Nonlinear Adaptive Control using Nonparametric Gaussian Process prior models

Roderick Murray-Smith,
Department of Computing Science, University of Glasgow
& Hamilton Institute, National University of Ireland, Maynooth
D. Sbarbaro,
Dept. de Ingeniería Eléctrica, Universidad de Concepción, Chile.
rod@dcs.ga.ac.uk, dsbarbar@die.udec.cl

UNIVERSITY
of
GLASGOW

Talk Overview

- Introduction to Gaussian process (GP) priors.
  - Nonparametric model with automatic structure optimisation
  - Analytic estimate of derivative variance.
- Model predictive control with GP priors.
  - Minimise Expected cost function without ignoring prediction variance
  - Automatic identification, and regularisation of control action
- Simulation example of MPC.
- Outlook for GPs in control.

Nonlinear models for adaptive control

- For unknown nonlinearities identify a nonlinear model from experimental data.
- Typically use flexible parametric models, e.g. Neural networks, Tagaki-Sugeno fuzzy models, Spline models etc.
- Not trivial to represent model or prediction uncertainty.
- Most work adopts Certainty Equivalence principle – ignore model uncertainty.
- Even if uncertainty is accounted for, typically expressed as uncertainty of parameters, ignoring uncertainty on model structure, or distance from data used to identify model.
- Problematic in sparsely populated off-equilibrium regions of state-space – model cannot warn user of uncertainty.
- We suggest an alternative, based on Gaussian process priors.

Gaussian Process priors

- Gaussian process priors as a nonparametric regression model.
- Instead of parameterising $y(x)$ as a neural network/polynomial, etc. we can place a prior directly on the space of functions.
- A Gaussian process represents the simplest form of prior over functions – we assume that any p points have a p-dimensional multivariate Normal distribution.
- Assume zero mean, so for the partitioned data $\mathbf{y}_1$ and $\mathbf{y}_2$

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \sim \mathcal{N}\left(0, \Sigma\right),$$

where the full covariance matrix Σ, is

$$\Sigma = \begin{bmatrix} \Sigma_1 & \Sigma_{12} \\ \Sigma_{21} & \Sigma_2 \end{bmatrix},$$

where $\Sigma_{21} = \Sigma_{12}^T$.

GP's con't

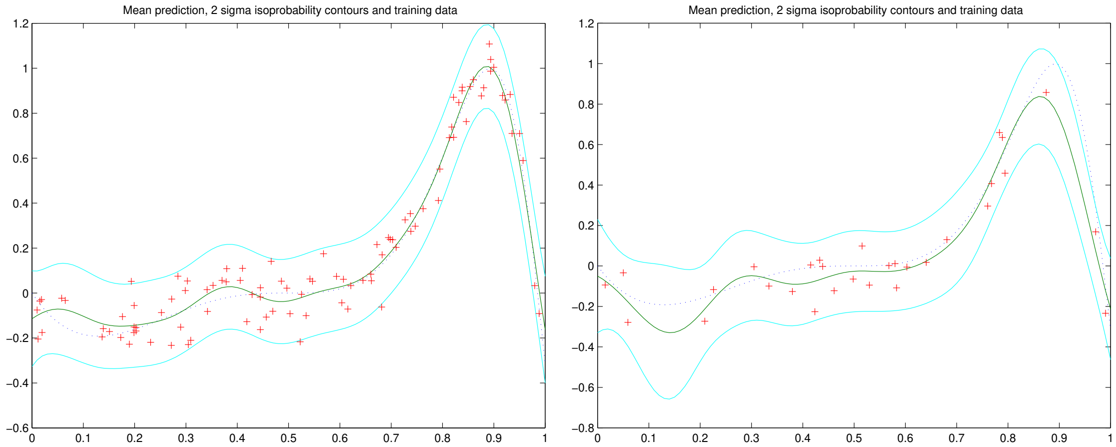
- The Gaussian Process is fully specified by a mean and its covariance function, so we denote the distribution $GP(\mu, C)$.
- The covariance function $C(x_i, x_j)$ expresses the expected covariance between y_i and y_j – we can therefore infer $\mathbf{y}_2$'s from constant $\mathbf{\Phi}_1, \mathbf{y}_1$'s rather than building explicit parametric models.

Partition joint probability into a marginal Gaussian process and a conditional Gaussian process. The marginal term,

$$P(\mathbf{y}_1|\mathbf{\Phi}_1) = (2\pi)^{-\frac{N_1}{2}}|\Sigma_1|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{y}_1^T\Sigma_1^{-1}\mathbf{y}_1\right).$$

The conditional part, $P(\mathbf{y}_2|\mathbf{\Phi}_1, \mathbf{y}_1, \mathbf{\Phi}_2) = \frac{P(\mathbf{y}_2,\mathbf{y}_1)}{P(\mathbf{y}_1)}$

$$= (2\pi)^{-\frac{N_2}{2}}|\Sigma_{2.1}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{y}_2 - \mu_{2.1})^T\Sigma_{2.1}^{-1}(\mathbf{y}_2 - \mu_{2.1})\right),$$



