Original paper

Soft Computing 6 (2002) 244 – 253 © Springer-Verlag 2002
DOI 10.1007/s005000100150

A dynamically-constructed fuzzy neural controller for direct model reference adaptive control of multi-input-multi-output nonlinear processes

Y. Frayman, L. Wang

Abstract Conventional industrial control systems are in majority based on the single-input-single-output design principle with linearized models of the processes. However, most industrial processes are nonlinear and multivariable with strong mutual interactions between process variables that often results in large robustness margins, and in some cases, extremely poor performance of the controller. To improve control accuracy and robustness to disturbances and noise, new design strategies are necessary to overcome problems caused by nonlinearity and mutual interactions. We propose to use a dynamically-constructed, feedback fuzzy neural controller (DCF-FNC) from the input–output data of the process and a reference model, for direct model reference adaptive control (MRAC) to deal with such problems. The effectiveness of our approach is demonstrated by simulation results on a real-world example of cold mill thickness control and is compared with the performances of the conventional PID controller and the cascade correlation neural network (CCN). Exploiting the advantage of intelligent adaptive control, both the CCN and our DCF-FNC significantly increases the control precision and robustness, compared to the linear PID controller, with our DCF-FNC giving the best results in terms of both accuracy and compactness of the controller, as well as being less computationally intensive than the CCN. We argue that our DCF-FNC feedback controller with both structure and parameter learning can provide a computationally efficient solution to control of many real-world multivariable nonlinear processes in presence of disturbances and noise.

Keywords Feedback fuzzy neural controller, Dynamic controller structure, Direct MRAC, Cold mill thickness control

1 Introduction

Industrial processes are often multivariable and have strongly nonlinear and time varying behavior. There also exist strong mutual interactions between process variables. Conventional control of industrial processes usually uses simple linear or linearized models to approximate the processes. For multi-input-multi-output (MIMO) processes it is normally very difficult to derive accurate models, due to complex nonlinear relationships among variables, time dependent changes in process dynamics, and model uncertainties when dealing with some physical phenomena. The severe nonlinearity and complexity of the processes thus result in large robustness margins, and in some cases, extremely poor performance of the controller. It is therefore necessary to develop solid control methodologies that are capable of coping with both nonlinearities and interactions, as well as time varying processes under the strong influence of disturbances and noise. In addition, nonlinear control schemes that employ more realistic and more complex descriptions for nonlinear processes require process models in the form of nonlinear differential equations. This limits its industrial applications, since such first principles models are not readily available in industrial practice due to a chronic lack of detailed and extensive process knowledge required for the development of these models.

We are especially interested in control of ill-defined, nonlinear, time varying, MIMO processes in the presence of process disturbances and measurement noise, such as the cold rolling mill, with a minimum amount of human intervention. Therefore, it is desirable to integrate an intelligent component to increase the flexibility of the control system, e.g., to be able to extract relations from the process, to change relations to improve control precision, to reason and to learn. As we shall show, a direct adaptive intelligent controller using process input–output data with both structural and parameter tuning, such as the dynamically-constructed, feedback fuzzy neural controller (DCF-FNC) discussed in the present paper, would fulfill the above objectives.

Most of the existing research in neural or fuzzy neural control has been concentrated on indirect control schemes, where the neural network or the fuzzy neural network is used to identify the process, with an open-loop mode of operation and static feedforward networks, and a controller is subsequently synthesized from this model [4, 20, 23, 28, 34, 35, 43]. We thus follow a different approach of direct model reference adaptive control (MRAC) [2] where a fuzzy neural network (DCF-FNC) *is the controller* and *no models of the process* are required (some of the present work has been briefly reported in [14, 15]).

In practical control applications it is necessary to use feedback controllers to deal with temporal changes of process parameters. In real-world process control the ability of the controller to adapt to process changes is vital. In this work we use a feedback controller to deal with temporal changes of process parameters.

Y. Frayman, L. Wang (✉)
School of Electrical and Electronic Engineering,
Nanyang Technological University, Block S2,
Nanyang Avenue, Singapore 639798
e-mail: elpwang@ntu.edu.sg

Our DCF-FNC requires two types of tuning: structural tuning and parameter tuning. Structural tuning concerns with the structure of the controller: the number of input–output variables, partition of each input variable universe of discourse (the number of membership functions), the number of hidden (rule) nodes, and the logical operation to compose hidden (rule) nodes. Parameter tuning concerns with adjustments of the position and the shape of membership functions (fuzzy weights). Other existing feedback fuzzy neural approaches (e.g., [18, 26]) concentrated on parameter tuning, selecting the structure on a trial-and-error basis. However, there are no simple ways to determine in advance the minimal size of the partition of the input's universe of discourse or the minimal size of the hidden (rule) layer necessary to achieve the desired performance. This type of structure selection may require in-depth knowledge about the underlying nonlinear process which is seldom available [40]. Another problem with a trial-and-error approach for structure selection [3, 18, 24, 26, 27, 36, 41] is that it increases the danger of over-parameterizing the network, which may lead to fitting noise into the network, and consequently to poor generalization ability [42].

Thus, to use a fuzzy neural controller (FNC) for practical nonlinear processes, one needs to dynamically determine its structure, i.e., to find the smallest network structure capable of achieving an optimal performance. Although some authors (e.g., [26]) claimed automatic construction of feedback FNCs, in reality, they used a predefined partition of the input university of discourse (predefined number of membership functions). This resulted in *completely predefined* structure of the FNC. Consequent FNC learning does not affect this predefined structure of the network. There exist some self-constructing FNCs (e.g., [28, 33]). However the use of two-stage sequential learning process (one for structure determination and another for control) limits their industrial applications as the controller will not be operational during the structure learning. In addition, in [28] learning is done from inputs and desired outputs of the controller. The desired outputs of the controller are not available in advance, that makes learning from the inputs and outputs of the controller impractical.

Similarly with neural networks, there are no simple ways to determine in advance the minimal number of hidden layers and the minimal size of each hidden layer necessary to achieve a desired performance. It is not uncommon to consider many architectures to determine the appropriate one. In order to make an multilayer perceptron (MLP) useful for real-time control, the network architecture has to be determined dynamically. There are several learning algorithms that construct the architecture during learning [1, 10, 16, 30, 31]. One particular self-constructing algorithm, the cascade correlation network (CCN) [10], was selected for comparisons with our DCF-FNC.

A direct MRAC based on our DCF-FNC offers a method for automatic discovery of an efficient controller. As we will show, such an approach is able to achieve good robustness in time varying environments through continuous adaptation. The learning of our DCF-FNC is different from the learning commonly used in neural control in that our DCF-FNC controller learns from a direct evaluation of accuracy with respect to the *outputs of the process* rather than from the *inputs and desired outputs of the controller* as e.g. in [28]. An important feature of the feedback structure of our DCF-FNC is that a convergence to a stable solution is guaranteed, as there is no feedback path between controller outputs and controller inputs [42].

