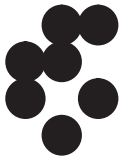

J. Stefan Institute, Ljubljana, Slovenia

Report **DP-8575**

Identification of pH neutralization process with neural networks and Gaussian process models (MAC project)

Juš Kocijan, Jozef Stefan Institute
Bojan Likar, Jozef Stefan Institute
Blaž Banko, Jozef Stefan Institute
Agathe Girard, University of Glasgow
Roderick Murray-Smith, University of Glasgow
Carl E. Rassmussen, University College London

March 2002

Contents

Chapter 1

Introduction

Whenever a new (control directed) modeling for dynamic systems is introduced there exist a wish to compare it with some already established method that is similar to the evaluated one. The purpose of this contribution is to evaluate Gaussian process (GP) dynamic models in a case study based comparison with artificial neural network (ANN) model with modeling for control design purpose in mind. The pH neutralisation process benchmark was used for the case study comparison between neural network model and GP model.

In the last decade, increases in computational power coupled with reductions in prices of personal computers have led to increased popularity of ANNs. ANNs have been used in the field of automatic control mainly as a tool for identification of non-linear dynamic processes and, together with fuzzy systems, are one of the most frequently used methods for black-box identification of non-linear systems. These models are often used for control design purpose [2].

Even though well establised and popular, ANNs still lack some properties that would even increase their use. For instance, they do not provide any information about the variance of the estimated response for each time sample. Also, a large number of data points is necessary in order to identify the model properly (although this last point would be less noticeable if the information about the estimated response's variance was available).

The use of Gaussian processes for identification of non-linear dynamic processes is discussed in this contribution. Unlike ANNs, GPs naturally provide the variance associated with the estimated output for each time sample. What needs to be investigated is if this information remains useful in the case of dynamic models where delayed outputs are lead to inputs. The second point of investigation is if dynamic features of process to be modeled can be captured with GPs as they are with ANNs.

In the next chapter, the pH neutralisation process is briefly described. This frequently used benchmark for comparison of both approaches is applied. Identification with neural networks and with Gaussian processes is presented in Chapter 3 and 4 respectively. The last chapter gives some concluding remarks.

Chapter 2

Process model

A simplified schematic diagram of the pH neutralization process taken from [1] is given in Fig. 2.1. The process consists of an acid stream (Q_1), buffer stream (Q_2) and base stream (Q_3) that

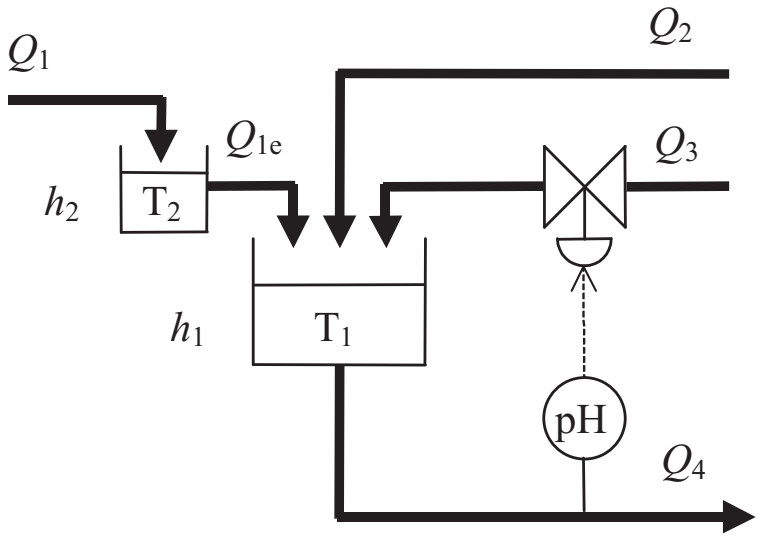

Figure 2.1: The pH neutralization system scheme

are mixed in a tank T_1. Prior to mixing, the acid stream enters the tank T_2 which introduces additional flow dynamics. The acid and base flow rates are controlled with flow control valves, while the buffer flow rate is controlled manually with a rotameter. The effluent pH (pH) is the measured variable. Since the pH probe is located downstream from the tank T_1, a time delay (T_d) is introduced in the pH measurement. In this study, the pH is controlled by manipulating the base flow rate. More detailed descriptions of the process with mathematical model and necessary parameters is presented in [1].

The dynamic model of the pH neutralization system shown in Fig. 2.1 is derived using the conservation equations and equilibrium relations. The model also includes valve and transmitter dynamics as well as hydraulic relationships for the tank outlet flows. Modeling assumptions include perfect mixing, constant density, and complete solubility of the ions involved. The Simulink simulation scheme of the process model is given in Fig. 2.2. This scheme is shown just to briefly demonstrate the complexity of the process, which contains various non-linear elements as well as implicitly calculated function (value of highly non-linear titration curve).

