System Identification, Prediction, Simulation and Control with Neural Networks

Sørensen, O.

Publication date:
1997

Document Version
Også kaldet Forlagets PDF

SYSTEM IDENTIFICATION, PREDICTION, SIMULATION AND CONTROL WITH NEURAL NETWORKS

OLE SØRENSEN
Aalborg University
Inst. of Electronic Systems, Dept. of Control Engineering
Fredrik Bajers vej 7C, DK-9220 Aalborg Ø, Denmark
phone +45 96 35 87 48, telefax +45 98 15 17 39, E-mail os@control.auc.dk

Abstract: The intention of this paper is to make a systematic examination of the possibilities of applying neural networks in those technical areas, which are familiar to a control engineer. In other words, the potential of neural networks in control applications is given higher priority than a detailed study of the networks themselves. With this end in view the following restrictions have been made:

- Amongst numerous neural network structures, only the Multi Layer Perceptron (a feed-forward network) is applied.
- Amongst numerous training algorithms, only the Recursive Prediction Error Method using a Gauss-Newton search direction is applied.
- Amongst numerous model types, often met in control applications, only the Non-linear ARMAX (NARMAX) model, representing input/output description, is examined.

A simulated example confirms that a neural network has the potential to perform excellent System Identification, Prediction, Simulation and Control of a dynamic, non-linear and noisy process. Further, the difficulties to control a practical non-linear laboratory process in a satisfactory way by using a traditional controller are overcomed by using a trained neural network to perform non-linear System Identification in a pole-placement control structure.
Keywords: Neural Network, NARMAX-model, Non-linear System Identification, Non-linear Control.

1 Introduction

With a background in Control Engineering, artificial neural networks are simply a combination of a special model structure and a learning algorithm. This combination suddenly opens up new exciting possibilities to identify and control 'difficult processes' (dynamic, non-linear and multivariable), which leads directly to two areas of application within control engineering:

- **Prediction and Simulation of industrial processes.** For processes, which are complex and difficult to model, an artificial neural network based predictor or simulator can give a reduction in the time spent for modelling, as well as a quantitatively better performance.
- **Controllers for industrial processes.** The use of artificial neural networks is a promising technique to control the vast majority of industrial processes, in which the stability problem is not complicated, but where there are great economic savings in optimizing the controlling of the process, provided that substantial efforts are not spent on optimizing. From an industrial point of view there are great expectations to the application of neural networks in control applications.

These two areas are the guidelines for the content of this paper. The purpose is not primarily to focus on the networks themselves, but rather to show how they can be used for modeling and control. In other words, to fill the gab between artificial neural networks and control engineering, and to illustrate, that modeling, based on artificial neural networks, is vastly related to System Identification, a discipline applied by control engineers for the last 2-3 decades.

Until now Classical and Modern Control Theory have been based on linear models, but the emergence in the late eighties of artificial neural networks suddenly made it possible to perform non-linear System Identification, thus giving a basis for better simulation and control of complex processes.

The model and control concept treated in this paper are generalized' to the non-linear case, based on the corresponding linear case.

2 The neural network

Hornik [1] has shown that a neural network structured as a Multi Layer Perceptron (MLP) containing

- an input layer (nodes)
- a hidden layer with a sufficient number of neuron functions including offsets
- an output layer with only linear neuron functions without offsets

has the capability to act as a universal approximator. A MLP of the above mentioned structure is in a short matrix notation shown in fig.1. The matrix W_1 represents the input weights, the matrix W_2 represents the output weights and F represents a vector function containing the non-linear neuron functions (here tanh-functions). The last column in W_1 represents the weights on the "1" giving the necessary offset in the network.

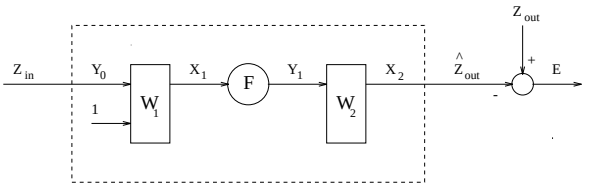

Figure 1: The architecture of a one hidden layer MLP with matrix notation.

This fact vastly simplifies the MLP, and it considerably reduces the time consumption for training, so this structure will be applied overall in this paper. For that reason relevant results for this one hidden layer MLP are summed up here.