The overall DCF-FNC is a global nonlinear function approximator which is linear in parameters with each combination of input node, rule node and output node acting as a local linear approximator. Each controller input is associated with corresponding fuzzy sets. Through logical combinations of these fuzzy inputs, the controller input space is partitioned into several fuzzy regions. A local controller is used within each region. The global controller output is obtained through smooth interpolations of the outputs of the local controllers. A major advantage of approximators that are linear in parameters is that for squared types of error functions, as the one we use, there is a unique global minimum [12, 13].

The local learning of our DCF-FNC is much less computationally intensive and much faster compared to global learning employed by standard feedforward neural networks [42], since in our FNN weight adaptation is done only for the part of the network (local network) that corresponds to a current data pair, whereas in a global network, such as the multilayer perceptron with back-error-propagation learning, the weight adaptation is done for the whole network.

The present paper is structured as follows. Section 2 describes our DCF-FNC, i.e., its learning algorithm and the overall control system structure. Computer simulations of our DCF-FNC on a real-world problem of thickness control in a cold rolling mill, together with comparisons with the standard PID controller and the CCN, are shown in Sect. 3. Section 4 summarizes the results in this paper.

2
DCF-FNC for direct MRAC of MIMO nonlinear processes

A general MIMO nonlinear process can be represented by the following

$$\vec{x}(t+1) = \vec{f}[\vec{x}(t), \vec{u}(t), \vec{d}(t)] \ , \tag{1}$$

$$\vec{y}(t) = \vec{g}[\vec{x}(t), \vec{v}(t)] \ . \tag{2}$$

where $\vec{x} \equiv (x_1, x_2, \ldots, x_n)$ is the process inputs, $\vec{u} \equiv (u_1, u_2, \ldots, u_m)$ are the control signals (the manipulated variables), $\vec{d} \equiv (d_1, d_2, \ldots, d_p)$ are disturbance inputs, $\vec{y} \equiv (y_1, y_2, \ldots, y_m)$ are the process outputs, $\vec{v} \equiv (j_1, j_2, \ldots, j_m)$ are the measurement noises, and t is the sampling time. Here n, m, and p are the dimensions of corresponding vectors. Given that the vector maps $\vec{f}$ and $\vec{g}$ are unknown, a DCF-FNC can be used to control the process with the only assumption that the inputs and the corresponding outputs of the process are measurable.

A reference model is used to designate the desired performance. The desired output response $\vec{y}_d$ is obtained from the output of the reference model subject to the same input $\vec{x}$, disturbance signal $\vec{d}$, and measurement noise $\vec{n}$ as the process under control. The parameters of the model are determined by the desired performance in terms of standard control objectives such as overshoot, settling time, and steady-state error.

The learning algorithm is designed to obtain the correct control signals (manipulated variables) u_l $(l = 1, 2, \dots, m)$ corresponding to the desired process outputs y_{dl}. The learning error ε_l, defined as the difference between the desired process responses y_{dl} and the measured process outputs y_l, is used as a learning criterion (squared error function).

$$\varepsilon_l = \frac{1}{2}(y_l - y_{dl})^2 \quad \cdot \tag{3}$$

$$\begin{array}{llllllllll} & \text{IF} & x_1(t) & \text{is} & A^i_{x_1} & \text{and} & \cdots & x_n(t) & \text{is} & A^i_{x_n} \\ & \text{and} & y_1(t-1) & \text{is} & A^i_{y_{11}} & \text{and} & \cdots & y_{m1}(t-1) & \text{is} & A^i_{y_{m1}} \\ & \text{and} \quad \cdots & y_1(t-c) & \text{is} & A^i_{y_{1c}} & \text{and} & \cdots & y_{mc}(t-c) & \text{is} & A^i_{y_{mc}} \\ & \text{THEN} & u_1(t) = w^i_1 & , & \cdots & , & u_m(t) = w^i_m & , & & \end{array} \tag{5}$$

The learning error ε_l asymptotically approaches zero or a pre-specified small positive value as the iteration number k increases.

$$\|\varepsilon_{lk}(t)\| \to 0 \ \text{ or } \ \|\varepsilon_{lk}(t)\| < \varepsilon_l, \quad t \in [0, T] \ \text{ as } k \to \infty \ , \tag{4}$$

where $\| \cdot \|$ is the norm.

The structure of our DCF-FNC is shown in Fig. 1. The controller has an input layer, a rule layer, and an output layer.

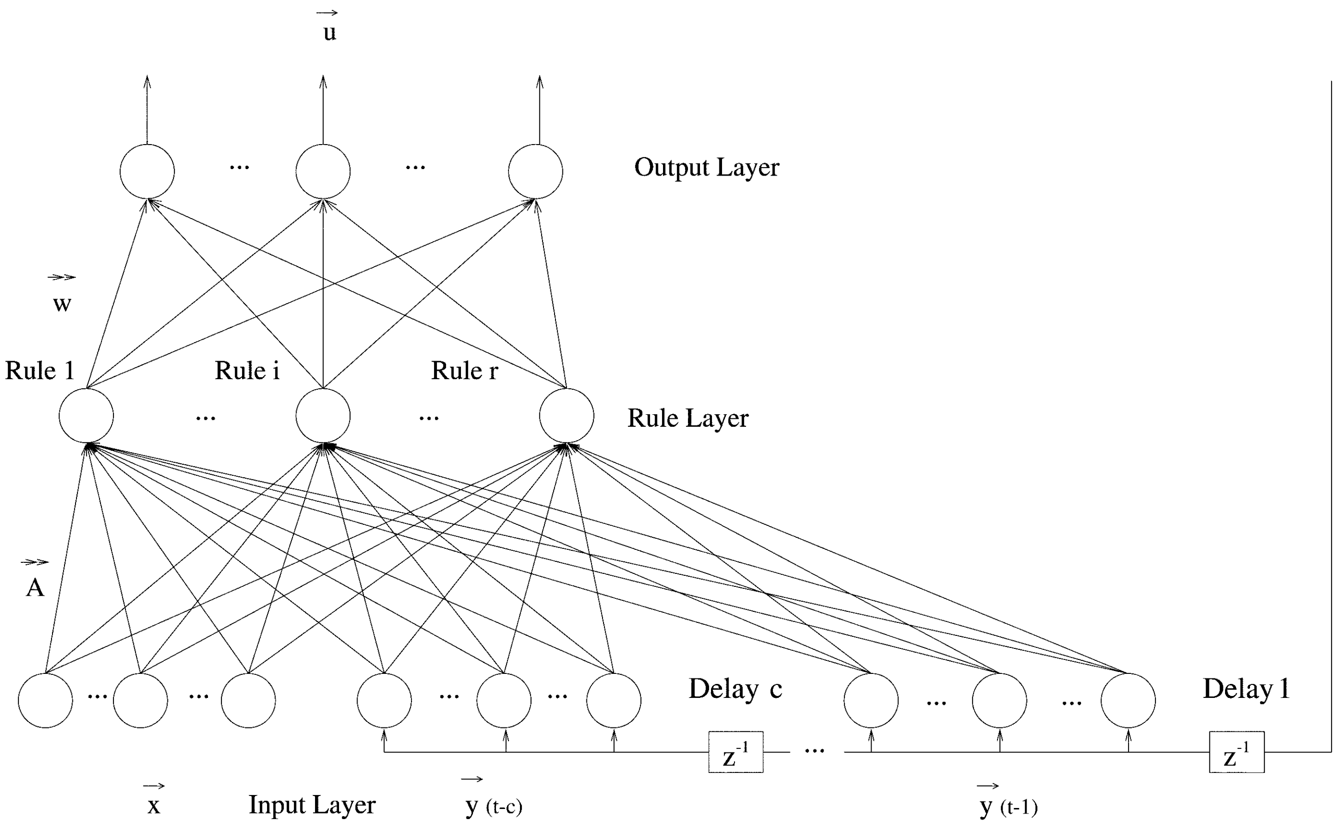
The input nodes represent input variables consisting of the current controller inputs (process inputs) and the previous outputs of the *process* (rather than the network). This feedback structure provides the possibility to include temporal information. The feedback also speeds up the convergence of the controller during learning.