(a) Model mean and 2σ contours, with target function and training data

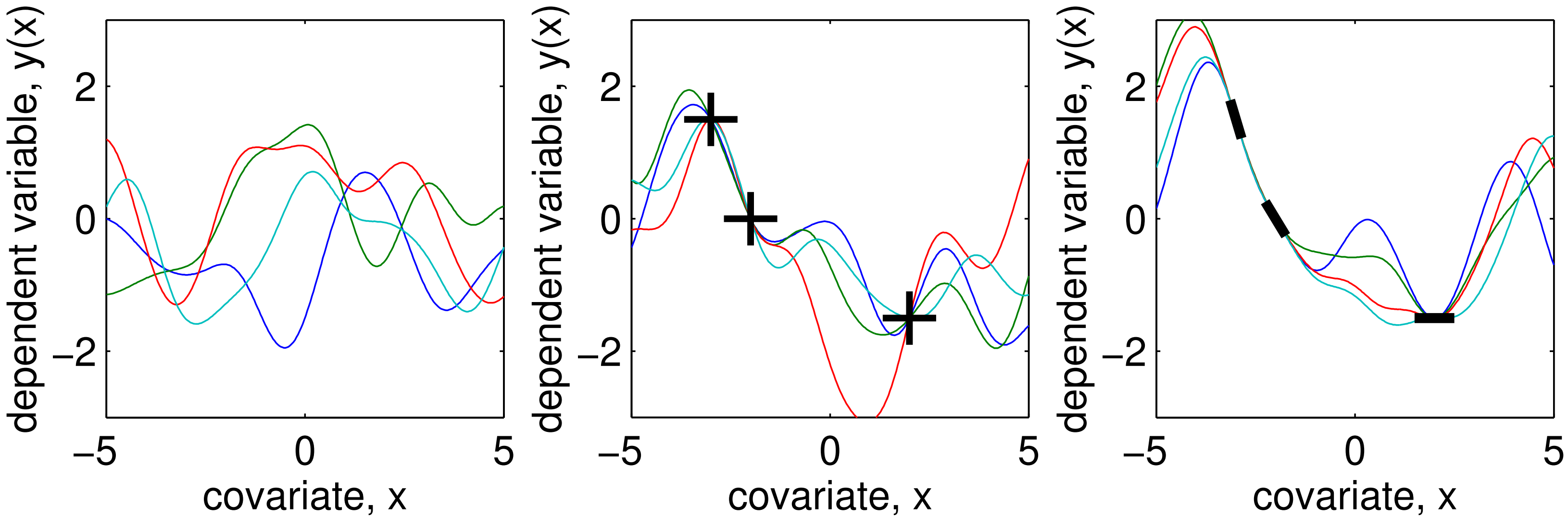
(b) Model mean and 2σ contours, with target function and training data

where, as in the straightforward multinormal case,

$$\begin{aligned}
\mu_{2.1} &= \Sigma_{12}^T \Sigma_1^{-1} \mathbf{y}_1 \\
\Sigma_{2.1} &= \Sigma_2 - \Sigma_{12}^T \Sigma_1^{-1} \Sigma_{21},
\end{aligned}$$

$y(\mathbf{\Phi}_2) = \mu_{2.1}$ as expected model output, with variance of $\sigma(\mathbf{\Phi}_2) = \Sigma_{2.1}$.

Constraining effect of function + derivative observations



(c) Examples of realisations drawn from a Gaussian process with $\alpha = \gamma_1 = 1$, left – no data, middle, showing the constraining effect of function observations (crosses), and right the effect of function & derivative observations (lines).

$$\begin{aligned} \mu_{2.1} &= \Sigma_{12}^T \Sigma_1^{-1} \mathbf{y}_1 \\ \Sigma_{2.1} &= \Sigma_2 - \Sigma_{12}^T \Sigma_1^{-1} \Sigma_{21}, \end{aligned}$$

The covariance function:

- Covariance function holds key to representation of complex functions.
- View covariance function as having model and noise terms

$$\Sigma_{1_{ij}} = C_m(\mathbf{\Phi}_{1_i}, \mathbf{\Phi}_{1_j}; \Theta) + C_n(\mathbf{\Phi}_{1_i}, \mathbf{\Phi}_{1_j}; \Theta)$$

- Noise covariance functions, e.g. uncorrelated diagonal matrix, $\delta_{ij}\mathcal{N}(\mathbf{\Phi}_1; \Theta)$, or correlated ARMA models, for example.
- Model covariance functions
  - Smoothing covariance functions $C_m(x_i, x_j; \Theta) = v_0\rho(|x_i - x_j|, \alpha)$. Points closer together are expected to have larger covariance - a smoothness expectation.
  - Linear covariance functions $C_m(x_i, x_j; \Theta) = a_0 + \sum_{k=1}^{p} a_k x_{ik} x_{jk}$,
  - Combination covariance functions (more later)

Smoothing covariance functions

A common combined covariance function,

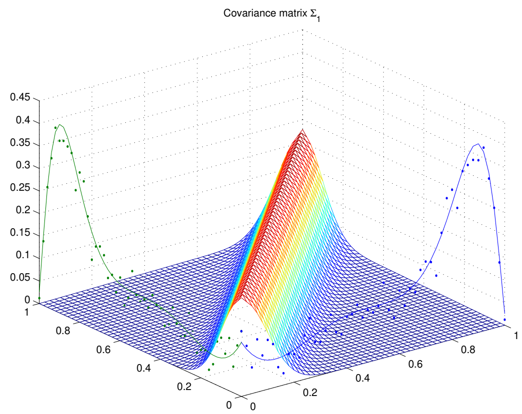
$$C(x_i, x_j; \Theta) = v_0 \rho(|x_i - x_j|, \alpha) + a_0 + \sum_{k=1}^{p} a_k x_{ik} x_{jk},$$

The distance function used here was

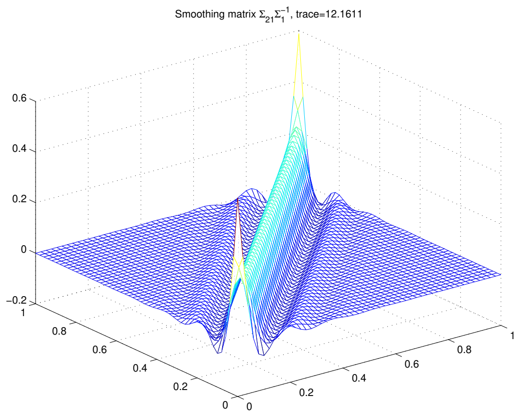
$$\rho(|x_i - x_j|, \alpha) = \exp\left(-\frac{1}{2} \sum_{k=1}^{p} \alpha_k (x_{ik} - x_{jk})^2 \right).$$

The α_k's determine how quickly the function varies in dimension k, which helps detect the importance of different input dimensions. The prior associated with this covariance function states that x's closer together should have higher covariance than points further apart.

