Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Engineering Applications of Artificial Intelligence 15 (2002) 541–550
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

A dynamically generated fuzzy neural network and its application to torsional vibration control of tandem cold rolling mill spindles

Lipo Wang*, Yakov Frayman

School of Electrical and Electronic Engineering, Nanyang Technological University, Block S1, Nanyang Avenue, Singapore 639798, Singapore

Received 27 January 2002; received in revised form 29 January 2003; accepted 3 February 2003

Abstract

Based on the model of Higgins and Goodman, we describe a dynamically generated fuzzy neural network (DGFNN) approach to control, from input–output data, using on-line learning. The DGFNN is complete with the following powerful features drawn or modified from the existing literature: (1) a small FNN is created from scratch—there is no need to specify initial network architecture, initial membership functions, or initial weights, (2) fuzzy rules are constantly combined and pruned to minimize the size of the network while maintaining accuracy, irrelevant inputs are detected and deleted; and (3) membership functions and network weights are trained with a backpropagation-type algorithm. We apply the DGFNN controller to a real-world application of controlling the torsional vibration of tandem cold-rolling mill spindles with a simulated plant. The results of the DGFNN controller are compared with the performances of a conventional proportional-integral controller and a neural controller using recurrent cascade correlation with quickpropagation through time. We show that while both neural approaches increase the control precision and robustness, the DGFNN controller gives the best results for reducing the speed deviation and suppressing the torsional vibration of the spindles, as well as is more computationally efficient.

Keywords: Dynamically generated fuzzy neural network; Tandem cold-rolling mill; Neurofuzzy control

1. Introduction

Complex industrial control problems are increasingly driving research and development of new intelligent control methods based on soft computing, such as fuzzy logic inference systems, crisp (nonfuzzy) neural networks, genetic algorithms, as well as various combinations of these techniques. Both fuzzy logic inference systems and multilayer perceptron neural networks (MLP) are universal approximators (Hornik et al., 1989; Cybenko, 1989; Funahashi and Nakamura, 1993; Wang and Mendel, 1992). This property makes them suitable for control. While these intelligent techniques have produced encouraging results, in particular control tasks (e.g., Hunt et al., 1992; Narendra and Pathasarathy, 1990; Narendra and Mukhopadhyay, 1994; Neumerkel et al., 1994; Sbarbaro-Hofer et al., 1993; Yen et al., 1995; Driankov et al., 1995; Jin et al., 1995), certain complex control problems cannot be solved by a single technique alone (Brown and Harris, 1994; Harris et al., 1993).

From the implementation point of view, it is undesirable to construct a controller by trail and error, which is common in fuzzy control, because the size of the rule matrix becomes prohibitively large for higher-order systems. Thus, the necessity for a dynamically created controller, based only on input–output measurements of plant variables, becomes vital.

The situation is similar with crisp neural networks: the architecture of an MLP, i.e., the number of hidden layers and the number of hidden neurons in each layer, is often selected on a trial-and-error basis. The network weights are often initialized to small random values. Several authors proposed learning algorithms that construct the architecture during learning for MLPs (Ash, 1989; Fahlman and Lebiere, 1990; Frean, 1990; Mezard and Nadal, 1989; Nadal, 1989; Giles et al., 1995), while Fahlman proposed a cascade correlation algorithm (Fahlman and Lebiere, 1990), as well as a recurrent cascade correlation (RCC) algorithm (Fahlman, 1991). These networks have long training times

*Corresponding author. Tel.: +65-6790-6372; fax: +65-6793-3318.
E-mail address: elpwang@ntu.edu.sg (L. Wang).

0952-1976/02/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0952-1976(03)00006-X

and are therefore not well-suited for real-time control in most practical situations.

There have been extensive research activities on fuzzy neural networks (FNNs); however, the majority of them are not self-constructing (e.g., Berenji and Khedkar, 1993; Horikawa et al., 1992; Jang, 1993; Kosko, 1991; Nauck and Kruse, 1993; Wang, 1994). Usage of bell-shaped membership functions is computationally intensive (Horikawa et al., 1992; Jang, 1993; Lin and Lee, 1994; Nie and Linkens, 1993, 1995; Wang, 1994; Juang and Lin, 1999). The use of fully overlapping triangular membership functions (e.g., Higgins and Goodman, 1994) can greatly reduce computational cost, which we shall use in this paper.

Higgins and Goodman (1994) proposed an algorithm to create an FNN according to input data. New membership functions are added at the point of maximum error on an as-needed basis, which will be adopted in the present paper. They then used an information-theoretic approach to simplify the rules. In contrast, we will combine rules using a computationally more efficient approach, i.e., a fuzzy similarity measure. In Higgins and Goodman (1994), no backpropagation-type error-gradient-descent learning was used to refine the weights and there were no rule-pruning or elimination of irrelevant inputs.

Juang and Lin (1999) also proposed a self-constructing FNN with on-line learning. New membership functions are added based on input–output space partitioning using a self-organizing clustering algorithm. This membership creation mechanism is not directly aimed at minimizing the output error as in Higgins and Goodman (1994). A backpropagation-type learning procedure was used to train network parameters. There were no rule-combination, rule-pruning, or elimination of irrelevant inputs.

Wang and Langari (1995) and Cai and Kwan (1998) used self-organizing clustering approaches to partition the input/output space, in order to determine the number of rules and their membership functions in an FNN through batch (off-line) training. Backpropagation-type error-minimizing algorithm is often used to train network parameters in various FNNs with batch training (e.g., Jang, 1993; Ishibuchi et al., 1994).

While the existing FNNs possess some attractive features, they do not provide a *complete framework* for control of complex systems. In this paper, we describe a dynamically generated fuzzy neural network (DGFNN) approach to control, from input–output data, using on-line learning. The DGFNN has the following powerful features drawn or modified from the existing literature: (1) a small FNN is created from scratch—there is no need to specify initial network architecture, initial membership functions, or initial weights; (2) fuzzy rules are constantly combined and pruned to minimize the size of the network while maintaining accuracy, irrelevant inputs are detected and deleted; and (3) membership functions and network weights are trained with a backpropagation-type algorithm. We apply the DGFNN controller to torsional vibration control of tandem cold-rolling mill spindles. The paper is structured as follows. Section 2 describes the DGFNN. In Section 3, we state the control problem—torsional vibration control of tandem cold-rolling mill spindles. We derive a mathematical model of the two-mass system for use with the proportional-integral (PI) control in Section 4, where the simulation results using the PI controller will be presented. Section 5 presents the simulation results using the RCC and the DGFNN, as well as a comparison of the PI, RCC, and DGFNN results. Finally, Section 6 summarizes the main findings of the paper.

