ELSEVIER

Fuzzy Sets and Systems 73 (1995) 291-312

FUZZY sets and systems

A model reference control structure using a fuzzy neural network

Yie-Chien Chen, Ching-Cheng Teng*

Institute of Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan

Received May 1994; revised August 1994

Abstract

In this paper, we present a design method for a model reference control structure using a fuzzy neural network. We study a simple fuzzy-logic based neural network system. Knowledge of rules is explicitly encoded in the weights of the proposed network and inferences are executed efficiently at high rate. Two fuzzy neural networks are utilized in the control structure. One is a controller, called the fuzzy neural network controller (FNNC); the other is an identifier, called the fuzzy neural network identifier (FNNI). Adaptive learning rates for both the FNNC and FNNI are guaranteed to converge by a *Lyapunov* function. The on-line control ability, robustness, learning ability and interpolation ability of the proposed model reference control structure are confirmed by simulation results.

Keywords: Fuzzy logic; Neural network; Fuzzy neural network; Model reference control

1. Introduction

Recently, fuzzy neural network control systems have been extensively studied. For instance, Lin [12] proposed a general neural network model for a fuzzy logic control and decision system, which is trained to control an unmanned vehicle by combining unsupervised and supervised learning. Horikawa et al. [6] presented a fuzzy neural network that learned expert control rules, while Lee [10] combined Barto's adaptive neurons with a fuzzy logic controller to deal with the pole balancing problem. However, in all of these systems a teacher responsible for training is required. Furthermore, adaptation to changes in the environment is not provided for.

The structure of the fuzzy neural network presented by Lin [12] consists of five layers, which is a little complicated. The proposed fuzzy neural network is a slight modification of that in [12, 6]. Thus, we have a four-layer fuzzy neural network structure and the calculation of the proposed system is simpler than *Horikawa's TYPE-I FNN*. The main advantages of the structure we adopted are (1) the ability to learn from experience, (2) a high computation rate, (3) the easily understandable manner in which knowledge acquired is expressed and (4) a high degree of robustness and fault tolerance.

* Corresponding author.

In the conventional adaptive control literature, there are two distinct adaptive control categories: (1) direct adaptive control and (2) indirect adaptive control [13]. In direct adaptive control, the parameters of the controller are directly adjusted to reduce some norm of the output error (between the plant and the reference model). On the other hand, in indirect adaptive control, the parameters of the plant are estimated and the controller is chosen assuming that the estimated parameters represent the true values of the plant parameters. In the control system, if the plant is unknown, many people simply ignore the sensitivity and use the direct control approach.

In this paper, we propose a model reference control structure that uses a fuzzy neural network. The proposed model reference control structure belongs to indirect adaptive control, and a controlled plant is identified by the fuzzy neural network identifier (FNNI), which provides information about the plant to the fuzzy neural network controller (FNNC). This structure is a real adaptation system that can learn to control a complex system and adapt to a wide range of variations in plant parameters. Unlike most other adaptive learning neural controllers [1, 2, 5, 9, 11, 13, 14, 17], the FNNC presented in this paper is based not only on the theory of neural network computing but also on that of fuzzy logic [3].

Though the proposed control scheme is a slight modification of those in [4, 13], we believe that our structure is more reasonable for a fuzzy logic control system. Since the place for the reference model (RM) in the proposed system is specially considered, the FNNC is designed such that the actual output of the system will track the desired output of the reference model. Moreover, we can simply take the error (between the actual output and the desired output) and the change in this error as the inputs for FNNC.

We also apply some of the theorems in [9] to develop convergence theorems for both FNNI and FNNC. To guarantee convergence and for faster learning, an analytical method based on the Lyapunov function is proposed to find the adaptive learning rates for FNNI and FNNC. This paper is organized as follows. In Section 2, a simple fuzzy-logic based neural network system is studied. Section 3 presents a model reference adaptive control structure using a fuzzy neural network. In this structure, the control action is updated on-line using the information stored in a fuzzy neural network identifier (FNNI). The convergence of the FNN-based system is investigated in Section 4. In Section 5, examples are presented to illustrate the performance of the control system. Concluding remarks are given in Section 6.

2. Fuzzy neural network

In this section we will present a simple fuzzy logic system implemented by using a multilayer feedforward neural network. A schematic diagram of the proposed fuzzy neural network (FNN) structure with three input variables, two term nodes for each input variable, two output nodes, and eight rule nodes is shown in Fig. 1. The system consists of four layers. Nodes in layer one are *input nodes* which represent input linguistic variables. Nodes in layer two are *membership nodes* which act like membership functions. Each membership node is responsible for mapping an input linguistic variable into a possibility distribution for that variable. The rule nodes reside in layer three. Taken together, all the layer three nodes form a fuzzy rule base. Layer four, the last layer, contains the output variable nodes.

The links between the membership nodes and the rule nodes are the antecedent links and those between the rule nodes and the output nodes are the consequence links. For each rule node, there is at most one antecedent link from a membership node of a linguistic variable. Hence there are $\prod_i |T(x_i)|$ rule nodes in the proposed FNN structure. Here $|T(x_i)|$ denotes the number of fuzzy partitions of input linguistic variable x_i. Moreover, all consequence links are fully connected to the output nodes and interpreted directly as the strength of the output action. In this way, the consequence of a rule is simply the product of the rule node output, which is the firing strength of the fuzzy rule and the consequence link. Thus, the overall net output is treated as a linear combination of the consequences of all rules instead of the complex composition, a rule of inference and the defuzzification process. This fuzzy neural network is a slight modification of the network

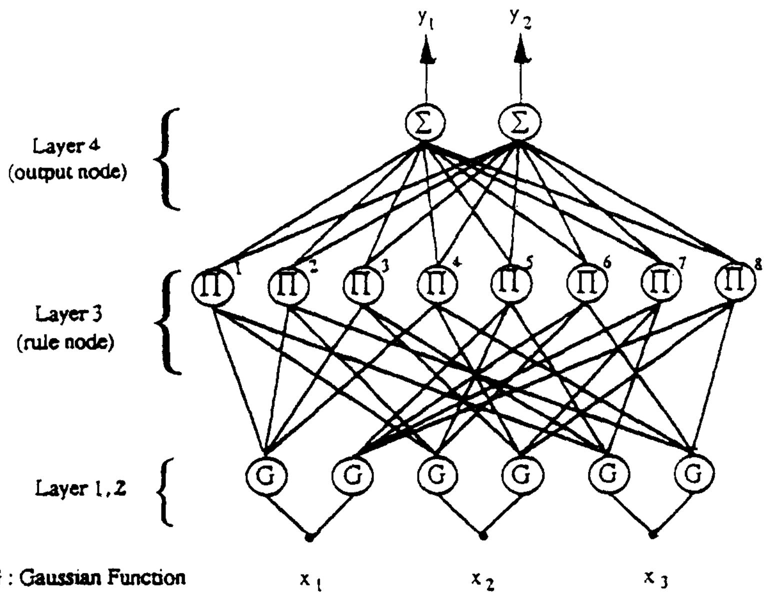

Fig. 1. Schematic diagram of a fuzzy neural network.

reported by Lin [12]. The interested readers are referred to Lin [12] for a more detailed explanation of the network.

2.1. Reasoning method

For an n-input–one-output system, let x_i be the ith input linguistic variable and define α_k as the firing strength of rule k, which is obtained from the product of the grades of the membership functions $\mu_{A_i^k}(x_i)$ in the antecedent. If w_k represents the kth consequence link weight, the inferred value y^* is then obtained from the weighted sum of its inputs, i.e. $\sum_k w_k\alpha_k$. The proposed fuzzy neural network realizes the inference as follows

$$R^k\colon \text{IF } x_1 \text{ is } A_1^k(x_1), \ldots, \text{and } x_n \text{ is } A_n^k(x_n), \quad \text{then } y = w_k,\ k = 1,2,\ldots,m$$

$$y^* = \sum_{k=1}^{m} \alpha_k w_k, \quad \alpha_k = \prod_{i=1}^{n} \mu_{A_i^k}(x_i).$$

The reasoning method is a variation of the reasoning method introduced by Sugeno [15], in which the consequence of a rule is a function of input variables. For the proposed FNN, this function is replaced by a constant value and a different defuzzification process is used.

2.2. Basic nodes operation

Next, we shall indicate the signal propagation and the basic function of every node in each layer.

Layer 1: input layer

For the jth node of layer 1, the net input and the net output are represented as:

$$net_j^1 = w_{ij}^1 \cdot x_i^1, \quad i = j, \qquad y_j^1 = f_j^1(net_j^1) = net_j^1,$$

where the weights w_{ij}^1 are assumed to be unity and x_i^1 represents the ith input to the jth node of layer 1.

Layer 2: membership layer

In this layer, each node performs a membership function. The Gaussian function, a particular example of radial basis functions, is adopted here as a membership function. Then,

$$net_j^2 = \mu_{A_{ij}}(m_{ij}, \sigma_{ij}) = -\frac{(x_i^2 - m_{ij})^2}{(\sigma_{ij})^2}, \qquad y_j^2 = f_j^2(net_j^2) = \exp(net_j^2),$$

where m_{ij} and σ_{ij} are, respectively, the mean (or center) and the variance (or width) of the Gaussian function in the jth term of the ith input linguistic variable x_i^2.