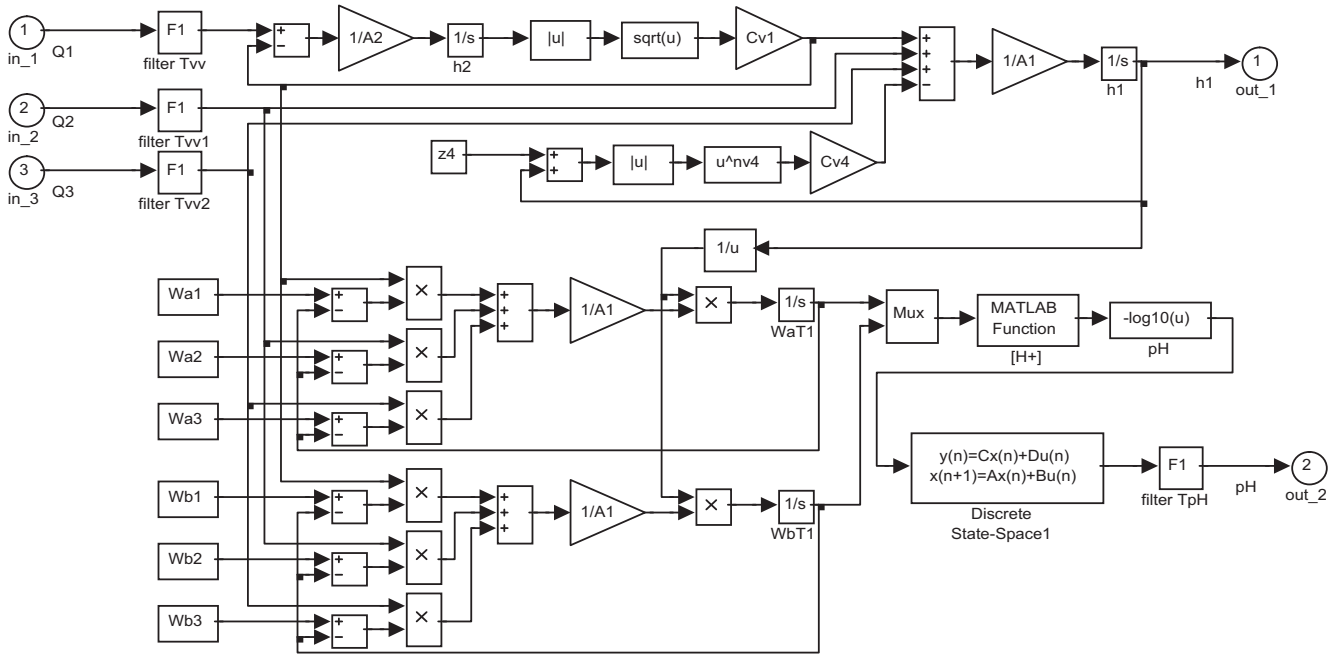

Figure 2.2: The Simulink simulation scheme of pH neutralization model

Chapter 3

Dynamic model identification with multilayer perceptron

As artificial neural networks are well established approach to system identification, number of computer tools exist to facilitate the identification. In our case, a Matlab programme package was used as the computation tool.

The process model was identified with multilayer perceptron with sigmoid activation function in hidden layer and linear function in output layer. The rationale for this choice will be explained later in the report.

3.1 Selection of input signals and model simulation

One of the most important parts of identification procedure for non-linear systems is the selection of identification and validation signals. Especially identification signal has to contain as many as possible of magnitude components over entire region of interest and provide sufficient excitation of system.

Appropriate selection of sampling time is important, because it determines the range of captured dynamics. However, only rules-of-thumb and iterative identification are approaches known for determination of sampling time without knowing the dynamics of the system to be identified.

To get a vague idea about our system dynamics some preliminary tests were pursued. The process model was excited with combination of step-like signals for estimation of dominant time constant and settling time. The dominant time constant was estimated in range between 65 and 185 s and settling time between 135 and 325 s (Fig. 3.1). This 'provisional' dynamics is necessary for estimation of appropriate sampling time. Based on responses and iterative cut-and-try procedure a sampling time of 25 seconds was selected. The sampling time was so big that dead-time mentioned in the previous chapter was lost.

The chosen identification signal was generated with generator of random noise with uniform distribution and sampling time of 50 seconds. The identification signal and system response are depicted in Fig. 3.2.

It can be observed from input signal histogram and magnitude frequency response that are given in Fig. 3.3 that magnitude and frequency content are satisfactory. That is because all components (magnitude and frequency) are contained in the signal.

The validation signal is poorer with magnitude and frequency components than the identification signal. The rationale behind this is that if identified model was excited with richer signal, than it has to respond well to the signal with less components. The validation signal

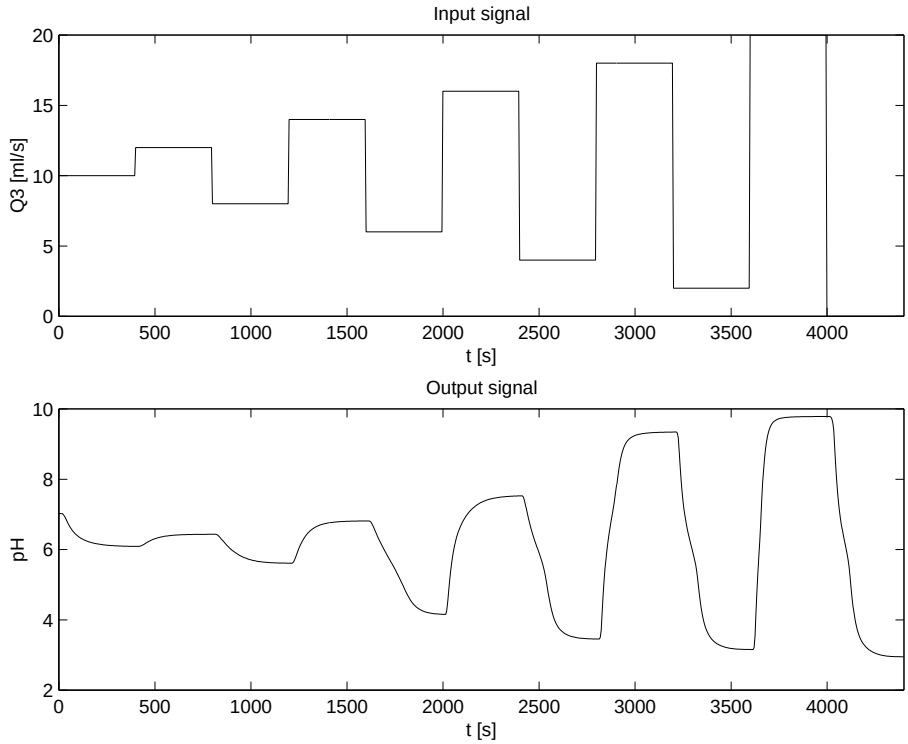

Figure 3.1: Input step-like signal (upper figure) and process model response (lower figure)