The output is calculated using

$$\hat{Z}_{out} = W_2\, F\left(W_1 \left\{ \begin{array}{c} Z_{in} \\ 1 \end{array} \right\}\right) \tag{1}$$

If the network has m_0 inputs and m_2 outputs a m_2 x m_0 matrix G, which here is named the gain matrix of the network, is defined

$$G \doteq \frac{d\hat{Z}_{out}}{dZ_{in}^T} = \left\{ \begin{array}{ccc} \frac{d\hat{Z}_{out}(1)}{dZ_{in}(1)} & \cdots & \frac{d\hat{Z}_{out}(1)}{dZ_{in}(m_0)} \\ \vdots & & \vdots \\ \frac{d\hat{Z}_{out}(m_2)}{dZ_{in}(1)} & \cdots & \frac{d\hat{Z}_{out}(m_2)}{dZ_{in}(m_0)} \end{array} \right\} \tag{2}$$

Consequently, G is the actual incremental (small signal) gain of the network, and a calculation gives

$$\begin{aligned} G &= \frac{d\hat{Z}_{out}}{dZ_{in}^T} \\ &= \frac{d\hat{Z}_{out}}{dY_1^T}\frac{dY_1}{dX_1^T}\frac{dX_1}{dZ_{in}^T} \\ &= W_2\, F^{'}(X_1)\, W_1^* \end{aligned} \tag{3}$$

where the following abbreviation is introduced

$$W_1^* \doteq W_1(exclusive\ last\ column), \tag{4}$$

Note, that G is very easily calculated by (3) from a trained neural network. G will be extensively used in the following.

3 The process model

3.1 NARMAX model

In [2] non-linear system identification using neural networks is discussed, but the simulation study is restricted to Non-linear ARX (NARX) models. This paper, however, adopts a NARMAX model and shows how the non-linear system identification can be performed, using a neural network trained by a method known from linear system identification.

A linear ARMAX model is written

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k) \tag{5}$$

where $y(k)$, $u(k)$ and $e(k)$ are output, input and noise respectively (especially, for a SISO process they are all scalars), A, B and C are polynomials in the backward time shift operator q^{-1}. The noise $e(k)$ is normal, white and with zero mean.

The polynomials are

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_p q^{-p} \\ B(q^{-1}) &= b_1 q^{-1} + \cdots + b_m q^{-m} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \cdots + c_p q^{-p} \end{aligned} \tag{6}$$

Here p and m denote the number of delayed out- and inputs respectively.

The output $y(k)$ is thus calculated using a difference equation

$$\begin{aligned} y(k) &= -a_1 y(k-1) \cdots - a_p y(k-p) \\ &\quad +b_1 u(k-1) \cdots + b_m u(k-m) \\ &\quad +e(k) + c_1 e(k-1) \cdots + c_p e(k-p) \end{aligned} \tag{7}$$

from which the optimal one step predictor is found

$$\begin{aligned} \hat{y}(k) &= -a_1 y(k-1) \cdots - a_p y(k-p) \\ &\quad +b_1 u(k-1) \cdots + b_m u(k-m) \\ &\quad +c_1 e(k-1) \cdots + c_p e(k-p) \\ y(k) &= \hat{y}(k) + e(k) \end{aligned} \tag{8}$$

With inspiration from this linear ARMAX model a Non-linear ARMAX (NARMAX) model is naturally defined

$$\begin{aligned}\hat{Y}(k) &= \mathcal{F}(Y(k-1),\cdots,Y(k-p),\\ &\quad U(k-1),\cdots,U(k-m),\\ &\quad E(k-1),\cdots,E(k-p),\theta)\\ Y(k) &= \hat{Y}(k)+E(k)\end{aligned} \tag{9}$$

where $\mathcal{F}$ is a non-linear vector function, θ represents the parameters and $E(k)$ is the prediction error.

The corresponding neural network model is shown in fig.2, which is a Recurrent Network, since it contains feedback loops around the network.

