Approximate Methods for Propagation of Uncertainty with Gaussian Process Models

Agathe Girard

A thesis submitted to the University of Glasgow for the degree of Doctor of Philosophy

October 2004

© 2004, Agathe Girard

Abstract

This thesis presents extensions of the Gaussian Process (GP) model, based on approximate methods allowing the model to deal with input uncertainty. Zero-mean GPs with Gaussian covariance function are of particular interest, as they allow to carry out many derivations exactly, as well as having been shown to have modelling abilities and predictive performance comparable to that of neural networks (Rasmussen, 1996a). With this model, given observed data and a new input, making a prediction corresponds to computing the (Gaussian) predictive distribution of the associated output, whose mean can be used as an estimate. This way, the predictive variance provides error-bars or confidence intervals on this estimate: It quantifies the model's degree of belief in its 'best guess'. Using the knowledge of the predictive variance in an informative manner is at the centre of this thesis, as the problems of how to propagate it in the model, how to account for it when derivative observations are available, and how to derive a control law with a cautious behaviour are addressed.

The task of making a prediction when the new input presented to the model is noisy is introduced. Assuming a normally distributed input, only the mean and variance of the corresponding non-Gaussian predictive distribution are computed (*Gaussian approximation*). Depending on the parametric form of the covariance function of the process, exact or approximate moments are derived. These results are then used for the multiple-step-ahead iterative forecasting of nonlinear dynamic systems, with propagation of the uncertainty. Within a nonlinear auto-regressive representation of the system, modelled by a GP, a one-step-ahead model is iterated up to the desired horizon. At each time-step, the uncertainty induced by each successive prediction is propagated, that is, the whole predictive distribution of the output just predicted is fed back into the state for the next time-step. Not only are the predictive variances of each delayed output accounted for, but also the cross-covariances between them. The approach is illustrated on the simulated Mackey-Glass chaotic time-series, as well as on two real-life dynamic processes, a gas-liquid separator and a pH neutralisation process.

The emphasis is on the use of Gaussian Processes for modelling nonlinear dynamic systems. GPs have not yet gained widespread popularity among the engineering community. It is well known that the modelling of such systems is in practice rendered difficult by the fact that most available data lies around equilibrium regions, and very few points in transient areas, and a common approach has been to consider linearisations around those equilibrium points. Derivative observations can be elegantly integrated in a GP model where function observations are already available, as shown in (Solak et al., 2003). As well as being in accord with engineering practice, derivative observations can potentially reduce the computational burden usually associated with GPs (typically, a linear region can be summarised by one derivative observation, instead of many function observations). For this *mixed* training set, the explicit expressions of the predictive mean and variance of a function output corresponding to a noise-free and to a noisy input are then derived, the latter being tackled within the *Gaussian approximation*.

The other field where GPs can present an advantage other over models is in the control of nonlinear dynamic systems. Commonly, researchers have used nonlinear parametric models and have adopted the *certainty equivalence principle* when deriving the control law, whereby the model's predictions are used as if they were the true system's outputs. Deriving controllers with 'cautious' and 'probing' features is difficult and has been the scope of much work in the control community. The propagation of uncertainty method is applied for a cautious controller, where the cautiousness is accounted for in a cost function that does not disregard the variance associated with the model's estimate.

Declaration

I declare that this thesis was composed by myself and has not, nor any similar dissertation, been submitted in any other application for a degree. It describes the work carried out between September 2000 and March 2004 in the Department of Computing Science at the University of Glasgow, under the supervision of Roderick Murray-Smith. The work contained therein is my own, as well as the result of collaboration with other researchers, as stated in the Introduction chapter, Section 1.3.

Acknowledgements

I've been lucky to do this PhD within the Multi-Agent Control Research Training Network, as it promoted and facilitated my exchange with other researchers, and also gave me the chance to meet Joaquin, Kai, Bojan, the two Gregor(s), whom have all become friends. I'm glad you took me on board Rod, that you gave me the chance to do this. As you know, it's been sometimes quite hard, and I might not have gone through it all if you hadn't been so patient, open-minded and understanding. It's been good working with you, as well as with Joaquin, Carl and Jus. Work-wise, I am also deeply grateful to Prof. Mike Titterington, for his comments, suggestions and explanations. There is also Jian Qing, who helped me shaping my ideas, John who patiently listened to me, and Ernest, whom with I had the best discussions, in my first year in Glasgow. And there is Prof. Bill Leithead, whom I deeply respect. I greatly value the few discussions we had, and how they can range from anything to everything!

I have come to really appreciate Glasgow, a beautiful city with lots to offer, chips & cheese and rain included. And it's here that I met Robert, Ali and Louise, Philippe, Matthew, Dagmar, Marigo, Henrik and Rolf, who are all so special and important to me. There is also everybody at the soup kitchen, and in particular Isa, Craig, Fionna and Jane, and the people in the street: A big thank you to all for what we share. Always present, Anais, Axel and Benjamin, who supported me when I needed it most. These last four years have taught me a lot. I have become more aware of what I don't know. My understanding and views have changed. Whatever I do next, I hope I'll be able to say this in the next four years! I welcome changes and the nonlinear dynamic aspect of our lives.

Scientific theories are not discoveries of the laws of nature but rather inventions of the human mind.(A. Papoulis)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

I present various extensions of the Gaussian Process model for use in engineering applications. Central to this work is the prediction of a system output when presented with a new noisy (or random) input. This problem is addressed using an analytical approximation that consists of computing only the mean and variance of the corresponding predictive distribution (Gaussian approximation). These results are then applied to the iterative multiple-step-ahead prediction of nonlinear time-series, showing how the uncertainty induced by each successive prediction can be propagated ahead in time. Also presented is a methodology for using the propagation of uncertainty framework in the control of nonlinear dynamic systems, and in the case where derivative observations of the system are available.

1.1 Motivation and background

Mathematical modelling seeks to describe, or *model*, a given phenomenon (system) from observations or measurements of it. It might be the weather we wish to forecast, the dynamics of an aeroplane we want to simulate, or getting more understanding about how our minds work. There are many possible levels of description, but the choice will be in general dictated by our knowledge of the phenomenon, our goals, tools and computational resources.

In this thesis, the emphasis is on systems whose responses (outputs) correspond to given causes (inputs). In this supervised learning setting, modelling then corresponds to finding the underlying data generative mechanism, that is, the mapping from the input ('causal') space to the output ('observational') space. To do so, empirical models are here considered, sometimes called black-box models, because they do not require a detailed understanding of the process or system under study, as opposed to first-principles models. These functional mappings can conveniently be divided into two classes: parametric and nonparametric. Whereas parametric models assume a given form of the mapping, no parametric form is fixed in advance for the nonparametric class, allowing a greater flexibility (we refer to (Hastie et al., 2001; Bishop, 1995) as general textbooks, and to (Gershenfeld, 1999) for a more broad-audience introduction to mathematical modelling).

The GP alternative

Gaussian Processes (GPs) came to the attention of the machine learning community in the nineties, after Neal showed that, in their Bayesian treatment, neural networks with one hidden layer converged to a Gaussian Process as the number of hidden neurons tends to infinity, given suitable priors for the weights (Neal, 1994; Neal, 1995). They became increasingly popular after Rasmussen carried out a thorough empirical comparison of the GP with more widely used models, showing that, in many instances, the GP outperforms them (Rasmussen, 1996a). Since then, a great deal of research has been done, dealing with diverse aspects of the model. Thorough introductions of the model and its relation to Bayesian kernel models can be found in (Williams, 2002; Mackay, 2003; Seeger, 2003).

The probabilistic nature of the GP model allows to directly define the space of admissible functions relating inputs to outputs, by simply specifying the mean and covariance functions of the process. In this framework, the observations correspond to an (incomplete) realisation of the process. Although a parametric form is actually pre-supposed (albeit not on the functional itself but on the family from which it can come from), the model is very powerful and flexible, with the Gaussian assumption keeping most derivations analytically tractable and simple.

Predictions and uncertainties

Most commonly, a model will be used to generalise from measurements, that is, to make predictions about new observations. But predicting the system's response to a given new input, alone, is not satisfactory. In general, we do not only wish to have an estimate of a quantity, we also need to quantify our degree of belief in that estimate (how (un)certain are we in our prediction? how much can we trust it?).

With the GP model, the learning task (teaching through observations) corresponds to tuning the parameters of the covariance function of the process to the data at hand, which, here, is simply done in a Maximum-Likelihood-like manner. With this model, given a new input, and conditional on past observations, we naturally obtain a prediction and the uncertainty attached to it, respectively given by the mean and variance of the predictive distribution of the future output. This distribution is Gaussian, readily obtained using the definition of conditional probabilities, as a consequence of the GP assumption.

This is not a perfect world...

What if the inputs are 'uncertain'? Noisy inputs can arise in different situations, for instance when using faulty sensors, such that the system senses the inputs imperfectly. Dealing with noisy inputs is a well known difficult task which is the scope of much research. In the statistics literature, models dealing with noisy regressors are known as errors-in-variables models (Kendall and Stuart, 1958), and the problem has been tackled using deterministic (Freedman et al., 2004) and Bayesian (Dellaportas and Stephens, 1995; Snoussi et al., 2002) approaches to recover the unknowns. In the machine learning community, it has been shown that closed-form solutions exist for Gaussian basis function networks (Ahmad and Tresp, 1993), and mixture models have been proved to deal naturally with missing features (Tresp et al., 1994; Ghahramani and Jordan, 1994a). In most cases, the major difficulty stems from the unknown probability distribution of the input, which has to be either assumed or learnt from the data. In this thesis, the task of dealing with noisy inputs is handled within an analytical approximation, assuming normally distributed inputs.

Accounting for time

Even though it has not yet been mentioned, dynamic systems, whose properties and behaviour vary with time, can be modelled in a similar way, provided a suitable representation and a few hypotheses (Takens, 1981). In time-series analysis (Box et al., 1994), a common approach is to assume a (possibly nonlinear) relationship between past and present values of an observed time-series. Within this nonlinear auto-regressive (NAR) representation of the system,[1] once a one-step-ahead model[2] has been identified, a challenging task is to forecast the value of the time-series at, say, k steps ahead in time. This is known as multiple-step-ahead iterative forecasting, where a one-step-ahead model is iterated up to the desired horizon (Farmer and Sidorowich, 1988; Judd and Small, 2000). Although an obvious naive way of performing iterative prediction is to feed back only previous estimates (i.e. the mean of the Gaussian predictive distribution in the GP case), it has been shown to be a sub-optimal solution (Ahmad and Tresp, 1993; Tresp and Hofmann, 1995).

The modelling of nonlinear dynamic systems with GPs is still in its infancy (Murray-Smith et al., 1999; Murray-Smith and Girard, 2001; Kocijan et al., 2003b; Kocijan et al., 2003a). The possible reason for that simply being that the engineering community is more used to parametric models, and the probabilistic GP model has not yet gained widespread popularity among it. Nevertheless, GPs appear to be well suited for modelling such systems. Indeed, the identification of nonlinear dynamic systems from experimental data is often rendered difficult by the fact that, usually, most of the data lie around equilibrium points, and only sparse data are available in transient regions (i.e. far from equilibrium). In such conditions, a GP proves to be efficient, as the model retains the available data, and performs inference conditional on the current state and local data. Also, the uncertainty of model predictions can be made dependent on local data density, and the model complexity directly relates to the amount of available data (more complex models needing more evidence to make them likely). This work on the propagation of uncertainty when predicting ahead in time, along with the incorporation of derivative observations and of the variance for the cautious control of systems, will hopefully contribute towards the wider use of Gaussian Processes in dynamic systems modelling.

1.2 Contribution and outline

Chapter 2 briefly introduces Gaussian random functions and the GP machinery as the model is used in regression tasks. In this thesis, since only zero-mean processes are considered, the covariance function alone defines the process. It is a parametric function of the inputs that gives the covariances between

[1]Or possibly a Nonlinear Auto-Regressive with eXogeneous inputs (NARX) structure, if control inputs are present.

[2]When the observation at time t is a function of that at time $t-1$, and possibly other delayed values.

the corresponding outputs. A popular covariance function is the Gaussian (squared exponential) one, conveying the belief that functions drawn from the process should be smooth and continuous. It is such that points close together in the input space lead to outputs that are more correlated than points further apart, with the covariance decaying exponentially. As well as having proved to be a useful covariance function in many applications (Rasmussen, 1996a), it is also very convenient as it enables many derivations to be analytically carried out exactly.

Assuming a GP model was identified using ‘clean’ (i.e. noise-free) inputs, the task of making a prediction at a noisy input is addressed in **Chapter 3**. It involves the integration of the predictive distribution over the input distribution, which cannot be done without approximations. This problem is solved by taking an analytical approach that consists of computing only the mean and variance of the new (non-Gaussian) predictive distribution, an approach that I refer to as the *Gaussian approximation*. Depending on the form of the covariance function of the process, these moments are derived exactly (in the cases of the Gaussian and the linear kernels), or approximately, within a Taylor approximation of the covariance function at hand. On a simple static example, this analytical approach is compared to the numerical approximation of the integral, approximating the true predictive distribution by simple Monte-Carlo. In experiments, the Taylor approximation is validated by computing the approximate moments using the Gaussian covariance function, and comparing them to the exact ones (this approximation is briefly discussed in Appendix B). At the end of this chapter, I indicate how the challenging task of training a model when the inputs are noisy can be tackled, within a similar *Gaussian approximation*.

In **Chapter 4**, the results presented in the previous chapters are applied to the modelling and the multiple-step-ahead iterative forecasting of nonlinear dynamic systems. Based on the results derived in Chapter 3, a methodology to propagate the uncertainty induced by each successive prediction is suggested. After each iteration, the whole predictive distribution of the output just predicted is fed back into the state for the next time-step. Therefore, the state is now a random vector, with mean composed of the delayed predictive means, and covariance matrix with the corresponding predictive variances on its diagonal. Not only are the predictive variances fed back, but the cross-covariances between the delayed outputs are also accounted for, resulting in a full input covariance matrix as the model proceeds ahead in time. On the Mackey-Glass chaotic time-series, the iterative prediction with propagation of the uncertainty, obtained within the *Gaussian approximation*, is compared to the Monte-Carlo solution. Also, the naive approach, using only the delayed predictive means, is shown to lead to poor predictions, highlighting the importance of accounting for the uncertainty induced by the successive predictions.

Chapter 5 illustrates the modelling with a Gaussian Process of two real-life applications. The gas-liquid separator process is part of a plant situated at the Jozef Stefan Institute in Slovenia. Based on measurements of the gas pressure and the water level in the reservoir, subject to the (controlled) openness of a pressure valve, the aim is to model the gas pressure. Using a subset-selection approach based on the Automatic Relevance Determination tool (Neal, 1995; MacKay, 1994), the identification of a zero-mean GP with Gaussian covariance function is first discussed. Then, a number of simulations (i.e. 'infinite-step'-ahead prediction) of the test signal are performed, and the predictions obtained with and without propagation of the uncertainty are examined, depending on the point at which the simulations are started.

The other application is the challenging pH neutralisation process benchmark (Henson and Seborg, 1994), where the measured pH is subject to control inputs. Although the process is well known to be nonlinear, the identification of a linear model first and then that of a GP on the residuals leads to a better one-step-ahead predictive performance than a GP alone. Using this 'mixed model', the iterative k-step-ahead prediction of a new signal, with and without propagation of the uncertainty, is performed. To do so, a modified version of the propagation of uncertainty algorithm is presented, to account for the interaction between the GP and the linear model at each time-step. Also, for this experiment, the iterative scheme is compared to the direct multi-step-ahead prediction method, where a model is trained to directly predict k steps ahead.

Finally, **Chapter 6** presents further extensions of the model. First, the incorporation of derivative observations in the GP model, which is of interest for at least two reasons. Local linear models (linearisations) are commonly found in engineering applications, and should therefore been taken into account in a model where function observations are available. Also, derivative observations can potentially reduce the computational burden usually associated with GPs, as few derivative points can lead to a similar performance to a model with more function observations (Solak et al., 2003). Conveniently, the derivative process of a GP being a GP itself (Pugachev, 1967), derivative observations can relatively easily and elegantly be incorporated into the model. For this *mixed* training set, the resulting predictive mean and variance of a function output corresponding to a new input are composed of a 'functional part', a 'derivatives part' and mixed components (arising in the variance). Within the *Gaussian approximation*, I then address the problem of predicting at a noisy input (the details for the computation of the predictive mean and variance, in the particular case of the Gaussian covariance function, can be found in Appendix C, Section C.1).

The importance of the uncertainty associated with a point prediction has already been outlined, as it quantifies one's degree of belief in an estimate. Therefore, it seems natural that if this prediction is to be used in a decision-making process, the uncertainty attached to it should be accounted for, in order to make 'cautious' decisions. In this line of thought, in a control context, the knowledge of the predictive variance can be used in an informative manner, to derive a 'cautious' cost function (Murray-Smith and Sbarbaro, 2002; Murray-Smith et al., 2003; Sbarbaro and Murray-Smith, 2003). The propagation of uncertainty method is then applied to a cautious controller, for the multi-step-ahead control of a MISO (multi-input single-output) system.

Appendix A provides some useful mathematical formulae used throughout the thesis. The multi-disciplinary nature of this work, involving aspects of computing science, statistics and control engineering, has led me to make the derivations as transparent as possible, to a broad readership, in order to ease the task of potentially interested researchers and engineers accessing and implementing these ideas.

1.3 Joint work and publications

The idea of propagating the uncertainties in Gaussian Processes followed discussions between Roderick Murray-Smith and Carl Edward Rasmussen. The derivations of the approximate moments when predicting at a noisy input were started together with Carl Edward Rasmussen (Girard et al., 2002). It is with Joaquin Quiñonero Candela that the exact moments were derived, in the case of the Gaussian covariance function (Girard et al., 2003), and Joaquin Quiñonero Candela outlined how these results directly apply to the Relevance Vector Machine model (Quinonero-Candela et al., 2003; Quinonero-Candela and Girard, 2002). In the present document, I have worked on the derivation of simpler expressions and how to link the different cases in a consistent manner. As for the training of a GP model with noisy inputs, the approach presented in Section 3.6 is significantly different from (Girard and Murray-Smith, 2003).[3] I am grateful to Professor Mike Titterington and Roderick Murray-Smith for their useful comments.

I derived the expressions for the incorporation of derivative observations into a GP model, which were implemented and tested by Jus Kocijan. Preliminary results can be found in (Kocijan et al., 2004b) and a paper will be submitted to the 16^{th} IFAC World Congress, to be held in 2005.

Although in the current document only additive white noise on the observed outputs is considered, I also worked on coloured noise models such as AR, MA or ARMA (Murray-Smith and Girard, 2001).

I have taken part in the evaluation of the GP model and the methodology presented in Chapter 4 on various dynamic systems (Kocijan et al., 2003b; Kocijan et al., 2003a). Roderick Murray-Smith introduced me to the application of GPs in a control context, and the propagation of uncertainty applied in that case led to (Murray-Smith et al., 2003; Kocijan et al., 2004c; Kocijan et al., 2004a).

In the remainder of this document, I will use the first person plural. Although this document is at my sole responsibility, I believe it represents the achievement of collaborative work, if only through useful discussions and comments from researchers I have met during the last four years.

[3] I would like to thank an anonymous reviewer whose comments on the results presented in the technical report helped me to revise my approach.

Chapter 2

Modelling with a Gaussian Process

This chapter intends to provide a comprehensive introduction to Gaussian random functions and Gaussian Processes, as used for modelling purposes. The emphasis is not on the mathematical theory behind the concepts involved, but rather on their practical utility for our ends. We hope our simple illustrations will convince the unfamiliar reader of the potential and flexibility of the model. In Section 2.4, we simply recall the Bayesian approach to parametric modelling and, how GPs relate to that framework.

2.1 Brief historic

In his chapter on Gaussian Processes (Mackay, 2003), MacKay goes back to 1880 for the first use of the model for time-series analysis (Lauritzen, 1999). In geostatistics, where the model is known as *kriging* (after (Krige, 1951)), it was developed by Matheron (Matheron, 1963), and much work is still being done in this field (Cressie, 1993). Also, in geophysics, Tarantola and Valette pioneered the Bayesian formulation of inverse problems using GPs (Tarantola and Valette, 1982). The model was clearly formulated to solve regression problems in statistics (O'Hagan, 1978) and gained popularity in the machine learning community, mainly after the works of (Neal, 1994; Neal, 1995; Rasmussen, 1996a). The Bayesian interpretation of the model can be found in (Williams and Rasmussen, 1996; Williams, 1997c; Neal, 1997; Mackay, 1997) and detailed introductions in (Williams, 2002; Mackay, 2003; Seeger, 2003). In these last two references in particular, the relation of the GP model to generalised radial basis functions, spline smoothing methods and kernel models such as support and relevance vector machines is presented (we refer to (Mackay, 1997; Tipping, 2001; Kimeldorf and Wahba, 1970) for these models).

2.2 Gaussian random functions

Random functions are very complex mathematical objects and a Gaussian process is the simplest random function in that it is fully characterised by its mean and covariance functions (Pugachev, 1967; Papoulis, 1991). If the argument is time, it will usually be called a Gaussian stochastic process, and a Gaussian random field if the argument represents a position in, say, $\mathcal{R}^3$.

Let $f(\mathbf{x})$ be a stochastic field, for $\mathbf{x} \in \mathcal{R}^D$, with mean function $m(\mathbf{x}) = E[f(\mathbf{x})]$ and covariance function $C(\mathbf{x}_i, \mathbf{x}_j) = \text{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)]$. We can denote the Gaussian Process (GP) $f(\mathbf{x})$ by

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

A Gaussian Process can be be thought of as a generalisation of multivariate Gaussian random variables to infinite sets: The process is Gaussian if all joint distributions are multivariate normal. Therefore, for *any* given set of inputs $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, the corresponding random variables $\{f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)\}$ have an n-dimensional normal distribution:

$$p(f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)|\mathbf{x}_1, \dots, \mathbf{x}_n) = \mathcal{N}(\mathbf{m}, \mathbf{\Sigma})$$

where $\mathbf{m}$ is the $n \times 1$ vector of expectations (mean values) and $\mathbf{\Sigma}$ the $n \times n$ matrix of covariances between all pairs of points, i.e.

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

The covariances between two outputs are given by the covariance function evaluated at the corresponding inputs: We have $\Sigma_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$, that is

$$\mathbf{\Sigma} = \begin{bmatrix} C(\mathbf{x}_1, \mathbf{x}_1) & \dots & C(\mathbf{x}_1, \mathbf{x}_n) \\ \dots & \dots & \dots \\ C(\mathbf{x}_n, \mathbf{x}_1) & \dots & C(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} .$$

In this thesis, we consider zero-mean processes, assuming no prior information is available to contradict this hypothesis. In the following example, we will see that this assumption is not overly restrictive in practice as a great variety of functions can be generated by a zero-mean process. The reason for the constant mean assumption is that we will mostly be interested in second-order stationary processes (recall that a process is second-order stationary if it has a constant mean and a covariance function that only depends on the distance between the inputs (Pugachev, 1967)). In that respect, the constant-mean (or, without loss of generality, zero-mean[1]) assumption is natural. We refer to (O'Hagan, 1978; Cressie, 1993) for non-constant mean processes.

In the remainder of this thesis, we then consider

$$p(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \; . \tag{2.1}$$

Picturing an object in three dimensions is at most what we are capable of and we are now facing an n-dimensional probability density. We can get a feeling of what it represents by visualising realisations of the process. We hope the following illustration will highlight how wide a variety of functions can be produced by a process with zero-mean and a simple covariance function.

2.2.1 A simple illustration

Consider a Gaussian Process $f(x)$, for a one-dimensional $x \in \mathcal{R}$, with zero-mean and covariance function

$$C_g(x_i, x_j) = v \exp\left[-\frac{1}{2}w(x_i - x_j)^2\right] \; . \tag{2.2}$$

For $x_i = x_j$, v corresponds to the variance of $f(x_i)$, $C_g(x_i, x_i) = \text{Cov}[f(x_i), f(x_i)] = v$. The correlation length, which represents the length along which successive values are strongly correlated (the correlation diminishing exponentially as the distance between the points increases), is defined to be $\lambda = 1/\sqrt{w}$. We will look at this particular covariance function in greater detail later.

Let GP1 be the GP for which $v = 1, w = 0.25$ and let GP2 be another GP with the same covariance function but with $v = 2, w = 1$.

[1]In practice, the zero-mean simplification can be dealt with by centring the data as $\mathbf{t} = \mathbf{t} - \bar{\mathbf{t}}$, where $\bar{\mathbf{t}}$ is the data sample mean. One can also add an extra constant term to the covariance function, reflecting how far the mean of the function is expected to fluctuate from the mean of the process (Mackay, 2003; Gibbs, 1997).

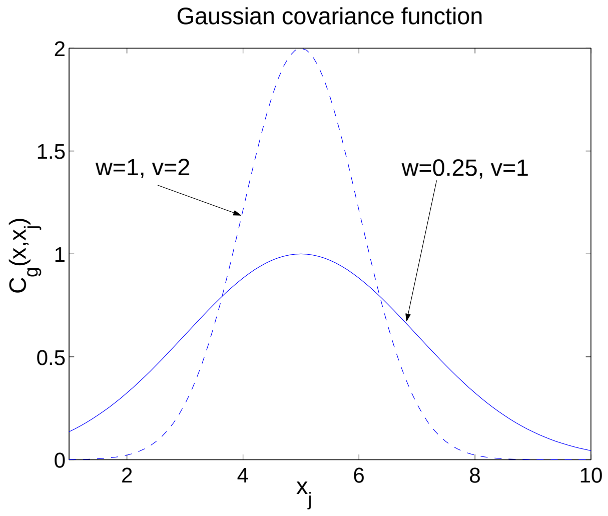

Figure 2.1: Gaussian covariance function $C_g(x, x_j)$, for $x = 5$ and $x_j \in [1, 10]$. The w parameter relates to the width of the kernel and the v parameter to the amplitude.

Figure 2.1 shows the covariance functions between $x = 5$ and $x_j \in [1, 10]$, for GP1 (continuous) and GP2 (dashed). For GP1, $w = 0.25$ implies a correlation length of 2, corresponding to a broader function that for GP2, for which the correlation length is 1. The smaller variance of GP1 ($v = 1$) implies a maximum (corresponding to $C(x = 5, x = 5)$) with lower amplitude than that of GP2, for which $v = 2$.

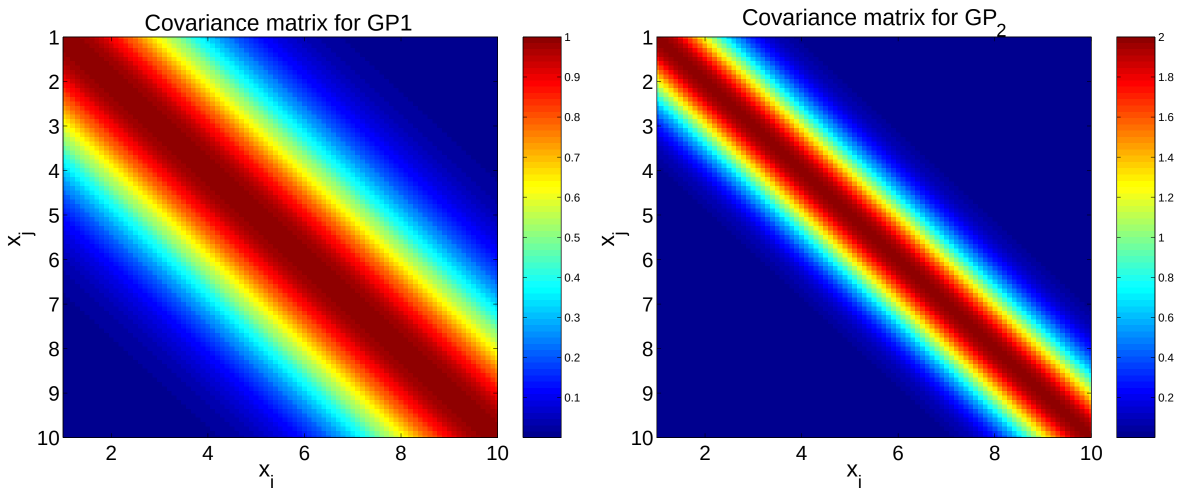

Figure 2.2: Symmetric covariance matrices with Gaussian kernel given by (2.2), for $x_i, x_j \in [1, 10]$. The amplitude of the covariances and the variances (on the diagonal) are controlled by v. For GP1 (left), with a smaller w than GP2, points further apart are still correlated.

For given values of x_i and x_j, the knowledge of only these two parameters enables us to compute the matrix of covariances between the corresponding outputs, $f(x_i)$ and $f(x_j)$. Figure 2.2 shows the covariance matrices formed using the Gaussian kernel (2.2), for x_i and x_j in $[1, 10]$. As the distance between inputs increases, the covariances between the points for GP1 (left) decreases less rapidly than for GP2 (right), because of the smaller value of the w parameter. Again, we can note the smaller variances for GP1 (diagonal terms), controlled by the v parameter.

Let us now illustrate the 'action' or impact of these two parameters on the possible realisations (sample functions) of the process.[2] Figure 2.3 (left) shows typical realisations drawn from GP1 (continuous lines) and GP2 (dashed lines). We can first notice that all the realisations are 'smooth', which is a characteristic of this covariance function. The samples from GP2 vary much more rapidly in the horizontal direction than do those of GP1, as well as having a larger amplitude of variation. These two observations are again to be related to the w and v parameters: A large w corresponds to a small correlation length, thus implying rapid horizontal variations, and a large variance v parameter allows the realisations to have larger fluctuations away from the zero-mean. One can think of w and v as knobs for controlling horizontal as well as vertical variations.

As noted earlier, $f(x)$ is a GP if for *any* given set of inputs $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, $\{f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)\}$ have an n-dimensional normal distribution. On the right panel of Figure 2.3, are the normalised histogram plots of 1000 samples of $f(x = 4)$ for GP1 (left) and GP2 (right), along with the 'true' probability distribution of this random variable ($f(x = 4) \sim \mathcal{N}(0, v)$, with $v = 1$ for GP1 and $v = 2$ for GP2). To further illustrate this point, Figure 2.4 shows the scatter-plot of 1000 samples of $f(x = 4), f(x = 6)$ and $f(x = 4), f(x = 9)$, for GP1 (left) and GP2 (right). Also plotted is the normalised joint distribution of the random variables, computed using the true mean and covariances. Recall that the covariance function gives the covariances between $f(x_i)$ and $f(x_j)$ as a function of x_i and x_j. In the particular case of the covariance function C_g, it is a *weighted* function (weighted by w) of the Euclidean distance between the inputs. Given the value of w for GP1 compared to GP2, we see that, for GP1, $p(f(x = 4), f(x = 6))$ presents a strong correlation between $f(x = 4)$ and $f(x = 6)$, whereas $p(f(x = 4), f(x = 9))$ corresponds to almost uncorrelated random variables (the

[2] It can easily be shown that realisations of a Gaussian Process can be obtained by the convolution of a white noise and the square root of the covariance function of the process.

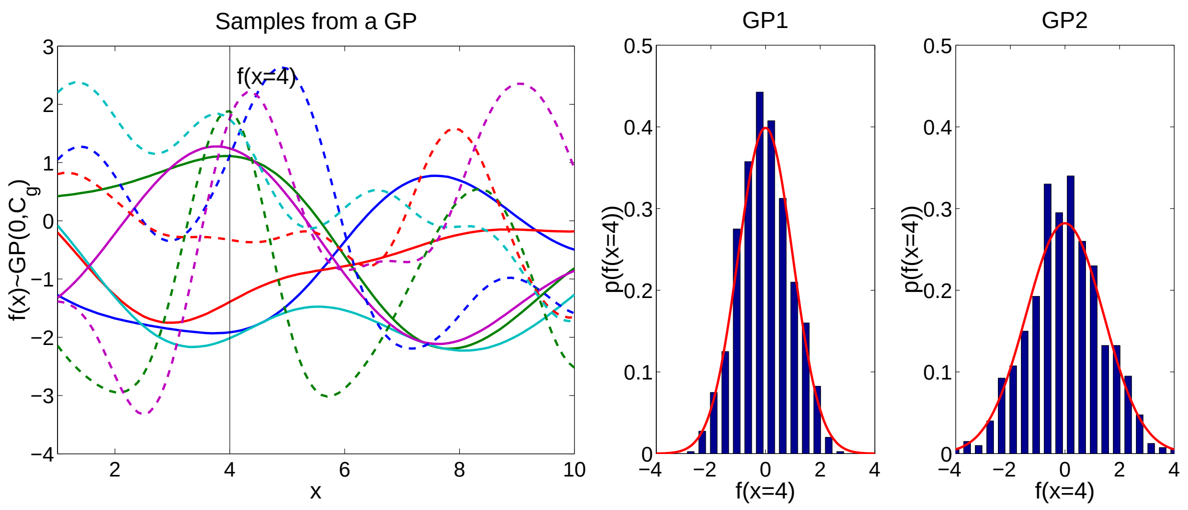

Figure 2.3: Left panel: Smooth samples from GP1 (continuous lines) and GP2 (dashed lines), two zero-mean Gaussian Processes with Gaussian covariance function, with different values of parameters. The short correlation length and the larger v parameters of GP2 imply realisations with rapid horizontal variation and with larger fluctuations about zero. At $x = 4$, the corresponding $f(x)$ is a random variable normally distributed with zero-mean and variance 1 for GP1 and 2 for GP2. The right panel shows the normalised histogram plot of the 1000 samples of $f(x = 4)$ and the true corresponding one-dimensional density function (red line).

inputs being further apart). However, for GP2, for which w is much larger, $x = 4$ and $x = 6$ are already too far apart to allow $f(x = 4)$ to be strongly correlated to $f(x = 6)$.

2.2.2 Covariance functions

Since we consider zero-mean processes, all that is needed to characterise the GP is its covariance function. Our simple example will have hopefully already highlighted its central role, as it conveys all the information on the kind of function generated by the process. The covariance function thus determines the properties of samples drawn from the GP and, when applied to regression, it controls how much the data are smoothed in estimating the underlying function. A wide choice of valid covariance functions can be found in (Abrahamsen, 1997; Genton, 2001).

Any form of covariance function is admissible, provided it generates a non-negative-definite covariance matrix. That is, any covariance function $C(.,.)$ must satisfy

$$\sum_{i,j} a_i a_j C(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

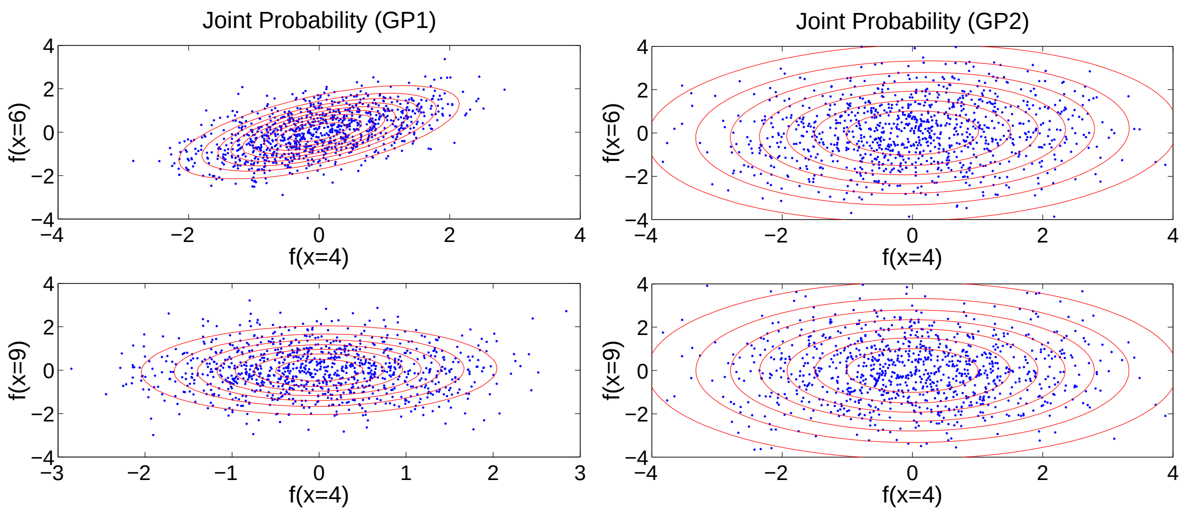

Figure 2.4: Joint distribution of $f(x=4), f(x=6)$ (top) and $f(x=4), f(x=9)$ (bottom) for GP1 (left) and GP2 (right). For GP2, the distance between $x=4$ and $x=6$ is already too large (wrt to w) to allow noticeable correlations between the corresponding $f(x=4)$ and $f(x=6)$. For GP1, we see that, as the distance between the inputs increases, the correlation decreases.

for any finite set of points $\mathbf{x}_1, \dots, \mathbf{x}_n$ and arbitrary real coefficients $a_1, \dots, a_n$.

Stationary covariance function

As previously mentioned, the process is (second-order) stationary if it has constant mean and

$$\mathrm{Cov}[f(\mathbf{x}_i), f(\mathbf{x}_j)] = C(||\mathbf{x}_i - \mathbf{x}_j||)$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{R}^D$ (note that $C(\mathbf{x})$ is $\mathrm{Cov}[f(\mathbf{x}), f(\mathbf{0})]$). In practice, such isotropic covariance functions are widely used. They are invariant by translation, so that the covariance between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ does not depend on the values of the corresponding inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ but only on the distance separating them. In geostatistics, the variogram, defined as

$$\mathrm{Var}[f(\mathbf{x}_i) - f(\mathbf{x}_j)] = 2\gamma(||\mathbf{x}_i - \mathbf{x}_j||)$$

is more widely used (see, e.g. (Cressie, 1993)). Such a process is called intrinsically stationary. Intrinsic stationarity is weaker than second-order stationarity as considered above but, if the latter holds, we have $\gamma(||\mathbf{x}_i - \mathbf{x}_j||) = C(\mathbf{0}) - C(||\mathbf{x}_i - \mathbf{x}_j||)$ where $\gamma(.)$ is the semi-variogram.

Let τ be the Euclidean distance $||\mathbf{x}_i - \mathbf{x}_j||$. A general class of stationary covariance functions is the Matern form,

$$C(\tau) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{2\sqrt{\nu}\tau}{\kappa}\right)^{\nu} K_{\nu}\left(\frac{2\sqrt{\nu}\tau}{\kappa}\right) ,$$

where Γ is the Gamma function and $K_{\nu}(.)$ is the modified Bessel function of the second kind whose order is the differentiability parameter $\nu > 0$. Then, ν controls the smoothness of typical sample functions which are $(\nu - 1)$ times differentiable. This class allows us to express the prior lack of knowledge about the sample function differentiability. (Gneiting, 2002) derived compactly supported kernels (covariance functions which vanish when the distance between the inputs is larger than a certain cut-off distance) from this class. Although not used in this thesis, these kernels are especially interesting as they can allow for computationally efficient sparse matrix techniques (a property much appreciated when we have to invert $N \times N$ matrices, where N is the size of the data set).

For $\nu \to \infty$, the Matern approaches the Gaussian squared exponential[3] covariance function. Such a covariance function has sample functions with infinitely many derivatives which are therefore smooth and continuous (as observed in the example in Section 2.2.1). In the machine learning community, this covariance function became a popular choice after Rasmussen demonstrated that a GP with such a covariance function performed as well as, if not better than, other popular models like neural networks (Rasmussen, 1996a). It is usually expressed as

$$C(\mathbf{x}_i, \mathbf{x}_j) = v \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right] \tag{2.3}$$

with $\mathbf{W}^{-1} = \mathrm{diag}[w_1 \dots w_D]$ and where each parameter w_d (sometimes referred to as roughness parameter) relates to the correlation length in direction d ($w_d = 1/\lambda_d^2$). As already mentioned, the correlation length represents the length along which successive values are strongly correlated, the correlation diminishing exponentially as the distance between the points increases. Then, if λ_d is large, a typical function is expected to be nearly constant in that direction and the corresponding input feature can then be thought of as being irrelevant, as in the Automatic Relevance Determination tool of MacKay and Neal (Neal, 1995; MacKay, 1994). Here, we only consider a diagonal $\mathbf{W}$, but a full matrix, allowing for modelling interactions between different input dimensions, can be used (Vivarelli

[3]This covariance function is sometimes referred to simply as ‘Gaussian covariance function’, in statistics, or ‘squared exponential covariance function’ in the machine learning community.

and Williams, 1999). The parameter v is the variance of the process, controlling the overall vertical scale of variation relative to the zero mean of the process in the output space (the vertical amplitude of variation of a typical function). As we will see in Chapter 3, this covariance function is of special importance to us as it allows for the exact evaluation of integrals involved when dealing with noisy inputs.

Non-stationary covariance function

If the covariances are believed not to depend on the distance between the points in the input space but on the values the inputs take, one can consider non-stationary covariance functions. The simplest one is the one corresponding to a linear trend,

$$C(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{D} \alpha_d x_i^d x_j^d \ , \tag{2.4}$$

where x_i^d is the d^{th} component of $\mathbf{x}_i \in \mathcal{R}^D$. This covariance function is easily derived by considering a linear model with a Gaussian prior on the parameters: Let $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ and $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w)$ with $\mathbf{\Sigma}_w = \text{diag}[\alpha_1 \dots \alpha_D]$. We then have

$$E[f(x_i)|\mathbf{x}_i] = E[\mathbf{x}_i^T\mathbf{w}] = \mathbf{x}_i^T E[\mathbf{w}] = 0$$

and therefore

$$\text{Var}[f(x_i)|\mathbf{x}_i] = E[(\mathbf{x}_i^T\mathbf{w})^2] = \mathbf{x}_i^T E[\mathbf{w}\mathbf{w}^T]\mathbf{x}_i = \mathbf{x}_i^T \mathbf{\Sigma}_w \mathbf{x}_i \ .$$

Similarly, the covariance between $f(x_i)$ and $f(x_j)$ is

$$\text{Cov}[f(x_i), f(x_j)|\mathbf{x}_i, \mathbf{x}_j] = \mathbf{x}_i^T \mathbf{\Sigma}_w \mathbf{x}_j = \sum_{d=1}^{D} \alpha_d x_i^d x_j^d \ .$$

Although we have not yet discussed the relationship between GPs and parametric models, we can already feel the tight link between Bayesian parametric modelling and the GP model, as here, the form of the covariance function can be thought of as dictated by the form of the parametric model assumed in the first place.

In (Paciorek and Schervish, 2004), a class of non-stationary covariance functions is introduced, which includes a non-stationary version of the Matern covariance. Also, (Gibbs, 1997) uses a similar form to that of the Gaussian covariance function (2.3), allowing for spatially varying length scales by letting λ_d be a function of $\mathbf{x}$.

Following the work of Neal (Neal, 1995), Williams derives analytically the covariance function corresponding to networks with sigmoidal and Gaussian hidden units (Williams, 1997a):

$$C(\mathbf{x}_i, \mathbf{x}_j) = \frac{2}{\pi} \sin^{-1} \left(\frac{2\tilde{\mathbf{x}}_i^T S \tilde{\mathbf{x}}_j}{\sqrt{(1 + 2\tilde{\mathbf{x}}_i^T S \tilde{\mathbf{x}}_i)(1 + 2\tilde{\mathbf{x}}_j^T S \tilde{\mathbf{x}}_j)}} \right) \tag{2.5}$$

that corresponds to the Multi-Layer Perceptron (neural network with sigmoidal transfer function), which is in the range $[-1, 1]$, where $\tilde{\mathbf{x}}$ is the input augmented by a unit entry (by analogy with the bias term), and S is the covariance matrix of the normally distributed hidden unit weight.[4]

An alternative to a non-stationary GP is to use a mixture of stationary processes, allowing for variable smoothness in different parts of the input space, as done in (Tresp, 2001; Rasmussen and Ghahramani, 2002; Shi et al., 2002).

2.3 GP for modelling noisy observations

We now turn to the use of Gaussian Processes in regression problems. Given a set of N D-dimensional inputs, $\mathbf{x}_i \in \mathcal{R}^D$, and corresponding observed scalar outputs, $t_i \in \mathcal{R}$, we wish to find a mapping between these inputs and outputs so as to be able to make predictions of the system's future responses, given new inputs. Due to external disturbances (such as measurement noise), the observations are seen as noisy versions of the 'true' system's responses. Although more 'realistic' noise models have been considered (Mackay, 1997; Gibbs, 1997; Goldberg et al., 1998; Murray-Smith and Girard, 2001), we restrict our attention to an additive white noise with a priori unknown variance v_t (the noise is independent and identically distributed across observations). The data generative mechanism can then be written

$$t_i = y_i + \epsilon_i \,, \tag{2.6}$$

[4]This covariance function has been implemented by Rasmussen (http://www.gatsby.ucl.ac.uk/∼carl/Code) in the form

$$C(\mathbf{x}_i, \mathbf{x}_j) = v \sin^{-1} \left(\frac{\tilde{\mathbf{x}}_i^T S \tilde{\mathbf{x}}_j}{\sqrt{(1 + \tilde{\mathbf{x}}_i^T S \tilde{\mathbf{x}}_i)(1 + \tilde{\mathbf{x}}_j^T S \tilde{\mathbf{x}}_j)}} \right) .$$

for $i = 1 \dots N$, where $y_i = f(\mathbf{x}_i)$ is the noise-free response of the system to the input $\mathbf{x}_i$ and $\epsilon_i \sim \mathcal{N}(0, v_t)$.

2.3.1 From the joint . . .

For the moment, we do not wish to specify the form of the covariance function and simply assume that it depends on a set of unknown parameters Θ. If $f(\mathbf{x})$ is a zero-mean GP with covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$, then, for given $\mathbf{x}_1, \dots, \mathbf{x}_n$, the joint probability distribution of $f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)$ is normal with zero mean vector and covariance matrix $\mathbf{\Sigma}$ such that $\Sigma_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$. The noise being white with variance v_t, we simply have

$$p(t_1, \dots, t_n | \mathbf{x}_1, \dots, \mathbf{x}_n) = \mathcal{N}(\mathbf{0}, \mathbf{K}_n) \quad \text{with} \quad \mathbf{K}_n = \mathbf{\Sigma} + v_t \mathbf{I} \, , \tag{2.7}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix.

We can split $t_1, \dots, t_n$ into two sets, or, for our purpose, into one vector $\mathbf{t} = [t_1, \dots, t_N]^T$ and one scalar t_* (and similarly for the corresponding inputs). Splitting $\mathbf{K}_n$ accordingly, we can write the joint distribution as follows:

$$p(\mathbf{t}, t_* | \mathbf{X}, \mathbf{x}_*) \propto \exp\left(-\frac{1}{2} \begin{bmatrix} \mathbf{t} \\ t_* \end{bmatrix}^T \begin{bmatrix} \mathbf{K} & \mathbf{k}(\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*)^T & k(\mathbf{x}_*) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{t} \\ t_* \end{bmatrix} \right) \, , \tag{2.8}$$

where $\mathbf{K}$ is now an $N \times N$ matrix, giving the covariances between t_i and t_j ($K_{ij} = C(\mathbf{x}_i, \mathbf{x}_j) + v_t \delta_{ij}$, for $i, j = 1 \dots N$ and where $\delta_{ij} = 1$ for $i = j$ and zero otherwise), $\mathbf{k}(\mathbf{x}_*)$ is an $N \times 1$ vector giving the covariances between t_i and t_*, such that $k_i(\mathbf{x}_*) = C(\mathbf{x}_*, \mathbf{x}_i)$ for $i = 1 \dots N$, and $k(\mathbf{x}_*)$ is the variance of t_*, that is $k(\mathbf{x}_*) = C(\mathbf{x}_*, \mathbf{x}_*) + v_t$.

It is from this joint distribution that we perform the learning and prediction task, by respectively marginalizing and conditioning on the observed data[5] (see Figure 2.5).

2.3.2 . . . To the marginal . . .

Once we have observed $\mathcal{D} = \{\mathbf{x}_i, t_i\}_{i=1}^N$, the likelihood of the data corresponds to the appropriate marginal part of $p(\mathbf{t}, t_* | \mathbf{x}, \mathbf{x}_*)$. It corresponds to the joint probability distribution *evaluated* at the

[5]Refer to Appendix A for a brief note on joint, marginal and conditional Gaussian distributions.

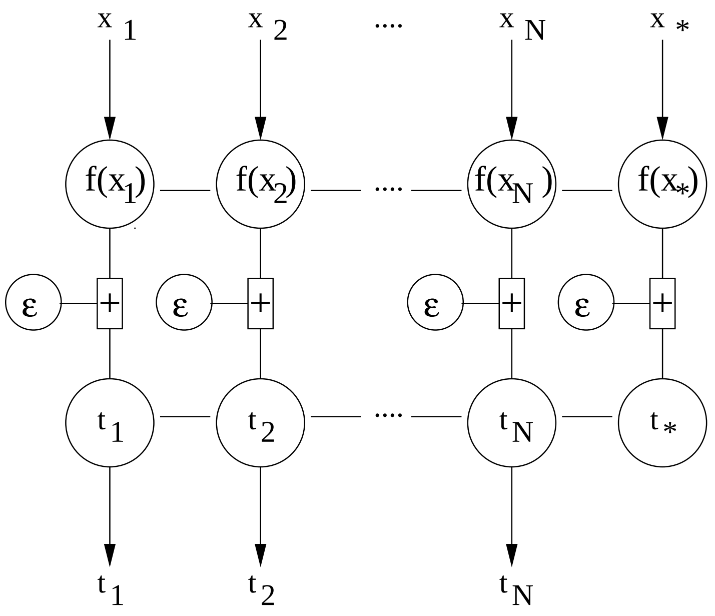

Figure 2.5: Modelling with Gaussian Processes (circled variables are random variables, non-circled ones are observed/given). For given $\{\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{x}_*\}$, the corresponding set of random variables $\{f(\mathbf{x}_1), \dots, f(\mathbf{x}_N), f(\mathbf{x}_*)\}$ have a joint multivariate Gaussian distribution. With the additive independent white noise assumption, the corresponding circled $\{t_1, \dots, t_N, t_*\}$ are also jointly normally distributed. The marginal part of the joint gives us the probability of the data $\{t_1, \dots, t_N\}$, and the posterior predictive distribution of $f(\mathbf{x}_*)$ (or equivalently t_*) corresponding to $\mathbf{x}_*$ is obtained by conditioning on the data and $\mathbf{x}_*$.

observed data:

$$p(t_1, \dots, t_N | \mathbf{x}_1, \dots, \mathbf{x}_N) = \mathcal{N}(\mathbf{0}, \mathbf{K}) \ , \tag{2.9}$$

where $\mathbf{K}$ is the $N \times N$ 'data covariance matrix'.