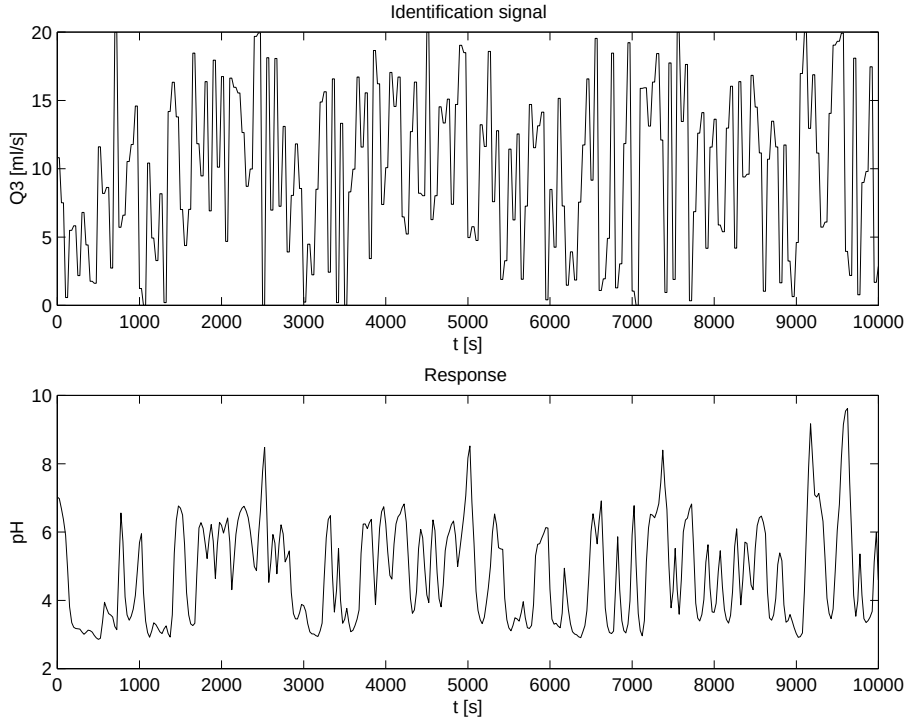

Figure 3.2: Identification signal (upper figure) and process model response (lower figure)

was obtained with generator of random noise with uniform distribution and sampling time of 500 seconds. The validation signal and system response are depicted in Fig. 3.4, while input signal histogram and magnitude frequency response are given in Fig. 3.5.

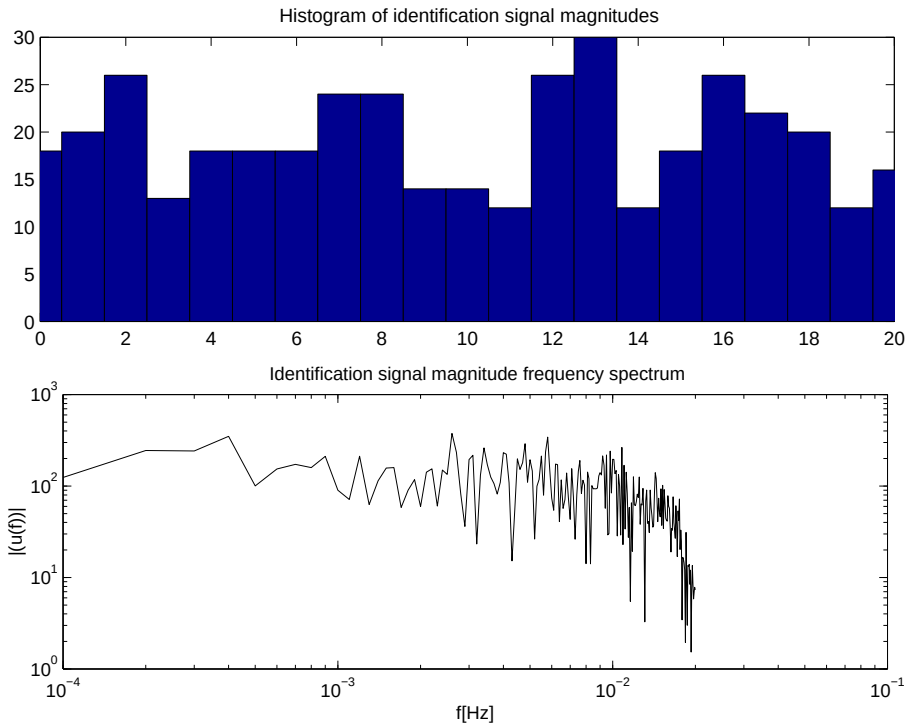

Figure 3.3: Identification signal histogram (upper figure) and magnitude frequency response (lower figure)

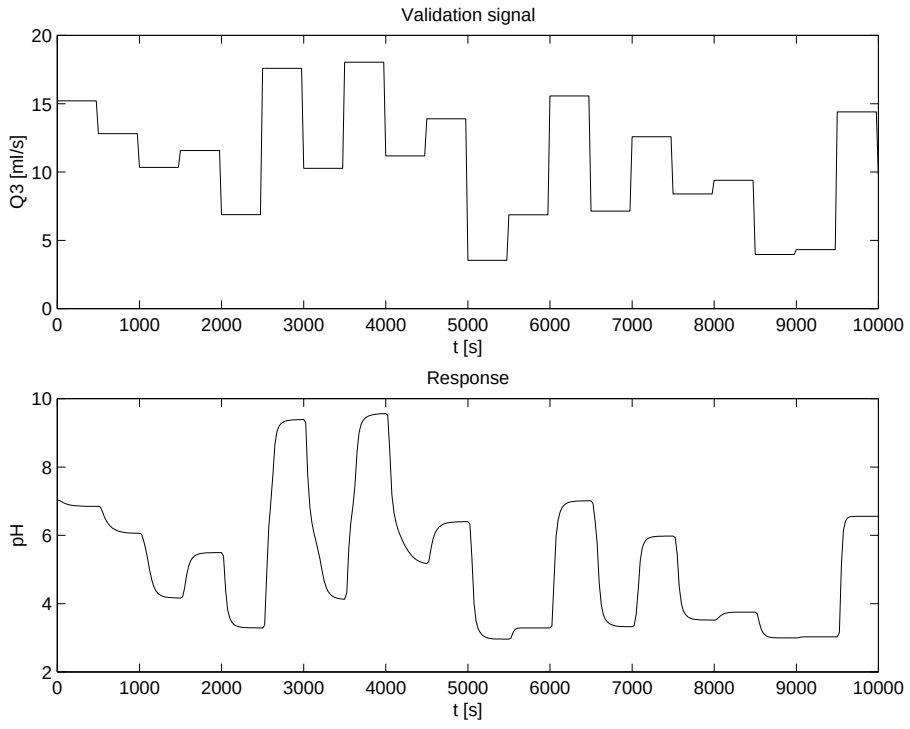

Figure 3.4: Validation signal (upper figure) and process model response (lower figure)