Layer 3: rule layer

The links in this layer are used to implement the antecedent matching. The matching operation or the fuzzy AND aggregation operation is chosen as the simple PRODUCT operation instead of the MIN operation. Then, for the jth rule node

$$net_j^3 = \prod_i^n w_{ij}^3 x_i^3, \qquad y_j^3 = f_j^3(net_j^3) = net_j^3,$$

where w_{ij}^3 is also assumed to be unity.

Layer 4: output layer

Since the overall net output is a linear combination of the consequences of all rules, the net input and output of the jth node in this layer are simply defined by

$$net_j^4 = \sum_i^m w_{ij}^4 x_i^4, \qquad y_j^4 = f_j^4(net_j^4) = net_j^4,$$

where the link weight w_{ij}^4 is the output action strength of the jth output associated with the ith rule.

Note that net_j, f_j are the summed net input (or activation level) and activation function of node j, respectively, and the superscript denotes the layer number. From the above configuration, by modifying the centers and widths of layer 2 and the link weights of layer 4, the membership functions can be fine-tuned and all the consequence strengths of fuzzy rules could be identified respectively. The learning process to train the proposed fuzzy neural network will be discussed in the following section.

2.3. Supervised gradient descent learning

The adjustment of the parameters in the proposed FNN can be divided into two tasks, corresponding to the IF (premise) part and THEN (consequence) part of the fuzzy logical rules. In the premise part, we need to initialize the center and width for Gaussian functions. To determine these initial terms, a self-organization-map (SOM) [8] and fuzzy-c-means (FCM) [16] are commonly used. Another simple and intuitive method of doing this is to use normal fuzzy sets to fully cover the input space. Since the final performance will depend mainly on supervised learning, we choose normal fuzzy sets in this paper. In the consequence part, the parameters are output singletons. These singletons are initialized with small random values, as in a pure neural network.

A supervised learning law is used to train the proposed model. The basis of this algorithm is simply gradient descent. The derivation is the same as that of the back-propagation learning law. By recursive applications of the chain rule, the error term for each layer is first calculated. The adaptation of weights to the corresponding layer is then given. Next, we will begin to derive the learning law for each layer in the feedbackward direction.

Layer 4: If the cost function to be minimized is defined as

$$E = \tfrac{1}{2}\sum_j (d_j^4 - y_j^4)^2 = \tfrac{1}{2}\sum_j (d_j^4 - f_j^4(net_j^4))^2,$$

where d_j^4 is the desired output and y_j^4 is the current output of the jth output node, the error term to be propagated is given by

$$\delta_j^4 = \frac{-\partial E}{\partial net_j^4} = \frac{-\partial E}{\partial f_j^4}\frac{\partial f_j^4}{\partial net_j^4} = d_j^4 - y_j^4$$

then, the weight w_{ij}^4 is updated by the amount

$$\Delta w_{ij}^4 = -\frac{\partial E}{\partial w_{ij}^4} = -\frac{\partial E}{\partial f_j^4}\frac{\partial f_j^4}{\partial net_j^4}\frac{\partial net_j^4}{\partial w_{ij}^4} = (d_j^4 - y_j^4)\cdot y_i^3 = \delta_j^4 \cdot y_i^3.$$

Layer 3: Since the weights in this layer are unity, none of them is to be modified. Only the error term needs to be calculated and propagated.

$$\delta_j^3 = \frac{-\partial E}{\partial net_j^3} = \frac{-\partial E}{\partial f_j^3}\frac{\partial f_j^3}{\partial net_j^3} = -\sum_k \frac{\partial E}{\partial net_k^4}\frac{\partial net_k^4}{\partial y_j^3} = \sum_k \delta_k^4 w_{jk}^4.$$

Layer 2: The multiplication operation is done in this layer. The adaptive rule for m_{ij} and σ_{ij} are as follows. First, the error term is computed,

$$\begin{aligned}\delta_j^2 &= \frac{-\partial E}{\partial net_j^2} = \frac{-\partial E}{\partial f_j^2}\frac{\partial f_j^2}{\partial net_j^2}\\ &= -\left(\sum_k \frac{\partial E}{\partial net_k^3}\frac{\partial net_k^3}{\partial y_j^2}\right)\cdot\frac{\partial f_j^2}{\partial net_j^2} = \left(\sum_k \delta_k^3 \prod_{i\neq j} y_i^2\right)\cdot \exp(net_j^2)\\ &= \left(\sum_k \delta_k^3 \prod_{i\neq j} y_i^2\right)\cdot y_j^2 = \sum_k \delta_k^3 \cdot y_k^3,\end{aligned}$$

where the subscript k denotes the rule node in connection with the jth node in Layer 2. Then, the adaptive rule of m_{ij} is

$$\Delta m_{ij} = -\frac{\partial E}{\partial m_{ij}} = -\frac{\partial E}{\partial net_j^2}\frac{\partial net_j^2}{\partial m_{ij}} = \delta_j^2 \cdot \frac{2(y_i^1 - m_{ij})}{\sigma_{ij}^2}$$

and the adaptive rule of σ_{ij} is

$$\Delta \sigma_{ij} = -\frac{\partial E}{\partial \sigma_{ij}} = -\frac{\partial E}{\partial net_j^2}\frac{\partial net_j^2}{\partial \sigma_{ij}} = \delta_j^2 \cdot \frac{2(y_i^1 - m_{ij})^2}{\sigma_{ij}^3}.$$

This completes the derivation of the supervised gradient descent learning algorithm. In the following section, the proposed FNN will be employed as a controller and an identifier. A controlled plant is identified by the FNNI, which provides information about the plant to the FNNC.

3. Model reference control structure using FNN

Fuzzy logic systems and neural networks can be exploited to emulate the capabilities of the human brain. Merging these two different disciplines makes it possible to develop a unified system that reasons and learns by experience and adapts to changes in plant parameters.

Fig. 2 shows the proposed model reference control structure using a fuzzy neural network. The proposed control scheme must perform two major tasks: (1) system identification and (2) plant control. The former is achieved by using the proposed fuzzy neural network identifier (FNNI) to estimate the dynamics of the controlled plant. The latter is achieved by using the proposed fuzzy neural network controller (FNNC) to generate the control signals.

The control action issued by the FNNC is updated by observing the controlled results through the FNNI. The adaptive FNNC has many useful features: (1) it can self-organize its control law during the control process; (2) it is able to make inferences using the control law encoded in the FNNC; (3) it is capable of high-speed parallel computation; and (4) it adapts automatically to changes in plant parameters.

In this paper the control law learned by the FNNC can be expressed explicitly in linguistic *cause-and-effect* rules, while a conventional neural controller can only encode the control law implicitly in variable weights. The structure of Fig. 2 is a slight modification of those in [4, 13].

3.1. Overall structure of the system

The fuzzy neural network identifier: FNNI

The purpose of the FNNI is to mimic the dynamic characteristics of the controlled plant. Training of the FNNI is similar to plant identification except that the plant identification here is done automatically by a fuzzy neural network which is capable of modelling nonlinear plants [3]. The FNNI is trained by the above algorithm to predict the state vector of the plant y_I, with the actual value of the state of the plant y_p used as the desired response. The training process ceases when the error signal between y_I and y_p becomes small enough. If changes in the system parameters or in the environment occur, the FNNI is triggered on again to begin relearning.

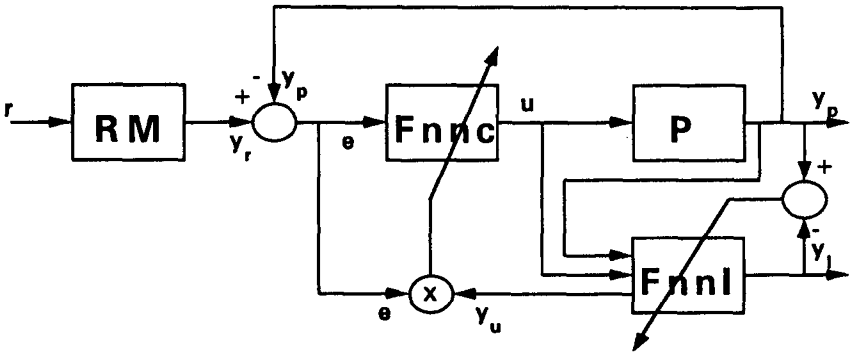

Fig. 2. Model reference control structure using fuzzy neural network.

The fuzzy neural network controller: FNNC

The fuzzy neural network here serves as a feedback controller. The FNNC is expected to approximate an optimal control surface. The optimal control surface is encoded in the form of fuzzy rules, which are represented by the interconnection weights embedded in the FNNC. Thus the weights can be modified to establish different control rules. As time goes by and the system accumulates more experience, it learns to control the plant more effectively. A controlled plant is identified by the FNNI, which provides information about the plant to the FNNC.

The reference model: RM

The reference model specifies the desired performance of the control system. The controller is designed such that the actual output of the system will track the desired output of the reference model. This goal can be achieved by minimizing $e = (y_r - y_p)$.

Note that our structure is different from that in [13], in which the reference model is placed on the upper side. We believe that our structure is more reasonable for a fuzzy system since we can simply take the error (between actual output and desired output) and the change in this error as the input for FNNC.