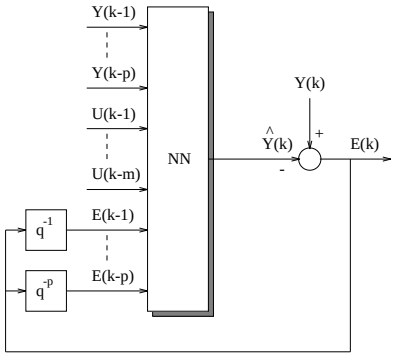

Figure 2: A neural network NARMAX model

During and after training the actual (on-line) gain matrix $G(k)$ can be extracted from the MLP. $G(k)$ is calculated by (3) and partitioned by

$$\begin{aligned}G(k) &= \frac{d\hat{Z}_{out}(k)}{dZ_{in}^T(k)}\\ &= \frac{d\hat{Y}(k)}{d\{Y^T(k-1),\cdots,E^T(k-p)\}}\\ &= \left\{\frac{\partial\hat{Y}(k)}{\partial Y^T(k-1)},\cdots,\frac{\partial\hat{Y}(k)}{\partial E^T(k-p)}\right\}\\ &= \{-\hat{a}_1(k),\cdots,-\hat{a}_p(k),\\ &\quad \hat{b}_1(k),\cdots,\hat{b}_m(k),\\ &\quad \hat{c}_1(k),\cdots,\hat{c}_p(k)\}\end{aligned} \tag{10}$$

where

$$\begin{aligned}\hat{a}_i(k) &= \frac{\partial\hat{Y}(k)}{\partial Y^T(k-i)},\quad i=1\cdots p\\ \hat{b}_i(k) &= \frac{\partial\hat{Y}(k)}{\partial U^T(k-i)},\quad i=1\cdots m\\ \hat{c}_i(k) &= \frac{\partial\hat{Y}(k)}{\partial E^T(k-i)},\quad i=1\cdots p\end{aligned} \tag{11}$$

For physical processes the individual elements of the input and output vectors are measured in physical units, which often are of quite different orders of magnitude. A training session on that basis is inconvenient for two reasons. Firstly, noting that the applied neuron functions are tanh-functions, the risk of operating at the extremes of the tanh-function is greater, thereby increasing the time taken for training. Secondly, the training session gives a higher priority to those elements of the output vector, which are accidentally measured in large physical units. For these reasons a scaling is performed. It is here chosen to scale all individual inputs and desired outputs in such a way that, based upon the training set, the mean value becomes zero and the standard deviation becomes one.

In the rest of this paper all measured signals are given a subscript 'm', contrary to the scaled signals.

3.2 Training method

Since fig.2 comprises feedback loops around the MLP, it is a Recurrent Network, and training a Recurrent Network is rather more complicated than training a normal 'static' MLP by the well-known Back Propagation algorithm [3], [4], [5].

A second order Recursive Prediction Error Method (RPEM) using a Gauss-Newton search direction has been applied, since this direction is more efficient than the gradient direction. In addition, the method is dominant in linear system identification.

The method is rather involved and it is explained in several textbooks [6] and papers [2], [5], [7].

4 A non-linear test process

A linear test process is applied by Ljung in [6]. It is a single input/single output (SISO), second order ARMAX process containing a colored noise contribution, and the input/output description is

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k) \tag{12}$$

where $y(k)$, $u(k)$ and $e(k)$ are output, input and noise respectively (all are scalars), A, B and C are polynomials in the backward time shift operator q^{-1}. The noise $e(k)$ is normal and white with zero mean and with a standard deviation $\sigma_e = 0.1$.

The polynomials are

$$\begin{aligned}A(q^{-1}) &= 1+a_1q^{-1}+a_2q^{-2}\\ &= 1-1.50q^{-1}+0.70q^{-2}\\ B(q^{-1}) &= b_1q^{-1}+b_2q^{-2}\\ &= 1.00q^{-1}+0.50q^{-2}\\ C(q^{-1}) &= 1+c_1q^{-1}+c_2q^{-2}\\ &= 1-1.00q^{-1}+0.20q^{-2}\end{aligned} \tag{13}$$

Now, this process is changed such that static, as well as dynamic, non-linearities are obtained by letting the coefficient a_2 depend on the current value of the output of the process as indicated in (14). Since the process contains one time delay, a_2 in fact depends on the one sample delayed output of the process.

$$a_2(k) = 0.70 + 0.10\frac{y(k-1)}{15} \tag{14}$$

This implies static as well as dynamic non-linearities. For $-10 < y < 10$ the DC-gain varies from 11.3 to 5.6, the process zero is permanently situated in $-0.5 + j0$, the noise zeros are permanently situated in $0.724 + j0$ and $0.276 + j0$, while the poles vary from $0.750 \pm j0.266$ to $0.750 \pm j0.452$, as illustrated in the pole-zero map in fig.3.