Visualising the covariance function



(d) Covariance matrix



(e) Smooth of training data

Matrices involved in inference for a one dimensional function shown. Note the local nature of the covariance function, with a strong bias around the diagonal. This indicates that the model believes data points closer to each other, should be similar – a smoothness expectation.

$$\hat{Y}_1 = SY_1$$

where $S = \Sigma_{21}\Sigma_1^{-1}$, and rows of S are effective kernels.

Optimisation of hyperparameters

The log of the evidence is

$$L = -\frac{1}{2}\log\det\Sigma_1 - \frac{1}{2}\mathbf{y}_1^T\Sigma_1^{-1}\mathbf{y}_1 - \frac{N_1}{2}\log 2\pi$$

and the gradient of the likelihood with respect to a hyperparameter θ of the covariance function is

$$\frac{\partial L}{\partial\theta} = -\frac{1}{2}\mathrm{tr}\left(\Sigma_1^{-1}\frac{\partial\Sigma_1}{\partial\theta}\right) + \frac{1}{2}\mathbf{y}_1^T\Sigma_1^{-1}\frac{\partial\Sigma_1}{\partial\theta}\Sigma_1^{-1}\mathbf{y}_1.$$

We optimised the covariance function's hyperparameters to maximise the evidence using a standard conjugate-gradient optimisation algorithm.

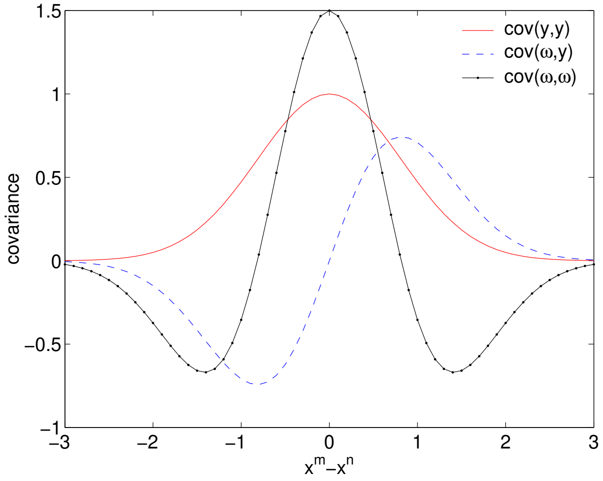
Gaussian process derivatives

- The analytic inference of derivative information from a model, and importantly, its uncertainty is potentially of great importance to control engineers designing or validating robust control laws. Other applications of models which base decisions on model derivatives will have similar potential benefits.
- The derivative of a Gaussian process remains a Gaussian process, given that covariance functions are typically differentiable, so the covariance between a derivative and function observation and the one between two derivative points satisfy

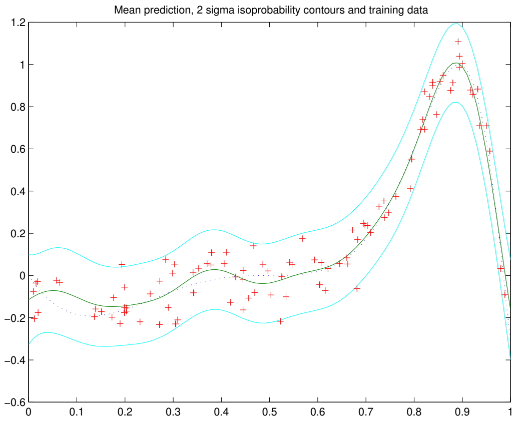
$$\mathrm{cov}(\omega^{j,m}, y^n) = \frac{\partial}{\partial x_j}\mathrm{cov}(y^m, y^n) \;\text{ and }\; \mathrm{cov}(\omega^{j,m}, \omega^{i,n}) = \frac{\partial^2}{\partial x_j \partial x_i}\mathrm{cov}(y^m, y^n).$$

- Derivative observations in Gaussian processes (O'Hagan 1992, Rasmussen 2003)
- Engineering applications in (Murray-Smith *et al.* 1999, Murray-Smith and Sbarbaro-Hofer 2002, Leith *et al.* 2002).

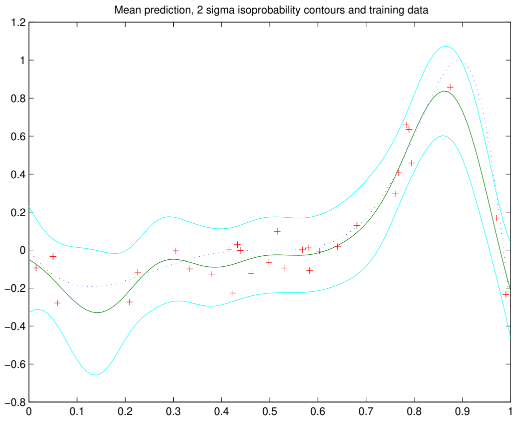
For typical GP covariance function



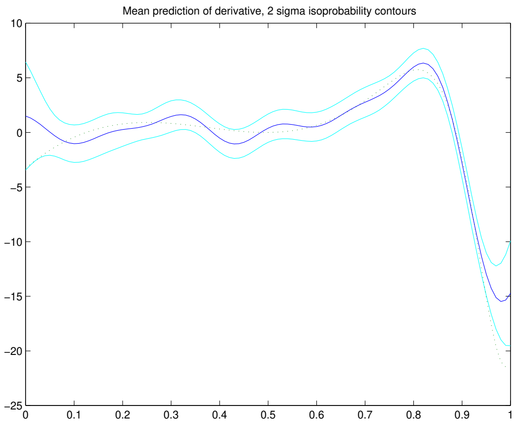
$$
\begin{aligned}
\mathrm{cov}(y^m, y^n) &= \alpha \exp(-\frac{1}{2}\|\mathbf{x}^m - \mathbf{x}^n\|_\Gamma^2) \\
\mathrm{cov}(\omega^{j,m}, y^n) &= -\alpha\gamma_j(x_j^m - x_j^n)\exp(-\frac{1}{2}\|\mathbf{x}^m - \mathbf{x}^n\|_\Gamma^2), \\
\mathrm{cov}(\omega^{j,m}, \omega^{i,n}) &= \alpha\gamma_j(\delta_{j,i} - \gamma_i(x_j^m - x_j^n)(x_i^m - x_i^n))\exp(-\frac{1}{2}\|\mathbf{x}^m - \mathbf{x}^n\|_\Gamma^2)
\end{aligned}
$$