3.2. Training the FNNI and FNNC

Let the cost function, E_I, for training pattern k be proportional to the sum of the square of the difference between the plant output $y(k)$ and the actual output $y_I(k)$ of FNNI, and let E_I be defined by

$$E_I = \tfrac{1}{2}[y(k) - y_I(k)]^2. \tag{1}$$

Then the gradient of error in Eq. (1) with respect to an arbitrary weighting vector $W_I \in \Re^n$ becomes

$$\frac{\partial E_I}{\partial W_I} = e_I(k)\frac{\partial e_I(k)}{\partial W_I} = -\,e_I(k)\frac{\partial y_I(k)}{\partial W_I} = -\,e_I(k)\frac{\partial O_I(k)}{\partial W_I}, \tag{2}$$

where $e_I(k) = y(k) - y_I(k)$ is the error between the plant and the FNNI response. $O_I(k)$ is the actual output of the identifier (FNNI).

The weight can be adjusted using a gradient method:

$$W_I(k+1) = W_I(k) + \Delta W_I(k) = W_I(k) + \eta_I\left(-\frac{\partial E_I}{\partial W_I}\right), \tag{3}$$

where η_I is a learning rate.

Let the cost function, E_C, for training pattern k be proportional to the sum of the square of the difference between the desired output $y_r(k)$ of the reference model and the plant output $y(k)$, and let E_C be defined by

$$E_C = \tfrac{1}{2}[y_r(k) - y(k)]^2. \tag{4}$$

Then the gradient of error in Eq. (4) with respect to an arbitrary weighting vector $W_C \in \Re^n$ becomes

$$\begin{aligned}\frac{\partial E_C}{\partial W_C} &= e_C(k)\frac{\partial e_C(k)}{\partial W_C} = -\,e_C(k)\frac{\partial y(k)}{\partial W_C}\\ &= -\,e_C(k)\frac{\partial y(k)}{\partial u(k)}\cdot\frac{\partial u(k)}{\partial W_C} = -\,e_C(k)y_u(k)\cdot\frac{\partial O_C(k)}{\partial W_C}\end{aligned} \tag{5}$$

where $e_C(k) = y_r(k) - y(k)$ is the error between the actual plant and the desired reference output, $O_C(k)$ is the output of the controller (FNNC) and $S = y_u(k) = \partial y(k)/\partial u(k)$ is the plant sensitivity.

The weight can be adjusted using a gradient method.

$$W_C(k+1) = W_C(k) + \Delta W_C(k) = W_C(k) + \eta_C\left(-\frac{\partial E_C}{\partial W_C}\right). \tag{6}$$

where η_C is a learning rate.

The plant sensitivity can be computed as follows:

$$\begin{aligned}\frac{\partial y_j}{\partial u_i} &= \frac{\partial O_{\mathrm{I}aj}^{(4)}}{\partial u_i}\\ &= \sum_{a=1}^{R_\mathrm{I}}\left\{\frac{\partial O_{\mathrm{I}aj}^{(4)}}{\partial O_{\mathrm{I}ka}^{(3)}}\cdot\frac{\partial O_{\mathrm{I}ka}^{(3)}}{\partial u_i}\right\} = \sum_{a=1}^{R_\mathrm{I}} W_{aj}\cdot\left\{\frac{\partial O_{\mathrm{I}ka}^{(3)}}{\partial u_i}\right\}\\ &= \sum_{a=1}^{R_\mathrm{I}} W_{aj}\cdot\left\{\sum_{k=1}^{N_{m_i}}\frac{\partial O_{ka}^{(3)}}{\partial O_{ik}^{(2)}}\cdot\frac{\partial O_{ik}^{(2)}}{\partial u_i}\right\} = \sum_{a=1}^{R_\mathrm{I}} W_{aj}\cdot\left\{\frac{\partial O_{ka}^{(3)}}{\partial O_{ik}^{(2)}}\cdot\frac{\partial O_{ik}^{(2)}}{\partial u_i}\right\}\\ &= \sum_{a=1}^{R_\mathrm{I}} W_{aj}\cdot\left\{\prod_{b\neq i} O_{bL}^{(2)}\cdot\frac{\partial O_{ik}^{(2)}}{\partial u_i}\right\} = \sum_{a=1}^{R_\mathrm{I}} W_{aj}\cdot\left\{O_{ka}^{(3)}\cdot(-2)\cdot\frac{(u_i - m_{ik})}{(\delta_{ik})^2}\right\},\end{aligned} \tag{7}$$

where m_{ik} and σ_{ik} are, respectively, the mean (or center) and the variance (or width) of the Gaussian function in the kth term of the ith input linguistic variable u_i. The superscript denotes the layer number. The link weight W_{aj} is the output action strength of the jth output associated with the ath rule. The output $O_{\mathrm{I}ka}^{(3)}$ denotes the output of the third layer of the ath node associated with the output of the second layer of the kth node. N_{m_i} is the number of fuzzy sets of the ith input linguistic variable u_i. R_I is the number of rules in FNNI.

4. Convergence

This section develops some convergence theorems for selecting appropriate learning rates. If a small value is given for the learning rate η, convergence will be guaranteed. In this case the speed of convergence may be very slow, however on the other hand, if a large value is given for the learning rate η, the system may become unstable. Therefore, choosing an appropriate learning rate η is very important.

A discrete-type Lyapunov function can be expressed as

$$V(k) = \tfrac{1}{2}e^2(k), \tag{8}$$

where $e(k)$ represents the error in the learning process. Thus, the change in the Lyapunov function is obtained by

$$\Delta V(k) = V(k+1) - V(k) = \tfrac{1}{2}[e^2(k+1) - e^2(k)]. \tag{9}$$

The error difference can be represented by

$$e(k+1) = e(k) + \Delta e(k) = e(k) + \left[\frac{\partial e(k)}{\partial W}\right]^\mathrm{T}\Delta W, \tag{10}$$

where ΔW represents a change in an arbitrary weighting vector in $\Re^n$.

4.1. Convergence of the FNNI

From Eqs. (2), (3) we have

$$\Delta W_{\mathrm{I}} = -\eta_{\mathrm{I}} \frac{\partial E_{\mathrm{I}}(k)}{\partial W_{\mathrm{I}}(k)} = -\eta_{\mathrm{I}} e_{\mathrm{I}}(k) \frac{\partial e_{\mathrm{I}}(k)}{\partial W_{\mathrm{I}}(k)}$$

$$= \eta_{\mathrm{I}} e_{\mathrm{I}}(k) \frac{\partial y_{\mathrm{I}}(k)}{\partial W_{\mathrm{I}}(k)} = \eta_{\mathrm{I}} e_{\mathrm{I}}(k) \frac{\partial O_{\mathrm{I}}(k)}{\partial W_{\mathrm{I}}(k)}, \tag{11}$$

where W_{I} and η_{I} represent an arbitrary weighting and the corresponding learning rate in the FNNI and $O_{\mathrm{I}}(k)$ is the output of the FNNI. Then we have a general convergence theorem from [9].

Theorem 4.1 (Ku and Lee [9]). *Let η_{I} be the learning rate for the weights or the parameters of the* FNNI *and let $P_{\mathrm{I,max}}$ be defined as $P_{\mathrm{I,max}} \equiv \max_k \|P_{\mathrm{I}}(k)\|$, where $P_{\mathrm{I}}(k) = \partial O_{\mathrm{I}}(k)/\partial W_{\mathrm{I}}$ and $\|\cdot\|$ is the usual Euclidean norm in $\Re^n$. Then convergence is guaranteed if η_{I} is chosen as follows: $0 < \eta_{\mathrm{I}} < 2/P_{\mathrm{I,max}}^2$.*

Remark 4.2 (*Ku and Lee* [9]). $\eta_{\mathrm{I}}(2-\eta_{\mathrm{I}}) > 0$ or $\eta_{\mathrm{I}}(2-\eta_{\mathrm{I}})/P_{\mathrm{I,max}}^2 > 0$. This implies that any $\eta_{\mathrm{I}}, 0 < \eta_{\mathrm{I}} < 2$, guarantees convergence. However, the maximum learning rate, which guarantees the optimal convergence, corresponds to $\eta_{\mathrm{I}} = 1$, i.e. $\eta_{\mathrm{I}}^* = 1/P_{\mathrm{I,max}}^2$.

The following theorem is a slight modification of [9].

Theorem 4.3. *Let η_{I}^O be the learning rate for the* FNNI *weights W_{I}^O. Then the learning rate is chosen as follows: $0 < \eta_{\mathrm{I}}^O < 2/R_{\mathrm{I}}$, where R_{I} is the number of rules in the* FNNI.

Proof. Let $P_{\mathrm{I}}(k) = \partial O_{\mathrm{I}}(k)/\partial W_{\mathrm{I}}^O = Z^{\mathrm{I}}(k)$, where $Z^{\mathrm{I}} = [Z_1^{\mathrm{I}}, Z_2^{\mathrm{I}}, \ldots, Z_{R_{\mathrm{I}}}^{\mathrm{I}}]^{\mathrm{T}}$, in which Z_j^{I} is the output value of the third layer of the FNNI and R_{I} is the number of rules in the FNNI. Then we have $Z_j^{\mathrm{I}} \leqslant 1$ for all j, $\|P_{\mathrm{I}}(k)\| \leqslant \sqrt{R_{\mathrm{I}}}$ and $P_{\mathrm{I,max}}^2 = \max_k \|P_{\mathrm{I}}(k)\|^2 = R_{\mathrm{I}}$. From Theorem 4.1 we obtain $0 < \eta_{\mathrm{I}}^O < 2/R_{\mathrm{I}}$. □

In order to prove Theorems 4.6 and 4.11 we will need the following lemmas.