Let $\boldsymbol{\Theta} = \{\Theta, v_t\}$ be the set of free parameters (parameters Θ of the covariance function and the noise variance v_t). Here, we take a maximum-likelihood approach and find the unknown parameters by minimising

$$\mathcal{L}(\boldsymbol{\Theta}) = -\log[p(\mathbf{t}|\mathbf{x})] = \frac{N}{2}\log(2\pi) + \frac{1}{2}\log|\mathbf{K}| + \frac{1}{2}\mathbf{t}^T\mathbf{K}^{-1}\mathbf{t} \ , \tag{2.10}$$

where $|\mathbf{K}|$ denotes the determinant of $\mathbf{K}$. For doing so, we use a conjugate gradient optimisation technique with line-search (Rasmussen, 1996a), that requires the computation of the derivatives of the $\mathcal{L}(\boldsymbol{\Theta})$ with respect to each parameter Θ_j of $\boldsymbol{\Theta}$:

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \Theta_j} = \frac{1}{2}\mathrm{Tr}\left[\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \Theta_j}\right] - \frac{1}{2}\mathbf{t}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \Theta_j}\mathbf{K}^{-1}\mathbf{t} \ , \tag{2.11}$$

where Tr denotes the trace. This requires the inversion of the $N \times N$ covariance matrix $\mathbf{K}$ at each iteration which can be computationally demanding as N increases (techniques have recently been developed to reduce the computational cost, see (Williams and Seeger, 2001; Seeger et al., 2003; Murray-Smith and Pearlmutter, 2003; Shi et al., 2002)). Note that, for non-zero-mean processes, the method of restricted (or residual) maximum-likelihood can be used (Patterson and Thompson, 1971). We refer to (Rasmussen, 1996a; Neal, 1997) for a Bayesian treatment, where priors are put on the parameters of the covariance function.

2.3.3 . . . And the conditional

Having found the set of most likely parameters, the predictive (posterior) distribution of t_* corresponding to a new given input[6] $\mathbf{x}_*$ is readily obtained by conditioning the joint probability distribution $p(\mathbf{t}, t_*|\mathbf{x}, \mathbf{x}_*)$ on the observed data $\mathcal{D}$ and $\mathbf{x}_*$. It can be shown (Von Mises, 1964) that this conditional distribution $p(t_*|\mathcal{D}, \mathbf{x}_*)$ is Gaussian, with mean

$$E[t_*|\mathcal{D}, \mathbf{x}_*] = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \mathbf{t} \tag{2.12}$$

and variance

$$\mathrm{Var}[t_*|\mathcal{D}, \mathbf{x}_*] = k(\mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}_*) \ . \tag{2.13}$$

We are in general more interested in the predictive distribution of the noise-free $f(\mathbf{x}_*)$. Denoting the predictive mean by $\mu(\mathbf{x}_*)$ and the predictive variance by $\sigma^2(\mathbf{x}_*)$, we directly have

$$\begin{cases} \mu(\mathbf{x}_*) = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \mathbf{t} \\ \sigma^2(\mathbf{x}_*) = C(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}_*) \ . \end{cases} \tag{2.14}$$

The most probable output $\mu(\mathbf{x}_*)$ can then be used as an estimate for the response of the system and $\sigma(\mathbf{x}_*)$, the associated uncertainty, can define a confidence interval for the predictor (error-bars $\pm 2\sigma(\mathbf{x}_*)$). As the data are used directly for making predictions, the uncertainty of the model predictions depends on the local data density, and the model complexity relates to the amount and the distribution of available data (Williams et al., 1995; Qazaz et al., 1996).

[6]Note that since there is no particular ordering of the inputs, $\mathbf{x}_*$ can be in $[\mathbf{x}_1, \mathbf{x}_N]$, thus corresponding to smoothing or filtering. If $\mathbf{x}_* < \mathbf{x}_1$ or $\mathbf{x}_* > \mathbf{x}_N$, it corresponds to an extrapolation task (or prediction in the last case).

There are two alternative ways of writing the predictive mean, giving more insight into this estimate: $\mu(\mathbf{x}_*)$ can be seen as a weighted sum of the observed targets,

$$\mu(\mathbf{x}_*) = \boldsymbol{\alpha}^T \mathbf{t} \quad \text{with} \quad \boldsymbol{\alpha}^T = \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}^{-1} \,, \tag{2.15}$$

where $\boldsymbol{\alpha}$ is the vector of weights (also called smoothing or effective kernel), or as a linear combination of $\mathbf{k}(\mathbf{x}_*)$ and the observations:

$$\mu(\mathbf{x}_*) = \mathbf{k}(\mathbf{x}_*)^T \boldsymbol{\beta} \quad \text{with} \quad \boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{t} \,. \tag{2.16}$$

As the number of data points increases, the value of the beta-coefficients becomes larger and the amplitude of the smoothing kernel (alpha-coefficients) smaller. The latter relates directly to the behaviour of $\mathbf{k}(\mathbf{x}_*)$, depending on the number of data points as well as on how far/close the new $\mathbf{x}_*$ is from the training inputs.

The following example illustrates the GP modelling of noisy data. In this particular case, we choose the data to actually come from a realisation of a zero-mean Gaussian Process with Gaussian covariance function, where $w = 0.04$ (corresponding to a correlation length of 5) and $v = 2$, given a one-dimensional argument x in $[0, 10]$. We select $N = 10$ training cases at random and corrupt the outputs with a white noise with variance 0.01. Starting the optimisation of the minus log-likelihood with an initial 'guess' of 1 for all parameters, it converges to $\mathcal{L}(\boldsymbol{\Theta}) = -0.9765$ after 100 iterations. The Maximum Likelihood (ML) parameters found are $w = 0.0531$, $v = 1.4363$ and $v_t = 0.0076$. Both v and v_t are under-estimated, whereas w is over-estimated but these values are satisfactory, considering the very small number of data points and their unevenly spread in the input space. We then make two predictions, at $x_* = 1$ and $x_* = 9$. Figure 2.6 (left) shows the underlying function (that is the realisation of the 'true' underlying GP), the training cases (crosses) and the predictions with their $\pm 2\sigma$ error-bars (circles). The right upper plot corresponds to the covariances between the test and training cases (for $x_* = 1$, crosses and $x_* = 9$, circles) and the bottom plot to the smoothing kernels.

For $x_* = 1$, which is between training points, the predictive variance is small, as the model is confident about its prediction, but for $x_* = 9$, the error-bars are significantly larger, as the the test input lies in a region where there are few or no training inputs. Indeed, the plot of the covariances

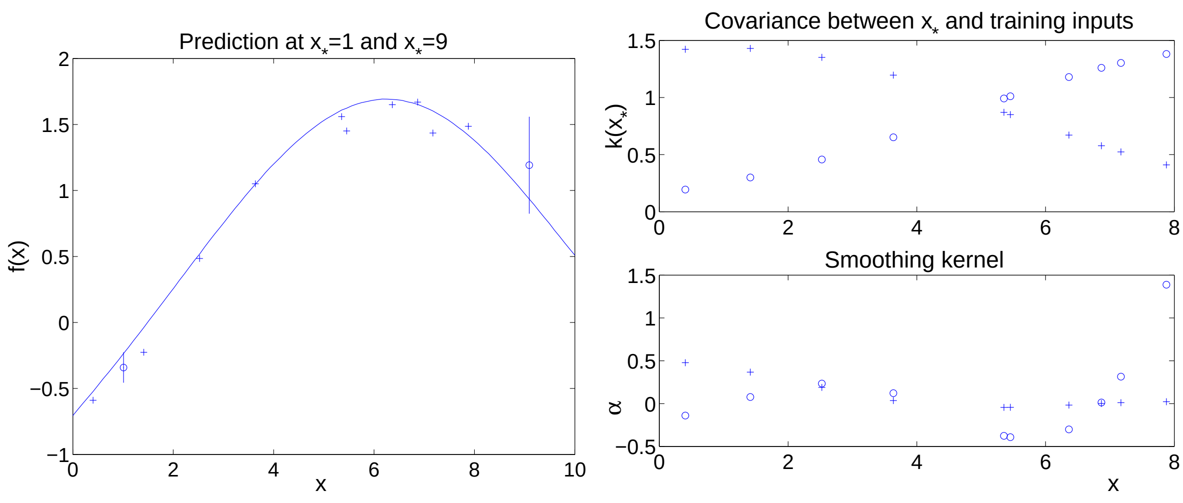

Figure 2.6: Left: GP modelling of the noisy observations (crosses). Mean predictions at $x_* = 1$ and $x_* = 9$ with their associated error-bars (circles), along with the true function (continuous line). Right: Covariances between the test and the training inputs (upper plots), and smoothing kernels (bottom plots). The crosses correspond to $x_* = 1$ and the circles to $x_* = 9$.

between the test and training inputs indicates that, those with $x_* = 9$ diminish more rapidly and to smaller values than those with $x_* = 1$.

Figure 2.7 shows samples drawn from the zero-mean GP prior (dashed lines) and from the predictive-posterior process, conditioned on the training data and 100 test inputs x_* in $[0, 10]$ (that is, the realisations are drawn from a 100-dimensional normal distribution, with mean vector $\mu(x_*)$ and covariance matrix $\sigma^2(x_*)$, computed for 100 x_*). We can notice the 'edge effect' from $x = 8$, where the samples start diverging, due to the lack of training data.

In all our experiments, we assess the predictive performance of the model by computing the average squared error,

$$E_1 = \frac{1}{N_t} \sum_{i=1}^{N_t} (y_i - \hat{y}_i)^2 \,,$$

and the average negative log-predictive density

$$E_2 = \frac{1}{2N_t} \sum_{i=1}^{N_t} \left[\log(2\pi) + \log(\sigma_i^2) + \frac{(y_i - \hat{y}_i)^2}{\sigma_i^2} \right] \,,$$

where y_i is the 'true' output, $\hat{y}_i$ is the model estimate (predictive mean), σ_i^2 the associated predictive variance and the average is over the number N_t of test points. This last measure of predictive perfor-

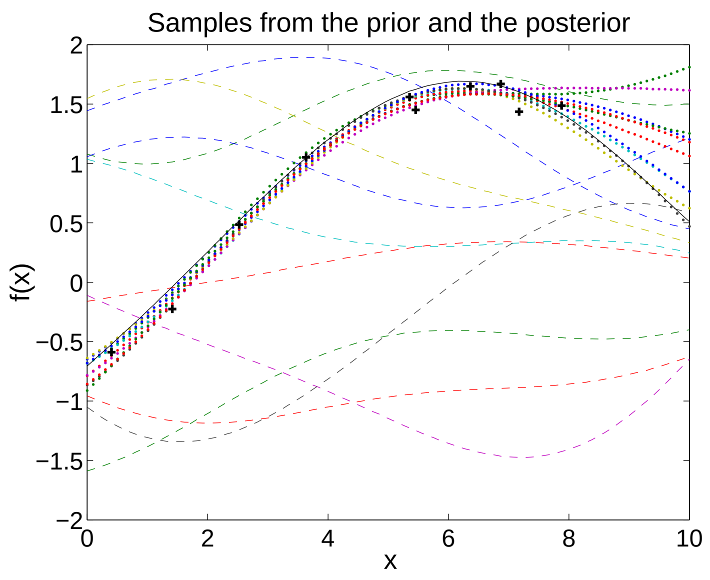

Figure 2.7: Samples from the prior process (dashed lines) and from the posterior (dotted lines), conditioned on the training data (crosses), for $x \in [0, 10]$.

mance is of greater interest than E_1 as it accounts for the model uncertainty (the predictive variance). It trades-off between the quality of the estimate and the accuracy of the model (how confident the model is in its prediction).

For this example, we obtain $E_1 = 0.0113$ and $E_2 = -0.1932$ for the prediction at $x_* = 1$ and $E_1 = 0.0657$, $E_2 = 0.2038$ for that at $x_* = 9$. The smaller E_1 and the more negative E_2, the 'better' the prediction is.

2.4 Relationship to Bayesian parametric modelling

For the reader more familiar with parametric models and the Maximum-Likelihood approach, this last section might be of interest, as we simply recall the Bayesian approach to parametric modelling and how GPs fit into this framework.

The Bayesian paradigm rests on Bayes' formula, which comes from a 'double use' of the definition of the joint probability density as the product of marginal and conditional densities: Let $\mathbf{z}_1$ and $\mathbf{z}_2$ be two continuous variables and $p(\mathbf{z}_1, \mathbf{z}_2)$ their normalised probability density. By definition, the marginal probability density for $\mathbf{z}_2$ is obtained by integrating $\mathbf{z}_1$ out, as

$$p(\mathbf{z}_2) = \int p(\mathbf{z}_1, \mathbf{z}_2) d\mathbf{z}_1 \ , \tag{2.17}$$

and the conditional probability density for $\mathbf{z}_1$ given $\mathbf{z}_2$ is

$$p(\mathbf{z}_1|\mathbf{z}_2) = \frac{p(\mathbf{z}_1, \mathbf{z}_2)}{\int p(\mathbf{z}_1, \mathbf{z}_2) d\mathbf{z}_1} = \frac{p(\mathbf{z}_1, \mathbf{z}_2)}{p(\mathbf{z}_2)} \ . \tag{2.18}$$

From these definitions, it follows that the joint probability density is given by

$$p(\mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{z}_1|\mathbf{z}_2) p(\mathbf{z}_2) \ . \tag{2.19}$$

Similarly, if we consider the marginal for $\mathbf{z}_1$ and the conditional for $\mathbf{z}_2$, we can write

$$p(\mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{z}_1) \ . \tag{2.20}$$

Bayes'rule is then simply obtained by equating (2.19) and (2.20), as

$$p(\mathbf{z}_1|\mathbf{z}_2) p(\mathbf{z}_2) = p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{z}_1) \ ,$$

leading to

$$p(\mathbf{z}_1|\mathbf{z}_2) = \frac{p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{z}_1)}{p(\mathbf{z}_2)} = \frac{p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{z}_1)}{\int p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{z}_1) d\mathbf{z}_1} \tag{2.21}$$

where the second equality is obtained using (2.17) along with (2.20).

Although Bayes' formula can first appear as a mathematical tautology, the beauty of the Bayesian approach is to interpret it as a combination of states of information which have been translated into probability densities. Equation (2.21) therefore tells us how to update our state of information on $\mathbf{z}_1$ given $\mathbf{z}_2$, i.e. how to go from $p(\mathbf{z}_1)$ to $p(\mathbf{z}_1|\mathbf{z}_2)$. The unconditioned $p(\mathbf{z}_1)$ is then called the *prior*, to convey the idea that it represents our state of knowledge before observing $\mathbf{z}_2$. In the same logic, the conditioned $p(\mathbf{z}_1|\mathbf{z}_2)$ is the *posterior* and $p(\mathbf{z}_2|\mathbf{z}_1)$, seen as a function of $\mathbf{z}_1$, is the *likelihood* of $\mathbf{z}_1$, based on $\mathbf{z}_2$. The denominator $p(\mathbf{z}_2)$, independent of $\mathbf{z}_1$ (the variable of interest), is the normalising constant, called *evidence* or marginal likelihood, obtained by integrating out the parameters $\mathbf{z}_1$. Up to the normalising constant $p(\mathbf{z}_2)$, the posterior of $\mathbf{z}_1$, given that $\mathbf{z}_2$ has been observed, is then proportional to its likelihood multiplied by its prior.

When applied to parametric data modelling, $\mathbf{z}_1$ corresponds to the parameters $\mathbf{w}$ the model depends on, and $\mathbf{z}_2$ to the observed data $\mathcal{D}$. Then, equation (2.21) enables us to update our prior on the parameters, where $p(\mathbf{z}_2|\mathbf{z}_1)$ corresponds to the probability of the data when the parameters are fixed:

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w}) \; . \tag{2.22}$$

Seen as a function of the parameters, $p(\mathcal{D}|\mathbf{w})$ is the likelihood of $\mathbf{w}$. For fixed $\mathbf{w}$, $p(\mathcal{D}|\mathbf{w})$ corresponds to the generative model for the data: Given our white noise assumption, $\epsilon \sim \mathcal{N}(0, v_t)$, $t_i = f(\mathbf{x}_i) + \epsilon_i$ reads as

$$\begin{aligned} E[t_i|\mathbf{x}_i, \mathbf{w}] &= f(\mathbf{x}_i) + E[\epsilon_i] = f(\mathbf{x}_i) \\ \mathrm{Var}[t_i|\mathbf{x}_i, \mathbf{w}] &= \mathrm{Var}[\epsilon_i] = v_t \; , \end{aligned}$$

and we have $p(\mathcal{D}|\mathbf{w}) = p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^N p(t_i|\mathbf{x}_i, \mathbf{w})$ where $p(t_i|\mathbf{x}_i, \mathbf{w}) = \mathcal{N}_{t_i}(f(\mathbf{x}_i), v_t)$.

The predictive distribution of $f(\mathbf{x}^*)$ corresponding to a new $\mathbf{x}^*$ is obtained by integration over the posterior distribution of the parameters:

$$p(f(\mathbf{x}_*)|\mathcal{D}, \mathbf{x}_*) = \int p(f(\mathbf{x}_*)|\mathcal{D}, \mathbf{x}_*, \mathbf{w})p(\mathbf{w}|\mathcal{D}, \mathbf{x}_*)d\mathbf{w} \; , \tag{2.23}$$

where it is in general assumed that the posterior is independent of the new input, such that we have $p(\mathbf{w}|\mathcal{D}, \mathbf{x}_*) = p(\mathbf{w}|\mathcal{D})$.

Note that the normalised distributions require the evaluation of $p(\mathcal{D}) = \int \prod_{i=1}^N p(t_i|\mathbf{w})p(\mathbf{w})d\mathbf{w}$ which is usually intractable, even in simple cases. It can be solved numerically, using Markov-Chain Monte-Carlo (MCMC) methods, to get samples from the posterior (Neal, 1993; Mackay, 1999), or, given an approximate analytical treatment, by computing the maximum a posteriori (MAP) estimator of $\mathbf{w}$ after maximisation of the posterior (Mackay, 1997).

As we have seen, the GP 'prior' is imposed directly on the joint $p(\mathbf{z}_1, \mathbf{z}_2)$, where $\mathbf{z}_1$ now represents the new function output y_* and $\mathbf{z}_2$ the set $y_1, \dots, y_N$ (or equivalently $t_1, \dots, t_N$). Then, given the independent noise assumption, $p(\mathbf{z}_2)$ corresponds to the marginal probability distribution for the data and the predictive distribution of y_*, $p(\mathbf{z}_1|\mathbf{z}_2)$, is obtained using equation (2.18). We thought recalling this point is important as it might be misleading calling the GP model Bayesian, as is sometimes done: The prior put on the space of functions comes from the very probabilistic nature of the model, not out of a use of Bayes' formula. At least in its most 'basic' form (the one considered here, that is without setting hyper-priors on the parameters of the covariance function), the GP machinery relies on the definition of conditional probabilities, not on Bayes' formula.

Finally, it might be helpful to interpret $t_i = f(\mathbf{x}_i) + \epsilon_i$ as follows: $f(\mathbf{x}_i)$ is *one* random variable (RV) 'coming from' a GP with zero-mean and covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$. Therefore, $f(\mathbf{x}_i)$ has zero-mean and variance $C(\mathbf{x}_i, \mathbf{x}_i)$. Assuming $\epsilon \sim \mathcal{N}(0, v_t)$ is independent of $f(\mathbf{x}_i)$, we have

$$\begin{aligned} E[t_i|\mathbf{x}_i] &= 0 \\ \mathrm{Var}[t_i|\mathbf{x}_i] &= C(\mathbf{x}_i, \mathbf{x}_i) + v_t \ . \end{aligned}$$

Now, for two inputs $\mathbf{x}_i$ and $\mathbf{x}_j$, the corresponding RVs $f(\mathbf{x}_i)$,$f(\mathbf{x}_j)$, 'coming from' the same GP, have a joint normal distribution, implying

$$\begin{aligned} p(t_i, t_j|\mathbf{x}_i, \mathbf{x}_j) &= \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathrm{Var}[t_i|\mathbf{x}_i] & \mathrm{Cov}[t_i, t_j|\mathbf{x}_i, \mathbf{x}_j] \\ \mathrm{Cov}[t_j, t_i|\mathbf{x}_j, \mathbf{x}_i] & \mathrm{Var}[t_j|\mathbf{x}_j] \end{bmatrix}\right) \\ &= \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} C(\mathbf{x}_i, \mathbf{x}_i) + v_t & C(\mathbf{x}_i, \mathbf{x}_j) \\ C(\mathbf{x}_j, \mathbf{x}_i) & C(\mathbf{x}_j, \mathbf{x}_j) + v_t \end{bmatrix}\right) . \end{aligned}$$

The very point of the GP model is to explicitly *model* the correlations between the observations. Random functions are very complicated mathematical objects but the way we use them can be thought of more as a 'tool' from which stems the feasibility of modelling covariances. Indeed, if we were to assume that $y_1, \dots, y_N$ corresponding to $x_1, \dots, x_N$ were jointly normal (with zero-mean for simplicity), this would mean having to determine $N(N+1)/2$ parameters (the entries of the $N \times N$ covariance matrix), with only N observations! But if we enter the realm of stochastic processes and view $y_1, \dots, y_N$ as an incomplete realisation of a Gaussian random function with zero-mean and covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$, the covariances between all pairs of points are simply obtained by computing $\mathrm{Cov}[y_i, y_j|\mathbf{x}_i, \mathbf{x}_j] = C(\mathbf{x}_i, \mathbf{x}_j)$ where $C(.,.)$ typically depends on very few parameters ($D + 1$ in the case of the covariance function given by (2.3)).

Chapter 3

Dealing with noisy inputs

In the previous chapter, we saw that, with the GP model, the predictive distribution of the output corresponding to a new noise-free input is Gaussian. We now address the problem of predicting the system output when the input is noisy (uncertain). In this case, the integration over the input distribution leads to a non-Gaussian predictive distribution. We present an analytical approach that consists of computing only the mean and variance of this distribution (Gaussian approximation). We show how, depending on the form of the covariance function of the process, we can evaluate these moments exactly or approximately (within a Taylor approximation of the covariance function). On a simple static numerical example, we compare our Gaussian approximation to the numerical approximation of the true predictive distribution by simple Monte-Carlo.

In Section 3.6, we indicate how a similar approximation can be taken to deal with the challenging problem of training a GP model with noisy inputs. Assuming a Gaussian covariance function, we show how the mean and covariance function of the noisy (non-Gaussian) process can be derived, accounting for the input noise.

3.1 Introduction

So far, we have only considered the inference task with noise-free inputs, but in many situations the inputs can be noisy, uncertain or ultimately missing (complete noise). Noisy inputs can arise in different situations, depending on the nature of a particular application.

3.1.1 Background

In statistics, models dealing with uncertain inputs are known as *errors-in-variables* models (Kendall and Stuart, 1958). Two principal error-in-variables models are the ‘classical model’ and Berkson’s model. For the classical model, the observed input is seen as varying around a true value, a situation arising when the system senses the inputs imperfectly, so that we observe a noise corrupted version of the true inputs. With Berkson’s model, the observed input is fixed and the true one is subject to random errors with zero mean. This model is useful for situations when the observed input is set to some value but the unobserved true input varies about this setting. Estimating the unknowns (model parameters and true inputs) is a difficult task for nonlinear models and many techniques consist of substituting a single value for the unseen input (as done in regression calibration, moment reconstruction (Freedman et al., 2004)). In (Snoussi et al., 2002), the problem is addressed using a stochastic version of the EM algorithm, treating the true inputs as hidden variables. In (Dellaportas and Stephens, 1995), a Bayesian approach is taken to infer the unknown parameters using MCMC techniques, showing that, in this framework, the formulation for a Berkson-type model is the same as that for a classical model (the distinction being made otherwise).

In the machine learning community, the emphasis is not so much on recovering the true value of the missing or noisy data but on the estimation of the parameters of the model when data is missing. In (Ahmad and Tresp, 1993), they show that closed form solutions exist when using Gaussian basis function networks and, in the case of noisy features, the solution depends on the form of the noise. Also, mixture models were proved to deal naturally with missing features, by a dual use of the EM algorithm (Ghahramani and Jordan, 1994a). For a feed-forward neural network, the missing features have to be integrated out in order to compute the likelihood of the data, thus requiring a model of the input density. In (Tresp et al., 1994), they estimate the unknown input distribution directly from the data using a Gaussian mixture model. In this chapter, the new input is assumed to be corrupted by a white noise and we assume that is has a Gaussian probability distribution.

3.1.2 Motivation

As we will see in greater detail in the next chapter, our interest in the prediction at a noisy input is motivated by the iterative multiple-step-ahead prediction of time-series. Let $y_1, \dots, y_t$ be the observed time-series. Assuming the simple model $y_{t+1} = f(y_t)$, we wish to predict the value of the time-series at, say, time $t + k$. Having formed the input/output pairs (where the input now corresponds to a delayed value of the time-series), we can train a GP to learn the mapping $f(.)$. With this one-step ahead model, the prediction of y_{t+k} is done by iterating the model up to k, i.e. by predicting y_{t+1}, y_{t+2}, and so on, up to y_{t+k}. Since the time-series is known up to time t, with the GP model, the predictive distribution of y_{t+1} is readily obtained: We have $p(y_{t+1}|\mathcal{D}, y_t) = \mathcal{N}(\mu(y_t), \sigma^2(y_t))$, as given by (2.14) evaluated at $\mathbf{x}_* = y_t$. For the next time-step, a naive approach would simply use $\mu(y_t)$ as an estimate for y_{t+1} and evaluate $p(y_{t+2}|\mathcal{D}, \hat{y}_{t+1}) = \mathcal{N}(\mu(\hat{y}_{t+1}), \sigma^2(\hat{y}_{t+1}))$, where $\hat{y}_{t+1} = \mu(y_t)$. As we will see, this approach is not advisable for two reasons: it is over-confident about the estimate (leading to predictions with unrealistically small uncertainties) and it is throwing away valuable information, namely, the variance associated with the estimate of y_{t+1}. If we wish to account for the uncertainty on y_{t+1}, we need to be able to evaluate $p(y_{t+2}|\mathcal{D}, y_{t+1})$, where $y_{t+1} \sim \mathcal{N}(\mu(y_t), \sigma^2(y_t))$. This means being able to derive the predictive distribution of y_{t+2} corresponding to the normally distributed random input y_{t+1}.

The necessity of being able to make a prediction at an uncertain or noisy input is also obviously relevant for static problems. In real experiments and applications, we use sensors and detectors that can be corrupted by many different sources of disturbances. We might then only observe a noise-corrupted version of the true input and/or the system senses the input imperfectly. Again, if the model does not account for this 'extra' uncertainty (as opposed to the uncertainty usually acknowledged on the observed outputs), the model is too confident, which is misleading and could potentially be dangerous if, say, the model's output were to be used in a decision-making process of a critical application. Note that, in the static case, the approach we suggest assumes prior knowledge of the input noise variance.

In the following section, we present the problem of making a prediction at a noisy input for the Gaussian Process model, and then highlight the analytical approximation that we take, leading to the computation of only the predictive mean and variance of the new predictive distribution.

3.2 Prediction at an uncertain input

We assume that, based on data $\mathcal{D} = \{\mathbf{x}_i, t_i\}_{i=1}^N$ (where the $\mathbf{x}_i$'s are noise-free), a zero-mean GP with covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$ has been used to model the input/output relationship, $t_i = y_i + \epsilon_i$, where $y_i = f(\mathbf{x}_i)$ and ϵ_i has zero-mean and variance v_t.

As we saw in Chapter 2, with this model, given a new 'test' input $\mathbf{x}$, and based on the observed data $\mathcal{D}$, the predictive distribution of the corresponding output $y = f(\mathbf{x})$ is readily obtained.[1] This distribution is Gaussian,[2] $p(y|\mathcal{D}, \mathbf{x}) = \mathcal{N}_y(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$, with mean and variance respectively given by

$$\begin{cases} \mu(\mathbf{x}) = \sum_{i=1}^{N} \beta_i C(\mathbf{x}, \mathbf{x}_i) \\ \sigma^2(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - \sum_{i,j=1}^{N} K_{ij}^{-1} C(\mathbf{x}, \mathbf{x}_i) C(\mathbf{x}, \mathbf{x}_j) \end{cases} \tag{3.1}$$

with $\boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{t}$. Figure 3.1 shows the predictive means (dashed line) and their 2σ error-bars (dotted lines) computed for 81 test inputs in $[1, 10]$. A Gaussian Process with zero-mean and Gaussian covariance function (given by equation (2.3)) was trained using only $N = 10$ input/output pairs (crosses), where the outputs correspond to noise corrupted versions of $f(x) = \sin(x)/x$ (the noise level is $v_t = 0.001$). Near the data points, the predictive variance (model uncertainty) is small, and increasing as the test inputs are moved far away from the training ones.

Now, let the new input be corrupted by some noise $\boldsymbol{\epsilon}_{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_x)$, such that $\mathbf{x} = \mathbf{u} + \boldsymbol{\epsilon}_{\mathbf{x}}$. That is, we wish to make a prediction at $\mathbf{x} \sim \mathcal{N}(\mathbf{u}, \boldsymbol{\Sigma}_x)$ and to do so, we need to integrate the predictive distribution $p(y|\mathcal{D}, \mathbf{x})$ over the input distribution[3]

$$p(y|\mathcal{D}, \mathbf{u}, \boldsymbol{\Sigma}_x) = \int p(y|\mathcal{D}, \mathbf{x}) p(\mathbf{x}|\mathbf{u}, \boldsymbol{\Sigma}_x) d\mathbf{x} \ , \tag{3.2}$$

where $p(y|\mathcal{D}, \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{x})}} \exp\left[-\frac{1}{2}\frac{(y-\mu(\mathbf{x}))^2}{\sigma^2(\mathbf{x})}\right]$. Since $p(y|\mathcal{D}, \mathbf{x})$ is a nonlinear function of $\mathbf{x}$, the

[1] Note that in the previous chapter we denoted the new input by $\mathbf{x}_*$ and the corresponding function output by $f(\mathbf{x}_*)$ but we change here our notation for simplicity.

[2] The notation $y \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$ indicates that y is 'distributed as', that is, $p(y) = \mathcal{N}_y(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$, for $y \in]-\infty, +\infty[$, where $\mathcal{N}_y$ is the normalised probability density.

[3] When the bounds are not indicated, it is assumed that the integrals are evaluated from $-\infty$ to $+\infty$.

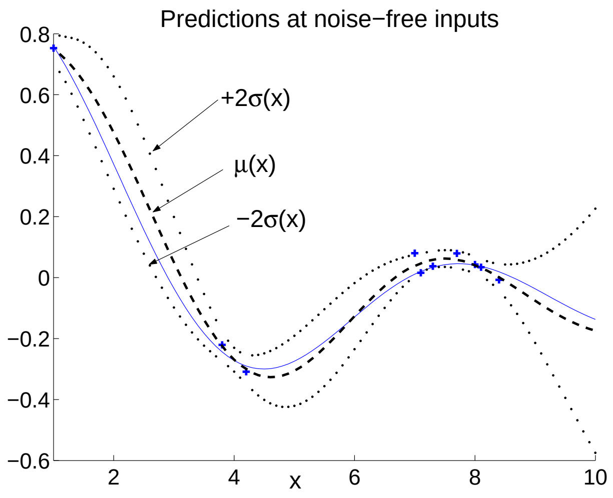

Figure 3.1: Predictive means (dashed line) and 2σ error-bars (dotted lines) corresponding to 81 noise-free test inputs. A zero-mean GP was trained on 10 training points (crosses) to learn the underlying function (continuous line).

new predictive distribution $p(y|\mathcal{D}, \mathbf{u}, \mathbf{\Sigma}_x)$ is not Gaussian and this integral cannot be solved without resorting to approximations.

3.2.1 Possible approximations

Many techniques are available to approximate intractable integrals of this kind. Approximation methods are divided into deterministic approximations and Monte-Carlo numerical methods. The most popular deterministic approaches are variational methods,[4] Laplace's method and Gaussian quadrature that consist of analytical approximations of the integral. We refer to (Mackay, 2003) for a review of these methods.

Numerical methods relying on Markov-Chain Monte-Carlo sampling techniques evaluate the integral numerically, thus approximating the true distribution (see e.g. (Neal, 1993)). In our case, the numerical approximation by simple Monte-Carlo is straightforward since we simply need to sample from a Gaussian distribution $\mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x)$. For each sample $\mathbf{x}^t$ from this distribution, $p(y|\mathcal{D}, \mathbf{x}^t)$ is

[4]See http://www.gatsby.ucl.ac.uk/vbayes/ for references.

normal, with mean and variance given by equations (3.1):

$$p(y|\mathcal{D},\mathbf{u},\mathbf{\Sigma}_x) \simeq \frac{1}{T}\sum_{t=1}^{T} p(y|\mathcal{D},\mathbf{x}^t) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{N}_y(\mu(\mathbf{x}^t),\sigma^2(\mathbf{x}^t)) \, . \tag{3.3}$$

The numerical approximation of $p(y|\mathcal{D},\mathbf{u},\mathbf{\Sigma}_x)$ is then a mixture of T Gaussians with identical mixing proportions, and as the number of samples T grows, the approximate distribution will tend to the true distribution. We refer to (Titterington et al., 1985) for a review of finite mixture models.

In Figure 3.2, the 'true' test inputs are $u = 2$ (left) and and $u = 6$ (right) but we observe x (asterisks). For 100 samples x^t from $p(x)$, centred at the noisy x with variance $v_x = 1$, we compute the corresponding predictive means $\mu(\mathbf{x}^t)$ (crosses) and their error-bars $\pm 2\sigma(x^t)$ (dots).

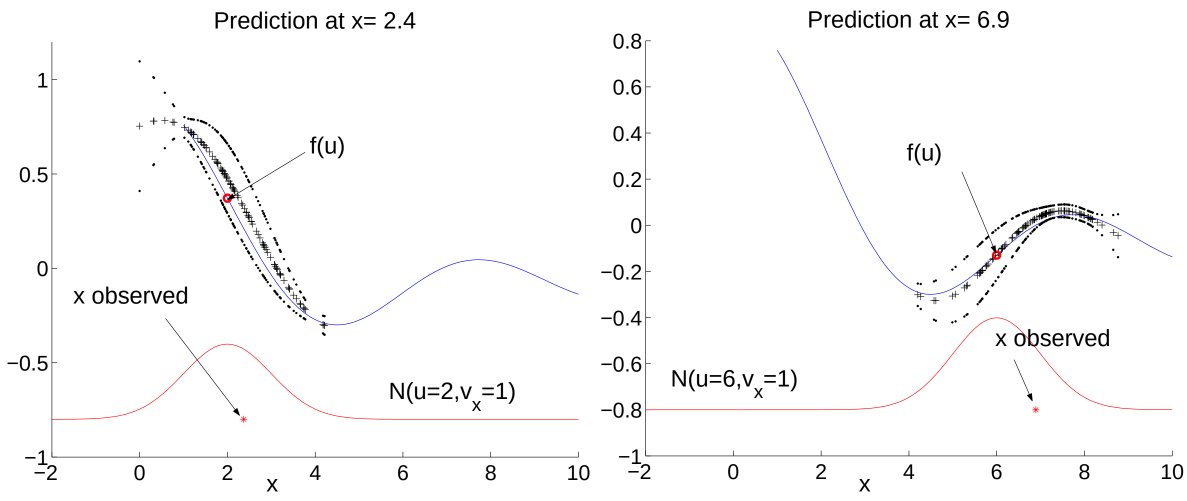

Figure 3.2: Monte-Carlo approximation for the prediction at a noisy input x (asterisk). The true input distribution is $x \sim \mathcal{N}_x(u, v_x)$, for $u = 2$ (left), $u = 6$ (right) and $v_x = 1$ (circles indicate the output $f(u)$ corresponding to the noise-free u). For 100 samples x^t from $p(x)$, with mean x and variance v_x, we compute $\mu(x^t)$ (crosses) and the associated error-bars $\pm 2\sigma(x^t)$ (dots).

The histograms of the samples at which predictions are made are shown in Figure 3.3. The circle and asterisk indicate the noise-free u and noisy inputs x respectively. After having computed the losses (squared error and negative log-predictive density) of the predictions associated with each x^t, we find the input value for which the loss is minimum (indicated by a triangle). Note how closer to the true input this x^t is, compared to the observed input.

We now focus on an analytical approximation which consists of computing only the first two

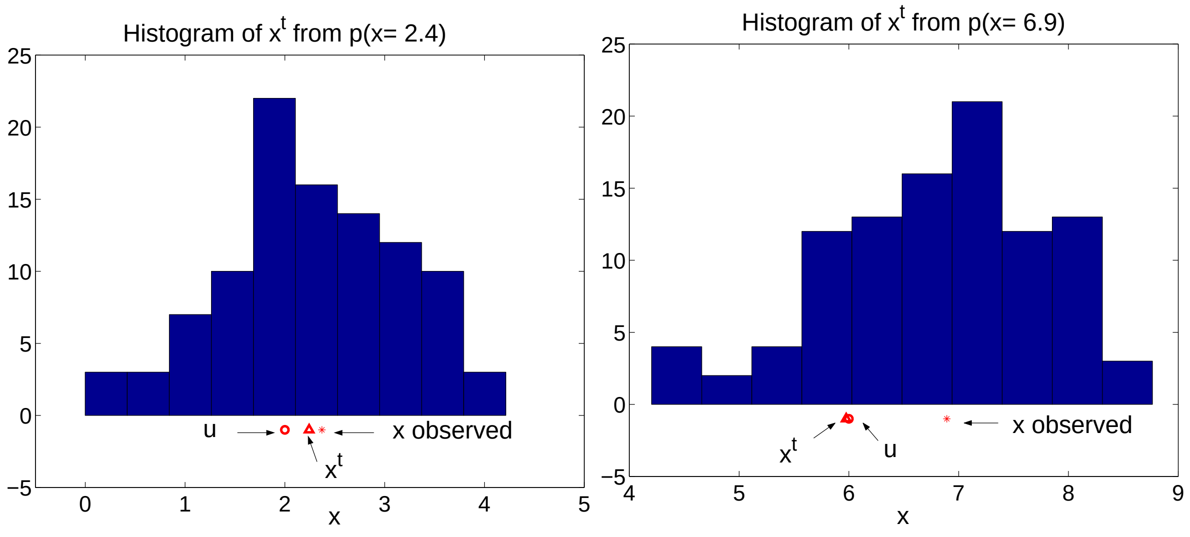

Figure 3.3: Histogram of the samples x^t from $p(x)$ at which predictions were made, when the true input (circle) is $u = 2$ (left) and $u = 6$ (right). Also plotted, the observed noisy input (asterisk), taken as the mean of $p(x)$, and the sample x^t that leads to the minimum loss (triangle).

moments, the mean and variance, of $p(y|\mathcal{D}, \mathbf{u}, \mathbf{\Sigma}_x)$.

3.2.2 Analytical approximation

To distinguish from $\mu(\mathbf{u})$ and $\sigma^2(\mathbf{u})$, the mean and variance of the Gaussian predictive distribution $p(y|\mathcal{D}, \mathbf{u})$ in the noise-free case, we denote by $m(\mathbf{u}, \mathbf{\Sigma}_x)$ the mean and by $v(\mathbf{u}, \mathbf{\Sigma}_x)$ the variance of the non-Gaussian predictive distribution $p(y|\mathcal{D}, \mathbf{u}, \mathbf{\Sigma}_x)$, corresponding to $\mathbf{x} \sim \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x)$. This can be interpreted as a *Gaussian approximation*, such that

$$p(y|\mathcal{D}, \mathbf{u}, \mathbf{\Sigma}_x) \approx \mathcal{N}(m(\mathbf{u}, \mathbf{\Sigma}_x), v(\mathbf{u}, \mathbf{\Sigma}_x)) \, .$$

From (3.2), the mean and variance are respectively given by

$$\begin{aligned} m(\mathbf{u}, \mathbf{\Sigma}_x) &= \int y \left\{ \int p(y|\mathcal{D}, \mathbf{x}) p(\mathbf{x}|\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \right\} dy \\ v(\mathbf{u}, \mathbf{\Sigma}_x) &= \int y^2 \left\{ \int p(y|\mathcal{D}, \mathbf{x}) p(\mathbf{x}|\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \right\} dy - m(\mathbf{u}, \mathbf{\Sigma}_x)^2 \, , \end{aligned}$$

where we have

$$\begin{aligned} \int y p(y|\mathcal{D}, \mathbf{x}) dy &= \mu(\mathbf{x}) \\ \int y^2 p(y|\mathcal{D}, \mathbf{x}) dy &= \sigma^2(\mathbf{x}) + \mu(\mathbf{x})^2 \, . \end{aligned}$$

We can then write

$$\begin{aligned} m(\mathbf{u},\mathbf{\Sigma}_x) &= \int \mu(\mathbf{x})p(\mathbf{x}|\mathbf{u},\mathbf{\Sigma}_x)d\mathbf{x} \\ v(\mathbf{u},\mathbf{\Sigma}_x) &= \int \sigma^2(\mathbf{x})p(\mathbf{x}|\mathbf{u},\mathbf{\Sigma}_x)d\mathbf{x} + \int \mu(\mathbf{x})^2 p(\mathbf{x}|\mathbf{u},\mathbf{\Sigma}_x)d\mathbf{x} - m(\mathbf{u},\mathbf{\Sigma}_x)^2 \end{aligned}$$

or, for short,[5]

$$m(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu(\mathbf{x})] \tag{3.4}$$

$$v(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\sigma^2(\mathbf{x})] + E_{\mathbf{x}}[\mu(\mathbf{x})^2] - m(\mathbf{u},\mathbf{\Sigma}_x)^2 \ . \tag{3.5}$$

Replacing $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ by their expressions, we have

$$E_{\mathbf{x}}[\mu(\mathbf{x})] = \sum_{i=1}^N \beta_i E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)] \tag{3.6}$$

$$E_{\mathbf{x}}[\sigma^2(\mathbf{x})] = E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x})] - \sum_{i,j=1}^N K_{ij}^{-1} E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)C(\mathbf{x},\mathbf{x}_j)] \tag{3.7}$$

$$E_{\mathbf{x}}[\mu(\mathbf{x})^2] = \sum_{i,j=1}^N \beta_i\beta_j E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)C(\mathbf{x},\mathbf{x}_j)] \ . \tag{3.8}$$

The new predictive mean and variance are then given by

$$\boxed{\begin{aligned} m(\mathbf{u},\mathbf{\Sigma}_x) &= \sum_{i=1}^N \beta_i E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)] \\ v(\mathbf{u},\mathbf{\Sigma}_x) &= E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x})] - \sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j) E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)C(\mathbf{x},\mathbf{x}_j)] - m(\mathbf{u},\mathbf{\Sigma}_x)^2 \end{aligned}} \tag{3.9}$$

Let

$$l = E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x})] = \int C(\mathbf{x},\mathbf{x})p(\mathbf{x})d\mathbf{x} \tag{3.10}$$

$$l_i = E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)] = \int C(\mathbf{x},\mathbf{x}_i)p(\mathbf{x})d\mathbf{x} \tag{3.11}$$

$$l_{ij} = E_{\mathbf{x}}[C(\mathbf{x},\mathbf{x}_i)C(\mathbf{x},\mathbf{x}_j)] = \int C(\mathbf{x},\mathbf{x}_i)C(\mathbf{x},\mathbf{x}_j)p(\mathbf{x})d\mathbf{x} \ , \tag{3.12}$$

where $p(\mathbf{x})$ is the input noise distribution.

How solvable integrals (3.10)-(3.12) are basically depends on the form of the covariance function.

[5]Note these equations could have been readily obtained using the law of iterated expectations and conditional variances (see appendix A).

1. If the covariance function is e.g. linear, Gaussian, polynomial (or a mixture of those), we can compute these integrals exactly and obtain the *exact* mean and variance. In Section 3.4, we derive the exact moments for the linear and Gaussian covariance functions.

2. Otherwise, we can again approximate (3.10)-(3.12) in a number of ways. Since we are mostly interested in closed form approximate solutions, we evaluate the integrals within a Taylor approximation of the covariance function around the mean $\mathbf{u}$ of $\mathbf{x}$ and obtain the *approximate* mean and variance.

Note that this second case, that requires approximations, might be required, if the form of the covariance function is definitely one for which one cannot solve the integrals exactly, or simply preferable, if the integrals are tractable but at the cost of long and tedious calculations. Also, assuming one has access to software like Mathematica or Matlab's symbolic toolbox to compute the derivatives, the solutions obtained using the proposed approximation provide a suitable performance/implementation trade-off.

Figure 3.4 schematises the situation and highlights the analytical approximation we take. We now turn to the evaluation of the mean and variance in the case where the covariance function is such that approximations are needed to evaluate integrals (3.10)-(3.12) analytically.

3.3 *Gaussian approximation*: Approximate moments

We are going to approximate these integrals analytically to obtain approximate moments, based on a Taylor approximation of the covariance function.

3.3.1 Delta method

We use the Delta method (also called Moment Approximation), which consists of approximating the integrand by a Taylor polynomial. In the one-dimensional case, the Delta method is stated as follows (Lindley, 1969; Papoulis, 1991): Let x be a random variable with mean $E_x[x] = u$ and variance

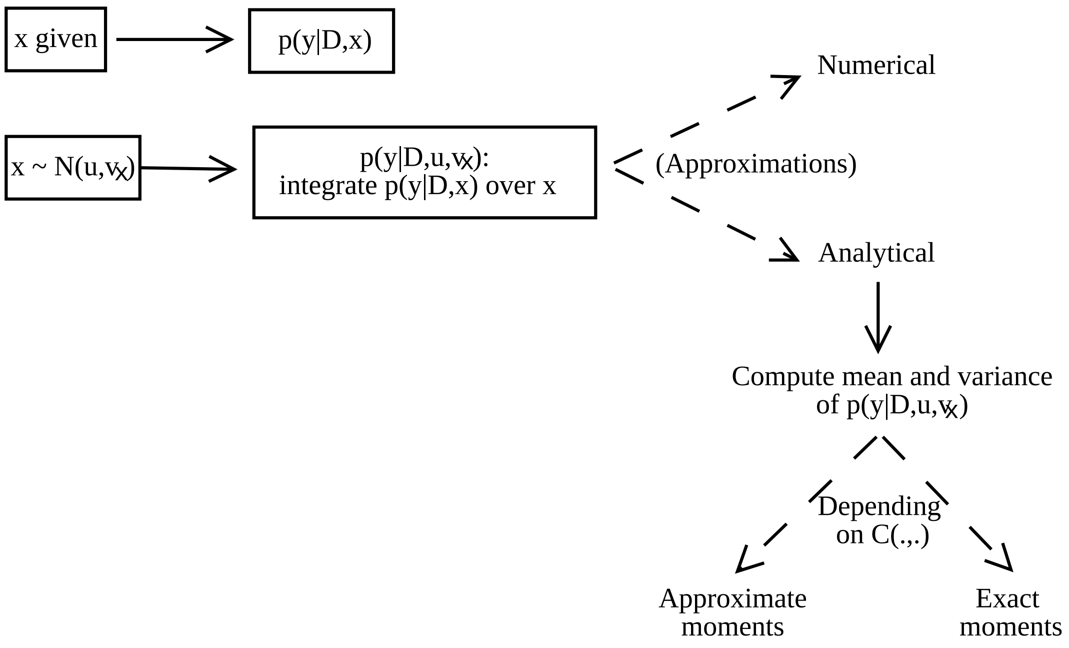

Figure 3.4: Dealing with a noisy test input: With the GP model, given the data D and a new input x, the predictive distribution of the corresponding output $y = f(x)$ is readily obtained. When x is noisy, or $x \sim \mathcal{N}(u, v_x)$, the corresponding predictive distribution is obtained by integration over x. Since $p(y|D,x)$ is nonlinear in x, the integral is analytically intractable. Although a numerical approximation of the integral is possible, we concentrate on an analytical approximation. We suggest computing the mean and the variance of the new predictive distribution, which is done exactly or approximately, depending on the parametric form of the covariance function $C(.,.)$.

$\mathrm{Var}_x[x] = v_x$, and $y = \phi(x)$. For sufficiently small $\sigma_x = \sqrt{v_x}$ and well-behaved ϕ we can write

$$E_x[y] \simeq \phi(u) + \frac{1}{2}v_x\phi''(u) \tag{3.13}$$

$$\mathrm{Var}_x[y] \simeq \phi'(u)^2 v_x \tag{3.14}$$

where ϕ' and ϕ'' are the first and second derivatives of ϕ evaluated at u.

These results are simply obtained by considering the expansion of $\phi(x)$ as Taylor series about u, up to the second order:

$$y = \phi(x) = \phi(u) + (x-u)\phi'(u) + \frac{1}{2}(x-u)^2\phi''(u) + O([(x-u)^3]) \, . \tag{3.15}$$

By taking the expectation on both sides, we directly find the approximation (3.13). For the variance,

$\mathrm{Var}_x[y] = E_x[y^2] - E_x[y]^2$, (3.14) corresponds to an approximation of the second order estimate:[6] Neglecting the term in v_x^2 for both $E[y^2]$ and $E[y]^2$, we have

$$\begin{aligned} E[y^2] &\approx \phi(u)^2 + v_x\phi'(u)^2 + \phi(u)\phi''(u)v_x \\ E[y]^2 &\approx \phi(u)^2 + \phi(u)\phi''(u)v_x \end{aligned}$$

leading to (3.14). This approximation is motivated by the fact that the Taylor approximation is useful for small standard deviations (if σ_x is small, by Chebychev's inequality $P(|x-u| > k\sigma_x) < \frac{1}{k^2}$), such that x will depart only a little from u except on rare occasions and therefore $(x-u)$ will be small (Lindley, 1969).

There are obviously conditions which $\phi(x)$ should fulfil to make the Taylor series possible (in the neighbourhood of u) and to avoid anomalies of behaviour away from u. As in (Lindley, 1969), we do not state such conditions and assume the covariance function to be such that the expressions are valid.