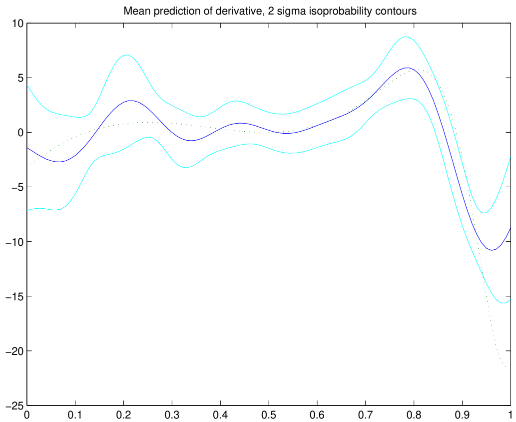
(f) Model mean and 2σ contours, with target function and training data



(g) Model mean and 2σ contours, with target function and training data



(h) Model derivative mean and 2σ contours. Note the edge effects.



(i) Model derivative mean and 2σ contours. Note the edge effects.

Practical consequences

- Derivative observations in the form of system linearisations are frequently used in control engineering, and many nonlinear identification campaigns will have linearisations of different operating regions as prior information.
- Engineers are used to interpreting linearisations, and find them a natural way of expressing prior knowledge, or constraints that a data-driven model should conform to.
- Gaussian processes:
  1. allow us to combine derivative and function observations easily,
  2. provide a seamless mechanism for consistently fusing linearisations (unlike multiple model methods, for example).
  3. Also has significant computational benefits for Gaussian processes in dynamics applications.

MPC using Gaussian Processes

Objective is to control a multi-input, single-output, affine nonlinear system of the form,

$$y(t+1) = f(x(t)) + g(x(t))u(t) + e(t+1) \tag{1}$$

where $x(t)$ is the state vector at time t. Defined as

$$x(t) = [y(t), \ldots, y(t-n), u(t-1), \ldots, u(t-m)],$$

$y(t+1)$ the output, $u(t)$ the current control vector, f and g are smooth nonlinear functions, and g is bounded away from zero.

The noise term $e(t)$ is assumed zero mean Gaussian, but with unknown variance σ^2.

1-step-ahead prediction, but work of (Girard, Rasmussen & Murray-Smith 2002) on k-step ahead should lead to k-step horizon MPC with GPs.

Proposed cost function

$$J \quad = \quad E\{(y_d(t+1) - y(t+1))^2\} + \lambda u(t)^2. \tag{2}$$

Using Var$\{y\} = E\{y^2\} - \mu_y^2$, where $\mu_y = E\{y\}$,

$$J \quad = \quad (y_d(t+1) - E\{y(t+1)\})^2 + \text{Var}\{y(t+1)\} + \lambda u(t)^2.$$

Note that we have not ‘added’ the model uncertainty term, Var$\{y(t\ +\ 1)\}$, to the classical quadratic cost function – most conventional work has ‘ignored’ it, or added extra terms to the cost function, or have pursued other sub-optimal solutions (Wittenmark 1995).

Calculation of derivative of J

$$\frac{\partial J}{\partial u(t)} = -2\left(y_d(t+1) - \mu_y\right)\frac{\partial \mu_y}{\partial u(t)} + \frac{\partial \text{Var}\{y(t+1)\}}{\partial u} + 2\lambda u(t). \quad (3)$$

With most models, estimation of Var$\{y\}$, or $\frac{\partial \text{Var}\{y\}}{\partial u(t)}$ would be difficult, but with the Gaussian process prior (assuming smooth, differentiable covariance functions - see (O'Hagan 1992)) we have straightforward analytic solutions.

$$\begin{aligned}
\frac{\partial \mu_y}{\partial u(t)} &= \frac{\partial \Sigma_{21}}{\partial u(t)}\Sigma_1^{-1}\mathbf{y}_1 \\
\text{Var}\{y\} &= \Sigma_2 - \Sigma_{21}\Sigma_1^{-1}\Sigma_{21}^T, \\
\frac{\partial \text{Var}\{y\}}{\partial u(t)} &= \frac{\partial \Sigma_2}{\partial u(t)} - \left(\Sigma_{21}\Sigma_1^{-1}\frac{\partial \Sigma_{21}^T}{\partial u(t)} + \frac{\partial \Sigma_{21}}{\partial u(t)}\Sigma_1^{-1}\Sigma_{21}^T\right),
\end{aligned}$$

Example: linear in parameters GP

As an introductory example, if we have a GP to represent a linear model, for a target system of the form

$$
\begin{aligned}
y(t+1) &= w_1 x(t) + w_2 u(t) + e(t+1) \\
&= [w_1 w_2]\phi^T(t) + e(t+1)
\end{aligned}
$$

where $\Sigma_1 = \sigma_n I + \mathbf{\Phi}_1 \Sigma_a \mathbf{\Phi}_1^T$, and $\Sigma_{21} = \phi \Sigma_a \mathbf{\Phi}_1$, where in this case we use the simple linear basis $\phi(t) = \left[\begin{array}{cc} x(t) & u(t) \end{array} \right]$,[1]

$$
\begin{aligned}
\mu_y &= \phi \Sigma_a \mathbf{\Phi}_1 \Sigma_1^{-1} \mathbf{y} = \phi^T B = \left[\begin{array}{cc} x(t) & u(t) \end{array} \right] \left[\begin{array}{c} b_1 \\ b_2 \end{array} \right] \\
\text{Var}\{y\} &= \sigma_n + \phi \Sigma_a \phi^T - \phi \Sigma_a \mathbf{\Phi}_1^T \Sigma_1^{-1} \mathbf{\Phi}_1 \Sigma_a \phi^T, \\
&= \phi (C - D) \phi^T
\end{aligned} \tag{4}
$$

