UNIVERSAL APPROACH TO NEURAL PROCESS CONTROL ILLUSTRATED ON A BIOMASS GROWTH MODEL

J.Jarmulak, E.J.H.Kerckhoffs, L.Rothkrantz

Delft University of Technology
Department of Technical Mathematics and Informatics
Julianalaan 132, 2628 BL Delft, the Netherlands
e-mail: eugene@kgs.twi.tudelft.nl

Abstract: The article presents a control scheme, based on use of neural networks, which can be applied to a control of complex nonlinear systems. The emphasis is on producing methodology and appropriate software which could later be used to construct control systems in a standard way. As usual when neural networks are used many experiments are required before desired results are reached. Therefore, a NeuroControl Workbench has been constructed which allows one to construct numerous experiments with a minimum of effort and to find the most promising control architecture. This article presents some results of control experiments with a bioreactor simulation.

Keywords: adaptive control, artificial intelligence, identifiers, model-based control, neural networks, nonlinear systems, predictive control, software tools.

1. INTRODUCTION

Neural control promises to offer a solution to the problem of controlling complex systems, which are difficult or impossible to model and control using traditional (non-AI) techniques, because of their nonlinearity, time-varying parameters, etc. This applies especially to bio-processes. Neural networks can be used to model and control such systems because of their ability to approximate broad classes of nonlinear functions. Also, because neural networks can be simply trained to achieve desired properties, it is possible that use of neural control will result in reducing the development costs for control systems.
The field of neural control is relatively new, and though already some practical applications of neural networks in process control exist, still much research has to be done before neural control systems will be used on a larger scale. Numerous experiments will have to be conducted because knowledge of neural networks is to a large extent based on heuristics. Through many experiments a better insight can be gained into possibilities and problems related to the use of neural networks for process control. A good understanding of neural control is necessary, before it can successfully be applied in practice.

At the beginning of this article several control architectures making use of neural networks are presented. Those architectures are encountered most often in the relevant literature. Next, the NeuroControl Workbench program will be presented, which has been constructed to experiment with various neurocontrol setups. The results of experiments with a simulated control of a bioreactor system will finally be presented.

2. AI IN PROCESS CONTROL

Most of the control system theory deals with control of linear systems. The problem is that in practice most of the systems encountered in agriculture are non-linear (at least to some extent). When no precise process model can be obtained, the control system will be able to achieve only suboptimal results. In practice, often simple PID controllers will be used. However, good results cannot be achieved by old, suboptimal techniques, therefore, better control systems have to be used. This, in turn, means that somehow ways have to be found to model even difficult, nonlinear processes.

Another problem is the development cost. The common PID controllers are essentially off-the-shelf components therefore the cost of implementing PID control systems is relatively low. More sophisticated controllers often require per-case design, which implies higher development costs. This is the reason why worse than theoretically possible control systems are often used.

Artificial intelligence offers techniques which can be a solution to both the requirements of high-quality control and low development costs. When neural networks are used, the proper control behaviour could be obtained simply by training the controller.

3 NEURAL CONTROL

Neural network theory has been developed in the hope of mimicking that human ability of acquiring same skill in a process of training. A neural network is capable of learning input output relationships just by presenting corresponding input-output pairs to it. Also, similarly as humans, neural networks posses the capability of responding in a more or less correct way when presented with so far unknown input (generalization).

Because of those characteristics, use of neural networks is usually considered when one has to do with input-output relationships that are difficult to capture in a form of mathematical equations or algorithms, especially if occurrence of new inputs can be expected during network use.

In a process control system two mappings are evident candidates to be represented by neural networks. The first one is the process itself, which may be difficult or impossible to model using other techniques. If one is able to obtain a neural network model of the process, it can be later used to find the correct process inputs. Another mapping that can be implemented by means of neural network is the controller itself.

