Intelligent Predictive Control of Nonlienar Processes Using

Nørgård, Peter Magnus; Sørensen, Paul Haase; Poulsen, Niels Kjølstad; Ravn, Ole; Hansen, Lars Kai

Published in:
Intelligent Control, 1996., Proceedings of the 1996 IEEE International Symposium on

Link to article, DOI:
10.1109/ISIC.1996.556218

Publication date:
1996

Document Version
Publisher's PDF, also known as Version of record

Proceedings of the 1996 IEEE International
Symposium on Intelligent Control
Dearborn, MI • September 15-18, 1996

Intelligent Predictive Control of Nonlinear Processes Using Neural Networks

M. Nørgaard*, P.H. Sørensen*, N.K. Poulsen**, O. Ravn*, & L.K. Hansen**

*Department of Automation, building 326. pmn, phs, or@iau.dtu.dk
**Department of Mathematical Modelling, building 321. nkp, lkh@imm.dtu.dk
Technical University of Denmark, 2800 Lyngby, Denmark

Abstract

This paper presents a novel approach to design of Generalized Predictive Controllers (GPC) for nonlinear processes. A neural network is used for modelling the process and a gain-scheduling type of GPC is subsequently designed. The combination of neural network models and predictive control has frequently been discussed in the neural network community. This paper proposes an approximate scheme, *Approximate Predictive Control (APC)*, which facilitates the implementation and gives a substantial reduction in the required amount of computations. The method is based on a technique for extracting linear models from a nonlinear neural network and using them in designing the control system. The performance of the controller is demonstrated in a simulation study of a pneumatic servo system.

1. Introduction

Neural networks have on many occasions been presented as an excellent generic model structure for identification of nonlinear systems [9]. Utilization of this ability in the design of control systems has thus evolved into being one of the main classes of applications within the field of intelligent control. Over the recent years, many different types of neural network based control systems have been suggested. In some schemes a neural network is trained to act as the controller while in others a more conventional type of design is applied to a neural network model of the process. An example of the latter is the generalized predictive controller (GPC) originally derived for linear process models in [1]. Predictive control is a criterion based design possessing the attractive property that it quite easily can be used in combination with a wide class of nonlinear model descriptions, e.g., neural networks. The idea of using a GPC based on neural network models is not new. See for example [8], [3], and [5]. Unfortunately, practical implementation of the controller is not without difficulties. From a computational perspective, the nonlinear predictive controller has very comprehensive demands. For processes with rapid dynamics this requirement will essentially prevent practical utilization of the controller. In addition there are a number of problems associated with the iterative optimization algorithm used for minimizing the GPC-criterion. Some issues that must be handled in the implementation are:

- Has the algorithm converged to an acceptable accuracy?
- Do numerical problems occur?
- Is the global minimum found (or does it matter that another minimum is found)?
- How many times should one execute the minimization algorithm and what is the best strategy for initializing the algorithm?

Consequently, the nature of the GPC-criterion implies that a number of ad-hoc solutions must be made, leading to a relatively complex controller implementation.

This paper presents an approximate scheme, *Approximate Predictive Control (APC)*, which essentially overcome all the problems mentioned above. The scheme is based on a method for extracting linear models from a nonlinear neural network and using these in the control system design. The idea of linearizing neural networks has previously been proposed in relation to control. [11] suggested an approximate pole placement controller and in [4] it was applied in an internal model control concept. Here the linearization is derived in a manner which allows for direct application of the GPC design method presented in [1]. To illustrate some of its major characteristics, the APC is applied to a pneumatic position servomechanism.

2. Nonlinear Modelling with Neural Networks

In our work we have chosen to restrict the attention to the so-called *MultiLayer Perceptron neural networks (MLP)* for modelling nonlinear processes. However, the particular choice of nonlinear model description is not vital for the controller design. Other types of generic nonlinear model structures might be used instead.