3.3.2 Approximate mean and variance

Let $m^{ap}(\mathbf{u}, \mathbf{\Sigma}_x)$ be the approximate mean, such that

$$m^{ap}(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_{i=1}^{N} \beta_i l_i^{ap}$$

with $l_i^{ap} = E_{\mathbf{x}}[C^{ap}(\mathbf{x}, \mathbf{x}_i)]$, and where $C^{ap}(\mathbf{x}, \mathbf{x}_i)$ corresponds to the second-order Taylor polynomial approximation of $C(\mathbf{x}, \mathbf{x}_i)$ around the mean $\mathbf{u}$ of $\mathbf{x}$,

$$C^{ap}(\mathbf{x}, \mathbf{x}_i) = C(\mathbf{u}, \mathbf{x}_i) + (\mathbf{x}-\mathbf{u})^T\mathbf{C}'(\mathbf{u}, \mathbf{x}_i) + \frac{1}{2}(\mathbf{x}-\mathbf{u})^T\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)(\mathbf{x}-\mathbf{u}) \ .$$

We directly have

$$l_i^{ap} = C(\mathbf{u}, \mathbf{x}_i) + \frac{1}{2}\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\mathbf{\Sigma}_x]$$

so that the approximate mean is

$$\boxed{m^{ap}(\mathbf{u}, \mathbf{\Sigma}_x) = \mu(\mathbf{u}) + \tfrac{1}{2}\textstyle\sum_{i=1}^{N}\beta_i \mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\mathbf{\Sigma}_x]} \tag{3.16}$$

[6]Note that (3.14) can also be seen as a first-order estimate: If we consider $y \approx \phi(u) + (x-u)\phi'(u)$, we have $E[y]^2 \simeq \phi(u)^2$ and

$$y^2 = \phi(u)^2 + 2\phi(u)(x-u)\phi'(u) + (x-u)^2\phi'(u)^2$$

leading to $E[y^2] = \phi(u)^2 + v_x\phi'(u)^2$ and therefore (3.14).

where $\mu(\mathbf{u}) = \sum_{i=1}^N \beta_i C(\mathbf{u}, \mathbf{x}_i)$ is the noise-free predictive mean computed at $\mathbf{u}$ (see Appendix B for a note on this approximation).

Similarly, the approximate variance is

$$v^{ap}(\mathbf{u}, \mathbf{\Sigma}_x) = l^{ap} - \sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j) l_{ij}^{ap} - m^{ap}(\mathbf{u}, \mathbf{\Sigma}_x)^2$$

with $l^{ap} = E_{\mathbf{x}}[C^{ap}(\mathbf{x}, \mathbf{x})]$ and $l_{ij}^{ap} = E_{\mathbf{x}}[C^{ap}(\mathbf{x}, \mathbf{x}_i)C^{ap}(\mathbf{x}, \mathbf{x}_j)]$, where $C^{ap}(.,.)$ is again the second order Taylor approximation of $C(.,.)$. We have

$$l^{ap} = C(\mathbf{u}, \mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{u})\mathbf{\Sigma}_x]$$

and

$$\begin{aligned} l_{ij}^{ap} \approx\; & C(\mathbf{u}, \mathbf{x}_i)C(\mathbf{u}, \mathbf{x}_j) + \mathrm{Tr}[\mathbf{C}'(\mathbf{u}, \mathbf{x}_i)\mathbf{C}'(\mathbf{u}, \mathbf{x}_j)^T\mathbf{\Sigma}_x] + \frac{1}{2}C(\mathbf{u}, \mathbf{x}_i)\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_j)\mathbf{\Sigma}_x] \\ & + \frac{1}{2}C(\mathbf{u}, \mathbf{x}_j)\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\mathbf{\Sigma}_x] \; , \end{aligned}$$

where the approximation comes from discarding terms of higher order than $\mathbf{\Sigma}_x$ in $C^{ap}(\mathbf{x}, \mathbf{x}_i)C^{ap}(\mathbf{x}, \mathbf{x}_j)$, as discussed in the previous section. Similarly, approximating $m^{ap}(\mathbf{u}, \mathbf{\Sigma}_x)^2$ by

$$\begin{aligned} m^{ap}(\mathbf{u}, \mathbf{\Sigma}_x)^2 \approx\; & \sum_{i,j=1}^N \beta_i\beta_j \Big(C(\mathbf{u}, \mathbf{x}_i)C(\mathbf{u}, \mathbf{x}_j) + \frac{1}{2}C(\mathbf{u}, \mathbf{x}_i)\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_j)\mathbf{\Sigma}_x] \\ & + \frac{1}{2}C(\mathbf{u}, \mathbf{x}_j)\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\mathbf{\Sigma}_x] \Big) \; , \end{aligned}$$

we find, after simplification,

$$\boxed{\begin{aligned} v^{ap}(\mathbf{u}, \mathbf{\Sigma}_x) =& \sigma^2(\mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{u})\mathbf{\Sigma}_x] - \sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j)\mathrm{Tr}[\mathbf{C}'(\mathbf{u}, \mathbf{x}_i)\mathbf{C}'(\mathbf{u}, \mathbf{x}_j)^T\mathbf{\Sigma}_x] \\ & - \frac{1}{2}\sum_{i,j=1}^N K_{ij}^{-1}(C(\mathbf{u}, \mathbf{x}_i)\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_j)\mathbf{\Sigma}_x] + C(\mathbf{u}, \mathbf{x}_j)\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\mathbf{\Sigma}_x]) \end{aligned}} \tag{3.17}$$

where $\sigma^2(\mathbf{u}) = C(\mathbf{u}, \mathbf{u}) - \sum_{i,j=1}^N K_{ij}^{-1} C(\mathbf{u}, \mathbf{x}_i)C(\mathbf{u}, \mathbf{x}_j)$ is the noise-free predictive variance.

Both approximate mean and variance are composed of the noise-free predictive moments plus correction terms. If $\mathbf{\Sigma}_x$ is assumed diagonal, these correction terms consist of the sum of the derivatives of the covariance function in each input dimension, weighted by the variance of the new test input in the same direction.

Figure 3.5 illustrates these results. The noise-free inputs are $u = 2, 6$ and 9.5 but we observe $x = 2.4, 6.9$ and 9.2 (asterisks), sampled from $p(x) = \mathcal{N}_x(u, v_x)$ with $v_x = 1$ (distributions plotted at $y = -0.8$). The circles indicate the function output $f(u)$ corresponding to the noise-free u's. The approximate means $m^{ap}(u, v_x)$ and associated uncertainties $\pm 2\sqrt{v}^{ap}(u, v_x)$ are plotted as triangles and dotted lines. We can compare them to the *naive* (noise-free) means, $\mu(u)$, with error-bars $\pm 2\sigma(u)$, which do not account for the noise on the input. The right hand-side displays plots of the covariance function (which is the squared exponential given by (2.3)) and its first and second derivatives, which appear in the expressions for the approximate mean and variance (the continuous lines correspond to the covariances at $x = 2.4$ and inputs in $[1, 10]$ and the dashed lines to those at $x = 6.9$), the circles indicating the covariances with the training points.

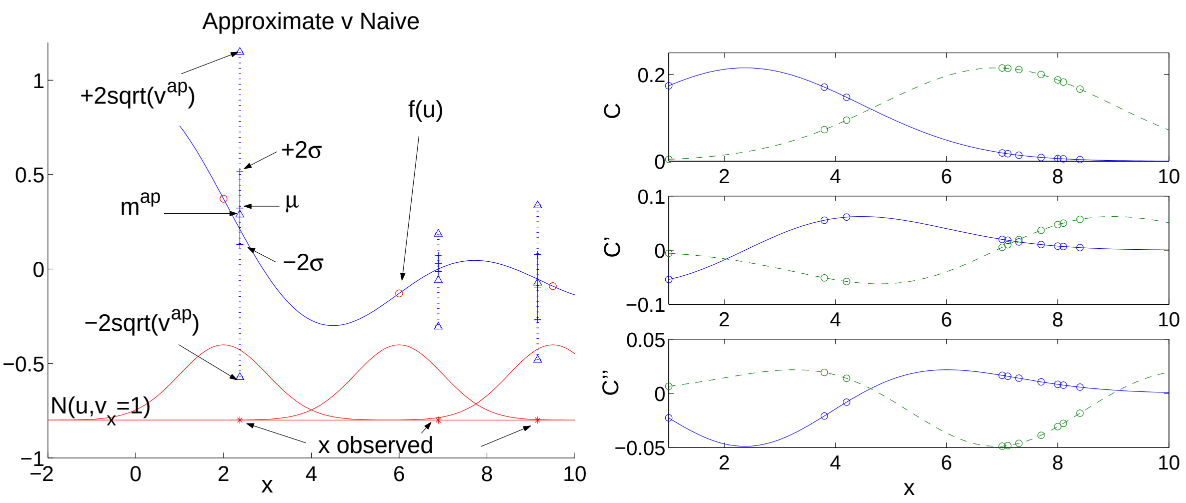

Figure 3.5: *Gaussian approximation* to the prediction at x (asterisk), noisy version of the true $u = 2, 6, 9.5$, where the noise has variance $v_x = 1$. Left: Approximate mean $m^{ap}(x)$ and uncertainty $\pm 2\sqrt{v^{ap}(x)}$ (triangles). The noise-free moments $(\mu(x) \pm 2\sigma(x))$ are indicated by crosses and the circles show the function outputs corresponding to the noise-free u's. Right: From top to bottom, covariance function and derivatives, between $x \in [1, 10]$ and the noisy test inputs ($x = 2.4$, continuous line, and $x = 6.9$, dashed line), the circles indicating the covariances with the training inputs.

3.3.3 Approximating $\mu(\mathrm{x})$ and $\sigma^2(\mathrm{x})$ directly

In (Girard et al., 2003; Girard et al., 2002), we derived the approximate mean and variance by solving equations (3.4) and (3.5) directly, that is replacing $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ by their first and second order Taylor approximations respectively. Applying (3.13) and (3.14) directly to $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$, we have

$$\begin{aligned} E_{\mathbf{x}}[\mu(\mathbf{x})] &\simeq \mu(\mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\boldsymbol{\mu}''(\mathbf{u})\boldsymbol{\Sigma}_x] && (3.18)\\ \mathrm{Var}_{\mathbf{x}}[\mu(\mathbf{x})] &\simeq \mathrm{Tr}[\boldsymbol{\mu}'(\mathbf{u})\boldsymbol{\mu}'(\mathbf{u})^T\boldsymbol{\Sigma}_x] && (3.19)\\ E_{\mathbf{x}}[\sigma^2(\mathbf{x})] &\simeq \sigma^2(\mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\boldsymbol{\sigma}^{2''}(\mathbf{u})\boldsymbol{\Sigma}_x]\,. && (3.20)\end{aligned}$$

Substituting into $m(\mathbf{u}, \boldsymbol{\Sigma}_x) = E_{\mathbf{x}}[\mu(\mathbf{x})]$, we have

$$m(\mathbf{u}, \boldsymbol{\Sigma}_x) = \mu(\mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\boldsymbol{\mu}''(\mathbf{u})\boldsymbol{\Sigma}_x]$$

with $\mu(\mathbf{u}) = \sum_i \beta_i C(\mathbf{u}, \mathbf{x}_i)$ and

$$\boldsymbol{\mu}'(\mathbf{u}) = \sum_i \beta_i \mathbf{C}'(\mathbf{u}, \mathbf{x}_i)\,, \quad \boldsymbol{\mu}''(\mathbf{u}) = \sum_i \beta_i \mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\,,$$

so that we can write

$$m(\mathbf{u}, \boldsymbol{\Sigma}_x) = \sum_i \beta_i \left(C(\mathbf{u}, \mathbf{x}_i) + \frac{1}{2}\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i)\boldsymbol{\Sigma}_x] \right)\,. \qquad (3.21)$$

Similarly, we have $v(\mathbf{u}, \boldsymbol{\Sigma}_x) = E_{\mathbf{x}}[\sigma^2(\mathbf{x})] + E_{\mathbf{x}}[\mu(\mathbf{x})^2] - m(\mathbf{u}, \boldsymbol{\Sigma}_x)^2 = E_{\mathbf{x}}[\sigma^2(\mathbf{x})] + \mathrm{Var}_{\mathbf{x}}[\mu(\mathbf{x})]$, that is

$$v(\mathbf{u}, \boldsymbol{\Sigma}_x) = \sigma^2(\mathbf{u}) + \mathrm{Tr}\left[\left(\frac{1}{2}\boldsymbol{\sigma}^{2''}(\mathbf{u}) + \boldsymbol{\mu}'(\mathbf{u})\boldsymbol{\mu}'(\mathbf{u})^T \right) \boldsymbol{\Sigma}_x \right]\,,$$

with $\sigma^2(\mathbf{u}) = C(\mathbf{u}, \mathbf{u}) - \sum_{ij} K_{ij}^{-1} C(\mathbf{u}, \mathbf{x}_i) C(\mathbf{u}, \mathbf{x}_j)$ and

$$\begin{aligned} \boldsymbol{\sigma}^{2'}(\mathbf{u}) &= \mathbf{C}'(\mathbf{u}, \mathbf{u}) - \sum_{ij} K_{ij}^{-1}[\mathbf{C}'(\mathbf{u}, \mathbf{x}_i) C(\mathbf{u}, \mathbf{x}_j) + C(\mathbf{u}, \mathbf{x}_i)\mathbf{C}'(\mathbf{u}, \mathbf{x}_j)]\\ \boldsymbol{\sigma}^{2''}(\mathbf{u}) &= \mathbf{C}''(\mathbf{u}, \mathbf{u}) - \sum_{ij} K_{ij}^{-1}[\mathbf{C}''(\mathbf{u}, \mathbf{x}_i) C(\mathbf{u}, \mathbf{x}_j) + C(\mathbf{u}, \mathbf{x}_i)\mathbf{C}''(\mathbf{u}, \mathbf{x}_j)\\ &\quad +2\mathbf{C}'(\mathbf{u}, \mathbf{x}_i)\mathbf{C}'(\mathbf{u}, \mathbf{x}_j)^T]\,. \end{aligned}$$

After simplification, we obtain

$$\begin{aligned} v(\mathbf{u}, \boldsymbol{\Sigma}_x) =& C(\mathbf{u}, \mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\mathbf{C}''(\mathbf{u}, \mathbf{u})\boldsymbol{\Sigma}_x] - \sum_{ij} K_{ij}^{-1}(C(\mathbf{u}, \mathbf{x}_i) C(\mathbf{u}, \mathbf{x}_j)\\ & + \frac{1}{2}\mathrm{Tr}[(\mathbf{C}''(\mathbf{u}, \mathbf{x}_i) C(\mathbf{u}, \mathbf{x}_j) + C(\mathbf{u}, \mathbf{x}_i)\mathbf{C}''(\mathbf{u}, \mathbf{x}_j))\boldsymbol{\Sigma}_x]) \qquad (3.22)\\ & - \sum_{ij}(K_{ij}^{-1} - \beta_i\beta_j)\mathrm{Tr}[\mathbf{C}'(\mathbf{u}, \mathbf{x}_i)\mathbf{C}'(\mathbf{u}, \mathbf{x}_j)^T\boldsymbol{\Sigma}_x]\,. \end{aligned}$$

Obviously, the Taylor approximation of $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ implies a second order Taylor approximation of the covariance function. Although these results are the same as those obtained when working with the covariance function, this approach lacks flexibility in that it does not highlight the dependence on the form of the covariance function and it is therefore not clear that exact moments can be computed in particular cases, as we will now illustrate.

3.4 *Gaussian approximation*: Exact moments

In the special cases of the linear and the Gaussian (squared exponential) covariance functions, we can evaluate integrals (3.10)-(3.12) exactly.

3.4.1 Case of the linear covariance function

Let us write the linear covariance function as $C_L(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{L} \mathbf{x}_j$ where $\mathbf{L} = \mathrm{diag}[\alpha_1 \ldots \alpha_D]$. In the noise-free case, the prediction at $\mathbf{u}$ leads to a Gaussian distribution with mean and variance

$$\begin{cases} \mu_L(\mathbf{u}) = \sum_{i=1}^{N} \beta_i C_L(\mathbf{u}, \mathbf{x}_i) \\ \sigma_L^2(\mathbf{u}) = C_L(\mathbf{u}, \mathbf{u}) - \sum_{i,j=1}^{N} K_{ij}^{-1} C_L(\mathbf{u}, \mathbf{x}_i) C_L(\mathbf{u}, \mathbf{x}_j) \ . \end{cases} \tag{3.23}$$

When we are predicting at a noisy input, the predictive mean and variance, now denoted by m^{ex_L} and v^{ex_L} (the subscript indicating the exact linear case), are given by

$$m^{ex_L}(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_{i=1}^{N} \beta_i l_i^{ex_L} \tag{3.24}$$

$$v^{ex_L}(\mathbf{u}, \mathbf{\Sigma}_x) = l^{ex_L} - \sum_{i,j=1}^{N} (K_{ij}^{-1} - \beta_i \beta_j) l_{ij}^{ex_L} - m^{ex_L}(\mathbf{u}, \mathbf{\Sigma}_x)^2 \tag{3.25}$$

where

$$\begin{aligned} l^{ex_L} &= E_{\mathbf{x}}[C_L(\mathbf{x}, \mathbf{x})] = \int \mathbf{x}^T \mathbf{L} \mathbf{x} \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \\ l_i^{ex_L} &= E_{\mathbf{x}}[C_L(\mathbf{x}, \mathbf{x}_i)] = \int \mathbf{x}^T \mathbf{L} \mathbf{x}_i \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \\ l_{ij}^{ex_L} &= E_{\mathbf{x}}[C_L(\mathbf{x}, \mathbf{x}_i) C_L(\mathbf{x}, \mathbf{x}_j)] = \int \mathbf{x}^T \mathbf{L} \mathbf{x}_i \mathbf{x}^T \mathbf{L} \mathbf{x}_j \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \ . \end{aligned}$$

Using the formula giving the expectation of a quadratic form under a Gaussian (see Appendix A), we directly obtain

$$\begin{aligned} l^{ex_L} &= \mathbf{u}^T\mathbf{L}\mathbf{u} + \mathrm{Tr}[\mathbf{L}\boldsymbol{\Sigma}_x] = C_L(\mathbf{u},\mathbf{u}) + \mathrm{Tr}[\mathbf{L}\boldsymbol{\Sigma}_x] \\ l_i^{ex_L} &= \mathbf{u}^T\mathbf{L}\mathbf{x}_i = C_L(\mathbf{u},\mathbf{x}_i) \\ l_{ij}^{ex_L} &= \mathbf{u}^T(\mathbf{L}\mathbf{x}_i\mathbf{x}_j^T\mathbf{L})\mathbf{u} + \mathrm{Tr}[\mathbf{L}\mathbf{x}_i\mathbf{x}_j^T\mathbf{L}\boldsymbol{\Sigma}_x] = C_L(\mathbf{u},\mathbf{x}_i)C_L(\mathbf{x}_j,\mathbf{u}) + \mathrm{Tr}[\mathbf{L}\mathbf{x}_i\mathbf{x}_j^T\mathbf{L}\boldsymbol{\Sigma}_x] \,. \end{aligned}$$

In the linear case, the new predictive mean is then the same as the noise-free one, as we have

$$\boxed{m^{ex_L}(\mathbf{u},\boldsymbol{\Sigma}_x) = \textstyle\sum_{i=1}^N \beta_i C_L(\mathbf{u},\mathbf{x}_i)} \tag{3.26}$$

On the other hand, the variance becomes

$$\begin{aligned} v^{ex_L}(\mathbf{u},\boldsymbol{\Sigma}_x) = C_L(\mathbf{u},\mathbf{u}) + \mathrm{Tr}[\mathbf{L}\boldsymbol{\Sigma}_x] - \sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j)\mathrm{Tr}[\mathbf{L}\mathbf{x}_i\mathbf{x}_j^T\mathbf{L}\boldsymbol{\Sigma}_x]) \\ - \sum_{i,j=1}^N K_{ij}^{-1} C_L(\mathbf{u},\mathbf{x}_i)C_L(\mathbf{x}_j,\mathbf{u}) \end{aligned} \tag{3.27}$$

after simplification of the $\beta_i\beta_j$ terms. Alternatively, in terms of the noise-free variance $\sigma_L^2(\mathbf{u})$,

$$v^{ex_L}(\mathbf{u},\boldsymbol{\Sigma}_x) = \sigma_L^2(\mathbf{u}) + \mathrm{Tr}[\mathbf{L}\boldsymbol{\Sigma}_x] - \sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j)\mathrm{Tr}[\mathbf{L}\mathbf{x}_i\mathbf{x}_j^T\mathbf{L}\boldsymbol{\Sigma}_x]) \,. \tag{3.28}$$

If we note that $\mathbf{C}'_L(\mathbf{u},\mathbf{x}_i) = \frac{\partial C_L(\mathbf{u},\mathbf{x}_i)}{\partial \mathbf{u}} = \mathbf{L}\mathbf{x}_i$ and $\mathbf{C}''_L(\mathbf{u},\mathbf{u}) = \frac{\partial^2 C_L(\mathbf{u},\mathbf{u})}{\partial\mathbf{u}\partial\mathbf{u}^T} = 2\mathbf{L}$, we can also write it as

$$\boxed{v^{ex_L}(\mathbf{u},\boldsymbol{\Sigma}_x) = \sigma_L^2(\mathbf{u}) + \tfrac{1}{2}\mathrm{Tr}[\mathbf{C}''_L(\mathbf{u},\mathbf{u})\boldsymbol{\Sigma}_x] - \textstyle\sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j)\mathrm{Tr}[\mathbf{C}'_L(\mathbf{x},\mathbf{x}_i)\mathbf{C}'_L(\mathbf{x},\mathbf{x}_j)^T\boldsymbol{\Sigma}_x])} \tag{3.29}$$

As we would expect, in this case, the predictive mean and variance exactly correspond to the approximate moments we would obtain within a first order approximation of the covariance function.

3.4.2 Case of the Gaussian covariance function

As noted in the previous chapter, the Gaussian covariance function

$$C_G(\mathbf{x}_i,\mathbf{x}_j) = v\exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right] \tag{3.30}$$

is of special importance as it has been shown that a GP with such a covariance function performed as well as other popular nonlinear models like neural networks (Rasmussen, 1996a).

Let $\mu_G(\mathbf{u})$ and $\sigma^2_G(\mathbf{u})$ denote the noise-free predictive mean and variance,

$$\begin{cases} \mu_G(\mathbf{u}) = \sum_{i=1}^{N} \beta_i C_G(\mathbf{u}, \mathbf{x}_i) \\ \sigma^2_G(\mathbf{u}) = C_G(\mathbf{u}, \mathbf{u}) - \sum_{i,j=1}^{N} K^{-1}_{ij} C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) \end{cases} \tag{3.31}$$

where, according to (3.30), $C_G(\mathbf{u}, \mathbf{u}) = v$. Now, for predicting at $\mathbf{x} \sim \mathcal{N}(\mathbf{u}, \mathbf{\Sigma}_x)$, we need to compute

$$m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_{i=1}^{N} \beta_i l^{exG}_i \tag{3.32}$$

$$v^{exG}(\mathbf{u}, \mathbf{\Sigma}_x) = l^{exG} - \sum_{i,j=1}^{N} (K^{-1}_{ij} - \beta_i \beta_j) l^{exG}_{ij} - m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x)^2 \ , \tag{3.33}$$

where we directly have $l^{exG} = E_{\mathbf{x}}[C_G(\mathbf{x}, \mathbf{x})] = v = C_G(\mathbf{u}, \mathbf{u})$, and

$$l^{exG}_i = E_{\mathbf{x}}[C_G(\mathbf{x}, \mathbf{x}_i)] \ , \quad l^{exG}_{ij} = E_{\mathbf{x}}[C_G(\mathbf{x}, \mathbf{x}_i) C_G(\mathbf{x}, \mathbf{x}_j)] \ .$$

For notational convenience, let us write the Gaussian covariance function as[7] $C_G(\mathbf{x}_i, \mathbf{x}_j) = c N_{\mathbf{x}_i}(\mathbf{x}_j, \mathbf{W})$, with $c = (2\pi)^{D/2} |\mathbf{W}|^{1/2} v$. Using the product of Gaussians formula (see Appendix A), we directly have

$$l^{exG}_i = c \int N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W}) \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} = c N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x) \ . \tag{3.34}$$

For the evaluation of l^{exG}_{ij}, we need to use this product twice, leading to

$$l^{exG}_{ij} = c^2 \int N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W}) N_{\mathbf{x}}(\mathbf{x}_j, \mathbf{W}) \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \tag{3.35}$$

$$= c^2 N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) \int N_{\mathbf{x}} \left(\frac{\mathbf{x}_i + \mathbf{x}_j}{2}, \frac{\mathbf{W}}{2} \right) \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) d\mathbf{x} \tag{3.36}$$

$$= c^2 N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}} \left(\frac{\mathbf{x}_i + \mathbf{x}_j}{2}, \mathbf{\Sigma}_x + \frac{\mathbf{W}}{2} \right) \ . \tag{3.37}$$

[7] Although we write $C_G(\mathbf{x}_i, \mathbf{x}_j)$ as a normalised probability density, we denote it by N since the variables involved are not random.

Exact predictive mean

Replacing l_i^{exG} by its expression in $m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x)$, we have

$$m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_{i=1}^{N} \beta_i c N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x) \ , \tag{3.38}$$

and we can then directly check that, as we would expect, $m(\mathbf{u}, \mathbf{\Sigma}_x = \mathbf{0}) = \mu_G(\mathbf{u})$. With $\mathbf{W}^{-1} = \mathrm{diag}[w_1, \dots, w_D]$, and assuming a diagonal $\mathbf{\Sigma}_x$, $\mathbf{\Sigma}_x = \mathrm{diag}[v_{x1}, \dots, v_{xD}]$, the d^{th} element on the diagonal of $(\mathbf{W} + \mathbf{\Sigma}_x)^{-1}$ is $\frac{w_d}{1+w_d v_{xd}}$. Recall that, in the noise-free case, w_d relates to the variation in direction d (linked to the correlation length). Now, this length-scale is 'widened', proportionally to the noise variance. Also, the vertical amplitude of variation, formally controlled by v, is now accordingly weighted down by $|\mathbf{I} + \mathbf{W}^{-1}\mathbf{\Sigma}_x|^{-1/2}$. It is an overall *flattening* phenomenon, with an increased correlation length and decreased vertical amplitude.

It is useful to write $m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x)$ as a *corrected* version of $\mu_G(\mathbf{u})$. Using the matrix inversion lemma, we have $(\mathbf{W} + \mathbf{\Sigma}_x)^{-1} = \mathbf{W}^{-1} - \mathbf{W}^{-1}(\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}\mathbf{W}^{-1}$, leading to

$$\boxed{m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_{i=1}^{N} \beta_i C_G(\mathbf{u}, \mathbf{x}_i) C_{corr}(\mathbf{u}, \mathbf{x}_i)} \tag{3.39}$$

where $C_G(\mathbf{u}, \mathbf{x}_i)$ is the Gaussian covariance function between $\mathbf{u}$ and $\mathbf{x}_i$ and with

$$C_{corr}(\mathbf{u}, \mathbf{x}_i) = |\mathbf{I} + \mathbf{W}^{-1}\mathbf{\Sigma}_x|^{-1/2} \exp\left[\frac{1}{2}(\mathbf{u} - \mathbf{x}_i)^T \Delta^{-1} (\mathbf{u} - \mathbf{x}_i)\right] \tag{3.40}$$

where $\Delta^{-1} = \mathbf{W}^{-1}(\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}\mathbf{W}^{-1}$, that simplifies into $\Delta^{-1} = \mathbf{W}^{-1} - (\mathbf{W} + \mathbf{\Sigma}_x)^{-1}$, again using the matrix inversion lemma.

Compared to the noise-free $\mu_G(\mathbf{u})$, the covariances between the new noisy input and the training inputs, formerly given by $C_G(\mathbf{u}, \mathbf{x}_i)$, are now weighted by $C_{corr}(\mathbf{u}, \mathbf{x}_i)$, thus accounting for the uncertainty associated with $\mathbf{u}$.

Exact predictive variance

For the variance, replacing l_{ij}^{exG} and l^{exG} by their expression, we find

$$\begin{aligned} v^{exG}(\mathbf{u},\mathbf{\Sigma}_x) &= C_G(\mathbf{u},\mathbf{u}) - c^2\sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j) N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \mathbf{\Sigma}_x + \frac{\mathbf{W}}{2}\right) \\ & -m^{exG}(\mathbf{u},\mathbf{\Sigma}_x)^2 \,, \end{aligned}$$

with

$$\begin{aligned} m^{exG}(\mathbf{u},\mathbf{\Sigma}_x)^2 &= c^2 \sum_{i,j=1}^N \beta_i\beta_j N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W}+\mathbf{\Sigma}_x) N_{\mathbf{u}}(\mathbf{x}_j, \mathbf{W}+\mathbf{\Sigma}_x) \\ &= c^2 \sum_{i,j=1}^N \beta_i\beta_j N_{\mathbf{x}_i}(\mathbf{x}_j, 2(\mathbf{W}+\mathbf{\Sigma}_x)) N_{\mathbf{u}}\left(\frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \frac{\mathbf{W}+\mathbf{\Sigma}_x}{2}\right) \,. \end{aligned}$$

For $\mathbf{\Sigma}_x = \mathbf{0}$, the new predictive variance becomes

$$v^{exG}(\mathbf{u},\mathbf{\Sigma}_x=\mathbf{0}) = C_G(\mathbf{u},\mathbf{u}) - c^2\sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j) N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \frac{\mathbf{W}}{2}\right) - m^{exG}(\mathbf{u},\mathbf{\Sigma}_x=\mathbf{0})^2 \,,$$

with $m^{exG}(\mathbf{u},\mathbf{\Sigma}_x=\mathbf{0})^2 = c^2\sum_{i,j=1}^N \beta_i\beta_j N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \frac{\mathbf{W}}{2}\right)$. Recalling that the predictive variance corresponding to a noise-free $\mathbf{u}$ computed using the Gaussian covariance function is given by

$$\begin{aligned} \sigma_G^2(\mathbf{u}) &= C_G(\mathbf{u},\mathbf{u}) - c^2\sum_{i,j=1}^N K_{ij}^{-1} N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W}) N_{\mathbf{u}}(\mathbf{x}_j, \mathbf{W}) \\ &= C_G(\mathbf{u},\mathbf{u}) - c^2\sum_{i,j=1}^N K_{ij}^{-1} N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\frac{\mathbf{x}_i+\mathbf{x}_j}{2}, \frac{\mathbf{W}}{2}\right) \end{aligned}$$

we can again check that $v^{exG}(\mathbf{u},\mathbf{\Sigma}_x=\mathbf{0}) = \sigma_G^2(\mathbf{u})$.

As was done for the predictive mean, we can find another form for $v^{exG}(\mathbf{u},\mathbf{\Sigma}_x)$ where the Gaussian covariance function appears weighted by a correction term. Using the matrix inversion lemma, we have $\left(\mathbf{\Sigma}_x + \frac{\mathbf{W}}{2}\right)^{-1} = \left(\frac{\mathbf{W}}{2}\right)^{-1} - \left(\frac{\mathbf{W}}{2}\right)^{-1}\left(\left(\frac{\mathbf{W}}{2}\right)^{-1} + \mathbf{\Sigma}_x^{-1}\right)^{-1}\left(\frac{\mathbf{W}}{2}\right)^{-1}$, and, with $\bar{\mathbf{x}} = \frac{\mathbf{x}_i+\mathbf{x}_j}{2}$, we can write

$$l_{ij}^{exG} = c^2 N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\bar{\mathbf{x}}, \mathbf{\Sigma}_x + \frac{\mathbf{W}}{2}\right) = c^2 N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\bar{\mathbf{x}}, \frac{\mathbf{W}}{2}\right) C_{corr_2}(\mathbf{u},\bar{\mathbf{x}}) \,,$$

where

$$C_{corr_2}(\mathbf{u},\bar{\mathbf{x}}) = \left|\left(\frac{\mathbf{W}}{2}\right)^{-1}\mathbf{\Sigma}_x + \mathbf{I}\right|^{-1/2} \exp\left[\frac{1}{2}(\mathbf{u}-\bar{\mathbf{x}})^T\Lambda^{-1}(\mathbf{u}-\bar{\mathbf{x}})\right] \tag{3.41}$$

with $\Lambda^{-1} = \left(\frac{\mathbf{W}}{2}\right)^{-1}\left(\left(\frac{\mathbf{W}}{2}\right)^{-1} + \mathbf{\Sigma}_x^{-1}\right)^{-1}\left(\frac{\mathbf{W}}{2}\right)^{-1} = 2\mathbf{W}^{-1} - \left(\frac{1}{2}\mathbf{W} + \mathbf{\Sigma}_x\right)^{-1}$. We can also show that $c^2 N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}}\left(\bar{\mathbf{x}}, \frac{\mathbf{W}}{2}\right) = C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)$, leading to

$$l_{ij}^{exG} = C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)C_{corr_2}(\mathbf{u},\bar{\mathbf{x}}) \tag{3.42}$$

and therefore

$$\boxed{v^{exG}(\mathbf{u},\mathbf{\Sigma}_x) = C_G(\mathbf{u},\mathbf{u}) - \sum_{i,j=1}^N (K_{ij}^{-1} - \beta_i\beta_j)C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)C_{corr_2}(\mathbf{u},\bar{\mathbf{x}}) - m^{exG}(\mathbf{u},\mathbf{\Sigma}_x)^2} \tag{3.43}$$

In terms of $\sigma_G^2(\mathbf{u})$ plus correction terms, we can rewrite the variance as

$$\begin{aligned} v^{exG}(\mathbf{u},\mathbf{\Sigma}_x) &= \sigma_G^2(\mathbf{u}) + \sum_{i,j=1}^N K_{ij}^{-1}C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)(1 - C_{corr_2}(\mathbf{u},\bar{\mathbf{x}})) \\ &+ \sum_{i,j=1}^N \beta_i\beta_j C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)(C_{corr_2}(\mathbf{u},\bar{\mathbf{x}}) - C_{corr}(\mathbf{u},\mathbf{x}_i)C_{corr}(\mathbf{u},\mathbf{x}_j)) \end{aligned} \tag{3.44}$$

having replaced $m^{exG}(\mathbf{u},\mathbf{\Sigma}_x)^2$ by its expression, using (3.39).

It can be shown that the predictive mean and variance obtained in the Gaussian case tend to the approximate mean and variance when $\mathbf{\Sigma}_x$ tends to zero. As with Figure 3.5 for the approximate moments, Figure 3.6 shows the exact predictive mean and error-bars (triangles) obtained when predicting at noisy inputs (asterisks).

3.5 Qualitative comparisons

Using the simple one-dimensional static example used throughout this chapter, we now compare the predictive distributions given by the Monte-Carlo approximation and by the *Gaussian approximation*, with moments computed both exactly and approximately. We use the following notations:

- MC, denotes the Monte-Carlo approximation to the true predictive distribution corresponding to a noisy input, i.e. $p(y|\mathcal{D},\mathbf{u},\mathbf{\Sigma}_x) \approx \frac{1}{T}\sum_t p(y|\mathcal{D},\mathbf{x}^t)$, where $T = 100$;

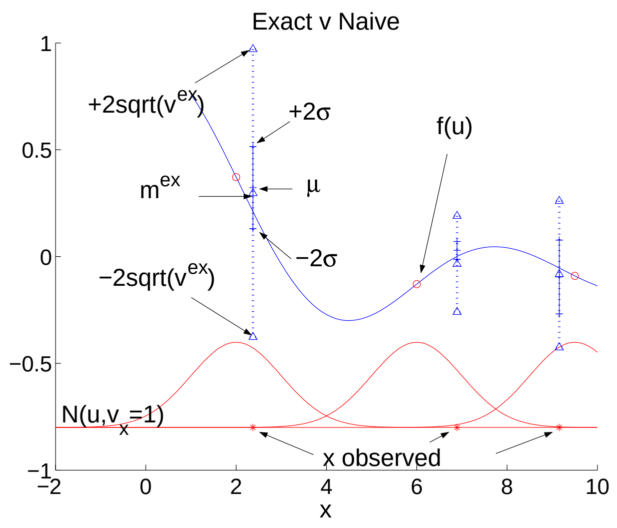

Figure 3.6: As in Figure 3.5, the triangles now indicate the exact predictive means with their error-bars, within the *Gaussian approximation* accounting for the uncertainty on the noisy inputs (asterisks).

- A, denotes the *Gaussian approximation* that computes only the mean and variance of this distribution, and specifically A_{ap} when these moments are computed using the Taylor approximation, and A_{ex} when they are computed exactly (both using the Gaussian covariance function);
- N, denotes the *naive* predictive mean and variances that do not account for the noise on the input.

Figure 3.7 shows the predictive distribution given by MC (continuous), N (dashed), A_{ap} (dots) and A_{ex} (asterisks), when the true noise-free input is 2 (left) and 6 (right) and the input noise variance v_x is 1. We can observe how the naive approach leads to a narrow distribution, peaked around its mean value, since it does not account for the uncertainty on the input. In this example, the distribution defined by the approximate moments is strongly similar to that defined by the exact ones, supporting the idea that the approximation is a valid and useful one. The Monte-Carlo approximation to the true distribution highlights how the true distribution is non-Gaussian. Nevertheless, our *Gaussian approximation* seems appropriate as it spreads about the area where most of the weight of the true distribution seems to lie.

Figure 3.8 shows the histogram of the losses (squared error E_1 on the left and minus log predictive density E_2 on the right) computed for each of the 100 samples given by the Monte-Carlo approximation, when predicting at $x = 2.4$ (left) and $x = 6.9$ (right). For these two predictions, the average

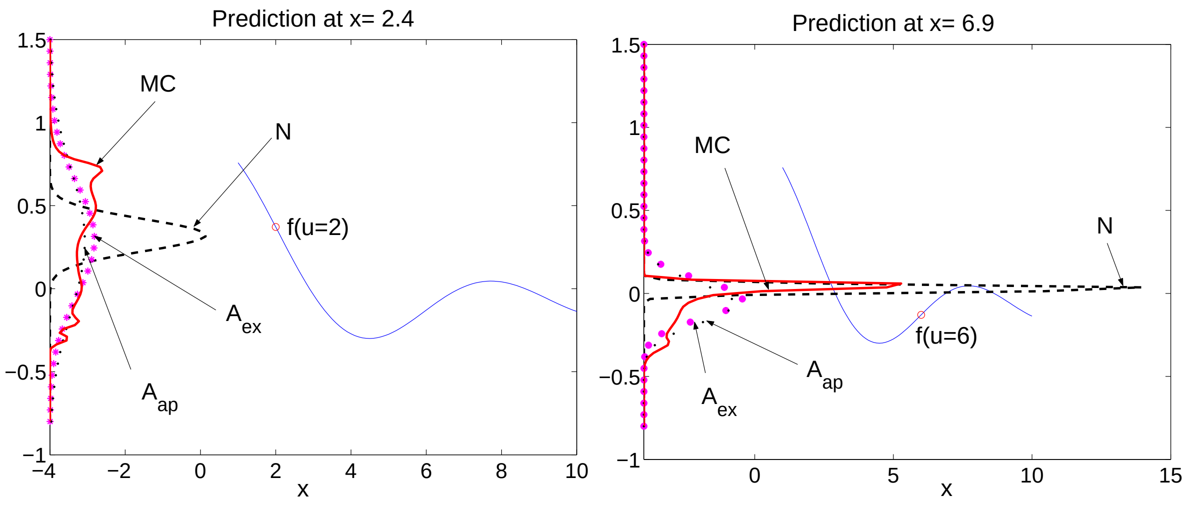

Figure 3.7: Predictive distributions (on the y-axis) obtained when predicting at a noisy input: MC is the numerical approximation by simple Monte-Carlo, A_{ex} and A_{ap} correspond to the *Gaussian approximation* with moments computed exactly and approximately. N is the *naive* predictive distribution that does not account for the noise on the input.

losses are $E_1 = 0.002$ and $E_2 = -1.46$, computed using the sample means and sample variances. With A_{ex}, we obtain $E_1 = 0.06$ and $E_2 = -0.14$, and $E_1 = 0.07$ and $E_2 = 0.09$ with A_{ap}. The naive approach leads to $E_1 = 0.02$ and $E_2 = -1.29$.

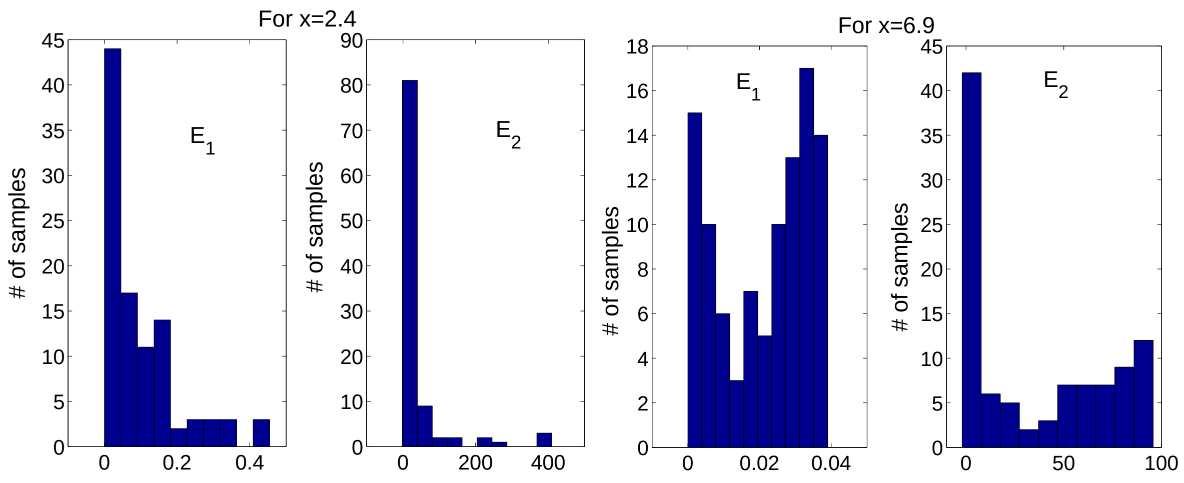

Figure 3.8: Squared error (E_1) and minus log-likelihood (E_2) computed for 100 samples of the Monte-Carlo approximation (for the observed noisy $x = 2.4$, left and $x = 6.9$, right).

3.6 Extension to the learning task

A similar approximation to that taken in this chapter can be used to solve the more challenging problem of training a model in the presence of noisy or uncertain inputs. Again, we make the assumption that the inputs are independent and normally distributed.

3.6.1 Defining a *noisy* process

For $y_i = f(\mathbf{x}_i)$, where $\mathbf{x}_i$ is noise-free, recall that the GP prior on f, with zero-mean and covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$, implies that

$$\begin{aligned} E[y_i|\mathbf{x}_i] &= \int y_i p(y_i) dy_i = 0 && (3.45) \\ \mathrm{Cov}[y_i, y_j|\mathbf{x}_i, \mathbf{x}_j] &= C(\mathbf{x}_i, \mathbf{x}_j) \ . && (3.46) \end{aligned}$$

If we now consider the situation where noisy inputs are sensed by the system, we have $y_i = f(\mathbf{x}_i)$ with $\mathbf{x}_i = \mathbf{u}_i + \boldsymbol{\epsilon}_{\mathbf{x}_i}$, where $\boldsymbol{\epsilon}_{\mathbf{x}_i} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{xi})$. Given $\mathbf{x}_i \sim \mathcal{N}(\mathbf{u}_i, \mathbf{\Sigma}_{xi})$, although the process is not Gaussian anymore, we can still determine the mean and covariance function of what we shall call the *noisy* process (as noted in (Seeger, 2003) page 48).

According to the law of iterated expectations, we can write

$$E[y_i|\mathbf{u}_i] = E_{\mathbf{x}}[E[y_i|\mathbf{x}_i]] = 0 \qquad (3.47)$$

since $E[y_i|\mathbf{x}_i] = 0$. Also, the law of conditional variances tells us that $\mathrm{Var}[y_i|\mathbf{u}_i] = E_{\mathbf{x}}[\mathrm{Var}[y_i|\mathbf{x}_i]] + \mathrm{Var}_{\mathbf{x}}[E_{\mathbf{x}}[y_i|\mathbf{x}_i]] = E_{\mathbf{x}}[\mathrm{Var}[y_i|\mathbf{x}_i]]$. Extending this result to the covariances leads to

$$\mathrm{Cov}[y_i, y_j|\mathbf{u}_i, \mathbf{u}_j] = \int\int C(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i, \mathbf{x}_j) d\mathbf{x}_i d\mathbf{x}_j \ , \qquad (3.48)$$

where we allow the noise to vary for each input, i.e. $p(\mathbf{x}_i) = \mathcal{N}_{\mathbf{x}_i}(\mathbf{u}_i, \mathbf{\Sigma}_{xi})$ and $p(\mathbf{x}_j) = \mathcal{N}_{\mathbf{x}_j}(\mathbf{u}_j, \mathbf{\Sigma}_{xj})$. Let $C_n(\mathbf{u}_i, \mathbf{u}_j)$ denote this 'noisy' covariance function giving the covariance between y_i and y_j. Assuming the inputs are independent given their characteristics, we can define

$$C_n(\mathbf{u}_i, \mathbf{u}_j) = \int\int C(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i) p(\mathbf{x}_j) d\mathbf{x}_i d\mathbf{x}_j \ , \qquad (3.49)$$

and, as before, how solvable the integral is depends on the form of the covariance function.

Approximation via Taylor expansion

We first consider the case where the covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$ is such that integral (3.49) is analytically intractable. Assuming we can perform a Taylor expansion of C around $(\mathbf{u}_i, \mathbf{u}_j)$, we can approximate the function by the following second order polynomial

$$\begin{aligned} C^{app}(\mathbf{x}_i, \mathbf{x}_j) &= C(\mathbf{u}_i, \mathbf{u}_j) + (\mathbf{x}_i - \mathbf{u}_i)^T \mathbf{C}'_{\mathbf{u}_i}(\mathbf{u}_i, \mathbf{u}_j) + (\mathbf{x}_j - \mathbf{u}_j)^T \mathbf{C}'_{\mathbf{u}_j}(\mathbf{u}_i, \mathbf{u}_j) \\ &+ \frac{1}{2}(\mathbf{x}_i - \mathbf{u}_i)^T \mathbf{C}''_{\mathbf{u}_i,\mathbf{u}_i}(\mathbf{u}_i, \mathbf{u}_j)(\mathbf{x}_i - \mathbf{u}_i) + \frac{1}{2}(\mathbf{x}_j - \mathbf{u}_j)^T \mathbf{C}''_{\mathbf{u}_j,\mathbf{u}_j}(\mathbf{u}_i, \mathbf{u}_j)(\mathbf{x}_j - \mathbf{u}_j) \\ &+ (\mathbf{x}_i - \mathbf{u}_i)^T \mathbf{C}''_{\mathbf{u}_i,\mathbf{u}_j}(\mathbf{u}_i, \mathbf{u}_j)(\mathbf{x}_j - \mathbf{u}_j) \ , \end{aligned}$$

where $\mathbf{C}'_{\mathbf{u}_i}(\mathbf{u}_i, \mathbf{u}_j) = \frac{\partial C(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i}$, $\mathbf{C}''_{\mathbf{u}_i,\mathbf{u}_i}(\mathbf{u}_i, \mathbf{u}_j) = \frac{\partial^2 C(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial x_i{}^T}$ and $\mathbf{C}''_{\mathbf{u}_i,\mathbf{u}_j}(\mathbf{u}_i, \mathbf{u}_j) = \frac{\partial^2 C(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i \partial \mathbf{x}_j}$, all evaluated at $\mathbf{x}_i = \mathbf{u}_i, \mathbf{x}_j = \mathbf{u}_j$. Replacing this approximation in (3.49), we have

$$C_n^{app}(\mathbf{u}_i, \mathbf{u}_j) = \int \int C^{app}(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i) p(\mathbf{x}_j) d\mathbf{x}_i d\mathbf{x}_j \ .$$

Integrating the integral with respect to $\mathbf{x}_i$ first, and then with respect to $\mathbf{x}_j$, we obtain

$$C_n^{app}(\mathbf{u}_i, \mathbf{u}_j) = C(\mathbf{u}_i, \mathbf{u}_j) + \frac{1}{2}\left(\mathrm{Tr}[\mathbf{C}''_{\mathbf{u}_i,\mathbf{u}_i}(\mathbf{u}_i, \mathbf{u}_j)\mathbf{\Sigma}_{xi}] + \mathrm{Tr}[\mathbf{C}''_{\mathbf{u}_j,\mathbf{u}_j}(\mathbf{u}_i, \mathbf{u}_j)\mathbf{\Sigma}_{xj}]\right) \ . \tag{3.50}$$

If we further assume the inputs are corrupted by the same type of noise (that is, with the same variance, $\mathbf{\Sigma}_{xi} = \mathbf{\Sigma}_{xj} = \mathbf{\Sigma}_x$), this expression simplifies into

$$C_n^{app}(\mathbf{u}_i, \mathbf{u}_j) = C(\mathbf{u}_i, \mathbf{u}_j) + \frac{1}{2}\mathrm{Tr}[(\mathbf{C}''_{\mathbf{u}_i,\mathbf{u}_i}(\mathbf{u}_i, \mathbf{u}_j) + \mathbf{C}''_{\mathbf{u}_j,\mathbf{u}_j}(\mathbf{u}_i, \mathbf{u}_j))\mathbf{\Sigma}_x] \ . \tag{3.51}$$

Note that, for a given covariance function $C(\mathbf{u}_i, \mathbf{u}_j)$, one should verify that $C_n^{app}(\mathbf{u}_i, \mathbf{u}_j)$ is a valid covariance function, i.e. leading to a positive semi-definite covariance matrix.[8]

Exact Gaussian case

When the covariance function is Gaussian, we can evaluate $C_n(\mathbf{u}_i, \mathbf{u}_j)$ exactly. As before, using the notation $cN_{\mathbf{x}_i}(\mathbf{x}_j, \mathbf{W})$ for the Gaussian covariance function, where $c = (2\pi)^{D/2}|\mathbf{W}|^{1/2}v$ and

[8]In the case of the Gaussian kernel, this implies a condition on the input noise variance. In one-dimension, we have $C(u_i, u_j) = v \exp\left[-\frac{1}{2}w(x_i^2 - x_j^2)\right]$, and $C''_{u_i,u_i}(u_i, u_j) = C''_{u_j,u_j}(u_i, u_j) = [-w + w^2(u_i - u_j)^2]C(u_i, u_j)$, so that

$$C_n^{app}(u_i, u_j) = C(u_i, u_j) + \frac{1}{2}v_x[C''_{u_i,u_i}(u_i, u_j) + C''_{u_j,u_j}(u_i, u_j)] = C(u_i, u_j) + v_x C''_{u_i,u_i}(u_i, u_j) \ ,$$

which should be positive. In particular, we have $C_n^{app}(u_i, u_i) = C(u_i, u_i) + v_x C''_{u_i,u_i}(u_i, u_i) = v(1 - v_x w)$, since $C(u_i, u_i) = v$ and $C''_{u_i,u_i}(u_i, u_i) = -wC(u_i, u_i)$. Therefore, for $C_n^{app}(u_i, u_i)$ to be positive, we need $1 - v_x w > 0$, that is $v_x < 1/w$.