Neural networks are also attractive, because of the possibility to reduce development costs for control systems. Instead of preforming complicated and costly system analysis, using either techniques of traditional control theory, or knowledge elicitation for the expert systems; a control system can be obtained by training the network(s) on example patterns.

3.1 Neural Networks for System Modelling

Availability of a model of the process to be controlled is usually an essential prerequisite of a controller design. For linear time-invariant systems the problem of determining system model (system identification) has been well studied and many methods are available. However, in practice, most of the encountered systems (especially bio-systems) are non-linear. Identification of certain classes of non-linear systems is not impossible, though complex and difficult.

The problem of system identification can be transformed into a problem of learning a mapping between a known input and output space. Neural networks can be trained to represent such mapping.

Model Description. A possible way to describe a system, is in the form of the following discrete equation (u is input, y is output):

$$y(k+1)=f\{y(k),y(k-1),\ldots,y(k-n+1);\; u(k),u(k-1),\ldots,u(k-m+1)\} \qquad (1)$$

This model called NARX (nonlinear ARX) is a very general description, that can be directly used for nonlinear system identification using static neural networks.. The system dynamics are introduced externally in the form of previous values of *u* and *y*.
Other models are of course possible. Some may make use of prior knowledge of the system, thus reducing complexity of the function *f*.

Neural Networks as Identifiers. In (Cybenko, 1989) it has been shown that a feedforward network with one hidden layer, having a sigmoid as the activation function, is capable of approximating any continuous function ^{N} ^{M}, with any desired accuracy. Now, it is possible to construct a neural identifier which consist of a static neural network and tapped-delay-lines (TDL's, sometimes also called time-delay-lines) which provide old plant input and output values for the network input, see fig. 1. This setup can be used for modelling of dynamic systems. As the neural network one can use, for example, a multilayer neural network of McCulloch-Pitts type (Aleksander, 1990), a radial basis function network, or a Cerebellum Model Articulation Controller (CMAC) (Krijgsman, 1992).

Instead of static, dynamic networks can be used for plant modelling. The dynamic character can be achieved by presence of internal recurrent connections, or by using a dynamic transfer function in the network nodes. When such dynamic networks are used, no TDL's are necessary.

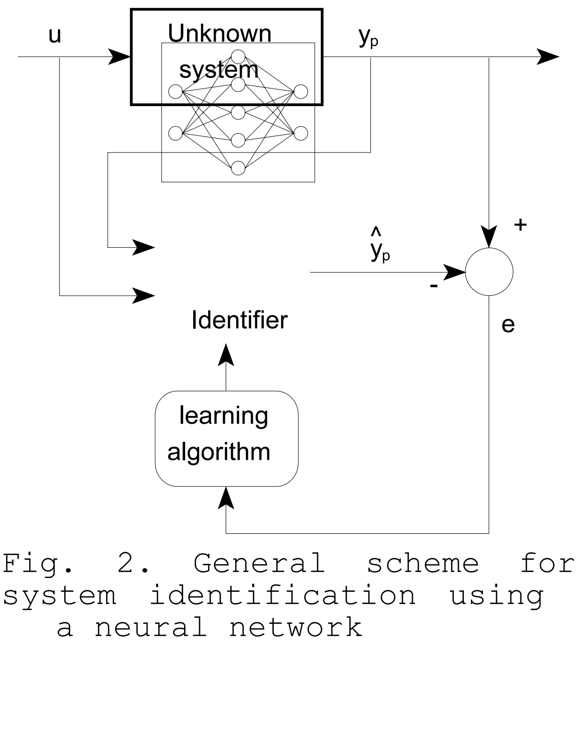

Fig. 2. General scheme for system identification using a neural network

Training Neural Identifier. A general scheme for plant identification using neural networks is shown in fig. 2. The identification can be done on-line (as suggested by the schema), i.e. during the run of the system, however most often corresponding values of the plant input and output are collected, and the identifier network is then trained off-line. In order to obtain a good model of the plant it is necessary that the training data spans the whole operating range of the plant. This implies that any on-line identification should take care that there is enough variation in the input signal (excitation condition (Wittenmark and Aström, 1989)).