0-7803-2978-3/96/$5.00 © 1996 IEEE

Many different types of MLP based model structures can be considered when identifying nonlinear processes. However, it is typically assumed that the process can be described by the general model

$$y(t) = \hat{y}(t|t-1,\theta) + e(t) = g\big(\varphi(t),\theta\big) + e(t)$$

where $\varphi(t)$ is the regression vector composed of past information, g is the function realized by the MLP-network, θ is the model parameters (the weights), and $e(t)$ is additive noise which is assumed to be white and independent of past information.

By inserting g, which is assumed to be a two-layer MLP network with tangent hyperbolic activation functions in the hidden units and a linear output unit, the predictor takes the form:

$$\hat{y}(t|\theta) = \sum_{j=1}^{q} W_j \tanh\left(\sum_{i=1}^{n_\varphi} w_{ji}\varphi_i(t) + w_{j0}\right) + W_0$$

where the components of θ, W_j and $w_{j,i}$, specifies hidden-to-output layer and input-to-hidden layer weights, respectively.

To reduce the degrees of freedom and because it is advantageous in many control system designs, it is common to consider model structures that are natural extensions of well-known linear model structures like ARX, ARMAX, and OE in that a similar regression vector is considered [9]:

- In the NARX structure the regressors are past inputs and outputs ($d>0$ denotes the time delay)

$$\varphi(t) = \big[y(t-1) \quad \cdots \quad y(t-n) \quad u(t-d) \quad \cdots \quad u(t-d-m)\big]^T$$

- In the NOE structure the regressors are past inputs and predictions

$$\varphi(t) = \big[\hat{y}(t-1) \quad \cdots \quad \hat{y}(t-n) \quad u(t-d) \quad \cdots \quad u(t-d-m)\big]^T$$

- In the NARMAX structure the regressors are past inputs, outputs, and prediction errors

$$\varphi(t) = \big[y(t-1) \quad \cdots \quad y(t-n) \quad u(t-d) \quad \cdots \quad u(t-d-m) \quad \varepsilon(t-1) \quad \cdots \quad \varepsilon(t-p)\big]^T$$

$\varepsilon(t)$ specifies the prediction error: $\varepsilon(t) = y(t) - \hat{y}(t)$.

For NOE and NARMAX model structures the regression vector depends of past predictions. The MLP-network is in this case called a *recurrent* network to emphasize the feedback from the output of the network to the input.

The weights are estimated from a set of corresponding input-output pairs

$$Z^N = \big\{[u(t), y(t)] : t = 1,\ldots,N\big\}$$

acquired in a practical experiment with the process.

The NNSYSID toolbox described in [6] provides a set of tools for inferring models as the above from a set of experimental data.

3. Instantaneous Linearization

[11] details a technique for linearizing neural network models around the current operating point. This *instantaneous linearization* technique is summarized in the following.

Assume that a NARX-model of the process under consideration has been identified:

$$y(t) = g\big(\varphi(t),\theta\big) + e(t)$$

and interpret the regression vector, $\varphi(t)$, as the *state* of the process. At time $t=\tau$ linearize g around the current state $\varphi(\tau)$ to obtain an approximate model:

$$\tilde{y}(t) = -a_1\tilde{y}(t-1) - \ldots - a_n\tilde{y}(t-n) + b_1\tilde{u}(t-d) + \ldots + b_m\tilde{u}(t-d-m) + e(t) - e(\tau)$$

where

$$a_i = -\left.\frac{\partial g(\varphi(t))}{\partial y(t-i)}\right|_{\varphi(t)=\varphi(\tau)}$$

$$b_i = \left.\frac{\partial g(\varphi(t))}{\partial u(t-d-i)}\right|_{\varphi(t)=\varphi(\tau)}$$

and

$$\tilde{y}(t-i) = y(t-i) - y(\tau-i)$$
$$\tilde{u}(t-i) = u(t-i) - u(\tau-i)$$

The derivative of the network output with respect to an input is given by:

$$\frac{\partial\hat{y}(y)}{\partial\varphi_i(t)} = \sum_{j=1}^{q} W_j w_{ji}\left[1 - \tanh^2\left(\sum_{k=1}^{n+m+1} w_{jk}\varphi_k(t) + w_{jo}\right)\right]$$

