (1998) Nonlinear Model Predictive Control: Current Status and Future Directions. *Computers and Chemical Engineering*, 23:187-202.

Hinton, G. E., and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing: Explorations in Microstructure of Cognition*. (D. E. Rumelhart and J.L. McClelland, eds.), Cambridge, MA: MIT Press.

Hinton, G. E. and van Camp, D. (1993) Keeping neural networks simple by minimizing the description length of the weights. *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, Santa Cruz 1992, pp. 5-13.

Holland, H. J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the U.S.A.* 79, 2554-2558.

Hornik, K., Stinchombe, M., and White, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Computation*, 2:359-366.

Jang, J.-S. R., and Sun, C.-T. (1995). Neuro-fuzzy modeling and control. *Proceedings of the IEEE 83*, 378-406.

Johansen, T. A., and Foss, B. A. (1992). A NARMAX model representation for adaptive control based on local models. *Modeling Identification and Control* 13, 25-39.

Johansen, T. A., and Foss, B. A (1993). Constructing NARMAX models using ARMAX models. *International Journal of Control*. 58(5), 1125-1153.

Johansen, T. A., and Foss, B. A. (1995a). Identification of non-linear system structure and parameters using regime decomoposition. *Automatica* 31, 321-326.

Johansen, T. A., and Foss, B. A. (1995b). Empirical modelling of a heat transfer process using local models and interpolation. In: Proc. Amer. Control Conference. Vol 5. pp. 3654-3658.

Johansen, T. A. (1997). On Tikhonov regularization, bias and variance in nonlinear system identification. *Automatica*, 33(3): 441-446.

Johansen, T. A., and Foss, B. A. (1997). Operating regime based process modelling and identification. *Computers and Chemical Engineering* 21, 159-176.

Juang, J. (1994). *Applied System Identification*. Prentice Hall.

Juričić, D. J., and Kocijan, J. (2006). Fault detection based on Gaussian process model. In I. Troch and F. Breitenecker, editors, *Proceedings of the 5th Vienna Symposium on Mathematical Modeling - MathMod*, Vienna, 2006.

Jutton, C., and Fambon, O. (1995). Pruning methods: A review. In *Proc. 3rd European Symposium on Artificial Neural Networks*. Brussels. pp. 129-140.

Kaelbling, L. P.; Littman, M. L.; and. Moore, A. W. (1996). "Reinforcement Learning: A Survey". *Journal of Artificial Intelligence Research* 4: 237–285.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671-680.

Kocijan, J., Banko, B., Likar, B., Girard, A., Murray-Smith, R., and Rasmussen, C. E. (2003a). A case based comparison of identification with neural networks and Gaussian process models. In *Proceedings of IFAC ICONS conference*, volume 1, pages 137-142, Faro.

Kocijan, J., Girard, A., Banko, B., and Murray-Smith. R. (2003b) Dynamic systems identification with Gaussian processes. In I. Troch and F. Breitenecker, editors, *Proceedings of 4th IMACS Symposium on Mathematical Modelling (MathMod)*, pages 776-784, Vienna.

Kocijan, J., Girard, A., and Leith, D. J. (2003c). Incorporating linear local models in Gaussian process model. Technical Report DP-8895, Institut Jožef Stefan, Ljubljana.

Kocijan, J., and Leith, D. J. (2004) Derivative observations used in predictive control. In *Proceedings of Melecon 2004*, volume 1, pages 379-382, Dubrovnik.

Kocijan, J., Murray-Smith, R., Rasmussen, C. E., and Girard, A. (2004) Gaussian process model based predictive control. In *Proceedings of 4th American Control Conference*, pages 2214-2218, Boston, MA.

Kocijan, J.,and Ažman, K. (2007). Gaussian Process Model Identification: A Process Engineering Case Study. In *Proceedings of the 16th International Conference on Systems Science*, Vol. 1, pages 418 - 427, Wroclaw.

Kocijan, J., and B. Likar, B. (2007) Gas-Liquid Separator Modelling and Simulation with Gaussian Process Models. In *Proceedings of the 6th EUROSIM Congress on Modelling and Simulation*

Kohonen T. (1990). The self-organizing map. Proceeding of the IEEE. 78:1464-1480.