In the pole-zero map a grid is shown. Lines of constant damping ratio ζ and normalized natural frequency ω_n are drawn in. The damping ratio lines are drawn from $\zeta = 0$ to 1 in steps of 0.1, while the natural frequency lines are drawn from 0 to π in steps of $\pi/10$.

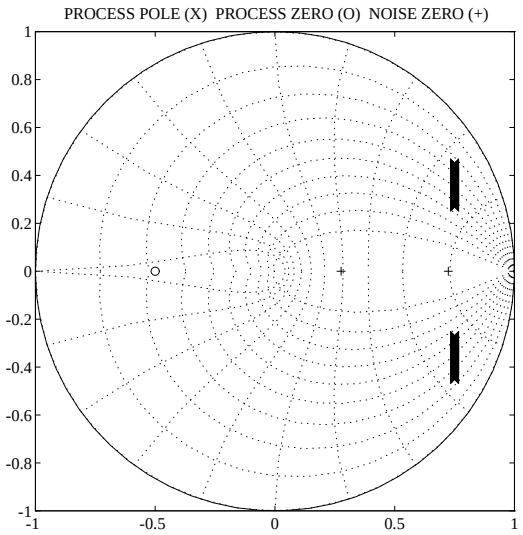

Figure 3: The displacements of process poles (x), process zeros (o) and noise zeros (+) for the non-linear test process

As input for the training set a Pseudo Random Binary Sequence (PRBS) with 500 samples is applied to ensure that the static as well as the dynamic properties are excited. As input for the test set a symmetrical staircase signal with 500 samples is applied to make the behavior of the process more transparent. The training set is shown in fig.4 and the test set is shown in fig.5, showing how the DC-gain and transient behavior depend on the size and the sign of the output.

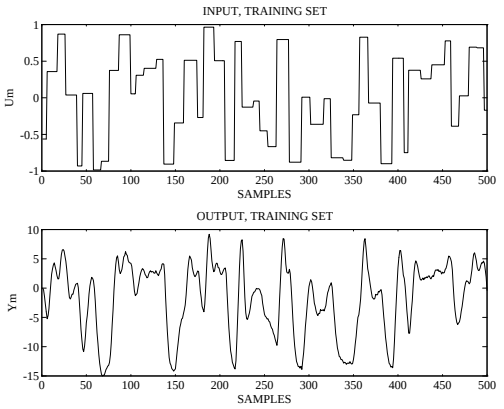

Figure 4: Training set for non-linear test process

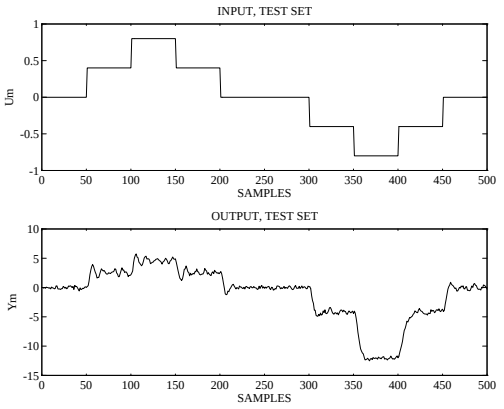

Figure 5: Test set for non-linear test process

5 Non-linear System Identification, Prediction and Simulation

The neural network NARMAX model is chosen with 10 tanh-neurons in the hidden layer and with $p = m = 2$. After having trained this NARMAX model with the Recursive Prediction Error Method using a Gauss-Newton search direction based on the training set shown in fig.4, the model is now tested with the test set shown in fig.5. The test concerns the ability of the model to perform *System identification, Prediction and Simulation.*

5.1 System identification

Since the process is non-linear, the identified coefficients are not constant. A pole-zero map is constructed from the actual identified coefficients based on the test set, and the displacements of the poles and zeros are shown in fig.6. This figure has to be compared with the pole-zero map of the simulated process

fig.3, and considering that a non-linear system identification of a non-linear and noisy process has been performed, the result is quite acceptable.