Lemma 4.4. *Let $g(y) = y e^{(-y^2)}$. Then $|g(y)| < 1$, $\forall y \in \Re$.*

Proof. We have $g'(y) = e^{-y^2} - 2y^2 e^{-y^2} = 0$, which implies that $y = \sqrt{2}/2$ and $y = -(\sqrt{2}/2)$ are two terminal values. Also, we have $g''(y) = (4y^3 - 6y)e^{-y^2}$, from which we obtain $g''(\sqrt{2}/2) = -2\sqrt{2}e^{-1/2} < 0$. So $y = \sqrt{2}/2$, $g(\sqrt{2}/2)$ is the maximum value. Also, $g''(-\sqrt{2}/2) = 2\sqrt{2}e^{-1/2} < 0$, so $y = -(\sqrt{2}/2)$, $g(-(\sqrt{2}/2))$ is a minimum value. Thus we have $|g(\sqrt{2}/2)| = |\sqrt{2}/2 e^{-1/2}| < 1$, $|g(-(\sqrt{2}/2))| = |-(\sqrt{2}/2)e^{-1/2}| < 1$. Therefore $|g(y)| < 1$, $\forall y \in \Re$. □

Lemma 4.5. *Let $f(y) = y^2 e^{(-y^2)}$. Then $|f(y)| < 1$, $\forall y \in \Re$.*

Proof. We have $f'(y) = 2ye^{-y^2} - 2y^3 e^{-y^2} = 0$, which implies that $y = -1, 0, 1$ are three terminal values. Also, we have $f''(y) = (2 - 10y^2 + 4y^4)e^{-y^2}$, from which we obtain $f''(0) = 2 > 0$, so $y = 0$, $f(0) = 0$ is a minimum value. Also, $f''(-1) = -4e^{-1} > 0$, so $y = -1$, $f(-1) = e^{-1}$ is a maximum value, and $f''(1) = -4e^{-1} > 0$, so $y = 1, f(1) = e^{-1}$ is a maximum value. Therefore $|f(y)| < 1$, $\forall y \in \Re$.

Theorem 4.6. *Let η_{I}^{m} and $\eta_{\mathrm{I}}^{\delta}$ be the learning rates for the* FNNI *parameters m_{I} and δ_{I}, respectively. Then the learning rates are chosen as follows: $0 < \eta_{\mathrm{I}}^{m} = \eta_{\mathrm{I}}^{\delta} < 2/R_{\mathrm{I}}[|W_{\mathrm{I,max}}^{O}|(2/\delta_{\mathrm{I,min}})]^{-2}$, where R_{I} is the number of rules in the* FNNI, *W_{I}^{O} is the weight of the* FNNI, *and δ_{I} is the variance parameter of the membership function for the* FNNI.

Proof. Since

$$
\begin{aligned}
P_{\mathrm{I}}(k) &= \frac{\partial O_{\mathrm{I}}(k)}{\partial m_{\mathrm{I}}}\\
&= \sum_{i=1}^{R_{\mathrm{I}}} \frac{\partial O_{\mathrm{I}}}{\partial O_{I_i}^{(3)}} \frac{\partial O_{I_i}^{(3)}}{\partial m_{\mathrm{I}}} = \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\sum_{j} \frac{\partial O_{\mathrm{I}}^{(3)}}{\partial O_{I_j}^{(2)}} \frac{\partial O_{I_j}^{(2)}}{\partial m_{\mathrm{I}}}\right\}\\
&= \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\prod_{k\neq j} O_{I_k}^{(2)} \cdot \frac{\partial O_{I_j}^{(2)}}{\partial m_{\mathrm{I}}}\right\} < \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\max\left(\frac{\partial O_{\mathrm{I}}^{(2)}}{\partial m_{\mathrm{I}}}\right)\right\}\\
&= \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\max\left(\left(\frac{2}{\delta_{\mathrm{I}}}\right)\left(\frac{x_{\mathrm{I}}-m_{\mathrm{I}}}{\delta_{\mathrm{I}}}\right)\exp\left[-\left(\frac{x_{\mathrm{I}}-m_{\mathrm{I}}}{\delta_{\mathrm{I}}}\right)^2\right]\right)\right\},
\end{aligned}
\tag{12}
$$

drawing on Lemma 4.4, we obtain $|[(x_{\mathrm{I}}-m_{\mathrm{I}})/\delta_{\mathrm{I}}]\exp[-((x_{\mathrm{I}}-m_{\mathrm{I}})/\delta_{\mathrm{I}})^2]| < 1$.
Then

$$
P_{\mathrm{I}}(k) < \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\max\left(\frac{2}{\delta_{\mathrm{I}}}\right)\right\} = \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\left(\frac{2}{\delta_{\mathrm{I,min}}}\right)\right\}. \tag{13}
$$

Thus

$$
\|P_{\mathrm{I}}(k)\| < \sqrt{R_{\mathrm{I}}}\,|W_{\mathrm{I,max}}^{O}|\left(\frac{2}{\delta_{\mathrm{I,min}}}\right). \tag{14}
$$

Drawing on Theorem 4.1, we obtain

$$
0 < \eta_{\mathrm{I}}^{m} < \frac{2}{P_{\mathrm{I,max}}^{2}} = \frac{2}{R_{\mathrm{I}}}\left[\frac{1}{|W_{\mathrm{I,max}}^{O}|(2/\delta_{\mathrm{I,min}})}\right]^2.
$$

Since

$$
\begin{aligned}
Q_{\mathrm{I}}(k) &= \frac{\partial O_{\mathrm{I}}(k)}{\partial \delta_{\mathrm{I}}}\\
&= \sum_{i=1}^{R_{\mathrm{I}}} \frac{\partial O_{\mathrm{I}}(k)}{\partial O_{I_i}^{(3)}} \frac{\partial O_{I_i}^{(3)}}{\partial \delta_{\mathrm{I}}} = \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\sum_{j} \frac{\partial O_{\mathrm{I}}^{(3)}}{\partial O_{I_j}^{(2)}} \frac{\partial O_{I_j}^{(2)}}{\partial \delta_{\mathrm{I}}}\right\}\\
&= \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\prod_{k\neq j} O_{I_k}^{(2)} \cdot \frac{\partial O_{I_j}^{(2)}}{\partial \delta_{\mathrm{I}}}\right\} < \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\max\left(\frac{\partial O_{\mathrm{I}}^{(2)}}{\partial \delta_{\mathrm{I}}}\right)\right\}\\
&= \sum_{i=1}^{R_{\mathrm{I}}} W_{I_i}^{O}\left\{\max\left(\left(\frac{2}{\delta_{\mathrm{I}}}\right)\left(\frac{x_{\mathrm{I}}-m_{\mathrm{I}}}{\delta_{\mathrm{I}}}\right)^2\exp\left[-\left(\frac{x_{\mathrm{I}}-m_{\mathrm{I}}}{\delta_{\mathrm{I}}}\right)^2\right]\right)\right\}
\end{aligned}
\tag{15}
$$

by Lemma 4.5, we have

$$\left|\left(\frac{x_I - m_I}{\delta_I}\right)^2 \exp\left[-\left(\frac{x_I - m_I}{\delta_I}\right)^2\right]\right| < 1.$$

Then

$$Q_I(k) < \sum_{i=1}^{R_I} W_{I_i}^o \left\{\max\left(\frac{2}{\delta_I}\right)\right\} = \sum_{i=1}^{R_I} W_{I_i}^o \left\{\left(\frac{2}{\delta_{I,\min}}\right)\right\}. \tag{16}$$

Thus

$$\|Q_I(k)\| < \sqrt{R_I}|W_{I,\max}^o|\left(\frac{2}{\delta_{I,\min}}\right). \tag{17}$$

Therefore, from Theorem 4.1 we can find that $0 < \eta_I^\delta < 2/P_{I,\max}^2 = (2/R_I)[|W_{I,\max}^o|(2/\delta_{I,\min})]^{-2}$. This completes the proof of the theorem. □

Remark 4.7. From Remark 4.2 the optimal learning rates of the FNNI are $\eta_I^{o*} = 1/R_I$

$$\eta_I^{m*} = \eta_I^{\delta*} = \frac{1}{R_I}\left[\frac{1}{|W_{I,\max}^o|(2/\delta_{I,\min})}\right]^2.$$

4.2. Convergence of the FNNC

From Eqs. (5), (6) we have

$$\Delta W_C = -\eta_C \frac{\partial E_C(k)}{\partial W_C(k)} = -\eta_C e_C(k) \frac{\partial e_C(k)}{\partial W_C(k)}$$

$$= \eta_C e_C(k) y_u(k) \frac{\partial u(k)}{\partial W_C(k)} = \eta_C e_C(k) y_u(k) \frac{\partial O_C(k)}{\partial W_C(k)}, \tag{18}$$

where W_C and η_C represent an arbitrary weight and the corresponding learning rate in the FNNC, $O_C(k)$ is the output of the FNNC, and $y_u(k) = \partial y(k)/\partial u(k)$ is the plant sensitivity. Then we have a general convergence theorem from [9].