Laarhoven, P. J. M. Van, and Aarts, E. H. L. (1987). *Simulation Annealing: Theory and Applications*. Dordrecht, Lancaster.

Lauritzen, S. L. (1981). Time Series Analysis in 1880: A Discussion of Contributions Made by T. N. Thiele. *International Statistical Review*, 49:319–333.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press.

Lee, P. M. (2004). *Bayesian Statistics – an introduction*. Third Edition. Hodder Arnold.

Leith, D. J., and Leithhead, W. E. (1999). Analytic framework for blended multiple model systems using linear local models. *International Journal of Control*. 72 (7/8), 605-619.

Leith, D. J., Murray-Smith, R., and Leithead, W. E. (2000). Nonlinear structure identification: A Gaussian process/velocity-based approach. In *Proceedings of the UKACC Control Conference*, Cambridge.

Leith, D. J., Leithead, W. E., Solak, E., and Murray-Smith, R. (2002) Divide and conquer identification using Gaussian processes. In *Proceedings of the 41st Conference on Decision and Control*, pages 624-629, Las Vegas, AZ.

Leithead, W. E., Leith, D. J., and Murray-Smith, R. A. (2000). Gaussian Process Prior/Velocity-based Framework for Nonlinear Modelling and Control, In *Irish Signals and Systems Conference*, Dublin.

Leontartis, I. J., and Billings, S. A. (1987) Model selection and Validation Methods for Non-Linear Systems. *International Journal of Control*, 45 311-341.

Likar, B., and. Kocijan, J. (2007). Predictive control of a gas-liquid separation plant based on a Gaussian process model. *Computers and Chemical Engineering*, 31(3), pages 142-152.

Ljung, L. (1999). *System Identification – Theory for the User*. Prentice Hall.

Luo, Z. and Wahba, G. (1997). Hybrid Adaptive Splines. J. *Amer. Statist. Assoc.*, 92:107–116.

MacKay D. J. C. (1991) *Bayesian Methods for Adaptive Models*, Ph.D thesis, California Institute of Technology.

MacKay, D. J. C. (1992a). Bayesian Interpolation. *Neural Computation*, 4(3):415–447.

MacKay, D. J. C. (1992b). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472.

Mackay, D. J. C. (1992c). The evidence framework applied to classification networks. *Neural Computation* 4 (5):698-714.

Mackay, D. J. C. (1994) Bayesian non-linear modelling for the energy prediction competition, ASHRAE Transactions Vol. 100, Pt. 2, PP. 1053-1062.
MacKay, D. J. C. (1997). *Gaussian Processes - A Replacement for Supervised Neural Networks?* Lecture notes for a tutorial at NIPS 1997.

MacKay, D J. C. (1998a) Intro to Monte Carlo methods. In Michael I. Jordan, editor**,** *Learning in Graphical Models*. The MIT Press, Cambridge, Massachusetts.

MacKay, D. J. C. (1998b). Introduction to Gaussian Processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning*. Springer-Verlag.

MacKay, D. J. C. (2003*). Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK.

Matheron, G. (1963). Principles of geostatistics, *Economic Geology* 58, 1246-1266.

McCulloch, W. S., and Pitts, W. (1943). A Logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115-133.

McLoone, S. (2000). *Nonlinear identification using local model networks*. PhD Thesis, Queens University of Belfast.

McLoone, S. E., Irwin, G. W., and McLoone, S. F. (2001). Constructing networks of continuous-time velocity-based models. IEE Proceedings – Control Theory and Applications 148o (5), 397-405.

Mehra, R. K. (1981). Choice of input signals. In *Trends and Progress in Systems Identification*. (P.Eykhoff, ed.). Pergamon Press, Elmsford, N.Y.

Moody, J., and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*. 1(2):281-294.

Murray-Smith, R. (1992). Local Model Networks and Local Learning. In *Fuzzy Duisburg '94*, pages 404-409.

Murray-Smith, R. (1994). *A Local Model Network Approach to Nonlinear Modelling*. PhD Thesis. Department of Computer Science, University of Strathclyde, Glasgow, Scotland.