3.2 Neural network structure selection and parameter optimization

As already mentioned in the previous section Matlab programme package was used for identification, in particular a Neural Network Based System Identification Toolbox [3]. It was chosen

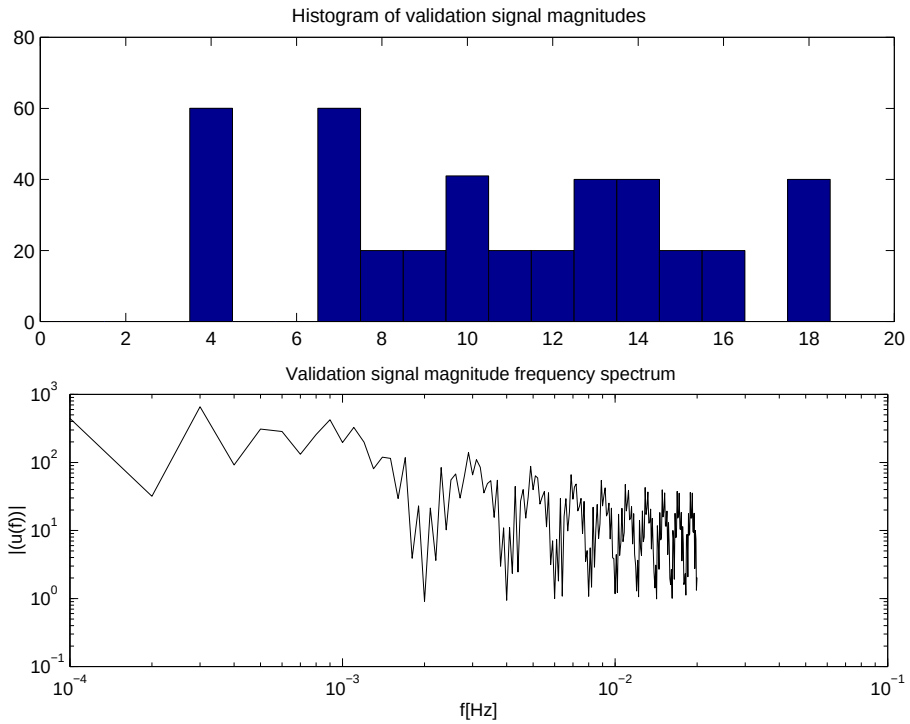

Figure 3.5: Validation signal histogram (upper figure) and magnitude frequency response (lower figure)

because it contains functions for learning, validation, simulation and optimization of multilayer perceptrons with special emphasis on identification of dynamic systems.

First all the data were scaled to have a mean value of 0 and variance of 1. For a neural network structure it was chosen a nonlinear autoregression model with exogeneous input (NARX, for neural networks NNARX) and Levenberg-Marquardt optimization method for parameter optimization. The hidden layer contained sigmoid activation functions and output layer contained a linear activation function. The activation functions are determined by use of the toolbox and could not be changed. However, selection of other non-linear function for hidden layer would not improve the quality of identified model.

The reason for NNARX structure was the relative simplicity of structure (only input and output delayed signals for regressors) and is usually selected as one of the first structure choices. The reasons for selection of optimization method was because a Levenberg-Marquard method is the standard method (not necessary the most efficient) for minimization of mean-square error criteria, due to its rapid convergence properties and robustness. An important factor of choice was also its availability in the used software.

Number of regressors (delayed inputs and outputs) was determined by toolbox function that determines the lag space. Given corresponding input and output sequences the function calculates a matrix of indices that can be helpful for determining a proper lag space structure. An insufficient lag space structure leads to a large index. While increasing the lag space the index will decrease until a sufficiently large lag space structure is reached. The knee-point of the plot is of interest because the lag space further will not change the index significantly [3].

The result of calculation of order index versus lag space is shown in Fig. 3.6. The diagram has a not distinctive knee-points at lags of 2,3 and 4 which means that it would not be unreasonable to assume that the system can be modeled by the model of form

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, u_{t-1}, u_{t-2}, u_{t-3}, u_{t-4}) \tag{3.1}$$

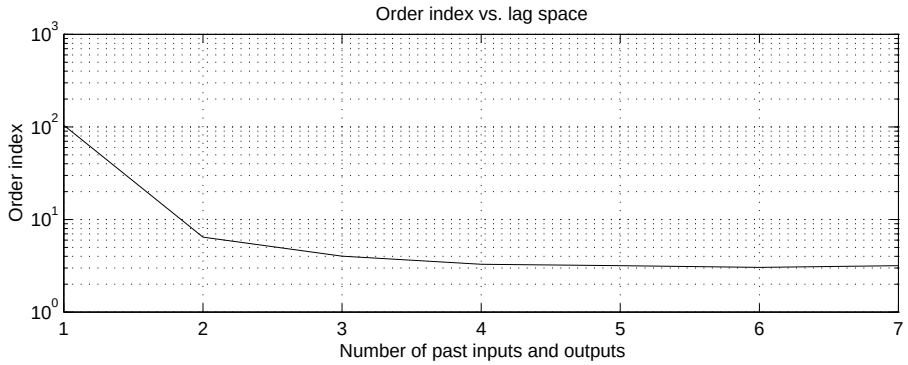

Figure 3.6: Diagram presenting number of regressors (order index) versus lag space

The optimal number of neurons in hidden layer was determined by optimization. The network was optimized for each possible number of hidden neurons in a certain range, starting from 2 and the model order from 4 to 6 (which corresponds to the number of delayed outputs contained in the vector of regressors). The evaluation function was determined as sum of absolute differences between process response and neural network response to identification input signal. The first 20 samples were not used for this evaluation due to undefined delayed values at the beginning of simulation.