[1] Other bases which included a nonlinear transformation of x, like the RBF neural networks used in (Fabri and Kadirkamanathan 1998), could be put into this framework.

where,

$$
\begin{aligned}
B &= \Sigma_a \mathbf{\Phi}^T \Sigma_1^{-1} \mathbf{y} \\
C &= \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \Sigma_a \\
D &= \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \Sigma_a \mathbf{\Phi}_1^T \Sigma_1^{-1} \mathbf{\Phi}_1 \Sigma_a
\end{aligned}
$$

for notational convenience. $d_{21} = d_{12}$ and $c_{12} = c_{21}$ (SISO).

$$
\begin{aligned}
\frac{\partial \text{Var}\{y\}}{\partial u} &= \phi \begin{bmatrix} c_{12} - d_{12} \\ c_{22} - d_{22} \end{bmatrix} + \begin{bmatrix} c_{21} - d_{21} \\ c_{22} - d_{22} \end{bmatrix} \phi^T \\
\frac{\partial J}{\partial u} &= -2(\big(y_d(t+1) - \begin{bmatrix} x(t) & u(t) \end{bmatrix} B\big)\, b_2^T + x(t)(c_{12} - d_{12}) + \\
&\quad u(t)(c_{22} - d_{22}) + \lambda u(t))
\end{aligned}
$$

Control law (linear)

At $\frac{\partial J}{\partial u} = 0$,

$$u(t) = ((y_d(t+1) - x(t)b_1)b_2^T + x(t)(c_{12} - d_{12}))(b_2 b_2^T + c_{22} - d_{22} + \lambda)^{-1} \quad (5)$$

- create variables $\alpha = x(t)(c_{12} - d_{12})$, and $\beta = c_{22} - d_{22}$.
- β term makes the use of a regularising constant λ redundant in many applications. Only when there is no model uncertainty will $\beta = 0$.

$$u = ((y_d(t+1) - x(t)b_1)b_2^T + \alpha)(b_2 b_2^T + \beta + \lambda)^{-1}$$

After $u(t)$ has been calculated, applied, and the output observed, we add the information $x(t), u(t), y(t+1)$ to the training set, and the new Σ_1 increases in size to $N_1 + 1 \times N_1 + 1$. Obviously this naive approach will only work for relatively small data sets, and for a more general solution, expected utility based selection of data for use in an active training set will be used.

Gaussian Process, nonlinear in x

$$y(t+1) = f(x(t)) + wu(t) + e(t+1)$$

Use Covar function $C(\phi_i, \phi_j) = C_{nl}(x_i, x_j) + C_{lin}(u_i, u_j)$, so $\Sigma = L + N$.

$$\begin{aligned}
\text{Var}\{y\} &= N_2 + L_2 - (N_{21} + L_{21})\Sigma_1^{-1}(N_{12} + L_{12}) \\
&= N_2 + u(t)\Sigma_a u(t)^T - (N_{21} + u(t)\Sigma_a U_1^T)\Sigma_1^{-1}(U_1\Sigma_a u(t)^T + N_{12}), \\
&= u(t)(C - D)u(t)^T - 2N_{21}\Sigma_1^{-1}U_1 C u(t)^T + N_2 - N_{21}\Sigma_1^{-1}N_{12} \\
&= u(t)(C - D)u(t)^T - 2Eu(t) + N_2 - N_{21}\Sigma_1^{-1}N_{12}
\end{aligned}$$

$$\begin{aligned}\frac{\partial J}{\partial u} &= 2\left(y_d(t+1) - (N_{21} + L_{21})\Sigma_1^{-1}\mathbf{y}_1\right)\frac{\partial L_{21}}{\partial u(t)}\Sigma_1^{-1}\mathbf{y}_1 + \\ &\quad \frac{\partial L_2}{\partial u} - \left(\Sigma_{12}^T\Sigma_1^{-1}\frac{\partial L_{21}}{\partial u(t)} + \frac{\partial L_{21}}{\partial u(t)}\Sigma_1^{-1}\Sigma_{12}^T\right) \\ &= 2\left(y_d(t+1) - N_{21}\Sigma_1^{-1}\mathbf{y}_1 - uB\right)B^T - E + u(C-D) + \lambda u),\end{aligned}$$

where,

$$\begin{aligned}B &= \frac{\partial L_{21}}{\partial u(t)}\Sigma_1^{-1}\mathbf{y} \\ C &= \Sigma_a \\ D &= \Sigma_a U_1^T \Sigma_1^{-1} U_1 \Sigma_a \\ E &= N_{21}\Sigma_1^{-1}U_1 C.\end{aligned}$$

At $\frac{\partial J}{\partial u} = 0$,

$$u(t) = ((y_d(t+1) - N_{21}\Sigma_1^{-1}\mathbf{y}_1)B^T + \alpha)(BB^T + \beta + \lambda)^{-1} \tag{6}$$

Affine in u, for given x

The above was for a linear weighting of the control $y(t+1) = f(x(t)) + wu(t)$ with the control input affecting the output through the covariance function $C_{lin}(u_i, u_j)$ which is then $U\Sigma_a U^T$.