Separating the portion of the expression containing components of the current state (regression) vector, the approximate model may alternatively be written in the form

$$y(t) = (1 - A(q^{-1}))y(t) + q^{-d}B(q^{-1})u(t) + \zeta(\tau) + e(t)$$

where the *linearization offset*, $\zeta(\tau)$, is determined by

$$\zeta(\tau) = y(\tau) + a_1 y(\tau-1) + \cdots + a_n y(\tau-n) - b_1 u(\tau-d-1) - \cdots - b_m u(\tau-d-m) - e(\tau)$$

and

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n}$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ldots + b_m q^{-m}$$

The approximate model can thus be interpreted as a linear ARX model additionally affected by a disturbance, $\zeta(\tau)$, which depends on the operating point. If the nonlinearities of the process are reasonably smooth, it is not unreasonable to model $\zeta(\tau)+e(t)$ as a random walk process (i.e., integrated white noise):

$$\zeta(\tau) + e(t) = \frac{v(t)}{\Delta}$$

This is a common trick that is often used in control and observer design for modelling approximately constant (or slowly varying) disturbances. The linearized model will in this case correspond to an *ARIX*-model (Integrated ARX).

It is straightforward to apply the same principle for linearizing a more general model like the NARMAX models.

It is obvious that instantaneous linearization is interesting in relation to control. The basic idea is illustrated in fig 1.

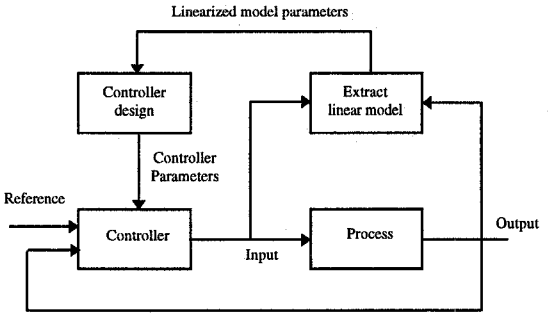

Figure 1. *Instantaneous linearization applied to control system design.*

As pointed out in [11], the structure of the concept is obviously related to the indirect adaptive controller examined in [14]. Instead of recursively *estimating* a linear model at each sampling instant, a linear model is *extracted* from a nonlinear neural network model. Most of the well-known linear design methods fit into this structure. If the model is deterministic one can for example use pole placement and if it is stochastic one can use a minimum variance controller or a GPC. The latter is discussed below.

4. Deriving the Control Law

The idea behind generalized predictive control is at each iteration to minimize a criterion of the following type:

$$J(t,U(t)) = \sum_{i=N_1}^{N_2}\left[r(t+i) - \hat{y}(t+i)\right]^2 + \rho\sum_{i=1}^{N_u}\left[\Delta u(t+i-1)\right]^2$$

with respect to the N_u future controls

$$U(t) = \left[u(t) \quad \ldots \quad u(t+N_u-1)\right]^T$$

and subject to the constraint

$$\Delta u(t+i) = 0, \quad N_u \le i \le N_2 - d$$

N_1 is denoted the minimum costing horizon, N_2 the prediction (or maximum costing) horizon, and N_u the (maximum) control horizon. r is the reference and ρ specifies a weighting factor penalizing variations in the controls. The optimization problem, which has to be solved at each sampling instant, results in a sequence of future controls, $U(t)$. From this sequence the first component, $u(t)$, is then applied to the process.