Each input node is connected to all rule nodes through membership functions (fuzzy weights). We use piecewise-linear triangular membership functions. This type of membership functions is simple to implement and is computationally efficient. The leftmost and rightmost membership functions are shouldered.

Rule node i represents fuzzy rule $(i = 1, 2, \dots, r)$, r being the number of rules (rule nodes):

where w^i_l is the weight connecting rule node i and output node l. A^i_q $(q = x_1, \dots, x_n, y_{11}, \dots, y_{mc})$ is the membership function of the antecedent part of rule node i for input node q, and z $(z = 1, 2, \dots, c)$ is the time delay. Each rule node is connected to all input nodes and output nodes for this rule, and these connections are adaptive during learning. The output of rule node i, i.e., the membership

Fig. 1. The structure of our dynamically-constructed, feedback fuzzy neural controller (DCF-FNC)



value μ_i of the premise of the ith rule, is calculated as a fuzzy AND using the product operator

$$\mu_i = A^i_{x1}(x_1) \cdots A^i_{x_n}(x_n) \cdot A^i_{y_{11}}(y_1(t-1)) \cdots \times A^i_{y_{m1}}(y_m(t-1)) \cdots A^i_{y_{1c}}(y_1(t-c)) \cdots \times A^i_{y_{mc}}(y_m(t-c)) \ . \qquad (6)$$

μ_i indicates the degree to which the compound antecedent of the rule is satisfied. The use of the product operator for fuzzy AND produces a smoother control surface, in comparison with the commonly used fuzzy *min* operator [3, 27–29, 36].

In the output layer each node receives inputs from all rule nodes connected to this output node and produces the actual output of the DCF-FNC. Output u_l of the DCF-FNC is obtained using the weighted average

$$u_l = \frac{\sum_i \mu_i w^i_l}{\sum_i \mu_i} \ . \qquad (7)$$

The use of a weighted average (following simplified fuzzy inference [38, 39]) allows us to avoid problems with the commonly used center-of-area (COA) defuzzification that can produce unpredictable results [27–29].

A block diagram of the overall DCF-FNC control system is presented in Fig. 2.

The DCF-FNC structure generation and parameter tuning algorithms are as follows. We generate the training input–output data for the selected reference model. We need to specify the allowable error threshold ε. We also need to specify the maximum number of rule nodes r as an alternative stopping criteria to prevent the controller from growing exponentially if the specified accuracy target is unachievable.

We start with the number of input nodes equal to the sum of the number of input variables and the number of output variables multiplied by the number of delays $(n + c \times m)$, and the number of output nodes equal to the number of output variables (m).

Initially we add two equally spaced, fully overlapping, and shouldered input membership functions along the operating range (universe of discourse) of each input variable. In such a way these membership functions satisfy ϵ-completeness, which means that for a given value of x of one of the inputs in the operating range, we can always find a linguistic label A such that the membership value $\mu_A(x) \geq \epsilon$. If the ϵ-completeness is not satisfied, there may be no rule applicable for a new data input. The initial rule layer is created using Eq. (5).

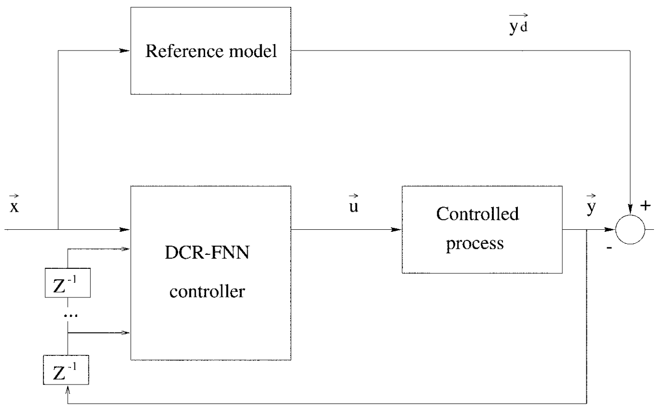


Fig. 2. A block diagram of the overall DCF-FNC control system

The controller is trained using the following learning rules

$$w^i_l(k+1) = w^i_l(k) - \eta \frac{\partial \varepsilon_l}{\partial w^i_l} \qquad (8)$$

for adaptation of the weights between the rule layer and the output layer, and

$$A^i_q(k+1) = A^i_q(k) - \eta \frac{\partial \varepsilon_l}{\partial A^i_q} \qquad (9)$$

for adaptation of membership functions (fuzzy weights) between the input layer and the rule layer. In Eqs. (8) and (9), η is the learning rate. We let the learning rate η vary to expedite convergence and to improve learning accuracy. We start with a relatively large learning rate to enhance the learning speed. Whenever actual error changes its sign, the learning rate is reduced according to the following iterative formula

$$\eta_{\text{new}} = r_c \eta_{\text{old}} \ , \qquad (10)$$

where r_c is a coefficient in the range (0, 1).

If the degree of overlapping of membership functions is greater than a pre-specified threshold (e.g. 0.9), we combine those membership functions. We use the following *fuzzy similarity measure* [6]

$$E(A_1, A_2) = \frac{M(A_1 \cap A_2)}{M(A_1 \cup A_2)} \ , \qquad (11)$$

where $\cap$ and $\cup$ denote the intersection and the union of two fuzzy sets A_1 and A_2, respectively, and $E(A_1, A_2)$ is a degree of $A_1 = A_2$. $M(\cdot)$ is the size of a fuzzy set, and $0 \leq E(A_1, A_2) \leq 1$. We can thus reduce the size of the rule base, which is necessary in order to protect the controller from the "curse of dimensionality". Combining the membership functions is done to eliminate the underperforming membership functions, and to replace them with new ones which are likely to perform better.