$\mathbf{W}^{-1} = \mathrm{diag}[w_1 \dots w_D]$, we need to evaluate

$$C_n^{exG}(\mathbf{u}_i, \mathbf{u}_j) = c \int \int N_{\mathbf{x}_i}(\mathbf{x}_j, \mathbf{W}) p(\mathbf{x}_i) p(\mathbf{x}_j) d\mathbf{x}_i d\mathbf{x}_j \ . \tag{3.52}$$

Using the product of Gaussians and integrating over $\mathbf{x}_i$ leads to

$$\int N_{\mathbf{x}_i}(\mathbf{x}_j, \mathbf{W}) p(\mathbf{x}_i) d\mathbf{x}_i = N_{\mathbf{x}_j}(\mathbf{u}_i, \mathbf{W} + \mathbf{\Sigma}_{xi})$$

and, integrating this result with respect to $\mathbf{x}_j$, we have

$$\int N_{\mathbf{x}_j}(\mathbf{u}_i, \mathbf{W} + \mathbf{\Sigma}_{xi}) p(\mathbf{x}_j) d\mathbf{x}_j = N_{\mathbf{u}_i}(\mathbf{u}_j, \mathbf{W} + \mathbf{\Sigma}_{xi} + \mathbf{\Sigma}_{xj}) \ .$$

The *noisy* covariance function can then be written $C_n^{exG}(\mathbf{u}_i, \mathbf{u}_j) = cN_{\mathbf{u}_i}(\mathbf{u}_j, \mathbf{W} + \mathbf{\Sigma}_{xi} + \mathbf{\Sigma}_{xj})$, that is

$$C_n^{exG}(\mathbf{u}_i, \mathbf{u}_j) = v|\mathbf{I} + \mathbf{W}^{-1}(\mathbf{\Sigma}_{xi} + \mathbf{\Sigma}_{xj})|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{u}_i - \mathbf{u}_j)^T (\mathbf{W} + \mathbf{\Sigma}_{xi} + \mathbf{\Sigma}_{xj})^{-1} (\mathbf{u}_i - \mathbf{u}_j)\right] \ . \tag{3.53}$$

Assuming that $\mathbf{\Sigma}_{xi} = \mathbf{\Sigma}_{xj} = \mathbf{\Sigma}_x$, we obtain

$$C_n^{exG}(\mathbf{u}_i, \mathbf{u}_j) = v|\mathbf{I} + 2\mathbf{W}^{-1}\mathbf{\Sigma}_x|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{u}_i - \mathbf{u}_j)^T (\mathbf{W} + 2\mathbf{\Sigma}_x)^{-1} (\mathbf{u}_i - \mathbf{u}_j)\right] \ . \tag{3.54}$$

For a diagonal $\mathbf{\Sigma}_x$, we have $\mathbf{W} + \mathbf{\Sigma}_x = \mathrm{diag}\left[\frac{1}{w_1} + v_{x1}, \dots, \frac{1}{w_D} + v_{xD}\right]$, so that each w parameter is weighted by the corresponding uncertainty in the given input dimension. We can then write

$$C_n^{exG}(\mathbf{u}_i, \mathbf{u}_j) = v|\mathbf{I} + 2\mathbf{W}^{-1}\mathbf{\Sigma}_x|^{-1/2} \exp\left[-\frac{1}{2}\sum_{d=1}^{D} \frac{w_d}{1 + 2w_d v_{xd}} (u_i^d - u_j^d)^2\right] \ . \tag{3.55}$$

Note that, unless one has prior knowledge of the input noise variance, and use (3.55) for fixed values of $v_{x1} \dots v_{xD}$, it might be preferable to learn a single parameter v' in place of $v|\mathbf{I} + 2\mathbf{W}^{-1}\mathbf{\Sigma}_x|^{-1/2}$.

3.6.2 Inference and prediction

Given the covariance function

$$C_n^{exG}(\mathbf{u}_i, \mathbf{u}_j) = v' \exp\left[-\frac{1}{2}\sum_{d=1}^{D} \frac{w_d}{1 + 2w_d v_{xd}} (u_i^d - u_j^d)^2\right] \ , \tag{3.56}$$

with parameters $\mathbf{\Theta} = \{w_1, \dots, w_D, v', v_{x1}, \dots, v_{xD}, v_t\}$, the learning and prediction tasks are no more difficult than in the 'noise-free' case presented in Section 2.3.1 of the previous chapter. Simply now, the minimisation of the likelihood $\mathcal{L}(\mathbf{\Theta})$ is with respect to D more parameters, namely $v_{x1}, \dots, v_{xD}$.

Similarly, the prediction at a new (noise-free) input $\mathbf{x}$ is simply obtained by conditioning on the training data and $\mathbf{x}$. The predictive distribution of the corresponding output is Gaussian with mean and variance

$$\begin{aligned} \mu(\mathbf{x}) &= \sum_{i=1}^{N} \beta_i C_n^{ex_G}(\mathbf{x}, \mathbf{u}_i) \\ \sigma^2(\mathbf{x}) &= C_n^{ex_G}(\mathbf{x}, \mathbf{x}) - \sum_{i,j=1}^{N} K_{ij}^{-1} C_n^{ex_G}(\mathbf{x}, \mathbf{u}_i) C_n^{ex_G}(\mathbf{x}, \mathbf{u}_j) \ , \end{aligned} \tag{3.57}$$

where $\boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{t}$, $K_{ij} = C_n^{ex_G}(\mathbf{x}_i, \mathbf{x}_j) + v_t \delta_{ij}$ and $\mathbf{t}$ is the vector of observed targets. The vector of covariances between the noise-free input and the noisy training inputs, $C_n^{ex_G}(\mathbf{x}, \mathbf{u}_i)$, is found directly by considering (3.53) and letting $\boldsymbol{\Sigma}_{xj}$ go to zero. Since we consider the case where the noise variance is the same for all inputs, we simply have

$$C_n^{ex_G}(\mathbf{x}, \mathbf{u}_i) = v' \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{u}_i)^T (\mathbf{W} + \boldsymbol{\Sigma}_x)^{-1} (\mathbf{x} - \mathbf{u}_i)\right] \ , \tag{3.58}$$

and similarly for $C_n^{ex_G}(\mathbf{x}, \mathbf{u}_j)$. Also, the variance between the new input and itself is given by $C_n^{ex_G}(\mathbf{x}, \mathbf{x}) = v'$.

It can be noted that, with the new kernel $C_n^{ex_G}(\mathbf{u}_i, \mathbf{u}_j)$, the length-scales are bounded below, meaning that the corresponding process does not allow for functions with very high resolution, an effect that could certainly be obtained by setting a suitable prior on $\mathbf{W}$. A last remark is that the noise on the inputs needs not be white, as the extension to coloured noise, as suggested in (Murray-Smith and Girard, 2001), should be straightforward.

3.7 Summary

Central to this chapter is the prediction at a noisy input. We have presented an analytical approximation allowing the GP model to compute the mean and variance of the predictive distribution of the output corresponding to an input corrupted by some noise with zero mean and variance $\boldsymbol{\Sigma}_x$.

Recall that in the noise-free case, the predictive distribution $p(y|\mathcal{D}, \mathbf{x})$ of the output y corresponding to a new $\mathbf{x}$ is Gaussian, with mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$. When the new input is noisy, such that $\mathbf{x} \sim \mathcal{N}(\mathbf{u}, \mathbf{\Sigma}_x)$, one has to integrate $p(y|\mathcal{D}, \mathbf{x})$ over the input distribution, leading to an intractable integral. In this case, the analytical approximation we propose consists of computing only the mean and variance of the corresponding new predictive distribution (*Gaussian approximation*). These moments are respectively given by

$$\begin{aligned} m(\mathbf{u}, \mathbf{\Sigma}_x) &= \sum_{i=1}^{N} \beta_i E_{\mathbf{x}}[C(\mathbf{x}, \mathbf{x}_i)] \\ v(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[C(\mathbf{x}, \mathbf{x})] - \sum_{i,j=1}^{N} (K_{ij}^{-1} - \beta_i \beta_j) E_{\mathbf{x}}[C(\mathbf{x}, \mathbf{x}_i) C(\mathbf{x}, \mathbf{x}_j)] - m(\mathbf{u}, \mathbf{\Sigma}_x)^2 \, , \end{aligned}$$

and can be computed exactly or approximately, depending on the form of the covariance function of the process.

We have shown how, for a Gaussian and a linear covariance functions, these *noisy* mean and variance could be computed exactly (given by (3.26) and (3.29) in the linear case, and by (3.39) and (3.43) in the case of the Gaussian kernel). For general covariance functions, we have suggested using a Taylor approximation, leading to approximate moments (given by (3.16) and (3.17)). A simple numerical comparison of our *Gaussian approximation* to the numerical approximation of the intractable integral by simple Monte-Carlo (equation (3.3)) has shown the validity of our approach.

In Section 3.6, we have introduced a similar approximation, to deal with the more difficult task of learning a model that accounts for the noise on the inputs. In the case of the Gaussian covariance function, we have shown that a model accounting for this extra noise (as opposed to the output noise) had a Gaussian kernel with its length-scales bounded below, proportionally to the input noise variance.

We now proceed to the modelling of nonlinear dynamic systems, and the application of the prediction at a noisy input to the iterative multi-step-ahead prediction task, with propagation of the uncertainty.

Chapter 4

Modelling nonlinear dynamic systems

One of the main objectives in time series analysis is forecasting. Whereas good one-step-ahead models can be relatively easily obtained, multiple-step-ahead ones constitute a far more challenging problem. We now focus on the modelling of nonlinear dynamic systems and propose to apply the methodology presented in the previous chapters to the iterative multiple-step ahead prediction of time-series with propagation of the uncertainty. Assuming a GP was trained to minimise one-step-ahead predictions, we show how we can formally incorporate the uncertainty about intermediate regressor values induced by each successive prediction as we predict ahead in time, thus updating the uncertainty on the current prediction. We illustrate the approach on the simulated Mackey-Glass chaotic time-series and compare the propagation of uncertainty algorithm within the Gaussian approximation and the Monte-Carlo alternative.

4.1 Introduction

When it comes to dynamic systems modelling, system identification is that branch dealing with the general process of extracting information about a system from measured input and output data. Once a model is identified, it can be used for simulation, prediction, controller design or analysis. We refer to (Ljung, 1999; Söderström and Stoica, 1989) as general textbooks on system identification. Typically, we can distinguish between 'fundamental models', derived from first principles, or empirical models. The reason why empirical models might be preferred is that no detailed understanding of the process is required to develop the model. Within the empirical model class, we find state-space (e.g. Kalman filters (Welch and Bishop, 1995; Julier and Uhlmann, 2000)) or input-output models. Although distinct theories have been developed for these two representations, it is always possible to convert an identified input-output model into a state-space model (see e.g. (Phan et al., 1998)). In the following, we consider the input-output class, represented as a Nonlinear Auto-Regressive (NAR) model. In discrete time, we have

$$y_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-L}) + \epsilon_{t+1}, \tag{4.1}$$

where y_{t+1} is the system output at time $t+1$ and $\mathbf{x}_{t+1} = [y_t, y_{t-1}, \dots, y_{t-L}]^T$ is the corresponding *state*. Here, the additive noise term ϵ_{t+1} reflects the fact that the next output will not be an exact function of past data.

This representation is motivated by viewing the observed one-dimensional time-series $y_1, \dots, y_t$ as a projection of the underlying dynamics, which lie in a higher dimensional space (Takens, 1981). The order of the NAR model, L, which corresponds to the number of delayed outputs (sometimes referred to as *lag* or *embedding dimension*), gives the dimension of the reconstructed space (Casdagli, 1989; Farmer and Sidorowich, 1988). It is important to note that this representation implicitly assumes that the statistical properties of the data are time independent, so that the task is finally reduced to the learning of a static mapping (as pointed out in (Bishop, 1995), pages $302-303$ for feed-forward neural networks). Roughly speaking, nonlinear system identification then involves model and structure selection (choice of a model for the mapping $f(.)$, selection of L), noise modelling, parameter estimation and model validation. Most commonly, neural networks have been used for the nonlinear mapping $f(.)$ (Principe et al., 1992a; Kuo and Principe, 1994; Principe and Kuo, 1995; Bakker et al., 1998). Support vector regressors have also been developed for prediction purposes (Mukherjee et al., 1997; Müller et al., 1997) and, recently, an extension of linear dynamical systems (see (Roweis and Ghahramani, 1997) for a review) using kernels has been suggested (Ralaivola and d'Alché Buc, 2003). The use of Gaussian Processes is still in its infancy for the modelling of dynamic systems (Murray-Smith et al., 1999; Murray-Smith and Girard, 2001; Kocijan et al., 2003b; Kocijan et al., 2003a).

We are particularly interested in multi-step-ahead time-series prediction, which is a much more challenging problem than the one-step-ahead prediction task. The problem is as follows: Given a time-series known, say, up to time t, we wish to predict (or get the predictive distribution of) the output at time $t + k$, where k is the predictive horizon. This corresponds to a missing or noisy data modelling problem,[1] where y_{t+k-1} down to y_t are missing. Currently, predicting ahead in time can be achieved directly or iteratively. With the direct method, a model is explicitly trained to learn to predict k steps ahead, e.g. assuming $y_{t+k} = f(y_t, y_{t-1}, \dots, y_{t-L})$; the model being therefore tailored for a fixed horizon k (which might actually be difficult to fix in advance). If k is large and the system very nonlinear, the drawback of the direct method is that it will in general require a large amount of data to get a good model, because of the larger amount of missing data between targets and inputs.

In the following, we focus on the iterative approach, that consists of the iteration of a one-step-ahead model such as (4.1), up to the desired horizon. In this case, there are different ways of dealing with the missing data. Numerical solutions consist of integrating over the unknown (or missing) variables, weighted by their conditional probability density (which is done, albeit in a classification context, in (Tresp and Hofmann, 1995; Tresp and Hofmann, 1998)). A naive way of iterating a one-step-ahead model is simply to substitute a single value for the missing value. This approach has been shown not to be optimal (Ahmad and Tresp, 1993; Tresp and Hofmann, 1995), as we will illustrate in our numerical examples. In (Tresp and Hofmann, 1998), stochastic sampling is shown to be superior to both simply iterating the system and using the extended Kalman filter.

Two obvious reasons for favouring the iterative method over the direct one are that accurate one-step-ahead models are usually easy to get and, also, any k-step-ahead forecast is available, up to the prediction horizon. However, the well known drawback of the iterative approach is the accumulation of errors as we predict ahead in time, as each subsequent iterated prediction uses no more information than was used in the first one-step prediction. In (Small and Judd, 1999; Judd and Small, 2000), they aim at improving long-term predictions by eliminating the systematic errors induced by each successive short term prediction. Here, we suggest not to eliminate the error but on the contrary to

[1]The missing variables can be seen as noisy variables for complete noise.

propagate it through the model as we predict ahead in time.[2]

4.2 Iterative multi-step ahead forecasting

We are now going to show how we can propagate in the model the uncertainty induced by each successive prediction. That is, at each time-step, we not only feed back the predictive mean (i.e. a single point estimate) but also the predictive variance, that represents the associated uncertainty.

We consider the NAR model (4.1), that is

$$y_t = f(\mathbf{x}_t) + \epsilon_t, \tag{4.2}$$

where $\mathbf{x}_t = [y_{t-1}, \dots, y_{t-1-L}]$ and the functional $f(.)$ is modelled by a zero-mean GP with covariance function $C(\mathbf{x}_i, \mathbf{x}_j)$. Although richer noise models, such as ARMA models could be considered (Murray-Smith and Girard, 2001), we simply assume that ϵ_t is a white noise.

With this one-step ahead model, in order to get $y_{t+k} = f(y_{t+k-1}, \dots, y_{t+k-L})$, we need $y_{t+k-1} = f(y_{t+k-2}, \dots, y_{t+k-2-L})$, down to $y_{t+1} = f(y_t, \dots, y_{t-L})$. Since the time-series is assumed to be known up to time t, the predictive distribution of y_{t+1} is readily obtained. We have $y_{t+1} \sim \mathcal{N}(\mu(\mathbf{x}_{t+1}), \sigma^2(\mathbf{x}_{t+1}))$, as given by equations (3.1). For y_{t+2}, a naive approach is to use only the point estimate of y_{t+1}, and consider $y_{t+2} = f(\hat{y}_{t+1}, y_t, \dots, y_{t+2-L})$, where $\hat{y}_{t+1} = \mu(\mathbf{x}_{t+1})$. Here, we suggest to feed-back the whole predictive distribution of y_{t+1} to account for the model's uncertainty $\sigma^2(\mathbf{x}_{t+1})$ on the estimate $\mu(\mathbf{x}_{t+1})$. We then have $y_{t+2} = f(y_{t+1}, y_t, \dots, y_{t+2-L})$, where the state $\mathbf{x}_{t+2}$ is now a random vector with mean $[\mu(\mathbf{x}_{t+1}), y_t, \dots, y_{t+2-L}]$ and zero covariance matrix, apart from the first entry which is $\sigma^2(\mathbf{x}_{t+1})$. This takes us back to Chapter 3 and the task of making a prediction given a random input.

[2]Our approach can be linked to the extended Kalman filter that summarises past data by an estimate of the mean and covariance of the variables.

4.2.1 Propagation of uncertainty algorithm within the *Gaussian approximation*

Within the approximation presented in Chapter 3, we can compute the mean and variance of y_{t+2}. Interpreting this approximation as Gaussian, we can write

$$y_{t+2} \sim \mathcal{N}(m(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}), v(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})) \ ,$$

where $\mathbf{u}_{t+2}$ and $\mathbf{\Sigma}_{t+2}$ are the mean and covariance matrix of $\mathbf{x}_{t+2}$. As we saw then, the predictive mean and variance can be computed exactly or approximately, depending on the form of the covariance function. Then, for the next time-step, we can feed both $m(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$ and $v(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$ back into $\mathbf{x}_{t+3}$, and repeat the process up to the desired horizon.

Here is a sketch of how we proceed:

- $t+1$, $\mathbf{x}_{t+1} = [y_t, \dots, y_{t-L}]^T$. Compute $y_{t+1} \sim \mathcal{N}(\mu(\mathbf{x}_{t+1}), \sigma^2(\mathbf{x}_{t+1}))$.
- $t+2$, $\mathbf{x}_{t+2} = [y_{t+1}, y_t, \dots, y_{t+1-L}]^T \sim \mathcal{N}(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$ with

$$\mathbf{u}_{t+2} = \begin{bmatrix} \mu(\mathbf{x}_{t+1}) \\ y_t \\ \vdots \\ y_{t+1-L} \end{bmatrix} \quad \text{and} \quad \mathbf{\Sigma}_{t+2} = \begin{bmatrix} \sigma^2(\mathbf{x}_{t+1}) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} .$$

Compute $y_{t+2} \sim \mathcal{N}(m(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}), v(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}))$.

- $t+3$, $\mathbf{x}_{t+3} = [y_{t+2}, y_{t+1}, \dots, y_{t+2-L}]^T$. We now have

$$\mathbf{x}_{t+3} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) \\ \mu(\mathbf{x}_{t+1}) \\ y_t \\ \vdots \\ y_{t+2-L} \end{bmatrix}, \begin{bmatrix} v(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) & \mathrm{Cov}[y_{t+2}, y_{t+1}] & 0 & \dots & 0 \\ \mathrm{Cov}[y_{t+1}, y_{t+2}] & \sigma^2(\mathbf{x}_{t+1}) & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 \end{bmatrix} \right) .$$

Compute $y_{t+3} \sim \mathcal{N}(m(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3}), v(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3}))$.

Repeating this procedure up to k, we finally have $y_{t+k} \sim \mathcal{N}(m(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k}), v(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k}))$. At time $t+k$, with $k > L$, the random input is $\mathbf{x}_{t+k} = [y_{t+k-1}, y_{t+k-2}, \dots, y_{t+k-L}]^T$, with mean $\mathbf{u}_{t+k}$, formed of the L delayed previously predicted means,

$$\mathbf{u}_{t+k} = \begin{bmatrix} m(\mathbf{u}_{t+k-1}, \mathbf{\Sigma}_{t+k-1}) \\ m(\mathbf{u}_{t+k-2}, \mathbf{\Sigma}_{t+k-2}) \\ \dots \\ m(\mathbf{u}_{t+k-L}, \mathbf{\Sigma}_{t+k-L}) \end{bmatrix}$$

and $L \times L$ covariance matrix $\mathbf{\Sigma}_{t+k}$ with the delayed predictive variances on its diagonal:

$$\mathbf{\Sigma}_{t+k} = \begin{bmatrix} v(\mathbf{u}_{t+k-1}, \mathbf{\Sigma}_{t+k-1}) & \mathrm{Cov}[y_{t+k-1}, y_{t+k-2}] & \dots & \mathrm{Cov}[y_{t+k-1}, y_{t+k-L}] \\ \mathrm{Cov}[y_{t+k-2}, y_{t+k-1}] & v(\mathbf{u}_{t+k-2}, \mathbf{\Sigma}_{t+k-2}) & \dots & \mathrm{Cov}[y_{t+k-2}, y_{t+k-L}] \\ \dots & \dots & \dots & \dots \\ \mathrm{Cov}[y_{t+k-L}, y_{t+k-1}] & \mathrm{Cov}[y_{t+k-L}, y_{t+k-2}] & \dots & v(\mathbf{u}_{t+k-L}, \mathbf{\Sigma}_{t+k-L}) \end{bmatrix} .$$

We now need to compute the cross-covariance terms of the input covariance matrix. They correspond to the covariances between the delayed outputs, that is $\mathrm{Cov}[y_{t+k-i}, y_{t+k-j}]$, for $i = 1 \dots L-1$ and $j = i+1 \dots L$. Since we fill in the input covariance matrix as we progress ahead in time, this is equivalent to computing the cross-covariances between the output and the input the time-step before, that is $\mathrm{Cov}[y_{t+k-1}, \mathbf{x}_{t+k-1}]$, discarding the last (*oldest*) element of $\mathbf{x}_{t+k-1}$. For simplicity in the notation, let us write $l = k-1$. We need to compute

$$\mathrm{Cov}[y_{t+l}, \mathbf{x}_{t+l}] = E[y_{t+l}\mathbf{x}_{t+l}] - E[y_{t+l}]E[\mathbf{x}_{t+l}] \;, \tag{4.3}$$

where $E[\mathbf{x}_{t+l}] = \mathbf{u}_{t+l}$ and $E[y_{t+l}] = m(\mathbf{u}_{t+l}, \mathbf{\Sigma}_{t+l})$. For the expectation of the product, we have

$$\begin{aligned} E[y_{t+l}\mathbf{x}_{t+l}] &= \int\int y_{t+l}\mathbf{x}_{t+l} p(y_{t+l}, \mathbf{x}_{t+l}) dy_{t+l} d\mathbf{x}_{t+l} \\ &= \int\int y_{t+l}\mathbf{x}_{t+l} p(y_{t+l}|\mathbf{x}_{t+l}) p(\mathbf{x}_{t+l}) dy_{t+l} d\mathbf{x}_{t+l} \\ &= \int \mathbf{x}_{t+l}\mu(\mathbf{x}_{t+l}) p(\mathbf{x}_{t+l}) d\mathbf{x}_{t+l} \;. \end{aligned}$$

Replacing $\mu(\mathbf{x}_{t+l})$ by its expression, $\mu(\mathbf{x}_{t+l}) = \sum_i \beta_i C(\mathbf{x}_{t+l}, \mathbf{x}_i)$, we have

$$E[y_{t+l}\mathbf{x}_{t+l}] = \sum_i \beta_i \int \mathbf{x}_{t+l} C(\mathbf{x}_{t+l}, \mathbf{x}_i) p(\mathbf{x}_{t+l}) d\mathbf{x}_{t+l} \;. \tag{4.4}$$

As in Chapter 3, we can evaluate this integral exactly or approximately, depending on the form of $C(.,.)$. Denoting $\mathbf{x}_{t+l}$ by $\mathbf{x}$ for notational convenience, let $I_i = \int \mathbf{x} C(\mathbf{x}, \mathbf{x}_i) p(\mathbf{x}) d\mathbf{x}$ be the integral we wish to evaluate.

Case of the Gaussian covariance function

We first derive the expressions of the cross-covariances exactly, in the case of the Gaussian covariance function. As in Section 3.4.2 of the previous chapter, we write the Gaussian covariance function $C_G(\mathbf{x}, \mathbf{x}_i)$ as $cN_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W})$, with $c = (2\pi)^{D/2}|\mathbf{W}|^{1/2}v$. So we wish to solve

$$I_i^{exG} = c \int \mathbf{x} N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W}) p(\mathbf{x}) d\mathbf{x} \tag{4.5}$$

where $p(\mathbf{x}) = \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x)$.

Using the product of Gaussians, we have $N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W})\mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x) = N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\mathcal{N}_{\mathbf{x}}(\mathbf{d}_i, \mathbf{D})$ with $\mathbf{D} = (\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}$ and $\mathbf{d}_i = \mathbf{D}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x^{-1}\mathbf{u})$. Substituting in I_i^{ex} gives

$$I_i^{exG} = cN_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x) \int \mathbf{x} \mathcal{N}_{\mathbf{x}}(\mathbf{d}_i, \mathbf{D}) d\mathbf{x} = cN_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\mathbf{d}_i \ ,$$

that is

$$I_i^{exG} = cN_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)(\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x^{-1}\mathbf{u}) \ . \tag{4.6}$$

Using the matrix inversion lemma, we can write

$$\begin{aligned} (\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}\mathbf{W}^{-1} &= [\mathbf{I} - \mathbf{W}(\mathbf{W} + \mathbf{\Sigma}_x)^{-1}] \\ (\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}\mathbf{\Sigma}_x^{-1} &= [\mathbf{I} - \mathbf{\Sigma}_x(\mathbf{W} + \mathbf{\Sigma}_x)^{-1}] \ , \end{aligned}$$

leading to

$$I_i^{exG} = cN_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)([\mathbf{I} - \mathbf{W}(\mathbf{W} + \mathbf{\Sigma}_x)^{-1}]\mathbf{x}_i + [\mathbf{I} - \mathbf{\Sigma}_x(\mathbf{W} + \mathbf{\Sigma}_x)^{-1}]\mathbf{u}) \ . \tag{4.7}$$

In this short notation, the cross-covariance terms are then given by

$$\mathrm{Cov}[y, \mathbf{x}] = \sum_i \beta_i I_i^{exG} - m^{exG}(\mathbf{u}, \mathbf{\Sigma}_x)\mathbf{u}$$

where $m^{exG}(\mathbf{u},\mathbf{\Sigma})$ is the predictive mean obtained in the Gaussian case. Recalling that we can write it as $\sum_i \beta_i c N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W}+\mathbf{\Sigma}_x)$, we have

$$\mathrm{Cov}[y,\mathbf{x}] = \sum_i \beta_i c N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W}+\mathbf{\Sigma}_x)([\mathbf{I}-\mathbf{W}(\mathbf{W}+\mathbf{\Sigma}_x)^{-1}]\mathbf{x}_i - \mathbf{\Sigma}_x(\mathbf{W}+\mathbf{\Sigma}_x)^{-1}\mathbf{u})$$

and we can check that, as should be the case, $\mathrm{Cov}[y,\mathbf{x}]=0$ if $\mathbf{\Sigma}_x$ is zero. Using $cN_{\mathbf{u}}(\mathbf{x}_i,\mathbf{W}+\mathbf{\Sigma}_x) = C(\mathbf{u},\mathbf{x}_i)C_{corr}(\mathbf{u},\mathbf{x}_i)$, with C_{corr} given by (3.40), we can finally write

$$\mathrm{Cov}[y_{t+l},\mathbf{x}_{t+l}] = \sum_i \beta_i C(\mathbf{u}_{t+l},\mathbf{x}_i)C_{corr}(\mathbf{u}_{t+l},\mathbf{x}_i)([\mathbf{I}-\mathbf{W}(\mathbf{W}+\mathbf{\Sigma}_{t+l})^{-1}]\mathbf{x}_i - \mathbf{\Sigma}_{t+l}(\mathbf{W}+\mathbf{\Sigma}_{t+l})^{-1}\mathbf{u}_{t+l})\ . \tag{4.8}$$

General case

If we cannot evaluate I_i exactly, as in the previous chapter, we use a Taylor approximation of the covariance function.

In the one-dimensional case, we need to compute $I_i = \int xC(x,x_i)p(x)dx$ with $p(x)=\mathcal{N}_x(u,v_x)$. Within a second-order approximation of the covariance function around u, we can write

$$\begin{aligned} I_i^{ap} &\approx \int x\left(C(u,x_i)+(x-u)C'(u,x_i)+\frac{1}{2}(x-u)^2C''(u,x_i)\right)p(x)dx \\ &\approx uC(u,x_i)+v_xC'(u,x_i)+\frac{1}{2}uv_xC''(u,x_i)\ , \end{aligned}$$

where we have used $\int x^2p(x)dx = v_x+u^2$ and $\int x^3p(x)dx = 3uv_x+u^3$.

Extending this result to L-dimensional inputs, we have

$$I_i^{ap} \approx \mathbf{u}^T C(\mathbf{u},\mathbf{x}_i) + \mathbf{C}'(\mathbf{u},\mathbf{x}_i)^T\mathbf{\Sigma}_x + \frac{1}{2}\mathbf{u}^T\mathrm{Tr}[\mathbf{C}''(\mathbf{u},\mathbf{x}_i)\mathbf{\Sigma}_x]\ , \tag{4.9}$$

and the cross-covariance terms are given by

$$\mathrm{Cov}[y,\mathbf{x}] = \sum_i \beta_i I_i^{ap} - m^{ap}(\mathbf{u},\mathbf{\Sigma}_x)\mathbf{u}\ ,$$

where $m^{ap}(\mathbf{u},\mathbf{\Sigma}) = \sum_{i=1}^N \beta_i\left[C(\mathbf{u},\mathbf{x}_i)+\frac{1}{2}\mathrm{Tr}[\mathbf{C}''(\mathbf{u},\mathbf{x}_i)\mathbf{\Sigma}_x]\right]$, thus simplifying $\mathrm{Cov}[y,\mathbf{x}]$ into

$$\mathrm{Cov}[y,\mathbf{x}] = \sum_i \beta_i \mathbf{C}'(\mathbf{u},\mathbf{x}_i)^T\mathbf{\Sigma}_x\ .$$

Therefore, the cross-covariance terms of the input covariance matrix at time $t+k$ are given by

$$\mathrm{Cov}[y_{t+l}, \mathbf{x}_{t+l}] = \sum_i \beta_i \mathbf{C}'(\mathbf{u}_{t+l}, \mathbf{x}_i)^T \mathbf{\Sigma}_{t+l} \ , \tag{4.10}$$

where $l = k - 1$. Note that we would have obtained the same result if we had simply considered a first-order Taylor approximation of the covariance function.

We can then compute the full input covariance matrix at each time-step. The advantage of this approach is that we account not only for the uncertainty induced by each successive prediction but also for the cross-covariances between the delayed predicted variables, enabling us access to the full joint distribution of the L delayed outputs.

We now turn to the numerical solution of the propagation of uncertainty when using the iterative method.

4.2.2 Monte-Carlo alternative

As already seen, if the time-series is assumed to be known up to time t, at $t+1$, we have $\mathbf{x}_{t+1} = [y_t, y_{t-1}, \dots, y_{t-L}]$ so that we can simply compute $y_{t+1} \sim \mathcal{N}(\mu(\mathbf{x}_{t+1}), \sigma^2(\mathbf{x}_{t+1}))$, using equations (3.1). For the next time-step, propagating the uncertainty induced by y_{t+1} implies the evaluation of

$$p(y_{t+2}|\mathcal{D}, \mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) = \int p(y_{t+2}|\mathcal{D}, \mathbf{x}_{t+2}) p(\mathbf{x}_{t+2}) d\mathbf{x}_{t+2} \ ,$$

where $\mathcal{D}$ is the set of training data and $\mathbf{x}_{t+2} = [y_{t+1}, y_t, \dots, y_{t+1-L}]$, so that $p(\mathbf{x}_{t+2}) = \mathcal{N}_{\mathbf{x}_{t+2}}(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$, with $\mathbf{u}_{t+2} = [\mu(\mathbf{x}_{t+1}), y_t, \dots, y_{t+1-L}]$ and $\Sigma^{11}_{t+2} = \sigma^2(\mathbf{x}_{t+1})$ and zero elsewhere.

Instead of computing only the mean and variance of $p(y_{t+2}|\mathcal{D}, \mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$, as done in the *Gaussian approximation*, we can approximate the integral numerically, by simple Monte-Carlo:

$$p(y_{t+2}|\mathcal{D}, \mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y_{t+2}|\mathcal{D}, \mathbf{x}^s_{t+2}) \ ,$$

where S is the number of samples and $\mathbf{x}^s_{t+2}$ is a sample of $p(\mathbf{x}_{t+2})$.[3] Now, for given $\mathbf{x}^s_{t+2}$, we know

[3]Note that at this point, sampling from $p(\mathbf{x}_{t+2})$ is equivalent to sampling y^s_{t+1} from $p(y_{t+1})$ and letting $\mathbf{x}^s_{t+2} = [y^s_{t+1}, y_t, \dots, y_{t+1-L}]$.

that the corresponding output is normally distributed. We can then write

$$p(y_{t+2}|\mathcal{D}, \mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) = \frac{1}{S}\sum_{s=1}^{S} \mathcal{N}_{y_{t+2}}(\mu(\mathbf{x}^s_{t+2}), \sigma^2(\mathbf{x}^s_{t+2}))\ ,$$

such that the distribution of y_{t+2} can be seen as a mixture of S Gaussians with same mixing weight $\frac{1}{S}$.

At $t+3$, things start complicating as we now have $\mathbf{x}_{t+3} = [y_{t+2}, y_{t+1}, \dots, y_{t+2-L}]$, where y_{t+1} is normal but y_{t+2} is a mixture of S Gaussians...

Instead of working out the whole distribution of y_{t+2}, a first approximation consists of considering one Gaussian from the mixture at a time. We can then write

- $t+2$: Compute $p(y_{t+2}|\mathcal{D}, \mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) = \frac{1}{S}\sum_{s=1}^{S} \mathcal{N}_{y_{t+2}}(\mu(\mathbf{x}^s_{t+2}), \sigma^2(\mathbf{x}^s_{t+2}))$
- $t+3$: Loop over s
  - $\mathbf{x}_{t+3} = [y_{t+2}, y_{t+1}, \dots, y_{t+2-L}]$, where we consider $y_{t+2} \sim \mathcal{N}(\mu(\mathbf{x}^s_{t+2}), \sigma^2(\mathbf{x}^s_{t+2}))$. We then have $p(\mathbf{x}_{t+3}) = \mathcal{N}_{\mathbf{x}_{t+3}}(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3})$ with

$$\mathbf{u}_{t+3} = \begin{bmatrix} \mu(\mathbf{x}^s_{t+2}) \\ \mu(\mathbf{x}_{t+1}) \\ y_t \\ \vdots \\ y_{t+2-L} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \sigma^2(\mathbf{x}^s_{t+2}) & \mathrm{Cov}[y_{t+2}, y_{t+1}] & 0 & \dots & 0 \\ \mathrm{Cov}[y_{t+1}, y_{t+2}] & \sigma^2(\mathbf{x}_{t+1}) & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 \end{bmatrix}$$

  - Evaluate

$$p(y_{t+3}|\mathcal{D}, \mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y_{t+3}|\mathcal{D}, \mathbf{x}^s_{t+3}) = \frac{1}{S}\sum_{s=1}^{S} \mathcal{N}_{y_{t+3}}(\mu(\mathbf{x}^s_{t+3}), \sigma^2(\mathbf{x}^s_{t+3}))$$

    where $\mathbf{x}^s_{t+3}$ is a sample from $p(\mathbf{x}_{t+3})$.
- End Loop

Each single Gaussian s therefore leads to a mixture of Gaussians for $p(y_{t+3}|\mathcal{D}, \mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3})$, implying S mixtures of Gaussians for time-step $t+3$ (behaving like a branching process), which is not computationally tractable.

This leads us to the following second approximation, where we consider one sample at a time, up to the desired horizon:

```
• Loop over $s$
    – $t+2$: $\mathbf{x}_{t+2} = [y_{t+1}, y_t, \dots, y_{t+1-L}] \sim \mathcal{N}(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$
      Sample $\mathbf{x}^s_{t+2}$ from $p(\mathbf{x}_{t+2})$
      Compute $p(y_{t+2}|\mathcal{D}, \mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) = \mathcal{N}_{y_{t+2}}(\mu(\mathbf{x}^s_{t+2}), \sigma^2(\mathbf{x}^s_{t+2}))$
    – $t+3$: $\mathbf{x}_{t+3} = [y_{t+2}, y_{t+1}, \dots, y_{t+2-L}] \sim \mathcal{N}(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3})$
      Sample $\mathbf{x}^s_{t+3}$ from $p(\mathbf{x}_{t+3})$
      Compute $p(y_{t+3}|\mathcal{D}, \mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3}) = \mathcal{N}_{y_{t+3}}(\mu(\mathbf{x}^s_{t+3}), \sigma^2(\mathbf{x}^s_{t+3}))$
                                   ⋮
    – Up to $t+k$.
• End Loop
```

This way, we effectively obtain S Gaussians for the output distribution at each time-step, thus approximating the 'true' mixture of Gaussians distributions.

In our experiments, we consider the following simplified version of the above algorithm:

```
• Loop over $s$
    – $t+1$, $\mathbf{x}_{t+1} = [y_t, y_{t-1}, \dots, y_{t-L}]$. Compute $y_{t+1} \sim \mathcal{N}(\mu(\mathbf{x}_{t+1}), \sigma^2(\mathbf{x}_{t+1}))$
    – For $\kappa = 2 \dots k$
        ∗ Draw a sample $y^s_{t+\kappa-1}$ from $p(y_{t+\kappa-1}|\mathcal{D}, \mathbf{x}_{t+\kappa-1})$
        ∗ Shift the time-window and form $\mathbf{x}_{t+\kappa} = [y^s_{t+\kappa-1}, \dots, y^s_{t+\kappa-L}]$
        ∗ Compute $y_{t+\kappa} \sim \mathcal{N}(\mu(\mathbf{x}_{t+\kappa}), \sigma^2(\mathbf{x}_{t+\kappa}))$
    – End For
```

- End Loop

The notable difference we would expect between this algorithm and the previous one would be in terms of 'smoothness' of the state sample at each time-step: In this last algorithm, the state is composed of delayed sampled outputs, each one of them sampled from its corresponding one dimensional Gaussian distribution (thus treating each of them independently). On the other hand, in the previous algorithm, the state sample comes from an L-dimensional distribution, with a full covariance matrix, thereby introducing some 'smoothness'. Nevertheless, sampling from an L-dimensional distribution will become cumbersome for large L. Although we have not investigated this point any further, we speculate that our approximation should not have a great impact on the overall numerical approximation to the true distribution.

4.3 k-step-ahead prediction of the Mackey-Glass chaotic time-series

This first numerical example is intended to illustrate the propagation of uncertainty algorithm. On the 100-step ahead prediction of the Mackey-Glass time-series, we are going to

- Compare the predictions given by the *Gaussian approximation* and in particular
  - When propagating the uncertainty: Assess the quality of the predictions computed approximately (A_{ap}), by comparing them to those computed exactly (A_{ex}), in the case of the Gaussian covariance function;[4]
  - Compare these predictions to those given by the *naive* approach (N), that does not account for the uncertainty induced by the successive predictions as it feeds back only the predictive means when predicting ahead in time.
- Compare the exact *Gaussian approximation* A_{ex} to the numerical approximation of the propagation of uncertainty algorithm by Monte-Carlo (MC).

Recall that, when predicting at $\mathbf{x} \sim \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x)$, the naive approach simply computes $\mu(\mathbf{u})$ and $\sigma^2(\mathbf{u})$ (given by equations (3.1)), whereas within the *Gaussian approximation*, the predictive mean and variance are given by (3.39) and (3.44), in the case of the Gaussian covariance function, and by (3.16) and

[4]To do so, the approximate moments are computed using a second-order Taylor approximation of the Gaussian covariance function.

(3.17) within the Taylor approximation.

The Mackey-Glass chaotic system constitutes a well-known challenging benchmark for the multiple-step-ahead prediction task, due to its strong non-linearity (Mackey and Glass, 1977). We consider $\frac{dy(t)}{dt} = -by(t) + a\frac{y(t-\tau)}{1+y(t-\tau)^{10}}$, with $a = 0.2$, $b = 0.1$ and $\tau = 17$. This continuous-time model is discretised and the series is re-sampled with period 1 and normalised. We then assume the following one-step-ahead NAR model $y_{t+1} = f(y_t, y_{t-1}, \dots, y_{t-L})$, where $L = 16$, and form a vector of N outputs, where $N = 500$ (taken at random from the series), that we corrupt by a white noise with variance 0.001, and the corresponding matrix of $N \times L$ inputs. We train a zero-mean Gaussian Process with Gaussian covariance function

$$C(\mathbf{x}_i, \mathbf{x}_j) = v \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right] \,, \tag{4.11}$$

where $\mathbf{W}^{-1} = \mathrm{diag}[w_1 \dots w_D]$ and with v set to 1. On the one-step-ahead prediction of the training data, we obtain an average squared error E_1 of 8.4870×10^{-4} and an average negative log-predictive density[5] E_2 of -2.1145. For the validation (test) set, we have $E_1 = 6.4734 \times 10^{-4}$ and $E_2 = -2.5297$.

We now proceed to make 100-step-ahead predictions of the test series and compare the different methods.

4.3.1 *Gaussian approximation*

We first compare the predictions obtained with and without propagation of the uncertainty, that is A_{ex} and A_{ap}, to the naive approach N.

Figures 4.1, 4.2, 4.3 show predictions from 1 to 100 steps ahead, for different starting points in the test series. The left plots show the mean predictions, with triangles and circles indicating those obtained when propagating the uncertainty and computing the means exactly (A_{ex}) and approximately (A_{ap}) respectively. The dots correspond to the mean predictions given by the naive approach N. For clarity, the corresponding error-bars ($\pm 2\times$ standard deviation) are shown on the right plots.

[5]See Chapter 2, Section 2.3.3 where these measures of predictive performance are introduced.

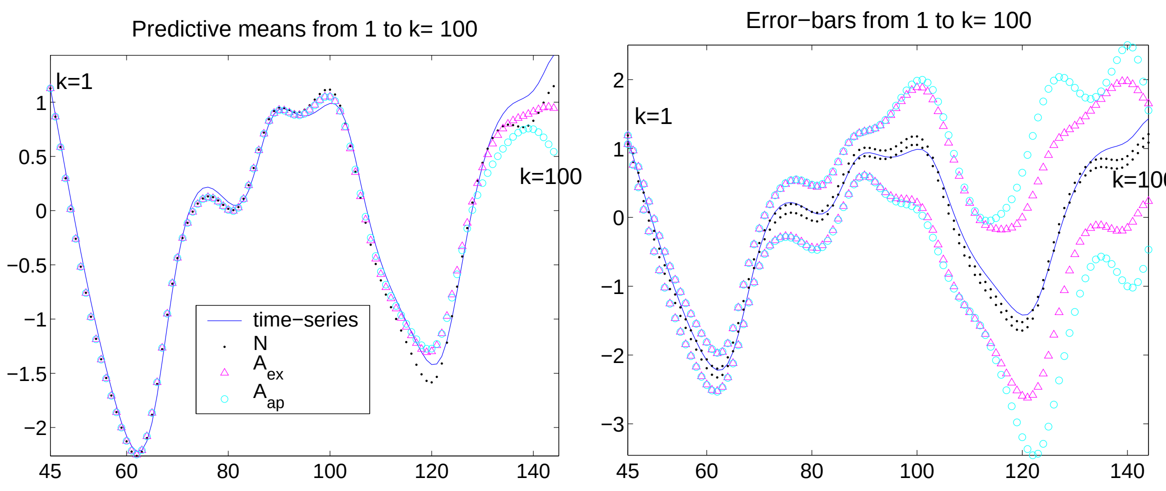

Figure 4.1: Iterative method in action, from $T = 45$: Predictions from 1 to 100 steps ahead of the Mackey-Glass time-series (continuous line). Left: Predictive means given by A_{ex} (triangles) and A_{ap} (circles), corresponding to propagation of the uncertainty, and those given by the naive approach N (dots). Right: Confidence intervals (error-bars of $\pm 2\times$ standard deviation) given by the different methods.

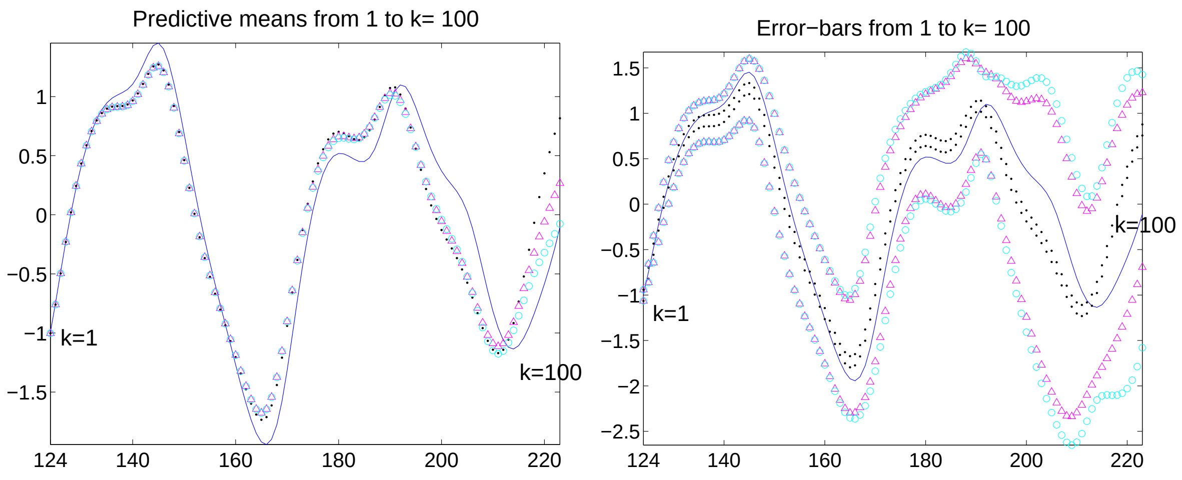

Figure 4.2: Predictions from 1 to 100 steps ahead starting from $T = 124$. Same legend as for Figure 4.1. Note how the predictive mean given by A_{ap} at $k = 100$ is the closest to the true value.

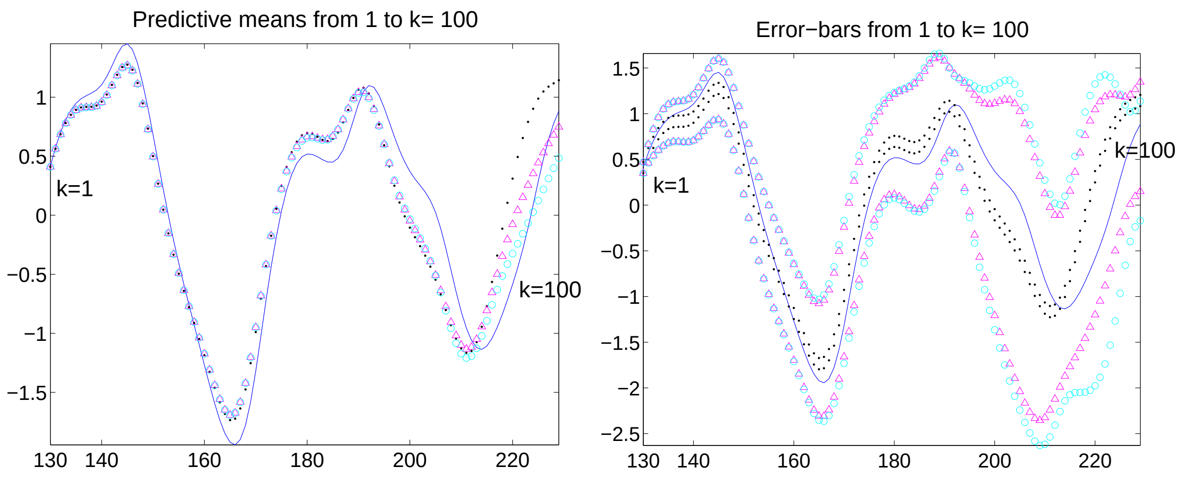

Figure 4.3: Predictions from 1 to 100 steps ahead starting from $T = 130$. Same legend as for Figure 4.1. Now, at $k = 100$, it is A_{ex} which leads to the best prediction.

Common to these three figures is the fact that, even though the naive method might lead, in some cases, to better mean predictions than the methods propagating the uncertainty (as is the case in Figure 4.1), the associated uncertainties are always very tight and not representative of how good the estimates are. In particular, at $k = 100$ in Figure 4.1, the mean prediction given by N is closer to the true value than either A_{ap} or A_{ex}. Nevertheless, the associated error-bars are very small and do not include the true value. Although the mean predictions given by the naive approach may significantly differ from the true time-series, this model is too confident about its estimates.

In terms of mean predictions, the three methods give similar estimates, up to around 70 steps ahead, and start diverging after that. Figure 4.2 illustrates the case when A_{ap} leads to a better prediction than A_{ex} at $k = 100$. Compared to the naive approach, the error-bars obtained when propagating the uncertainty are more informative. Although difficult to interpret, it is interesting to note how the error-bars given by both A_{ex} and A_{ap} vary, as they do not simply increase with k (we could expect the model to become more and more uncertain, as predictions at large k are based on the same amount of information as those at small k). Although, in general, the model's uncertainty starts increasing quite rapidly, it does not prevent the model from rectifying itself, leading to smaller error-bars as it has more confidence into its estimates, at least for this particular simulation.

