Gaussian Process priors with Uncertain Inputs: Multiple-Step-Ahead Prediction*

Agathe Girard
Dept. of Computing Science
University of Glasgow
Glasgow, UK
agathe@dcs.gla.ac.uk

Carl Edward Rasmussen
Gatsby Unit
University College London
London, UK
edward@gatsby.ucl.ac.uk

Roderick Murray-Smith
Dept. of Computing Science
University of Glasgow
& Hamilton Institute
National University of Ireland
Maynooth, Ireland
rod@dcs.gla.ac.uk

Abstract

We consider the problem of multi-step ahead prediction in time series analysis using the non-parametric Gaussian process model. k-step ahead forecasting of a discrete-time non-linear dynamic system can be performed by doing repeated one-step ahead predictions. For a state-space model of the form $y_t = f(y_{t-1}, \ldots, y_{t-L})$, the prediction of y at time $t+k$ is based on the estimates $\hat{y}_{t+k-1}, \ldots, \hat{y}_{t+k-L}$ of the previous outputs. We show how, using an analytical Gaussian approximation, we can formally incorporate the uncertainty about intermediate regressor values, thus updating the uncertainty on the current prediction. In this framework, the problem is that of predicting responses at a random input and we compare the Gaussian approximation to the Monte-Carlo numerical approximation of the predictive distribution. The approach is illustrated on a simulated non-linear dynamic example, as well as on a simple one-dimensional static example.

*Technical Report TR-2002-119, Department of Computing Science, University of Glasgow, October, 2002.

1 Introduction

One of the main objectives in time series analysis is forecasting and in many real life problems, one has to predict ahead in time, up to a certain time horizon (sometimes called *lead* time or prediction horizon). Furthermore, knowledge of the uncertainty of the prediction is important. Currently, the multiple-step ahead prediction task of a discrete-time non-linear dynamic system is achieved by either explicitly training a *direct* model to predict k steps ahead, or by doing repeated one-step ahead predictions up to the desired horizon, which we call the *iterative method*.

There are a number of reasons why the iterative method might be preferred to the 'direct' one. Firstly, the direct method makes predictions for a fixed horizon only, making it computationally demanding if one is interested in different horizons. Furthermore, the larger k, the more training data we need in order to achieve a good predictive performance, because of the larger number of 'missing' data between t and $t + k$. On the other hand, the iterated method provides any k-step ahead forecast, up to the desired horizon, as well as the joint probability distribution of the intermediate points.

In the Gaussian process modelling approach, one computes predictive distributions whose means serve as output estimates. (O'Hagan, 1978) was one of the first to introduce the Gaussian process (GP) for regression but it really started being a popular non-parametric modelling approach after the publication of (Neal, 1995). In (Rasmussen, 1996), it is shown that GPs can achieve a predictive performance comparable to (if not better than) other modelling approaches like neural networks or local learning methods. We will show that for a k-step ahead prediction which ignores the accumulating prediction variance, the model is not conservative enough, with unrealistically small uncertainty attached to the forecast. An alternative solution is presented for iterative k-step ahead prediction, with propagation of the prediction uncertainty.

This report is organised as follows. First, we recall the main equations used in Gaussian Process modelling. Then, we derive the expressions of the predictive mean and variance when predicting at an uncertain input and show how we can use these results for the iterative multiple-step ahead forecasting of time-series. We illustrate the approach on static and dynamic examples and we finish with some conclusions.

2 Modelling with Gaussian Processes

For a comprehensive introduction to Gaussian Process modelling, please refer to (Mackay, 1997), (Williams and Rasmussen, 1996), or the more recent review (Williams, 2002).

2.1 The GP prior model

Formally, the random function or stochastic process $f(x)$ is a Gaussian process with mean $m(x)$ and covariance function $C(x^p, x^q)$, if its values at a finite number of points $f(x^1), \dots, f(x^n)$ are seen as the components of a random vector normally distributed. That is, for each n:

$$f(x^1), \dots, f(x^n) \sim \mathcal{N}(0, \Sigma) , \tag{1}$$

where Σ is an $n \times n$ covariance matrix whose entries give the covariances between each pair of points and which is a function of the corresponding inputs, $\Sigma_{pq} = \mathrm{Cov}(f(x^p), f(x^q)) = C(x^p, x^q)$.

2.1.1 Role of the covariance function

The choice of the covariance function is of great importance and, as much as possible, should reflect one's prior knowledge or assumption about the underlying function (e.g. smoothness, continuity assumptions). Here, as a special case, we assume that the process is stationary: the mean is constant (chosen to be zero) and the covariance function only depends on the distance between the inputs x. A common choice is

$$C(x^p, x^q) = v_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d^p - x_d^q)^2}{w_d^2}\right], \tag{2}$$

where D is the input dimension. Figure 1 shows an example (in 1-D) of the covariance matrix corresponding to this function. v_1 and w_d are hyperparameters of the covariance function.

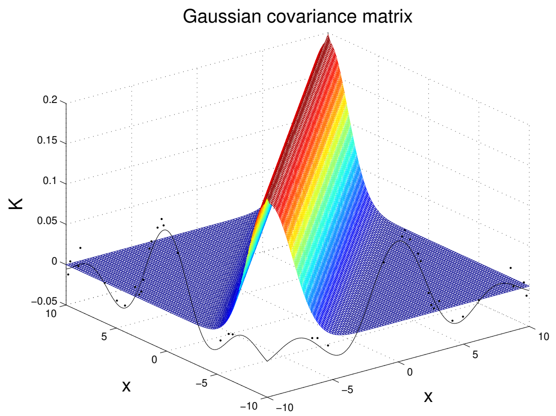

Figure 1: 1-D example of a data covariance matrix $K = \Sigma + v_0 I$ where Σ has been computed using the Gaussian covariance function given by (2) and v_0 is the white noise variance. The matrix plotted is 100×100, for $x \in [-10, 10]$, but the learning was based on $N = 20$ data points (black dots, here divided by a factor of 10 so as to be at the same scale as the covariance). The corresponding hyperparameters are $v_1 = 0.1906$, $w = 1.9288$, and $v_0 = 0.0073$ (found by Maximum Likelihood, see section 2.2).

This particular choice of Gaussian (squared exponential) covariance function corresponds to a prior assumption that the underlying function f is *smooth* and *continuous* (figure 2 shows samples from a GP with such a covariance function). It accounts for a high correlation between the outputs of cases with nearby inputs. The parameter v_1 gives the overall scale of correlations and the w parameters allow a different distance measure for each input dimension d (correlation length in direction d). For a given problem, these parameters are adjusted to the data at hand and, for irrelevant inputs, the corresponding w_d will tend to zero.[1]

It can be noted that the covariance function in the GP framework is very similar to the *kernel* used in the Support Vector Machines community. In theory, the only restriction on the choice of covariance function is that it has to generate a non-negative definite covariance matrix (more discussion about alternative choices of covariance functions can be found in (Mackay, 1997)).

[1] Automatic Relevance Determination idea developed by MacKay and Neal (Neal, 1995)

2.2 Gaussian Process for regression

We assume a statistical model $y = f(x) + \epsilon$ with an additive uncorrelated Gaussian white noise with variance v_0, $\epsilon \sim \mathcal{N}(0, v_0)$. Given a set of N data pairs $\mathcal{D} = \{y^i, x^i\}_{i=1}^N$ and a GP prior on $f(x)$, with zero-mean and Gaussian covariance function such as (2), our aim is to get the predictive distribution of the function value $f(x^*)$ at a new (given) input x^*.