At the end of this lengthy procedure the optimal parameters were obtained as follows. Three delayed inputs in addition to input and three delayed outputs in addition to output signal were chosen as regressors. Ten neurons were selected for hidden layer.

To avoid redundant connections between neurons pruning of neural networks was pursued. Only one connection was determined as redundant. The structure of neural network is given in Fig. 3.7.

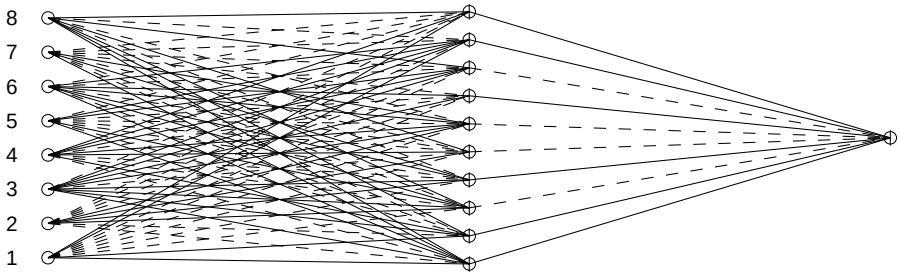

Figure 3.7: Schematic neural network structure

3.3 Validation of results

Visual inspection of the plot comparing predictions to data used for validation is probably the most important validation tool [3]. Responses of neural network and comparison with process model response to the identification and validation input signal is given in Fig. 3.8. Very good fit can be observed for identification input signal which is understandable, since this signal was used for optimization. Response to validation in put signal is obtained by simulation (not by k-step ahead prediction).

Fitting of the response for validation signal:

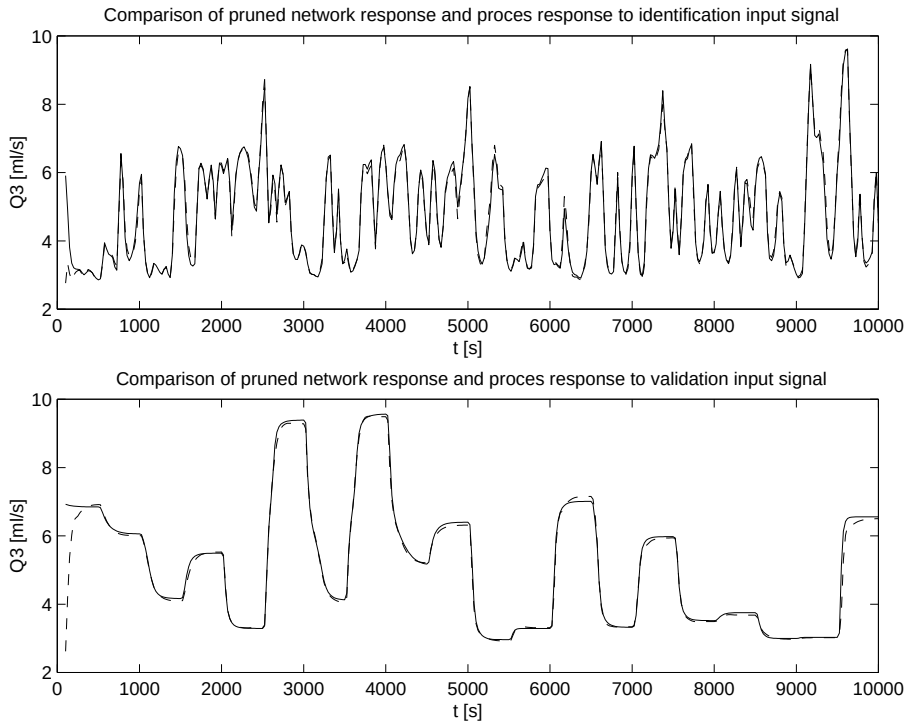

Figure 3.8: Responses of neural network (dashed line) and process model (full line) on identification input signal (upper figure) and responses of neural network (dashed line) and process model (full line) on validation input signal (lower figure)

- average absolute test error

$$AE = \frac{1}{N} \sum \mid \hat{\mathbf{y}} - \mathbf{y} \mid = 0.0672 \tag{3.2}$$

  where N is the number of used data, $\mathbf{y}$ is the process response (target) in the test set, and $\hat{\mathbf{y}}$ is the simulated value;

- average squared test error

$$SE = \frac{1}{N} \sum (\hat{\mathbf{y}} - \mathbf{y})^2 = 0.0142 \tag{3.3}$$

The auto correlation function of error between network response and process model response on validation signal and the cross correlation between error and input validation signal was used for evaluation of results. The autocorrelation is shown in Fig. 3.9 while the cross correlation is shown in Fig. 3.10.

From all presented results it can be seen that neural networks model relatively well captures the dynamics of the pH neuralization process model. The resultant neural network is not too large to be handled and was relatively routinely obtained with the selected software tool.