2. A DGFNN approach to control

Let us first establish our notations. A control system can be represented by the following state–space model

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

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

where $\vec{u}$ and $\vec{y}$ are the system input and output vectors, respectively, and $\vec{x}$ is the system state vector. Eq. (1) is the identification, and Eq. (2) is the action (control). Given that the vector maps $\vec{f}$ and $\vec{g}$ are unknown, a reference model with input $\vec{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$ and output $\vec{y}_{\mathrm{d}} = [y_{\mathrm{d}1}, y_{\mathrm{d}2}, \ldots, y_{\mathrm{d}m}]^{\mathrm{T}}$ is used to designate the desired performance. The desired output response $\vec{y}_{\mathrm{d}}$ is obtained from the output of the reference model subject to the step signal (for load disturbance) or the other disturbance signals. The parameters of the model are determined by the desired performance in terms of overshoot, settling time, and steady-state error.

The structure of the DGFNN is shown in Fig. 1. The network consists of four layers, i.e., the input layer, the input membership functions layer, the rule layer, and the output layer. The input nodes represent input variables. The input membership function layer generates input membership functions for the numerical inputs. Each input node is connected to all membership functions nodes for this input. The input membership function nodes are term nodes which act as membership functions. We will use piecewise linear-triangular membership functions (Higgins and Goodman, 1994), since they are computationally efficient. Membership functions fully overlap and their total membership at any point is equal to one. One vertex lies at the center of the region and has membership value unity, and the other two vertices lie at the centers of the two

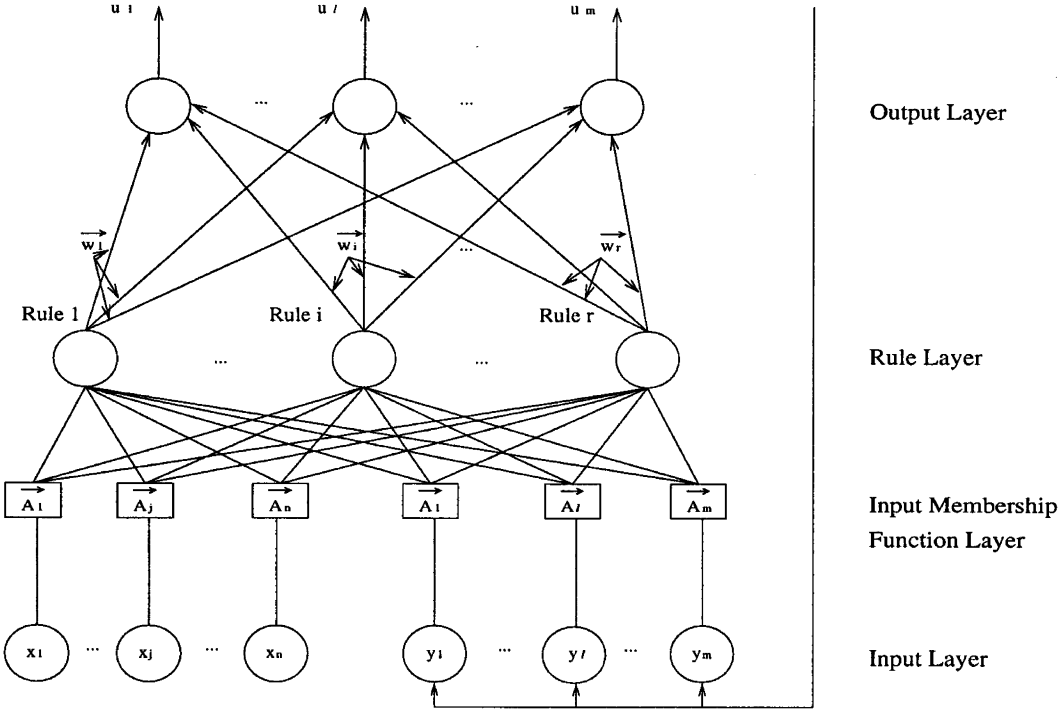

Fig. 1. Structure of the presented fuzzy neural network structure.

neighboring regions, respectively, with membership values zero. Consequently, we need to specify only the positions of the peaks A_j^i, where i is a rule number, j is an input number, but not the width of the function, to describe a membership function completely.

The rule layer nodes represent fuzzy rules using the following form for rule i:

$$\begin{aligned}\text{Rule } i: \quad &\text{IF } x_1 \text{ is } A_1^i \text{ and } \cdots x_n \text{ is } A_n^i \\ &\text{and } y_1 \text{ is } A_1^i \text{ and } \cdots y_{\mathrm{m}} \text{ is } A_{\mathrm{m}}^i \\ &\text{THEN } u_1 = w_1^i \text{ and } \cdots u_n = w_n^i, \end{aligned} \tag{3}$$

where i is the rule number, A_j^i is the membership function of the antecedent part, w_{m}^i is the real number of the consequent part. They are connected to appropriate input membership function nodes and output nodes.

The membership value μ_i of the premise of the ith rule is calculated as fuzzy and using the product operator, which gives a smoother trade-off between rules compared to the minimum:

$$\mu_i = A_1^i(x_1)A_2^i(x_2)\cdots A_n^i(x_n). \tag{4}$$

The connection weights between input nodes, the input membership function layer, and the rule layer are constant unity weights. Links between the rule layer and the output layer are adaptive during learning. In the output layer each node receives inputs from all rule nodes connected to this output node and produces the actual output of the controller. The output u_l of the fuzzy inference can be derived from the following equation:

$$u_l = \frac{\sum \mu_i w_l^i}{\sum \mu_i} \quad (l = 1, 2, \ldots, m), \tag{5}$$

where μ_i is the membership value of the ith inference rule. Among the commonly used defuzzification strategies, the simplified fuzzy inference allows for simple network implementation and is computationally efficient (Sugeno and Tanaka, 1991; Takagi and Sugeno, 1985). There is a feedback from the output of the network to the input layer (Elman, 1990).