2.2.1 Learning by Maximum Likelihood

The likelihood of the data is simply

$$\mathbf{y} \sim \mathcal{N}(0, K) \; , \tag{3}$$

where $\mathbf{y}$ is the $N \times 1$ vector of targets and K is the $N \times N$ 'data covariance matrix', such that $K_{pq} = \Sigma_{pq} + v_0 \delta_{pq}$ where δ_{pq} is non-zero only when $p = q$.

In a Maximum Likelihood framework, we then adjust the vector of hyperparameters $\Theta = [w_1 \dots w_D \; v_1 \; v_0]^T$ so as to maximise the log-likelihood

$$\mathcal{L}(\Theta) = \log p(\mathbf{y}) = -\frac{1}{2} \log |K| - \frac{1}{2} \mathbf{y}^T K^{-1} \mathbf{y} - \frac{1}{2} N \log(2\pi) \; , \tag{4}$$

which requires the calculation of the derivative of $\mathcal{L}(\Theta)$ with respect to each hyperparameter Θ_j, given by

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta_j} = -\frac{1}{2} \mathrm{Tr} \left[K^{-1} \frac{\partial K}{\partial \Theta_j} \right] + \frac{1}{2} \mathbf{y}^T K^{-1} \frac{\partial K}{\partial \Theta_j} K^{-1} \mathbf{y} \; , \tag{5}$$

where Tr denotes the trace.

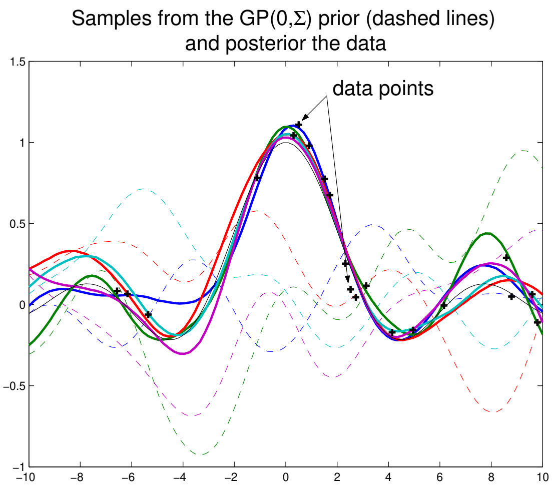

Figure 2: Realisations from a GP with 0-mean and training covariance matrix $K = \Sigma + v_0 I$ (see figure 1). Dashed lines: Samples from the *a priori* GP (not taking account of the data). Solid lines: Samples from the posterior, i.e. conditioned on the training points. Also shown (black), the true function $f(x) = \sin x / x$ with training data (true noise level of 0.01).

2.2.2 Prediction at x^*

For a given x^*, the predictive distribution of $f(x^*)$, or equivalently of y^*, is simply obtained by conditioning on the training data to obtain $p(f(x^*)|x^*, \mathcal{D})$.

The joint distribution of the variables being Gaussian, this conditional distribution is also Gaussian with mean and variance

$$\mu(x^*) = \mathbf{k}(x^*)^T K^{-1}\mathbf{y} \tag{6}$$
$$\sigma^2(x^*) = k(x^*) - \mathbf{k}(x^*)^T K^{-1}\mathbf{k}(x^*) \tag{7}$$

where $\mathbf{k}(x^*) = [C(x^1, x^*), \ldots, C(x^N, x^*)]^T$ is the $N \times 1$ vector of covariances between the new point and the training targets and $k(x^*) = C(x^*, x^*)$.

The predicted mean $\mu(x^*)$ serves as an estimate of the function output, $\hat{f}(x^*)$, with uncertainty $\sigma(x^*)$. It is also a point estimate for the corresponding noisy target $\hat{y}^*$, with variance $\sigma^2(x^*) + v_0$. Figure 3 shows the predictions at four new inputs. We see that the prediction at x^* far away from the data, or near the edges, leads to a predicted point with a variance larger than that at x^* nearby the training inputs. This is because $\mathbf{k}(x^*)$ decreases as the distance between x^* and the training points increases (see figure 9), thus decreasing the $\mathbf{k}(x^*)^T K^{-1}\mathbf{k}(x^*)$ term and therefore increasing the variance.

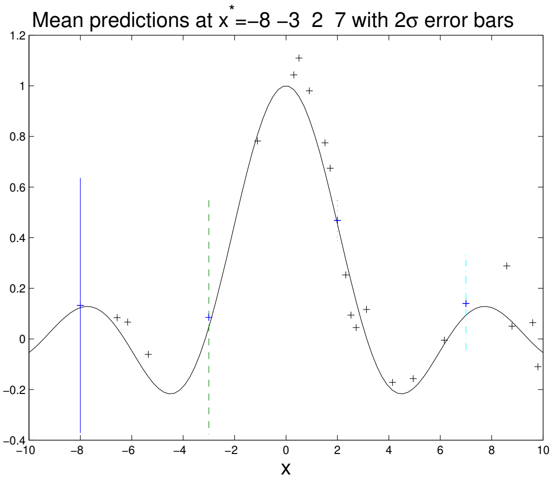

Figure 3: Mean predictions at $x^* = -8, -3, 2, 7$ with their 2σ error-bars, computed using equations (6) and (7) resp. For x^* far away from the data, or near the edges, the corresponding uncertainty is large. Also plotted, the true function and the 20 training points (black crosses).

It is worth noting that the predictive mean $\mu(x^*)$ can be seen as a weighted sum of the training data

$$\mu(x^*) = \mathbf{s}^T\mathbf{y}\,, \quad \text{with} \quad \mathbf{s}^T = \mathbf{k}(x^*)^T K^{-1} \tag{8}$$

where $\mathbf{s}$ is called the *smoothing* or *effective* kernel.

Alternatively, one can see it as a linear combination of the covariance between the new x^* and the training points (Williams, 2002):

$$\mu(x^*) = \mathbf{k}(x^*)^T\alpha\,, \quad \text{with} \quad \alpha = K^{-1}\mathbf{y}\,. \tag{9}$$

Either the smoothing kernels or the alpha-coefficients depend on the density of the points, through the inverse of the covariance matrix. Figure 4 shows the smoothing kernels and alpha-coefficients corresponding to the case when there are only 20 training points and when the number of training points is large ($N = 100$). For a large number of training points, the value of the alpha-coefficients is higher than for a few points. On the contrary, the amplitude of the smoothing kernels is smaller.

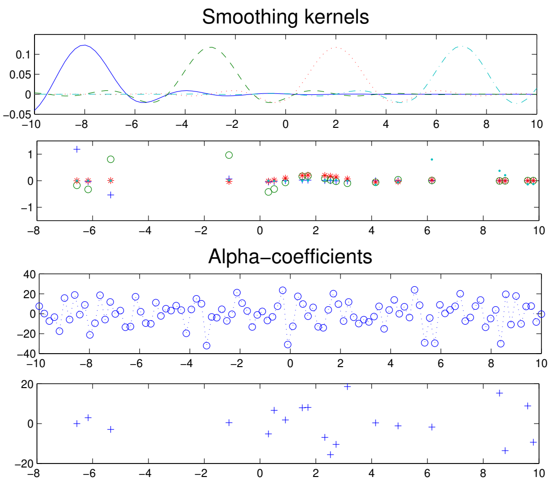

Figure 4: First two plots: Smoothing kernels $\mathbf{s}^T = \mathbf{k}(x^*)^T K^{-1}$ for a large (top) and a small (bottom) number of training points (N). The solid line and crosses correspond to $x^* = -8$, for $N = 100$ and $N = 20$ resp., the dashed line and circles to $x^* = -3$, the dotted line and asterisks to $x^* = 2$ and the dash-dot line and points to $x^* = 7$. The last two plots show the alpha-coefficients $\alpha = K^{-1}\mathbf{y}$ for $N = 100$ and $N = 20$ (the points are not joined when $N = 20$ due to the sparse nature of the training set).