Murray-Smith, R., and Gollee, H. (1994). A constructive learning algorithm for local model networks. In *IEEE Workshop on Computer-Intensive Methods in Control and Signal Processing*, Prague, Czech Republic, pp. 21-29.

Murray-Smith, R., and Johansen, T. A. (1995). Local learning in local model networks. In *4th IEE Intern. Conf. on Artificial Neural Networks*. pp. 40-46.

Murray-Smith, R., and Hunt, K. J. (1995). Local Model architectures for nonlinear modelling and control. In: Hunt, K. J., Irwin, G.R., Warwick, K. (editors) *Neural Network Engineering in Dynamic Control Systems, Advances in Industrial Control*. Springer, Berlin, pp. 61-82.

Murray-Smith, R. and Johanson, T. A. (1997). *Multiple Model Approaches to Modelling and Control.* Taylor and Francis.

Murray-Smith, R., Johansen, T. A., Shorten, R. (1999). On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. In *Proceedings of European Control Conference*, BA-14, Karslruhe.

Murray-Smith, R., and Girard, A. (2001) Gaussian process priors with ARMA noise models. In *Proceedings of Irish Signals and Systems Conference*, pages 147-152, Maynooth.

Murray-Smith, R., and Sbarbaro, D. (2002). Nonlinear adaptive control using nonparametric Gaussian process prior models. In *Proceedings of IFAC 15th World Congress*, Barcelona.

Murray-Smith, R., Shorten, R., and Leith, D. (2002). Nonparametric models of dynamic systems. In C. Cowans, editor, *Proceedings of IEE Workshop on Nonlinear and Non-Gaussian signal processing - N2SP,* Peebles, UK.

Murray-Smith, R., Sbarbaro, D., Rasmussen, C. E., and Girard, A. (2003). Adaptive, cautious, predictive control with Gaussian process priors. In *Proceedings of 13th IFAC Symposium on System Identification*, pages 1195-1200, Rotterdam.

Neal, R. M. (1992). Bayesian training of backpropagation networks by the hybrid Monte Carlo Method. Technical Report CRG-TR-92-1, Dept. of Computer Science, University of Toronto.

Neal, R. M. (1993) Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer, New York. Lecture Notes in Statistics 118.

Neal, R. M. (1997). Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of Statistics, University of Toronto.

Nelles, O., and Isermann, R. (1996). A new technique for determination of hidden layer parameters in RBF networks. In *IFAC World Congress*, pages 453-457, San Francisco, USA.

Nelles, O., Fink, A., and Isermann, R. (2000). Local linear model trees (lolimot) toolbox for nonlinear system identification. In *12th IFAC Symposium on System Identification (SYSID)*, Santa Barbara, USA. Pp. 845-850.

Nelles, O. (2001) *Nonlinear System Identification*. Springer.

Nise, N. S. (2003). *Control Systems Engineering*. John Wiley & Sons; (Fourth Edition).

O'Hagan, A. (1978). Curve Fitting and Optimal Design for Prediction. *Journal of the Royal Statistical Society* B, 40:1–42. (with discussion). Pp. 28, 30, 94.

Paciorek C.J., (2003). *Nonstationary Gaussian Processes for Regression and Spatial Modelling*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvannia.

Paciorek, C. and Schervish, M. J. (2004). Nonstationary Covariance Functions for Gaussian Process Regression. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems* 16. MIT Press.

Pantaleón-Prieto, C. J., de María. F. D., and Figueiras-Vidal, A. (1993). On training RBF networks. In *Neural Networks and their Industrial and Cognitive Applications*, Nimes, France, pp. 279-288.

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York. Third Edition.

Park, J., and Sandberg, I. W. (1991).Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246-257.

Pfeiffer, B-M, and Isermann, R. (1994). Criteria for successful applications of fuzzy control. *Engineering Applications of Artificial Intelligence*, 7(3):245-253.

Plutowski, M. (1994). *Selecting Training Exemplars for Neural Network Learning*. Ph.D. Thesis, University of California, San Diego, USA.