If the accuracy of the DCF-FNC is satisfactory, the algorithm stops. Otherwise, if the number of rule nodes is less than the specified maximum, we add a new rule node together with its fuzzy weight (membership function) at the point of the maximum process output error. Fuzzy weight (membership function) is added for all inputs. One vertex of this membership function is placed at the point of the maximum process output error and has membership value unity; the other two vertices lie at the centers of the two neighboring regions, respectively, and have membership values zero. In such a way we are able to reduce the error efficiently. By firstly eliminating the errors whose deviation from the target values is the greatest, we can speed up the convergence of the controller substantially. Next, weights are adapted and the process is repeated until either we obtained a satisfactory performance, or the maximum pre-specified size of the controller (number of rule nodes) is exceeded.

3 Computer simulations

In this sections, we evaluate our DCF-FNC's ability to track reference signals and robustness to process disturbances and measurement noise by numerical simulations on a real-world example of cold mill thickness control. We then compare the results with the performances of the conventional PID controller and the CCN for the same problem.

3.1 Problem description

During rolling of metals, especially the final cold rolling of sheet materials, variations in gauge (thickness) along the length of the material must be controlled within very tight tolerances. Fast-acting hydraulic systems with a bandwidth of a few hundreds rad/sec are commonly used to achieve such tolerances.

In a single-stand reversing strip mill (Fig. 3), the incoming strip is supplied from a pay-off reel at one side of the mill, reduced in thickness as it is passed between the mill work rolls, and recoiled by a tension reel at the other side of the mill. The roll gap is then reduced, and the process is repeated in the reverse direction. The process continues until the outgoing strip is of the desired final thickness. The thickness of the rolled strip is predominantly determined by the gap between the work rolls, although there are other contributing parameters, such as the tension in the strip, hardness variations in incoming materials, and hardening of the material during rolling. The roll gap is initially set by electrical screw-down drives. Once the strip is threaded, changes to the roll gap are done by extending/contracting hydraulic cylinders. Automatic gauge control (AGC) for cylinder control is used in two modes, pressure (load) control or position control. Figure 4 [7] shows a typical arrangement of a control loop for cylinder position. For pressure control, position transducers are replaced by pressure transducers.

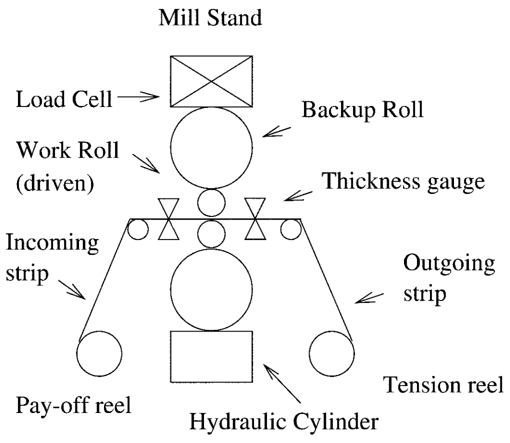


Fig. 3. A single-stand reversing strip mill

The cylinder position reference signal will have in practice contributions from several other control loops not shown in Fig. 4. For example, the outgoing strip thickness will be measured and any deviation from the desired value requires reduction of the reference signal for cylinder position. However, for physical reasons, it is impossible to measure the outgoing thickness until some distance downstream of the roll gap. This introduces a transport lag, which severely degrades the gauge performance when compensating for short-duration errors. In high performance mills, the incoming thickness at the roll gap is also measured and used in a feedforward control loop, so that the cylinders are adjusted in line with variations in the roll gap. Another common disturbance to the operation of the position control loop is the roll eccentricity, arising from imperfect roll grinding and roll wear.

Improvements in thickness and corresponding gap position in cold rolling mills have been motivated by the demand for increasing thickness accuracy. Conventional thickness control systems are based on the single-input-single-output design principle. Until recently, because of the fast dynamics of hydraulic AGC loops, control was implemented using standard PI or PID controllers. The rolling process, however, is a typical multivariable system with strong mutual interactions between the strip thickness, the roll gap position, the rolling force, and the strip hardness. Consequently, a new design strategy is necessary to overcome the problem of mutual interactions and to improve thickness accuracy. As the steel industry requires a better product quality with reduced production costs, a viable scheme for thickness control is of substantial interest to the industry. Considerable industrial research effort [7, 19, 25, 32, 37] has been devoted to finding the best possible solution.

The presence of the computer in AGC loops enables us to investigate the use of intelligent control methods, such as fuzzy neural networks. We propose to use our DCF-FNC controller with the process input–output data and a reference model for direct MRAC.

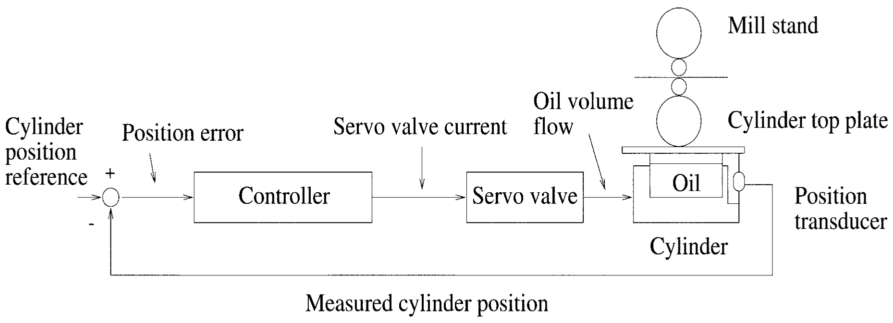


Fig. 4. Automatic gauge control (AGC) gap position loop

3.2 Simulation model of a cold rolling mill

In our simulation studies we have used an AGC position control servo valve and capsule model (Fig. 5) developed from the physical insight of the process [7] to replace the servo valve and capsule in Fig. 4 (the position transducer is assumed to have negligible dynamics). Although the main emphasis is on the gap position control, the other parts of the mill need to be taken into account (such as the force control loop), as their disturbances and noise affect the performance of the position control. This results in a complete model representation of the combined mill position control, disturbances and noise (Fig. 6) [19]. In Fig. 6, G represents a hydraulic roll-gap model for closed-loop position control of Fig. 5. G_e is an eccentricity disturbance model:

$$G_e = 852.224/((s^2 + 0.0314s + 985.96) \times (s^2 + 0.0294s + 864.36)) \ , \tag{12}$$

and G_h is an input thickness and hardness disturbance model:

$$G_h = 0.333/(s + 0.333) \ . \tag{13}$$

Experimental testing [19] verified the use of an eccentricity model comprising two lightly damped oscillators driven by zero-mean white noise with covariance $E\{\omega(t)\omega(\tau)\} = x_1^2$, where $x_1 = 0.00012$.