3 Prediction at a random input

If we now wish to predict the distribution of the function value, $p(f(x^*))$, at the random variable x^*, with $x^* \sim \mathcal{N}(\mu_{x^*}, \Sigma_{x^*})$, the predictive distribution is obtained by integrating over the input distribution

$$p(f(x^*)|\mu_{x^*}, \Sigma_{x^*}, \mathcal{D})) = \int p(f(x^*)|x^*, \mathcal{D}))p(x^*)dx^* \,, \tag{10}$$

where $p(f(x^*)|x^*, \mathcal{D})) = \frac{1}{\sigma(x^*)\sqrt{2\pi}} \exp\left[-\frac{1}{2}\frac{(f(x^*)-\mu(x^*))^2}{\sigma^2(x^*)}\right]$ with mean $\mu(x^*)$ and variance $\sigma^2(x^*)$ as given by equations (6) and (7) respectively.

3.1 Numerical approximation

Given that the integral (10) is analytically intractable ($p(f(x^*)|x^*)$ is a complicated function of x^*), one possibility is to perform a numerical approximation of the integral by a simple Monte-Carlo approach:

$$p(f(x^*)|\mu_{x^*}, \Sigma_{x^*}, \mathcal{D})) = \int p(f(x^*)|x^*, \mathcal{D}))p(x^*)dx^* \simeq \frac{1}{T}\sum_{t=1}^{T} p(f(x^*)|x^{*t}, \mathcal{D})) \,, \tag{11}$$

where x^{*t} are (independent) samples from $p(x^*)$, which are, in this Gaussian case, readily obtained.

Figure 5 illustrates the Monte-Carlo approximation for the $1D$ static case. We can also use this approach to assess the goodness of the Gaussian approximation, i.e. if the error-bars of the Gaussian approximation encompass all the samples up to step k, we can conclude that the approximation is valid up to that step. Figure 6 shows the histogram of the mean predictions corresponding to 100 samples from $p(x^*)$, for $\Sigma_{x^*} = 1$ (left) and $\Sigma_{x^*} = 0.5$ (right).

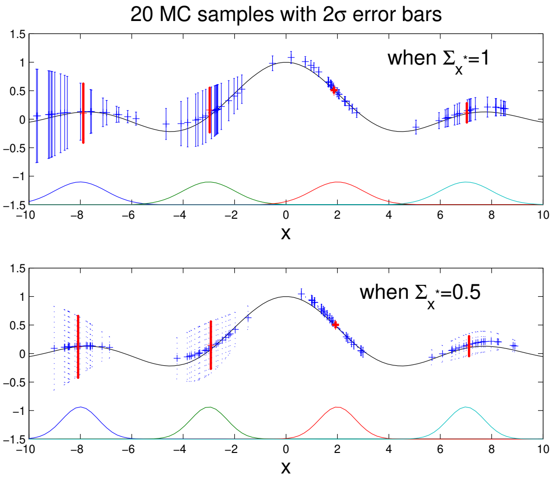

Figure 5: Monte-Carlo approximation when predicting at $x^* \sim \mathcal{N}(\mu_{x^*}, \Sigma_{x^*})$, for $\mu_{x^*} = -8, -3, 2, 7$, $\Sigma_{x^*} = 1$ (top) and $\Sigma_{x^*} = 0.5$ (bottom). For 20 samples x^{*t} from the different $p(x^*)$, we plot the predicted means with their 2σ error-bar (dotted lines). Also plotted, the sample-mean and sample-(2)standard deviation at the mean of the sampled x^{*t} (thick solide lines).

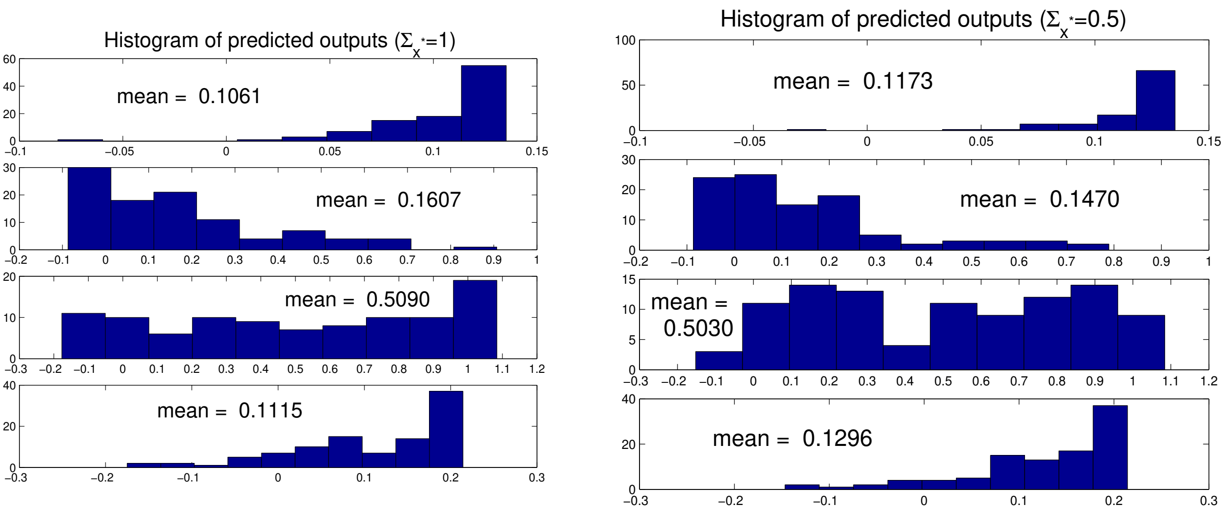

Figure 6: Histograms of predicted means corresponding to 100 samples from $x^* \sim \mathcal{N}(\mu_{x^*}, \Sigma_{x^*})$, for $\mu_{x^*} = -8, -3, 2, 7$, (top to bottom), $\Sigma_{x^*} = 1$ (left) and $\Sigma_{x^*} = 0.5$ (right). Also given, the value of the sample-mean.

3.2 Gaussian approximation

The analytical Gaussian approximation consists in only computing the mean and variance of $f(x^*)|\mu_{x^*}, \Sigma_{x^*}, \mathcal{D})$. They are obtained using respectively the law of iterated expectations and law of conditional variances:

$$m(x^*) = E_{x^*}[E_{f(x^*)}[f(x^*)|x^*]] = E_{x^*}[\mu(x^*)] \quad (12)$$

$$\begin{aligned} v(x^*) &= E_{x^*}[\mathrm{var}_{f(x^*)}(f(x^*)|x^*)] + \mathrm{var}_{x^*}(E_{f(x^*)}[f(x^*)|x^*]) \\ &= E_{x^*}[\sigma^2(x^*)] + \mathrm{var}_{x^*}(\mu(x^*)) \end{aligned} \quad (13)$$

where E_{x^*} indicates the expectation under x^*.