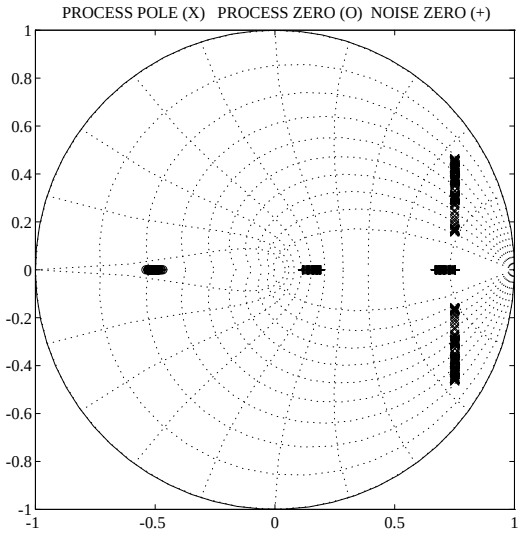

Figure 6: The displacements of process poles (x), process zeros (o) and noise zeros (+). System identification with NARMAX model

5.2 Prediction

In fig.7 the NARMAX model is used as a one step predictor.

Fig.7a shows the measured output Y_m and the predicted output $Y_{m,pred}$, which, in this scale, are almost coincident. The difference is the prediction error.

Fig.7b shows the prediction error E_m, and it 'looks' white.

Fig.7c shows correlation functions. Left, the auto correlation function of the prediction error is shown, and it is obvious that the prediction error is almost white within the 95% confidence interval. Right, the cross correlation function between the prediction error and the input is shown, and it indicates that within the same confidence interval the prediction error does not correlate with the input. These two graphs provide very relevant information, when testing the success of the training, and combined they confirm that the training has been satisfactory.

Fig.7d shows the current variation of the six identified scaled model coefficients $\hat{a}_1, \hat{a}_2, \hat{b}_1, \hat{b}_2, \hat{c}_1$ and $\hat{c}_2$. Only one of them ($\hat{a}_2$) varies, while the other five are almost constant, which corresponds very well to the way in which the non-linearity was introduced in section 4.

The variance of the simulated process noise was (section 4) $\sigma_e^2 = 0.1^2$, and according to [6] the theoretically obtainable value of the variance of the prediction error from a test set, $\sigma_{E,t}^2$, depends on the number of network parameters, d, and the number of samples, N.

$$\begin{aligned} \sigma_{E,t}^2 &= \sigma_e^2 \left(1 + \frac{d}{N}\right) = 0.1^2 \left(1 + \frac{80}{500}\right) = 0.0116 \\ \sigma_{E,t} &= 0.1077 \end{aligned} \quad (15)$$

This result has to be compared with the obtained $\sigma_E = 0.1113$ (see fig.7c), again an indication of successful training.

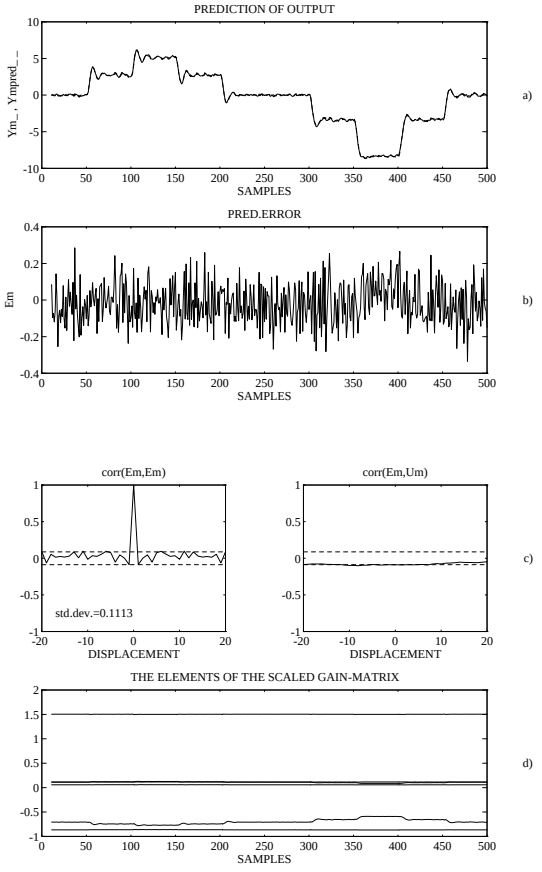

Figure 7: Prediction with NARMAX model

5.3 Simulation

The capability of the trained neural network NARMAX model to act as a simulator will now be demonstrated. For a simulator the process output is not available during the simulation, so instead of fig.2 the simulation is based on fig.8.