As the physical data for hardness variation in cold rolled strip mills is very expensive, we used a first order lag driven by zero-mean white noise [19] with covariance $E\{\xi(t)\xi(\tau)\} = x_2^2\delta(t-\tau)$, where $x_1 = 0.00007$ and δ is the Kronecker delta function.

The disturbances and noise were applied concurrently to represent a real situation when the disturbances and noise in the rolling mill are present at the same time.

To generate roll force and gauge variations a small change model is used [19]. There gauge $h(t)$ satisfies:

$$h(t) = \frac{M_m M_s^{-1}}{1 + M_m M_s^{-1}}\delta s(t) + \frac{1}{1 + M_m M_s^{-1}}\delta H(t) \ , \tag{14}$$

where $M_m = 1.039 \times 10^9$ N/m and $M_s = 9.81 \times 10^8$ N/m are the mill and the strip moduli, respectively, and $s(t)$ is the roll gap setting.

Thus the measured roll force $z(t)$ is:

$$\delta z(t) = M_m(\delta h(t) - \delta s(t)) + n(t) \ , \tag{15}$$

where $n(t)$ represents measurement noise with covariance $E\{n(t)n(\tau)\} = x_3^2\delta(t-\tau)$, where $x_3 = 1000$.

The thickness control in a cold rolling mill requires the output gauge to be regulated in the presence of disturbances using the measured roll gap position and the roll force. Here we have an inferential control problem where it is not the measured variables that are controlled, but the measured variables are used to achieve control of another system variable – the strip gauge (thickness).

3.3 Simulation results

We used as a reference model Butterworth's [2] characteristic equation for the 5th order system

$$F(s) = s^5 + 3.24\omega_n s^4 + 5.24\omega_n^2 s^3 + 5.24\omega_n^3 s^2 + 3.24\omega_n^4 s + \omega_n^5 \ , \tag{16}$$

where ω_n is a natural frequency of the system $\omega_n = 200$ rad/s. This form of characteristic equation gives us a damping ratio $\xi = 0.71$, and the settling time can be determined through approximate relationship $t_s \approx 4/\xi\omega_n$. The delay time t_d can also be approximated from the following relationship $t_d \approx (1 + 0.7\xi)/\omega_n$. Consequently, the number of delays in Eq. (5) $c = 2$. A 4 Hz square wave with the amplitude of 10 μm as in [7] (Fig. 7) applied to cylinder position reference was used as a test signal.

We used the input–output data generated using above reference model for training and testing of our DCF-FNC. We have generated 1000 input–output data pairs using fifth order Runge–Kutta integrator [2] with a sampling time $t = 0.005$ s normalized in the range $[-1, 1]$. The DCF-FNC was trained with on-line learning. We used first 500 samples for training, the other 500 for validation of con-

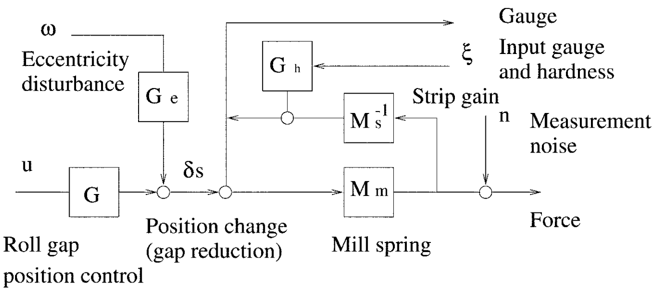


Fig. 6. An overall single-stand model for the cold rolling mill

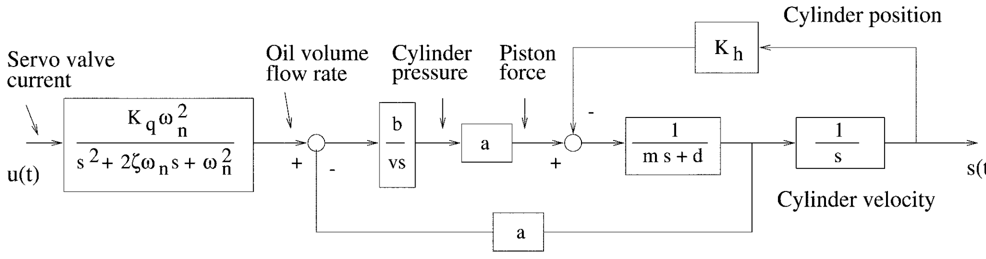


Fig. 5. Servo valve and cylinder model of a position loop. Here $s = dx/dt, \omega_n = 120$ rad/s is the servo valve natural frequency, $\zeta = 1.1$ is the valve damping ratio, $K_q = 25$ m^3/(sA) is the flow gain, $b = 1.4 \times 10^9$ N/m^2 is the oil compressibility, $a = 0.58$ m^2 is the capsule cross-sectional area, $v = 0.0232$ m^3 is the capsule volume, $m = 1 \times 10^5$ kg is the mass of the mill, $K_h = 4.5 \times 10^9$ N/m is the mill housing stiffness, and $d = 5 \times 10^6$ is a damping term

troller's ability to generalize, and the whole data set for final testing of the resulted controller. For comparison the same data was used with a CCN with the quickpropagation learning algorithm [9, 44] (learning rate $\eta = 0.005$, maximum growth parameter $\mu = 1.75$, weight decay term $\nu = 0.0001$, maximum tolerated difference between a teaching value and an output $d_{\max} = 0.1$). In addition, two PID controllers was used for the same process (one for the position loop, another for the force loop). The parameters of PID controllers, i.e., gain $K_c = 15.5$, integral time constant $\tau_I = 1.6$, and derivative time constant $\tau_D = 2$, were selected according to the industry standard Ziegler–Nichols' method [2].

Simulation results are presented in Figs. 8–11. Figure 8 shows the uncontrolled time response (root mean squared error (RMSE) = 0.1724 for the position and 0.1232 for the rolling force). While reducing the deviations due to disturbances and noise of the position of roll gap and the rolling force to some extend, the PID controller showed rather poor performance (RMSE = 0.0897 for the position and 0.0787 for the rolling force in Fig. 9). The reasons for the poor performance of the conventional "optimal" PID controller may be attributed to the high non-stationarity and/or the coupling of the rolling process. Fixed PID controllers are thus not able to capture the underlying

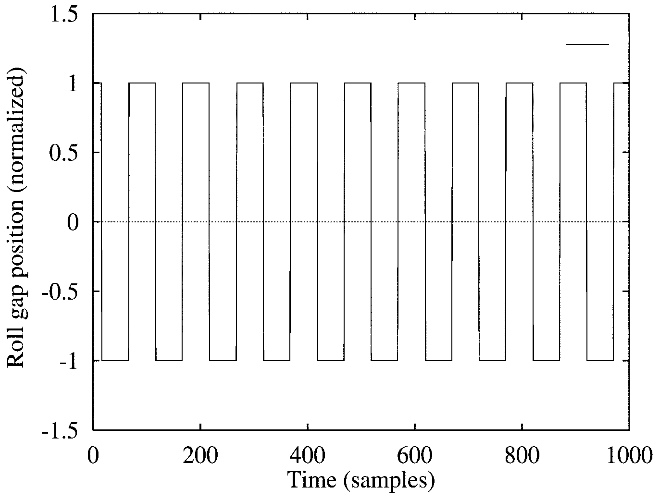


Fig. 7. The desired square wave reference signal (sampling time $t = 0.005$ s in Figs. 7–11)