Assume that an approximate ARX model has been obtained by instantaneous linearization:

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + \zeta(\tau) + e(t)$$

If $\zeta(\tau)+e(t)$ is modelled as integrated white noise and the future predictions are determined to accomplish minimum variance, the setting corresponds exactly to the one considered in [1]. Thus, we refer to this reference for a derivation. An implementation of the controller is described in [7].

Extension to more complex models (such as ARIMAX) and more general criteria is discussed in [2] and [10].

5. Example

A simulation study is now presented to provide some insights into the characteristics of the proposed predictive control strategy. The control object is the pneumatic position servomechanism also studied in [13]. The servomechanism is illustrated in fig. 2.

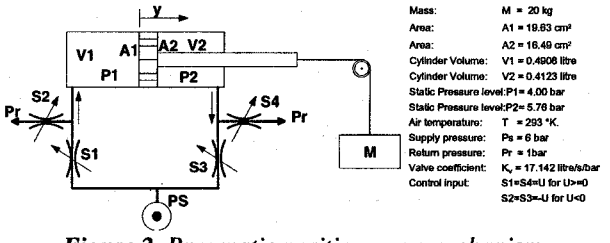

Figure 2. *Pneumatic position servomechanism*

The servomechanism consists of a linear compressed air cylinder lifting an inertial weight. The cylinder is fed from a system of servovalves which open proportional to a control signal. The servovalve opening characteristics are

approximately linear, so that the main nonlinear behavior is due to the cylinder itself. The cylinder chambers compression dynamics are position dependent and the servo valve flow characteristic is nonlinear. The sampling period of the control system is set to 0.1 seconds.

In order to train a neural network model of the servomechanism, an experiment has initially been performed to collect a set of data. Since the servomechanism contains an integration, the experiment was carried out in closed-loop using a manually tuned PI-controller. A data set of 3500 samples was generated. The first 3000 samples were used for training while the remaining 500 samples were kept for validation. The reason for collecting this rather large data set was to ensure the entire range of operation was represented in the data.

A NARX model with 12 *tanh* units in the hidden layer was trained on the data set with a Levenberg-Marquardt algorithm [6]. Due to the size of the data set there was only little sensitivity towards the choice of network architecture. For this reason it was not necessary to consider issues like weight decay and pruning algorithms. The servomechanism is a fourth order system and the network input was thus composed of the four past output measurements and four past controls, respectively.

The trained neural network model was then used in a simulation where the pneumatic servomechanism was controlled by an APC. The design parameters of the controller were selected to:

$$N_1=d=1,\ N_2=10,\ N_u=2,\ and\ \rho=0.05$$

The simulation gave the result displayed in fig. 3. The reference signal was a series of steps with the magnitude varied inside the permitted output range.

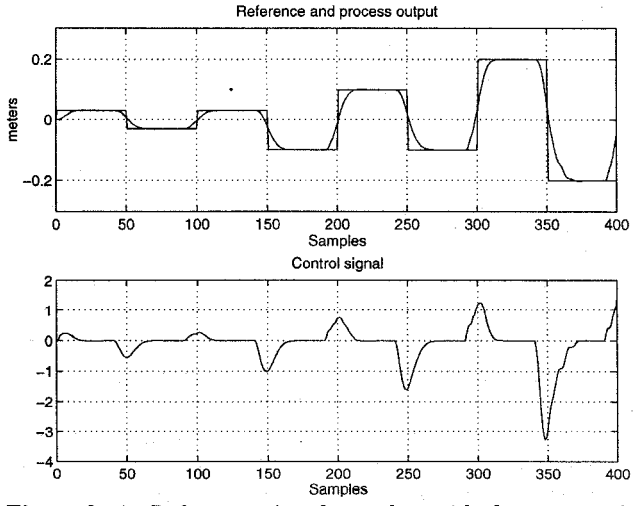

Figure 3. *A: Reference signal together with the output of the process. B: Control signal*

It is seen that a close and offset free reference tracking was achieved for all steps (fig. 3a). Also the predictive nature of the controller can be seen in the response, in that the controller anticipates future set-point changes. Fig. 3b shows that this was accomplished with a very smooth control signal. Moreover, fig. 3.b indicates that the process is in fact nonlinear: The magnitude of the control signal is not proportional to the reference change. To further investigate the nonlinearities, the coefficients of the extracted linear models are displayed in fig. 4. This type of plot gives a good indication of the "degree of nonlinearity."