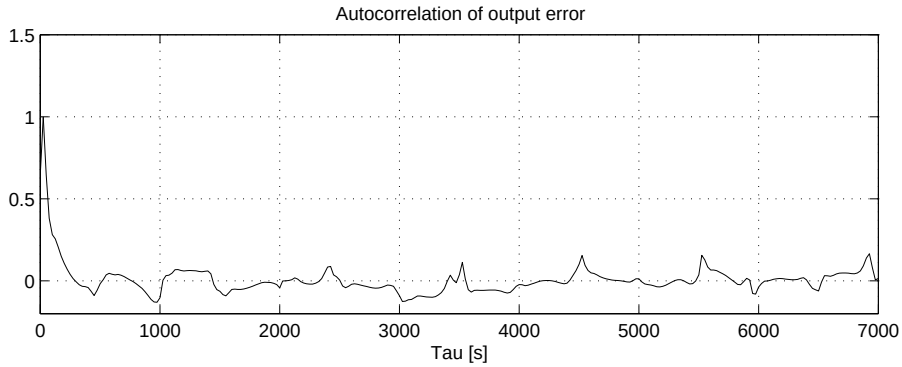

Figure 3.9: Auto correlation function of simulation error

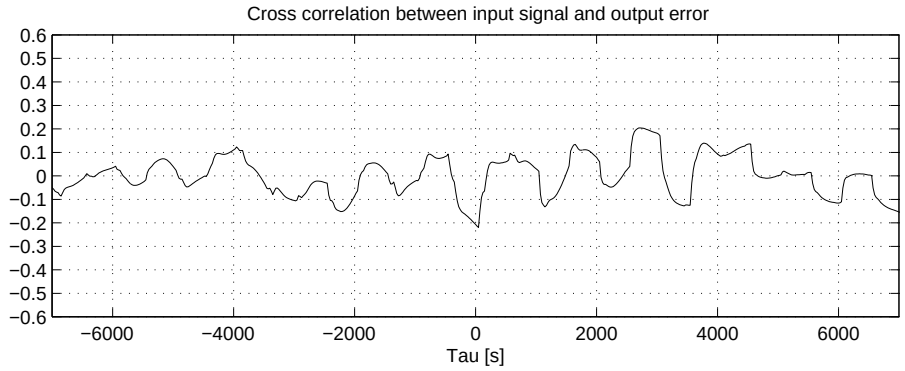

Figure 3.10: Cross correlation function of validation input signal and simulation error

Chapter 4

Dynamic model identification with Gaussian processes

4.1 A quick review of modeling with Gaussian Processes

Model assumption

A Gaussian Process is a collection of random variables which have a joint multivariate Gaussian distribution: $y^1, \ldots, y^n \sim \mathcal{N}(0, \Sigma)$, where Σ_{pq} gives the covariance between points y^p and y^q.

Assuming a relationship of the form $y = f(x)$ between the inputs x and outputs y, we have $\mathrm{Cov}(y^p, y^q) = C(x^p, x^q)$, where $C(.,.)$ is some function with the property that it generates a positive definite covariance matrix. This means that the covariance between the variables that represent the outputs for cases number p and q is a function of the inputs corresponding to the same cases p and q.

In general, a stationary (depends only on the distance between points[1]) Gaussian covariance function which depends on a set of parameters (see equation (4.1) will be used.

$$C(x^p, x^q) \;=\; v_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D} w_d (x_d^p - x_d^q)^2\right] \tag{4.1}$$

where D is the input dimension.

Dynamic models

A state-space model of the form $y_t = f(y_{t-1}, y_{t-2}, \ldots, u_{t-1}, u_{t-2}, \ldots)$ is considered, such that the current output depends on previous outputs, as well as on previous control inputs. Let $\mathbf{x}$ denote the state vector composed of the previous y's and u's.

Making one step ahead predictions is simple but rarely relevant. One step ahead predictions may often look very accurate, even though the estimated model is quite far from what it should be. This is particularly true when the system is sampled rapidly compared to its (fastest) dynamics. Rather, we wish to make k-step ahead predictions. Currently, in the GP framework, this has been achieved by either training the model to learn *how* to make k-step ahead predictions (*direct method*) or by simulating the system (repeated one-step ahead predictions up to k - *iterative method*). That is, at each time step, by feeding back the mean prediction (estimate of the output). This corresponds to $y_t = f(\hat{y}_{t-1}, \hat{y}_{t-2}, \ldots, u_{t-1}, u_{t-2}, \ldots)$ where $\hat{y}$ denotes the estimate. The iterative approach is preferred to the direct method because

[1] Points close together are more correlated than points far apart.

it provides us with predictions for any k-step ahead, unlike the direct method which is only valid for the k-step ahead points.

Learning

Given a set of training cases $\{t^i, x^i\}_{i=1}^N$, where x^i is (possibly) a D-dimensional input vector, the hyperparameters of the covariance function should be learned.

It is assumed that the data are noisy versions of the function outputs. That is, if it is assumed $t = f(x) + \epsilon$ where ϵ is an uncorrelated (white) noise with variance v_0. Then, the covariance between the training cases is given by

$$\mathrm{Cov}(t^p, t^q) \;=\; K_{pq} \;=\; C(x^p, x^q) \;+\; v_0 \delta_{pq} \tag{4.2}$$

in which the noise contribution is non zero only when $x^p = x^q$. The vector of hyperparameters to learn is then $\Theta = [w_1 \ldots w_D \;\; v_1 \;\; v_0]^T$.

The learning is done by maximizing the marginal log-likelihood

$$\mathcal{L}(\theta) = -\frac{1}{2}\log|K| - \frac{1}{2}\mathbf{t}^{\mathrm{T}} K^{-1}\mathbf{t} - \frac{N}{2}\log(2\pi) \tag{4.3}$$

where $\mathbf{t}$ is the $N \times 1$ vector of training targets and K is the $N \times N$ training covariance matrix .

Predicting

For a new test input x^*, the predictive distribution of the corresponding output is $y|x^* \sim \mathcal{N}(\mu_y(x^*), \sigma_y^2(x^*))$ with[2]

$$\mu_y(x^*) \;=\; \mathbf{k}(x^*)^T \; K^{-1} \; \mathbf{t} \tag{4.4}$$

$$\sigma_y^2(x^*) \;=\; k(x^*) \;-\; \mathbf{k}(x^*)^T \; K^{-1} \; \mathbf{k}(x^*) \tag{4.5}$$

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

4.2 Selection of regressors and parameter optimization

The same data were used as for the artificial neural networks (see previous chapter).

The code implementing the Gaussian process modeling was specifically written by C. Rasmussen[3], for the Matlab software.

There is a hyperparameter correponding to each regressor 'component' so that, after the learning, if a hyperparameter is zero or near zero it means that the corresponding regressor 'component' has little impact so could be removed. In our case a state vector (regressor) composed of 4 delayed inputs and outputs was chosen. This means that the model of the form

$$y_t = f(y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, u_{t-1}, u_{t-2}, u_{t-3}, u_{t-4}) \tag{4.6}$$

was used. This structure is very similar to the one of artificial neural networks. Effectively, the regressors are based on the process dynamics, and not on a specific model.

[2]Actually, the predictive distribution is also conditioned on the training cases and the vector of optimized hyperparameters but these will be omitted for simplicity in the notations.