Theorem 4.8. (Ku and Lee [9]). *Let η_C be the learning rate for the weights or the parameters of the* FNNC *and let $P_{C,\max}$ be defined as $P_{C,\max} \equiv \max_k \|P_C(k)\|$, where $P_C(k) = \partial O_C(k)/\partial W_C$ and $\|\cdot\|$ is the usual Euclidean norm in $\Re^n$, and let $S = y_u(k)$. Then convergence is guaranteed if η_C is chosen as follows: $0 < \eta_C < 2/S^2 P_{C,\max}^2$.*

Remark 4.9. (*Ku and Lee* [9]). $\eta_C(2 - \eta_2) > 0$ or $\eta_2(2 - \eta_2)/S^2 P_{C,\max}^2 > 0$. This implies that any η_2, $0 < \eta_2 < 2$, guarantees convergence. However, the maximum learning rate which guarantees the optimal convergence corresponds to $\eta_2 = 1 \rightarrow \eta_C^* = 1/S^2 P_{C,\max}^2$.

Theorem 4.10. *Let η_C^o be the learning rate for the* FNNC *weights W_C^o. Then the learning rate is chosen as follows: $0 < \eta_C^o < (2/R_C)(1/S^2)$, where R_C is the number of rules in the* FNNI *and $S = y_u(k)$ is plant sensitivity.*

Proof. Let $P_C(k) = \partial O_C(k)/\partial W_C^O = Z^C(k)$, where $Z^C = [Z_1^C, Z_2^C, \dots, Z_{R_C}^C]^T$, in which Z_j^C is the output value of the third layer of the FNNC and R_C is the number of rules in the FNNC. Then we have
$Z_j^C \leqslant 1$ for all j, $\|P_C(k)\| \leqslant \sqrt{R_C}$ and $P_{C,\max}^2 = R_C$. From Theorem 4.8 we obtain $0 < \eta_C^o < (2/R_C)(1/S^2)$.

Theorem 4.11. *Let η_C^m and η_C^δ be the learning rates for the FNNC parameters m_C and δ_C, respectively. Then the learning rates are chosen as*

$$0 < \eta_C^m = \eta_C^\delta < \frac{2}{R_C}\frac{1}{S^2}\left[\frac{1}{|W_{C,\max}^O|(2/\delta_{C,\min})}\right]^2,$$

where R_C is the number of rules in the FNNC, *W_C^O the weights of the* FNNC, *δ_C the variance parameter of the membership function for the* FNNC, *and $S = y_u(k)$ is the plant sensitivity.*

Proof. Since

$$\begin{aligned}
P_C(k) &= \frac{\partial O_C(k)}{\partial m_C} \\
&= \sum_{i=1}^{R_C} \frac{\partial O_C(k)}{\partial O_{C_i}^{(3)}}\frac{\partial O_{C_i}^{(3)}}{\partial m_C} = \sum_{i=1}^{R_C} W_{C_i}^O\left\{\sum_j \frac{\partial O_C^{(3)}}{\partial O_{C_j}^{(2)}}\frac{\partial O_{C_j}^{(2)}}{\partial m_C}\right\} \\
&= \sum_{i=1}^{R_C} W_{C_i}^O\left\{\prod_{k\neq j} O_{C_k}^{(2)}\cdot\frac{\partial O_{C_j}^{(2)}}{\partial m_C}\right\} < \sum_{i=1}^{R_C} W_{C_i}^O\left\{\max\left(\frac{\partial O_C^{(2)}}{\partial m_C}\right)\right\} \\
&= \sum_{i=1}^{R_C} W_{C_i}^O\left\{\max\left(\left(\frac{2}{\delta_C}\right)\left(\frac{x_C - m_C}{\delta_C}\right)\exp\left[-\left(\frac{x_C-m_C}{\delta_C}\right)^2\right]\right)\right\}
\end{aligned} \tag{19}$$

by Lemma 4.4, we have $|((x_C - m_C)/\delta_C)\exp[-((x_C - m_C)/\delta_C)^2]| < 1$.
Then

$$P_C(k) < \sum_{i=1}^{R_C} W_{C_i}^O\left\{\max\left(\frac{2}{\delta_C}\right)\right\} = \sum_{i=1}^{R_C} W_{C_i}^O\left\{\left(\frac{2}{\delta_{C,\min}}\right)\right\}. \tag{20}$$

Thus

$$\|P_C(k)\| < \sqrt{R_C}|W_{C,\max}^O|(2/\delta_{C,\min}). \tag{21}$$

Therefore, from Theorem 4.8 we can find that

$$0 < \eta_C^m < \frac{2}{P_{C,\max}^2}\frac{1}{S^2} = \frac{2}{R_C}\frac{1}{S^2}\left[\frac{1}{|W_{C,\max}^O|(2/\delta_{C,\min})}\right]^2.$$

Moreover, since

$$\begin{aligned}
Q_C(k) &= \frac{\partial O_C(k)}{\partial \delta_C} \\
&= \sum_{i=1}^{R_C} \frac{\partial O_C(k)}{\partial O_{C_i}^{(3)}}\frac{\partial O_{C_i}^{(3)}}{\partial \delta_C} = \sum_{i=1}^{R_C} W_{C_i}^O\left\{\sum_j \frac{\partial O_C^{(3)}}{\partial O_{C_j}^{(2)}}\frac{\partial O_{C_j}^{(2)}}{\partial \delta_C}\right\}
\end{aligned}$$

$$= \sum_{i=1}^{R_C} W_{C_i}^{O} \left\{ \prod_{k \neq j} O_{C_k}^{(2)} \cdot \frac{\partial O_{C_j}^{(2)}}{\partial \delta_C} \right\} < \sum_{i=1}^{R_C} W_{C_i}^{O} \left\{ \max\left(\frac{\partial O_C^{(2)}}{\partial \delta_C} \right) \right\}$$

$$= \sum_{i=1}^{R_C} W_{C_i}^{O} \left\{ \max\left(\left(\frac{2}{\delta_C}\right) \left(\frac{x_C - m_C}{\delta_C}\right)^2 \exp\left[-\left(\frac{x_C - m_C}{\delta_C}\right)^2 \right] \right) \right\} \tag{22}$$

by Lemma 4.5 we have $|((x_C - m_C)/\delta_C)^2 \exp[-((x_C - m_C)/\delta_C)^2]| < 1$.

Then

$$Q_C(k) < \sum_{i=1}^{R_C} W_{C_i}^{O} \left\{ \max\left(\frac{2}{\delta_C}\right) \right\} = \sum_{i=1}^{R_C} W_{C_i}^{O} \left\{ \left(\frac{2}{\delta_{C,\min}}\right) \right\}. \tag{23}$$

Thus

$$\| Q_C(k) \| < \sqrt{R_C} |W_{C,\max}^{O}| \left(\frac{2}{\delta_{C,\min}} \right). \tag{24}$$

Therefore, from Theorem 4.8 we obtain

$$0 < \eta_C^{\delta} < \frac{2}{P_{C,\max}^2} = \frac{2}{R_C} \left[\frac{1}{|W_{C,\max}^{O}| (2/\delta_{C,\min})} \right]^2.$$

This completes the proof of the theorem. □

Remark 4.12. From Remark 4.9 the optimal learning rates of the FNNC are

$$\eta_C^{O*} = \frac{1}{R_C} \frac{1}{S^2}, \qquad \eta_C^{m*} = \eta_C^{\delta*} = \frac{1}{R_C} \frac{1}{S^2} \left[\frac{1}{|W_{C,\max}^{O}| (2/\delta_{C,\min})} \right]^2,$$

where $S = y_u(k) = \partial y_I(k)/\partial u(k)$ is the plant sensitivity.

5. Simulation results

In this section we test the model reference control structure using two different examples. The number of inputs for the FNNC is denoted by n_C and that of the FNNI by n_I. R_C and R_I denote the number of rules in the FNNC and FNNI. P_C and P_I are the inputs to the FNNC and FNNI.

Example 1 (*Interpolation ability, Ku and Lee* [9]). This example demonstrates the interpolation ability of the FNN control system by applying it to a flight control application. During the training process, only a few trim points are trained. After a few training cycles, an untrained trim point is applied and tested in the FNN control system.

In this case the plant can be described by the Laplace transfer function

$$P(s) = \frac{1.0}{s^2 + 2.0s + 1.0}.$$

The reference model is described by

$$H(s) = \frac{4.0}{s^2 + 2.82s + 4.0}.$$

Here, three sets of initial conditions for $(y(0), \dot{y}(0))$, $(0.0, 0.0)$, $(0.1, 0.3)$, $(0.5, 0.75)$, are selected as training sets.