3.2 Neural Control Architectures

This section presents several control architectures which make use of neural networks. Most of them use a neural network to represent a forward or inverse plant model. They are essentially a result of combining traditional control theory with the use of neural networks. A conceptually different type of neural controller is the controller which makes use of reinforcement learning, which, ideally, should mimic the way biological control systems work (Bersini, 1992).

Direct-Inverse Controller. In direct-inverse controller the control signal is found by passing the desired plant output through the inverse plant model. If an exact plant inverse is available this leads to dead-beat control. The inverse model is obtained similarly as shown in figure 2, but with the roles of *u* and y_p reversed.

Inverse control is usually avoided in the traditional control-system design. One of the reasons is the frequently occurring instability

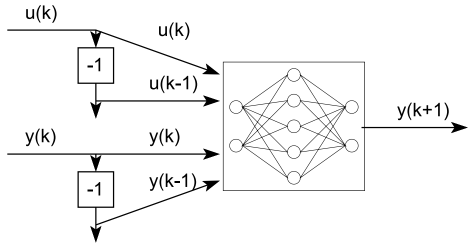

Fig. 1. Static feed-forward network for modelling of dynamic systems

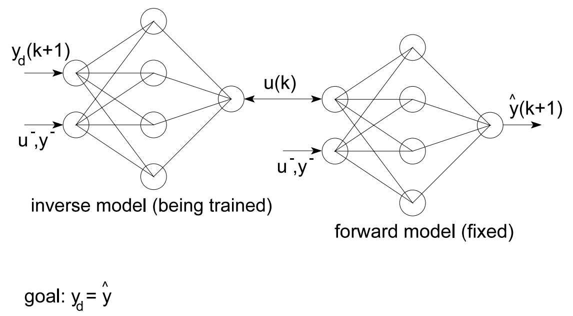

Fig. 3. The idea of obtaining an inverse model by forward modelling

of the plant inverse. Use of neural networks adds new problems to the inverse control, see (Jarmulak, 1994). This leads to the conclusion that the direct-inverse controller is without practical importance.

Forward-Modelled Inverse Controller. In this type of controller the inverse model of the plant is obtained during a two phase process. First, a forward model of the plant is obtained. Then the network which is to learn the inverse plant model is placed before the forward neural model, see fig. 4. The combined network is then trained to minimize error between the setpoint $y_d(k+1)$ and the plant output as predicted by the neural model $\hat{y}(k+1)$. During training, the weights of the forward model remain unchanged, only the weights of the inverse network are adjusted. This method, called forward modelling, has been reported to make the first network converge to an inverse when the plant is invertible. If the plant is not invertible then the first network still converges to a good controller.

If the combination of the controller and forward model networks is being taught to approximate not an identity transformation, but some dynamic system (reference model), a model reference controller can be obtained.

Iterative Inversion Controller. The controllers described in the previous sections are nothing else than (approximate) inverse models of the plant. The inverse model lets us find (in one calculation) the plant input value that results in the desired plant output.

If a plant model is available, one could also attempt to find the correct plant input by a trial-and-error method, that is by trying several input values in order to determine the one for which the output of the plant model is equal (or close enough) to the desired plant output. If the whole search procedure is done in a systematic manner, so that in every iteration it brings plant output closer to the desired value, we obtain the inverse in an iterative manner (Hoskins,1992).

The disadvantage of this type of controller is that the iterative procedure may require a lot of time to execute. However, it is possible to train a neural network using the results of the iterative inversion. Even, if that network provides only approximately good results, they can be good starting points for the iterative inversion procedure, in this way reducing time necessary to calculate the inverse.