Poggio, T. and Girosi, F. (1990). Networks for Approximation and Learning. Proceedings of IEEE, 78:1481–1497.

Powell, M. J. D. (1985). Radial basis functions for multivariable interpolation: A review. In *IMA Conference on Algorithms for the Approximation of Functions and Data*, Pages 143-167, Shrivenham, UK.

Press, S. J. (1989). *Bayesian Statistics: Principles, Models and Applications*. New York, Wiley.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes* in C. Cambridge University Press, Second edition.

Qin S. J., and Badgwell T. A., (2000). An overview of nonlinear model predictive control applications. In Allgower F., Zheng A. (eds.), *Nonlinear model predictive control*, Birkhauser Verlag, 369-392.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939-1959.

Quinonero-Candela, J., Rasmussen, C. E., and Williams, C. K. I. (2007). Approximation Methods for Gaussian Process Regression. In Leon Bottou, Olivier Chapelle, Dennis DeCoste and Jason Weston, editors, *Large Scale Learning Machines*, pages 203-223, Cambridge, MA. MIT Press.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, Dept. of Computer Science, University of Toronto.

Rasmussen, C. E. and Ghahramani, Z. (2002). Infinite Mixtures of Gaussian Process Experts. In Diettrich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems* 14. MIT Press.

Rasmussen, C. E. and Williams, C. K. I (2006) *Gaussian Processes for Machine Learning*. MIT Press.

Reed, R. (1999). Pruning algorithms – a survey. *IEEE Transactions on Neural Networks* 4 (5), 740-747.

Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M. (1983). *Engineering Optimization – Methods and Applications*. John Wiley & Sons, London.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8. MIT Press, Cambridge

Sampson, P. D., and Guttorp, P. (1992). Nonparametric Estimation of Nonstationary Covariance Structure. *Journal of American Statistical Association*, 87:108-119.

Sbarbaro, D., and Murray-Smith, R. (2005). *Switching and Learning in Feedback Systems*, volume 3355 of *Lecture Notes in Computer Science*, chapter Self-tuning control of nonlinear systems using Gaussian process prior models, pages 140-157. Springer, Heidelberg.

Scales, L. E. (1985). *Introduction to Non-Linear Optimization*. Computer and Science Series. Macmillan, London.

Seeger, M. (2003). *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations.* PhD thesis, School of Informatics, University of Edinburgh.

Seeger, M., Williams, C. K. I., and Lawrence, N. (2003). Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In Bishop, C. and Frey, B. J., editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.
Seeger, M. (2004). Gaussian Processes for Machine Learning. *International Journal of Neural Systems*. 14:69-106.

Shi, J. Q., Murray-Smith, R., and Titterington, D. M. (2003). Bayesian Regression and Classification Using Mixtures of Gaussian Processes. *International Journal of Adaptive Control and Signal Processing*. 17:1-16.

Shorten, R., and Murray-Smith, R. (1994). On Normalising Basis Function networks. In *4th Irish Neural Networks Conference*. University College, Dublin.

Shorten, R., Murray-Smith, R., and Bjørgan. (1999). On the interpretation of local models in blended multiple model structures. *International Journal of Control*, vol. 72, no. 7/8 pp 620-628.

Sjöberg J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.-Y., Hjalmarsson, H., and Juditsky, A. (1995) Non-linear Black-box modeling in system identification: A unified overview. *Automatica*, 31(12):1691-1724.

Skilling, J. (1993). Bayesian numerical analysis, in W. T. Grandy, Jr. and P. Milonni (eds), *Physics and Probability*, C.U.P., Cambridge.

Smith, A. F. M. and Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion). *Journal of the Royal Statistical Society* B55, 3-102.

Smola, A. J. and Bartlett, P. L. (2001). Sparse Greedy Gaussian Process Regression. In Leen, T. K., Diettrich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems* 13, pages 619–625. MIT Press.

Snelson, E., Rasmussen, C. E., and Ghahramani. Z. (2004) Warped Gaussian processes. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems* 16. MIT Press, Cambridge, MA, 2004.