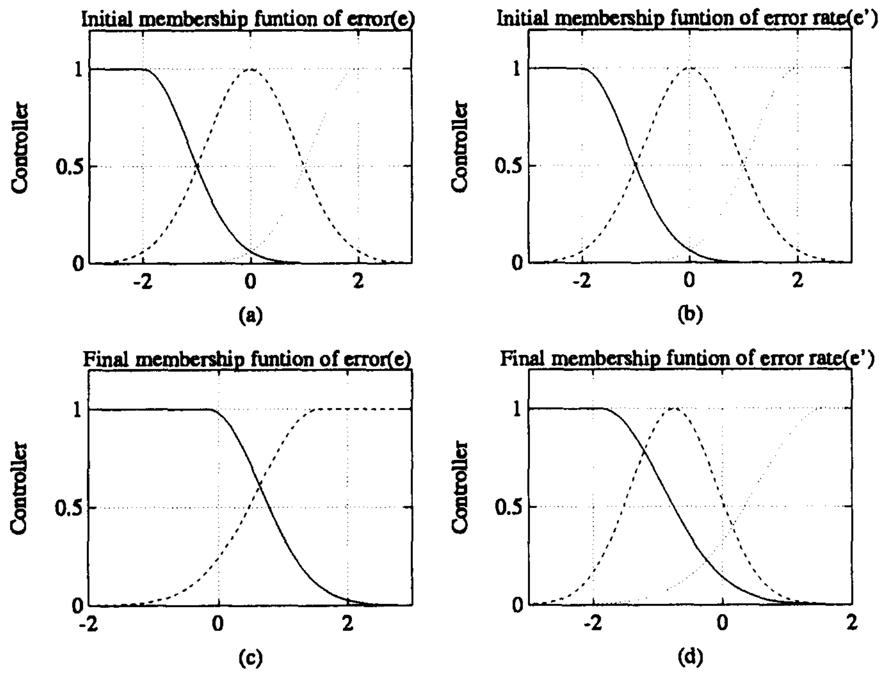

Fig. 3. Initial membership functions of controller: (a) error, (b) error rate. Final membership functions of controller: (c) error, (d) error rate.

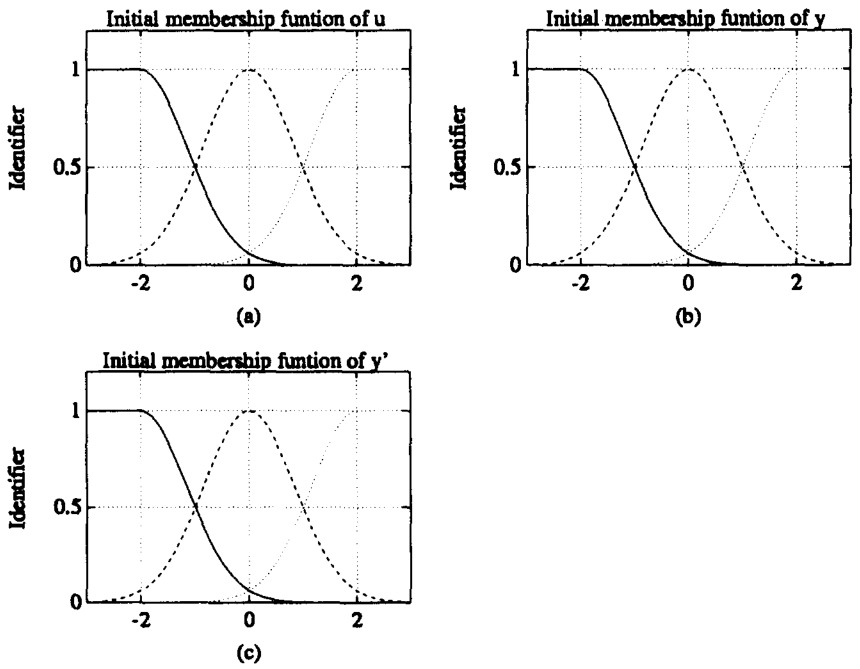

Fig. 4. Initial membership functions of identifier: (a) u, (b) y, (c) y'.

Assume that for the FNNC, $P_C = \{e(t), \dot{e}(t)\}$, each input variable has three fuzzy partition sets, so $R_C = 3 \times 3 = 9$ rules. For FNNI, $P_I = \{u(t), y(t), \dot{y}(t)\}$, each input variable has three fuzzy partition sets, so $R_I = 3 \times 3 \times 3 = 27$ rules. See Figs. 3(a) and (b), 4(a)–(c). Each cycle takes 10 s. After 18 cycles, the plant can be controlled very effectively. A test set (0.8, 1.0) is applied to the system, with a step size of 0.02 s.

In each cycle, if the value of a particular consequence link rule is smaller than $1/R_C = \frac{1}{9}$ for the FNNC or $1/R_I = \frac{1}{27}$ for the FNNI, then we eliminate that rule. In the final simulation result, we find that the FNNC has four rules and the FNNI has eight rules. See Figs. 3(c) and (d), 5(a)–(c), and Table 1. The final simulation result is shown in Fig. 6.

The results show that the FNN control system has the ability to interpolate control response if an untrained set is closed to all trained sets.

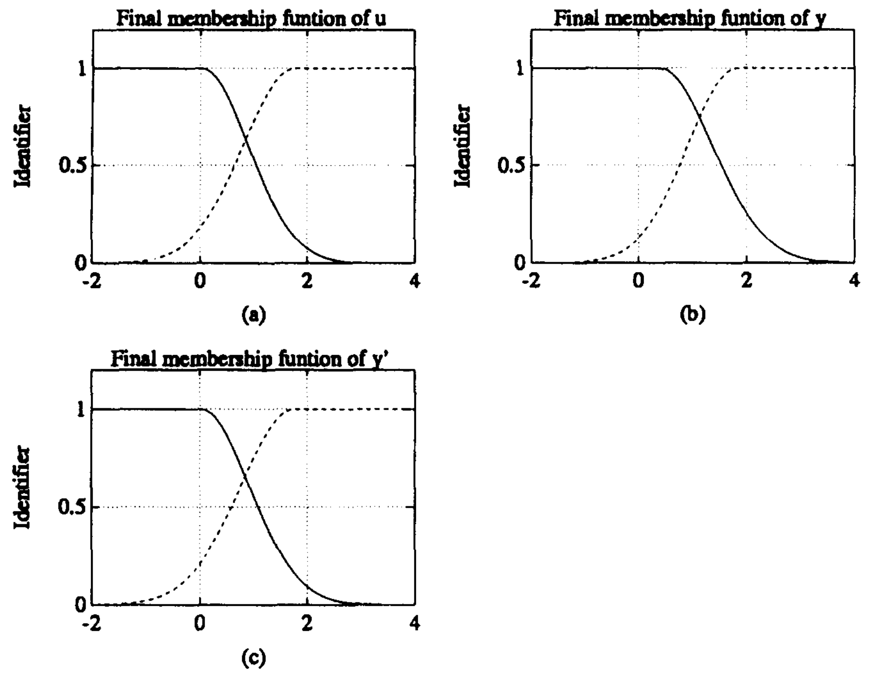

Fig. 5. Final membership functions of identifier: (a) u, (b) y, (c) y'.

Table 1
Learned rule weight matrix for the FNNC

	$\dot{e}$		
e	NM	ZE	PM
NM	0.000	0.000	0.000
ZE	− 1.001	− 0.826	2.352
PM	0.000	2.488	0.000

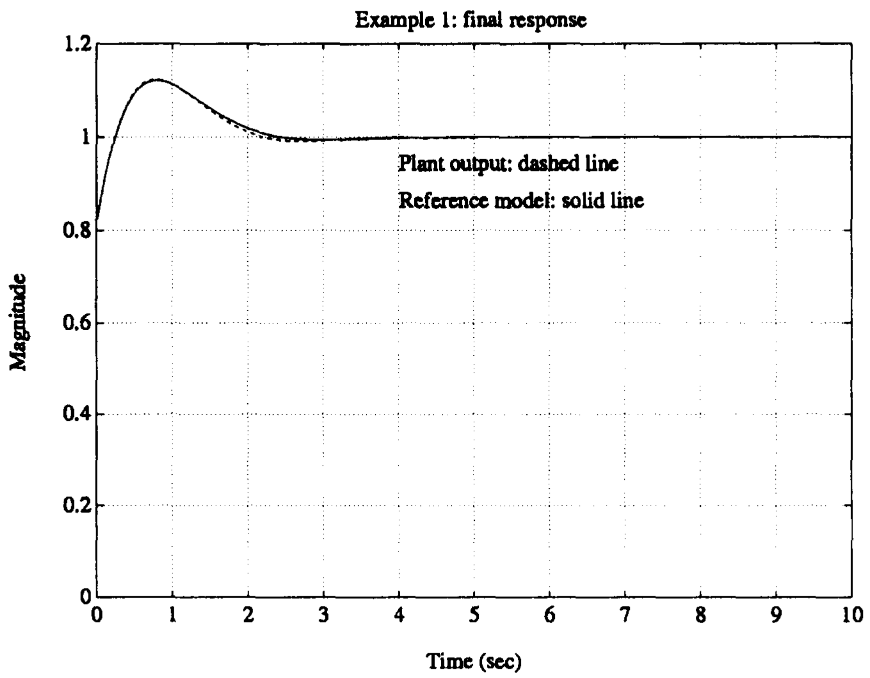

Fig. 6. Final system response for Example 1. Plant output is indicated by the dashed line, reference model by the solid line.

Example 2 (*A BIBO nonlinear plant* [13,9]). In this case the plant is described by the difference equation

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k).$$

The reference model is described by the difference equation

$$y_r(k+1) = 0.6\, y_r(k) + r(k),$$

where $r(k) = \sin(2\pi k/10) + \sin(2\pi k/25)$.

Assume that for the FNNC, $P_C = \{e(k), \Delta e(k)\}$, each input variable has five fuzzy partition sets, so $R_C = 5 \times 5 = 25$ rules. For FNNI, $P_I = \{u(k), y(k)\}$, each input variable has five fuzzy partition sets, so $R_I = 5 \times 5 = 25$ rules. See Figs. 7(a) and (b), 8(a) and (b).