Figure 4.4 shows the 50-step (left) and 100-step (right) ahead predictive means with their confidence intervals (note that in these plots each point **is** a k-step ahead prediction). The upper plots correspond to the predictive means given by the naive approach, with their 2σ error-bars (which are so tight that one cannot distinguish them from the means). The middle and bottom plots correspond to A_{ap} and A_{ex} respectively, where the shaded area represents the uncertainty region.

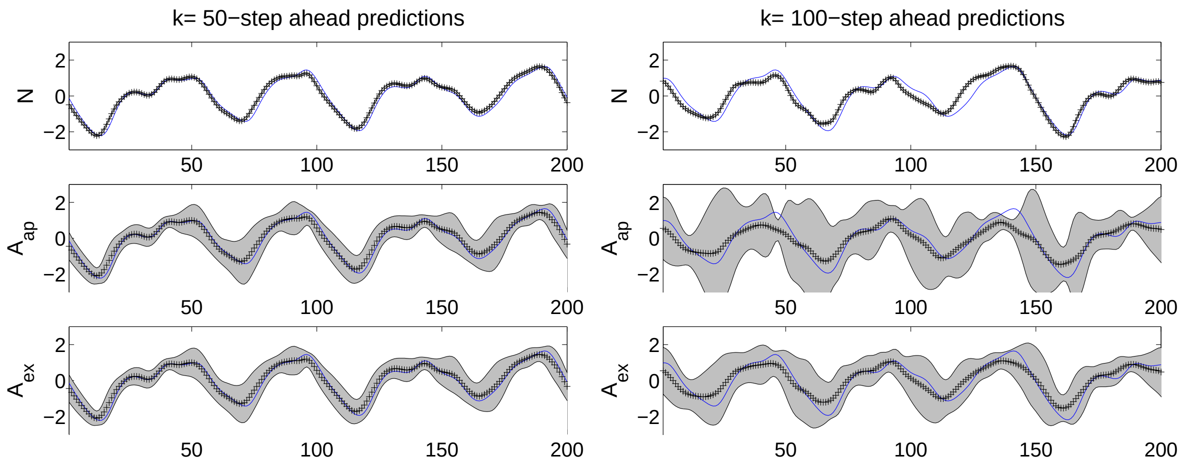

Figure 4.4: 50-step (left) and 100-step (right) ahead predictions of the test time-series (continuous thin line). From top to bottom, moments given by the naive approach and by the propagation of uncertainty approach with moments computed approximately and exactly. The crosses indicate the predictive means and the shaded regions correspond to the confidence/uncertainty intervals. (Note that each point is a k-step ahead prediction).)

Again, the inadequacy of the naive approach is evident, with error-bars that are not anymore representative of the model's uncertainty. Given that the approximate moments were computed using the Gaussian covariance function, these plots provide direct qualitative comparison between A_{ap} and A_{ex}. For the 50-step-ahead predictions, the predictions look very similar. For $k = 100$, we can see that A_{ap} is generally more 'cautious', with larger error-bars than A_{ex}. What we are interested in is having variances that actually reflect the quality of the estimates (i.e. inaccurate mean predictions should have large error-bars). This requirement being fulfilled for almost all k-step ahead predictions (apart from around 50 and 150 of the 100-step ahead predictions), we feel confident the approximate moments lead to 'sensible' results.

For each 100^{th} point predicted, Figure 4.5 shows the corresponding squared error E_1 (left) and minus log-predictive density E_2 (right). (For clarity, we omit E_2 given by the naive approach as it is very large compared to that obtained for A_{ex} and A_{ap}.)

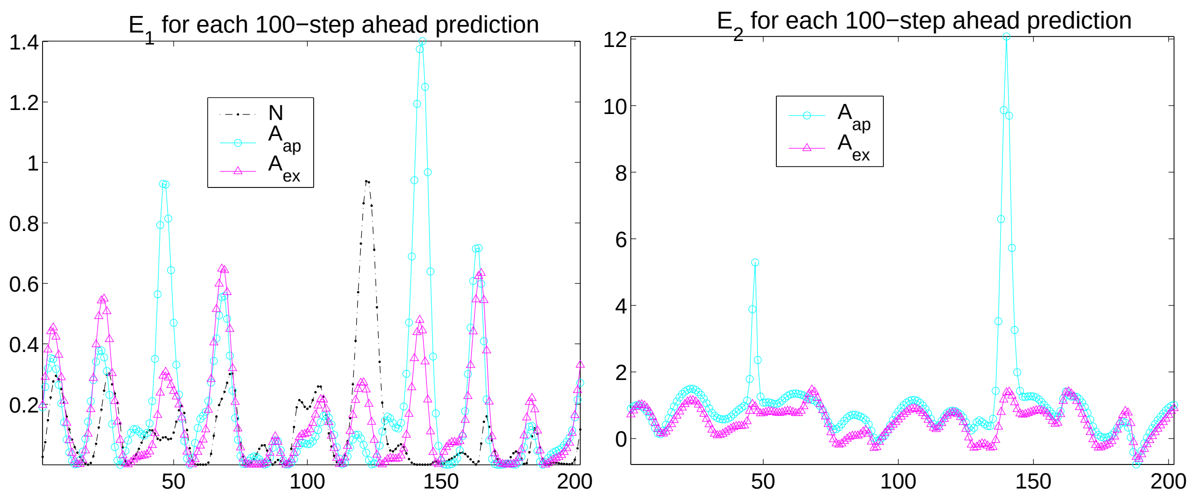

Figure 4.5: Left: Squared error for each 100^{th} predicted point, for all three approaches. Right: Minus log-predictive density, for A_{ex} and A_{ap}.

This figure provides us with quantitative evidence for the above conclusions. In terms of E_1, which measures the quality of predictions by only comparing the estimate (predictive mean) to the true value, there are points for which the naive approach is going to be better. However, in terms of E_2, which accounts for the model uncertainty, the results are far worse for the naive method than when propagating the uncertainty. Also, this plot demonstrates that A_{ap} is effectively comparable to A_{ex}.

4.3.2 Exact *Gaussian approximation* v Monte-Carlo

We now turn to comparing A_{ex}, based on the exact moments obtained within the *Gaussian approximation*, to MC, the Monte-Carlo approximation to the propagation of uncertainty algorithm.

For the Monte-Carlo approximation, we compute $S = 500$ samples at each time step, resulting in a $N_t \times S \times k$ matrix of predictive means (and similarly for the variances), where N_t is the number of starting points (202). From $t + 1$ to $t + 100$, Figure 4.6 shows the predictive means given by MC, their average (used to compute the losses, thick line), and the means given by A_{ex} (crosses). The right plot shows the corresponding uncertainties (error-bars).

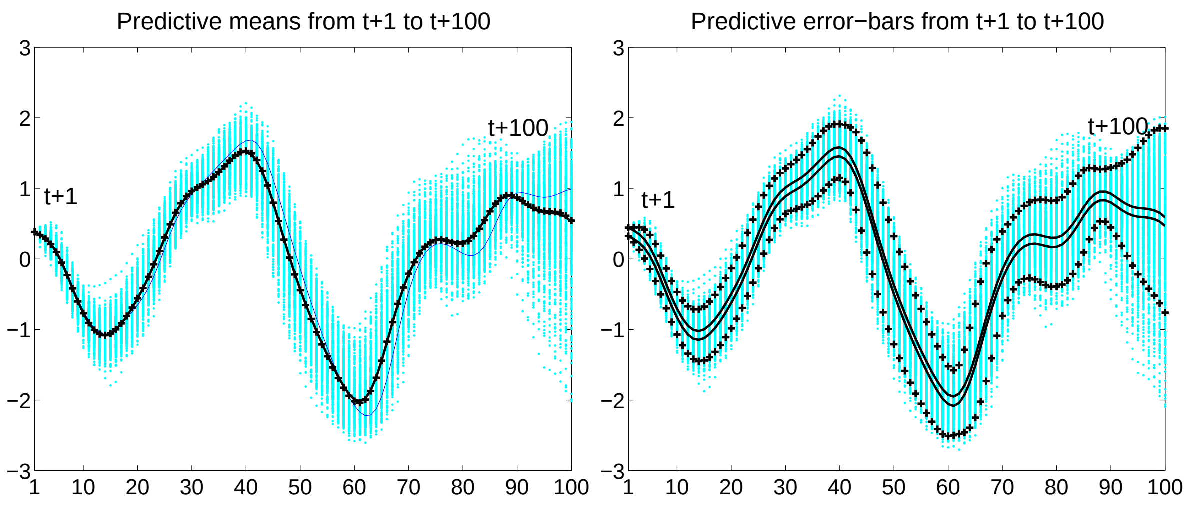

Figure 4.6: From $t+1$ to $t+100$: Predictive means (left) and error-bars (right) given by MC (500 samples and average, thick line) and by A_{ex} (crosses).

Until about 60 steps ahead, the error-bars of A_{ex} are similar to those of the Monte-Carlo approximation (as they include most of the samples). The plot of the predictive distributions at $t+10$, $t+60$ and $t+100$ on Figure 4.7 shows that up to 60 steps ahead, the Gaussian approximation is close to the true distribution. But as the number of steps increases, the true distribution approximated by MC departs more and more from the Gaussian distribution assumed by A_{ex}.

Finally, Figure 4.8 shows the evolution of the average losses with increasing k for all methods. The losses for MC correspond to those computed using the average sample mean and variance. These results for the Monte-Carlo approximation might look surprising, but one should keep in mind that estimating the quality of this approximation with these losses is not representative (since the distribution is not normal).

4.4 Summary

Assuming a one-step-ahead NAR model of the form

$$y_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-L}) + \epsilon_{t+1},$$

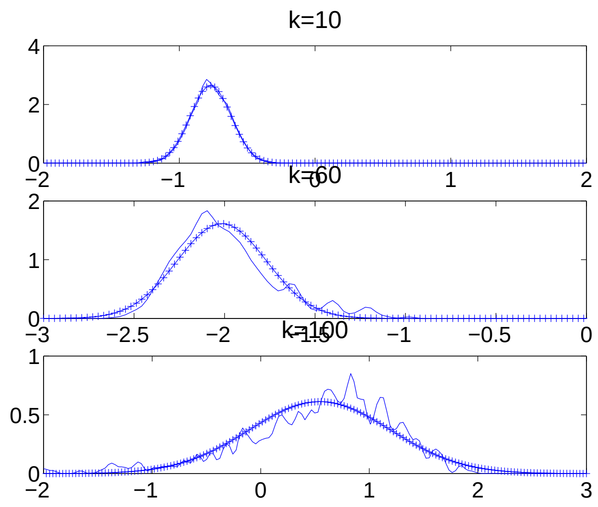

Figure 4.7: Predictive distribution given by MC (continuous line) and as approximated by the *Gaussian approximation* (crosses), at $k = 10, 60, 100$.

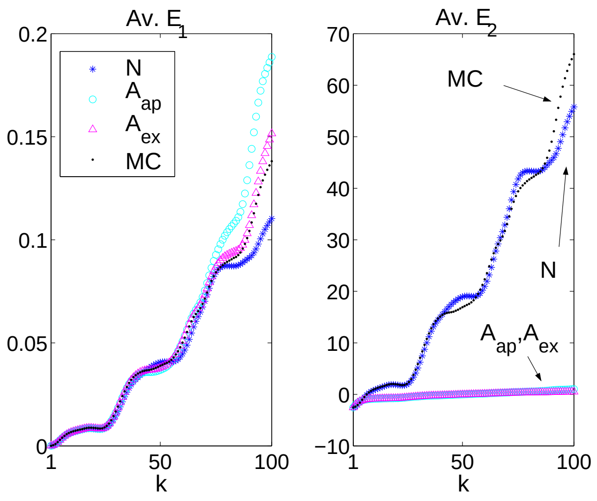

Figure 4.8: Evolution of the average squared error (left) and minus log-predictive density (on a log-scale, right) as the number of steps ahead increases from 1 to 100.)

where $f(.)$ is modelled by a GP with zero-mean and covariance function $C(.,.)$, we have shown how one could derive the (approximate) predictive distribution of y_{t+k}, where the uncertainty induced by each successive prediction (y_{t+1} to y_{t+k-1}) is being accounted for. At time $t + k$, using the results of the previous chapter, we can write $y_{t+k} \sim \mathcal{N}(m(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k}), v(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k}))$, where $\mathbf{u}_{t+k}$ and $\mathbf{\Sigma}_{t+k}$ are the vector of expectations and covariance matrix of the corresponding random input $\mathbf{x}_{t+k} = [y_{t+k-1}, y_{t+k-2}, \ldots, y_{t+k-L}]^T$. In this setting, the mean vector $\mathbf{u}_{t+k}$ is formed of the L delayed previously predicted means (output estimates) and $\mathbf{\Sigma}_{t+k}$ is the $L \times L$ covariance matrix with the delayed predictive variances on its diagonal (associated uncertainties) and the cross-covariances between the delayed outputs (see Section 4.2.1).

On the Mackey-Glass chaotic system, we compare the iterative 100-step-ahead prediction of the time-series obtained using the propagation of uncertainty algorithm within the *Gaussian approximation* (as discussed above) and within the Monte-Carlo numerical approximation (see Section 4.2.2). This experiment shows that, as the number of steps increases, the true distribution (approximated by Monte-Carlo) departs from the Gaussian assumption. Nonetheless, the *Gaussian approximation* proves to be valid up to around 60 steps ahead.

When compared to a naive approach that predicts ahead in time by only considering the output estimate at each time-step (treating it as if it were the true observed value), our approximate propagation of uncertainty algorithm shows indeed to improve the predictive performance of the model. The naive method is consistently over-confident about its estimates, with associated uncertainties which are no longer representative (very tight error-bars not encompassing the true values). On the other hand, a model propagating the uncertainty as it proceeds ahead in time leads to informative error-bars on its estimates. Importantly, such a model does not only see its predictive variance increasing with the predictive horizon.

Chapter 5

Real-life applications

We now illustrate the modelling and forecasting of two real-life processes with the Gaussian Process model and the methodology presented in Chapter 4, focusing on the iterative k-step ahead predictions within the Gaussian approximation. For the gas-liquid separator process, the identification of the NARX structure is performed using a subset-selection approach, based on the Automatic Relevance Determination tool and Occam's razor principle. A test pressure signal is then simulated by the model, with and without propagation of the uncertainty. The second application is a pH neutralisation process. After fitting a linear model, a GP is used to model the residuals and with this 'mixed model', we perform the iterative k-step-ahead prediction of a test pH signal, with and without propagation of the uncertainty. The modified propagation of uncertainty algorithm, accounting for the linear part, is presented. Also, the iterative approach is compared to the direct multi-step-ahead prediction method where the model is tailored for predicting k steps ahead.

5.1 Modelling the gas-liquid separator process

The gas-liquid separator process is part of a plant (Figure 5.1) situated at the Jozef Stefan Institute (Slovenia), which serves as a test bed for various control engineering problems. In our case, the gas is simply oxygen and the liquid is water. A pump feeds the gas-water mixture into a separator where the gas is separated from the water. The internal pressure in the separator then increases and a pressure valve is opened to blow out the gas from the separator to the next unit.

Figure 5.1: The gas-liquid separator part of the process plant at the Jozef Stefan Institute (Slovenia). A pump feeds the gas-water mixture into a reservoir where the gas is separated from the water.

Figure 5.2 shows a schematic diagram of the identification structure. The separator has two valves, one at the top controlling the pressure $\mathbf{p}$, and one at the bottom, for controlling the water level $\mathbf{h}$ inside the separator. The openness of the pressure and level valves are indicated by $\mathbf{u_p}$ and $\mathbf{u_h}$ respectively. $\mathbf{r_h}$ corresponds to the desired set-point for the water level. Although the separator is a multivariate process, with two inputs and two outputs, it is controlled as two distinct univariate processes, for simplicity. The pressure is controlled by a predictive controller and the water level by a PI controller. Since the pressure dynamics depend on the water level, the identification of these dynamics is achieved by controlling the pressure valve and varying the water level.

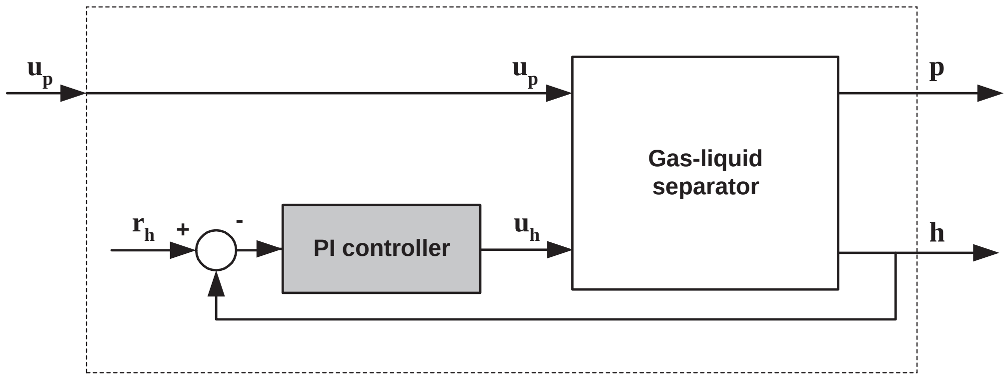

Figure 5.2: Schematic representation of the identification of the pressure dynamics (provided by Bojan Likar).

In this application, we can distinguish between the quantisation noise, due to the transformation from continuous to digital signals, and the noise on the measurement devices (sensor noise). The valve openness signal (a pseudo-random sequence with discrete levels) is noise-free, but the noise on the water level signal is coloured (due to the controller), as therefore is that on the gas pressure measurement. Typically, the overall noise is broadband, with continuous spectrum, but, in the range of the process dynamics, the behaviour is similar to that of a white noise.

5.1.1 Model selection

In the following, we denote by p the measured gas pressure, that we wish to model, by l the measured (controlled) water level and by o the pressure valve openness. Figure 5.3 shows the signals used for the identification and validation of the GP model.

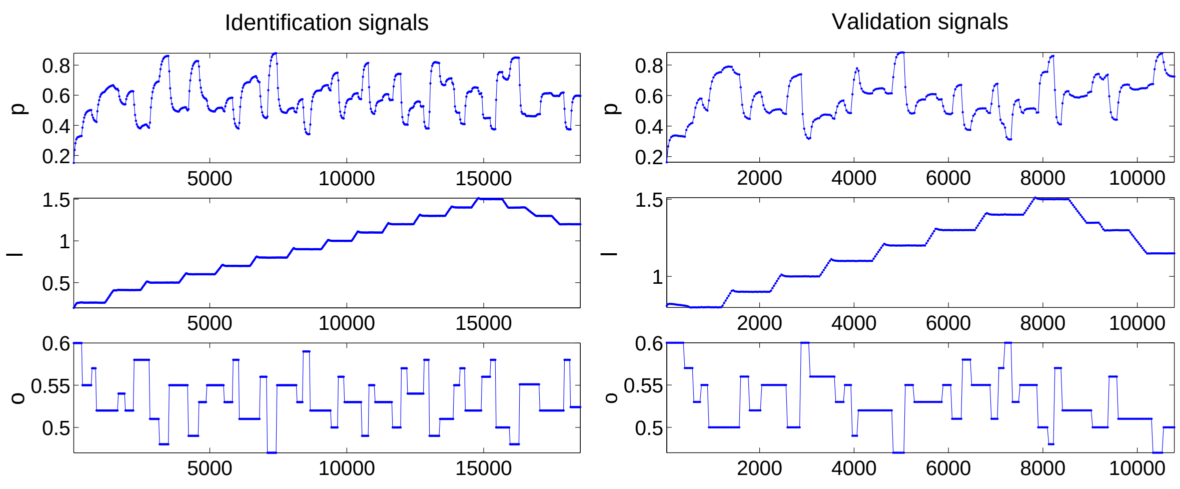

Figure 5.3: Identification (left) and validation (right) signals for the modelling of the gas-liquid separator. From top to bottom: pressure, water level and openness of the pressure valve, as a function of time.

For modelling the pressure in the separator, we assume the following NARX (nonlinear auto-regressive with exogeneous inputs) structure:

$$p(t) = f(p(t-1), \dots, p(t-L_1), l(t-1), \dots, l(t-L_2), o(t-1), \dots, o(t-L_3)) + \epsilon_t \ , \qquad (5.1)$$

where $p(t)$ is the gas pressure[1] at time t, $l(t-1)$ and $o(t-1)$ are the one-sample-delayed water level and openness of the pressure valve, both used as control inputs, and ϵ_t is a white noise with

[1] In experiments, we subtract the sample mean from the pressure signals.

unknown variance v_t. Using the identification signals, we then form the $N \times D$ matrix of inputs $\mathbf{x} = [p(t-1), \dots, p(t-L_1), l(t-1), \dots, l(t-L_2), o(t-1), \dots, o(t-L_3)]^T$, where $D = L_1 + L_2 + L_3$, and the $N \times 1$ vector of corresponding targets.

Although interactions are likely to exist between the delayed water level and the gas pressure in the separator, we use a GP with Gaussian covariance function and diagonal $\mathbf{W}$:

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

This means that there is one parameter w_d for each delayed variable. As in the Automatic Relevance Determination tool, developed by Neal and MacKay for the Bayesian treatment of neural networks (Neal, 1995; MacKay, 1994), these parameters give an indication of the relative importance of one delayed variable over an other. A large w_d implies a large contribution of the input in direction d to the covariance, which can then be interpreted as an 'important' input for prediction. Also, we have seen that w_d is inversely proportional to the horizontal scale of variation in direction d. Therefore, a large w_d corresponds to rapid variations in the same direction, in contrast with a small w_d, associated with a large correlation length and slow variation in that direction, implying that a change in the state will have little effect on the corresponding outputs.

Note that unless all inputs are normalised so as to vary in the same range, one should not compare the weights between different variables: The w of one variable may be 'small', if compared to that of another variable, but still be 'large' for the one variable considered. In that respect, we will only compare the w's relative to one variable only and denote by w_i^k the weight corresponding to the i^{th} delayed variable k, reflecting the relevance of the corresponding delayed variable.

Since we do not know a priori the optimal number of delayed variables, we perform a method of subset selection and train GPs with different structures (that is with varying number of delayed variables in the state $\mathbf{x}$). We first consider straightforward models, for which $L_1 = L_2 = L_3 = L$, for decreasing values of L:

- $M_1, \mathbf{x} = [p(t-1), p(t-2), p(t-3), l(t-1), l(t-2), l(t-3), o(t-1), o(t-2), o(t-3)]^T$;
- $M_2, \mathbf{x} = [p(t-1), p(t-2), l(t-1), l(t-2), o(t-1), o(t-2)]^T$;

- $M_3, \mathbf{x} = [p(t-1), l(t-1), o(t-1)]^T$.

We also test

- $M_4, \mathbf{x} = [p(t-1), l(t-1), l(t-3), o(t-1)]^T$, and

- $M_5, \mathbf{x} = [p(t-1), l(t-3), o(t-1)]^T$,

the rationale behind the choice of M_4 and M_5 being to keep only those relevant w_i^k, as found by previous models (treating one variable at a time). Table 5.1 indicates the Maximum Likelihood (ML) estimates of the w parameters corresponding to the delayed pressure variables appearing in the different models. Similarly, Tables 5.2 and 5.3 report the ML estimates of the parameters associated with the delayed water level and valve openness respectively.

Table 5.1: ML estimate of the w parameters for the delayed pressure values for the different models (for clarity, we indicate the corresponding delayed variable within brackets).

Model	w_3^p $[p(t-3)]$	w_2^p $[p(t-2)]$	w_1^p $[p(t-1)]$
M_1	0.0669	0.003	1.4702
M_2		0.2379	1.5962
M_3			2.4249
M_4			0.8621
M_5			2.4194

Table 5.2: ML estimate of the w parameters corresponding to delayed values of the water level.

Model	w_3^h $[l(t-3)]$	w_2^h $[l(t-2)]$	w_1^h $[l(t-1)]$
M_1	0.0165	0.0022	0.0124
M_2		0.0029	0.0009
M_3			0.1392
M_4	0.1897		0.1773
M_5	0.1337		

Starting with the model M_1, we see from Table 5.1 that most weight is assigned with w_1^p corresponding to $p(t-1)$, which is three orders of magnitude larger than w_2^p and two orders of magnitude

Table 5.3: ML estimate of the w parameters corresponding to delayed values of the valve openness.

Model	w_3^v $[o(t-3)]$	w_2^v $[o(t-2)]$	w_1^v $[o(t-1)]$
M_1	0.0764	2.0372	14.1725
M_2		1.3142	11.1741
M_3			45.9108
M_4			29.2038
M_5			48.2743

larger than w_3^p. Similarly, Table 5.2 shows that the weights associated with $l(t-1)$ and $l(t-3)$ are of the same order, both one order of magnitude larger than the weight associated with $l(t-2)$. As for the pressure, $o(t-1)$ is by far the most relevant delayed value of the valve openness. Keeping those delayed variables whose associated w^k is of the same order, we have the state structure of M_4, that is $\mathbf{x} = [p(t-1), l(t-1), l(t-3), o(t-1)]^T$. After training a GP using this structure, we find that the most relevant delayed variable for the pressure is $p(t-1)$, that of the water level is $l(t-3)$, and that for the openness of the valve is $o(t-1)$. These delayed variables then form the state for M_5.

We compare the models and rate them according to their predictive performance. For each model, we compute the one-step-ahead prediction of the identification data and the corresponding average squared error E_1 and average negative log-predictive density E_2. Table 5.4 indicates the results obtained, as well as the minus log-likelihood $\mathcal{L}(\mathbf{\Theta})$ of each model after learning.

Table 5.4: Negative log-likelihood and losses achieved by each model on the one-step ahead prediction of the identification data.

	M_1	M_2	M_3	M_4	M_5
$\mathcal{L}(\mathbf{\Theta})$ ($\times 10^3$)	-1.3565	-1.3558	-1.3423	-1.3469	-1.3389
E_1 ($\times 10^{-4}$)	4.6419	4.7662	4.9696	4.6757	4.9668
E_2	-2.4187	-2.4062	-2.3862	-2.4169	-2.3866

The model which has the highest probability (smallest minus log-likelihood) is M_1 and the one with lowest is M_5. However, in terms of losses, from best to worst, we have M_1, M_4, M_2, M_5, M_3. The fact that M_4, which considers $\mathbf{x} = [p(t-1), l(t-1), l(t-3), o(t-1)]^T$, does better than M_5 is an indication of the importance of the delayed $l(t-1)$, as M_5 ignores that delayed variable.

Also, the poorer result of M_2, compared to M_1, can be explained by the fact that the model, with $\mathbf{x} = [p(t-1), p(t-2), l(t-1), l(t-2), o(t-1), o(t-2)]^T$, is 'forced' to find a weighting between $l(t-1), l(t-2)$ and $p(t-1), p(t-2)$ although M_1 highlighted the greater significance of $l(t-3)$ and $p(t-3)$ over $l(t-2)$ and $p(t-2)$ respectively.

If we compute the ratios $E_1^{M_1}/E_1^{M_3}$ and $E_2^{M_1}/E_2^{M_3}$ (corresponding to the ratios of the training errors of the best model over that of the worst model), we find 0.9341 and 1.0136 respectively, meaning that models M_1 and M_3 are actually quite similar. It can then be argued that the increase of model complexity in M_1 might not be worth the computational effort, in terms of predictive performance of the model, when compared to a much simpler one like M_3 (following the Occam's razor principle, (Rasmussen and Ghahramani, 2001), favouring simple models in the face of the complexity/performance trade-off). Note if we artificially set to zero the parameters corresponding to $p(t-3), p(t-2), l(t-3), l(t-2), o(t-3), o(t-2)$, the minus log-likelihood of M_1 increases to -1.3326×10^3, making it less likely than M_3. This highlights the fact that simply ignoring the inputs thought not to be relevant is not enough: The model has to be re-trained, to allow a re-weighting of the remaining parameters.

In the light of these results, we choose to model the pressure signal using the structure of model M_3, that is

$$p(t) = f(p(t-1), l(t-1), o(t-1)) + \epsilon_t \, . \tag{5.2}$$

Recall that, for this model, after training a GP with Gaussian covariance function, we have $w_1 = 2.4249$, corresponding to $p(t-1)$, $w_2 = 0.1392$ and $w_3 = 45.9108$, corresponding to $l(t-1)$ and $o(t-1)$ respectively. Also, $v = 0.0748$ and the estimated noise level is $v_t = 0.0005$.

5.1.2 Validation

One-step-ahead prediction

We first test the model on the one-step-ahead prediction of the 'validation' pressure signal, which is composed of 336 points. Figure 5.4 (left) shows the mean predictions with their 2σ error-bars (to which we have added the model's estimate of the output noise variance v_t, since we predict the noisy time-series). The corresponding average losses obtained are $E_1 = 0.0010$ and $E_2 = -1.9106$.

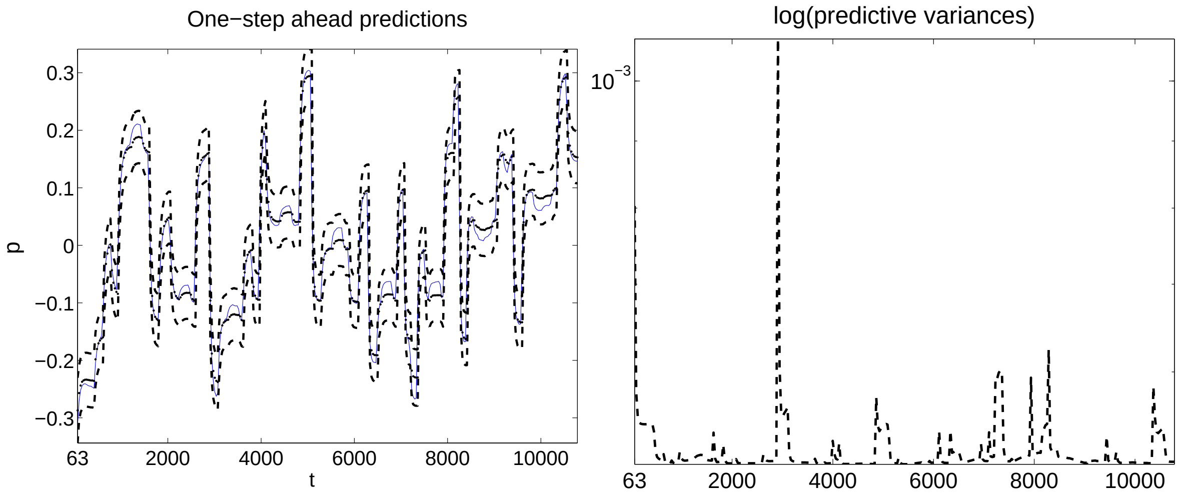

Figure 5.4: One-step ahead prediction of the test series, starting at $t = 63$ ms (thin line). Left: Mean predictions (thick continuous line) with their 2σ error-bars (dotted lines). Right: Plot of the predictive variances only.

The plot of the log of the predictive variances (right) clearly indicates that there are regions where the model becomes less accurate and confident about its predictions, which correspond to regions where the pressure varies slowly. This can be explained by the fact that the model was trained on rapidly varying signals (see the identification signals Figure 5.3) and therefore does not deal very well with slowly varying signals (explaining the greater uncertainty and the constant over- or under-shooting of the mean predictions in those regions). Also note the peak around 3000, due to the rapid changes of the pressure signal between two slowly varying regions.

Simulation

Since we do not know a priori the maximum prediction horizon (that is, the horizon up to which predictions are still 'reasonably good'), we run a simulation of the test set. In practice, the simulation, or 'infinite-step'-ahead prediction, corresponds to an N_t-steps ahead prediction, where N_t is the length of the test series.

For this simple model, the propagation of uncertainty within the *Gaussian approximation* is straightforward, as the water level l and the openness of the valve o signals are assumed to be known up to N_t, and there is only one delayed value of the pressure in the state (and therefore no cross-covariances to be computed as we predict ahead in time). Assuming we know the pressure p up to time t, we simply have

- $t+1$, $\mathbf{x}_{t+1} = [p(t), l(t), o(t)]^T$: Compute $p(t+1) \sim \mathcal{N}(\mu(\mathbf{x}_{t+1}), \sigma^2(\mathbf{x}_{t+1}) + v_t)$.
- $t+2$, $\mathbf{x}_{t+2} = [p(t+1), l(t+1), o(t+1)]^T \sim \mathcal{N}(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2})$ with

  $$\mathbf{u}_{t+2} = [\mu(\mathbf{x}_{t+1}, l(t+1), o(t+1)]^T$$

  and

  $$\mathbf{\Sigma}_{t+2} = \begin{bmatrix} \sigma^2(x_{t+1}) + v_t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

  provided we pass on the exact $l(t+1)$ and $o(t+1)$ to the model.
  Compute $p(t+2) \sim \mathcal{N}(m(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}), v(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) + v_t)$.
- $t+3$, $\mathbf{x}_{t+3} = [p(t+2), l(t+2), o(t+2)]^T \sim \mathcal{N}(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3})$, where $\mathbf{u}_{t+3} = [m(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}), l(t+2), o(t+2)]^T$ and

  $$\mathbf{\Sigma}_{t+3} = \begin{bmatrix} v(\mathbf{u}_{t+2}, \mathbf{\Sigma}_{t+2}) + v_t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

  Compute $p(t+3) \sim \mathcal{N}(m(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3}), v(\mathbf{u}_{t+3}, \mathbf{\Sigma}_{t+3}) + v_t)$.
- ...Up to the end of the test set.

In this experiment, the mean and variance at each time-step are computed using the exact equations corresponding to the Gaussian covariance function (A_{ex}).

The simulation (or 336-step-ahead prediction) of the test signal is shown in Figure 5.5. We compare the simulation with propagation of uncertainty (A_{ex}, as explained above) to the naive simulation

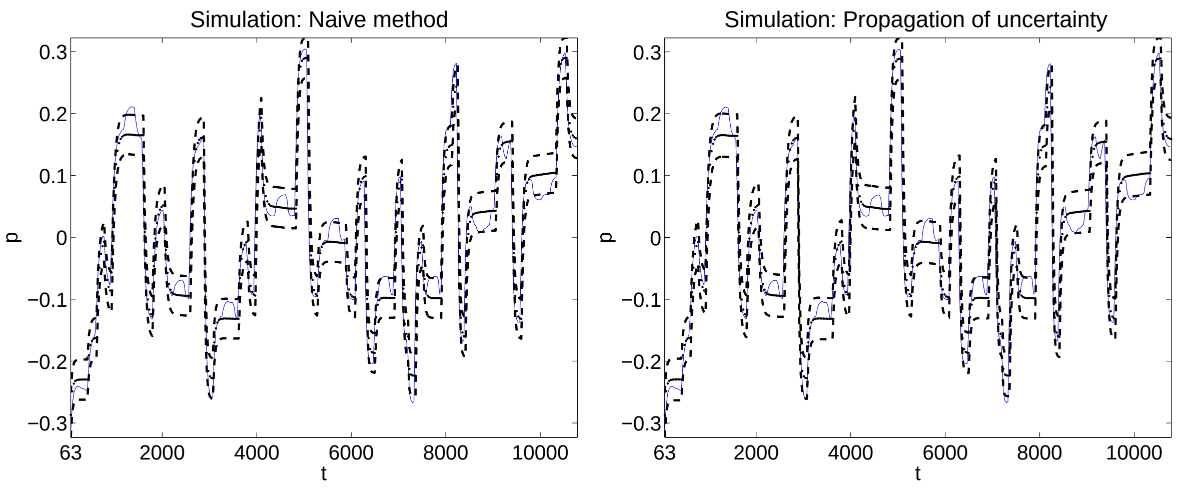

Figure 5.5: Simulation of the test pressure signal (thin line) from $t = 63$ ms. Mean predictions(thick continuous line) with their associated error-bars (dotted lines), given by the naive approach (left) and by with propagation of the uncertainty (right).

(N) that does not account for the uncertainty induced by each successive prediction. The left plot shows the predictions given by N and the right one those given by A_{ex}.

The mean predictions given by N and A_{ex} are extremely similar (the difference between the two is of the order 10^{-4}). Again, we can notice the under- and over-shooting of the model's estimates in slowly varying areas, most probably due to the fact that the model was trained using much more rapidly varying signals. Although not quite clear from these plots, the predictive variances (standard deviations squared) obtained with A_{ex} show more variability than those obtained with N, although both are in the same range. For this simulation, we have $E_1^{ex} = E_1^n = 0.0014$, $E_2^n = -1.5289$ and $E_2^{ex} = -1.6463$, where the upper-script ^{ex} refers to A_{ex} and ^{n} to N. So, in terms of negative log-predictive density, the propagation of uncertainty does better than the naive approach.

The simulation was started from the first point of the test set, for which the one-step ahead prediction errors are $E_1^{t_0} = 6.2501 \times 10^{-4}$ and $E_2^{t_0} = -2.2555$ (where ^{t_0} indicates the first point, that is at $t = 63$). We would expect a link between the point at which we start the simulation, the average performance of the model and the variation between naive and propagation of uncertainty approaches. That is, starting from a 'good' point (a point for which the one-step ahead predictive losses are good), we would expect the average losses (over the simulation) to be better than those obtained when starting from a 'bad' point, and the two methods to differ only slightly, the difference increasing if the starting point is one at which the model is already unsure about its prediction.

In order to validate our hypothesis, we ran the following experiments:

- S_1: Start the simulation at $t_1 = 5087$, corresponding to the point at which the one-step-ahead losses are the worst (that is $E_1^{t_1} = 0.0473$ and $E_2^{t_1} = 40.3721$);
- S_2: Start the simulation at $t_2 = 2847$, corresponding to the best one-step-ahead squared error, that is $E_1^{t_2} = 7.6528 \times 10^{-11}$ (for this point, we have $E_2^{t_2} = -2.8686$);
- S_3: Start the simulation at $t_3 = 767$, the point corresponding to the best one-step-ahead negative log-predictive density, that is $E_2^{t_3} = -2.8709$ ($E_1^{t_3} = 2.0305 \times 10^{-7}$ for this point).

The fact that the best one-step-ahead squared error and negative log-predictive density do not coincide with the same point highlights the difference between these two measures of 'goodness' of predictions. Recall that, whereas the squared error loss is only a function of the estimate (the predictive mean), the negative log-predictive density trades off the estimate and the uncertainty attached to it (an accurate estimate with large variance will have a larger E_2 than a rougher estimate with small variance).

In terms of average[2] squared error, we obtain $E_1^n = E_1^{ex} = 0.002$ for S_1 and $E_1^n = E_1^{ex} = 0.0019$ for S_2. For S_3, we have $E_1^n = 0.0013$ and $E_1^{ex} = 0.0012$. Table 5.5 reports the value of the average negative log-predictive density, computed for A_{ex} and N, for each simulation. To assess further the difference between the two methods, we compute the ratio of the corresponding losses: The closer this ratio is to one, the more similar are the two methods.

From this table, we conclude that A_{ex} does consistently better than the naive approach, and, as expected, the performance of the simulation is indeed affected by the starting point, if only slightly (better losses for S_2 and S_3 than for S_1). Also, the difference between the two methods is greater when starting at a 'bad' point than at a 'good' one (for S_1, the ratio E_2^{ex}/E_2^n is further from one than that

[2]Note that S_1 corresponds to the shortest simulation (with 179 points), whereas S_2 and S_3 imply the prediction of 249 and 314 points respectively. In order to compare fairly the losses between the simulations and average over the same number of points, we stopped the S_2 and S_3 simulations after 179 steps.

Table 5.5: Average negative log-predictive density for the different simulations.

Simulation	E_2^n	E_2^{ex}	E_2^{ex}/E_2^n
S_1	-0.9917	-1.1515	1.1611
S_2	-1.1226	-1.2886	1.1479
S_3	-1.6931	-1.8	1.0631

for S_2 and S_3), showing that the propagation of uncertainty copes better with the model's uncertainty than does the naive method, which simply ignores it.

The above reasoning can in fact be generalised to any point (not necessarily the starting point of a simulation): In general, once the model has passed through a 'bad' point, we expect the following predictions to be less accurate, as the model becomes less reliable from then on, with the two methods possibly diverging (although for some systems the model can 'catch' the state again).

For this application in particular, the propagation of uncertainty algorithm does not greatly improve the predictions: Although the system is dynamic and nonlinear, the GP, using the simple NARX structure with three delayed variables, could capture the dynamics of the model well enough to make good predictions over the whole length of the test signal.

5.2 The pH neutralisation process

We now turn to the more challenging problem of modelling a nonlinear pH neutralisation process, as depicted in Figure 5.6.

The process consists of an acid stream (Q_1), a buffer stream (Q_2) and a base stream (Q_3) that are mixed in a tank (T_1). The effluent pH is the measured variable and, in this study, it is controlled by manipulating the base flow rate with a flow control valve. This real dynamic system is well-known for its complexity as it contains various nonlinear elements. We refer to (Henson and Seborg, 1994) for more details.

After investigation into the identification of the process (Kocijan et al., 2002), it was found that

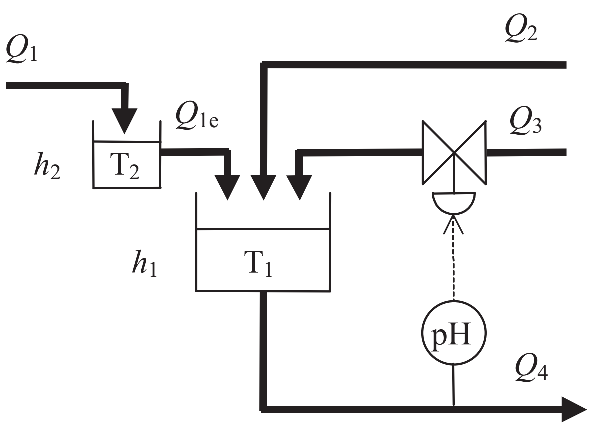

Figure 5.6: Schematic diagram of the pH neutralisation process.

the following NARX structure was suitable[3] to model the pH signal y, given the control input u:

$$y(t) = f(y(t-1), \dots, y(t-L), u(t-1), \dots, u(t-L)) \ , \tag{5.3}$$

with $L = 8$. Figure 5.7 shows portions of the signals used for identification (left) and validation (right) of the model.

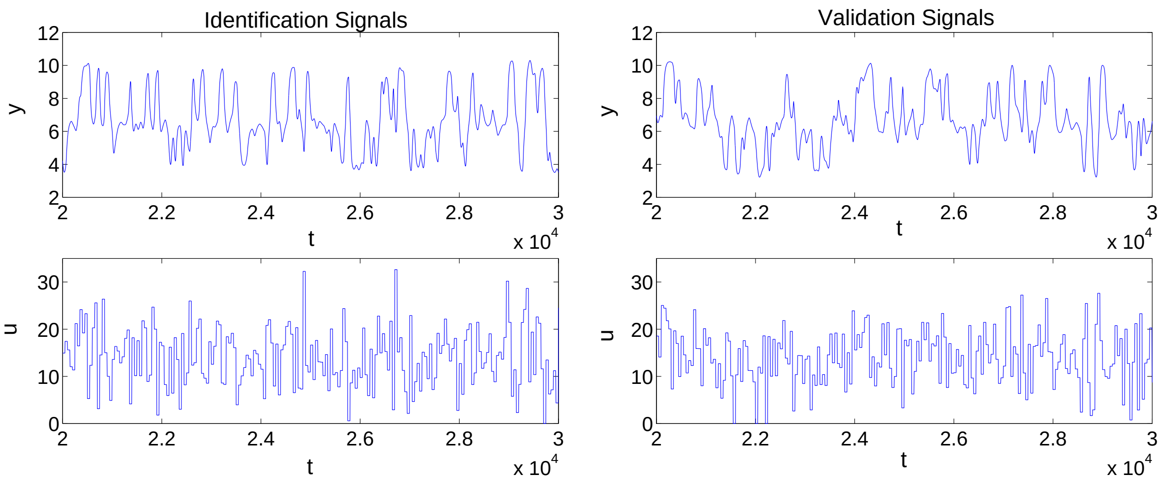

Figure 5.7: Portions of the real signals used for identification (left) and validation (right) of the model. The upper plots correspond to the measured pH as a function of time and the lower ones to the control input.

[3]That is in terms of trade-off between model complexity and average error on the training set.

5.2.1 'Mixed-model' identification

Since the signals are very long (15952 points), and the GP, as we use it,[4] cannot handle such a large data-set, we sub-sample the identification signals, resulting in a $N \times D$ matrix of training inputs, where $N = 1226$ and $D = 16$, and the corresponding $N \times 1$ vector of target pH values.

Although the process is nonlinear, a great deal of variation of the pH signal can be explained by a linear model and we decide to first fit a linear model to the data and then model the residuals with a GP (again with Gaussian covariance function). The classical linear model is

$$y_l(t) = \sum_{d=1}^{2L+1} x^d(t) w_l^d \, , \tag{5.4}$$

where w_l^d, for $d = 1 \dots 2L+1$, are the parameters of the linear model (w_l^{2L+1} corresponding to the bias) and $\mathbf{x}^+(t) = [y(t-1), \dots, y(t-L), u(t-1), \dots, u(t-L), 1]$ is the augmented input (the unit entry taking care of the bias). The Least Squares (LS) linear parameters are then given by $\mathbf{w}_l^{LS} = \mathbf{A}^{-1}\mathbf{b}$, where $A^{d,e} = \sum_{i=1}^N x_i^d x_j^e$ and $b^d = \sum_{i=1}^N y_i x_i^d$, for $d, e = 1 \dots 2L+1$. For this linear model, the average squared error on the training set is 0.0011. Figure 5.8 shows a portion of the residuals $y(t) - y_l(t)$ where $y_l(t) = \mathbf{w}_l^{LS^T} \mathbf{x}^+(t)$.

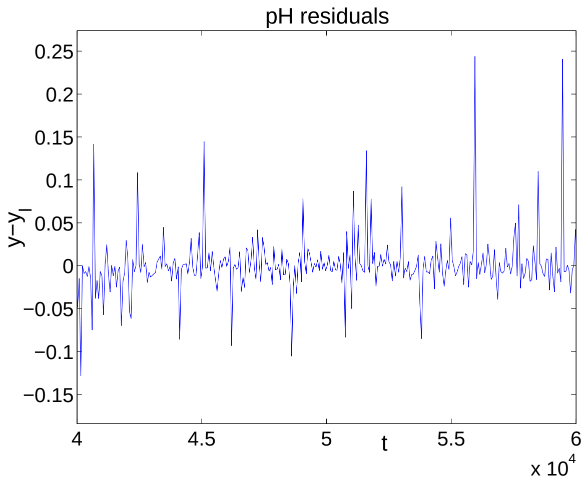

Figure 5.8: Plot of a subset of the residuals, after fitting a linear model.

[4]Refer to Chapter 2, Section 2.3.2.

Assuming

$$y_{gp}(t) = f(y(t-1), \ldots, y(t-L), u(t-1), \ldots, u(t-L)) \ , \tag{5.5}$$

where $y_{gp}(t) = y(t) - y_l(t)$, we model these residuals with a GP with Gaussian covariance function. For a given $\mathbf{x}(t)$, the GP then provides us with the predictive distribution $y_{gp}(t) \sim \mathcal{N}(\mu(\mathbf{x}(t)), \sigma^2(\mathbf{x}(t)))$, where $\mu(\mathbf{x}(t))$ corresponds to the estimate of the residual at time t.

Blending the linear and the GP models together, the estimate of the system's output $y(t)$ is given by $\mu(\mathbf{x}(t)) + y_l(t)$, with uncertainty $\pm 2\sigma(\mathbf{x}(t))$. For this 'mixed model', the training errors (i.e. average errors on the one-step-ahead predictions of the training set) are $E_1 = 1.1838 \times 10^{-7}$ and $E_2 = -5.6479$ (if we had fitted a GP directly onto the original data, we would have obtained $E_1 = 3.3420 \times 10^{-6}$ and $E_2 = -4.6854$). The training errors of the linear-GP model are then one order of magnitude better than those obtained for the GP alone. Figure 5.9 shows a portion of the one-step-ahead prediction of the validation pH signal (which contains 15952 points). The test errors are $E_1 = 3.9396 \times 10^{-4}$ and $E_2 = -3.3816$ (compared to $E_1 = 0.0022$ and $E_2 = -2.9150$ for the GP alone).

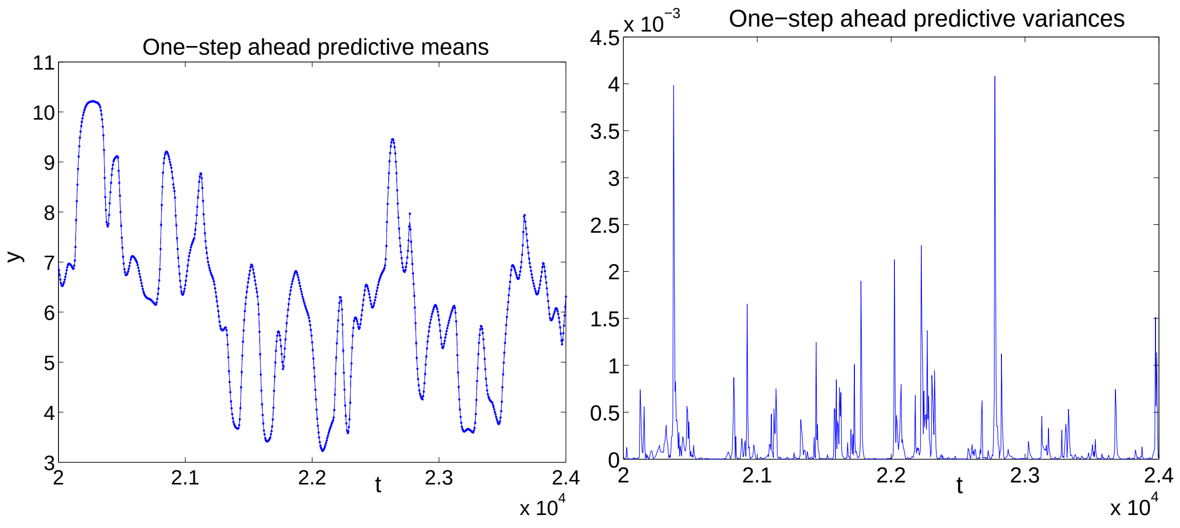

Figure 5.9: One-step-ahead prediction of the validation pH signal (continuous line) from $t = 2$ to $t = 2.4 \times 10^4$ ms with the model's mean predictions (dotted line). The right plot shows the corresponding predictive variances.