The main aim of the learning algorithm is to obtain the correct control signal $\vec{u}_{\mathrm{d}} = [u_{\mathrm{d}1}, u_{\mathrm{d}2}, \ldots, u_{\mathrm{d}m}]^{\mathrm{T}}$ corresponding to the desired output $\vec{y}_{\mathrm{d}}$. The learning error $\vec{\varepsilon} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{\mathrm{m}}]^{\mathrm{T}}$, defined as the difference between the desired response $\vec{y}_{\mathrm{d}}$ and the measured process output $\vec{y} = [y_1, y_2, \ldots, y_{\mathrm{m}}]^{\mathrm{T}}$, is used as a learning criterion:

$$\varepsilon = \tfrac{1}{2}(y - y_{\mathrm{d}l})^2, \tag{6}$$

where ε is reduced towards a pre-specified small value $\varepsilon_l > 0$ during learning.

The DGFNN structure-generation and learning algorithms used in this paper are as follows:

1. Select input x, output y, and the reference model. Generate the training input–output data for the selected reference model. Specify the allowable error threshold ε or the maximum number of rules (rule nodes) for learning to stop. Start with the number of

input nodes equal to the number of input variables, and the number of output nodes equal to the number of output variables. The hidden (rule layer) is empty, i.e., there are initially no rules in the rule base.

2. Initially, add two input membership functions nodes at input minimum and maximum in the data set for each input (Higgins and Goodman, 1994).
3. Add an additional membership function node at zero point for each input variable as we are more concerned with the performance of the system during the steady state than during the transient process. Create the initial rule base layer using Eq. (3).
4. Train the network using the following general learning rule:

$$y_l^i(k+1) = y_l^i(k) - \eta \frac{\partial \varepsilon_l}{\mathrm{d}y_l^i}. \tag{7}$$

The learning rules for w_l^i and A_j^i are:

$$w_l^i(k+1) = w_l^i(k) - \eta \frac{\partial \varepsilon_l}{\partial w_l^i}, \tag{8.1}$$

$$A_q^i(k+1) = A_q^i(k) - \eta \frac{\partial \varepsilon_l}{\partial A_q^i}, \tag{8.2}$$

where η is the learning rate. We let the learning rate η vary to improve the speed of convergence, as well as the learning performance (accuracy): whenever $\partial \varepsilon_l$ changes its sign, the learning rate is reduced to $\eta_{\mathrm{new}} = r_{\mathrm{c}}\ \eta_{\mathrm{old}}$, where r_{c} is a coefficient in the range $(0, 1)$.

5. If the degree of overlapping of membership functions is greater than a threshold, combine those membership functions. We use the following *fuzzy similarity measure* (Dubois and Prade, 1982):

$$E(A_1, A_2) = \frac{M(A_1 \cap A_2)}{M(A_1 \cup A_2)}, \tag{9}$$

where $\cap$ and $\cup$ denote the intersection and union of two fuzzy sets A_1 and A_2, respectively. $M(\cdot)$ is the size of a fuzzy set, and $0 \leqslant E(A_1, A_2) \leqslant 1$. If an input variable ends up with only one membership function, which means that this input is irrelevant, we delete the input. We can thus eliminate irrelevant inputs and reduce the size of the rule base, thereby making the network more computationally more efficient. If the classification accuracy of the DGFNN is below the requirement, and the number of rules is less than the specified maximum, go to step 5. Otherwise, go to step 6.

6. Add an additional membership function for each input at its value at the point of the maximum output error, following Higgins and Goodman (1994). One vertex of the additional membership function is placed at the value at the point of the maximum output error and has the membership value unity; the other two vertices lie at the centers of the two neighboring regions, respectively, and have membership values zero. As the output of the network is not a binary 0 or 1, but a continuous function in the range from 0 to 1, by firstly eliminating the error whose deviation from the target value is the greatest, we can speed up the convergence of the network substantially.
7. Go to step 3 (retrain the network with the updated rule layer using the algorithm given in step 3).

For control systems, we are usually more concerned with control precision during steady states than during transient processes. Thus, we need more membership functions near the steady state. They have to be added when the output variable is near the reference value and changes only slightly, as inputs are almost steady with

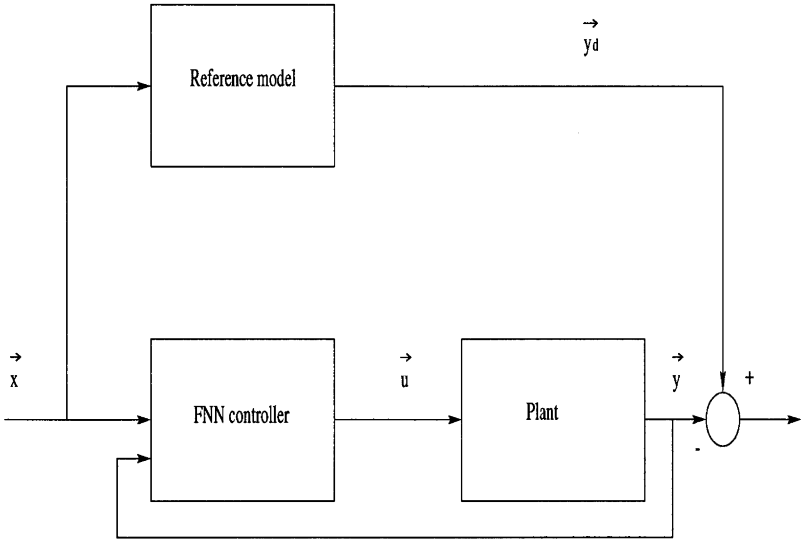

Fig. 2. Block diagram of the FNN control system.

very little change. Consequently, the reduction of steady-state error below 1% is not possible by the main algorithm described above. To achieve the desired response, we need to have additional membership functions in the region $[-0.02, 0.02]$, which will create additional control variable $u_{\text{add}}(k)$. To compensate for each stationary output error, the variable u_{add} at sampling time k is determined by a recursive mathematical expression

$$u_{\text{add}}(k) = u_{\text{add}}(k-1) + \Delta u_{\text{add}}(k). \tag{10}$$

The total control variable becomes

$$u'(k) = u(k) + u_{\text{add}}(k). \tag{11}$$

The recursive calculation provides an integral fuzzy neural controller part for compensation of stationary errors, as its value increases with time, if the output error is constant.