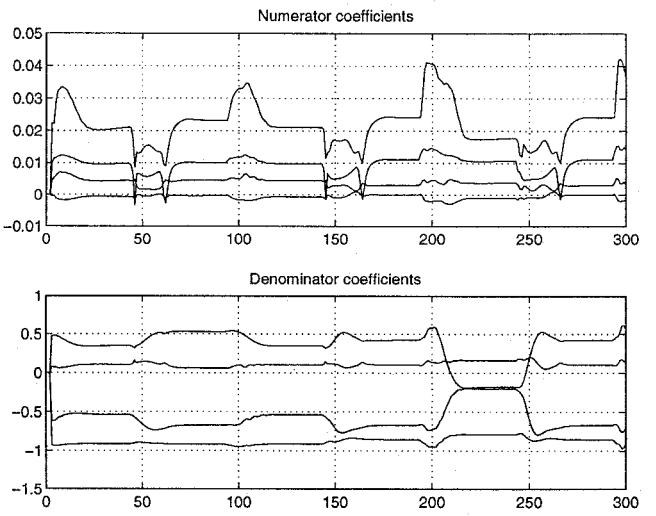

Figure 4. *Numerator and denominator coefficients of the extracted linear models.*

Perhaps a better illustration of the variations in process dynamics is accomplished by showing the location of the poles in the complex plane. This has been done in fig. 5.

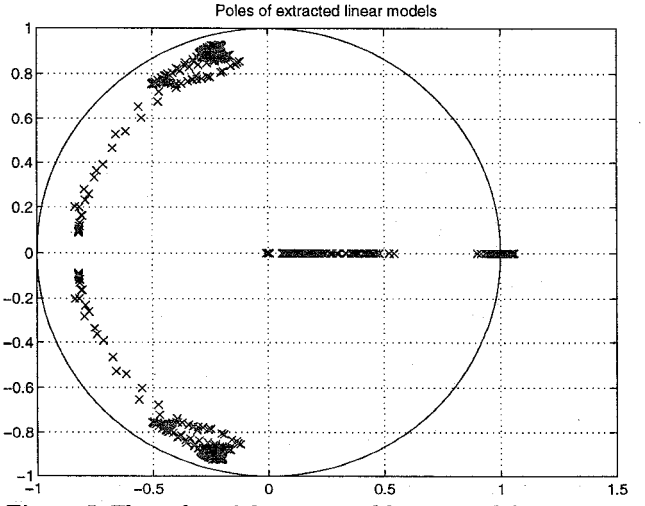

Figure 5. *The poles of the extracted linear models.*

Obviously the process has a complex pair of poles, an integrator and an additional real pole.

However, it is possible to obtain further information from the linearized model. In [12] a continuous-time model of the servomechanism was derived. This model was linearized in the stationary point $(u,y)=(0,0)$, which gave:

$$H(s) = \frac{k}{s(s + 2\zeta\omega s + \omega^2)}$$

where the natural frequency is $\omega \approx 22$ *rad/sec* and the damping factor is $\zeta \approx 0.034$.