[3]GP code available at http://www.gatsby.ucl.ac.uk/ edward/code/gp/

There are mainly two possible approaches: one is that Gaussian process model is learned (identified) based on data 'per se', or it can be (but it is not necessary) learned on residuals from a linear regression model when there is a strong linear trend. Both possibilities were tested and the approach that gives least error (based on data 'per se', but with mean value removed) was chosen. The data were modeled directly as a Gaussian process with covariance function described by equation (4.7).

$$\mathrm{Cov}(t^p, t^q) \;=\; K_{pq} \;=\; v_1 \exp\left[-\frac{1}{2}\sum_{d=1}^{D} w_d (x_d^p - x_d^q)^2\right] \;+\; v_0 \delta_{pq} \tag{4.7}$$

The optimization method used for identification of Gaussian process model was a conjugent gradient with line-searches. The Polack- Ribiere flavour of conjugate gradients is used to compute search directions, and a line search using quadratic and cubic polynomial approximations and the Wolfe-Powell stopping criteria is used together with the slope ratio method for guessing initial step sizes. Additionally a number of checks are made to make sure that exploration is taking place and that extrapolation will not be unboundedly large [4].

Obtained hyperparameters after optimization were as follows:

- $w_1 = 0.6648$ $(y(k))$; $w_2 = 0.1063$ $(y(k-1))$; $w_3 = 0.0024$ $(y(k-2))$; $w_4 = 0.0007$ $(y(k-3))$ which correspond to previous outputs;
- $w_5 = 0.0002$ $(u(k))$; $w_6 = 0.0275$ $(u(k-1))$; $w_7 = 0.0046$ $(u(k-2))$; $w_8 = 0$ $(u(k-3))$ which correspond to previous inputs (in brackets);
- $v_0 = 0.0045$ which is estimated noise variance and
- $v_1 = 2.339$ which is the estimate of the vertical scale of variation.

Results of simulation of the obtained 4th order Gaussian process model and its assessment is given in the following section.

4.3 Validation of results

Responses (estimates or predicted mean) of Gaussian process model and comparison with process model response to the identification and validation input signal are given in Fig. 4.1. Residuals of Gaussian process model responses on identification and validation signals are given in Fig. 4.2.

As in the case with neural network a very good fit can be observed for identification input signal which was used for optimization.

Fitting of the response for validation signal:

- average absolute test error

$$AE = \frac{1}{N}\sum \mid \hat{\mathbf{y}} - \mathbf{y} \mid = 0.0785 \tag{4.8}$$

  where N is the number of predicted points, $\mathbf{y}$ is the process response (target) in the test set, and $\hat{\mathbf{y}}$ is the simulated value;

$$SE = \frac{1}{N}\sum (\hat{\mathbf{y}} - \mathbf{y})^2 = 0.0143 \tag{4.9}$$

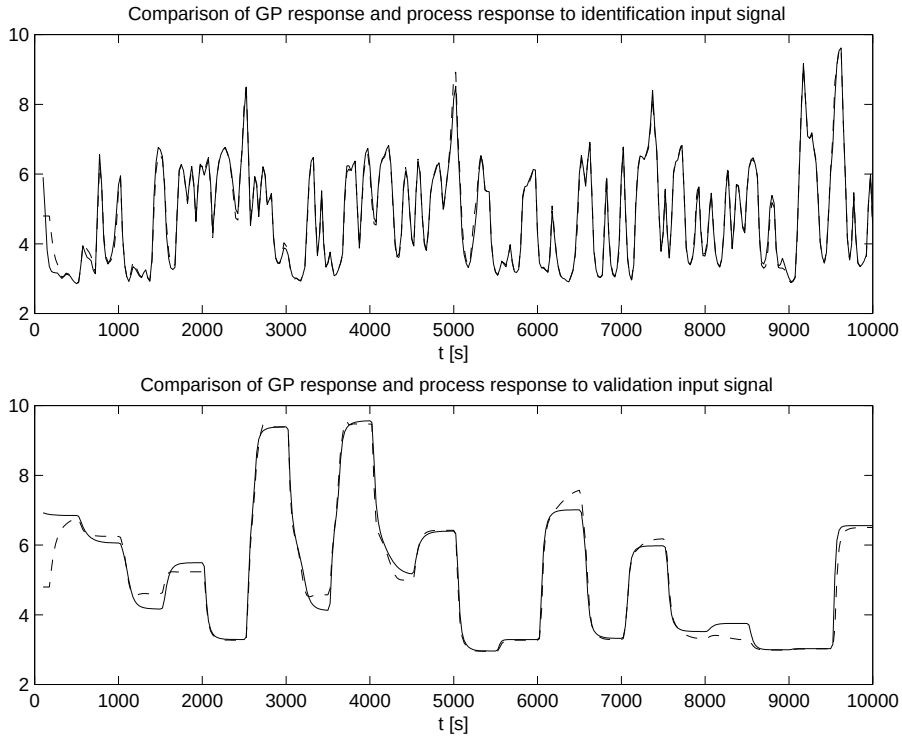

Figure 4.1: Responses of Gaussian process model (dashed line) and process model (full line) on identification input signal (upper figure) and responses of Gaussian process model (dashed line) and process model (full line) on validation input signal (lower figure)