This means that the simulator model is

$$
\begin{aligned}
\hat{Y}(k) &= \mathcal{F}\Big(\hat{Y}(k-1),\cdots,\hat{Y}(k-p),\\
&\quad U(k-1),\cdots,U(k-m),\\
&\quad 0,\cdots,0,\theta\Big)\\
E(k) &= Y(k)-\hat{Y}(k) \qquad (16)
\end{aligned}
$$

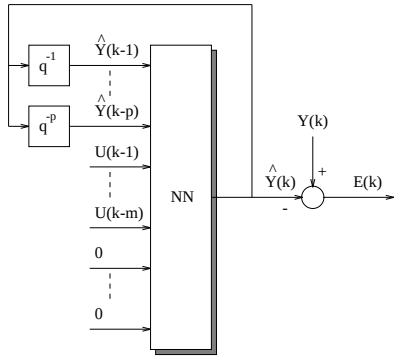

Figure 8: Simulation with NARMAX model

The delayed net outputs are used as inputs, and the correction with the previous prediction errors is dismissed, since E is kept at zero. In fact, this is a true open-loop noise-free simulation. The result is shown in fig.9.

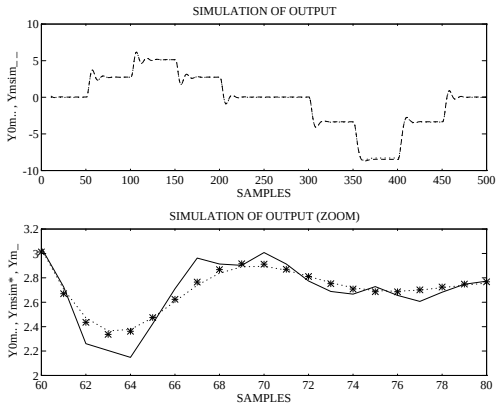

Figure 9: Simulation with NARMAX model

The upper part of fig.9 shows that the simulated noise-free network output $Y_{m,sim}$ with $E = 0$ almost coincident with the simulated noise-free process output Y_{0m} with $E_m = 0$.

In the lower part of fig.9 (a zoom) the solid line represents the measured noisy process output Y_m, the dotted line is the simulated noise-free process output Y_{0m}, and the stars are the simulated network output $Y_{m,sim}$.

It is noticed, that the trained NARMAX model is capable of acting as an almost perfect simulator, completely removing the colored noise from the process.

6 Non-linear Control

The trained neural network, representing a model of the non-linear process, will now be used for an on-line extraction of the actual estimated process parameters, thus allowing an one-line tuning of the controller parameters, depending on the actual process conditions. In fact this method has similarities with a Gain Scheduling method.

The control concept is a pole placement control implemented with a RST-controller of second order. This control concept is well known from linear control theory and is fully described in [8] and [7].

Extraction of the small signal parameters in an actual linearized model neglects the DC-level of the signals, thus demanding an integral action from the controller.

The implementation of the resulting controller, including a neural network model for parameter estimation, is shown in fig.10. All signals in the figure are indicated by capital letters to emphasize that it concerns large-signals. Signals, which are measured in physical units, are supplied with an index 'm', contrary to signals, which are scaled to dimension-less quantities. In section 3.1 it was explained why and how to scale the signals.

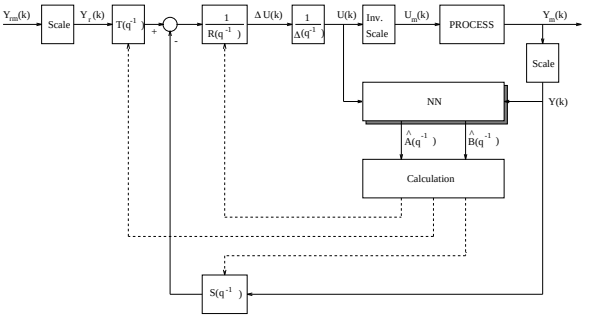

Figure 10: Implementation of RST-controller using a trained neural network for parameter estimation

The process is the non-linear test process from section 4, and the neural network model is the NARMAX model from section 3.1

The scaled actual gain matrix $G(k)$ is calculated by (3) and the actual values of the estimated a- and b-coefficients are extracted from $G(k)$ by (10) in order to calculate the actual controller parameters.