As mentioned earlier, the process is in fact of order four. However, in the stationary point $(u,y)=(0,0)$ a zero cancels the additional pole, which results in a third order model. According to [12], it is reasonable to assume that the linearized models in continuous time in general will have the following structure:

$$H(s) = \frac{B(s)}{(s + p_1)(s + p_2)(s + 2\zeta\omega s + \omega^2)}$$

where p_1 is always zero (the integration) and where $\omega \approx 22$ *rad/sec* and $\zeta \approx 0.034$ for small variations around $(u,y)=(0,0)$. $B(s)$ describes the numerator polynomial. The four parameters (ω, ζ, p_1, p_2) determined from the denominator coefficients of the linearized models are shown as functions of time in fig. 6 (natural frequency and damping factor) and fig. 7 (integrator and additional pole).

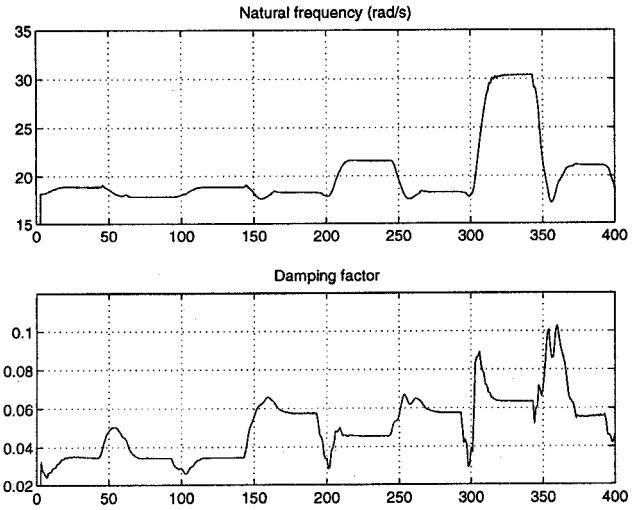

Figure 6. ***A:*** *Natural frequency and* ***B:*** *Damping factor.*

Not only are the values of natural frequency and damping factor (fig. 6) close to their expected values for small outputs, they generally behave as expected for larger variations too.

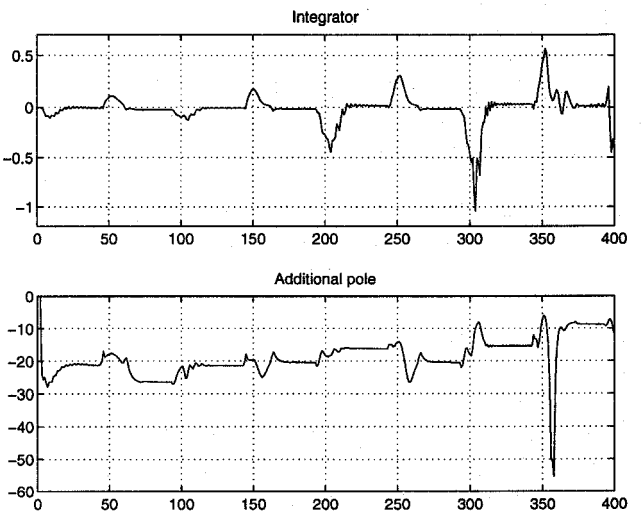

Figure 7. ***A:*** *Integration and* ***B:*** *Additional pole.*

The pole p_1 shown in fig. 7a is close to zero as expected, but significant variations occur when the servo is not in steady state. The physical interpretation of the second pole, p_2 (fig. 7b) is less clear. In the stationary point $(u,y)=(0,0)$ it was cancelled by a zero, which corresponds to the dynamics being partly unobservable.

To briefly summarize the experience gathered in this study, it must be concluded that the proposed controller seem to work well on a nonlinear process. In addition the instantaneous linearization technique provides a valuable physical insight about the dynamics of the process.

6. Conclusions

A design method related to generalized predictive control and which lends itself to control of unknown nonlinear processes has been proposed. The method is characterized by being simple to implement in practice: in comparison to the true nonlinear predictive controller investigated in [5], a substantial reduction in computational effort has been accomplished. In addition, a number of serious drawbacks have been eliminated.

The foundation for the method is the instantaneous linearization technique. Apart from opening up for the application of most linear control design methods, such as GPC, it produces an excellent physical understanding of the process as a spin-off.