The block diagram of the DGFNN control system is presented in Fig. 2.

3. Torsional vibration control of tandem cold-rolling mill spindles: the control problem definition

The function of the tandem cold-rolling mill (Fig. 3) is to reduce the thickness of the incoming strip, supplied in a coil at room temperature, by a factor of 2–10, so the outgoing strip should have a uniform thickness with typical dimensions of 20–50 in width, and 0.007–0.012 in thickness (Bryant, 1973). The product is commonly used by the automotive industry and consumer goods manufacturers.

To improve the gauge accuracy of rolled strip, the roll speed control system must assure an extremely high accuracy, so that the angular velocity is at a pre-specified value of 20 rad/s. In reality, the resonance point of torsional vibration, which depends on the inertia of the DC drive motor and roll, as well as the torsional rigidity of the intermediate shaft and spindle, is often present at 40–90 rad/s. This frequency

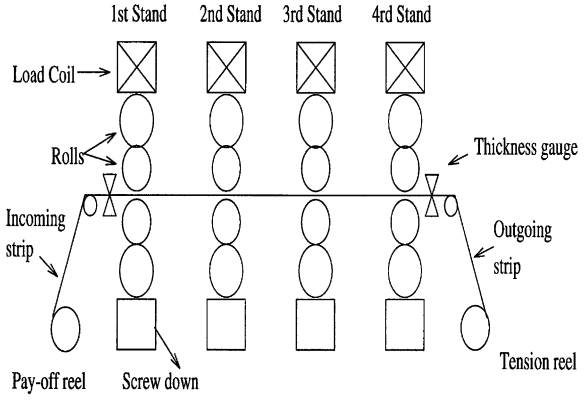

Fig. 3. Schematic diagram of a tandem cold-rolling mill.

determines the behavior of the system. The larger (smaller) the frequency, the softer (harder) the system, and hence the greater (lesser) the possible torsion of the shaft. Thus, the resonance frequency also determines the speed of the controller. For example, a softer system requires a lower roll speed to avoid extensive torsion of the shaft and oscillations of the two masses against each other. To avoid adverse effects of high resonance, the convenient PI control systems are adjusted to a lower frequency response of 5–15 rad/s. Consequently, the mill speed is not properly corrected by the control system at a sufficiently high speed. The speed drop is so great when the strip enters the mill that it takes a long time to recover from such a speed drop. The result is inability to obtain a desired gauge accuracy (Harakawa et al., 1987; Kawaguchi and Ueyama, 1989). To solve this problem, we propose a novel mill speed control system, using an FNN approach.

4. Tandem cold-rolling mill as a two-mass system

In this section, we develop a mathematical model of the system for use with the PI controller.

It is not sufficient to merely concentrate on the engine while controlling the speed of an electrical drive. In reality, a totally fixed connection between drive and the mechanism is not possible. As a first step in the mathematical modeling of the system, the ideal situation of a fixed connection is usually assumed. If the results are not satisfactory, the elastic soft connection and possibly other nonlinearities, such as slack or Coulomb friction in transmissions, clutches and bearings, have to be taken into consideration. Then a model of an elastic two-mass system has to be used (Kuo, 1991). In many situations, a fixed shaft connection can be assumed. When dealing with the control of a rolling mill, the system should be modeled with two or more masses. The more flexible the shaft and thus the system, the higher the influence of nonlinearities, friction and slack. The consequences are stick-slip limit cycles. Especially in multimass systems, it either is impossible or requires enormous efforts to avoid these phenomena with conventional control theory. In addition, the real motor shows a nonlinear dependency of the angular acceleration from the angular velocity, representing the friction of the controlled system. Other nonlinearities are caused by a hysteresis of the motor shaft and a higher friction of the motor with time, e.g., by solidification of grease.

The basic physical representation of the process and the relevant variables involved in the control problem are given in Fig. 4. The rolling mill is represented as an electromechanical system, where the mechanism, i.e., roll through the gear train, is driven by an electrical motor. Connections between the motor and the roll are

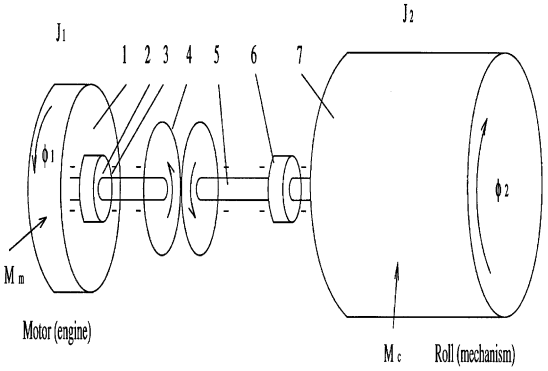

Fig. 4. Equivalent model of the roll drive system: 1, electrical motor; 2 and 6, spindles; 3, bearings; 4, gear train; 5, shaft; and 7, roll.

elastic because of the twisting of the shaft, the rigidity of the gear train, and the connecting spindles.

According to Fig. 4, we use the following mathematical description of the system, assuming that the moment of inertia of the gear train is very small compared to the moments of inertia of both the motor and the roll, as well as the moments of friction applied to them:

$$M_m = M_{12} - b_{12}(\dot{\phi}_1 - \dot{\phi}_2) - M_{f1} = J_1\ddot{\phi}_1, \tag{12}$$

$$M_{12} = c_{12}(\phi_1 - \phi_2), \tag{13}$$

$$M_{12} - b_{12}(\dot{\phi}_2 - \dot{\phi}_1) - M_c - M_{f2} = J_2\ddot{\phi}_2. \tag{14}$$

Here M_m is the moment of the motor. M_c is the moment of the roll. ϕ_1 and ϕ_2 are the angles of rotation of each mass. $\dot{\phi}_1$, $\dot{\phi}_2$, $\ddot{\phi}_1$, and $\ddot{\phi}_2$ are the first and the second derivatives of the angles. b_{12} is the friction coefficient. c_{12} is the coefficient of rigidity. M_{f1} and M_{f2} are the moments of the friction for each mass. M_{12} is the moment between the masses. J_1 and J_2 are the moments of inertia for the motor and the roll, respectively.