Now, $\mu(x^*)$ and $\sigma^2(x^*)$, as given by equations (6) and (7), are functions of the random argument x^* and we need further approximations to be able to compute these moments.

3.3 Predictive mean $m(x^*)$

We approximate $\mu(x^*)$ by its first order Taylor expansion around μ_{x^*}:

$$\mu(x^*) = \mu(\mu_{x^*}) + \left.\frac{\partial \mu(x^*)}{\partial x^*}\right|_{x^*=\mu_{x^*}}^T (x^* - \mu_{x^*}) + O(||x^* - \mu_{x^*}||^2) \quad (14)$$

with $\mu(\mu_{x^*}) = \mathbf{k}(\mu_{x^*})^T K^{-1}\mathbf{y}$ and where $||x^* - \mu_{x^*}||^2 = (x^* - \mu_{x^*})^T(x^* - \mu_{x^*})$.

Then, according to equation (12), we have

$$E_{x^*}[\mu(x^*)] \approx E_{x^*}\left[\mu(\mu_{x^*}) + \left.\frac{\partial \mu(x^*)}{\partial x^*}\right|_{x^*=\mu_{x^*}}^T (x^* - \mu_{x^*})\right] = \mu(\mu_{x^*}) \, . \quad (15)$$

Therefore, we see that, within a first order Taylor expansion, the mean prediction at a random x^* does not provide any correction over the zeroth order, we have

$$m(x^*) = \mathbf{k}(\mu_{x^*})^T K^{-1}\mathbf{y} \, . \quad (16)$$

Of course, the goodness of the first order approximation depends on the curvature of $\mu(x^*)$, as well as on the variance of x^*: in one dimension, it can easily be seen geometrically that if $\mu(x^*)$ has high curvature at μ_{x^*}, this approximation will not be valid, unless Σ_{x^*} is very small (see figure 7). The test inputs were chosen so as to illustrate different features: small function gradient and high density of training points at $x = 7$, small function gradient but near the edge (few training points) at $x = -8$, high function gradient and high density of training points at $x = 2$ and at last high function gradient with few training points for $x = -3$.

3.4 Predictive variance $v(x^*)$

To compute the variance, we first need to calculate $E_{x^*}[\sigma^2(x^*)]$. For this, we need to approximate $\sigma^2(x^*)$ and a natural choice is its second order Taylor expansion around μ_{x^*}:

$$\begin{aligned} \sigma^2(x^*) &= \sigma^2(\mu_{x^*}) + \left.\frac{\partial \sigma^2(x^*)}{\partial x^*}\right|_{x^*=\mu_{x^*}}^T (x^* - \mu_{x^*}) \\ &+ \frac{1}{2}(x^* - \mu_{x^*})^T \left.\frac{\partial^2 \sigma^2(x^*)}{\partial x^* \partial x^{*T}}\right|_{x^*=\mu_{x^*}} (x^* - \mu_{x^*}) + O(||x^* - \mu_{x^*}||^3) \end{aligned} \quad (17)$$

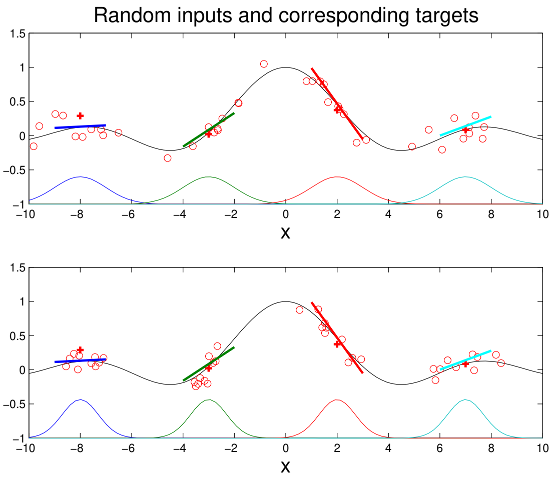

Figure 7: Function outputs (dots) and targets (crosses) corresponding to 10 different inputs taken from $x^* \sim \mathcal{N}(\mu_{x^*}, \Sigma_{x^*})$, for $\mu_{x^*} = -8, -3, 2, 7$ and $\Sigma_{x^*} = 1$ (top) and 0.5 (bottom). Also plotted, the gradients at the different μ_{x^*}. If Σ_{x^*} is large and the gradient is steep, targets from $p(x^*)$ will be quite far from the target at μ_{x^*}.

with $\sigma^2(\mu_{x^*}) = k(\mu_{x^*}) - \mathbf{k}(\mu_{x^*})^T K^{-1}\mathbf{k}(\mu_{x^*})$. Thus, we have

$$E_{x^*}[\sigma^2(x^*)] \approx \sigma^2(\mu_{x^*}) + E_{x^*}\left[\frac{1}{2}(x^* - \mu_{x^*})^T \left.\frac{\partial^2\sigma^2(x^*)}{\partial x^*\partial x^{*T}}\right|_{x^*=\mu_{x^*}} (x^* - \mu_{x^*})\right] , \tag{18}$$

and using the formula giving the expectation of a quadratic form under a Gaussian,[2] we arrive at

$$E_{x^*}[\sigma^2(x^*)] \approx \sigma^2(\mu_{x^*}) + \frac{1}{2}\mathrm{Tr}\left\{\left.\frac{\partial^2\sigma^2(x^*)}{\partial x^*\partial x^{*T}}\right|_{x^*=\mu_{x^*}} \Sigma_{x^*}\right\} . \tag{19}$$

Furthermore, using (14), we have

$$\begin{aligned}\mathrm{var}_{x^*}(\mu(x^*)) &\approx \mathrm{var}_{x^*}\left(\mu(\mu_{x^*}) + \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|^T_{x^*=\mu_{x^*}}(x^* - \mu_{x^*})\right) \\ &= \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|^T_{x^*=\mu_{x^*}} \Sigma_{x^*} \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|_{x^*=\mu_{x^*}} .\end{aligned} \tag{20}$$

This leads to

$$v(x^*) = \sigma^2(\mu_{x^*}) + \frac{1}{2}\mathrm{Tr}\left\{\left.\frac{\partial^2\sigma^2(x^*)}{\partial x^*\partial x^{*T}}\right|_{x^*=\mu_{x^*}} \Sigma_{x^*}\right\} + \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|^T_{x^*=\mu_{x^*}} \Sigma_{x^*} \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|_{x^*=\mu_{x^*}} \tag{21}$$

which we can also write

$$v(x^*) = \sigma^2(\mu_{x^*}) + \mathrm{Tr}\left\{\Sigma_{x^*}\left(\frac{1}{2}\left.\frac{\partial^2\sigma^2(x^*)}{\partial x^*\partial x^{*T}}\right|_{x^*=\mu_{x^*}} + \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|_{x^*=\mu_{x^*}} \left.\frac{\partial\mu(x^*)}{\partial x^*}\right|^T_{x^*=\mu_{x^*}}\right)\right\} \tag{22}$$

[2]
$$\int (x-\mu)^T\Sigma^{-1}(x-\mu)\mathcal{N}(m,S)dx = (\mu-m)^T\Sigma^{-1}(\mu-m) + \mathrm{Tr}[\Sigma^{-1}S]$$

with

$$
\begin{aligned}
\frac{\partial \mu(x^*)}{\partial x_d^*} &= \frac{\partial \mathbf{k}(x^*)^T}{\partial x_d^*} K^{-1}\,\mathbf{y} && (23)\\
\frac{\partial^2 \sigma^2(x^*)}{\partial x_d^* \partial x_e^*} &= \frac{\partial^2 k(x^*)}{\partial x_d^* \partial x_e^*} - 2\frac{\partial \mathbf{k}(x^*)^T}{\partial x_d^*} K^{-1} \frac{\partial \mathbf{k}(x^*)}{\partial x_e^*} - 2\frac{\partial^2 \mathbf{k}(x^*)^T}{\partial x_d^* \partial x_e^*} K^{-1}\mathbf{k}(x^*) && (24)
\end{aligned}
$$

for $d, e = 1 \dots D$ (input dimension) and partial derivatives evaluated at $x^* = \mu_{x^*}$.