Naturally the method has its shortcomings. When the nonlinearities are not reasonably smooth, the linearized models will be valid only in the proximity of the current operating point. In practice this implies that the design will also be highly sensitive to overparameterized models. In fact, it may be advantageous to underparameterize the network deliberately (or use a large weight decay) to

impose a certain smoothness on the network. If the nonlinearities of the process cause the APC design to fail it may instead be used for generating an initial control sequence for the iterative minimization scheme used in the "true" nonlinear predictive controller.

Although this is hard to tell from fig. 3, an interesting property of criterion based designs applied to nonlinear processes is that one must expect that the characteristics of the controller will depend heavily on the present operating regime. In principle it is therefore necessary to verify the behavior of the closed-loop system over the entire operating range. The nature of the inverse and model-reference type strategies discussed in [3] are quite different in this respect in that the desired closed-loop behavior is here imposed directly in the training of a neural network controller.

7. References

[1] Clarke, D.W., C. Mothadi, P.S. Tuffs, "Generalized Predictive Control-Part I. The Basic Algorithm," Automatica, Vol. 23, No. 2, pp. 137-148, 1987.

[2] Clarke, D.W., C. Mothadi, P.S. Tuffs, "Generalized Predictive Control - Part II. Extensions and Interpretations," Automatica, Vol. 23, No. 2, pp. 149-160, 1987.

[3] Hunt, K.J., D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, "Neural Networks for Control Systems - A Survey," Automatica, Vol. 28, No. 6, pp. 1083-1112, 1992.

[4] Lightbody, G., G.W. Irwin,"A Novel Internal Model Control Structure," Proc. of the ACC, Seattle, Washington, Vol. 1, pp. 350-354, 1995.

[5] Nørgaard, M., P.H. Sørensen, "Generalized Predictive Control of a Nonlinear System using Neural Networks," Preprints, 1995 International Symposium on Artificial Neural Networks, Hsinchu, Taiwan, ROC, pp. B1-33-40, 1995.

[6] Nørgaard, M., O. Ravn, L.K. Hansen, N.K. Poulsen, "The NNSYSID Toolbox - A MATLAB Toolbox for System Identification with Neural Networks," 1996 IEEE Symposium on Computer-Aided Control System Design, Dearborn, Michigan, USA.

[7] Nørgaard, M., O. Ravn, N.K. Poulsen, L.K. Hansen, "NNCTRL - A CANCSD ToolKit for MATLAB," 1996 IEEE Symposium on Computer-Aided Control System Design, Dearborn, Michigan, USA.

[8] Saint-Donat, J., N. Bhat, T.J. McAvoy, "Neural Net Based Model Predictive Control." Int. J. Control, Vol. 54, No. 6, pp. 1453-1468, 1991.

[9] Sjöberg, J., H. Hjalmerson, L. Ljung, "Neural Networks in System Identification", Preprints 10th IFAC symposium on SYSID, Copenhagen, Denmark. Vol.2, pp. 49-71, 1994.

[10] Soeterboek, R., "Predictive Control: A Unified Approach," Prentice Hall, 1992.

[11] Sørensen, O., "Neural Networks in Control Applications", Ph.D. Dissertation, Department of Control Engineering, Aalborg University, Denmark, 1994.

[12] Sørensen, P.H., "Report on the Mechanical and Electrical Configuration of a Digital Valve Controlled Pneumatic Position Servo Mechanism," unpublished notes.

[13] Sørensen, P.H., P.K. Sinha, K. Al-Mutib, "Identification of a Pneumatic Servo Mechanism using Neural Networks." Proc. Int. Conf. on Machine Automation, Tampere, Finland, Vol.2, pp. 499-512, 1994.

[14] Åström, K.J, B. Wittenmark, "Adaptive Control," 2nd Edition, Addison-Wesley, 1995.