In the case of a weighting dependent on x, $y(t+1) = f(x(t)) + g(x(t))u(t)$, we can make the covariance contribution from the control inputs an affine function where the gradient will wander with state $x(t)$. I.e. $\Sigma_a = C(x_i, x_j)$, so

$$C_{aff}(\phi_i, \phi_j) = u_i C_u(x_i, x_j) u_j \tag{7}$$

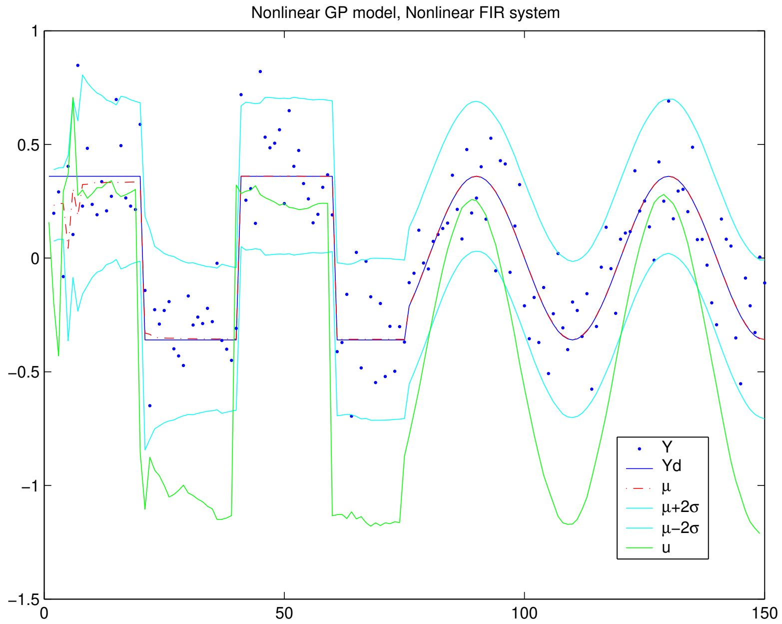
The covariance function C_u can be parameterised in any suitable way. Here we use the same structure as in C_{nl} above but we use different values of the hyperparameters to those used in C_{nl}.

Experimental results

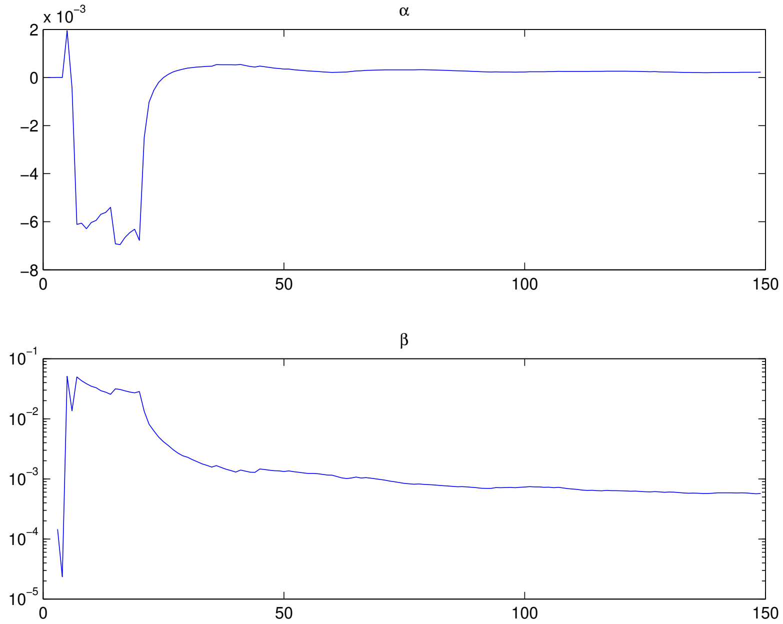
- To illustrate the feasibility of the approach we used it to control a target plant[2] based on noisy observed responses.
- We start off with only *two* training points, and add subsequent data to the model during operation. The model has had no prior adaptation to the system before the experiment.
- Model hyperparameters are adapted after each iteration using a conjugate descent algorithm. The priority here is to show to control engineers that the models are capable of high performance, even when no priors have been placed on their hyperparameters.
- The examples are taken from related dual control literature, to better allow comparison of approaches. We set $\lambda = 0$, and the state $x(t) = y(t-1)$.

[2] Adapted from (Fabri and Kadirkamanathan 1998).

FIR nonlinear plant



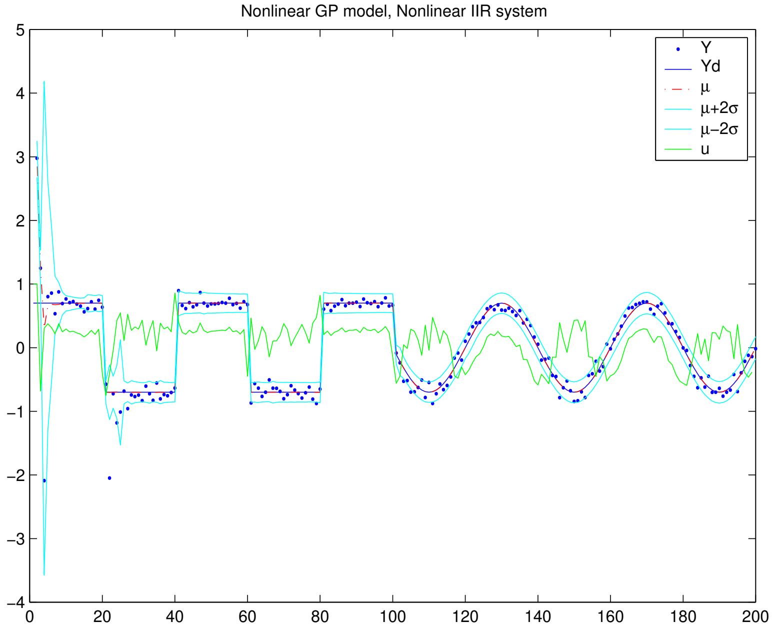
(j) Simulation of nonlinear GP-based controller