We now derive the dimensionless (normalized) equations for the moments of the system to obtain a unified approach to the system. We select as base motor angular rotation

$$\phi_b = \int_0^1 \omega_b \, \mathrm{d}t. \tag{15}$$

We add to the system equations the equation for armature current for $F = \text{const}$, which we use as a motor DC motor with separate excitation and armature voltage control. We assume that the friction of the masses is very small, i.e., $M_{f1} = M_{f1} = 0$, to make the problem manageable. The dimensionless equations are thus the following:

$$\bar{\phi}_1 = \frac{1}{s}\bar{\omega}_1, \tag{16}$$

$$\bar{\phi}_2 = \frac{1}{s}\bar{\omega}_2, \tag{17}$$

$$\bar{i}_a = \bar{M}_m = \frac{1}{T_a s}\left(\frac{\bar{e}_c - \omega_1}{\rho_a} - i_a\right), \tag{18}$$

$$\bar{\omega}_1 = \frac{1}{T_{m1}s}[\bar{M}_m - \bar{M}_y - k_c(\bar{\omega}_1 - \bar{\omega}_2)], \tag{19}$$

$$\bar{M}_y = \frac{1}{T_c s}(\bar{\omega}_1 - \bar{\omega}_2), \tag{20}$$

$$\bar{\omega}_2 = \frac{1}{T_{m2}s}[\bar{M}_y - \bar{M}_c - k_c(\bar{\omega}_1 - \bar{\omega}_2). \tag{21}$$

Here $T_{m1} = J_1\omega_b/M_b$ is the mechanical time constant of the motor. $T_{m2} = J_2\omega_b/M_b$ is the mechanical time constant of the roll. $k_c = b\omega_b/M_b$ is the friction coefficient of the spindle and is assumed to be zero in our case. $T_c = M_b/c\omega_b$ is the mechanical time constant of the torsional rigidity of the spindle. T_a is the mechanical time constant of the armature voltage. e_c is the armature voltage. $s \equiv \mathrm{d}/\mathrm{d}t$. ρ_a is the armature resistance.

Using the absolute changes of the variables, instead of the variables themselves, and assuming that the load is independent of the speed, we can linearize the system and derive a transfer function from the block diagram (Fig. 5) and the equations above.

We use the following measured state variables (inputs to the controller). $x_1 = \Delta\bar{i}_a$ is the armature current. $x_2 = \Delta\bar{w}_1$ is the angular velocity of the roll. $x_3 = \Delta\bar{M}_y$ is the torque between the roll and the motor. $x_4 = \Delta\bar{w}_2$ is the angular velocity of the motor. We use the armature voltage $u_1 = \Delta\bar{e}_c$ as the control variable and the torque of the roll (load) $u_2 = \Delta\bar{M}_c$ as the disturbance variable—the two inputs to the plant. The plant has following two outputs. $y_1 = \Delta\bar{i}_a$ is the armature current. $y_2 = \Delta\bar{w}_1$ is the angular velocity of the roll.

The state representation of the system is

$$\dot{X} = AX + BU,$$

$$Y = CX,$$

$$A = \begin{bmatrix} \frac{-1}{T_a} & \frac{-1}{\rho_a T_a} & 0 & 0 \\ \frac{1}{T_{m1}} & 0 & \frac{-1}{T_{m1}} & 0 \\ 0 & \frac{1}{T_c} & 0 & \frac{-1}{T_c} \\ 0 & 0 & \frac{1}{T_{m2}} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{\rho_a T_a} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{-1}{T_{m2}} \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$X = \begin{bmatrix} \Delta\bar{i}_a \\ \Delta\bar{\omega}_1 \\ \Delta\bar{M}_y \\ \Delta\bar{\omega}_2 \end{bmatrix}, \quad U = \begin{bmatrix} \Delta\bar{e}_c \\ \Delta\bar{M}_c \end{bmatrix}, \quad Y = \begin{bmatrix} \Delta\bar{i}_a \\ \Delta\bar{\omega}_1 \end{bmatrix}.$$

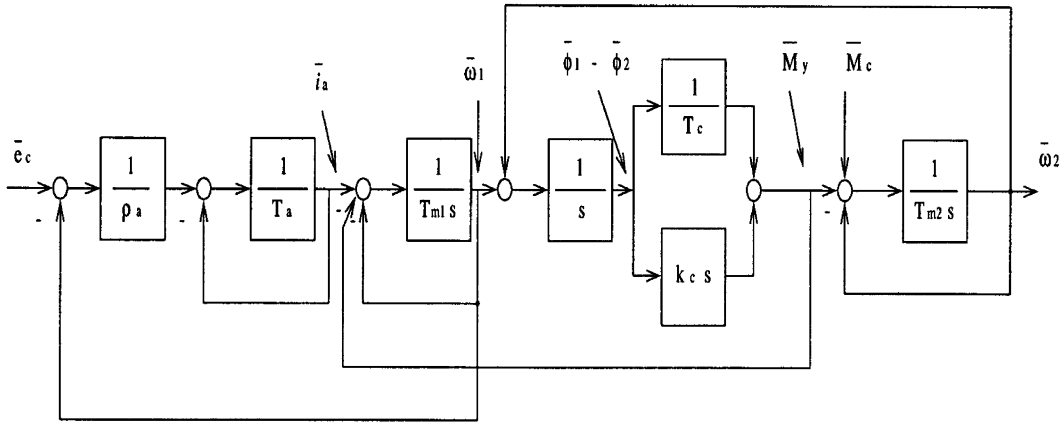

Fig. 5. Block diagram of the modeled system shown in Fig. 4.

The transfer function of the system is

$$G(s) = \frac{1}{F(s)\rho_a T_a T_{m1} T_c T_{m2}}$$

$$\times \begin{bmatrix} T_m(T_y^2 s^2 + 1) & 1 \\ \gamma T_y^2 s^2 + 1 & -\rho_a(T_a s + 1) \\ T_{m2} s & \rho_a T_{m1} s(T_a s + 1) + 1 \\ 1 & -\rho_a(T_a s + 1) + T_c s \end{bmatrix}, \quad (22)$$

where $T_m = T_{m1} + T_{m2}$ is the total mechanical time constant of the system. $\gamma = T_m/T_{m1} = (J_1 + J_2)/J_1$ is the ratio of the masses, $\gamma = 1$ in our case. $T_y = \sqrt{(T_c T_{m1} T_{m2})/T_m}$ is the mechanical time constant of the elastic oscillation of the masses.