Model-Based Predictive Controller. In a model-based predictive controller, a plant model is used to predict (over a certain prediction horizon) future values of the plant output ($\hat{y}(k+1)$, $\hat{y}(k+2)$,...) resulting from a certain sequence of the plant input ($u(k)$, $u(k+1)$,...). Those predicted values are used to calculate the value of a chosen cost function. The objective is to find such a sequence of plant inputs for which the cost function has its minimum. The, so calculated, plant input $u(k)$ will be applied to the plant. After the next plant output $y(k+1)$ is known, the whole procedure is repeated.

Usually, it is assumed that the plant input remains constant over the whole prediction horizon (constant-future strategy), this significantly simplifies the problem of finding the minimum, compared to a strategy which puts no restrictions on the input trajectory.

Typical cost function might look like this:

$$H_p \quad i \ 1 \quad (\boldsymbol{y}_d(k\ i)\ \hat{\boldsymbol{y}}(k\ i))^2 \quad \Delta^2\boldsymbol{u}(k\ 1\ i)\ (1)$$

Instead of simply subtracting the vectors, it is possible to attach importance factors to every element of the vector. This provides a very flexible way to influence the behaviour of the whole system.

For systems of which a linear model is known, there are analytical solutions to the problem of finding the minimum of the cost function. This has the advantage of giving a simple (at least in calculation) formula for the correct plant input.

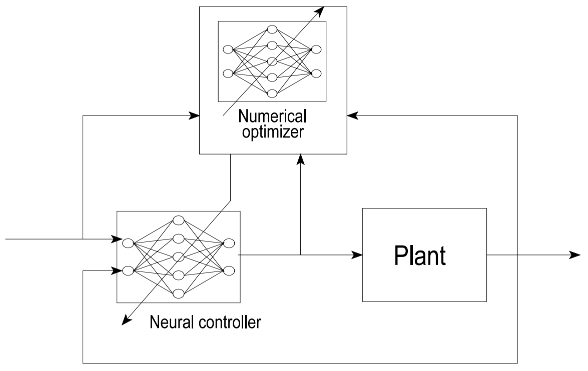

Fig. 4. Model-based predictive controller (parallel setup)

When the available model is in the form of a neural network mapping, the minimum of the cost function has to be found by means of a numerical algorithm. This has both advantages and disadvantages. A very significant advantage of the numerical methods is that the cost criterium can be much more complicated than the example given above, taking various factors into account. However, numerical minimization algorithms are usually very time consuming (especially if a minimum of a multivariable function has to be found), what may make them unsuitable for certain real time applications.A solution to the calculation-time problem may lie in training a neural network to provide the same results as those calculated numerically, a situation similar as described in the previous section.

The run-time predictive control architecture could also be designed like in fig. 5. Here the plant output is calculated by a single evaluation of the previously trained controller network. The numerical minimizing procedure runs parallel to the system and keeps on training the controller.

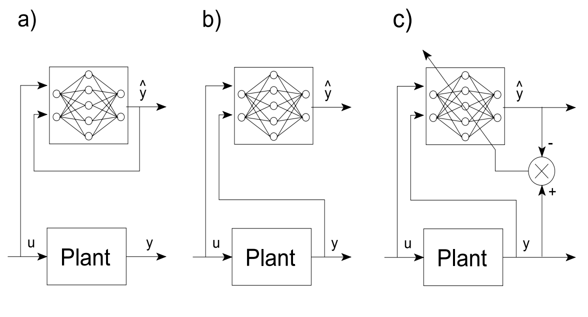

Fig. 5. Methods of run-time connection of the identifier: a) free-running (series-parallel), b) parallel, c) adaptive

Controller and Identifier During Run-Time. Essentially, there are two ways the identifier can be connected within the system: it can be free-running (sometimes called series-parallel connection), or connected parallel to the plant, see fig. 6 a&b. Almost always the parallel connection will be used. If the identifier is free-running, then inaccuracies of the model tend to accumulate, which can lead to unacceptable differences between identifier and plant output.