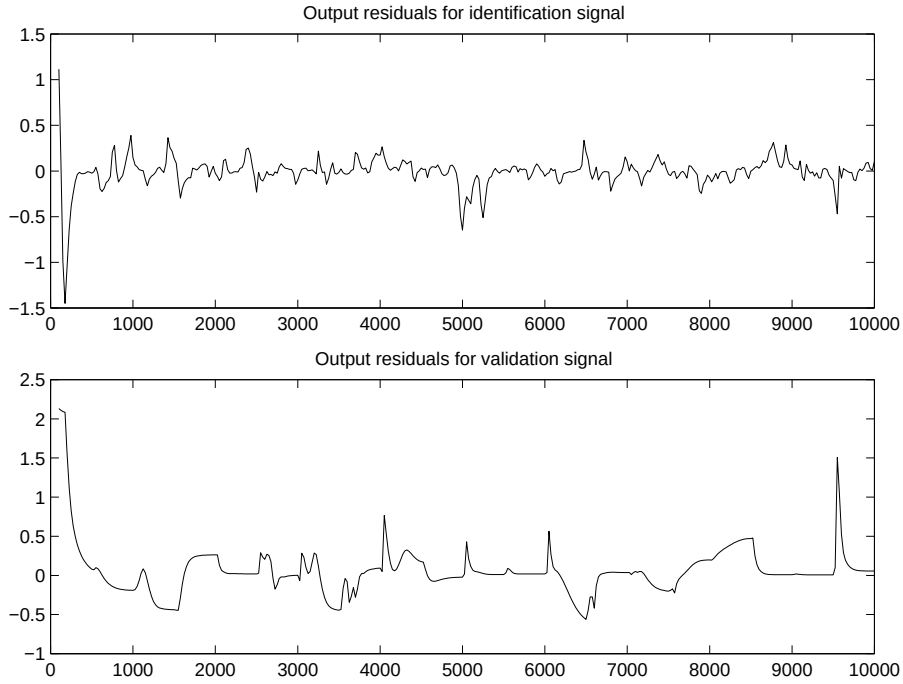

Figure 4.2: Residuals of Gaussian process model responses for identification input signal (upper figure) and for validation input signal (lower figure)

The auto correlation function of error between Gaussian process model response and process model response on validation signal and the cross correlation between error and input validation signal was used for evaluation of results. The autocorrelation is shown in Fig. 4.3 while the

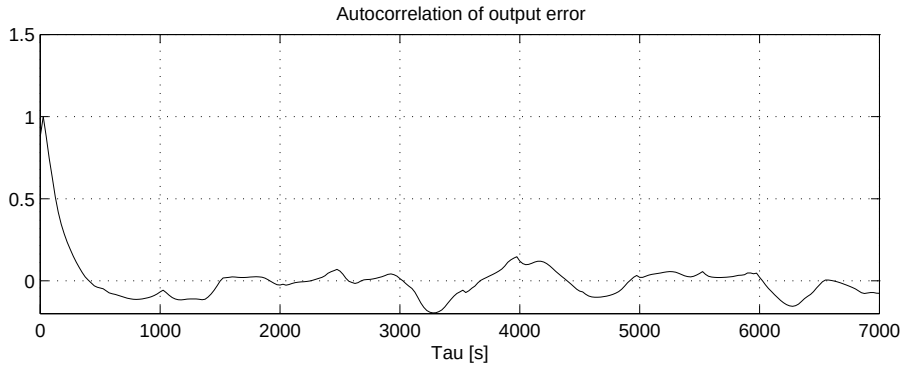

Figure 4.3: Auto correlation function of simulation error

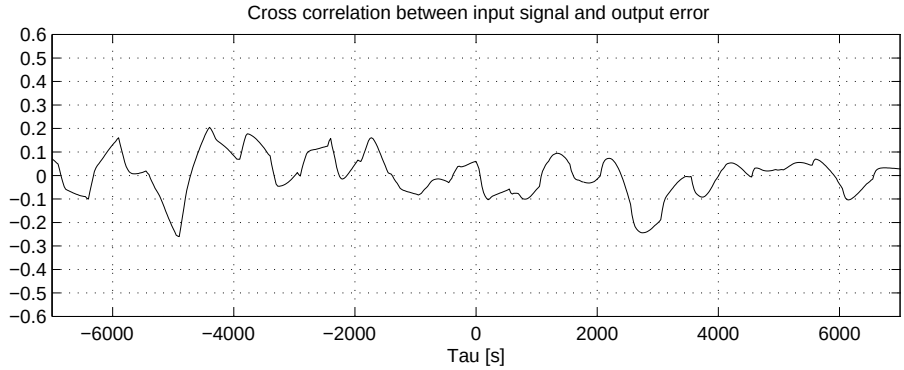

Figure 4.4: Cross correlation function of validation input signal and simulation error

cross correlation is shown in Fig. 4.4.

From the presented results it can be seen that Gaussian process models can also be used for identification of dynamic models, in our case the dynamics of the pH neuralization process model. The number of parameters which in the case of Gaussian parameters are vector of hyperparameters is much less than the number of weights in neural network.

Chapter 5

Conclusions

Two models of dynamical system were presented in the report. The first one is the well established artificial neural network model and the second one is Gaussian process model. The purpose of this report was to show that Gaussian process models are also useful for dynamic models identification via comparison on a case study, namely pH process model identification. The purpose of model developing was its use for system control.

The following conclusions can be drawn from obtained results:

- Gaussian process models can be used effectively for dynamic model identification.
- There were no problems encountered with the GP, when dealing with a rather small number of data, but this model is difficult to apply to large data sets because of computational reasons (inversion of the $N \times N$ training covariance matrix). However, efforts are continuing to solve this problem.
- Obtained Gaussian process model is relatively simple to implement and contains smaller number of structure parameters than corresponding neural network. Designer needs to make a choice about the regressors (which is not something that is method related, but common to all black-box modeling approaches), initial values of hyper parameters and covariance function. Designer of neural networks has to decide on number of layers and nodes in addition.
- Both models have shown comparable results for about the same choice of regressors though better results were obtained by artificial neural networks.
- There are still possibilities for further development of Gaussian process models identification, especially in the direction of improved (faster) optimization method and development of software which is more time optimal.

Bibliography

[1] M.A. Henson, D.E. Seborg (1994): Adaptive Nonlinear Control of a pH Neutralization Process, IEEE Trans. Control System Technology, Vol. 2, No. 3, 169-183.

[2] M. Nørgaard, O. Ravn, N.K. Poulsen, L.K. Hansen (2000): Neural Networks for Modelling and Control of Dynamic Systems, Springer-Verlag, London.

[3] M. Nørgaard (2000), Neural Network Based System Identification Toolbox, Version 2, Department of Automation, Technical Report 00-E-891, Technical University of Denmark, Lyngby.

[4] C.E. Rasmussen (1996), Evaluation of Gaussian Processes and other Methods for Non-Linear Regression, Ph.D. Disertation, Graduate department of Computer Science, University of Toronto, Toronto.