Snelson, E., and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, eds, *Advances in Neural Information Processing Systems* 18, Cambridge, MA. MIT Press.

Snelson, E. L. (2007). *Flexible and efficient Gaussian process models for machine learning*. PhD Thesis, University of London.

Söderström, T. and Stoica, P. (1989). *System Identification*. Prentice Hall International, London.

Solak, E., Murray-Smith, R., Leithead, W. E., Leith, D., and Rasmussen, C. E. (2003). Derivative Observations in Gaussian Process Models of Dynamic Systems. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems* 15, pages 1033–1040. MIT Press.

Stein, M. L. (1999). *Interpolation of Spatial Data*. Springer-Verlag, New York.

Sutton, R. S.; and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Takagi, T., and Sugeno, M. (1985). Fuzzy identification of systems and its application to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(1):116-132.

Takens, F. (1981) Detecting strange attractors in turbulence. Lecture Notes in Mathematics 898, Springer-Verlag, Berlin.

Thodberg, H. H. (1996). A review of Bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks*, vol. 7. pp. 56-72.

Thompson, K., and Murray-Smith, D. J. (2006) Implementation of Gaussian process models for nonlinear system identification. In I. Troch and F. Breitenecker, editors, *Proceedings of the 5th Vienna Symposium on Mathematical Modeling - MathMod*, Vienna.

Thrun, S. B. (1992). The role of Exploration in Learning Control. In *Handbook of Intelligent Control: Neural Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold.

Tikhonov, A. N., and Arsenin, V. Y. (1977). *Solutions of Ill-Posed Problems*. Wiley, New York.

Unbehauen, H. and Rao, G. P. (1987). *Identification of Continuous-Time Systems*. North-Holland, Amsterdam.

Van den Hof, P. M. J. (1997). Closed-loop issues in system identification. In *IFAC Symposium on System Identification*, pages 1651 – 1664, Fukuoka, Japan.

Vanderplaats, G. N. (1984). *Numerical Optimization Techniques for Engineering Design*. Series in Mechanical Engineering. McGraw-Hill, New York.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer Verlag, New York.

Wahba, G. (1990). *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia, PA. CBMS-NSF Regional Conference series in applied mathematics.

Wahba, G., Johnson, D. R., Gao, F., and Gong, J. (1995). Adaptive Tuning of Numerical Weather Prediction Models: Randomized CCV in Three-and Four-Dimentional Data Assimilation. *Monthly Weather Review*, 123:3358-3369.

Wang, J. M., Fleet,D. J., and Hertzmann, A. (2005). Gaussian process dynamical models. In *Advances in Neural Information Processing Systems*, volume 18, pages 1441-1448. MIT Press.

Werntges, H. W. (1993). Partitions of unity improve neural function approximators. In *IEEE International Conference on Neural Networks (ICNN)*, volume 2, pages 914-918, San Francisco, USA.

Wettschereck, D., and Dietterich, T. (1992). Improving the performance of radial basis function networks by learning center locations. In J. E. Moody, S. J. Hanson, and R. P Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 1133-1140. Morgan Kaufmann, San Mateo.

Wiener, N. (1948). *Cybernetics*, Wiley.

Williams, C. K. I., and Rasmussen, C. E. (1996). Gaussian processes for regression, in D. S. Touretzky, M. C. Mozer and M. E. Hasselmo, (eds), *Advances in Neural Information Processing Systems* 8, MIT Press.

Williams, C. K. I. (1998). Computation with Infinite Neural Networks. *Neural Computation*, 10(5):1203–1216.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In Leen, T. K., Diettrich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems* 13, pages 682–688. MIT Press.

Williams, C. K. I., Rasmussen, C. E., Schwaighofer, A., and Tresp, V. (2002). Observations on the Nyström Method for Gaussian Process Prediction. Technical report, University of Edinburgh.

Wolberg, J. (2005). *Data Analysis Using the Least-Squares Method*. Springer-Verlag, New York.

Wothke, W. (1993). Nonpositive definite matrices in structural modelling. In K. A. Bollen & J. S. Long (Eds.), *Testing structural equation models*. Newbury Park, CA: Sage.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338-353.