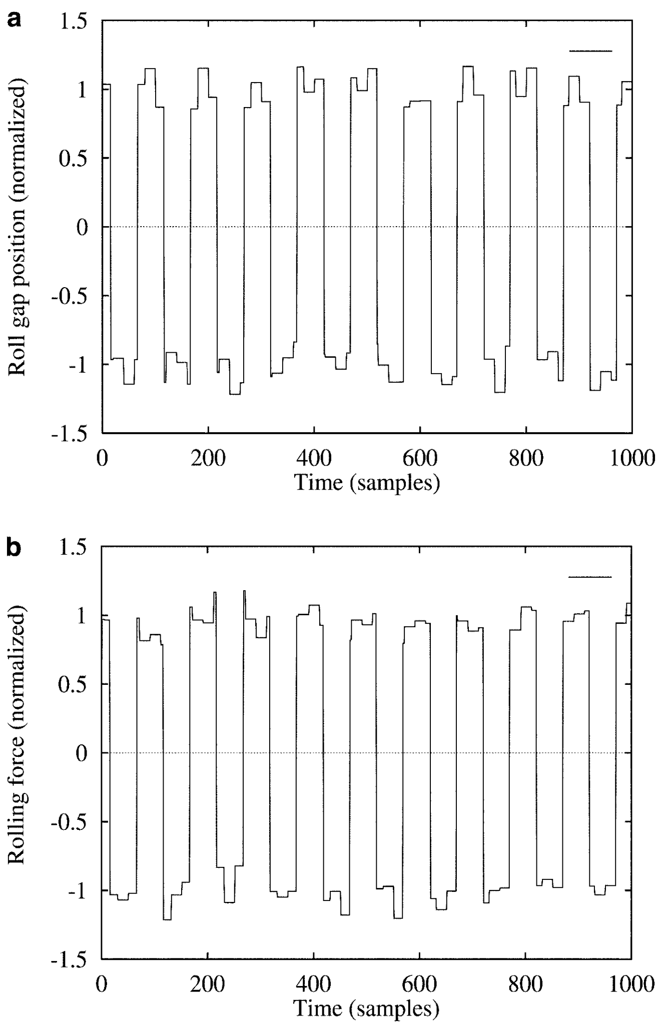


Fig. 8a, b. The time response for uncontrolled **a** roll gap position (root mean squared error (RMSE) = 0.1724) and **b** rolling force (RMSE = 0.1232)

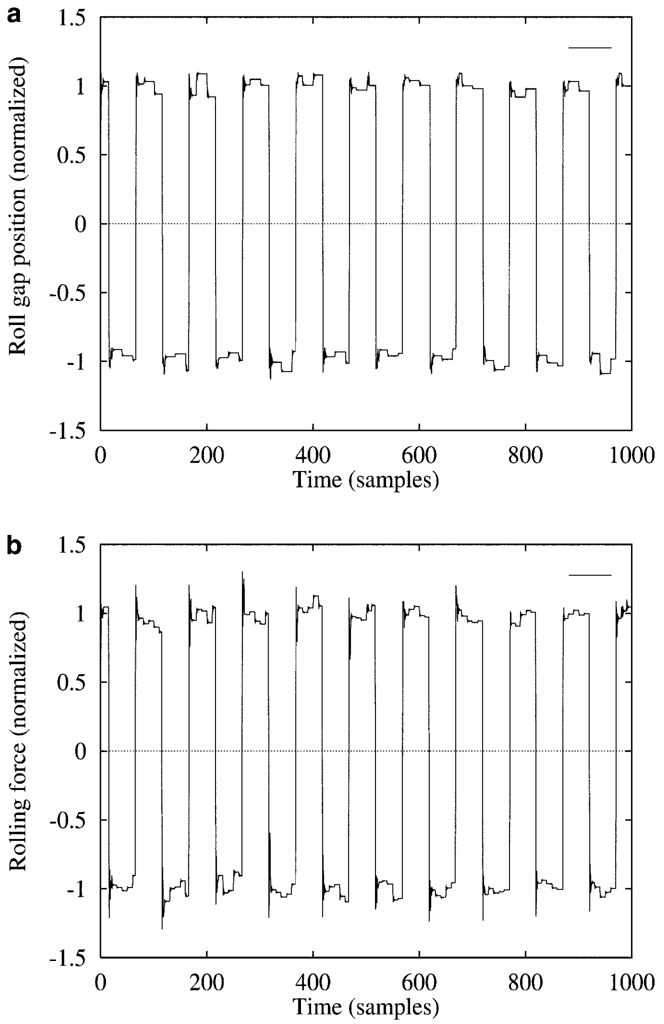
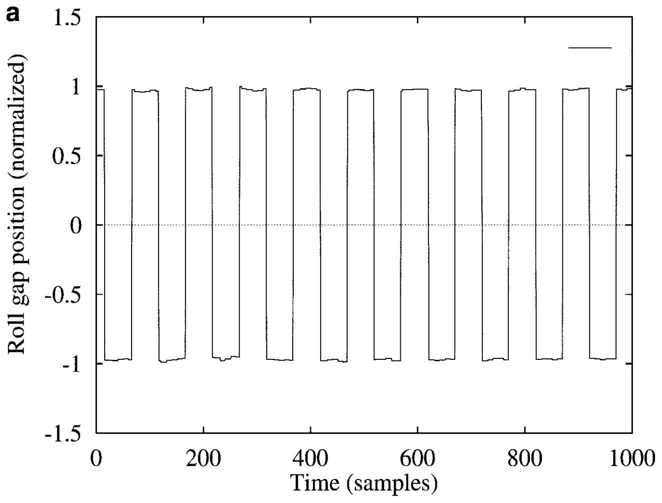


Fig. 9a, b. The time response for **a** the roll gap position with the PID controller (RMSE = 0.0897) and **b** the rolling force (RMSE = 0.0787)