The characteristic equation of the system is

$$F(s) = \frac{\rho_a T_m s(T_a s + 1)(T_y^2 s^2 + 1) + (\gamma T_y^2 s^2 + 1)}{\rho_a T_a T_{m1} T_c T_{m2}}. \quad (23)$$

The two-mass system is an oscillating system with a resonance frequency of

$$\omega_n = \sqrt{\frac{c_{12}(J_1 - J_2)}{J_1 J_2}}. \quad (24)$$

Here c_{12} is the coefficient (modulus) of torsional rigidity of the spindle,

$$c_{12} = \frac{G I_p}{l}. \quad (25)$$

G is the modulus of rigidity. I_p is the polar moment of the inertia of the area. l is the length of the spindle.

Applying disturbance load torque M_c as a step function to a uncontrolled roll drive system, we can see that the open-loop system is marginally stable and have a dynamic error of 18.5% and a steady-state error of 6.12% (Fig. 6). Additionally, for the same step change in disturbance load torque, the angular velocity of the motor shows constant undamped oscillation due to slack (Fig. 7).

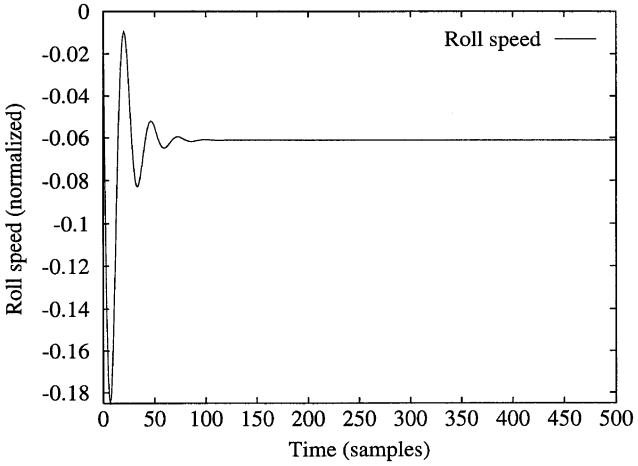

Fig. 6. Response of the roll speed to a step change of the load torque for an uncontrolled system.

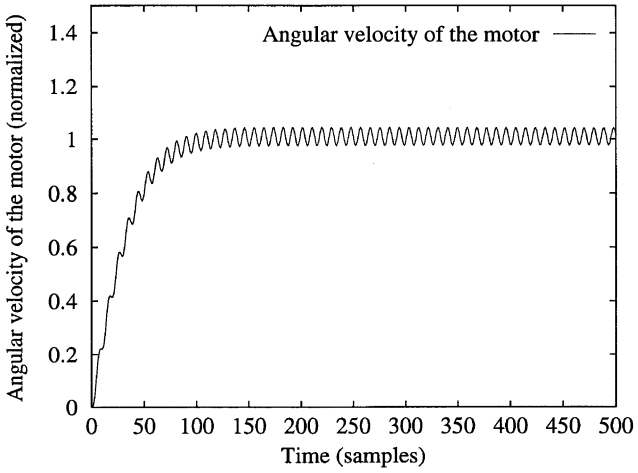

Fig. 7. Response of the motor angular velocity to a step change of the load torque for an uncontrolled system.

We have modeled the two-mass system with a fourth-order linearized state–space model, and have used the PI state controller for speed and current control. State

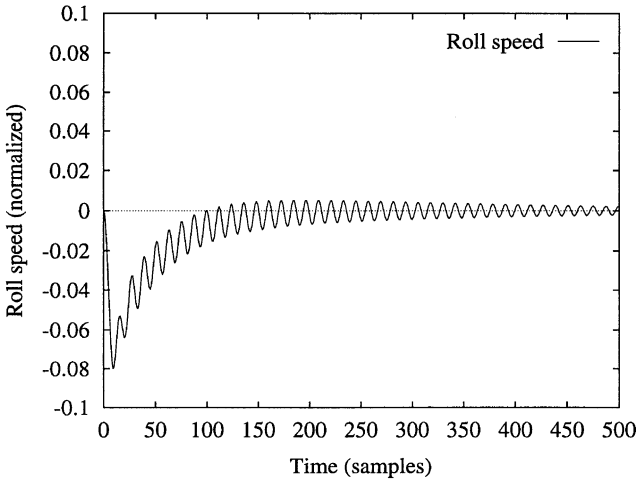

Fig. 8. Response of the roll speed to a step change of the load torque using the PI controller.

controllers are better suitable for a process with many state variables than simple PI controllers. The resulting graph (Fig. 8) shows the "ringing-type" damped oscillation of the masses, characteristic for such systems. The oscillation of the masses is a sign that, after the perturbation, the PI controller is no longer optimized for the parameters of the system, due to unmodeled high-order dynamics. In a system with slack, it can be observed that the slack causes a limit cycle. The amplitude is dependent on the amount of slack. The frequency is determined by the natural frequency of the system ω_n. Through a typical overshoot, the slack area is generally passed. This causes an oscillation which is worsened by integrating the deviation between the desired and the actual value which creates large regulation moments, and consequently some high amplitude oscillations. Hence, the PI controller is not suitable for such a system.

5. Simulation results of the RCC and DGFNN approaches

We will use as a reference model Butterworth's (Åström and Wittenmark, 1990; Ogata, 1987) characteristic equation for the fourth-order system, as our system is of the fourth order

$$F(s) = s^4 + 2.6\omega_n s^3 + 3.4\omega_n^2 s^2 + 2.6\omega_n^3 s + \omega_n^4, \qquad (26)$$

where ω_n is a natural frequency of the system.