For the desired closed-loop poles, the controller poles are chosen as two coincident real poles in 0.85, while the observer poles are chosen as two coincident real

poles in 0.50.

Fig.11 shows the input and output with this pole placement designed RST-controller, using a non-linear neural network NARMAX model for on-line parameter estimation of the non-linear and noisy process. It is observed from the figure, that the output response, as desired, is critical damped for positive as well as for negative steps, and that the input, as expected, is asymmetric and smoothed.

Fig.12 illustrates, in comparison, what happens, when the parameter estimation is performed from a traditional system identification of a linear model of the non-linear and noisy process. The output behaves quit differently for positive and negative steps. (In fact this result is obtained by using a trained neural network NARMAX model, containing only *linear* neurons, for on-line parameter estimation).

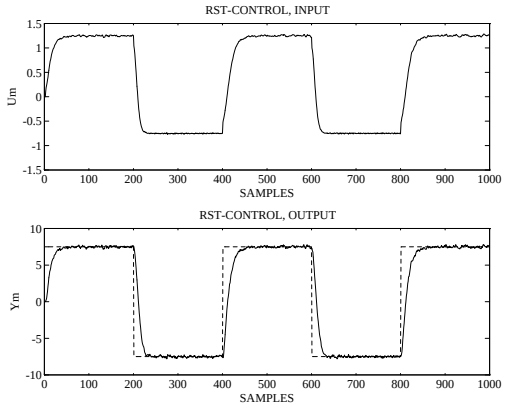

Figure 11: RST-controller using a non-linear NARMAX model

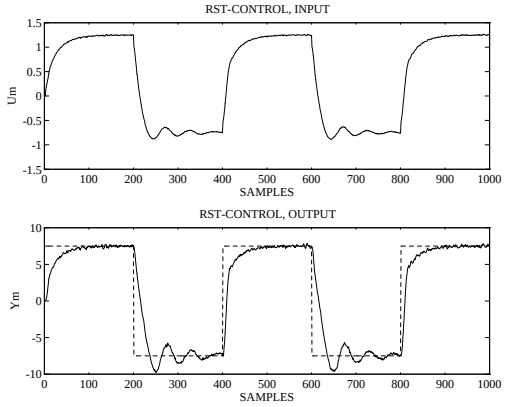

Figure 12: RST-controller using a linear NARMAX model

7 A practical laboratory process

Now a simple single input/single output laboratory process is considered. The process considered is a laboratory setup, in which cold water is led into a cylindrical tank containing an outlet in the bottom. The control purpose is to maintain the water level in spite of several disturbances, mainly the water outlet. The process is equipped with industrial actuators and sensors, but unfortunately they are contaminated with considerably electrical noise. However, the process is accepted as it is. No efforts have been made to establish inner loops around the valves, no filtering of the measured signals are performed, and no knowledge of the process model is assumed. The difficulties are considered to be extra challenges to a neural network based controller.

A serious non-linearity is introduced by placing a fixed ball in the tank. The ball has a diameter approximately equal to three quarters of the diameter of the tanks cross section.

A sketch of the process is shown in fig.13, and the process model structure is shown in fig.14.

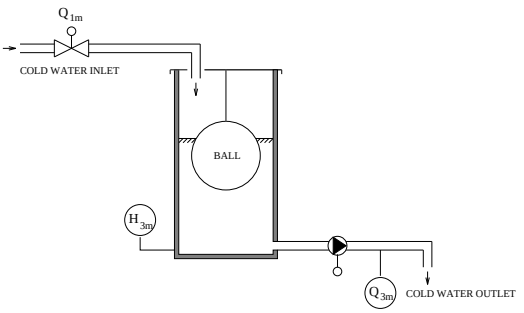

Figure 13: The single input/single output process

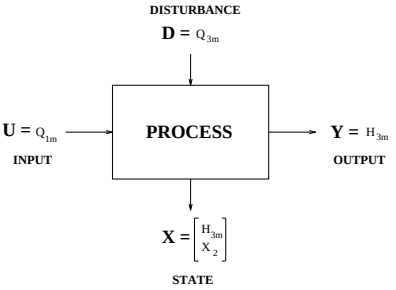

Figure 14: The process model structure

- The controlling input is the voltage Q_{1m} to the valve, supplying the cold water.
- The disturbing input is the outlet flow Q_{3m} of the cold water.
- The measurable output is the water level H_{3m}.
- The state is chosen to second order, the first state being the measurable output H_{3m}.