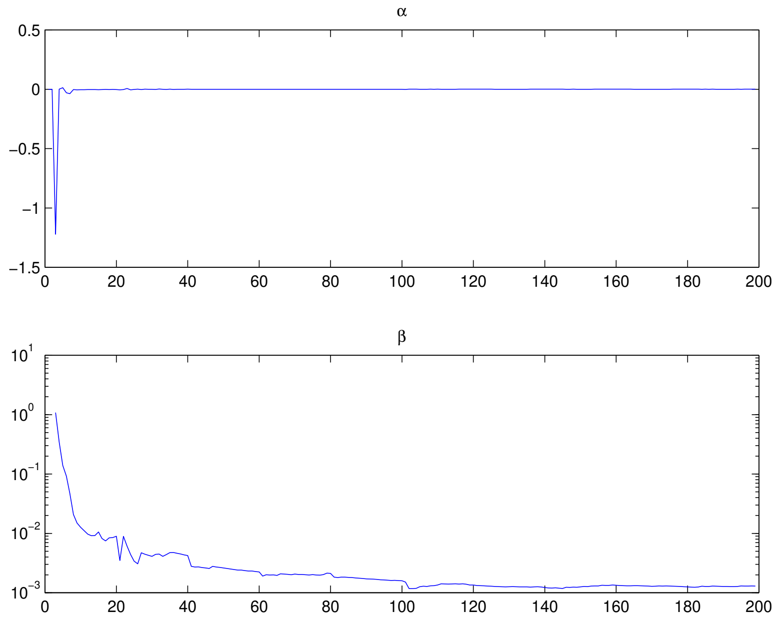
(k) α (exploration term) and β (regularisation term)

Target plant

System is $y(t) = \sin(y(t-1) + \cos(3y(t-1)) + 2u(t)$, subject to noise with variance $\sigma^2 = 0.001$.

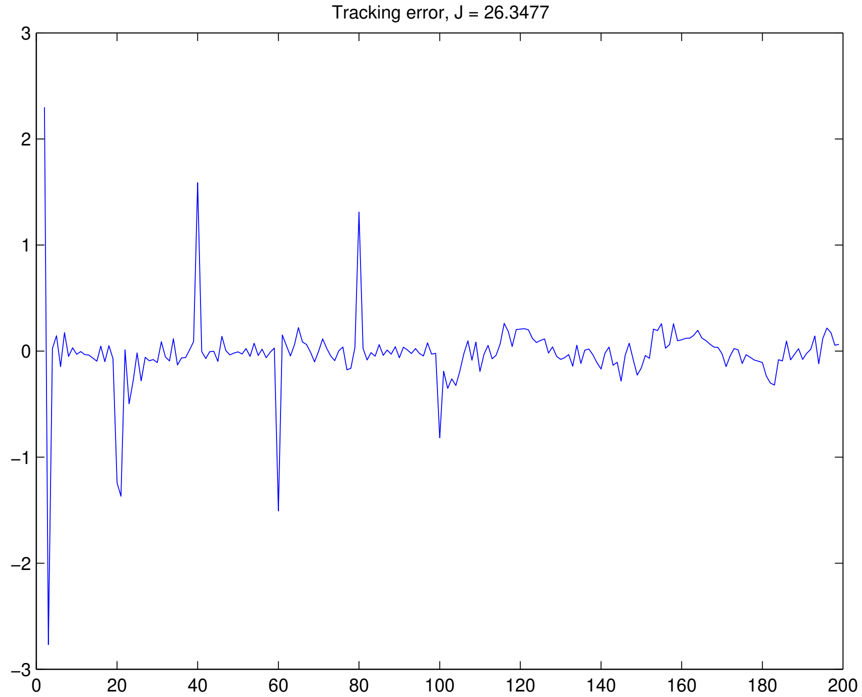


(l) Simulation of nonlinear GP-based controller



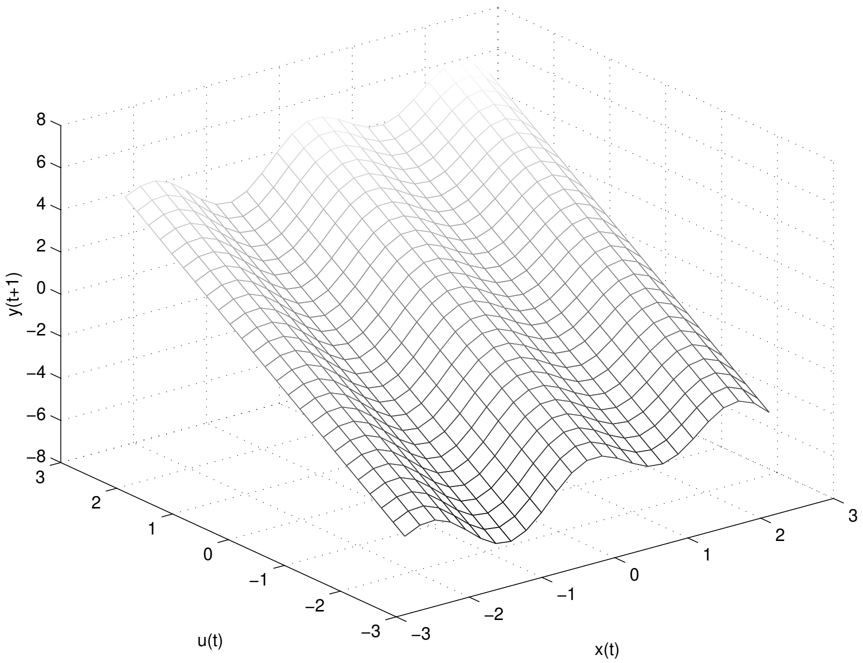
(m) α (exploration term) and β (regularisation term)