This form of characteristic equation gives us a damping ratio $\xi = \sqrt{2}/2$, and the settling time can be determined through approximate relationship $t_s = 4/\xi\omega_n$. The same characteristic equation was used to tune the PI speed and current controllers in the previous section. We use the input–output data generated using this model to train the DGFNN and the RCC. We generate 1000 input–output data pairs normalized in the interval $[-1, 1]$ using a sampling time $T_s = 0.005$ s. The data are splitted into two sets, i.e., 500 samples for training, another 500 for validation of the network's generalization ability, and the whole data set is used for final testing of a trained network. The same data are used with an RCC using quickpropagation through time (Fahlman, 1988; Zipser, 1990) learning algorithm implemented with the Stuttgart Neural Network Simulator (1995), with a learning rate $\eta = 0.005$, a maximum growth parameter $\mu = 1.75$, a weight decay term $v = 0.0001$, and a maximum tolerated difference between a teaching value and an output $d_{max} = 0.1$. Both the DGFNN and the RCC are trained using on-line learning. After few hundred trials, we find that to get steady-state error below 1% the RCC requires, on average, 7000 epoch. Furthermore, the RCC is unsuccessful 90% of the time due to memory limitations and random weight initialization. On the contrary, the DGFNN is successful every time, and required on average 80 epoch to achieve similar results. Simulation results are presented in Fig. 9, where the result of the PI controller is also presented for comparison. We note that Fig. 9 shows the best performance of the RCC, as the RCC gives widely different results due to random weight initialization, whereas the DGFNN always produces satisfactory results. The mean squared error of the DGFNN is $\mathrm{RMSE}_{FNN} = 0.001244$ and that of the RCC, when successful, is $\mathrm{RMSE}_{RCC} = 0.001190$. Hence, though the RCC, when at its best performance, is able to achieve slightly better results in controlling the torsional vibration of tandem cold-rolling mill spindles in the presence of the load torque disturbances than the DGFNN, the RCC is not suitable for real-time implementation due to memory and speed limitations, as well as for the requirements of reliable performance. In comparison, the DGFNN is far more promising for real-time control.

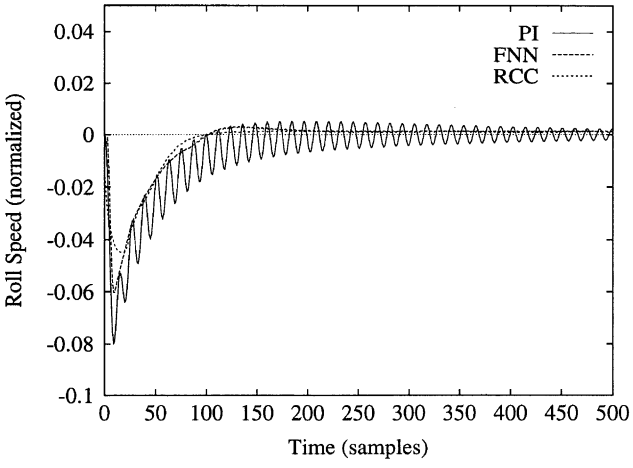

Fig. 9. Responses of the roll speed to a step change of the load torque for the PI, the FNN, the RCC controllers.

6. Conclusion

In this paper we have described a dynamically generated fuzzy neural network (DGFNN) which can be used to control a complex system with a minimal knowledge about the system, such as input–output measurements, and with satisfactory performance both for tracking the required signal and for disturbance rejection. The DGFNN is able to reduce heavy computation involved in crisp neural control, since the DGFNN algorithm updates the weights only for the nodes representing the maximum error at the time, i.e., it updates weights locally, unlike the MLP in which all weights are updated (global updating). The DGFNN also eliminates the problem of trial-and-error constructions common to many fuzzy controllers. Furthermore, unlike the MLP, the DGFNN is capable of expressing the knowledge acquired from input–output data in terms of fuzzy inference rules, thus avoiding black-box behaviors or a separate phase for rule extraction.

The DGFNN is complete with the following powerful features drawn or modified from the existing literature: (1) a small FNN is created from scratch—there is no need to specify initial network architecture, initial membership functions, or initial weights, (2) fuzzy rules are constantly combined and pruned to minimize the size of the network while maintaining accuracy, irrelevant inputs are detected and deleted, and (3) membership functions and network weights are trained with a backpropagation-type algorithm. While the existing FNNs possess *some* of these features, none of the existing FNNs seems to possess *all* these features.

The DGFNN with both structure and parameter learning can provide a computationally efficient solution to control of many real-world complex systems in the presence of disturbances. As a demonstration of the applicability of the DGFNN, we have used it for torsional vibration control of tandem cold-rolling mill spindles and compared with the PI and the RCC controllers. We have shown that the conventional PI controller was not able to suppress the torsional vibration of the spindles. While both the RCC and the DGFNN increase the control precision and robustness compared to the PI controller, the DGFNN controller gives more reliable performance and is far more computationally efficient, i.e., it learns much faster and requires less computer memory, and is therefore more promising for real-time control in practical applications.

Appendix. The system parameters

DC motor with separate excitation.
Nominal power = 4300 kW.
Nominal voltage = 750 V.
Nominal current = 6100 A.
Mechanical time constant of the elastic oscillation of the masses $T_y = 0.0074$ s.
Mechanical time constant of the motor $T_{m1} = 0.102$ s.
Mechanical time constant of the armature voltage $T_a = 0.055$ s.
Armature resistance $\rho_a = 0.104$.
Mechanical time constant of the roll $T_{m2} = 0.102$ s.
Mechanical time constant of the torsional rigidity of the spindle $T_c = 0.00107$ s.
Sampling frequency $T = 0.005$ s.

References

Ash, T., 1989. Dynamic node creation in backpropagation networks. Connection Science 1 (4), 365–375.

Åström, K., Wittenmark, B., 1990. Computer Controlled Systems, Theory and Design, 2nd Edition. Prentice-Hall, Englewood Cliffs, NJ.

Berenji, H., Khedkar, P., 1993. Learning and tuning fuzzy logic controllers through reinforcements. IEEE Transactions on Neural Networks 3 (5), 724–740.

Brown, M., Harris, C., 1994. Neurofuzzy Adaptive Modelling and Control. Prentice-Hall, Englewood Cliffs, NJ.

Bryant, G., 1973. Automation of Tandem Mills. The Iron and Steel Institute, London.

Cai, L.Y., Kwan, H.K., 1998. Fuzzy classifications using fuzzy inference networks. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 28 (3), 334–347.

Cybenko, G., 1989. Approximations by superposition of a sigmoidal function. Mathematics of Control, Signals, and Systems 2, 303–314.

Driankov, D., Hellendoorn, H., Palm, R., 1995. Some research directions in fuzzy control. In: Nguyen, H., Sugeno, M., Tong, R., Yager, R. (Eds.), Theoretical Aspects of Fuzzy Control. Wiley, New York, pp. 281–312.