Compared to the predicted variance obtained in the 'non-random' input case, we now have a correction term of the zeroth order which involves the computation of the first and second derivatives of the covariance function (2).

Figure 8 illustrates the Gaussian approximation when predicting at a random x^*.

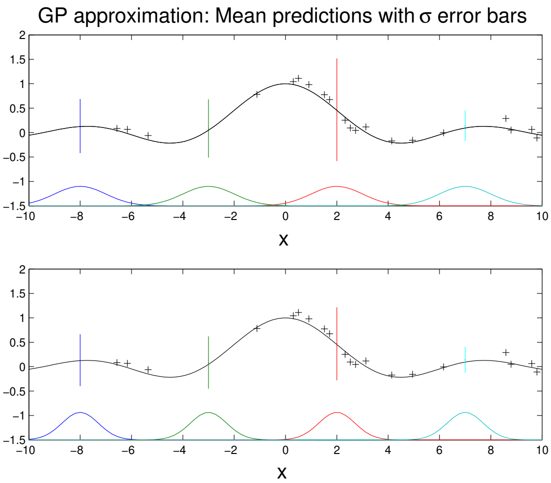

Figure 8: Gaussian approximation when predicting at $x^* \sim \mathcal{N}(\mu_{x^*}, \Sigma_{x^*})$, for $\mu_{x^*} = -8, -3, 2, 7$, $\Sigma_{x^*} = 1$ (top) and $\Sigma_{x^*} = 0.5$ (bottom). Plot of the predicted means with their 2σ error-bars.

3.4.1 Computing the derivatives

We need to compute the first and second derivatives of the covariance between the new points and the training points, with respect to changes in the components of those points. We have

$$
k^i(x^*) = C(x^i, x^*) = v_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D} w_d (x_d^i - x_d^*)^2\right] \quad (25)
$$

where $k^i(x^*)$ is the i^{th} component of $\mathbf{k}(x^*)$. The first and second derivatives evaluated at $x^* = \mu_{x^*}$ are then given by

$$
\left\{
\begin{aligned}
\left.\frac{\partial k^i(x^*)}{\partial x_d^*}\right|_{x^*=\mu_{x^*}} &= w_d(x_d^i - \mu_{x^*d})k^i(\mu_{x^*}) \\
\left.\frac{\partial^2 k^i(x^*)}{\partial x_d^* \partial x_e^*}\right|_{x^*=\mu_{x^*}} &= w_d[-\delta_{de} + (x_d^i - \mu_{x^*d})w_e(x_e^i - \mu_{x^*e})]k^i(\mu_{x^*})\,.
\end{aligned}
\right. \quad (26)
$$

Figure 9 shows $\mathbf{k}$, $d\mathbf{k}/dx$ and $d^2\mathbf{k}/dx^2$ in the one dimensional case. It is straightforward that we have $\frac{\partial k(x^*)}{\partial x_d^*} = \frac{\partial^2 k(x^*)}{\partial x_d^* \partial x_e^*} = 0$ since $k(x^*) = C(x^*, x^*) = v_1$.

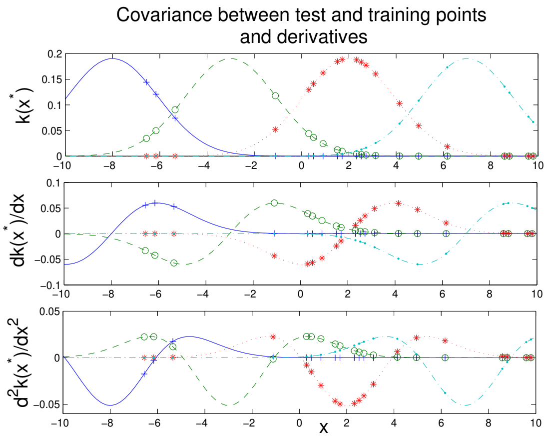

Figure 9: Covariance between test (x^*) and training points (top) with first (middle) and second (bottom) derivatives. The functions are plotted for 100 training points and also marked at the 20 training points previously used. The solid lines and crosses correspond to $x^* = -8$, dashed lines and circles to $x^* = -3$, dotted lines and asterisks to $x^* = 2$ and finally dash-dot lines and points to $x^* = 7$. Close to the training points, the term $\partial \mathbf{k}(x^*)/\partial x^*$ is close to zero, implying a very small contribution to the variance from the term $\partial \mu(x^*)/\partial x^*$.

Replacing in (23) we obtain

$$\left.\frac{\partial \mu(x^*)}{\partial x_d^*}\right|_{x^*=\mu_{x^*}} = [w_d(\mathbf{x}_d - \mu_{x^*d}).\mathbf{k}(\mu_{x^*})]^T K^{-1}\,\mathbf{y}\,, \tag{27}$$

for the derivative of the mean, and in (24), we get for the second derivative of the variance

$$\begin{aligned}
\left.\frac{\partial^2 \sigma^2(x^*)}{\partial x_d^* \partial x_e^*}\right|_{x^*=\mu_{x^*}} &= -2w_d w_e\{[(\mathbf{x}_d - \mu_{x^*d}).\mathbf{k}(\mu_{x^*})]^T K^{-1}[(\mathbf{x}_e - \mu_{x^*e}).\mathbf{k}(\mu_{x^*})] \\
&\quad +[(\mathbf{x}_d - \mu_{x^*d}).(\mathbf{x}_e - \mu_{x^*e}).\mathbf{k}(\mu_{x^*})]^T K^{-1}\mathbf{k}(\mu_{x^*})\} \qquad (28)\\
&\quad +2w_d\mathbf{k}(\mu_{x^*})^T K^{-1}\mathbf{k}(\mu_{x^*})\delta_{de} \qquad (29)
\end{aligned}$$

where $\mathbf{x}_d$ is the $N \times 1$ vector of input data in the d^{th} dimension and $a.b$ denotes the componentwise product of vectors a and b. Note that

$$2w_d\mathbf{k}(\mu_{x^*})^T K^{-1}\mathbf{k}(\mu_{x^*})\delta_{de} = 2w_d(k(\mu_{x^*}) - \sigma^2(\mu_{x^*}))\delta_{de}\text{with}k(\mu_{x^*}) = v_1. \tag{30}$$

4 Application to the iterative k-step ahead prediction of time series

We wish to apply these results to the multiple-step ahead prediction task of time series. Currently, this can be achieved by either training the model to learn how to predict k steps ahead (direct method) or by making repeated one-step ahead predictions (iterative method). In what follows, we are concerned with the iterative approach and suggest to propagate the uncertainty as we predict ahead in time.