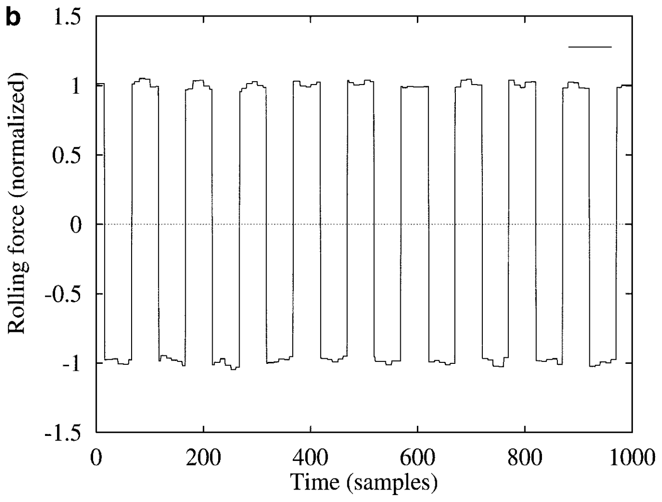
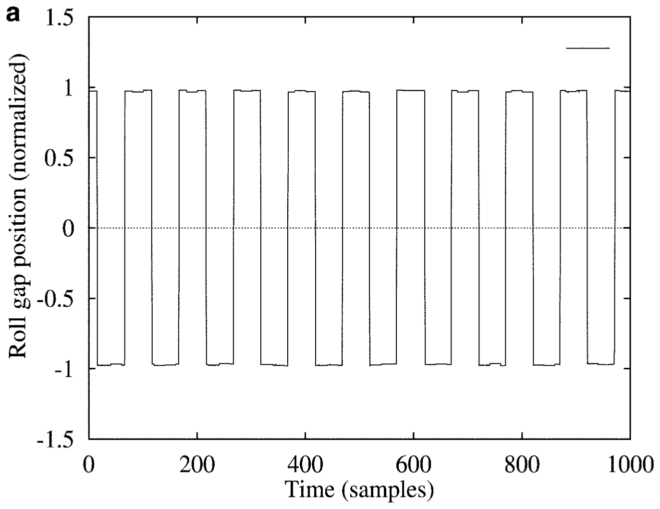


Fig. 10a, b. The best performance of the CCN for **a** the roll gap position (RMSE = 0.0312) and **b** the rolling force (RMSE = 0.0518) after the data normalization. The CCN almost always fails to learn or leads to poor performance without such normalization



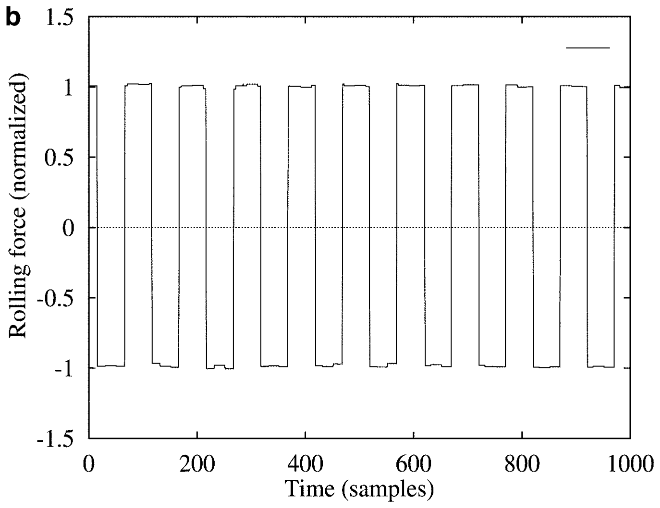


Fig. 11a, b. The time response with our DCF-FNC controller for **a** the roll gap position (RMSE = 0.0285) and **b** the rolling force (RMSE = 0.0127). The DCF-FNC leads to a very similar performances with or without data normalization, in contrast to CCN

process behavior. The performance of the CCN is much better, although still not completely satisfactory (RMSE = 0.0312 for the position and 0.0518 for the rolling force in Fig. 10). The response of the CCN-controlled rolling force becomes significantly worse compared to that of the position, which shows that the CCN controller is not able to keep up with the fast changes in the process behavior. In contrast, our DCF-FNC controller is able to produce significantly better results compared to both the PID controller and the CCN (RMSE = 0.0285 for the position and 0.0127 for the rolling force in Fig. 11). If we take into account that disturbances are in the force control loop (Fig. 6) we notice that out DCF-FNC produces a significantly tighter control of disturbances compared to the CCN (Figs. 10b and 11b). As both the roll gap position and the rolling force errors contribute to the strip thickness error, we can conclude that control by the DCF-FNC is much better compared to control by the CCN. Both the DCF-FNC and the CCN produce much better control compared to the PID controller, which indicates a distinct advantage of nonlinear control over linear control in case of a nonlinear industrial process. Apart from giving better accuracy than the CCN does, our DCF-FNC has a much smaller size (24 hidden nodes) in comparison with the CCN (37 nodes).

For the CCN to perform adequately, all input and outputs variable *have to be* strictly normalized within [−1, 1]. Failure to do so will result either in a poor network performance, or inability to learn at all. This could possess a problem with application of CCN to a real process, as the process variables are usually of a very different range and variance. In contrast, the DCF-FNC is much less sensitive to normalization of the variables due to the use of fuzzy logic. To deal with the problem termed by Fahlman [10] as the *moving target problem*, each hidden node in the CCN is frozen after installation into the network. Although this feature prevents the network from unlearning (forgetting) the old information in order to allow the network to follow well several different targets as in our case, this is not always beneficial in real-world situation, especially with time-varying processes as it limits the network's ability to adapt to changes. Freezing hidden nodes in the CCN can be viewed as equivalent to a fuzzy neural network with a fixed grid partition, in contrast to a variable grid partition employed by the DCF-FNC. Furthermore, the deep structure of the CCN can also result in practical problems, as the speed of signal propagation through the network may be too slow to keep up the process changes for a fast plant, especially if the resulting network size is relatively large. These may be the reasons why the CCN is unable to achieve the same performance as our DCF-FNC.

Both control accuracy and computational efficiency signify that direct MRAC based on our DCF-FNC with both structure and parameters tuning can lead to marked improvements in control of industrial processes in the presence of process disturbances and measurement noise.

4 Conclusion

In this work we presented our dynamically-constructed, feedback fuzzy neural controller (DCF-FNC) for direct model reference adaptive control (MRAC). The effectiveness of our approach was demonstrated by simulation results on a real-world example of thickness control in cold-rolling mills and was compared with the performances of the conventional PID controller and the cascade correlation neural network (CCN). Both the CCN and our DCF-FNC significantly increase the control precision and robustness compared to the linear PID controller, with our DCF-FNC giving the best result in terms of both accuracy and the compactness of the controller, as well as being less computationally intensive than the CCN. We argue that our DCF-FNC controller with both structure and parameter learning can provide a computationally efficient solution to control of many real-world nonlinear processes in the presence of process disturbances and measurement noise.