5.2.2 k-step ahead predictions with the linear model

If we now wish to make multiple-step-ahead predictions and propagate the uncertainty, we need to update the algorithm presented in Section 4.2.1 of the previous chapter, to account for the linear part at each time-step.

Modified algorithm

For simplicity, we denote by $\mathbf{x}_k$ the input at time $t+k$. At each time-step, '1.' refers to the contribution of the linear model and '2.' to that of the GP. The algorithm is now as follows:

- $t+1, \mathbf{x}_1 = [y(t), \dots, y(t-L), u(t), \dots, u(t-L)]^T$

  1. $y_l(t+1) = \mathbf{w}_l^{LS^T}\mathbf{x}_1^+$ (where $\mathbf{x}_1^+$ is the augmented input vector)
  2. $y_{gp}(t+1) \sim \mathcal{N}(\mu_{gp}(\mathbf{x}_1), \sigma^2_{gp}(\mathbf{x}_1))$

  $\rightarrow y(t+1) = y_l(t+1) + y_{gp}(t+1) \sim \mathcal{N}(m(\mathbf{x}_1), v(\mathbf{x}_1))$ with

$$m(\mathbf{x}_1) = \mu_{gp}(\mathbf{x}_1) + y_l(t+1), \quad v(\mathbf{x}_1) = \sigma^2_{gp}(\mathbf{x}_1)$$

- $t+2, \mathbf{x}_2 = [y(t+1), y(t), \dots, y(t+1-L), u(t+1), \dots, u(t+1-L)]^T \sim \mathcal{N}(\mathbf{u}_2, \mathbf{\Sigma}_2)$ with

$$\mathbf{u}_2 = [m(\mathbf{x}_1), y(t), \dots, y(t+1-L), u(t+1), \dots, u(t+1-L)]^T ,$$

  and $\Sigma_2^{11} = v(\mathbf{x}_1)$ and zero elsewhere,

  1. $y_l(t+2) \sim \mathcal{N}(m_l(\mathbf{u}_2, \mathbf{\Sigma}_2), v_l(\mathbf{u}_2, \mathbf{\Sigma}_2))$
  2. $y_{gp}(t+2) \sim \mathcal{N}(m_{gp}(\mathbf{u}_2, \mathbf{\Sigma}_2), v_{gp}(\mathbf{u}_2, \mathbf{\Sigma}_2))$

  $\rightarrow y(t+2) = y_l(t+2) + y_{gp}(t+2) \sim \mathcal{N}(m(\mathbf{u}_2, \mathbf{\Sigma}_2), v(\mathbf{u}_2, \mathbf{\Sigma}_2))$ with

$$\begin{aligned} m(\mathbf{u}_2, \mathbf{\Sigma}_2) &= m_l(\mathbf{u}_2, \mathbf{\Sigma}_2) + m_{gp}(\mathbf{u}_2, \mathbf{\Sigma}_2) \\ v(\mathbf{u}_2, \mathbf{\Sigma}_2) &= v_l(\mathbf{u}_2, \mathbf{\Sigma}_2) + v_{gp}(\mathbf{u}_2, \mathbf{\Sigma}_2) + 2\mathrm{Cov}[y_l(t+2), y_{gp}(t+2)] \end{aligned}$$

- $t+3, \mathbf{x}_3 = [y(t+2), y(t+1), y(t), \dots, y(t+2-L), u(t+2), \dots, u(t+2-L)]^T \sim \mathcal{N}(\mathbf{u}_3, \mathbf{\Sigma}_3)$

$$\mathbf{u}_3 = [m(\mathbf{u}_2, \mathbf{\Sigma}_2), m(\mathbf{x}_1), y(t), \dots, y(t+2-L), u(t+2), \dots, u(t+2-L)]^T$$

$$\mathbf{\Sigma}_3 = \begin{bmatrix} v(\mathbf{u}_2, \mathbf{\Sigma}_2) & \mathrm{Cov}[y(t+2), y(t+1)] & 0 & \dots & 0 \\ \mathrm{Cov}[y(t+1), y(t+2)] & v(\mathbf{x}_1) & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix},$$

1. $y_l(t+3) \sim \mathcal{N}(m_l(\mathbf{u}_3, \mathbf{\Sigma}_3), v_l(\mathbf{u}_3, \mathbf{\Sigma}_3))$
2. $y_{gp}(t+3) \sim \mathcal{N}(m_{gp}(\mathbf{u}_3, \mathbf{\Sigma}_3), v_{gp}(\mathbf{u}_3, \mathbf{\Sigma}_3))$

$\rightarrow y(t+3) \sim \mathcal{N}(m(\mathbf{u}_3, \mathbf{\Sigma}_3), v(\mathbf{u}_3, \mathbf{\Sigma}_3))$.

- ... Up to the desired horizon.

At each time-step, the predictive mean $m_{gp}(.)$ and variance $v_{gp}(.)$ given by the GP are computed exactly (for the Gaussian kernel) using equations (3.39) and (3.44).

We now need to compute $m_l(.)$ and $v_l(.)$, the predictive mean and variance given by the linear model, the cross-covariance between the linear and the GP models, arising in the predictive variance of the output at each time-step, and the cross-covariance terms of the input covariance matrix.

Linear part In general, at time-step k, given the input $\mathbf{x}_k \sim \mathcal{N}(\mathbf{u}_k, \mathbf{\Sigma}_k)$, it is straightforward that the linear contribution $y_l(t+k)$ has mean and variance

$$\begin{aligned} m_l(\mathbf{u}_k, \mathbf{\Sigma}_k) &= \mathbf{w}_l^{LS^T} \mathbf{u}_k^+ \\ v_l(\mathbf{u}_k, \mathbf{\Sigma}_k) &= \mathbf{w}_l^{LS-^T} \mathbf{\Sigma}_k \mathbf{w}_l^{LS-} \end{aligned}$$

where $\mathbf{w}_l^{LS-}$ is the vector of parameters without the bias term and $\mathbf{u}_k^+$ is the augmented input vector.

Output distribution At $t+k$, the 'mixed model' output is $y(t+k) = y_{gp}(t+k) + y_l(t+k) \sim \mathcal{N}(m(\mathbf{u}_k, \mathbf{\Sigma}_k), v(\mathbf{u}_k, \mathbf{\Sigma}_k))$ with mean

$$m(\mathbf{u}_k, \mathbf{\Sigma}_k) = m_{gp}(\mathbf{u}_k, \mathbf{\Sigma}_k) + m_l(\mathbf{u}_k, \mathbf{\Sigma}_k)$$

and variance

$$v(\mathbf{u}_k, \mathbf{\Sigma}_k) = v_{gp}(\mathbf{u}_k, \mathbf{\Sigma}_k) + v_l(\mathbf{u}_k, \mathbf{\Sigma}_k) + 2\mathrm{Cov}[y_{gp}(t+k), y_l(t+k)] \,.$$

We have $\text{Cov}[y_{gp}(t+k), y_l(t+k)] = E[y_{gp}(t+k)y_l(t+k)] - E[y_{gp}(t+k)]E[y_l(t+k)]$, where $E[y_{gp}(t+k)] = m_{gp}(\mathbf{u}_k, \mathbf{\Sigma}_k)$ and $E[y_l(t+k)] = m_l(\mathbf{u}_k, \mathbf{\Sigma}_k)$. We therefore need to compute

$$E[y_{gp}(t+k)y_l(t+k)] = \sum_i \beta_i \mathbf{w}_l^{LS-T} \int \mathbf{x}_k C(\mathbf{x}_k, \mathbf{x}_i) p(\mathbf{x}_k) d\mathbf{x}_k \ ,$$

where $\mathbf{x}_i$ denotes the i^{th} training case. As already seen in Chapter 4, Section 4.2.1, this integral can be evaluated exactly or approximately, depending on the form of the covariance function. In the case of the Gaussian kernel used in this experiment, we can directly write

$$\begin{aligned}\text{Cov}[y_{gp}(t+k), y_l(t+k)] = &\sum_i \beta_i C(\mathbf{u}_k, \mathbf{x}_i) C_{corr}(\mathbf{u}_k, \mathbf{x}_i) \left[(\mathbf{I} - \mathbf{W}(\mathbf{W} + \mathbf{\Sigma}_k)^{-1})\mathbf{x}_i \right. \\ &\left. - \mathbf{\Sigma}_k(\mathbf{W} + \mathbf{\Sigma}_k)^{-1}\mathbf{u}_k \right] \mathbf{w}_l^{LS-} \ .\end{aligned}$$

Input distribution Therefore, at time $t+k$, the input $\mathbf{x}_k = [y(t+k-1), \dots, y(t+k-L), u(t+k-1), \dots, u(t+k-L)]$ is a $2L \times 1$ random vector, normally distributed, with mean

$$\mathbf{u}_k = [m(\mathbf{u}_{k-1}, \mathbf{\Sigma}_{k-1}), \dots, m(\mathbf{u}_{k-L}, \mathbf{\Sigma}_{k-L}), u(t+k-1), \dots, u(t+k-L)]$$

and covariance matrix $\mathbf{\Sigma}_k$ with the delayed variances $v(\mathbf{u}_{k-i}, \mathbf{\Sigma}_{k-i})$ on the first L diagonal elements (the remaining elements $L+1, \dots, 2L$ corresponding to the delayed control input being zero). As seen in Section 4.2.1, computing the cross-covariances between the delayed outputs at each time-step corresponds to computing $\text{Cov}[y(t+k-l), \mathbf{x}_{k-l}]$, discarding the last element of $\mathbf{x}_{k-l}$. That is, with $y(t+k-l) = y_{gp}(t+k-l) + y_l(t+k-l)$,

$$\text{Cov}[y(t+k-l), \mathbf{x}_{k-l}] = \text{Cov}[y_{gp}(t+k-l), \mathbf{x}_{k-l}] + \text{Cov}[y_l(t+k-l), \mathbf{x}_{k-l}] \ ,$$

where we already know $\text{Cov}[y_{gp}(t+k-l), \mathbf{x}_{k-l}]$ (given by equation (4.8) for the Gaussian covariance function). For the linear part, we simply have

$$\begin{aligned}\text{Cov}[y_l(t+k-l), \mathbf{x}_{k-l}] &= E[y_l(t+k-l)\mathbf{x}_{k-l}] - E[y_l(t+k-l)]E[\mathbf{x}_{k-l}] \\ &= \mathbf{w}_l^{ML-T}[\mathbf{u}_{k-1}\mathbf{u}_{k-1}^T + \mathbf{\Sigma}_{k-1}] - \mathbf{w}_l^{ML-T}\mathbf{u}_{k-1}\mathbf{u}_{k-1}^T \\ &= \mathbf{w}_l^{ML-T}\mathbf{\Sigma}_{k-1} \ .\end{aligned}$$

We are now going to apply these results to the 20-step ahead prediction of the test pH signal. Furthermore, we compare the predictive performance of the iterative method to that of the direct model, trained to predict k-steps ahead (where again we consider a linear model first and then a GP).

Results

We perform k-step-ahead predictions, where $k = 20$, starting at 226 points, taken at random from the test set. Figure 5.10 shows the iterative k-step-ahead predictive means and Figure 5.11 shows the predictive variances, for $k = 5, 10, 15, 20$, obtained for the naive approach N (using the 'certainty principle' on the estimates at each time-step) and the propagation of uncertainty A_{ex} (using the exact moments). Note that no time index is given because each point **is** a k-step ahead prediction.

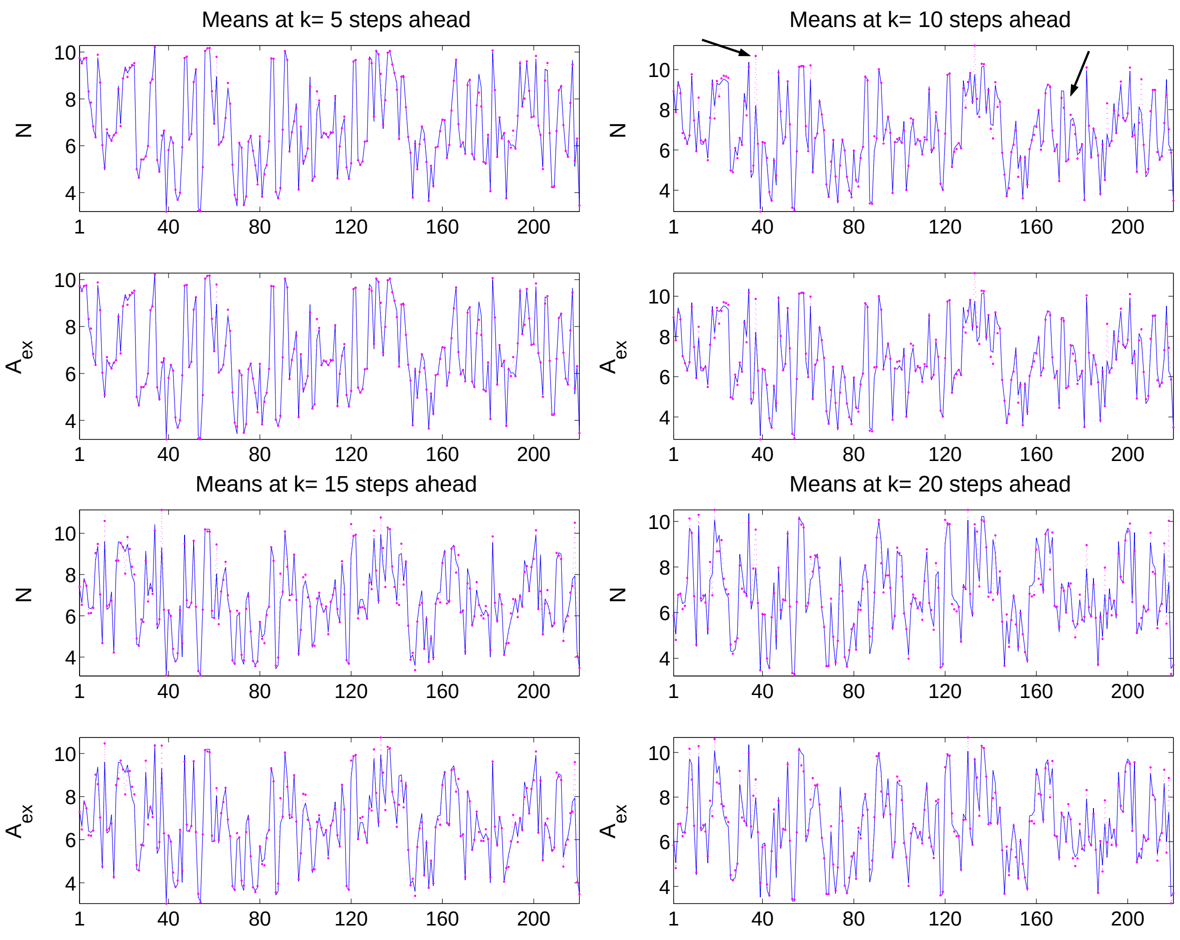

Figure 5.10: True pH values (continuous line) and predictive means (dotted lines) at k steps ahead (note that each point is a k step ahead prediction, so that the x-axis simply indicates the sample number, not time). In each figure, the top plots indicate the means given by the naive approach and the bottom ones those obtained when propagating the uncertainty.

From Figure 5.10, we see that the 5-step-ahead predictive means given by N and A_{ex} are similar but start differing from $k = 10$. Around the 40^{th} 10-step-ahead prediction (indicated by an arrow), the

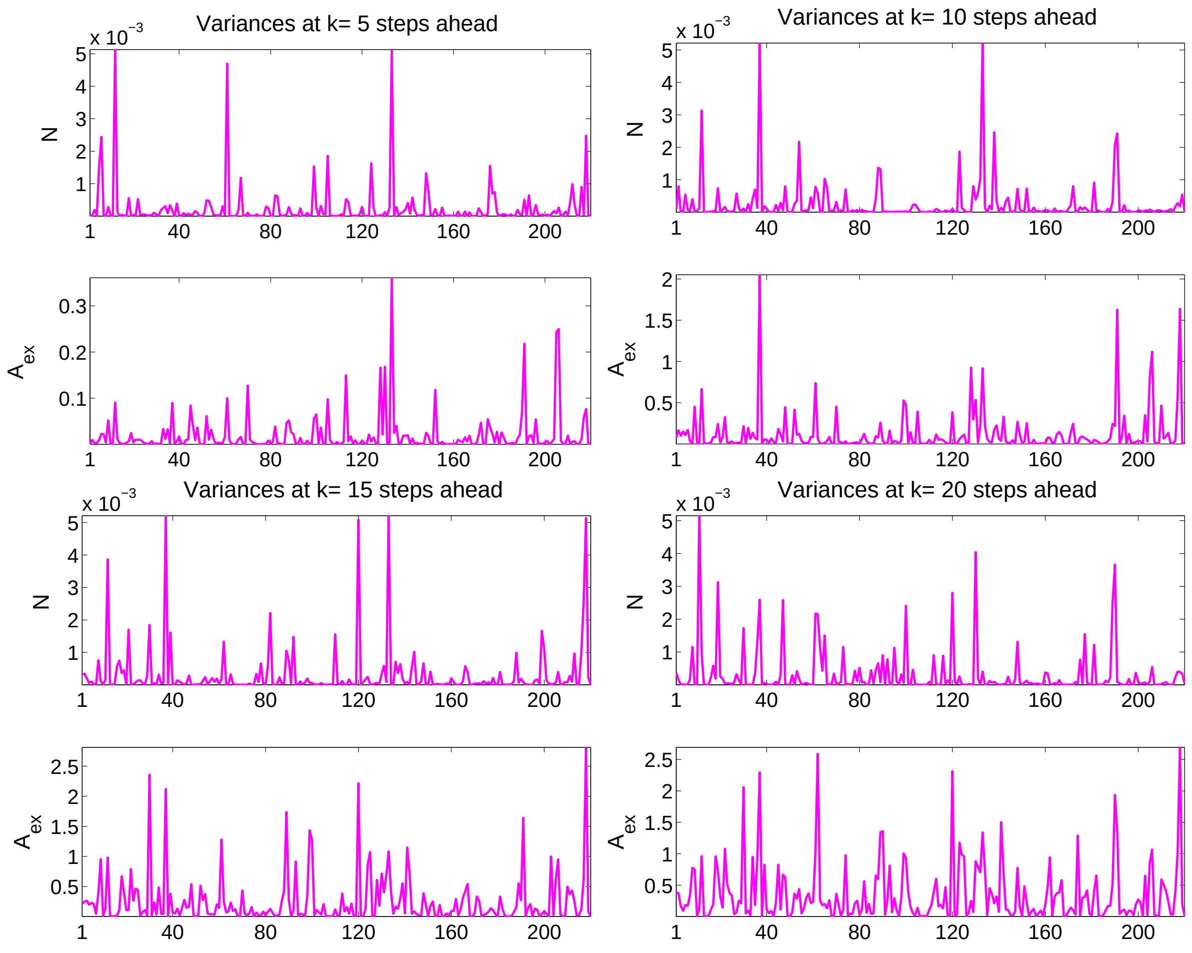

Figure 5.11: k-step-ahead predictive variances, for different values of k. Same legend as for Figure 5.10.

naive approach overshoots more than A_{ex}, and undershoots around the 170^{th} prediction (see arrow). The same behaviour occurs for $k = 15$ and $k = 20$, around the 220^{th} point. In terms of variances, we can note that, whatever the prediction horizon is, the predictive variances given by the naive approach are always very small. Also, the variances given by A_{ex} are one order of magnitude different between $k = 5$ and $k = 10$, but stay in the same range for $k > 10$ (there is no 'blowing up effect' for large k). Regardless of the amplitude of the variances, we see that the models do not necessarily agree on which points are 'certain' and which ones are not (e.g. for $k = 5$, the 10^{th} prediction is very uncertain according to N but not for A_{ex}, similarly for the 140^{th} 15-step ahead prediction).

Let us now look more closely at the predictions from one to 20-steps ahead for different starting points. Let M be the point at which the variance given by the exact method at 20 steps ahead is maximum and m the point at which it is minimum. Figure 5.12 shows the prediction from one to $k = 20$ steps ahead, starting at M and m (the crosses indicating the moments given by A_{ex} and the dots those given by A_n). From these plots, we see that, starting from m, the mean predictions given by the two methods are really good. Interestingly, the variances given by A_{ex} start decreasing after 6 steps ahead. When starting from M, as the mean predictions get worse, the uncertainty of the model A_{ex} also increases, which is not the case for the naive model.

The values of the average errors given by Table 5.6 confirm our remarks concerning the iterative predictions, with and without propagation of the uncertainty.

Table 5.6: Average squared error E_1 and negative log-predictive density E_2 for different values of k.

	$k = 5$		$k = 10$		$k = 15$		$k = 20$	
	E_1	E_2	E_1	E_2	E_1	E_2	E_1	E_2
N	0.0275	105	0.1187	725	0.1880	2096	0.2243	13652
A_{ex}	0.0263	-1.1315	0.0899	-0.3946	0.1005	0.0028	0.1233	0.2053
D	0.0562	-0.0194	0.0481	0.1431	0.0456	-0.0523	0.0554	0.0184

As k increases, A_{ex} becomes one order of magnitude better than N, in terms of E_1. The very large values for E_2 obtained for the naive approach can be explained by the fact that, as k increases, the model's uncertainty stays small, even though the estimates are not accurate. We also report the losses

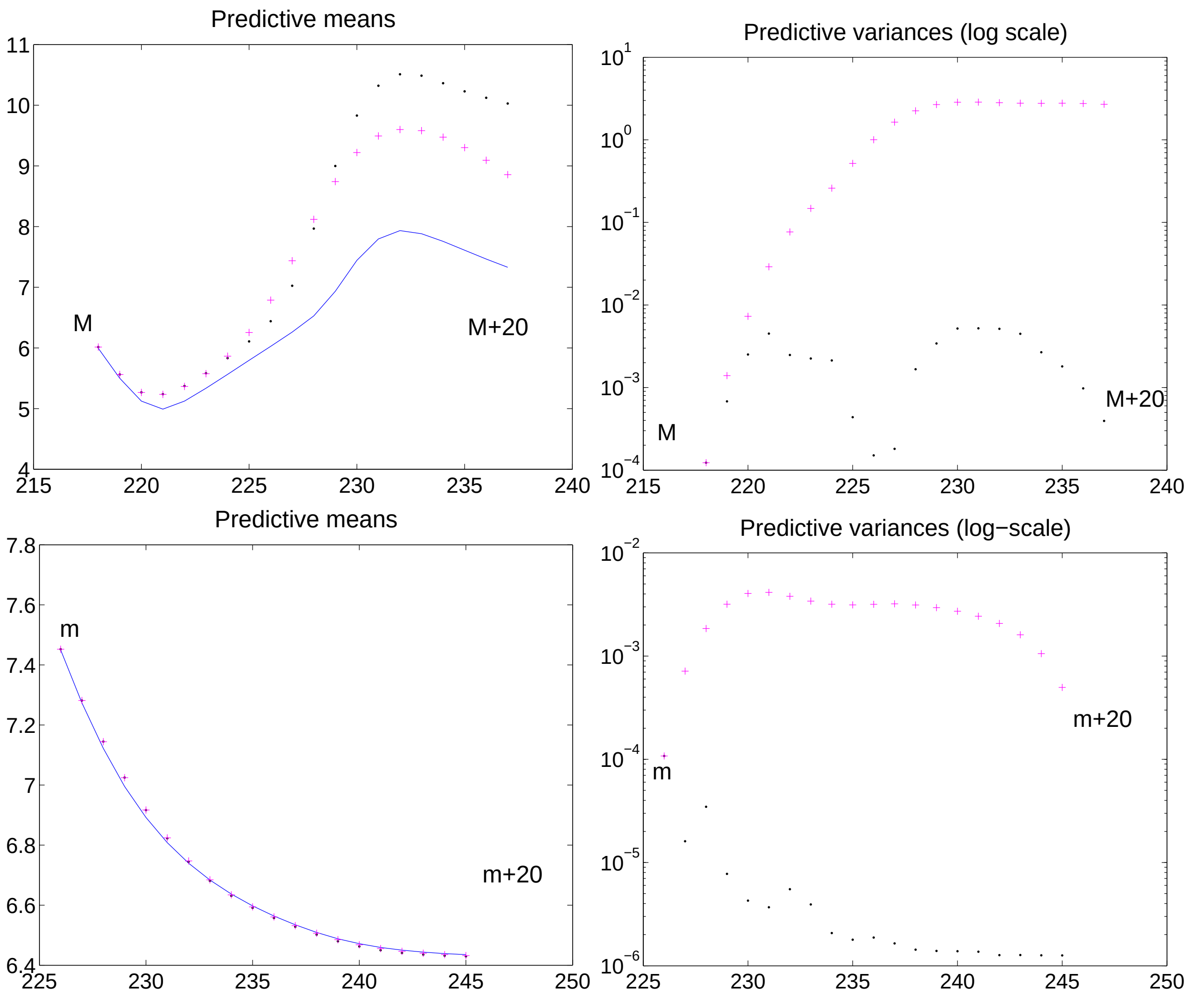

Figure 5.12: Top plots: pH signal (continuous line) from 1 to 20 steps ahead, starting at point M, for which the variance at 20 steps ahead is maximum. The crosses correspond to the means (left) and variances (right) obtained when propagating the uncertainty and the dots to the naive approach. Bottom plots: Same as above, starting at m, the point at which the variance at 20 steps ahead is minimum.

obtained when training a model to directly predict k-steps-ahead (D). We can see that for relatively short prediction horizons, the iterative method with propagation of uncertainty performs better than the direct method, which is due to the accuracy of the one-step ahead model. But as the predictive horizon grows larger, the direct model shows a better predictive performance. Although the number of missing data increases with k, for this particular example, the direct model does not face difficulties to learn the mapping as the number of data points is very large (1228 training cases are available to learn the 20-step-ahead model, for which the input dimension is 35).

5.3 Concluding remarks

In this chapter, we have highlighted how the GP model could be effectively used to model and forecast real dynamic systems. In particular, for the gas-liquid process, we have seen how the parameters of the covariance function could be used to identify the structure of the NARX model. We simply would like to recall that the ARD tool should be used with caution. Only those parameters associated with the same (delayed) variables should be compared, not all parameters between them, as different variables are likely to vary in different ranges. Also, simply setting to zero the parameters of less relevant variables is not enough: A model with the corresponding simpler structure (i.e. ignoring those variables) should be re-trained, as we saw that the likelihood of M_1 is less than that of M_3, when setting to zero those parameters in M_1 corresponding to the variables not accounted for in M_3.

Another important remark is that, the intuitive idea that a model trained on a ‘rich’ rapidly varying signal should automatically generalise well on ‘simpler’ signals is not true, as we could see for the gas-liquid system. The model trained on rapidly varying signals has mean predictions over- or under-shooting in those regions where the test signal varies slowly. Fortunately, this is reflected in the larger predictive variances, indicating that the model is less confident in its estimates.

The comparison of the predictions obtained with and without propagation of the uncertainty indicate that, in general, propagating the uncertainty does improve the multiple-step-ahead predictions, if only with respect to the predictive variances. Also, if the model passes through a ‘new’ region, inducing a larger predictive variance for the corresponding outputs, a model propagating the uncertainty is more likely to be able to ‘catch’ the state again and ‘rectify’ its predictions afterwards.

At last, we have shown how a linear model could be elegantly added and accounted for in the k-step-ahead iterative prediction with propagation of the uncertainty.

Likewise, we are now going to show how derivative observations can be easily accounted for in the model, and then extend the propagation of uncertainty algorithm in this case.

Chapter 6

Including derivative observations and allowing cautious control

In this chapter, we present the methodology for using Gaussian Processes, and the approach we have developed to account for the uncertainty, in two important applications. In Section 6.1, we first show how derivative observations can be incorporated into a GP model where function observations are already available, which is of particular importance in engineering applications, for the identification of nonlinear dynamic systems from experimental data. For this mixed training set, we then derive the expressions of the predictive mean and variance of a function output corresponding to a noisy input, within the Gaussian approximation. The second part of this chapter is devoted to the use of Gaussian Processes in a control context. Of particular interest are the Model Predictive Control framework and adaptive controllers, where the controllers present a 'cautious' feature, arising in an objective function that does not disregard the model's uncertainty.

6.1 Incorporating derivative observations in a GP model

Accounting for derivative observations, either directly measured or obtained from linearisations, in the GP framework has been addressed in (Solak et al., 2003). In the following, we first explicitly write down the expressions of the predictive mean and variance of a function output, accounting for the *mixed* nature of the training set. We then use these results to derive the mean and variance of the output corresponding to a noisy test input.

6.1.1 Derivative process

Let us first consider the Gaussian Process $y = f(x)$ with a one-dimensional argument x, with mean function $m_y(x)$ and covariance function $C_y(x_i, x_j)$.

It can be shown (Pugachev, 1967) that the derivative process $f'(x) = \frac{\partial f(x)}{\partial x}$, that we denote by y^1, is also a Gaussian Process with mean and covariance function respectively given by

$$m_{y^1}(x) = \frac{\partial m_y(x)}{\partial x}, \quad C_{y^1}(x_i, x_j) = \frac{\partial^2 C_y(x_i, x_j)}{\partial x_i x_j} \; . \tag{6.1}$$

Also, the cross-covariance function between the two processes is

$$C_{yy^1}(x_i, x_j) = \frac{\partial C_y(x_i, x_j)}{\partial x_j} \; . \tag{6.2}$$

In the general case where the argument $\mathbf{x}$ is D-dimensional, we will consider one derivative at a time, denoting by y^d the first derivative of y in direction d, i.e. $y^d = \frac{\partial f(\mathbf{x})}{\partial x^d}$. In this case, the covariance function of y^d is

$$C_{y^d}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial^2 C_y(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_i^d x_j^d} \; , \tag{6.3}$$

the cross-covariance between y^d and y is

$$C_{yy^d}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial C_y(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_j^d} \; , \tag{6.4}$$

and that between y^d and, say, y^e is

$$C_{y^d y^e}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial^2 C_y(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_i^d \partial x_j^e} \; . \tag{6.5}$$

We refer to Appendix A for operations on random functions.

6.1.2 Output predictive distribution for a new input

Consider the case where we have N observations of the function output, $y_1, \dots, y_N$, and N_d observations of the first derivative[1] in direction d, $y_1^d, \dots, y_{N_d}^d$ so that we allow the number of points to differ

[1]The derivative observations need not be taken at the same inputs as the function observations.

from one derivative[2] to another. Note that, in the following, we make the simplification of considering noise-free function and derivative observations.

We refer to such a data set as a *mixed* training set. The complete vector of observations is then

$$\mathbf{t} = [y_1 \dots y_N, y_1^1, \dots, y_{N_1}^1, \dots, y_1^d, \dots, y_{N_d}^d, \dots, y_1^D, \dots, y_{N_D}^D]^T ,$$

and we can partition the data covariance matrix $\mathbf{K}$ accordingly, into blocks corresponding to the function observations only, the derivative observations and the cross-covariances between them:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{yy} & \dots & \mathbf{K}_{yy^d} & \dots & \mathbf{K}_{yy^D} \\ \vdots & & \vdots & & \vdots \\ \mathbf{K}_{y^d y} & \dots & \mathbf{K}_{y^d y^d} & \dots & \mathbf{K}_{y^d y^D} \\ \vdots & & \vdots & & \vdots \\ \mathbf{K}_{y^D y} & \dots & \mathbf{K}_{y^D y^d} & \dots & \mathbf{K}_{y^D y^D} \end{bmatrix} ,$$

where $\mathbf{K}_{yy}$ denotes the $N \times N$ matrix of covariances between function observations ($K_{yy}^{ij} = \mathrm{Cov}[y_i, y_j]$, for $i, j = 1 \dots N$), $\mathbf{K}_{yy^d}$ the $N \times N_d$ matrix of covariances between function observations and derivatives in direction d ($K_{yy^d}^{ij} = \mathrm{Cov}[y_i, y_j^d]$, for $i = 1 \dots N$, $j = 1 \dots N_d$), $\mathbf{K}_{y^d y^d}$ the $N_d \times N_d$ matrix of covariances between derivative observations in direction d ($K_{y^d y^d}^{ij} = \mathrm{Cov}[y_i^d, y_j^d]$, for $i, j = 1 \dots N_d$) and, in general, $\mathbf{K}_{y^d y^e}$ the $N_d \times N_e$ matrix of covariances between derivative observations in direction d and direction e ($K_{y^d y^e}^{ij} = \mathrm{Cov}[y_i^d, y_j^e]$, for $i = 1 \dots N_d$, $j = 1 \dots N_e$ and $d, e = 1 \dots D$). All these covariances can be readily computed for a given covariance function, using the formulae given in 6.1.1.

Given the full vector of observations and the covariance matrix $\mathbf{K}$, the learning can be achieved as in Section 2.3.2 of Chapter 2, by minimising $\mathcal{L}(\boldsymbol{\Theta}) = -\log[p(\mathbf{t}|\mathbf{X})]$, where $\mathbf{X}$ is the matrix of the inputs at which function and derivative observations are made and $\boldsymbol{\Theta}$ are the parameters of the covariance function (see (Solak et al., 2003)). Note that the model need not be trained on the full data set, as the training of the model can be achieved using the function observations only. Providing the derivative observations would not affect the model (in terms of carrying more information likely to

[2]By which we understand the first derivative in a given dimension.

change the value of the parameters learnt), they could then be added to the model, or possibly be used in place of some function observations for making predictions, the latter enabling a speed-up in the computations.

Given a new input $\mathbf{x}$, as in the 'usual' GP case (Chapter 2, Section 2.3.3), the predictive distribution of the corresponding function point $y = f(\mathbf{x})$ is Gaussian, with mean and variance respectively given by

$$\mu(\mathbf{x}) = \sum_i \beta_i C(\mathbf{x}, \mathbf{x}_i) \tag{6.6}$$

$$\sigma^2(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - \sum_{i,j} K_{ij}^{-1} C(\mathbf{x}, \mathbf{x}_i) C(\mathbf{x}, \mathbf{x}_j) \ , \tag{6.7}$$

where $\boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{t}$ and $i, j = 1 \dots (N + N_1 + \dots + N_D)$.

When computing the covariances between the function output at the new $\mathbf{x}$ and the observations at the training points, we can distinguish between function and derivative observations and write

$$\mu(\mathbf{x}) = \sum_{i=1}^{N} \beta_i \mathrm{Cov}[y, y_i] + \sum_{d=1}^{D} \sum_{i=1}^{N_d} \beta_i \mathrm{Cov}[y, y_i^d] \ , \tag{6.8}$$

that is

$$\mu(\mathbf{x}) = \sum_{i=1}^{N} \beta_i C_y(\mathbf{x}, \mathbf{x}_i) + \sum_{d=1}^{D} \sum_{i=1}^{N_d} \beta_i C_{yy^d}(\mathbf{x}, \mathbf{x}_i) \tag{6.9}$$

for the mean, and similarly

$$\begin{aligned}\sigma^2(\mathbf{x}) =& \mathrm{Cov}[y, y] - \left[\sum_{i,j=1}^{N} K_{ij}^{-1} \mathrm{Cov}[y, y_i] \mathrm{Cov}[y, y_j] + \sum_{d,e=1}^{D} \sum_{i=1}^{N_d} \sum_{j=1}^{N_e} K_{ij}^{-1} \mathrm{Cov}[y, y_i^d] \mathrm{Cov}[y, y_j^e] \right. \\ & \left. + 2 \sum_{d=1}^{D} \sum_{i=1}^{N} \sum_{j=1}^{N_d} K_{ij}^{-1} \mathrm{Cov}[y, y_i] \mathrm{Cov}[y, y_j^d] \right] \ ,\end{aligned} \tag{6.10}$$

or

$$\begin{aligned}\sigma^2(\mathbf{x}) =& C_y(\mathbf{x}, \mathbf{x}) - \left[\sum_{i,j=1}^{N} K_{ij}^{-1} C_y(\mathbf{x}, \mathbf{x}_i) C_y(\mathbf{x}, \mathbf{x}_j) + \sum_{d,e=1}^{D} \sum_{i=1}^{N_d} \sum_{j=1}^{N_e} K_{ij}^{-1} C_{yy^d}(\mathbf{x}, \mathbf{x}_i) C_{yy^e}(\mathbf{x}, \mathbf{x}_j) \right. \\ & \left. + 2 \sum_{d=1}^{D} \sum_{i=1}^{N} \sum_{j=1}^{N_d} K_{ij}^{-1} C_y(\mathbf{x}, \mathbf{x}_i) C_{yy^d}(\mathbf{x}, \mathbf{x}_j) \right]\end{aligned} \tag{6.11}$$

for the variance.

Let

$$\mu_y(\mathbf{x}) = \sum_{i=1}^{N} \beta_i C_y(\mathbf{x}, \mathbf{x}_i) \tag{6.12}$$

$$\sigma^2_y(\mathbf{x}) = C_y(\mathbf{x}, \mathbf{x}) - \sum_{i,j=1}^{N} K^{-1}_{ij} C_y(\mathbf{x}, \mathbf{x}_i) C_y(\mathbf{x}, \mathbf{x}_j) \tag{6.13}$$

be the mean and variance relative to the function observations only,

$$\mu_{y^d}(\mathbf{x}) = \sum_{i=1}^{N_d} \beta_i C_{yy^d}(\mathbf{x}, \mathbf{x}_i) \tag{6.14}$$

$$\sigma^2_{y^d y^e}(\mathbf{x}) = -\sum_{i=1}^{N_d} \sum_{j=1}^{N_e} K^{-1}_{ij} C_{yy^d}(\mathbf{x}, \mathbf{x}_i) C_{yy^e}(\mathbf{x}, \mathbf{x}_j) \tag{6.15}$$

the moments relative to the derivative observations, and

$$\sigma^2_{yy^d}(\mathbf{x}) = -\sum_{i=1}^{N} \sum_{j=1}^{N_d} K^{-1}_{ij} C_y(\mathbf{x}, \mathbf{x}_i) C_{yy^d}(\mathbf{x}, \mathbf{x}_j) \ , \tag{6.16}$$

the part accounting for the cross-covariance between function and derivative observations. The predictive mean and variance can then be written

$$\mu(\mathbf{x}) = \mu_y(\mathbf{x}) + \sum_{d=1}^{D} \mu_{y^d}(\mathbf{x}) \tag{6.17}$$

$$\sigma^2(\mathbf{x}) = \sigma^2_y(\mathbf{x}) + \sum_{d,e=1}^{D} \sigma^2_{y^d y^e}(\mathbf{x}) + 2 \sum_{d=1}^{D} \sigma^2_{yy^d}(\mathbf{x}) \ , \tag{6.18}$$

reflecting the ‘mixed’ nature of the data set.

6.1.3 Prediction at a noisy input

We now turn to the problem of making a prediction given a noisy or random $\mathbf{x} \sim \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x)$. We do so in the *Gaussian approximation* presented in Chapter 3, Section 3.2.2. That is, we only compute the mean and variance of the (non-Gaussian) predictive distribution. As seen then, these two moments are given by

$$m(\mathbf{u}, \mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu(\mathbf{x})] \tag{6.19}$$

$$v(\mathbf{u}, \mathbf{\Sigma}_x) = E_{\mathbf{x}}[\sigma^2(\mathbf{x})] + E_{\mathbf{x}}[\mu(\mathbf{x})^2] - m(\mathbf{u}, \mathbf{\Sigma}_x)^2 \ , \tag{6.20}$$

where $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ are now given by equations (6.17) and (6.18) respectively.

Replacing $\mu(\mathbf{x})$ by its expression, we have

$$m(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}\left[\mu_y(\mathbf{x}) + \sum_{d=1}^{D}\mu_{y^d}(\mathbf{x})\right] = E_{\mathbf{x}}[\mu_y(\mathbf{x})] + \sum_{d=1}^{D} E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]\ .$$

Let $m_y(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu_y(\mathbf{x})]$ and $m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]$. The predictive mean corresponding to the noisy $\mathbf{x}$ can be written

$$m(\mathbf{u},\mathbf{\Sigma}_x) = m_y(\mathbf{u},\mathbf{\Sigma}_x) + \sum_{d=1}^{D} m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)\ , \tag{6.21}$$

which corresponds to the sum of the 'noisy' predictive mean when only function observations are available (which is the case derived in Chapter 3), and that corresponding to the derivative observations in each dimension.

We can work out a similar expression for the variance. Replacing $\sigma^2(\mathbf{x})$ by its expression, we first have

$$\begin{aligned} E_{\mathbf{x}}[\sigma^2(\mathbf{x})] &= E_{\mathbf{x}}\left[\sigma_y^2(\mathbf{x}) + \sum_{d,e=1}^{D}\sigma^2_{y^d y^e}(\mathbf{x}) + 2\sum_{d=1}^{D}\sigma^2_{yy^d}(\mathbf{x})\right] \\ &= E_{\mathbf{x}}[\sigma_y^2(\mathbf{x})] + E_{\mathbf{x}}\left[\sum_{d,e=1}^{D}\sigma^2_{y^d y^e}(\mathbf{x})\right] + 2\sum_{d=1}^{D}E_{\mathbf{x}}[\sigma^2_{yy^d}(\mathbf{x})]\ . \end{aligned}$$

Also,

$$\begin{aligned} E_{\mathbf{x}}[\mu(\mathbf{x})^2] &= E_{\mathbf{x}}\left[\mu_y(\mathbf{x})^2 + 2\mu_y(\mathbf{x})\sum_{d=1}^{D}\mu_{y^d}(\mathbf{x}) + \left(\sum_{d=1}^{D}\mu_{y^d}(\mathbf{x})\right)^2\right] \\ &= E_{\mathbf{x}}[\mu_y(\mathbf{x})^2] + 2\sum_{d=1}^{D}E_{\mathbf{x}}[\mu_y(\mathbf{x})\mu_{y^d}(\mathbf{x})] + E_{\mathbf{x}}\left[\left(\sum_{d=1}^{D}\mu_{y^d}(\mathbf{x})\right)^2\right] \end{aligned}$$

with $E_{\mathbf{x}}\left[\left(\sum_{d=1}^{D}\mu_{y^d}(\mathbf{x})\right)^2\right] = \sum_{d,e=1}^{D}E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x})]$.

Putting the different terms together, we have

$$\begin{aligned} v(\mathbf{u},\mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma_y^2(\mathbf{x})] + \sum_{d,e=1}^{D} E_{\mathbf{x}}[\sigma_{y^d y^e}^2(\mathbf{x})] + 2\sum_{d=1}^{D} E_{\mathbf{x}}[\sigma_{yy^d}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x})^2] \\ &+ \sum_{d,e=1}^{D} E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x})] + 2\sum_{d=1}^{D} E_{\mathbf{x}}[\mu_y(\mathbf{x})\mu_{y^d}(\mathbf{x})] - m(\mathbf{u},\mathbf{\Sigma}_x)^2 \ , \end{aligned}$$

where

$$m(\mathbf{u},\mathbf{\Sigma}_x)^2 = E_{\mathbf{x}}[\mu_y(\mathbf{x})]^2 + E_{\mathbf{x}}\left[\sum_{d=1}^{D}\mu_{y^d}(\mathbf{x})\right]^2 + 2\sum_{d=1}^{D} E_{\mathbf{x}}[\mu_y(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]$$

and $E_{\mathbf{x}}\left[\sum_{d=1}^{D}\mu_{y^d}(\mathbf{x})\right]^2 = \sum_{d,e=1}^{D} E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^e}(\mathbf{x})]$. Letting

$$v_y(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\sigma_y^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x})^2] - E_{\mathbf{x}}[\mu_y(\mathbf{x})]^2 \tag{6.22}$$

$$v_{y^d y^e}(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\sigma_{y^d y^e}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x})] - E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^e}(\mathbf{x})] \tag{6.23}$$

$$v_{yy^d}(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\sigma_{yy^d}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x})\mu_{y^d}(\mathbf{x})] - E_{\mathbf{x}}[\mu_y(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})] \ , \tag{6.24}$$

we can finally write

$$v(\mathbf{u},\mathbf{\Sigma}_x) = v_y(\mathbf{u},\mathbf{\Sigma}_x) + \sum_{d,e=1}^{D} v_{y^d y^e}(\mathbf{u},\mathbf{\Sigma}_x) + 2\sum_{d=1}^{D} v_{yy^d}(\mathbf{u},\mathbf{\Sigma}_x) \ . \tag{6.25}$$

Due to the mixed nature of the training set, the new predictive variance is not only the sum of the new variances but also accounts for cross-terms.

Case of the Gaussian kernel

As in Chapter 3, Section 3.4.2, we can compute the *noisy* predictive mean and variance exactly. We find (see Appendix C, Section C.1 for the detailed calculations)

$$m(\mathbf{u},\mathbf{\Sigma}_x) = \sum_i \beta_i C_G(\mathbf{u},\mathbf{x}_i) C_{corr}(\mathbf{u},\mathbf{x}_i)\left(1 + \sum_{d=1}^{D} w_d(c_i^d - x_i^d)\right) \tag{6.26}$$

where c_i^d is the d^{th} component of $(\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x^{-1}\mathbf{u})$, and

$$\begin{aligned} v(\mathbf{u}, \mathbf{\Sigma}_x) = C_G(\mathbf{u}, \mathbf{u}) + \sum_{i,j}(\beta_i\beta_j - K_{ij}^{-1}) \Bigg[1 + \sum_{d,e=1}^{D} w_d w_e (C_{de} + c_{ij}^d c_{ij}^e - x_j^e c_{ij}^d - x_i^d c_{ij}^e + x_i^d x_j^e) \\ +2\sum_{d=1}^{D} w_d(c_{ij}^d - x_j^d) \Bigg] C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) - m_y(\mathbf{u}, \mathbf{\Sigma}_x)^2 \\ - \sum_{d,e=1}^{D} m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x) m_{y^e}(\mathbf{u}, \mathbf{\Sigma}_x) - 2 m_y(\mathbf{u}, \mathbf{\Sigma}_x) \sum_{d=1}^{D} m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x) \end{aligned} \tag{6.27}$$

where C_{de} is the (d, e) entry of $\mathbf{C} = \left(\left(\frac{\mathbf{W}}{2}\right)^{-1} + \mathbf{\Sigma}_x^{-1}\right)^{-1}$ and c_{ij}^d the d^{th} element of $\mathbf{C}\left(\left(\frac{\mathbf{W}}{2}\right)^{-1} \frac{\mathbf{x}_i + \mathbf{x}_j}{2} + \mathbf{\Sigma}_x^{-1}\mathbf{u}\right)$.

If we now wish to use this model for the modelling of a nonlinear dynamic system and perform an iterative k-step ahead prediction with propagation of the uncertainty, we can simply apply the algorithm presented in Chapter 4, Section 4.2.1. Only in this case is the input covariance matrix slightly changed as we need to account for the derivative observations when computing the cross-covariance terms. The new expressions of these terms can be found in Appendix C, Subsection C.1.3. We refer to (Kocijan et al., 2004b) for numerical results.

6.2 GPs *in control*

We now turn to the use of Gaussian Processes for the control of nonlinear dynamic systems. In the control community, nonlinear control has mostly been based on nonlinear parametric models (e.g. neural networks), and it is only recently that GPs have appeared on the control scene (Murray-Smith et al., 1999; Leith et al., 2000).

Nonlinear Model Predictive Control (NMPC) is a methodology that refers to a class of control algorithms in which a nonlinear dynamic model of the plant is used to predict and optimise the future behaviour of the process (Henson, 1998; Qin and Badgwell, 2000). It can be formulated as follows (Henson, 1998): A sequence of control moves is computed to minimise an objective function (also referred to as loss or cost function) which includes predicted future values of the controlled outputs, obtained from the nonlinear model. Then, only the control input u computed for the present time-step is implemented, and the prediction horizon is moved one step forward, and the procedure repeated again, using new process measurements. The ability of these controllers to deal with constraints and multi-variable systems has made them very popular in industry. However, the approach relies heavily on the quality of the nonlinear model of the process. It is common that most of the available data lie around equilibrium points, and only sparse data are available in transient regions, thus jeopardising the identification and performance of a parametric model. In this respect, Gaussian Processes appear to be better suited for the task, as the data are used directly in prediction. This way, the uncertainty of the model's predictions can be made dependent on local data density, and the model complexity directly relates to the amount of available data. GPs have recently been successfully tested for NMPC (Kocijan et al., 2004c), with constraints on the variance, using the propagation of uncertainty algorithm proposed in Chapter 4 to compute the future outputs.

When deriving the control moves, most researchers have considered cost functions where the model's predictions are used as if they were the true system's outputs (Åström, 1995). These adaptive controllers, based on the *certainty equivalence principle*, therefore do not account for the uncertainty on the model's predictions, and the control actions taken do not influence the uncertainty either. Controllers achieving *caution*, by accounting for the model's uncertainty, and *probing* (exploration), by going into yet unexplored regions, have been the scope of much interest in the control community (Wittenmark, 1995). Assuming a quadratic cost function, this kind of *dual* controller aims at finding a control input u_t which minimises a K-stage criterion, as

$$J = E\left[\sum_{k=1}^{K}(y^d_{t+k} - y_{t+k})^2\right] \, , \tag{6.28}$$

where y^d is the reference signal and y is the controlled output. The numerical cost associated with this optimisation has usually led to sub-optimal or ad-hoc solutions to regularise the control behaviour when following the reference signal, depending on the model accuracy (Fabri and Kadirkamanathan, 1998). In the following, we present a controller with a 'cautious' feature as it is used in the multi-step-ahead optimisation of the control moves, with propagation of the uncertainty.