4.1 "Naive" iterative k-step ahead prediction

Consider the time series $y^{t_1}, \dots, y^t$ and the state-space model

$$\begin{cases} x^{t_i} = [y^{t_i-1}, \dots, y^{t_i-L}]^T \\ y^{t_i} = f(x^{t_i}) + \epsilon^{t_i} \end{cases} \tag{31}$$

where the *state* x at time t_i is composed of previous outputs, up to a given lag[3] L and the (white) noise has variance v_0.

The iterative k-step ahead prediction method works as follows: it predicts only one time step ahead, using the estimate of the output of the current prediction, as well as previous outputs (up to the lag L), as the input to the prediction of the next time step, until the prediction k steps ahead is made.

Using the model (31) and assuming the data is known up to, say, time step t, the prediction of y at $t+k$ is computed via

$$\begin{array}{rcl} x^{t+1} = [y^t, y^{t-1}, \dots, y^{t+1-L}]^T & \rightarrow & f(x^{t+1}) \sim \mathcal{N}(\mu(x^{t+1}), \sigma^2(x^{t+1})) \\ & & \hat{y}^{t+1} = \mu(x^{t+1}) \\ x^{t+2} = [\hat{y}^{t+1}, y^t, \dots, y^{t+2-L}]^T & \rightarrow & f(x^{t+2}) \sim \mathcal{N}(\mu(x^{t+2}), \sigma^2(x^{t+2})) \\ & & \hat{y}^{t+2} = \mu(x^{t+2}) \\ & \vdots & \\ x^{t+k} = [\hat{y}^{t+k-1}, \hat{y}^{t+k-2}, \dots, \hat{y}^{t+k-L}]^T & \rightarrow & f(x^{t+k}) \sim \mathcal{N}(\mu(x^{t+k}), \sigma^2(x^{t+k})) \\ & & \hat{y}^{t+k} = \mu(x^{t+k}) \end{array}$$

where the point estimates $\mu(x^{t+k-i})$ are computed using equation (6). This setup does not account for the uncertainty induced by each successive prediction (variance $\sigma^2(x^{t+k-i}) + v_0$ associated to each $\hat{y}$, given by (7)).

4.2 Propagating the uncertainty

Using the results derived in the previous section, we propose to formally incorporate the uncertainty information about the future regressor. That is, as we predict ahead in time, we now view the lagged outputs as random variables.

In this framework, if, as before, data are known up to time t and we wish to predict k steps ahead, we now have

- at $t+1$,

$$x^{t+1} \sim \mathcal{N}\left(\begin{bmatrix} y^t \\ \dots \\ y^{t+1-L} \end{bmatrix}, \begin{bmatrix} 0 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & 0 \end{bmatrix} \right)$$

  predict $y^{t+1} \sim \mathcal{N}(m(x^{t+1}), v(x^{t+1}) + v_0)$, using (16) and (22), with $x^* = x^{t+1}$

- at $t+2$,

$$x^{t+2} \sim \mathcal{N}\left(\begin{bmatrix} m(x^{t+1}) \\ \dots \\ y^{t+2-L} \end{bmatrix}, \begin{bmatrix} v(x^{t+1}) + v_0 & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & 0 \end{bmatrix} \right)$$

  predict $y^{t+2} \sim \mathcal{N}(m(x^{t+2}), v(x^{t+2}) + v_0)$

$$\vdots$$

[3] In this report, we are not concerned with the identification of the lag and assume it has a known, fixed value.

- at $t+k$,

$$x^{t+k} \sim \mathcal{N}\left(\begin{bmatrix} m(x^{t+k-1}) \\ \dots \\ m(x^{t+k-L}) \end{bmatrix}, \begin{bmatrix} v(x^{t+k-1}) + v_0 & \dots & \mathrm{cov}(y^{t+k-1}, y^{t+k-L}) \\ \dots & \dots & \dots \\ \mathrm{cov}(y^{t+k-L}, y^{t+k-1}) & \dots & v(x^{t+k-L} + v_0) \end{bmatrix}\right)$$

  predict $y^{t+k} \sim \mathcal{N}(m(x^{t+k}), v(x^{t+k}) + v_0)$.

Input distribution At time $t+k$, we have the random input vector $x^{t+k} = [y^{t+k-1}, \dots, y^{t+k-L}]^T$ with mean formed by the predicted means of the lagged outputs $y^{t+k-\tau}$, $\tau = 1, \dots, L$, given by (16).

The $L \times L$ input covariance matrix has the different predicted variances on its diagonal: $\Sigma^{ii}_{x^{t+k}} = v(x^{t+k-i})$, for $i = 1 \dots L$, computed with (22).

The cross-covariance terms are obtained as follows: at time step $t+k$, we predict y^{t+k} and then, for the next time step, we need to compute the covariance between y^{t+k} and $[y^{t+k-1}, \dots, y^{t+k+1-L}]$. That is, in general, we want to compute $\mathrm{cov}(y^{t+k}, x^{t+k})$:

$$\mathrm{cov}(y^{t+k}, x^{t+k}) = E[y^{t+k}x^{t+k}] - E[y^{t+k}]E[x^{t+k}] \tag{32}$$

with $E[y^{t+k}]$ given by (16) and $E[x^{t+k}] = \mu_{x^{t+k}}$. We have

$$E[y^{t+k}x^{t+k}] = \int\int y^{t+k}x^{t+k}p(y^{t+k}|x^{t+k})p(x^{t+k})dy^{t+k}dx^{t+k} \tag{33}$$

$$= \int x^{t+k}\left[\mu(\mu_{x^{t+k}}) + \frac{\partial\mu(x^{t+k})}{\partial x^{t+k}}\bigg|^T_{x^{t+k}=\mu_{x^{t+k}}}(x^{t+k} - \mu_{x^{t+k}})\right]p(x^{t+k})dx^{t+k} \tag{34}$$

which gives $E[y^{t+k}x^{t+k}] = \mu(\mu_{x^{t+k}})\mu_{x^{t+k}} + \frac{\partial\mu(x^{t+k})}{\partial x^{t+k}}\Big|^T_{x^{t+k}=\mu_{x^{t+k}}}\Sigma_{x^*}$.

So that the cross-covariance terms are given by

$$\mathrm{cov}(y^{t+k}, x^{t+k}) = \frac{\partial\mu(x^{t+k})}{\partial x^{t+k}}\bigg|^T_{x^{t+k}=\mu_{x^{t+k}}}\Sigma_{x^{t+k}}\ . \tag{35}$$

5 Illustrative examples

The first example is intended to provide a basis for comparing our Gaussian approximation to the Monte-Carlo sampling from the 'true' distribution (numerical approximation of the integral (10)) when the uncertainty is propagated as we predict ahead in time. The second one, inspired from real-life applications, enables us to analyse the difference between the iterative k-step ahead prediction when propagating the uncertainty and when using only the output estimates.

The predictive performance of the different methods is assessed computing the average absolute error (AE), the average squared error (SE) and average minus log predictive density[4] ($mLPD$).

[4]To evaluate these losses in the case of Monte-Carlo sampling, we use the sample mean and sample variance.