Each cycle takes 100 s. Adaptive learning rates are used, starting from the initial rates of $\eta_C^O = 0.2$ and $\eta_I^O = 0.04$. The learning rates adapt to reduce the tracking error. The learning rates for the FNNC and FNNI are η_C^O, η_C^m, η_I^O and η_I^m, which are shown in Fig. 9. After 30 cycles this problem can be controlled very effectively. See Figs. 10 and 11. In each cycle, if the value of a particular consequence link rule is smaller than $1/R_C = 1/25$ for the FNNC or $1/R_I = 1/25$ for the FNNI, then the rule is eliminated. In the final simulation result, we find that the FNNC has 25 rules and the FNNI has 25 rules. See Figs. 7(c) and (d), 8(c) and (d), and Tables 2, 3. The final result is shown in Fig. 11.

To examine the adaptive ability of the model reference control structure, we repeat the simulation with the same conditions as those shown above, except that reference $r(k)$ is modified as follows:

(a) After 30 cycles, the reference input is changed to $r(k) = \sin(2\pi k/25)$, see Fig. 12.
(b) After 30 cycles, the reference input is changed to an impulse signal, see Fig. 13.
(c) After 30 cycles, we add a disturbance of 2.0 to the system at 20 s and another at 40 s, see Fig. 14.

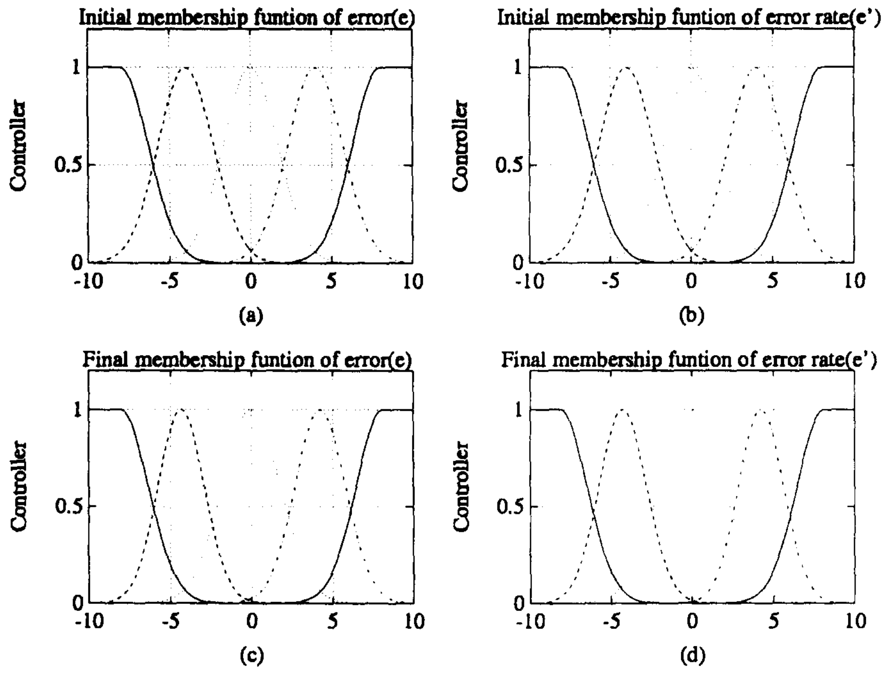

Fig. 7. Initial membership functions of controller: (a) error, (b) error rate. Final membership functions of controller: (c) error, (d) error rate.

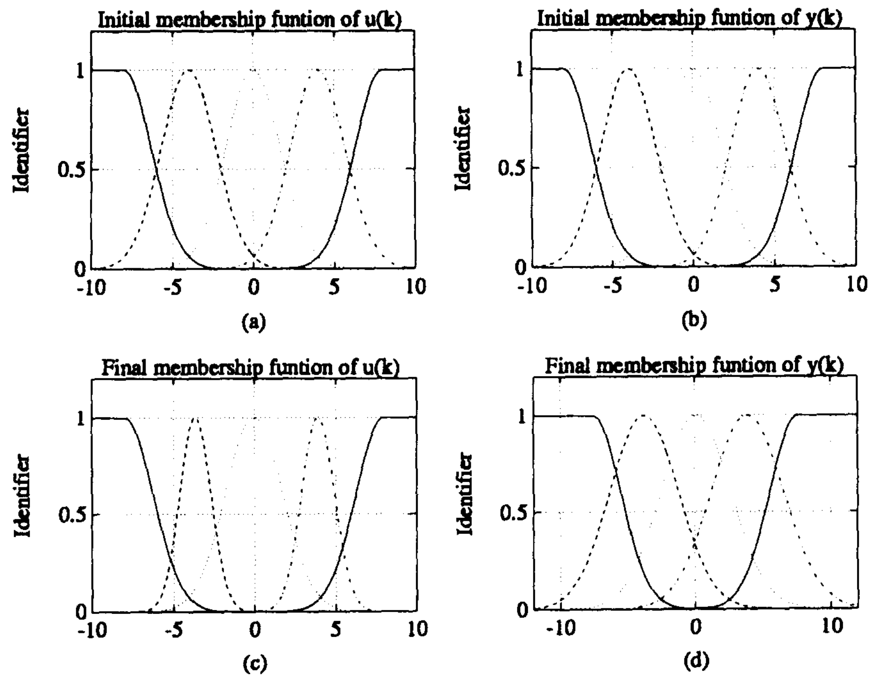

Fig. 8. Initial membership functions of identifier: (a) $u(k)$, (b) $y(k)$. Final membership functions of identifier: (c) $u(k)$, (d) $y(k)$.

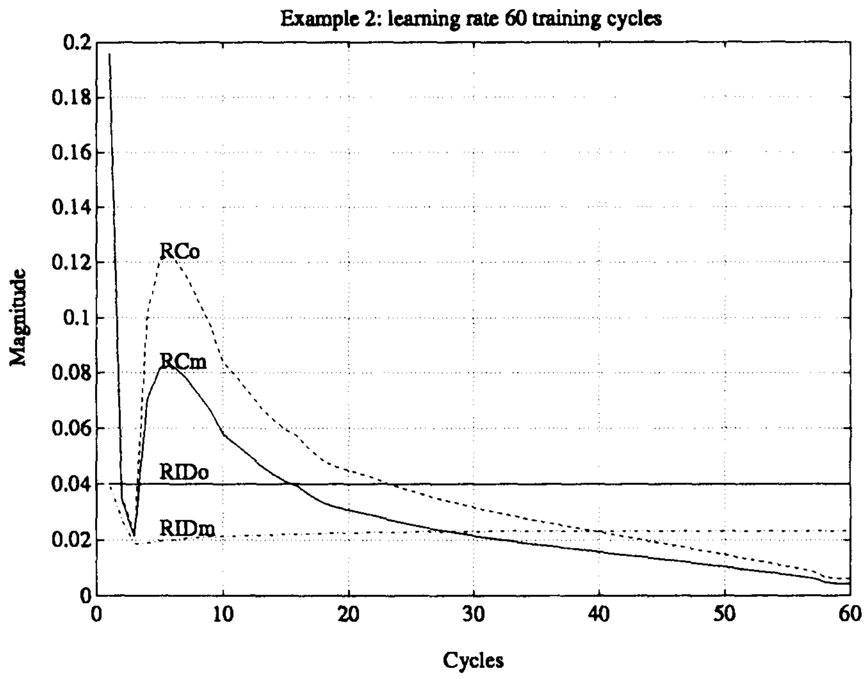

Fig. 9. Adaptive learning rates of the FNNC and FNNI during 60 training cycles.

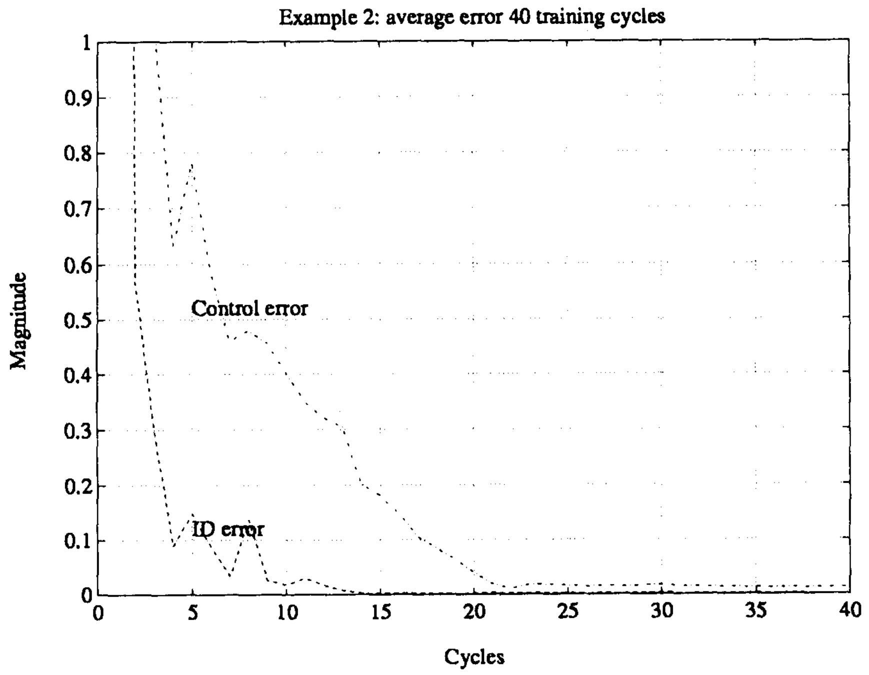

Fig. 10. Average error of the FNNC and FNNI during 40 training cycles.