6.2.1 One-step-ahead cautious control

Consider a nonlinear a multi-input single-output (MISO) system,

$$y_{t+1} = f(y_t, \dots, y_{t-n}, u_t, u_{t-1}, \dots, u_{t-m}) + e_{t+1} \ , \tag{6.29}$$

where y_{t+1} is the controlled output, u_t is the current 'manipulated' input, and e_{t+1} is assumed to be a white noise with unknown variance v_t. We denote by $[\mathbf{x}_t, u_t]$ the state at $t+1$, where $\mathbf{x}_t$ is composed of the delayed outputs and inputs. We wish to control this system using the following cost function, which includes a penalty term on the control effort:

$$J_{t+1} = E[(y^d_{t+1} - y_{t+1})^2] + \lambda u^2_{t+1} \ . \tag{6.30}$$

Expanding the term $E[(y^d_{t+1} - y_{t+1})^2]$, we have

$$\begin{aligned} E[(y^d_{t+1} - y_{t+1})^2] &= E[{y^d_{t+1}}^2 - 2y^d_{t+1}y_k + y^2_{t+1}] \\ &= {y^d_{t+1}}^2 - 2y^d_{t+1}E[y_{t+1}] + E[y^2_{t+1}] \ . \end{aligned}$$

Using $E[y^2_{t+1}] = \mathrm{Var}[y_{t+1}] + E[y_{t+1}]^2$, the cost at time $t+1$ can be written

$$J_{t+1} = (y^d_{t+1} - E[y_{t+1}])^2 + \mathrm{Var}[y_{t+1}] + \lambda u^2_{t+1} \ . \tag{6.31}$$

This cost function accounts naturally for the uncertainty associated with $E[y_{t+1}]$, which arises by doing simple manipulations, as noted by (Murray-Smith and Sbarbaro, 2002). That most researchers have ignored the variance term is likely to be related to the model used in the first place, as it might be difficult to obtain this uncertainty, and to compute the derivatives of the cost function with respect to it. Indeed, the optimal control input is now found by solving $\frac{\partial J_{t+1}}{\partial u_t} = 0$, that is

$$\frac{\partial J_{t+1}}{\partial u_t} = -2\left(y^d_{t+1} - E[y_{t+1}]\right)\frac{\partial E[y_{t+1}]}{\partial u_t} + \frac{\partial \mathrm{Var}[y_{t+1}]}{\partial u_t} + 2\lambda u_t = 0 \ . \tag{6.32}$$

For a GP with zero-mean and covariance function $C(.,.)$, $E[y_{t+1}]$ is simply the predictive mean at $t+1$,

$$\mu([\mathbf{x}_t, u_t]) = \sum_i \beta_i C([\mathbf{x}_t, u_t], [\mathbf{x}_i, u_i]) \ , \tag{6.33}$$

and $\mathrm{Var}[y_{t+1}]$ its associated variance,

$$\sigma^2([\mathbf{x}_t, u_t]) = C([\mathbf{x}_t, u_t], [\mathbf{x}_t, u_t]) - \sum_{i,j} K^{-1}_{ij} C([\mathbf{x}_t, u_t], [\mathbf{x}_i, u_i]) C([\mathbf{x}_t, u_t], [\mathbf{x}_j, u_j]) \ , \tag{6.34}$$

where $\boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{t}$ and $\mathbf{K}$ is the data covariance matrix computed using the pairs $\mathbf{x}_i, u_i$ used for training. In this case, it is straightforward that

$$\begin{aligned}
\frac{\partial \mu([\mathbf{x}_t, u_t])}{\partial u_t} &= \sum_i \beta_i \frac{\partial C([\mathbf{x}_t, u_t], [\mathbf{x}_i, u_i])}{\partial u_t} \\
\frac{\partial \sigma^2([\mathbf{x}_t, u_t])}{\partial u_t} &= \frac{\partial C([\mathbf{x}_t, u_t], [\mathbf{x}_t, u_t])}{\partial u_t} - \sum_{i,j} K_{ij}^{-1} \left(\frac{\partial C([\mathbf{x}_t, u_t], [\mathbf{x}_i, u_i])}{\partial u_t} C([\mathbf{x}_t, u_t], [\mathbf{x}_j, u_j]) \right. \\
&\left. + C([\mathbf{x}_t, u_t], [\mathbf{x}_i, u_i]) \frac{\partial C([\mathbf{x}_t, u_t], [\mathbf{x}_j, u_j])}{\partial u_t} \right) .
\end{aligned}$$

We refer to (Murray-Smith and Sbarbaro, 2002; Sbarbaro and Murray-Smith, 2003) for various examples, illustrating how a controller which does not disregard the variance information leads to a robust control action during adaptation.

6.2.2 Exploration and multi-step-ahead control

The total cost of controlling a system from $t+1$ to, say, time $t+K$, where K is our control horizon, is given by

$$J = \frac{1}{K} \sum_{k=1}^{K} J_k ,$$

where, for simplicity in the notation, we now denote by J_k the cost at time $t+k$ given by

$$J_k = (y_k^d - E[y_k])^2 + \mathrm{Var}[y_k] + \lambda u_{k-1}^2 . \tag{6.35}$$

If we want to propagate the uncertainty as we are making predictions ahead in time, and assuming the covariance function of the process is the Gaussian one, $E[y_k]$ and $\mathrm{Var}[y_k]$ are given by (3.39) and (3.43) of Chapter 3 (only for $k=1$ do we use (6.33) and (6.34)). Let denote by $m([\mathbf{u}_{k-1}, \boldsymbol{\Sigma}_{k-1}, u_{k-1}])$ the predictive mean of y_k, and its variance by $v([\mathbf{u}_{k-1}, \boldsymbol{\Sigma}_{k-1}, u_{k-1}])$, or, for short, m_k and v_k, corresponding to

$$\mathbf{x}_{k-1} = [y_{k-1}, \dots, y_{k-1-n}, u_{k-2}, \dots, u_{k-2-m}]^T \sim \mathcal{N}(\mathbf{u}_{k-1}, \boldsymbol{\Sigma}_{k-1}) ,$$

with

$$\mathbf{u}_{k-1} = [m_{k-1}, \dots, m_{k-1-n}, u_{k-2}, \dots, u_{k-2-m}]^T ,$$

where $m_{k-i} = E[y_{k-i+1}]$, and

$$\mathbf{\Sigma}_{k-1} = \begin{bmatrix} v_{k-1} & \dots & \text{Cov}[y_{k-1}, y_{k-1-n}] & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & 0 & \dots & 0 \\ v_{k-1-n} & \dots & \text{Cov}[y_{k-1-n}, y_{k-1}] & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where $v_{k-i} = \text{Var}[y_{k-i+1}]$.

Now, the multi-step-ahead control of the system consists of finding the control signal such that the total cost is minimum over the whole horizon. That is, we wish to find $u_0, \dots, u_{K-1}$ such that J is minimum, which is achieved by solving

$$\frac{\partial J}{\partial(u_0, \dots, u_{K-1})} = \frac{\partial}{\partial(u_0, \dots, u_{K-1})} \sum_{k=1}^{K} J_k = 0 \, . \tag{6.36}$$

A 'local approximation' then consists of computing the optimal control input for each time-step, i.e. to find u_{i-1} such that J_i is minimum. However, it is likely that the resulting control signal $u_0^{opt}, \dots, u_{K-1}^{opt}$, composed of all the local optimal contributions, is far from the overall optimal one $(u_0, \dots, u_{K-1})^{opt}$. Following (Murray-Smith et al., 2003), the minimisation of the total cost requires the computation of the *sensitivity equations*, which tell us how a change in the control signal at time i affects the total cost. We need to compute

$$\frac{\partial J}{\partial u_i} \propto \sum_{k=i+1}^{K} \frac{\partial J_k}{\partial u_i} \, , \tag{6.37}$$

since the cost at time $k \leq i$ is independent of u_i (J_k only depends on u_{k-1}).

Sensitivity equations

We therefore need to solve

$$\frac{\partial J_k}{\partial u_i} = -2(y_k^d - m_k)\frac{\partial m_k}{\partial u_i} + \frac{\partial v_k}{\partial u_i} \, , \tag{6.38}$$

where $m_k = m([\mathbf{u}_{k-1}, \mathbf{\Sigma}_{k-1}, u_{k-1}])$ and and $v_k = v([\mathbf{u}_{k-1}, \mathbf{\Sigma}_{k-1}, u_{k-1}])$. In the case of the Gaussian covariance function, we can write

$$m_k = \sum_i \beta_i C([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_i) C_{corr}([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_i) \; , \tag{6.39}$$

where $\mathbf{s}_i = [\mathbf{x}_i, u_i]$ denotes the training states and control inputs, and $C_{corr}(.,.)$ is given by (3.40), and

$$\begin{aligned} v_k = C([\mathbf{u}_{k-1}, u_{k-1}], [\mathbf{u}_{k-1}, u_{k-1}]) - \sum_{i,j=1}^{N} (K_{ij}^{-1} - \beta_i\beta_j) C([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_i) C([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_j) \\ C_{corr_2}([\mathbf{u}_{k-1}, u_{k-1}]\bar{\mathbf{s}}) - \sum_{i,j=1}^{N} \beta_i\beta_j C([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_i) C([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_j) \\ C_{corr}([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_i) C_{corr}([\mathbf{u}_{k-1}, u_{k-1}], \mathbf{s}_j) \; , \end{aligned} \tag{6.40}$$

where $\bar{\mathbf{s}} = \frac{\mathbf{s}_i + \mathbf{s}_j}{2}$ and $C_{corr_2}(.,.)$ is given by (3.41).

For $k = i + 1$, the differentiation with respect to u_i can then be done term-wise, in a rather straightforward manner. On the other hand, for $k > i + 1$, we need to account for the interactions between the delayed outputs. The derivatives of m_k and v_k with respect to u_i are then obtained using the chain rule:

$$\frac{\partial m_k}{\partial u_i} = \frac{\partial m_k}{\partial \mathbf{u}_{k-1}} \frac{\partial \mathbf{u}_{k-1}}{\partial u_i} + \frac{\partial m_k}{\partial \mathbf{\Sigma}_{k-1}} \frac{\partial \mathbf{\Sigma}_{k-1}}{\partial u_i} \tag{6.41}$$

$$\frac{\partial v_k}{\partial u_i} = \frac{\partial v_k}{\partial \mathbf{u}_{k-1}} \frac{\partial \mathbf{u}_{k-1}}{\partial u_i} + \frac{\partial v_k}{\partial \mathbf{\Sigma}_{k-1}} \frac{\partial \mathbf{\Sigma}_{k-1}}{\partial u_i} \; . \tag{6.42}$$

Given the derivative $\partial J / \partial u_t$, we can then find the optimal control sequence. We refer to (Murray-Smith et al., 2003) for preliminary results. In Appendix C, Section C.2, we simply provide the expressions for the predictive mean and variance corresponding to a MISO system affine in u. Note that these equations can be used directly in the case of a *mixed* training set, formed of derivative as well as function observations, provided the above partial derivatives are computed numerically (we leave it to the reader to compute the analytical derivatives).

6.2.3 A probabilistic view?

So far, the cost of controlling a system has been expressed as the expected squared difference $E[(y_k^d - y_k)^2]$ (ignoring the control term). It would seem a natural thing to compute the variance, therefore viewing J_k as a random quantity. We can then define $J_k = (y_k^d - y_k)^2$, with expectation $E[J_k] = E[(y_k^d - y_k)^2]$ and variance $\mathrm{Var}[J_k] = \mathrm{Var}[(y_k^d - y_k)^2]$. As we have already shown, we have

$$E[J_k] = E[(y_k^d - y_k)^2] = (y_k^d - E[y_k])^2 + \mathrm{Var}[y_k] \ ,$$

or simply

$$E[J_k] = (y_k^d - m_k)^2 + v_k,$$

where $m_k = E[y_k]$ and $v_k = \mathrm{Var}[y_k]$. Expanding the square difference and taking the variance of the corresponding products, the variance of J_k is given by

$$\mathrm{Var}[J_k] = \mathrm{Var}[y_k^{d^2} - 2y_k^d y_k + y_k^2] = 4y_k^{d^2}\mathrm{Var}[y_k] + \mathrm{Var}[y_k^2] \ .$$

We now need to express $\mathrm{Var}[y_k^2]$. To do so, let us consider the standard normal variable $z_k = \frac{y_k - m_k}{\sqrt{v}_k}$, so that $z_k \sim \mathcal{N}(0,1)$. Therefore, z_k^2 is chi-squared distributed, with mean 1 and variance 2. We then have $y_k = \sqrt{v}_k z_k + m_k$, and

$$y_k^2 = (\sqrt{v}_k z_k + m_k)^2 = v_k z_k^2 + 2\sqrt{v}_k z_k m_k + m_k^2 \ ,$$

leading to

$$\mathrm{Var}[y_k^2] = \mathrm{Var}[v_k z_k^2 + 2\sqrt{v}_k z_k m_k + m_k^2] = v_k^2 \mathrm{Var}[z_k^2] + 4v_k m_k^2 \mathrm{Var}[z_k] = 2v_k^2 + 4v_k m_k^2 \ ,$$

obtained by taking the variance on both side and using $\mathrm{Var}[z_k^2] = 2$ and $\mathrm{Var}[z_k] = 1$.

We can then consider $J_k = (y_k^d - y_k)^2$ with

$$E[J_k] = (y_k^d - m_k)^2 + v_k \tag{6.43}$$

$$\mathrm{Var}[J_k] = 4y_k^{d^2} v_k + 2v_k^2 + 4v_k m_k^2 \ . \tag{6.44}$$

Therefore, rather than performing the minimisation of $E[J_k]$ only, one could constrain it on the associated variance $\mathrm{Var}[J_k]$, or even blend the knowledge of both $E[J_k]$ and $\mathrm{Var}[J_k]$ together in some ‘meta-cost’ function, thereby viewing J_k as a ‘latent variable’. This ‘hierarchical’ view could enable more rational decisions when selecting the final control policies. A somehow similar idea has been recently suggested in a reinforcement learning control context, where a Gaussian Process represents the value function (Rasmussen and Kuss, 2004), although only the expectation is used in the paper.

We have not explored the potential interest of this approach and an investigation would be needed to appreciate the validity and usefulness of such an approach.

6.3 Summary

In this last chapter, we have presented further extensions of the propagation of uncertainty framework. We have first shown how the predictive distribution of a function output corresponding to a new noisy input could be derived, when the training set was formed of function as well as derivative observations of a system. Again, these results allow to perform the iterative multiple-step-ahead prediction of a system of which derivative observations are available, which is often the case in engineering applications (linearisations of measured data around equilibrium points). Also, a methodology to account for the uncertainty in a control context has been presented. Using a cost function that does not disregard the output uncertainty (cautious controller), we have shown how we could again perform the multi-step-ahead control of a system while propagating the uncertainty ahead in time.

This chapter only deals with the methodological aspect and we refer to (Kocijan et al., 2004b) for preliminary numerical results concerning the use of derivative observations, and to (Murray-Smith et al., 2003) for the cautious multi-step-ahead control of MISO systems.

Chapter 7

Conclusions

In this thesis, we have explored a number of extensions that could be made to the Gaussian Process model, most of them related to the informative use of the model's uncertainty, in order to improve the modelling, forecasting and control of nonlinear dynamic systems. We have suggested an approach to account for the noise on the inputs, when making predictions. From this, we have derived an algorithm to propagate the uncertainty ahead in time, for the iterative multiple-step-ahead prediction of dynamic systems. We have also contributed to the development of tools that will hopefully make the GP an attractive model in the eyes of the engineering community, as the model can now make effective use of derivative observations, when these are available, and of its predictive uncertainty, in a cost function, thus enabling the cautious control of systems.

As we have seen, although a Gaussian Process is a set of infinitely many random variables, it is in practice reduced to the size of the sets we are interested in, resulting in a process finally modelled by a joint multi-dimensional Gaussian distribution. Even though we have not taken a Bayesian approach, whereby priors are put on the parameters of the covariance function of the process and then combined to the data to compute posterior distributions, the very probabilistic nature of the model is enough to lead to a distribution as the solution to the prediction task. As a result of the neat properties of the Gaussian assumption, this predictive distribution is also Gaussian, fully specified by its mean and variance which can then be used respectively as an estimate and the associated uncertainty of the model's output.

With this model, we have then addressed the problem of making a prediction at a normally distributed random input. We solve it within an analytical approximation, whereby we only compute the mean and variance of the corresponding predictive distribution (*Gaussian approximation*). Based on the parametric form of the covariance function, we derive exact moments, for the linear and squared exponential (Gaussian) covariance functions, or approximate ones, relying on a second-order Taylor approximation. Also, using a similar *Gaussian approximation*, we show how the challenging problem of learning with noisy inputs can be tackled, resulting in a covariance function where the length-scales are weighted down by the input uncertainty (in the case of the Gaussian covariance function). Although the emphasis is on the application of these results to the k-step-ahead prediction of nonlinear dynamic systems, predicting at a random input is of interest in the static case. If the system senses the inputs imperfectly, due to faulty sensors or some other external disturbance, we can now account for this extra uncertainty, although prior knowledge of the input noise variance is assumed in this case. In the dynamic case, we show how the uncertainty induced by each successive prediction can be propagated ahead in time, when making iterative multiple-step-ahead predictions of a nonlinear dynamic system represented by a one-step-ahead nonlinear auto-regressive model. The derived algorithm formally accounts for the predictive variance of each delayed output, as well as for the cross-covariances among them, thereby providing the model with full knowledge of the characteristics of the random state (mean and covariance matrix), as it progresses ahead in time. At each time-step, the predictive mean and variance and the cross-covariance elements can be computed exactly, in the case of the Gaussian covariance function, or within the Taylor approximation.

Our experimental results on the simulated Mackey-Glass chaotic time-series suggest that our *Gaussian approximation*, when compared to the numerical approximation of the true distribution by simple Monte-Carlo, is well grounded. We are aware that more work should be done towards the Monte-Carlo approximation and its evaluation (the measures of predictive performance that we used, i.e. squared error loss and negative log-predictive density, not being ideal for non-Gaussian distributions). These experiments also validate the results obtained within the Taylor approximation of the *Gaussian approximation*, as we compare them to the exact ones, using the Gaussian kernel. We know that the integrals required for the computation of the predictive mean and variance are solvable exactly in the case of linear, Gaussian and polynomial (or a mixture of those) covariance functions. However, we have not defined the family of valid covariance functions for which the Taylor approximation was needed. Let us simply recall that, when using this approximation, one should not forget to check the continuity properties of the function and positive-definiteness of the resulting approximation, as well as the other requirements inherent in the Taylor approximation. Our main focus is nonetheless on the Gaussian covariance function, which has been predominantly used when modelling with Gaussian Processes, and which is shared by many other kernel models.

In general, we highlight the well-founded nature of the propagation of uncertainty approach by comparing it to a naive method that only feeds back estimates, which are thereby viewed by the model as if they were the true outputs. This naive approach can be misleading, as the error-bars on its predictions stay small, no matter how far ahead we predict in time and how good its estimates are. On the other hand, propagating the uncertainty leads to more reliable predictions, suggesting that the full matrix of covariances, accounting for all the uncertainties between the delayed outputs, has a real impact on future predictions. Note that, as a first approximation, one could simply consider a diagonal input covariance matrix, with only the delayed predictive variances on the diagonal, although we do not see why one should do so, as the computational cost of computing the cross-covariance terms is low.

After investigation whether the approach would generalise to real systems, the results prove encouraging. We first show how the GP can effectively model the gas pressure in a gas-liquid separator process. Based on the Automatic Relevance Determination tool (Neal, 1995; MacKay, 1994) and on Occam's razor principle (Rasmussen and Ghahramani, 2001), we identify the model with the best complexity/performance trade-off. Although the signals used for identification are significantly different from those used for the validation of the model (rapid/slowly varying signals), the model successfully captures the dynamics of the system, as can be seen from its performance on the prediction of the test signal over its whole length. For this process, further experiments could consider the incorporation of a less crude model for the noise than the white one (i.e. a coloured noise model, as suggested in (Murray-Smith and Girard, 2001), or a full nonlinear model, such as in (Goldberg et al., 1998)). Also, for the Gaussian covariance function, including a full covariance matrix $\mathbf{W}$, allowing for correlations between input dimensions (as done in (Vivarelli and Williams, 1999)), could improve the modelling of the interactions between the variables.

The identification of a model for the pH of a chemical process showed to be improved when a linear model was first fitted onto the original data and a GP used to model the residuals. We then showed how we could elegantly incorporate this linear model into our framework to perform multiple-step-ahead predictions, therefore accounting for the linear part and its interaction with the GP model at each time-step. When comparing the propagation of uncertainty to the naive approach, we again notice how the latter leads to unreliable predictive distributions, peaked around the mean even though it is not close (with respect to some given metric) to the true response (such that the variance is no longer an indicator of the reliability of the model). As for the Mackey-Glass system, we observe that the predictive variances given by the exact moments, within the *Gaussian approximation*, do not simply increase with the predictive horizon k. Whether such a desired behaviour happens or not is likely to depend on the properties of the system being modelled. On this system, the comparison of the iterative approach to the direct method, where a model is trained to predict k steps ahead, leads us to think that the iterative scheme can indeed be valuable. Although for this particular example the direct model becomes better than the iterative one, as the predictive horizon increases, favouring a direct model is arguable as one model is only valid for a given horizon, which can prove difficult or even infeasible in practice (the training of the model can be time-consuming and the predictive horizon is often not known in advance).

It is well known that the modelling of nonlinear dynamic systems can be very difficult, for reasons ranging from the properties of the underlying of the system to our computational resources. A frequent problem encountered in practice is the uneven spread of the data in the operating regime of the system. Often, the available data is confined around the equilibrium points of the system, and few or no data can be measured in transient regions. A common approach has been to build local linear models and blend them to obtain a nonlinear model, covering the whole operating range (Murray-Smith et al., 1999; Murray-Smith and Johansen, 1997). Nevertheless, this approach proves difficult when it comes to choosing the number of local models, the type of learning (global/local), etc. In that respect, GPs are likely to improve the modelling of such systems, thanks to their flexibility and the way inference is performed, by conditioning on the current state and local data. So far, their major burden has been the computational cost associated with them (the inversion of the $N \times N$ data covariance matrix). Although many techniques are now available to reduce this cost (Williams and Seeger, 2001; Seeger et al., 2003; Murray-Smith and Pearlmutter, 2003; Shi et al., 2002), the methodology we have presented (accounting for derivative observations) is original and possibly more appealing as it enables us to not only potentially reduce the computational cost, but also to be in accord with engineering practice, by summarising measured data in the vicinity of equilibrium points by a derivative observation. In conjunction with the results presented on the application of GPs in a control context, and the promising results of our experiments, we hope that the model will be more widely used in the engineering community. Also, we believe that our results are general enough to be applicable to other kernel models, as they do to Relevance Vector Machines (Quinonero-Candela et al., 2003; Quinonero-Candela and Girard, 2002).

Ultimately, we want to invent and create mathematical tools and concepts to explain the world around us and mimic the way we understand our brain processes information and makes decisions. In that respect, we would be inclined towards Bayesian approaches, as we believe they reflect our own way of thinking. Paraphrasing E.T. Jaynes: *'When confronted with a new situation, our brain is forming plausible conclusions, based on prior information and evidence about the situation we are reasoning on, in order to 'compute' an updated state of information, also called posterior, that reflects our new degree of belief'*. Even though we have not considered the Gaussian Process model in a Bayesian setup, the simple form used in this thesis has proved useful for the modelling of nonlinear systems and has enabled the rather easy and elegant derivation of the 'tools' and extensions presented here. Also, our preference for analytical approximations over numerical ones has been motivated by the fact they can sometimes be less computationally demanding than numerical methods (although, in our case, the numerical approximation is straightforward). More importantly, the analytical *Gaussian approximation* is easier to interpret than the numerical solution. As we have seen, the Monte-Carlo approximation leads to non-Gaussian distributions, which are more difficult to summarise; an aspect which is likely to slow its uptake in engineering contexts.

We hope the results presented in this thesis will encourage and motivate further experiments and research as we believe the potential of the GP model has not yet been fully explored. Future work could include the incorporation of time in the model (either by allowing the parameters of the covariance function to be time-dependent or the covariance function to be itself a function of time), and the extension to multiple outputs, thus enabling the modelling of systems with time-varying dynamics and correlated measured outputs.

Appendix A

Mathematical formulae

In the following, we recall some useful formulae and results used throughout this thesis.

Law of iterated expectations and conditional variances

Let X be a random variable (RV). For y given, $p(X|Y = y)$ has mean $E[X|Y = y]$ and variance $\mathrm{Var}[X|Y = y]$. Now if Y is a RV, both the expectation and variance are a function of Y, hence RVs themselves. It can be shown that

$$\begin{aligned} E[X] &= E[E[X|Y]] \\ \mathrm{Var}[X] &= E[\mathrm{Var}[X|Y]] + \mathrm{Var}[E[X|Y]] \ . \end{aligned}$$

Taylor series and approximation

Taylor's formula

If $f(x)$ has derivatives up to order n at the point $x = x_0$, then the polynomial

$$f_n(x) = f(x_0) + (x - x_0)f'(x_0) + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \ ,$$

where $f^{(n)}(x_0)$ denotes the n^{th} derivative of $f(x)$ evaluated at $x = x_0$, is called the n^{th}-order Taylor approximation to f at x_0.

Taylor's formula is

$$f(x) = f_n(x) + R_n(x)$$

where $R_n(x)$ is called the remainder term.

Let $f^{(n+1)}(x)$ exist and be continuous in an open interval I and let $x_0 \in I$. Then, given $x \in I$, there is a point c between x_0 and x such that

$$R_n(x) = \frac{(x-x_0)^{n+1}}{(n+1)!} f^{(n+1)}(c) \; .$$

This expression is called the Lagrange form of the remainder.

If the $(n+1)^{st}$ derivative of f satisfies

$$m \leq f^{(n+1)}(x) \leq M$$

for all x in some interval about x_0, then, for all x in this interval, we have

$$m\frac{(x-x_0)^{n+1}}{(n+1)!} \leq R_n(x) \leq M\frac{(x-x_0)^{n+1}}{(n+1)!}$$

if $(x-x_0)^{n+1} > 0$ (otherwise the reverse inequalities hold).[1]

Product of Taylor series

Let $f(x) = \sum_{n=0}^{\infty} f_n(x-a)^n$, where $f_n = \frac{f^{(n)}(a)}{n!}$ and $g(x) = \sum_{n=0}^{\infty} g_n(x-a)^n$, where $g_n = \frac{g^{(n)}(a)}{n!}$. Then the Taylor series for $f(x)g(x)$ is the product of the Taylor series for $f(x)$ with that of $g(x)$:[2]

$$f(x)g(x) = \sum_{n=0}^{\infty} h_n(x-a)^n \quad \text{where} \quad h_n = \sum_{k=0}^{n} f_k g_{n-k} \; .$$

For two second-order truncated series $f(x) = f_0 + f_1(x-a) + f_2(x-a)^2$ and $g(x) = g_0 + g_1(x-a) + g_2(x-a)^2$, we then have

$$\begin{aligned} f(x)g(x) &= h_0 + h_1(x-a) + h_2(x-a)^2 \\ &= f_0g_0 + (f_0g_1 + f_1g_0)(x-a) + (f_0g_2 + f_1g_1 + f_2g_0)(x-a)^2 \; . \end{aligned}$$

[1]Although proof of these results could be found in any maths book, we recommend the Calculus with Theory I, Lecture o, of the MIT open course http://ocw.mit.edu/OcwWeb/Mathematics/index.htm.

[2]From http://www.math.pitt.edu/∽sparling/23014/23014convergence2/node8.html

Operations on random functions

The following results are taken from (Pugachev, 1967).

Let $X(t)$ be a random process of argument t, with mean function $m_x(t)$ and covariance function $C_x(t,t')$. Similarly, let $Y(t)$ be another process, with mean and covariance functions $m_y(t)$ and $C_y(t,t')$.

Addition of random processes

Let $Z(t) = X(t) + Y(t)$. It can be shown that the mean function $m_z(t)$ of $Z(t)$ is

$$m_z(t) = m_x(t) + m_y(t) \ .$$

If $X(t)$ and $Y(t)$ are correlated, the covariance function $C_z(t,t')$ of $Z(t)$ is

$$C_z(t,t') = C_x(t,t') + C_y(t,t') + C_{xy}(t,t') + C_{yx}(t,t')$$

where $C_{xy}(t,t')$ and $C_{yx}(t,t')$ are the cross-covariance functions between $X(t)$ and $Y(t)$.

In general, for an arbitrary number of random functions, if

$$Z(t) = \sum_{n=1}^{N} X_n(t) \ ,$$

then the mean and covariance functions of $Z(t)$ are

$$\begin{aligned} m_z(t) &= \sum_{n=1}^{N} m_n(t) \\ C_z(t,t') &= \sum_{n,n'=1}^{N} C_{nn'}(t,t') \ . \end{aligned}$$

That is to say, the expected value of $Z(t)$ is equal to the sum of the expected values of the random functions $X_n(t)$, and the covariance function of $Z(t)$ is the sum of the covariance functions of the different $X_n(t)$ plus the sum of the cross-covariance functions of the terms in the sum.

Differentiation of a random function

Let $Y_1(t) = X'(t)$, the first derivative of $X(t)$. $Y_1(t)$ has mean $m_{y_1}(t) = m_x(t)'$ (i.e. the derivative of $m_x(t)$) and covariance function $C_{y_1}(t,t') = \frac{\partial^2 C_x(t,t')}{\partial t \partial t'}$ (i.e. the second mixed partial derivative of $C_x(t,t')$).

In general, the mean $m_{y_p}(t)$ and covariance function $C_{y_p}(t,t')$ of the derivative of order p, $Y_p(t) = X^{(p)}(t)$, for the random function $X(t)$, are

$$\begin{aligned} m_{y_p}(t) &= m_x(t)^{(p)} \\ C_{y_p}(t,t') &= \frac{\partial^{2p} C_x(t,t')}{\partial t^p \partial t'^p} \ . \end{aligned}$$

The cross-covariance functions for the derivatives, of arbitrary order, of $X(t)$ are given by

$$C_{y_p y_q}(t,t') = \frac{\partial^{p+q} C_x(t,t')}{\partial t^p \partial t'^q} \ .$$

Matrix identities

Given the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$:

Basic formulae

$$\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB} + \mathbf{AC} \quad (\mathbf{A}+\mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T \quad (\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \quad (\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1}$$

$$|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}| \quad |\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|}$$

provided the inverses exist ($|.|$ denotes the determinant).

Matrix inversion lemma

$$(\mathbf{A} + \mathbf{XBX}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{X}(\mathbf{B}^{-1} + \mathbf{X}^T\mathbf{A}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}^{-1}$$

where $\mathbf{A}$ and $\mathbf{B}$ are square and invertible matrices, not necessarily of the same dimension.

Gaussian identities

Let $\mathcal{N}_{\mathbf{x}}(\mathbf{a}, \mathbf{A})$ be a D-dimensional Gaussian density for $\mathbf{x}$:

$$\mathcal{N}_{\mathbf{x}}(\mathbf{a}, \mathbf{A}) = (2\pi)^{-D/2}|\mathbf{A}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{a})^T\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a})\right] .$$

Marginal and conditional distributions

Let $\mathbf{z} \sim \mathcal{N}(\mathbf{d}, \mathbf{D})$. If we split $\mathbf{z}$ into two vectors $\mathbf{x}$ (of size N_1) and $\mathbf{y}$ (of size N_2), we can write the joint normal distribution as

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix}\right)$$

where $\mathbf{C}$ is the $N_1 \times N_2$ matrix of cross-covariances between $\mathbf{x}$ and $\mathbf{y}$.

The marginal distributions are then

$$\mathbf{x} \sim \mathcal{N}(\mathbf{a}, \mathbf{A}) \quad \text{and} \quad \mathbf{y} \sim \mathcal{N}(\mathbf{b}, \mathbf{B}) \; ,$$

and the conditional distributions

$$\begin{aligned} \mathbf{x}|\mathbf{y} &\sim \mathcal{N}(\mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y}-\mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T) \\ \mathbf{y}|\mathbf{x} &\sim \mathcal{N}(\mathbf{b} + \mathbf{C}^T\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}), \mathbf{B} - \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C}) \; . \end{aligned}$$

Product of Gaussians

$$\mathcal{N}_{\mathbf{x}}(\mathbf{a}, \mathbf{A})\mathcal{N}_{\mathbf{x}}(\mathbf{b}, \mathbf{B}) = \mathbf{z}\mathcal{N}_{\mathbf{x}}(\mathbf{c}, \mathbf{C})$$

where

$$\mathbf{c} = \mathbf{C}(\mathbf{A}^{-1}\mathbf{a} + \mathbf{B}^{-1}\mathbf{b}), \quad \mathbf{C} = (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1}$$

and with the constant $\mathbf{z}$ usually found expressed as

$$\mathbf{z} = (2\pi)^{-D/2}|\mathbf{C}|^{1/2}|\mathbf{A}|^{-1/2}|\mathbf{B}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{a}^T\mathbf{A}^{-1}\mathbf{a} + \mathbf{b}^T\mathbf{B}^{-1}\mathbf{b} - \mathbf{c}^T\mathbf{C}^{-1}\mathbf{c})\right] .$$

Using the above matrix identities, we can verify that this simplifies into

$$\mathbf{z} = (2\pi)^{-D/2}|\mathbf{A}+\mathbf{B}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{a}-\mathbf{b})^T(\mathbf{A}+\mathbf{B})^{-1}(\mathbf{a}-\mathbf{b})\right] .$$

That is $\mathbf{z} = \mathcal{N}_{\mathbf{a}}(\mathbf{b}, \mathbf{A}+\mathbf{B})$ or $\mathbf{z} = \mathcal{N}_{\mathbf{b}}(\mathbf{a}, \mathbf{A}+\mathbf{B})$.

Expectation of a quadratic form under a Gaussian

Let $\mathbf{x} \sim \mathcal{N}(\mathbf{a}, \mathbf{A})$. Then

$$\int_{\mathbf{x}} (\mathbf{x}-\mathbf{b})^T \mathbf{B}^{-1} (\mathbf{x}-\mathbf{b}) \mathcal{N}_{\mathbf{x}}(\mathbf{a}, \mathbf{A}) d\mathbf{x} = (\mathbf{b}-\mathbf{a})^T \mathbf{B}^{-1} (\mathbf{b}-\mathbf{a}) + \mathrm{Tr}[\mathbf{B}^{-1}\mathbf{A}] \, .$$

Gaussian covariance function and derivatives

Let $C_{ij} = C(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\left(-\frac{1}{2}\sum_{d=1}^{D} w^d (x_i^d - x_j^d)^2\right)$, where D is the input dimension.

First derivative

The first derivative $\mathbf{C}'_{ij} = \frac{\partial C_{ij}}{\partial \mathbf{x}_i}$ is a $D \times 1$ vector whose d^{th} component can be written

$$\frac{\partial C_{ij}}{\partial x_i^d} = -w^d (x_i^d - x_j^d) C_{ij} = -\frac{\partial C_{ij}}{\partial x_j^d} \, .$$

Second derivatives

The (d,e) entry of the second derivative with respect to $\mathbf{x}_i$, $\frac{\partial^2 C_{ij}}{\partial \mathbf{x}_i \partial \mathbf{x}_i{}^T}$, is given by

$$\frac{\partial^2 C_{ij}}{\partial x_i^d \partial x_i^e} = \frac{\partial}{\partial x_i^e}\left[-w^d (x_i^d - x_j^d) C_{ij}\right] = -w^d C_{ij} \delta_{de} - w^d (x_i^d - x_j^d) \frac{\partial C_{ij}}{\partial x_i^e}$$

that is

$$\frac{\partial^2 C_{ij}}{\partial x_i^d \partial x_i^e} = [-w^d \delta_{de} + w^d (x_i^d - x_j^d) w^e (x_i^e - x_j^e)] C_{ij} = \frac{\partial^2 C_{ij}}{\partial x_i^d \partial x_i^e}$$

for $d, e = 1 \ldots D$, and where $\delta_{de} = 1$ if $d = e$, $\delta_{de} = 0$ otherwise.

Also, the (d,e) entry of $\mathbf{C}''_{ij} = \frac{\partial^2 C_{ij}}{\partial \mathbf{x}_i \partial \mathbf{x}_j{}^T}$ is given by

$$\begin{aligned}
\frac{\partial^2 C_{ij}}{\partial x_i^d \partial x_j^e} &= \frac{\partial}{\partial x_j^e}\left[-w^d (x_i^d - x_j^d) C_{ij}\right] \\
&= w^d C_{ij} \delta_{de} - w^d (x_i^d - x_j^d) \frac{\partial C_{ij}}{\partial x_j^e} \\
&= [w^d \delta_{de} - w^d (x_i^d - x_j^d) w^e (x_i^e - x_j^e)] C_{ij}
\end{aligned}$$

for $d, e = 1 \ldots D$. For $d = e$ we simply have $\frac{\partial^2 C_{ij}}{\partial x_i^d \partial x_j^d} = [w^d - {w^d}^2 (x_i^d - x_j^d)^2] C_{ij}$.

Appendix B

Note on the Taylor approximation

In Chapter 3 we have shown how, when predicting at a noisy input, we could compute the mean and variance of the *noisy* non-Gaussian predictive distribution (*Gaussian approximation*) exactly, in the case of the Gaussian and the linear covariance function, or approximately, within a Taylor approximation of the covariance function. We are now going to look more closely at the Taylor approximation and the associated error.

For the sake of simplicity, the following discussion is done in the case of one-dimensional inputs. Let the new test input be $x \sim \mathcal{N}(u, v_x)$. As seen in Chapter 3, the computation of the mean $m(u, v_x)$ and the variance $v(u, v_x)$ of the corresponding output y requires the evaluation of $l = E_x[C(x,x)]$, $l_i = E_x[C(x,x_i)]$ and $l_{ij} = E_x[C(x,x_i)C(x,x_j)]$. For the time being, consider l_i, where

$$l_i = \int_{-\infty}^{+\infty} C(x,x_i)p(x)dx.$$

Provided $C(.,.)$ is such that the integral is not analytically tractable, we resort to a second-order Taylor approximation to $C(x,x_i)$. For x in some interval I around u, we can write (see Appendix A)

$$C(x,x_i) = C^{ap}(x,x_i) + R_2(x,x_i) \ ,$$

where $C^{ap}(x,x_i)$ is the second-order polynomial around the mean u of x,

$$C^{ap}(x,x_i) = C(u,x_i) + (x-u)C'(u,x_i) + \frac{1}{2}(x-u)^2C''(u,x_i)$$

and $R_2(x, x_i)$ is the remainder, written in its Lagrange form as

$$R_2(x, x_i) = \frac{1}{6}(x-u)^3 C^{(3)}(c, x_i) \ ,$$

for some c between x and u.

Let $I = [a, b]$. We then have

$$l_i = \int_{-\infty}^{+\infty} C(x, x_i)p(x)dx = \int_{-\infty}^{+\infty} [C^{ap}(x, x_i) + R_2(x, x_i)]p(x)dx \ .$$

Although the Taylor approximation is valid only in I, if that interval is chosen so as to be in the region where x lies most of the time (say $I = [u - 3\sqrt{v_x}, u + 3\sqrt{v_x}]$), and provided $C(.,.)$ is well behaved, we can write

$$l_i = l_i^{ap} + \int_{-\infty}^{+\infty} R_2(x, x_i)p(x)dx \ ,$$

where $l_i^{ap} = C(u) + \frac{v_x}{2}C''(u)$.

The error induced by the approximation is then

$$|l_i - l_i^{ap}| = \left| \int_{-\infty}^{+\infty} R_2(x, x_i)p(x)dx \right| = \left| \int_{-\infty}^{+\infty} \frac{1}{6}(x-u)^3 C^{(3)}(c, x_i)p(x)dx \right| \ ,$$

where c depends on x. Therefore, a lower bound on the error is

$$|l_i - l_i^{ap}| \leq \frac{1}{6}\int_{-\infty}^{+\infty} |x-u|^3 |C^{(3)}(c, x_i)| p(x)dx \leq \frac{M}{6} \ ,$$

where $M = \max_x\{|x-u|^3|C^{(3)}(c(x), x_i)|\}$, emphasising the dependence of c on x.

Similar bounds could be derived for l and l_{ij}. Nevertheless, we do not investigate this point any further (refer to (Evans and Swartz, 1997) for the derivation of exact error-bounds when approximating integrands by a Taylor polynomial). Note that the above assumes that the function C is such that the approximation holds. We have not defined the corresponding family of ‘valid’ covariance functions but we speculate that covariance functions with compact support (i.e. vanishing when the distance between the inputs is larger than a certain cut-off distance) would be good candidates.

Appendix C

Appendix to Chapter 6

This appendix provides the detailed calculations of the results presented in Chapter 6. The first section deals with the mixed GP, formed of function and derivative observations and Section C.2 with the control of a MISO system.

C.1 Mixed training set: Prediction at a noisy input

In the case where the training data is formed of derivative as well as function observations (presented in Section 6.1), we now derive the expressions for the mean and variance (hereafter referred to as *noisy* mean and *noisy* variance), when predicting at a noisy input $\mathbf{x} \sim \mathcal{N}_{\mathbf{x}}(\mathbf{u}, \mathbf{\Sigma}_x)$, assuming the covariance function is Gaussian.

The last subsection (C.1.3) also provides the details for the computation of the cross-covariance terms of the input covariance matrix when this *mixed* set is used in an iterative multiple-step ahead prediction framework with propagation of the uncertainty.

Recall that the *noisy* predictive mean and variance we want to compute are given by (Chapter 6.1, Section 6.1.3)

$$\begin{aligned}
m(\mathbf{u}, \mathbf{\Sigma}_x) &= m_y(\mathbf{u}, \mathbf{\Sigma}_x) + \sum_{d=1}^{D} m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x) \\
v(\mathbf{u}, \mathbf{\Sigma}_x) &= v_y(\mathbf{u}, \mathbf{\Sigma}_x) + \sum_{d,e=1}^{D} v_{y^d y^e}(\mathbf{u}, \mathbf{\Sigma}_x) + 2\sum_{d=1}^{D} v_{yy^d}(\mathbf{u}, \mathbf{\Sigma}_x)
\end{aligned}$$

where $m_y(\mathbf{u}, \mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu_y(\mathbf{x})]$, $m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]$ and

$$\begin{aligned}
v_y(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma_y^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x})^2] - E_{\mathbf{x}}[\mu_y(\mathbf{x})]^2 \\
v_{y^d y^e}(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma_{y^d y^e}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x})] - E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^e}(\mathbf{x})] \\
v_{yy^d}(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma_{yy^d}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x})\mu_{y^d}(\mathbf{x})] - E_{\mathbf{x}}[\mu_y(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})] \; .
\end{aligned}$$

In the case of the Gaussian covariance function, we already have the expressions for $m_y(\mathbf{u}, \mathbf{\Sigma}_x)$ and $v_y(\mathbf{u}, \mathbf{\Sigma}_x)$, which were derived in Chapter 3, Section 3.4.2. We have

$$m_y(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_{i=1}^{N} \beta_i C_G(\mathbf{u}, \mathbf{x}_i) C_{corr}(\mathbf{u}, \mathbf{x}_i) \; , \tag{C.1}$$

where $C_G(.,.)$ is the Gaussian kernel and $C_{corr}(\mathbf{u}, \mathbf{x}_i)$ is as given by (3.40), and

$$v_y(\mathbf{u}, \mathbf{\Sigma}_x) = C_G(\mathbf{u}, \mathbf{u}) - \sum_{i,j=1}^{N} (K_{ij}^{-1} - \beta_i\beta_j) C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) - m_y(\mathbf{u}, \mathbf{\Sigma}_x)^2 \; , \tag{C.2}$$

with $C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}})$ given by (3.41) and where $\bar{\mathbf{x}} = \frac{\mathbf{x}_i + \mathbf{x}_j}{2}$.

In the case of the Gaussian covariance function, the covariance function of the derivative process in direction d is given by (see Appendix A)

$$C_{y^d}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial^2 C_y(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_i^d x_j^d} = [w_d - w_d^2 (x_i^d - x_j^d)^2] C_y(\mathbf{x}_i, \mathbf{x}_j) \; ,$$

the cross-covariance between y^d and y is

$$C_{yy^d}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial C_y(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_j^d} = w_d (x_i^d - x_j^d) C_y(\mathbf{x}_i, \mathbf{x}_j) \; ,$$

and that between the derivatives in direction d and e is

$$C_{y^d y^e}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\partial^2 C_y(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_i^d \partial x_j^e} = [w_e \delta_{de} - w_d w_e (x_i^d - x_j^d)(x_i^e - x_j^e)] C_y(\mathbf{x}_i, \mathbf{x}_j) \; ,$$

where $C_y(\mathbf{x}_i, \mathbf{x}_j) = C_G(\mathbf{x}_i, \mathbf{x}_j)$, which, as in Chapter 3 Section 3.4.2, we denote by $cN_{\mathbf{x}_i}(\mathbf{x}_j, \mathbf{W})$, where $\mathbf{W}^{-1} = \text{diag}[w_1 \ldots w_D]$ and $c = (2\pi)^{D/2} |\mathbf{W}|^{1/2} v_0$.

We can now turn to the computation of $m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x)$, $v_{y^d y^e}(\mathbf{u}, \mathbf{\Sigma}_x)$ and $v_{yy^d}(\mathbf{u}, \mathbf{\Sigma}_x)$.

C.1.1 *Noisy* mean

We first need to compute $m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) = E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]$ where, in the case of the Gaussian covariance function, we have (see Section 6.1.2, omitting the limits of the summation)

$$\mu_{y^d}(\mathbf{x}) = \sum_i \beta_i C_{yy^d}(\mathbf{x},\mathbf{x}_i) = cw_d \sum_i \beta_i (x^d - x_i^d) N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W}) \ .$$

We then need to compute

$$\begin{aligned} m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) &= cw_d \sum_i \beta_i \int (x^d - x_i^d) N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W}) p(\mathbf{x}) d\mathbf{x} \\ &= cw_d \sum_i \beta_i \left(\int x^d N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W}) p(\mathbf{x}) d\mathbf{x} - x_i^d \int N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W}) p(\mathbf{x}) d\mathbf{x} \right) \\ &= cw_d \sum_i \beta_i \left[l_i^2 - x_i^d l_i^1 \right] \ . \end{aligned}$$

Using the product of Gaussians $N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})\mathcal{N}_{\mathbf{x}}(\mathbf{u},\mathbf{\Sigma}_x) = \mathbf{z}_i \mathcal{N}_{\mathbf{x}}(\mathbf{c}_i,\mathbf{C})$, with

$$\mathbf{c}_i = \mathbf{C}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x^{-1}\mathbf{u}) \ , \quad \mathbf{C} = (\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1} \ , \quad \mathbf{z}_i = N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x) \ , \tag{C.3}$$

we have

$$l_i^1 = \int N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})\mathcal{N}_{\mathbf{x}}(\mathbf{u},\mathbf{\Sigma}_x) d\mathbf{x} = \mathbf{z}_i \int \mathcal{N}_{\mathbf{x}}(\mathbf{c}_i,\mathbf{C}) d\mathbf{x} = \mathbf{z}_i \ , \tag{C.4}$$

and

$$l_i^2 = \int x^d N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W}) p(\mathbf{x}) d\mathbf{x} = \mathbf{z}_i \int x^d \mathcal{N}_{\mathbf{x}}(\mathbf{c}_i,\mathbf{C}) d\mathbf{x} = \mathbf{z}_i c_i^d \ , \tag{C.5}$$

where c_i^d is the d^{th} component of $\mathbf{c}_i$. Replacing l_i^1, l_i^2 and $\mathbf{z}_i$ by their expressions, we finally have

$$m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) = cw_d \sum_i \beta_i N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)[c_i^d - x_i^d]$$

or, using $cN_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x) = C_G(\mathbf{u},\mathbf{x}_i)C_{corr}(\mathbf{u},\mathbf{x}_i)$ (as found in Chapter 3, Section 3.4.2),

$$m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) = w_d \sum_i \beta_i (c_i^d - x_i^d) C_G(\mathbf{u},\mathbf{x}_i) C_{corr}(\mathbf{u},\mathbf{x}_i) \ , \tag{C.6}$$

where c_i^d is the d^{th} component of $(\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x^{-1}\mathbf{u})$.

We can then compute $m(\mathbf{u},\mathbf{\Sigma}_x) = m_y(\mathbf{u},\mathbf{\Sigma}_x) + \sum_{d=1}^D m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)$. Replacing $m_y(\mathbf{u},\mathbf{\Sigma}_x)$ and $m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)$ by their expressions finally gives

$$\boxed{m(\mathbf{u},\mathbf{\Sigma}_x) = \sum_i \beta_i C_G(\mathbf{u},\mathbf{x}_i) C_{corr}(\mathbf{u},\mathbf{x}_i) \left(1 + \sum_{d=1}^D w_d (c_i^d - x_i^d)\right)} \tag{C.7}$$

C.1.2 *Noisy* variance

To compute the variance $v(\mathbf{u},\mathbf{\Sigma}_x)$, we need $v_{y^dy^e}(\mathbf{u},\mathbf{\Sigma}_x)$ and $v_{yy^d}(\mathbf{u},\mathbf{\Sigma}_x)$.

Computing $v_{y^dy^e}(\mathbf{u},\mathbf{\Sigma}_x)$

We have

$$\begin{aligned} v_{y^dy^e}(\mathbf{u},\mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma^2_{y^dy^e}(\mathbf{x})] + E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x})] - E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})]E_{\mathbf{x}}[\mu_{y^e}(\mathbf{x})] \\ &= E_{\mathbf{x}}[\sigma^2_{y^dy^e}(\mathbf{x})] + E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x})] - m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)m_{y^e}(\mathbf{u},\mathbf{\Sigma}_x)\ , \end{aligned}$$

where $m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)$ and $m_{y^e}(\mathbf{u},\mathbf{\Sigma}_x)$ are given by (C.6).