5.1 A $2D$ non-linear dynamic system

We present an example of a non-linear, second order, dynamic system Ω, composed of a blend of two affine models, $\Omega : \dot{\mathbf{x}} = f_1(\mathbf{x}) + f_2(\mathbf{x})$, where $f_i(\mathbf{x}) = \phi(\mathbf{x})(A_i\mathbf{x} + d_i)$, $\phi(\mathbf{x}) = \exp(- \parallel \mathbf{x} - \mathbf{c} \parallel / \sigma_i^2)$ and $\mathbf{c}_i = A_i^{-1} d_i$, with $A_1 = A_2 = \begin{bmatrix} 2 & 5 \\ -10 & -3 \end{bmatrix}$, $d_1 = \begin{bmatrix} 10 \\ -10 \end{bmatrix}$, $d_2 = -d_1$, $\sigma_1 = 1$, $\sigma_2 = 2$. This system has two stable equilibrium points and a stable limit cycle. Because the data was acquired by starting the simulation at a random initial position, and simulating for a fixed period, we find a number of interesting features which are typical of many real world examples when modelling from observed data.

The identification data consist of 6 simulations (200 points per simulation) with different starting points and the two test trajectories result from two different simulations.

Assuming a model of the type $x_{t+1} = f(x_t)$, where x is two-dimensional, we create the input and target matrices corresponding to each trajectory. We then model each dimension with a separate Gaussian process: $x_{t+1}^i = f^i(x_t)$, for $i = 1, 2$. Figure 10 shows the predictions when predicting from $k = 1$ to $k = 10$ steps ahead, when starting the simulation at six different points. Figure 11 shows the true trajectory along with the mean predictions with their 2σ uncertainties (crosses) when propagating the uncertainty with the Gaussian approximation and 30 samples from the true predictive distribution.

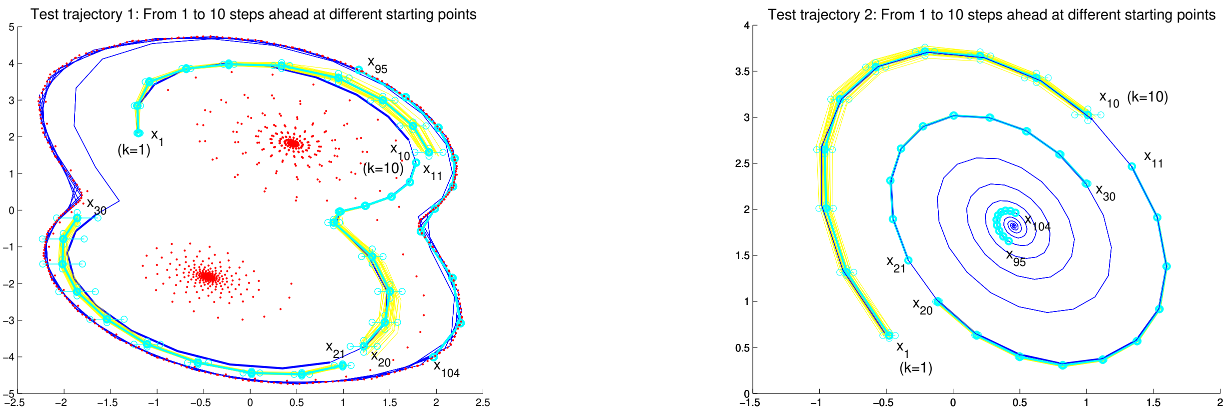

Figure 10: Iterative method in action: simulation from 1 to 10 steps ahead for different starting points of the test trajectories. Mean predictions with 2σ error bars (circles), along with 30 samples from the true distribution, when propagating the uncertainty, for the two test trajectories. Also shows (left) the training data (dots).

These figures clearly show that the Gaussian approximation is valid, with error bars encompassing the samples from the true distribution. This is confirmed quantitatively by the losses (see table 1).

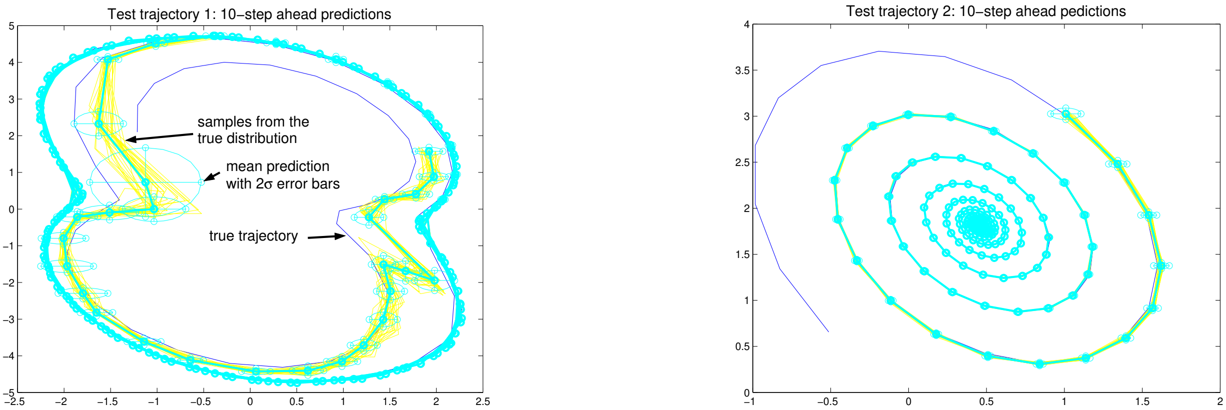

Figure 11: 10-step ahead prediction for the two test trajectories (the first point here corresponds to x_{10} in figure 10, etc). Same legend as figure 10.

Table 1: Losses for the Gaussian approximation (GP) and the sampling from the true distribution (MC) on the two test trajectories. The losses are given for each of the two dimensions.

		AE dim. 1	SE dim. 2	SE dim. 1	SE dim. 2	mLPD dim. 1	mLPD dim. 2
Traj. 1	GP	0.037	0.071	0.016	0.061	−0.040	6.471
(Fig. 11, left)	MC	0.037	0.072	0.016	0.061	3.059	3.474
Traj. 2	GP	0.003	0.003	$0.282.10^{-4}$	$0.337.10^{-4}$	−4.559	−3.656
(Fig. 11, right)	MC	0.003	0.003	$0.272.10^{-4}$	$0.358.10^{-4}$	−3.745	−3.763

5.2 Prediction of a pH process simulation

We apply the method on a pH neutralisation process benchmark ([2]). The training and test data consists of pH values (outputs y of the process) and a control input signal (u).

With a model of the form $y_t = f(y_{t-8}, \ldots, y_{t-1}, u_{t-8}, \ldots, u_{t-1})$, we have a training input matrix of size 1228×16, with the corresponding vector of targets and a different set of 15952 test data (all data had their mean removed). After maximization of the likelihood, the ARD tool indicates that the lagged outputs contributing the most are y_{t-1},y_{t-2} and y_{t-3} whose hyperparameters are one order of magnitude larger than the those corresponding to other outputs. In the same manner, more *weight* is given to u_{t-5}.

Figure 12 shows the simulation from 1 to 10 steps ahead starting at two different points and figure 13 the plots the 10-step ahead predicted points, at the beginning and at the end of the validation set. On both figures, we plot the true data and the mean predictions with their 2σ uncertainties obtained when propagating the uncertainty (circles) or not (crosses). For the 10-step ahead predictions, we have the following losses. The average absolute error and squared error are 0.3255 and 0.4090 respectively, whether one propagates the uncertainty or not. On the other hand, the average minus log predictive density is much better when propagating the uncertainty (0.6527 against 2763.1!).