A situation when the free-running connection may be of advantage occurs when due to, for example, large measurement noise, there is more confidence in the neural network output then in the measurement of the plant output. Of course some precautions have to be made against identifier diverging from the plant.

The identifier training may be continued on line. If the plant changes its behaviour during the run-time, the identifier network will be able to adapt to the change. However, for most of the neural network types, the adaptation will be rather slow. All of the neural control architectures described in the previous sections can be made adaptive by training the identifier on-line. Figure 6 c shows schematically the adaptive connection.

On-line identifier training has to be done with caution. Overtraining of the identifier network in a small area of the operating range may distort the identifier mapping in other areas. To prevent those mapping distortions, low values of the learn rate and momentum have to be used. A more computationally costly solution to identifier overtraining would be to train the identifier on a set of several past input-output patterns - a "sliding window" method.

If a controller network is present in the controller setup, it is almost always trained during the run-time (on line). This may make the controller respond better within the actual operating range. However, again, the danger exists that the controller will become overtrained on the limited range of values and may respond incorrectly when the system moves to other part of the operating range. Fortunately, the distortion of the controller network mapping does not have as serious consequences as the distortion of the identifier mapping, because the controller system contains feedback connections. The controller performance may well suffer but the consequences of distortion are usually not disastrous.

Other Neural-Control Architectures. Other control architectures are of course possible. For example, one could combine a traditional controller, which itself guarantees a good, though not optimal performance, with a (parallelly connected) neural controller which would be trained to optimize the working of the traditional controller. In this setup an incorrect output of the neural part would not have disastrous results.Other ideas for neural control can be found in (Hunt *et al.*, 1992) and in (Narendra and Parthasarathy, 1990).

Neural networks may also be used to perform on-line estimation of certain process parameters, which either cannot be directly measured, or whose on-line measurement is not possible, see for example (Willis et al., 1992).

4. NEUROCONTROL WORKBENCH

The previous sections already demonstrated the need for use of neural networks in process control. However, before neural controllers are used on a wide scale in real-world setups still a lot of research has to be done. The reason for this is that though some theoretical results are available, they usually make only vague statements about how they will translate into practical implementations. For example, the already mentioned theorem of Cybenko, makes no statements about the number of nodes necessary to model a function. The practical implementation will

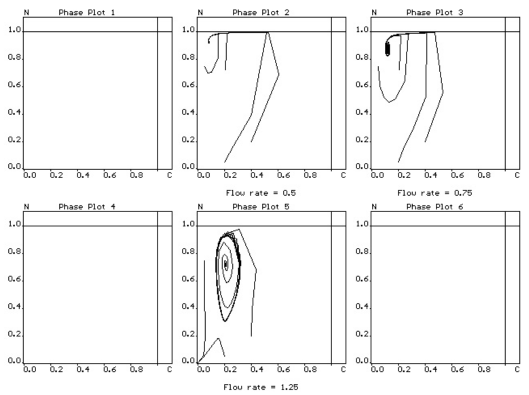

Fig. 6. Bioreactor phase plots for various values of flow. Plots 1-4 show an attractor, 5 shows spiral source and saddle point, 6 wash-out mode.

Table 1 Control architectures available in the Workbench

Control architecture	Neural models used	Can be adaptiv e	For plant type
PID	None	No	SISO
Direct inverse	Inverse	Not well	MIMO
Forward-modelled inverse	Forward & Inverse	Yes	MIMO
Iterative inversion	Forward	Yes	MIMO
Model-based predictive	Forward	Yes	MIMO

however have to do with only a limited number of nodes.

In 1993 a small project dealing with surveying possibilities of controlling a bioreactor was done by Loke and Cembrano (1993). During the project, it became clear that using standard software tools (e.g. NeuralWorks, ACSL simulation tool, C) for research in neural control requires too much time, therefore a NeuroControl Workbench has been constructed (Jarmulak, 94).