For the Gaussian covariance function, we have

$$\begin{aligned} \sigma^2_{y^dy^e}(\mathbf{x}) &= -\sum_{i,j} K^{-1}_{ij} C_{yy^d}(\mathbf{x},\mathbf{x}_i)C_{yy^e}(\mathbf{x},\mathbf{x}_j) \\ &= -c^2 w_d w_e \sum_{i,j} K^{-1}_{ij}(x^d - x^d_i)N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})(x^e - x^e_j)N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})\ , \end{aligned}$$

and also

$$\mu_{y^d}(\mathbf{x})\mu_{y^e}(\mathbf{x}) = c^2 w_d w_e \sum_{i,j} \beta_i\beta_j (x^d - x^d_i)N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})(x^e - x^e_j)N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})\ .$$

We can then write

$$\begin{aligned} v_{y^dy^e}(\mathbf{u},\mathbf{\Sigma}_x) &= c^2 w_d w_e \sum_{i,j}(\beta_i\beta_j - K^{-1}_{ij})E_{\mathbf{x}}[(x^d - x^d_i)N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})(x^e - x^e_j)N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})] \\ &\quad - m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)m_{y^e}(\mathbf{u},\mathbf{\Sigma}_x)\ , \end{aligned}$$

or

$$v_{y^dy^e}(\mathbf{u},\mathbf{\Sigma}_x) = c^2 w_d w_e \sum_{i,j}(\beta_i\beta_j - K^{-1}_{ij})\left[L^2_{ij} - x^e_j L^{3d}_{ij} - x^d_i L^{3e}_{ij} + x^d_i x^e_j L^1_{ij}\right] - m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)m_{y^e}(\mathbf{u},\mathbf{\Sigma}_x)$$

with

$$\begin{aligned} L^1_{ij} &= E_{\mathbf{x}}[N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})] \\ L^2_{ij} &= E_{\mathbf{x}}[x^d x^e N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})] \\ L^{3e}_{ij} &= E_{\mathbf{x}}[x^e N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})]\ , \end{aligned}$$

with L_{ij}^{3d} same as L_{ij}^{3e}, albeit in d. Using the product of Gaussians, we have

$$\begin{aligned}\int N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W}) N_{\mathbf{x}}(\mathbf{x}_j, \mathbf{W}) p(\mathbf{x}) d\mathbf{x} &= N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) \int N_{\mathbf{x}} \left(\frac{\mathbf{x}_i + \mathbf{x}_j}{2}, \frac{\mathbf{W}}{2} \right) p(\mathbf{x}) d\mathbf{x} \\ &= N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}} \left(\frac{\mathbf{x}_i + \mathbf{x}_j}{2}, \frac{\mathbf{W}}{2} + \mathbf{\Sigma}_x \right) \int \mathcal{N}_{\mathbf{x}}(\mathbf{c}_{ij}, \mathbf{C}) d\mathbf{x}\end{aligned}$$

with

$$\mathbf{c}_{ij} = \mathbf{C} \left(\left(\frac{\mathbf{W}}{2} \right)^{-1} \frac{\mathbf{x}_i + \mathbf{x}_j}{2} + \mathbf{\Sigma}_x^{-1} \mathbf{u} \right), \quad \mathbf{C} = \left(\left(\frac{\mathbf{W}}{2} \right)^{-1} + \mathbf{\Sigma}_x^{-1} \right)^{-1} . \qquad \text{(C.8)}$$

We can then write $L_{ij}^1 = N_{\mathbf{x}_i}(\mathbf{x}_j, 2\mathbf{W}) N_{\mathbf{u}} \left(\frac{\mathbf{x}_i + \mathbf{x}_j}{2}, \frac{\mathbf{W}}{2} + \mathbf{\Sigma}_x \right)$, and we have $c^2 L_{ij}^1 = C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}})$ (from Chapter 3, Section 3.4.2). This leads to

$$\begin{aligned} c^2 L_{ij}^2 &= C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) \int x^d x^e \mathcal{N}_{\mathbf{x}}(\mathbf{c}_{ij}, \mathbf{C}) d\mathbf{x} \\ &= C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) (C_{de} + c_{ij}^d c_{ij}^e) \end{aligned} \qquad \text{(C.9)}$$

and

$$\begin{aligned} c^2 L_{ij}^{3e} &= C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) \int x^e \mathcal{N}_{\mathbf{x}}(\mathbf{c}_{ij}, \mathbf{C}) d\mathbf{x} \\ &= C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) c_{ij}^e \end{aligned} \qquad \text{(C.10)}$$

and similarly for L_{ij}^{3d}, where c_{ij}^d is the d^{th} component of $\mathbf{c}_{ij}$ and C_{de} the (d, e) entry of the $D \times D$ matrix $\mathbf{C}$, as given by (C.8).

We can then write

$$\begin{aligned} v_{y^d y^e}(\mathbf{u}, \mathbf{\Sigma}_x) = w_d w_e \sum_{i,j} (\beta_i \beta_j - K_{ij}^{-1}) C_G(\mathbf{u}, \mathbf{x}_i) C_G(\mathbf{u}, \mathbf{x}_j) C_{corr_2}(\mathbf{u}, \bar{\mathbf{x}}) \Big[C_{de} + c_{ij}^d c_{ij}^e \\ - x_j^e c_{ij}^d - x_i^d c_{ij}^e + x_i^d x_j^e \Big] - m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x) m_{y^e}(\mathbf{u}, \mathbf{\Sigma}_x) \ . \end{aligned} \qquad \text{(C.11)}$$

Computing $v_{yy^d}(\mathbf{u}, \mathbf{\Sigma}_x)$

The last element we need to compute the *noisy* variance is

$$\begin{aligned} v_{yy^d}(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma_{yy^d}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x}) \mu_{y^d}(\mathbf{x})] - E_{\mathbf{x}}[\mu_y(\mathbf{x})] E_{\mathbf{x}}[\mu_{y^d}(\mathbf{x})] \\ &= E_{\mathbf{x}}[\sigma_{yy^d}^2(\mathbf{x})] + E_{\mathbf{x}}[\mu_y(\mathbf{x}) \mu_{y^d}(\mathbf{x})] - m_y(\mathbf{u}, \mathbf{\Sigma}_x) m_{y^d}(\mathbf{u}, \mathbf{\Sigma}_x) \ , \end{aligned}$$

where $m_y(\mathbf{u},\mathbf{\Sigma}_x)$ and $m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)$ are given by (C.1) and (C.6) respectively, and with

$$\begin{aligned}\sigma^2_{yy^d}(\mathbf{x}) &= -\sum_{i,j} K^{-1}_{ij} C_y(\mathbf{x},\mathbf{x}_i)C_{yy^d}(\mathbf{x},\mathbf{x}_j) && \text{(C.12)}\\ &= -c^2 w_d \sum_{i,j} K^{-1}_{ij} N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})(x^d - x^d_j)N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W}) && \text{(C.13)}\end{aligned}$$

and

$$\mu_y(\mathbf{x})\mu_{y^d}(\mathbf{x}) = c^2 w_d \sum_{i,j} \beta_i N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})\beta_j(x^d - x^d_j)N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W}) \; . \qquad \text{(C.14)}$$

We then have

$$\begin{aligned}v_{yy^d}(\mathbf{u},\mathbf{\Sigma}_x) &= c^2 w_d \sum_{i,j}(\beta_i\beta_j - K^{-1}_{ij})E_{\mathbf{x}}[N_{\mathbf{x}}(\mathbf{x}_i,\mathbf{W})(x^d - x^d_j)N_{\mathbf{x}}(\mathbf{x}_j,\mathbf{W})] - m_y(\mathbf{u},\mathbf{\Sigma}_x)m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)\\ &= c^2 w_d \sum_{i,j}(\beta_i\beta_j - K^{-1}_{ij})[L^{3d}_{ij} - x^d_j L^1_{ij}] - m_y(\mathbf{u},\mathbf{\Sigma}_x)m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) \; .\end{aligned}$$

Using the previous results, we can directly write

$$v_{yy^d}(\mathbf{u},\mathbf{\Sigma}_x) = w_d \sum_{i,j}(\beta_i\beta_j - K^{-1}_{ij})C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)C_{corr_2}(\mathbf{u},\bar{\mathbf{x}})[c^d_{ij} - x^d_j] - m_y(\mathbf{u},\mathbf{\Sigma}_x)m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x) \; , \qquad \text{(C.15)}$$

where again c^d_{ij} is the d^{th} element of $\mathbf{C}\left(\left(\frac{\mathbf{W}}{2}\right)^{-1}\frac{\mathbf{x}_i+\mathbf{x}_j}{2} + \mathbf{\Sigma}_x^{-1}\mathbf{u}\right)$, with $\mathbf{C} = \left(\left(\frac{\mathbf{W}}{2}\right)^{-1} + \mathbf{\Sigma}_x^{-1}\right)^{-1}$.

Finally...

We can now compute $v(\mathbf{u},\mathbf{\Sigma}_x) = v_y(\mathbf{u},\mathbf{\Sigma}_x) + \sum_{d,e=1}^D v_{y^d y^e}(\mathbf{u},\mathbf{\Sigma}_x) + 2\sum_{d=1}^D v_{yy^d}(\mathbf{u},\mathbf{\Sigma}_x)$. Replacing the different terms by their expressions, we finally have

$$\boxed{\begin{aligned}v(\mathbf{u},\mathbf{\Sigma}_x) = C_G(\mathbf{u},\mathbf{u}) &+ \sum_{i,j}(\beta_i\beta_j - K^{-1}_{ij})\left[1 + \sum_{d,e=1}^D w_d w_e(C_{de} + c^d_{ij}c^e_{ij} - x^e_j c^d_{ij} - x^d_i c^e_{ij} + x^d_i x^e_j)\right.\\ &\left. +2\sum_{d=1}^D w_d(c^d_{ij} - x^d_j)\right] C_G(\mathbf{u},\mathbf{x}_i)C_G(\mathbf{u},\mathbf{x}_j)C_{corr_2}(\mathbf{u},\bar{\mathbf{x}}) - m_y(\mathbf{u},\mathbf{\Sigma}_x)^2\\ &- \sum_{d,e=1}^D m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)m_{y^e}(\mathbf{u},\mathbf{\Sigma}_x) - 2m_y(\mathbf{u},\mathbf{\Sigma}_x)\sum_{d=1}^D m_{y^d}(\mathbf{u},\mathbf{\Sigma}_x)\end{aligned}} \qquad \text{(C.16)}$$

C.1.3 Cross-covariance terms for the iterative multi-step-ahead forecasting

Suppose we now wish to apply the multiple-step-ahead iterative forecasting algorithm presented in Chapter 4, Section 4.2.1. With this model formed of *mixed* observations, nothing is changed apart from the input covariance matrix.

For $t+k$, at $\mathbf{x}_{t+k} \sim \mathcal{N}(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k})$, we compute $y_{t+k} \sim \mathcal{N}(m(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k}), v(\mathbf{u}_{t+k}, \mathbf{\Sigma}_{t+k}))$, where $m(.)$ and $v(.)$ correspond to the above *noisy* mean and variance. At that time-step, the mean of the random state, $\mathbf{u}_{t+k}$, is composed of the delayed previously predicted means, and the covariance matrix $\mathbf{\Sigma}_{t+k}$ has the corresponding variances on its diagonal. As in Section 4.2.1, the cross-covariance terms of $\mathbf{\Sigma}_{t+k}$ are given by

$$\mathrm{Cov}[y_{t+l}, \mathbf{x}_{t+l}] = E[y_{t+l}\mathbf{x}_{t+l}] - E[y_{t+l}]E[\mathbf{x}_{t+l}] \tag{C.17}$$

with $E[\mathbf{x}_{t+l}] = \mathbf{u}_{t+l}$, $E[y_{t+l}] = m(\mathbf{u}_{t+l}, \mathbf{\Sigma}_{t+l})$, and $E[y_{t+l}\mathbf{x}_{t+l}] = \int \mathbf{x}_{t+l}\mu(\mathbf{x}_{t+l})p(\mathbf{x}_{t+l})d\mathbf{x}_{t+l}$.

We now have $\mu(\mathbf{x}_{t+l}) = \mu_y(\mathbf{x}_{t+l}) + \sum_{d=1}^{D} \mu_{y^d}(\mathbf{x}_{t+l})$, so that we need to compute

$$\begin{aligned}
E[y_{t+l}\mathbf{x}_{t+l}] &= \int \mathbf{x}_{t+l}\mu_y(\mathbf{x}_{t+l})p(\mathbf{x}_{t+l})d\mathbf{x}_{t+l} + \sum_{d=1}^{D}\int \mathbf{x}_{t+l}\mu_{y^d}(\mathbf{x}_{t+l})p(\mathbf{x}_{t+l})d\mathbf{x}_{t+l} \\
&= c\sum_i \beta_i \Bigg[\int \mathbf{x}_{t+l}N_{\mathbf{x}_{t+l}}(\mathbf{x}_i, \mathbf{W})p(\mathbf{x}_{t+l})d\mathbf{x}_{t+l} \\
&\quad + \sum_{d=1}^{D} w_d \int \mathbf{x}_{t+l}(x^d - x_i^d)N_{\mathbf{x}_{t+l}}(\mathbf{x}_i, \mathbf{W})p(\mathbf{x}_{t+l})d\mathbf{x}_{t+l}\Bigg] .
\end{aligned}$$

Denoting $\mathbf{x}_{t+l}$ by $\mathbf{x}$ for notational convenience, we have $I_i = \int \mathbf{x}N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W})p(\mathbf{x})d\mathbf{x}$ and $I_i^d = \int \mathbf{x}(x^d - x_i^d)N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W})p(\mathbf{x})d\mathbf{x}$. As before, using $N_{\mathbf{x}}(\mathbf{x}_i, \mathbf{W})p(\mathbf{x}) = N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\mathcal{N}_{\mathbf{x}}(\mathbf{c}_i, \mathbf{C})$, with

$$\mathbf{C} = (\mathbf{W}^{-1} + \mathbf{\Sigma}_x^{-1})^{-1}, \quad \mathbf{c}_i = \mathbf{C}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x^{-1}\mathbf{u}) , \tag{C.18}$$

we have

$$I_i = N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\int \mathbf{x}\mathcal{N}_{\mathbf{x}}(\mathbf{c}_i, \mathbf{C}) = N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\mathbf{c}_i$$

and

$$\begin{aligned}
I_i^d &= N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\left(\int \mathbf{x}x^d\mathcal{N}_{\mathbf{x}}(\mathbf{c}_i, \mathbf{C})d\mathbf{x} - x_i^d\int \mathbf{x}\mathcal{N}_{\mathbf{x}}(\mathbf{c}_i, \mathbf{C})d\mathbf{x}\right) \\
&= N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\left(\mathbf{C}^{[d]} + \mathbf{c}_i c_i^d - x_i^d\mathbf{c}_i\right) ,
\end{aligned}$$

where $\mathbf{C}^{[d]}$ is the d^{th} column of the matrix $\mathbf{C}$ and c_i^d is the d^{th} element of $\mathbf{c}_i$.

We therefore have

$$E[y\mathbf{x}] = c\sum_i \beta_i N_{\mathbf{u}}(\mathbf{x}_i, \mathbf{W} + \mathbf{\Sigma}_x)\left[\mathbf{c}_i + \sum_{d=1}^{D} w_d\left(\mathbf{C}^{[d]} + \mathbf{c}_i c_i^d - x_i^d \mathbf{c}_i\right)\right] . \tag{C.19}$$

Finally, we can write the cross-covariance terms as

$$\begin{aligned}\mathrm{Cov}[y_{t+l}, \mathbf{x}_{t+l}] = & \sum_i \beta_i C_G(\mathbf{u}_{t+l}, \mathbf{x}_i) C_{corr}(\mathbf{u}_{t+l}, \mathbf{x}_i)\left[\mathbf{c}_i + \sum_{d=1}^{D} w_d\left(\mathbf{C}^{[d]} + \mathbf{c}_i c_i^d - x_i^d \mathbf{c}_i\right)\right] \\ & - m(\mathbf{u}_{t+l}, \mathbf{\Sigma}_{t+l})\mathbf{u}_{t+l}\end{aligned} \tag{C.20}$$

with $\mathbf{c}_i = \mathbf{C}(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_{t+l}^{-1}\mathbf{u}_{t+l})$, $\mathbf{C} = (\mathbf{W}^{-1} + \mathbf{\Sigma}_{t+l}^{-1})^{-1}$, where $\mathbf{C}^{[d]}$ is the d^{th} column of $\mathbf{C}$ and c_i^d is the d^{th} element of $\mathbf{c}_i$.

C.2 GP modelling of an affine MISO system

Consider the affine nonlinear system

$$y_{t+1} = f(\mathbf{x}_t) + g(\mathbf{x}_t)u_t + e_{t+1} , \tag{C.21}$$

where the state vector at time t is $\mathbf{x}_t = [y_t, \dots, y_{t-n}, u_{t-1}, \dots, u_{t-m}]^T$, y_{t+1} is the one-step-ahead system's output, u_t is the current control signal, e_{t+1} is a white noise with variance v_t and $f(.)$ and $g(.)$ are two smooth nonlinear functions.

Following (Sbarbaro and Murray-Smith, 2003), we can model this system using a Gaussian Process with zero-mean and a covariance function reflecting the 'functional' part, $f(\mathbf{x}_t)$, and the 'control' part, as an affine function, just as in (C.21):

$$C([\mathbf{x}_i \; u_i], [\mathbf{x}_j \; u_j]) = C_x(\mathbf{x}_i, \mathbf{x}_j) + u_i C_u(\mathbf{x}_i, \mathbf{x}_j)u_j . \tag{C.22}$$

For simplicity, we use the same 'Gaussian structure' for both C_x and C_u:

$$C_a(\mathbf{x}_i, \mathbf{x}_j) = v_a \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}_a^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right] , \tag{C.23}$$

for $a = x, u$ and $\mathbf{W}_a^{-1} = \mathrm{diag}[w_{a1}, \dots, \mathbf{w}_{aD}]$, therefore allowing different parameters for the two covariance functions.

C.2.1 Predictive distribution of y_{t+1}

Assuming we have observed $y_1, \dots, y_t$, the predictive distribution of y_{t+1} is readily obtained. We can write the predictive mean and variance in such a way to highlight the 'control input' and 'state' parts:

$$\begin{aligned}
\mu([\mathbf{x}_t, u_t]) &= \sum_i \beta_i [C_x(\mathbf{x}_t, \mathbf{x}_i) + u_t C_u(\mathbf{x}_t, \mathbf{x}_i) u_i] \\
\sigma^2([\mathbf{x}_t, u_t]) &= v_x + u_t^2 v_u - \sum_{ij} K_{ij}^{-1} [C_x(\mathbf{x}_t, \mathbf{x}_i) + u_t C_u(\mathbf{x}_t, \mathbf{x}_i) u_i] \\
&\quad [C_x(\mathbf{x}_t, \mathbf{x}_j) + u_t C_u(\mathbf{x}_t, \mathbf{x}_j) u_j] \, ,
\end{aligned}$$

using $C([\mathbf{x}_t, u_t], [\mathbf{x}_i, u_i]) = C_x(\mathbf{x}_t, \mathbf{x}_i) + u_t C_u(\mathbf{x}_t, \mathbf{x}_i) u_i$ and $C([\mathbf{x}_t, u_t], [\mathbf{x}_t, u_t]) = C_x(\mathbf{x}_t, \mathbf{x}_t) + u_t^2 C_u(\mathbf{x}_t, \mathbf{x}_t)$, where, according to equation (C.23), $C_x(\mathbf{x}_t, \mathbf{x}_t) = v_x$ and $C_u(\mathbf{x}_t, \mathbf{x}_t) = v_u$. More simply, we have

$$\mu([\mathbf{x}_t, u_t]) = \mu_x(\mathbf{x}_t) + u_t \mu_u(\mathbf{x}_t) \qquad \text{(C.24)}$$

with

$$\begin{aligned}
\mu_x(\mathbf{x}_t) &= \sum_i \beta_i C_x(\mathbf{x}_t, \mathbf{x}_i) && \text{(C.25)} \\
\mu_u(\mathbf{x}_t) &= \sum_i \beta_i C_u(\mathbf{x}_t, \mathbf{x}_i) u_i \, , && \text{(C.26)}
\end{aligned}$$

and

$$\sigma^2([\mathbf{x}_t, u_t]) = \sigma_x^2(\mathbf{x}_t) + u_t^2 \sigma_u^2(\mathbf{x}_t) - 2 u_t \sigma_{x,u}^2(\mathbf{x}_t) \qquad \text{(C.27)}$$

with

$$\begin{aligned}
\sigma_x^2(\mathbf{x}_t) &= v_x - \sum_{ij} K_{ij}^{-1} C_x(\mathbf{x}_t, \mathbf{x}_i) C_x(\mathbf{x}_t, \mathbf{x}_j) && \text{(C.28)} \\
\sigma_u^2(\mathbf{x}_t) &= v_u - \sum_{ij} K_{ij}^{-1} C_u(\mathbf{x}_t, \mathbf{x}_i) u_i C_u(\mathbf{x}_t, \mathbf{x}_j) u_j && \text{(C.29)} \\
\sigma_{x,u}^2(\mathbf{x}_t) &= \sum_{ij} K_{ij}^{-1} C_x(\mathbf{x}_t, \mathbf{x}_i) C_u(\mathbf{x}_t, \mathbf{x}_j) u_j \, . && \text{(C.30)}
\end{aligned}$$

C.2.2 Predicting y_{t+k}

As in Chapter 3, we can compute the predictive mean and variance at time $t + k$, accounting for the uncertainty on the previous points and the 'mixed' (functional and control) form of the covariance function.

Let $\mathbf{x}(t+k-1) \sim \mathcal{N}(\mathbf{u}, \mathbf{\Sigma}_x)$ be the input corresponding to $y(t+k)$. The mean $m(\mathbf{u}, \mathbf{\Sigma}_x)$ and variance $v(\mathbf{u}, \mathbf{\Sigma}_x)$ of $y(t+k)$ are given by

$$\begin{aligned} m(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\mu(\mathbf{x}(t+k-1))] & \text{(C.31)} \\ v(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\sigma^2(\mathbf{x}(t+k-1))] + E_{\mathbf{x}}[\mu(\mathbf{x}(t+k-1))^2] - m(\mathbf{u}, \mathbf{\Sigma}_x)^2 \ , & \text{(C.32)} \end{aligned}$$

where $\mu(\mathbf{x}(t+k-1))$ and $\sigma^2(\mathbf{x}(t+k-1))$ are given by (C.24) and (C.27) computed at $\mathbf{x}(t+k-1)$. We therefore need to evaluate

$$\begin{aligned} m(\mathbf{u}, \mathbf{\Sigma}_x) &= E_{\mathbf{x}}[\mu_x(\mathbf{x}(t+k-1)) + u(t+k-1)\mu_u(\mathbf{x}(t+k-1))] \\ &= E_{\mathbf{x}}[\mu_x(\mathbf{x}(t+k-1))] + u(t+k-1)E_{\mathbf{x}}[\mu_u(\mathbf{x}(t+k-1))] \\ &= \sum_i \beta_i E_{\mathbf{x}}[C_x(\mathbf{x}(t), \mathbf{x}_i)] + u(t+k-1)\sum_i \beta_i u_i E_{\mathbf{x}}[C_u(\mathbf{x}(t), \mathbf{x}_i)] \ . \end{aligned}$$

From Chapter 3, we can directly write

$$l_i^a = E_{\mathbf{x}}[C_a(\mathbf{x}(t), \mathbf{x}_i)] = |I + \mathbf{W}_a^{-1}\mathbf{\Sigma}_x|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{u})^T(\mathbf{W}_a + \mathbf{\Sigma}_x)^{-1}(\mathbf{x}_i - \mathbf{u})\right] \ ,$$

where $a = \{x, u\}$, leading to

$$m(\mathbf{u}, \mathbf{\Sigma}_x) = \sum_i \beta_i (l_i^x + u(t+k-1)u_i l_i^u) \ . \qquad \text{(C.33)}$$

For the variance, replacing $\sigma^2(\mathbf{x}(t+k-1))$ by its expression, we have

$$\begin{aligned} v(\mathbf{u}, \mathbf{\Sigma}_x) = {} & E_{\mathbf{x}}[\sigma_x^2(\mathbf{x}(t+k-1))] + u(t+k-1)^2 E_{\mathbf{x}}[\sigma_u^2(\mathbf{x}(t+k-1))] \\ & -2u(t+k-1)E_{\mathbf{x}}[\sigma_{x,u}^2(\mathbf{x}(t+k-1))] + E_{\mathbf{x}}[\mu(\mathbf{x}(t+k-1))^2] - m(\mathbf{u}, \mathbf{\Sigma}_x)^2 \end{aligned}$$

which requires $E_{\mathbf{x}}[\sigma_a^2(\mathbf{x}(t+k-1))] = v_a - \sum_{ij} K_{ij}^{-1} L_{ij}^{aa}$ and $E_{\mathbf{x}}[\sigma_{x,u}^2(\mathbf{x}(t+k-1))] = \sum_{ij} K_{ij}^{-1} u_j L_{ij}^{xu}$, where

$$\begin{aligned} L_{ij}^{aa} &= E_{\mathbf{x}}[C_a(\mathbf{x}(t+k-1), \mathbf{x}_i)C_a(\mathbf{x}(t+k-1), \mathbf{x}_j)] \\ L_{ij}^{xu} &= E_{\mathbf{x}}[C_x(\mathbf{x}(t+k-1), \mathbf{x}_i)C_u(\mathbf{x}(t+k-1), \mathbf{x}_j)] \ , \end{aligned}$$

for $a = \{x, u\}$. The computation of L_{ij}^{aa} and L_{ij}^{xu} is very similar to what we have done before. We arrive at

$$\begin{aligned} L_{ij}^{aa} = |I + 2\mathbf{W}_a^{-1}\mathbf{\Sigma}_x|^{-1/2} & \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T(2\mathbf{W}_a)^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right] \\ & \exp\left[-\frac{1}{2}(\bar{\mathbf{x}} - \mathbf{u})^T(\mathbf{W}_a/2 + \mathbf{\Sigma}_x)^{-1}(\bar{\mathbf{x}} - \mathbf{u})\right] \ , \end{aligned} \qquad \text{(C.34)}$$

for $a = \{x, u\}$ and with $\bar{\mathbf{x}} = \frac{\mathbf{x}_i + \mathbf{x}_j}{2}$, and

$$\begin{aligned} L_{ij}^{xu} &= |I + (\mathbf{W}_x^{-1} + \mathbf{W}_u^{-1})\mathbf{\Sigma}_x|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{W}_x + \mathbf{W}_u)^{-1}(\mathbf{x}_i - \mathbf{x}_j)\right] \\ &\exp\left[-\frac{1}{2}(\mathbf{x}_{xu} - \mathbf{u})^T(\mathbf{W}_{xu} + \mathbf{\Sigma}_x)^{-1}(\mathbf{x}_{xu} - \mathbf{u})\right] , \end{aligned} \tag{C.35}$$

where $\mathbf{W}_{xu} = (\mathbf{W}_x^{-1} + \mathbf{W}_u^{-1})^{-1}$ and $\mathbf{x}_{xu} = \mathbf{W}_{xu}(\mathbf{W}_x^{-1}\mathbf{x}_i + \mathbf{W}_u^{-1}\mathbf{x}_j)$. Also, we have $E_{\mathbf{x}}[\mu(\mathbf{x}(t+k-1))^2] = E_{\mathbf{x}}[(\mu_x(\mathbf{x}(t+k-1)) + u(t+k-1)\mu_u(\mathbf{x}(t+k-1)))^2]$. Expanding the square and replacing μ_x and μ_u by their expressions gives

$$\begin{aligned} E_{\mathbf{x}}[\mu(\mathbf{x}(t+k-1))^2] &= E_{\mathbf{x}}[\mu_x(\mathbf{x}(t+k-1))^2] + 2u(t+k-1)E_{\mathbf{x}}[\mu_x(\mathbf{x}(t+k-1))\mu_u(\mathbf{x}(t+k-1))] \\ &\quad + u(t+k-1)^2 E_{\mathbf{x}}[\mu_u(\mathbf{x}(t+k-1))^2]] \\ &= \sum_{ij}\beta_i\beta_j\left(L_{ij}^{xx} + 2u(t+k-1)u_j L_{ij}^{xu} + u(t+k-1)^2 u_i u_j L_{ij}^{uu})\right) . \end{aligned}$$

We can then write

$$\begin{aligned} v(\mathbf{u}, \mathbf{\Sigma}_x) &= v_x + u(t+k-1)^2 v_u - \sum_{ij}(K_{ij}^{-1} - \beta_i\beta_j)\left[L_{ij}^{xx} + u(t+k-1)^2 L_{ij}^{uu}\right. \\ &\quad \left. + 2u(t+k-1)u_j L_{ij}^{xu}\right] - m(\mathbf{u}, \mathbf{\Sigma}_x)^2 , \end{aligned} \tag{C.36}$$

with Or, replacing $m(\mathbf{u}, \mathbf{\Sigma}_x)^2$ by its expression and after simplification,

$$\begin{aligned} v(\mathbf{u}, \mathbf{\Sigma}_x) &= \sigma^2(\mathbf{u}) + \sum_{i,j} K_{ij}^{-1}(C_x(\mathbf{u}, \mathbf{x}_i)C_x(\mathbf{u}, \mathbf{x}_j) - L_{ij}^{xx}) + \sum_{i,j}\beta_i\beta_j(L_{ij}^{xx} - l_i^x l_j^x) \\ &\quad + 2u(t+k-1)\sum_{i,j} u_j[\beta_i\beta_j(L_{ij}^{xu} - l_i^x l_j^u) + K_{ij}^{-1}(C_x(\mathbf{u}, \mathbf{x}_i)C_u(\mathbf{u}, \mathbf{x}_j) - L_{ij}^{xu})] \\ &\quad + u(t+k-1)^2\sum_{i,j} u_i u_j[\beta_i\beta_j(L_{ij}^{uu} - l_i^u l_j^u) + K_{ij}^{-1}(C_u(\mathbf{u}, \mathbf{x}_i)C_u(\mathbf{u}, \mathbf{x}_j) - L_{ij}^{uu})] . \end{aligned}$$

In these equations, the first n elements of $\mathbf{u}$, mean of $\mathbf{x}(t+k-1) = [y(t+k-1), \dots, y(t+k-1-n), u(t+k-2), \dots, u(t+k-m-1)]^T$, correspond to the delayed predictive means. As for $\mathbf{\Sigma}_x$, its first n diagonal elements are the corresponding predictive variances, while the cross-covariance terms are given by

$$\mathrm{Cov}(y(t+k), \mathbf{x}(t+k-1)) = \mathbf{l}^x\mathbf{c}^{xT}\boldsymbol{\beta} + u(t+k-1)(\mathbf{u}.\mathbf{l}^u\mathbf{c}^u)^T\boldsymbol{\beta} - m(\mathbf{u}, \mathbf{\Sigma}_x)\mathbf{u} \tag{C.37}$$

where $m(\mathbf{u}, \mathbf{\Sigma}_x)$ is given by (C.33) and $\mathbf{c}_i^a = C^a(\mathbf{W}_a^{-1}\mathbf{x}_i + \mathbf{\Sigma}_x\mathbf{u})^{-1})$ where $C^a = (\mathbf{W}_a^{-1} + \mathbf{\Sigma}_x)^{-1}$, for $a = \{x, u\}$.

Bibliography

Åström, K. J. (1995). Adaptive Control Around 1960. *Proc. Decision and Control*, pages 2784–2789.

Åström, K. J. and Wittenmark, B. (1990). *Computer-controlled systems. Theory and design*. Prentice Hall.

Abrahamsen, P. (1997). A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center.

Ahmad, S. and Tresp, V. (1993). Some Solutions to the Missing Feature Problem in Vision. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems*, volume 5, pages 393–400. Morgan Kaufmann, San Mateo, CA.

Bakker, R., Schouten, J. C., van den Bleek, C. M., and Giles, C. L. (1998). Neural Learning of Chaotic Dynamics: The Error Propagation Algorithm. In *Proc. Int. Joint Conf. on Neural Networks*, pages 2483–2488.

Barber, D. and Bishop, C. (1998). Ensemble learning for multi-layer networks. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems*, volume 10, pages 395–401. MIT Press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time series analysis. Forecasting and control*. Prentice Hall.

Carlin, B. P. and Louis, T. A. (2000). *Bayes and empirical Bayes methods for data analysis*. Chapman&Hall.

Casdagli, M. (1989). Nonlinear prediction of chaotic times series. *Physica D*, 35:335–356.

Chen, R., Yang, L., and Hafner, C. (2004). Nonparametric multi-step ahead prediction in time series analysis. *Journal of the Royal Statistical Society B*. To appear.

Cressie, N. A. C. (1993). *Statistics for spatial data*. Wiley.

Dellaportas, P. and Stephens, D. A. (1995). Bayesian Analysis of Errors-in-Variables Regression Models. *Biometrics*, 51:1085–1095.

Evans, M. and Swartz, T. (1997). An algorithm for the approximation of integrals with exact error bounds. Technical report, Department of Statistics, University of Toronto.

Fabri, S. and Kadirkamanathan, V. (1998). Dual adaptive control of nonlinear stochastic systems using neural networks. *Automatica*, 34(2):245–253.

Farmer, J. D. and Sidorowich, J. J. (1988). Exploiting chaos to predict the future and reduce noise. Technical Report LA-UR-88-901, Los Alamos National Laboratory, New Mexico.

Freedman, L., Fainberg, V., Kipnis, V., and Carroll, R. J. (2004). A new method for dealing with measurement error in explanatory variables of regression models. *Biometrics*. To appear.

Genton, M. G. (2001). Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312.

Gers, F. A., Eck, D., and Schmidhuber, J. (2001). Applying LSTM to Time Series Predictable through Time-Window Approaches. *Lecture Notes in Computer Science*, 2130:669+.

Gershenfeld, N. (1999). *The Nature of Mathematical Modeling*. Cambridge University Press.

Ghahramani, Z. and Jordan, M. I. (1994a). Learning from Incomplete Data. Technical Report 108, MIT Center for Biological and Computational Learning.

Ghahramani, Z. and Jordan, M. I. (1994b). Supervised learning from incomplete data via the EM approach. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6. San Mateo, CA: Morgan Kaufmann.

Ghahramani, Z. and Roweis, S. (1999). Learning nonlinear dynamical systems using an EM algorithm. In Kearns, M. S., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems*, volume 11, pages 599–605. MIT Press.

Gibbs, M. N. (1997). *Bayesian Gaussian Processes for regression and classification*. PhD thesis, University of Cambridge.

Girard, A. and Murray-Smith, R. (2003). Learning a Gaussian Process prior model with uncertain inputs. Technical Report TR-2003-144, Computing Science Department, University of Glasgow.

Girard, A., Rasmussen, C., and Murray-Smith, R. (2002). Gaussian Process Priors with Uncertain Inputs: Multiple-Step Ahead Prediction. Technical Report TR-2002-119, Computing Science Department, University of Glasgow.

Girard, A., Rasmussen, C., Quinonero-Candela, J., and Murray-Smith, R. (2003). Gaussian Process Priors With Uncertain Inputs – Application to Multiple-Step Ahead Time Series Forecasting. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15, pages 545–552. MIT Press.

Gneiting, T. (2002). Compactly Supported Correlation Functions. *Journal of Multivariate Analysis*, 83:493–508.

Goldberg, P. W., Williams, C. K. I., and Bishop, C. M. (1998). Regression with Input-dependent Noise. In M. I. Jordan, M. J. K. and Solla, S. A., editors, *Advances in Neural Information Processing Systems*, volume 10, pages 493–499. Lawrence Erlbaum.

Goutte, C. (1997). *Statistical learning and regularisation for regression. Application to system identification and time series modelling*. PhD thesis, Universite de Paris 6.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning. Data mining, Inference and Prediction*. Springer.

Henson, M. (1998). Nonlinear Model Predictive Control: Current Status and Future Directions. *Computers and Chemical Engineering*, 23:187–202.

Henson, M. and Seborg, D. (1994). Adaptive Nonlinear Control of a pH Neutralization Process. In *IEEE Trans. Control System Technology*, volume 2, pages 169–183.

Jaakola, T. S. and Jordan, M. I. (1999). Improving the mean field approximation via the use of mixture distributions. In *Learning in Graphical Models*, pages 163–173. MIT Press.

Judd, K. and Small, M. (2000). Towards long-term prediction. *Physica D*, 136(1-2):31–44.

Julier, S. and Uhlmann, J. K. (2000). A new extension of the Kalman filter to nonlinear systems. In *The Proceedings of AeroSense*, Orlando, Florida. The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls.

Kendall, M. and Stuart, A. (1958). *The advanced theory of statistics*, volume 2. C. Griffin & Co Ltd.

Kimeldorf, G. S. and Wahba, G. (1970). A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41(2):495–502.

Kocijan, J., Banko, B., Likar, B., Girard, A., Murray-Smith, R., and Rasmussen, C. E. (2003a). A case based comparison of identification with neural network and Gaussian process models. In *International Conference on Intelligent Control Systems and Signal Processing*, pages 137–142.

Kocijan, J., Girard, A., Banko, B., and Murray-Smith, R. (2003b). Dynamic systems identification with Gaussian processes. In *4th MATHMOD Vienna, 4th IMACS Symposium on Mathematical Modelling*, volume 2, pages 776–784.

Kocijan, J., Girard, A., Banko, B., and Murray-Smith, R. (2004a). Dynamic systems identification with Gaussian processes. *Journal of Mathematical and Computer Modelling of Dynamical Systems*. To appear.

Kocijan, J., Girard, A., and Leith, D. J. (2002). Identification of pH neutralization process with neural networks and Gaussian Process models. Technical Report DP-8575, Jozef Stefan Institute, Ljubljana.

Kocijan, J., Girard, A., and Leith, D. J. (2004b). Incorporating Linear Local Models in Gaussian Process Model. Technical Report DP-8895, Jozef Stefan Institute, Ljubljana.

Kocijan, J., Murray-Smith, R., Rasmussen, C. E., and Girard, A. (2004c). Gaussian process model based predictive control. In *American Control Conference, Boston*.

Krige, S. G. (1951). A statistical approach to some basic valuations problems on the Witwatersrand. *J. Chem. Metall. Min. Soc.*, 52(6):119–139.

Kuo, J. and Principe, J. (1994). Reconstructed Dynamics and Chaotic Signal Modeling. In *IEEE International Conference on Neural Networks*, volume 5, pages 3131–3136.

Lauritzen, S. L. (1999). Aspects of T. N. Thiele's contributions to statistics. In *Bulletin of the International Statistical Institute*, volume 58.

Leith, D. J., Murray-Smith, R., and Leithead, W. E. (2000). Nonlinear structure identification: A Gaussian Process prior/Velocity-based approach. In *Control 2000, Cambridge*.

Lemm, J. C. (1999). Mixtures of Gaussian Process Priors. Technical Report MS-TP1-99-5, Institut für Theoretische Physik, Universität Münster.

Lindley, D. V. (1969). *Introduction to Probability and Statistics from a Bayesian viewpoint*. Cambridge University Press.

Ljung, L. (1999). *System identification: Theory for the user*. Prentice Hall.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

MacKay, D. J. C. (1994). Bayesian methods for backpropagation networks. In Domany, E., van Hemmen, J. L., and Schulten, K., editors, *Models of Neural Networks III*, chapter 6, pages 211–254. Springer-Verlag, New York.

MacKay, D. J. C. (1995). Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505.

Mackay, D. J. C. (1997). Gaussian processes: A replacement for supervised neural networks? Technical report, Cavendish Laboratory, Cambridge University.

Mackay, D. J. C. (1999). Introduction to Monte Carlo methods. In Jordan, M., editor, *Learning In Graphical Models*, pages 175–204.

Mackay, D. J. C. (2003). *Information theory, Inference and Learning Algorithms*. Cambridge University Press.

Mackey, M. C. and Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, 197:287–289.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58:1246–1266.

Minka, T. P. (2001). Using lower bounds to approximate integrals. Available at http://www.stat.cmu.edu/∽minka/papers/rem.html.

Mukherjee, S., Osuna, E., and Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machines. In *NNSP'97*, Amelia Island, Florida. IEEE.

Müller, K., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., and Vapnik, V. (1997). Predicting time series with support vector machines. In *Proceedings of ICANN'97*, Lausanne.

Murray-Smith, R. (1998). Modelling human control behaviour with a Markov-chain switched bank of control laws. In *7th IFAC Symposium on Man-Machine Systems, Kyoto*, pages 521–526.

Murray-Smith, R. and Girard, A. (2001). Gaussian Process priors with ARMA noise models. In *Irish Signals and Systems Conference, Maynooth*, pages 147–152.

Murray-Smith, R. and Johansen, T. A. (1997). *Multiple model approaches to modelling and control*. Taylor&Francis.

Murray-Smith, R., Johansen, T. A., and Shorten, R. (1999). On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. In *European Control Conference, Karlsruhe, 1999*, pages BA–14.

Murray-Smith, R. and Pearlmutter, B. (2003). Transformations of Gaussian process priors. Technical Report TR-2003-149, University of Glasgow, Scotland, UK.

Murray-Smith, R. and Sbarbaro, D. (2002). Nonlinear adaptive control using non-parametric Gaussian process prior models. In *15th IFAC Triennial World Congress. International Federation of Automatic Control*.

Murray-Smith, R., Sbarbaro, D., Rasmussen, C. E., and Girard, A. (2003). Adaptive, Cautious, Predictive control with Gaussian Process priors. In *IFAC International Symposium on System Identification, Rotterdam*.

Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-92-1, Department of Computer Science, University of Toronto.

Neal, R. M. (1994). Priors for infinite networks. Technical Report CRG-TR-94-1, Department of Computer Science, University of Toronto.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

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.

Newman, P. and Leonard, J. (2002). A matrix orientated note on joint, marginal, and conditional multivariate Gaussian distributions. Technical report, MIT. http://www.robots.ox.ac.uk/∽pnewman/index.html.

O'Hagan, A. (1978). On curve fitting and optimal design for regression (with discussion). *Journal of the Royal Statistical Society B*, 40:1–42.

O'Hagan, A. (1994). *Bayesian Inference*. Kendall's Advanced Theory of Statistics. Edward Arnold.

Paciorek, C. J. 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*, volume 16. MIT Press. To appear.

Papoulis, A. (1991). *Probability, random variables, and stochastic processes*. McGraw-Hill.

Patterson, H. D. and Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. *Biometrika*, 58:545–554.

Phan, M. Q., Lim, R. K., and Longman, R. W. (1998). Unifying Input-Ouput and State-Space Perspectives of Predictive Control. Technical Report 3044, Department of Mechanical and Aerospace Engineering.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497.

Principe, J. and Kuo, J. (1995). Dynamic modeling of chaotic time series with neural networks. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 311–318. MIT Press.

Principe, J., Rathie, A., and Kuo, J. (1992a). *Nonlinear Dynamics of the Brain*, chapter Prediction of chaotic time series with neural networks, pages 250–258. World Scientific.

Principe, J. C., Rathie, A., and Kuo, J. M. (1992b). Prediction of chaotic time series with neural networks and the issue of dynamic modeling. *Int. J. of Bifurcation and Chaos*, 2(4):989–996.

Pugachev, V. S. (1967). *Theory of random functions and its application to control problems*. Pergamon Press.

Qazaz, C. S., Williams, C. K. I., and Bishop, C. M. (1996). An upper bound on the the Bayesian Error Bars for Generalized Linear Regression. Technical Report NCRG/96/005, Neural Computing Research Group, Aston University.

Qin, S. J. and Badgwell, T. A. (2000). *Nonlinear Model Predictive Control*, chapter An overview of nonlinear model predictive control applications, pages 369–392. Birkhauser Verlag.

Quinonero-Candela, J. and Girard, A. (2002). Prediction at an Uncertain Input for Gaussian Processes and Relevance Vector Machines – Application to Multiple-Step Ahead Time-Series Forecasting. Technical report, Informatics and Mathematical Modelling, Technical Univesity of Denmark.

Quinonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. E. (2003). Propagation of Uncertainty in Bayesian Kernels Models – Application to Multiple-Step Ahead Forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 701–4.

Ralaivola, L. and d'Alché Buc, F. (2003). Dynamical Modeling with Kernels for Nonlinear Time Series Prediction. In Thrun, S., Saul, L., , and Schölkopf, B., editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press. To appear.

Rasmussen, C. E. (1996a). *Evaluation of Gaussian Processes and other methods for non-linear regresion*. PhD thesis, University of Toronto.

Rasmussen, C. E. (1996b). A Practical Monte Carlo Implementation of Bayesian Learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 598–604. MIT Press.

Rasmussen, C. E. (2000). The infinite Gaussian Mixture Model. In S.A. Solla, T. L. and Mller, K.-R., editors, *Advances in Neural Information Processing Systems*, volume 12, pages 554–560. MIT Press.

Rasmussen, C. E. (2003). Gaussian processes to speed up Hybrid Monte Carlo for expensive Bayesian integrals. In Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M., and West, M., editors, *Bayesian Statistics 7*. Oxford University Press.

Rasmussen, C. E. and Ghahramani, Z. (2001). Occam's razor. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13, pages 294–300. MIT Press.

Rasmussen, C. E. and Ghahramani, Z. (2002). Infinite Mixtures of Gaussian Process Experts. In T. Dietterich, S. Becker, Z. G., editor, *Advances in Neural Information Processing Systems*, volume 14, pages 881–888. MIT Press.

Rasmussen, C. E. and Kuss, M. (2004). Gaussian Processes in Reinforcement Learning. In Thrun, S., Saul, L., , and Schölkopf, B., editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press. To appear.

Roweis, S. and Ghahramani, Z. (1997). A unifying review of libnear gaussian models. *Neural Computation*, 11(2):305–345.

Sbarbaro, D. and Murray-Smith, R. (2003). Self-tuning control of non-linear systems using Gaussian process prior models. Technical Report TR-2003-143, Department of Computing Science, University of Glasgow.

Seeger, M. (2003). *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh.

Seeger, M. (2004). Bayesian Gaussian Processes for Machine Learning. *International Journal of Neural Systems*. To appear.

Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In Bishop, C. M. and Frey, B. J., editors, *Proceedings of the Ninth International Workshop on AI and Statistics*. To appear.

Shi, J. Q., Murray-Smith, R., and Titterington, D. M. (2002). Bayesian Regression and Classification Using Mixtures of Gaussian Processes. Technical Report TR-2002-114, Department of Computing Science, University of Glasgow.

Small, M. and Judd, K. (1999). Variable prediction steps and long term prediction. Technical report, Department of Mathematics and Statistics, University of Western Australia.

Smith, L. A. (2000). Disentangling Uncertainty and Error: On the Predictability of Nonlinear Systems. In Mees, A. I., editor, *Nonlinear Dynamics and Statistics*, pages 31–64. Birkhauser.

Smith, L. A., Ziehmann, C., and Fraedrich, K. (1999). Uncertainty dynamics and predictability in chaotic systems. *Quart. J. R. Meteorol. Soc.*, 125:2855–2886.

Snoussi, H., Seghouane, A., Mohammad-Djafari, A., and Fleury, G. (2002). Learning in presence of input noise using the stochastic EM algorithm. In *Bayesian Inference and Maximum Entropy Methods, MaxEnt Workshops*, pages 135–149.

Söderström, T. and Stoica, P. (1989). *System Identification*. Prentice Hall, Englewood Cliffs, NJ.

Solak, E., Murray-Smith, R., Leithead, W. E., Leith, D. J., and Rasmussen, C. E. (2003). Derivative Observations in Gaussian Process Models of Dynamic Systems. In S. Becker, S. T. and

Obermayer, K., editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1033–1040. MIT Press.

Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D. and Young, L., editors, *Dynamical Systems and Turbulence*, volume 898, pages 366–381. Springer-Verlag.

Tarantola, A. and Valette, B. (1982). Inverse problems = Quest for Information. *Journal of Geohpysics*, 50:159–170.

Tipping, M. E. (2001). Sparse Bayesian learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244.

Titterington, D. M., Smith, A. F. M., and Makov, U. E. (1985). *Statistical Analysis of Finite Mixture Distributions*. John Wiley & Sons.

Tresp, V. (2001). Mixture of Gaussian Processes. In T. K. Leen, T. G. Diettrich, V. T., editor, *Advances in Neural Information Processing Systems*, volume 13, pages 654–660. MIT Press.

Tresp, V., Ahmad, S., and Neuneier, R. (1994). Training Neural Networks with Deficient Data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6, pages 128–135. Morgan Kaufmann Publishers, Inc.

Tresp, V. and Hofmann, R. (1995). Missing and Noisy Data in Nonlinear Time-Series Prediction. In S. F. Girosi, J. Mahoul, E. M. and Wilson, E., editors, *Neural Networks for Signal Processing*, volume 24 of *IEEE Signal Processing Society, New York*, pages 1–10.

Tresp, V. and Hofmann, R. (1998). Nonlinear Time-Series Prediction with Missing and Noisy Data. *Neural Computation*, 10(3):731–747.

Triantafyllopoulos, K. (2003). On the central moments of the multidimensional Gaussian distribution. *The Mathematical Scientist*, 28:125–128.

Vivarelli, F. and Williams, C. K. I. (1999). Discovering Hidden Features with Gaussian Processes Regression. In M. J. Kearns, S. A. Solla, D. A. C., editor, *Advances in Neural Information Processing Systems*, volume 11, pages 613–619. MIT Press.

Von Mises, R. (1964). *Mathematical Theory of Probability and Statistics*, chapter VIII, pages 416–429. Academic Press.

Wahba, G. (1992). Multivariate function and operator estimation, based on smoothing splines and reproducing kernels. In Casdagli, M. and Eubank, S., editors, *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity*, volume XII. Addison-Wesley.

Welch, G. and Bishop, G. (1995). An introduction to the Kalman Filter. Technical Report TR 95-041, University of North Carolina.

West, M. and Harrison, J. (1997). *Bayesian forecasting and dynamic models*. Springer.

Williams, C. K. I. (1997a). Computation with infinite neural networks. Technical Report NCRG-97-025, Department of Computer Science and Applied Mathematics, Aston University.

Williams, C. K. I. (1997b). Computing with Infinite Networks. In M. C. Mpzer, M. I. Jordan, T. P., editor, *Advances in Neural Information Processing Systems*, volume 9, pages 295–301. MIT Press.

Williams, C. K. I. (1997c). Prediction with Gaussian Processes: From linear regression to linear prediction and beyond. Technical Report NCRG-97-012, Dept of Computer Science and Applied Mathematics. Aston University.

Williams, C. K. I. (2002). Gaussian Processes. To appear in The handbook of Brain Theory and Neural Networks, Second edition, MIT Press.

Williams, C. K. I., Qazaz, C. S., Bishop, C. M., and Zhu, H. (1995). On the relationship between Bayesian error bars and the input data density. In *Proceedings Fourth IEE International Conference on Artificial Neural Networks*, pages 160–165.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian Processes for Regression. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 514–520. MIT Press.

Williams, C. K. I. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In T. K. Leen, T. G. Diettrich, V. T., editor, *Advances in Neural Information Processing Systems*, volume 13, pages 682–688. MIT Press.

Wittenmark, B. (1995). Adaptive dual control methods: An overview. In *Preprints of 5th IFAC symposium on Adaptive Systems in Control and Signal processing, Hungary*, pages 67–72.

Wright, W. A. (1999). Bayesian approach to neural-network modelling with input uncertainty. In *IEEE Transactions on Neural Networks*, volume 10, pages 1261–1270.