References

1. **Ash T** (1989) Dynamic node creation in backpropagation networks, Connection Science **1**: 365–375
2. **Åström K, Wittenmark B** (1990) Computer Controlled Systems, Theory and Design, 2nd edn. Prentice-Hall, Upper Saddle River, New Jersey, U.S.A.
3. **Berenji H, Khedkar P** (1993) Learning and tuning fuzzy logic controllers through reinforcements, IEEE Trans Neural Networks **3**: 724–740
4. **Brown M, Harris C** (1994) Neurofuzzy Adaptive Modelling and Control, Prentice-Hall, Upper Saddle River, New Jersey, U.S.A.
5. **Carpenter GA, Grossberg S, Rosen DB** (1992) Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks, **3**: 698–712
6. **Dubois D, Prade H** (1982) A unifying view of comparison indices in a fuzzy set theoretic framework. In: Yager RR (ed.) Fuzzy Sets and Possibility Theory: Recent Developments, Pergamon, NY
7. **Dutton K, Groves CN** (1996) Self-tuning control of a cold mill automatic gauge control system, Int J Control **65**: 573–588
8. **Elman J** (1990) Finding structure in time, Cognitive Science **14**: 179–211
9. **Fahlman S** (1988) Faster-learning variations on back-propagation: an empirical study. In: Sejnowski T, Hinton G, Touretzky D (eds) 1988 Connectionist Models Summer School, Morgan Kaufmann, San Mateo, CA, pp 38–51
10. **Fahlman S, Lebiere C** (1990) The cascade-correlation learning architecture. In: Touretzky DS (ed.) Advances in Neural Information Processing Systems 2, Morgan Kaufmann, San Mateo, CA, pp 524–532
11. **Fahlman S** (1991) The recurrent cascade-correlation learning architecture. Technical Report CMU-CS-91-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.
12. **Farrell J, Baker W** (1993) Learning control systems. In: Antsaklis P, Passino K (eds) Intelligent Autonomous Control Systems, Kluwer Academic, Norwell, MA, U.S.A.
13. **Farrell J** (1996) Neural control. In: Levine WS (ed.) The Control Handbook, CRC Press, Boca Raton, FL, USA., pp 1017–1030
14. **Frayman Y, Wang L** (1997) A fuzzy neural approach to speed control of an elastic two-mass system. In: Verma B, Yao X (eds) Proc 1997 Int Conf Computational Intelligence and Multimedia Applications, Queensland University of Technology, Gold Coast, pp 341–345
15. **Frayman Y, Wang L** (1998) Dynamically constructed recurrent fuzzy neural network for cold rolling mill thickness control. Proc 11th Australian Joint Conference on Artificial Intelligence, pp 15–23
16. **Frean M** (1990) The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Computation **1**: 198–209
17. **Funahashi K, Nakamura Y** (1993) Approximation of dynamical systems by continuous time recurrent neural networks. Neural Networks **6**: 801–806
18. **Gorrini V, Bersini H** (1994) Recurrent fuzzy system. Proc 3rd IEEE Int Conf Fuzzy Systems, pp 193–198
19. **Grimble MJ** (1995) Polynomial solution of the standard H_∞ control problem for strip mill gauge control, IEE Proc-Control Theory Appl, vol 142, pp 515–525
20. **Harris C, Moore C, Brown M** (1993) Intelligent Control: Aspects of Fuzzy Logic and Neural Nets, World Scientific, Singapore
21. **Hecht-Nielsen R** (1990) Neurocomputing, Addison-Wesley Reading, MA
22. **Horikawa S, Furahashi T, Uchikawa Y** (1992) On fuzzy modelling using neural networks with the backpropagation algorithm, IEEE Trans Neural Networks **3**: 801–806
23. **Hunt K, Sbarbaro D, Zbikowski R, Gawthrop P** (1992) Neural networks for control systems – a survey. Automatica **28**: 1083–1112
24. **Jang J-SR** (1993) ANFIS: adaptive-network-based fuzzy inference systems, IEEE Trans Systems, Man, and Cybernetics **23**: 665–685
25. **Katori H, Yositani N, Ueyama T, Nyu S** (1992) Application of two degree of freedom control system to automatic gauge control, Proc 1992 American Control Conf, vol 1, pp 806–810
26. **Khan E, Unal F** (1994) Recurrent fuzzy logic using neural networks, Proc 1994 IEEE Nagoya World Wisepersons Workshop, pp 48–55
27. **Kosko B** (1991) Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence, Prentice Hall, Englewood Cliffs, NJ
28. **Lin C-J, Lin CT** (1996) Reinforcement learning for an ART-based adaptive learning control network, IEEE Trans Neural Networks **7**: 709–731
29. **Lin C-T, Lee CSG** (1994) Neural Fuzzy Control Systems with Structure and Parameter Learning, World Scientific, Singapore
30. **Mezard M, Nadal J** (1989) Learning in feedforward layered networks: the tiling algorithms, J Physics A **22**: 2191–2203
31. **Nadal J** (1989) Study of a growth algorithm for feedforward neural network, Int J Neural Systems **1**: 55–59
32. **Middleton RN, Goodwin GC** (1990) Digital Control and Estimation, Prentice-Hall, Englewood Cliffs, NJ

33. **Nie J, Linkens D** (1995) Fuzzy-Neural Control: Principles, Algorithms and Applications. Prentice Hall, Upper Saddle River, New Jersay, U.S.A.
34. **Narendra K, Pathasarathy K** (1990) Identification and control of dynamical systems using neural networks, IEEE Trans Neural Networks **1**: 4–27
35. **Narendra K, Mukhopadhyay S** (1994) Adaptive control of nonlinear multivariable systems using neural networks, Neural Networks **7**: 737–752
36. **Nauck D, Kruse R** (1993) A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In: Proc IEEE Int Conf Neural Networks ICNN'93, pp 1022–1027
37. **Postlethwaite I, Geddes J** (1994) Gauge control in tandem cold rolling mills: a multivariable case study using H^∞ optimization. Proc 3rd IEEE Conf Control Applications, vol 3, pp 1551–1556
38. **Sugeno M, Tanaka K** (1991) Successive identification of fuzzy model and its application to prediction of complex system, Fuzzy Sets Syst **42**: 315–344
39. **Takagi T, Sugeno M** (1985) Fuzzy identification of systems and its applications to modelling and control, IEEE Trans Syst Man Cybernetics **15**: 116–132
40. **Tan S, Yu Y** (1996) Adaptive fuzzy modeling of nonlinear dynamical systems, Automatica **32**: 637–643
41. **Wang L-X** (1994) Adaptive Fuzzy Systems and Control: Design and Stability Analysis. PTR Prentice Hall, Upper Saddle River, NJ, U.S.A.
42. **Wasserman PD** (1993) Advanced Methods in Neural Computing, Van Nostrand Reinhold, New York, NY, USA.
43. **Yen J, Langari R, Zadeh L** (eds) (1995) Industrial Applications of Fuzzy Logic and Intelligent Systems, IEEE Press, Piscataway, NJ, U.S.A.
44. **Zipser D** (1990) Subgrouping reduces complexity and speeds up learning in recurrent networks. In: Touretzky DS (ed.) Advances in Neural Information Processing Systems 2, Morgan Kaufmann, San Mateo, CA, pp 638–641