Dubois, D., Prade, H., 1982. A unifying view of comparison indices in a fuzzy set theoretic framework. In: Yager, R.R. (Ed.), Fuzzy Sets and Possibility Theory: Recent Developments. Pergamon, New York.

Elman, J., 1990. Finding structure in time. Cognitive Science 14 (2), 179–211.

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.

Fahlman, S., 1991. The recurrent cascade-correlation learning architecture. Technical Report CMU-CS-91-100, School of Computer Science, Carnegie Mellon University.

Fahlman, S., Lebiere, C., 1990. The cascade-correlation learning architecture. In: Touretzky, D.S. (Ed.), Advances in Neural Information Processing Systems, Vol. 2. Morgan Kaufmann, San Mateo, CA, pp. 524–532.

Frean, M., 1990. The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Computation 1, 198–209.

Funahashi, K., Nakamura, Y., 1993. Approximation of dynamical systems by continuous time recurrent neural networks. Neural Networks 6 (6), 801–806.

Giles, C.L., Chen, D., Sun, G.Z., Chen, H.H., Lee, Y.C., Goudreau, M.W., 1995. Constructive learning of recurrent neural networks: limitations of recurrent cascade correlation and a simple solution. IEEE Transactions on Neural Networks 6, 829–836.

Harakawa, T., Yui, K., Sumitani, E., 1987. Development of spindle torsional vibration control system using observer for tandem cold mill in steel production process. Automatic Control—Proceedings of the 10th Triennial World Congress of the IFAC, Vol. 3. pp. 109–113.

Harris, C., Moore, C., Brown, M., 1993. Intelligent Control: Aspects of Fuzzy Logic and Neural Nets. World Scientific, Singapore.

Higgins, C.M., Goodman, R.M., 1994. Fuzzy rule-based networks for control. IEEE Transactions on Fuzzy Systems 2 (1), 82–88.

Horikawa, S., Furahashi, T., Uchikawa, Y., 1992. On fuzzy modelling using neural networks with the backpropagation algorithm. IEEE Transactions on Neural Networks 3, 801–806.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks 2 (5), 359–366.

Hunt, K., Sbarbaro, D., Zbikowski, R., Gawthrop, P., 1992. Neural networks for control systems—a survey. Automatica 28 (6), 1083–1112.

Ishibuchi, H., Tanaka, H., Okada, H., 1994. Interpolation of fuzzy if–then rules by neural networks. International Journal of Approximate Reasoning 10, 3–27.

Jang, J.-S.R., 1993. ANFIS: adaptive-network-based fuzzy inference systems. IEEE Transactions on Systems, Man, and Cybernetics 23 (3), 665–685.

Jin, L., Nikiforuk, P., Gupta, M., 1995. Fast neural learning and control of discrete-time nonlinear systems. IEEE Transactions on Systems, Man, and Cybernetics 25 (3), 479–488.

Juang, C., Lin, C., 1999. A recurrent self-organizing neural fuzzy inference network. IEEE Transactions on Neural Networks 10 (4), 828–845.

Kawaguchi, T., Ueyama, T., 1989. Steel Industry. Control System, Vol. 2. Gordon and Breach Science Pub., New York, NY.

Kosko, B., 1991. Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence. Prentice-Hall, Englewood Cliffs, NJ.

Kuo, B., 1991. Automatic Control Systems, 6th Edition. Prentice-Hall, Englewood Cliffs, NJ.

Lin, C.-T., Lee, C.S.G., 1994. Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. IEEE Transactions on Fuzzy Systems 2 (1), 46–63.

Mezard, M., Nadal, J., 1989. Learning in feedforward layered networks: the tiling algorithms. Journal of Physics A 22, 2191–2203.

Nadal, J., 1989. Study of a growth algorithm for feedforward neural network. International Journal of Neural Systems 1, 55–59.

Narendra, K., Mukhopadhyay, S., 1994. Adaptive control of nonlinear multivariable systems using neural networks. Neural Networks 7 (5), 737–752.

Narendra, K., Pathasarathy, K., 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1 (1), 4–27.

Nauck, D., Kruse, R., 1993. A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In: Proceedings of the IEEE International Conference on Neural Networks ICNN'93, San Francisco, CA, pp. 1022–1027.

Neumerkel, D., Franz, J., Krüger, L., Hidiroglu, A., 1994. Real-time application of neural model predictive control for an induction servo drive. Proceedings of the Third IEEE Conference on Control Applications, Vol. 1, Glasgow, UK, pp. 433–438.

Nie, J., Linkens, D., 1993. Learning control using fuzzified self-organizing radial basis function network. IEEE Transactions on Fuzzy Systems 1 (4), 280–287.

Nie, J., Linkens, D., 1995. Fuzzy-Neural Control: Principles, Algorithms and Applications. Prentice-Hall, Englewood Cliffs, NJ.

Ogata, K., 1987. Discrete-Time Control Systems. Prentice-Hall, Englewood Cliffs, NJ.

Sbarbaro-Hofer, D., Neumerkel, D., Hunt, K., 1993. Neural control of a steel rolling mill. IEEE Control Systems Journal 13 (3), 69–75.

SNNS (Stuttgart Neural Network Simulator) User Manual, 1995. Version 4.1, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems (IPVR), 6/95.

Sugeno, M., Tanaka, K., 1991. Successive identification of fuzzy model and its application to prediction of complex system. Fuzzy Sets and Systems 42, 315–344.

Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modelling and control. IEEE Transactions on Systems, Man and Cybernetics 15 (1), 116–132.

Wang, L.-X., 1994. Adaptive Fuzzy Systems and Control: Design and Stability Analysis. PTR Prentice-Hall, Englewood Cliffs, NJ.

Wang, L., Langari, R., 1995. Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques. IEEE Transactions on Fuzzy Systems 3, 454–458.

Wang, L.-X., Mendel, J., 1992. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Transactions on Neural Networks 3 (5), 807–814.

Yen, J., Langari, R., Zadeh, L. (Eds.), 1995. Industrial Applications of Fuzzy Logic and Intelligent Systems. IEEE Press, New York.

Zipser, D., 1990. Subgrouping reduces complexity and speeds up learning in recurrent networks. In: Touretzky, D.S. (Ed.), Advances in Neural Information Processing Systems, Vol. 2. Morgan Kaufmann, San Mateo, CA, pp. 638–641.