The Workbench, which is a MSDOS program, allows the user to experiment with various neural-control setups in a flexible way. To achieve this there is a choice of several plant simulations which can be controlled. Some of the plants are predefined (e.g. a bioreactor, a water column), while it is also possible to define completely new plants. For the experiments one can choose from 5 control architectures listed in table 1.
The neural network type implemented in the Workbench is the common backpropagation network. The user can construct networks with maximally 3 hidden layers. The transfer function used is the hyperbolic tangent.

The workbench is designed in a object-oriented way, both internally and externally (user interface). The user can manipulate any of the six main workbench objects separately; those objects are: plant, identifier, controller, training set, setpoint generator, and run-time display. The way those objects interact with one another depends on the chosen control architecture.

5. CONTROLLING BIOREACTOR

The Workbench has been successfully used to control the bioreactor simulation as described in (Anderson and Miller, 1990). The bioreactor is a constant volume, isothermic, continuous flow, stirred tank reactor. It contains water, biological cells and nutrient. Water, containing nutrient, is continuously added and stirred into the contents of the tank. The volume of the solution in the tank is maintained at the constant level by removing tank contents at a rate equal to the incoming rate. This flow rate *w* is the variable with which the bioreactor may be controlled. The state of the bioreactor can be expressed by the amount of cells *C*, and the amount of nutrient *N* in the tank at a certain time.

The differential equations modelling the dynamic behaviour of the system are as follows:

$$\frac{dC}{dt} \quad Cw \quad C(1 \quad N)e^{\frac{N}{\gamma}}$$

$$\frac{dN}{dt} \quad Nw \quad C(1 \quad N)e^{\frac{N}{\gamma}}\frac{1 \quad \beta}{1 \quad \beta \quad N}$$

in which β and γ are constants that determine the rate of cell growth and nutrient consumption. *C* and *N* are, respectively, dimensionless cell mass and substrate concentrations.

The dynamics of the system are highly nonlinear. Additionally, the presence of unstable behaviour in some operating conditions, as well as limit cycles (oscillations) make the control problems challenging. Figure 8 shows phase plane trajectories of this system for various values of *w*.

To model the bioreactor system a network with a 8-20-20-2 configuration was used. The network input (8 nodes) consisted additionally from *C(k), C(k-1), C(k-2), N(k), N(k-1)*, *N(k-2),* $C_d(k)$ and $N_d(k)$. A training set of 400 patterns generated random over part of the operating range (*C[0.0, 0.5]* and *N[0.1, 0.9]*) was used to train this network. A single input-output pattern was generated by choosing a random plant state and, beginning from it, running the plant for enough iterations, so that all the identifier TDL's get filled. For each iteration a new random input is applied to the plant.

A batch training method (cumulative delta rule) was used to train the identifier network off-line. After ca. 1200000 training iterations (3000 epochs within 30 minutes on 60Mhz Pentium) an RMS error of 0.02 (1% of the output range of the network) was reached. During the system run the identifier error remained below this 1% of operating range for most of the time.

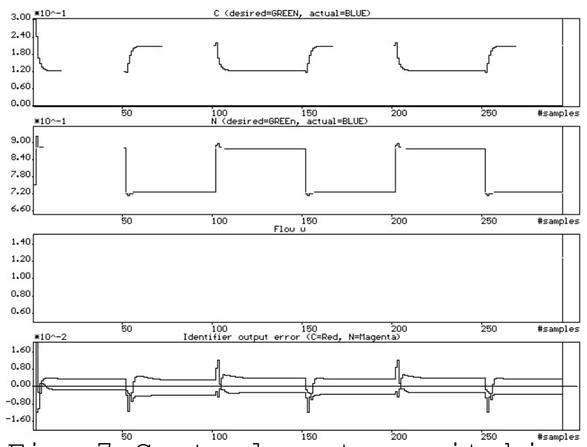

Fig. 7.Control system switching between setpoints (C,N) equal (0.1207,0.8801) and (0.2107,0.7226)

A model-based predictive controller was used to control the bioreactor, as it was found to give by far the best results (Jarmulak, 1994), and the greatest flexibility in specifying the control-system behaviour (by means of the cost-function factors). Very important is also a fact that there is a lot of traditional control theory about predictive controllers. This theory can be used to analyze the stability properties of predictive controllers that use neural plant models.