- The reference for H_{3m} is varying between 0.15 m and 0.425 m.
- The sampling interval is 2 *sec.*

Without any knowledge to the process model, a traditional PI-controller is tuned 'by hand' to give the 'best' performance and the result is shown in fig.15.

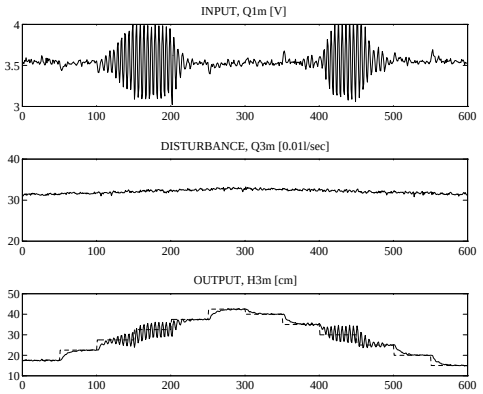

Figure 15: Linear control with a tuned PI-controller. The data are used for training a neural network controller

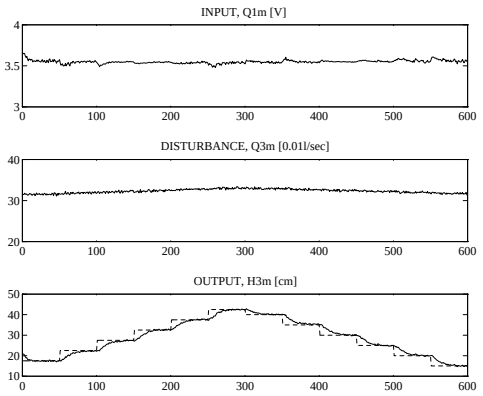

Figure 16: Non-linear control with a neural network performing actual System Identification

It is observed, that the PI-controller is tuned to produce a nice performance for water levels below and above the ball, whereas instability occurs when the water level is passing the ball.

Now the data shown in fig.15 are used to train a neural network model in order to make a better control of the process.

- The Process model is the NARMAX model, shown in fig.2.
- The training algorithm is the Recursive Prediction Error Method (RPEM).
- The number of hidden neurons is 5.
- The control concept is pole-placement control with non-linear neural network System Identification, shown in fig.10.
- For the desired closed-loop poles, the controller poles are chosen as two coincident real poles in 0.85, while the observer poles are chosen as two coincident real poles in 0.50.

An almost perfect control is obtained, as shown in fig.16.

8 Conclusion

The simulated and the practical test examples confirm that a neural network, trained to model a dynamic, non-linear and noisy process, has the potential to perform excellent System Identification, Prediction, Simulation and Control of the process.

References

[1] Kurt Hornik et al. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[2] S. Chen et al. Non-linear system identification using neural networks. *Int. J. Control*, 6:1191–1214, 1990.

[3] Robert Hecht-Nielsen. *Neurocomputing.* Addison Wesley, 1989.

[4] Kumpati S. Narendra et al. Identification and control of dynamical systems using neural networks. *IEEE Transaction on Neural Networks*, 1(1):4–27, march 1990.

[5] S. A. Billings et al. A comparison of the backpropagation and recursive prediction error algorithms for training neural networks. *Mechanical Systems and Signal Processing*, 3:233–255, 1991.

[6] Lennart Ljung. *System identification, theory for the user.* Prentice-Hall, first edition, 1987.

[7] Ole Sørensen. Non-linear pole-placement control with a neural network. *European Journal of Control*, 2(1):36–43, 1996.

[8] Karl J. Åström and Björn Wittenmark. *Computer Controlled Systems: Theory and Design.* Prentice-Hall, second edition, 1990.