Tracking error

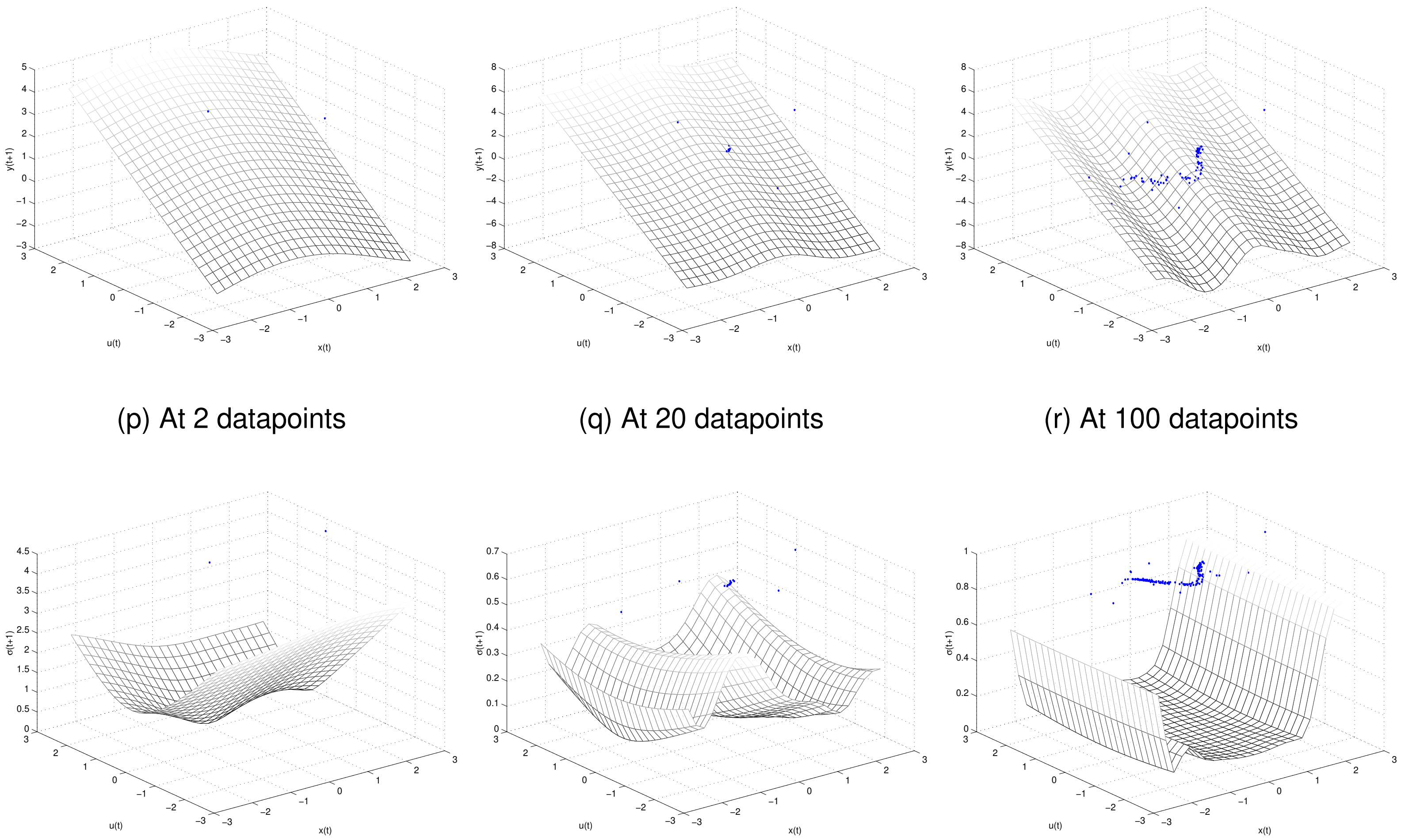


(n) Tracking error

Target system



(o) True mapping



(p) At 2 datapoints (q) At 20 datapoints (r) At 100 datapoints

(s) $\sigma(x, u)$ (t) $\sigma(x, u)$ (u) $\sigma(x, u)$

Figure 1: Mean surface (mesh) $y(t+1) = f(x(t), u(t))$ over the space $x \times u$, with $\sigma(x(t), u(t))$ uncertainty surfaces.

Outlook

- Why restrict ourselves to parametric model structures if we really just want flexible grey-box models?
- GPs can implement flexible models directly (at a computational cost)
- Adaptive control based on nonlinear, nonparametric GP model.
- Not ignoring the variance component of cost function and model derivatives.
- Leads to automatic regularisation of control behaviour in areas of state-space where model is uncertain.
- Work in progress:
  - Dealing with large data sets.
  - k-step horizons for MPC
  - Use of appropriate covariance functions for dynamic systems

Relevant literature

References

Fabri, S. and V. Kadirkamanathan (1998). Dual adaptive control of nonlinear stochastic systems using neural networks. *Automatica* **34**(2), 245–253.

Gibbs, Mark N. (1997). Bayesian Gaussian Processes for Regression and Classification. PhD thesis. University of Cambridge.

Leith, D. J., W. E. Leithead, E. Solak and R. Murray-Smith (2002). Divide & conquer identification: Using Gaussian process priors to combine derivative and non-derivative observations in a consistent manner. In: *Conference on Decision and Control*. submitted.

MacKay, D. J. C. (1999). Introduction to Gaussian Processes. NIPS'97 Tutorial notes.

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

Murray-Smith, R., T. A. Johansen and R. Shorten (1999). On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. In: *European Control Conference, Karlsruhe, 1999*. pp. BA–14.

O'Hagan, A. (1992). Some Bayesian numerical analysis. In: *Bayesian Statistics 4* (J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, Eds.). pp. 345–363. Oxford University Press.

Rasmussen, C. E. (1996). Evaluation of Gaussian Processes and other Methods for Non-Linear Regression. PhD thesis. Graduate department of Computer Science, University of Toronto.

Rasmussen, C. E. (2003). Gaussian processes to speed up Hybrid Monte Carlo for expensive Bayesian integrals. Draft: available at http://www.gatsby.ucl.ac.uk/ edward/pub/gphmc.ps.gz.

Wittenmark, B. (1995). Adaptive dual control methods: An overview. In: *Preprints of 5th IFAC symposium on Adaptive Systems in Control and Signal processing, Hungary*. pp. 67–72.