The prediction horizon was chosen equal 1. The minimization algorithm used in the numerical part of the controller comes from (Powell, 1964). Figure 9 shows the control simulation switching between two of the three setpoints listed in (Anderson and Miller, 1990). As can be seen the results are very good.

6. CONCLUSIONS

It is easy to notice that when a proper software tool is available, a construction of a control system which uses neural networks becomes a relatively easy task. Of course, the methods of system modelling still have to be improved to guarantee that the obtained model is always close enough to the real system. Special attention has to be payed to those areas of the state space where output values remain almost constant, and where the mapping of the gradients becomes more important than mapping of the absolute output values. The current work on the Workbench goes into the direction of making the process of learning the mapping more robust.

An aspect, common to all the applications of neural networks is the computation intensiveness. This makes it difficult to apply neural networks to control electro-mechanical systems, which usually require fast response times. In the bioprocesses however, the time scale is much more slower, leaving enough time to calculate the correct control signal in between the sample times. It can therefore be expected that, the first applications of neural control on a large scale will be to control biological systems.

REFERENCES

Alexander, I., H. Morton (1991). *An Introduction to Neural Computing*, Chapman & Hall

Anderson, C.W., W.T. Miller (1990). Challenging Control Problems. In: *Neural Networks for Control* (T.Miller, R.S.Sutton, P.J.Werbos, Ed.), MIT Press

Bersini, H. (1992). Reinforcement learning and recruitment mechanism for adaptive distributed control. In: *Proceedings of the 1992 Symposium on Artificial Intelligence in Real-Time Control*, preprint, pp.331-337, Delft, the Netherlands.

Aström, K.J., B. Wittenmark, (1990). *Computer-Controlled Systems. Theory and Design*, 2nd ed., Prentice-Hall.

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Mathematical Control Signals and Systems*, Vol. 2, pp. 303-314.

Hoskins, D.A., J.N. Hwang, J. Vagners (1992). Iterative Inversion of Neural Networks and Its Application to Adaptive Control. *IEEE Transactions on Neural Networks*, Vol. 3, No. 2, March 92

Hunt, K.J., D. Sbarbaro, R. Zbikowski, P.J. Gawthrop (1992). Neural Networks for Control Systems - A Survey. *Automatica*, Vol. 28, No. 6, Nov. 92

Jarmulak, J. (1994) *NeuroControl Workbench*, Master degree thesis, Delft University of Technology, the Netherlands

Krijgsman, A.J. (1992). Associative Memories: The CMAC approach. In: *Applications of Artificial Intelligence in Process Control* (L. Boullart, A. Krijgsman, R.A. Vingerhoeds, Ed), Pergamon Press

Loke, R.E., G. Cembrano (1993). *Neural Adaptive Control of a Bioreactor*, Master degree thesis, Delft University of Technology, the Netherlands

Narendra, K.S., K. Parthasarathy (1990). Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, Vol. 1, No. 1

Powell, M.J.D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7, pp. 155-162.

Willis,M.J., G.A. Montague, C. di Massimo, M.T. Tham, A.J. Morris (1992). Artificial Neural Networks in Process Estimation and Control. *Automatica*, Vol. 28, No 6, pp. 1181-1187.

Wittenmark, B., K.J. Aström (1989). *Adaptive Control*, Addison-Wesley