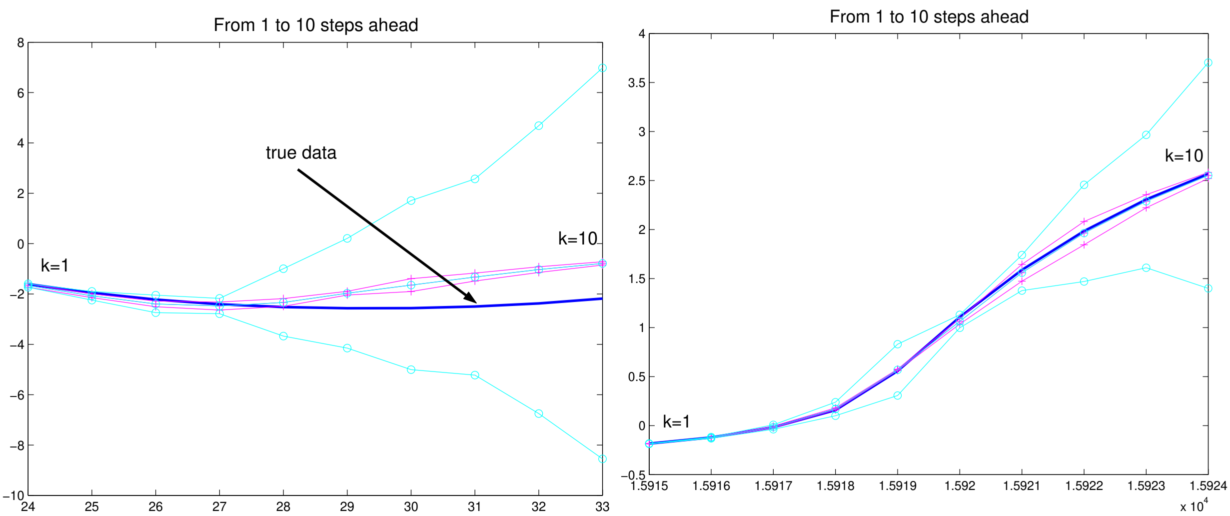

Figure 12: From 1 to 10 steps ahead: two windows of predicted means with their 2σ uncertainties with (circles) and without (crosses) propagation of the uncertainty at different times within the validation set.

As we predict ahead and propagate the uncertainty, the uncertainty does not necessarily increase monotically with k, but is also affected by changes in the control signal, see figure 12. On the other hand, we see that when the uncertainty is not propagated, the model is too confident with very small error bars. In this example, the important role of the control signal has to be noted, partly explaining the big changes in uncertainties observed on the 10-step ahead predictions plot (figure 13).

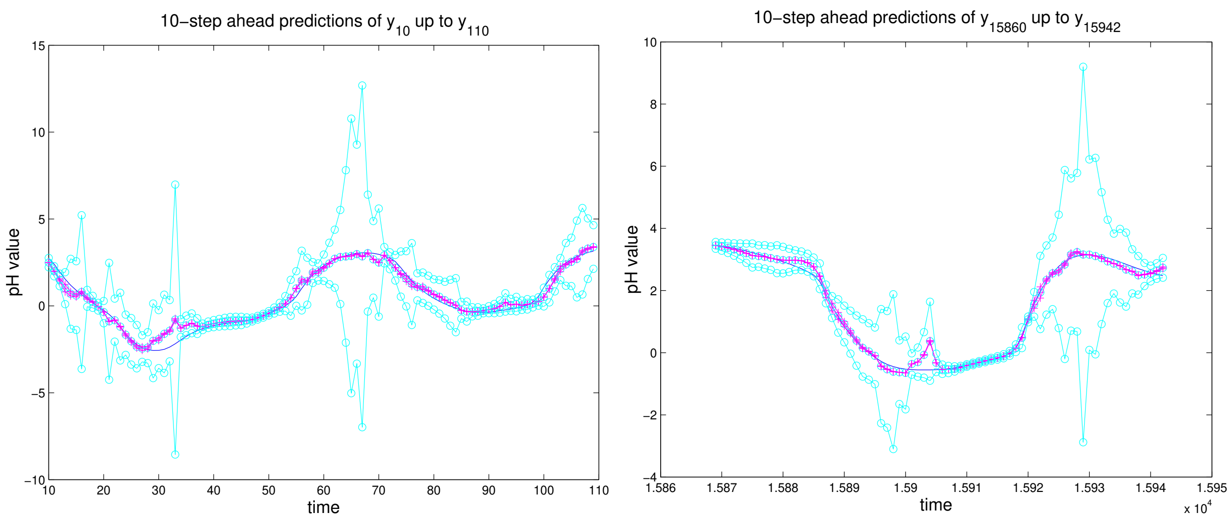

Figure 13: 10-step ahead mean predictions with their 2σ error bars, when propagating the uncertainty (circles) and when using the previous estimates only (crosses). These predictions are taken at the beginning and at the end of the test set.

6 Conclusions

We have presented an approximation which allows us to use knowledge of the variance on inputs to Gaussian process models to achieve more realistic prediction variance. This is useful in its own right in the case of noisy model inputs.

Iterating this approach allows us to use it as a method for efficient propagation of uncertainty in multi-step ahead time-series predictions. In experiments on simulated dynamic systems, comparing our Gaussian approximation to Monte Carlo simulations, we found that the propagation method is comparable to Monte Carlo simulations, and that both approaches achieved more realistic error bars than a naive approach which ignores the uncertainty on current state.

This method can help understanding the underlying dynamics of a system, as well as being useful, for instance, in a model predictive control framework where knowledge of the accuracy of the model predictions over the whole prediction horizon is required (see (Murray-Smith and Sbarbaro-Hofer, 2002) for a model predictive control law based on Gaussian processes taking account of the prediction uncertainty).

Work is currently in progress towards an exact solution within the Gaussian approximation, that is, without the Taylor expansions of the mean and variance but using their original expressions and computing the integrals analytically.

Acknowledgements

Many thanks to Professor Mike Titterington for his useful comments and corrections. The authors gratefully acknowledge the support of the *Multi-Agent Control* Research Training Network - EC TMR grant HPRN-CT-1999-00107, and support from EPSRC grant *Modern* *statistical approaches to off-equilibrium modelling for nonlinear system control* GR/M76379/01.

References

Henson, M. A. and Seborg, D. E. (1994). Adaptive nonlinear control of a ph neutralisation process. In *IEEE Trans Control System Technology*, volume 2, pages 169–183.

Mackay, D. J. C. (1997). Gaussian Processes: A replacement for supervised Neural Networks? Technical report, Cavendish Laboratory, Cambridge University. Lecture notes for a tutorial at NIPS 1997.

Murray-Smith, R. and Sbarbaro-Hofer, D. (2002). Nonlinear adaptive control using non-parametric Gaussian Process prior models. In *15th IFAC World Congress on Automatic Control, Barcelona*.

Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. PhD thesis, Dept. of Computer Science, University of Toronto.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B*, 40:1–42.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and other methods for non-linear regresion*. PhD thesis, Dept. of Computer Science, University of Toronto.

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. and Rasmussen, C. E. (1996). Gaussian Processes for Regression. In *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press.