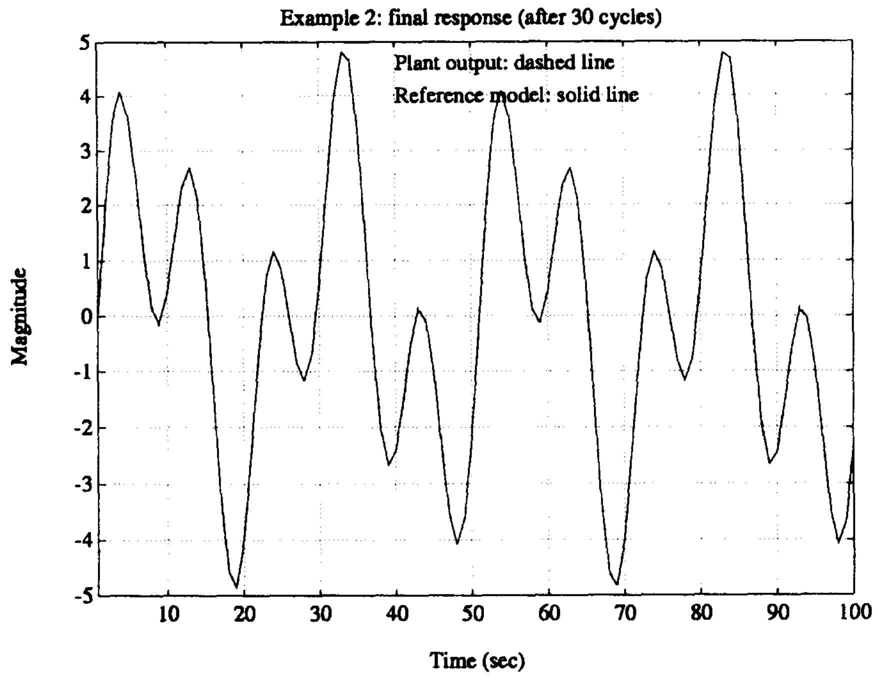

Fig. 11. Final system response for Example 2. Plant output is indicated by the dashed line, reference model by the solid line.

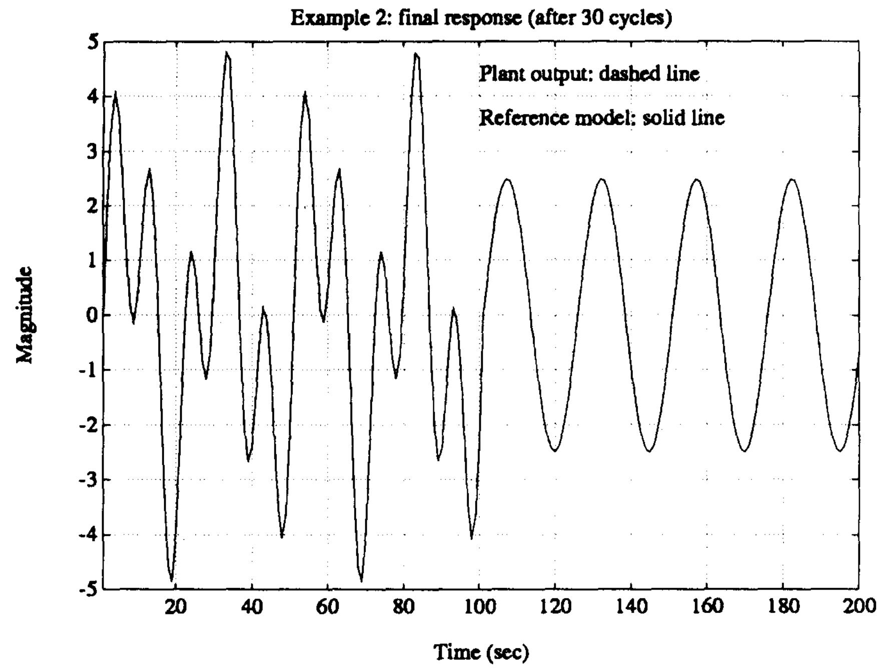

Fig. 12. Final system response for Example 2. Plant output is indicated by the dashed line, reference model by the solid line, tested adaptation for sinusoid signal.

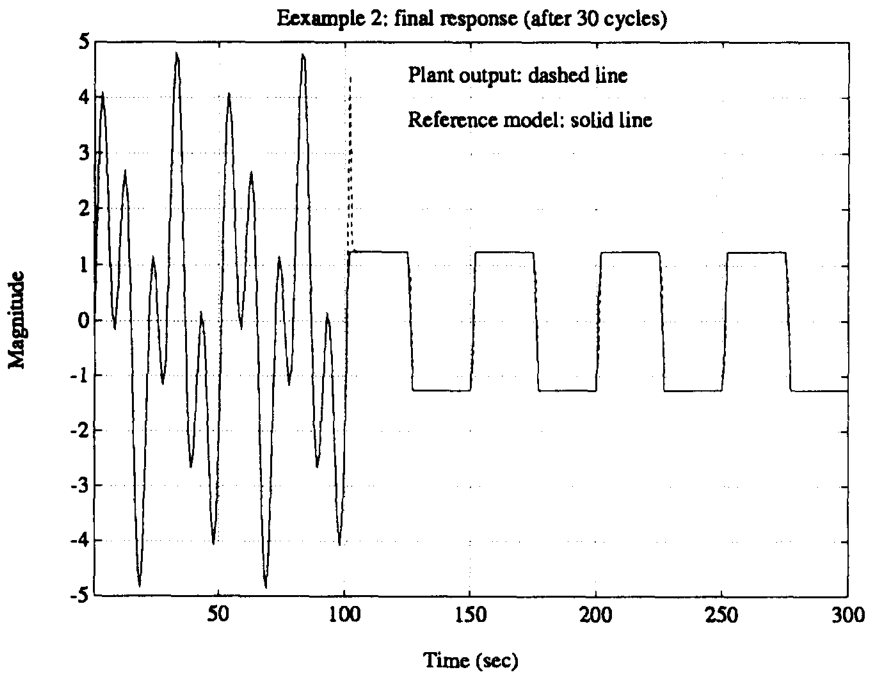

Fig. 13. Final system response for Example 2. Plant output is indicated by the dashed line, reference model by the solid line, tested adaptation for impulse signal.

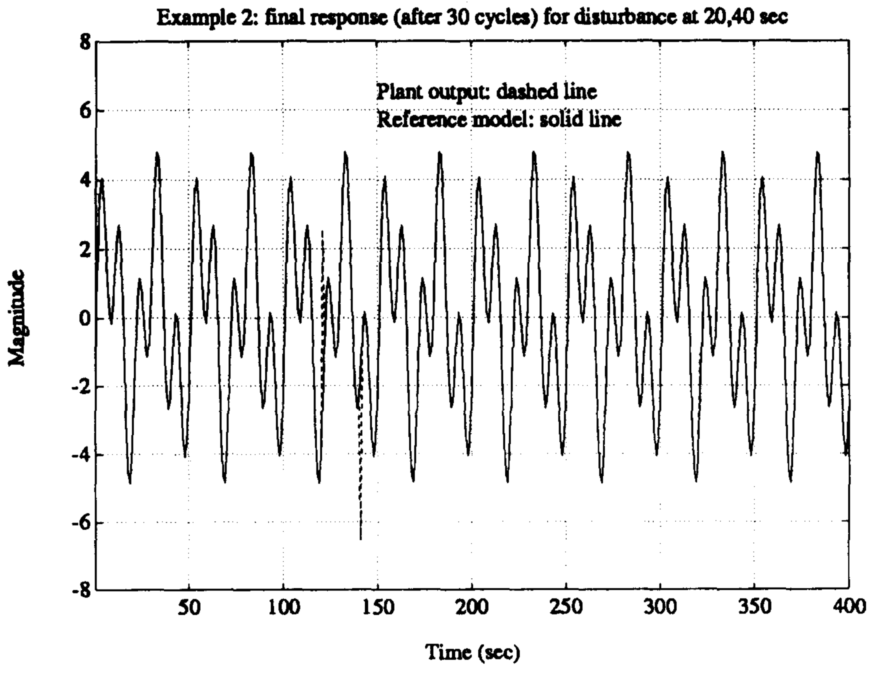

Fig. 14. Final system response for Example 2. Plant output is indicated by the dashed line, reference model by the solid line, tested system robustness for added disturbances.

Table 2
Learned rule weight matrix for the FNNC

	$\Delta e(k)$				
$e(k)$	NB	NS	ZE	PS	PB
NB	− 3.266	2.135	− 1.944	2.664	1.445
NS	0.023	− 1.625	− 1.925	− 0.813	1.006
ZE	− 1.398	− 0.971	0.185	− 1.545	− 0.092
PS	− 0.534	− 0.521	1.011	− 1.308	0.395
PB	− 1.956	− 1.845	− 1.126	− 1.184	− 1.658

Table 3
Learned rule weight matrix for the FNNI

	$u(k)$				
$y(k)$	NB	NS	ZE	PS	PB
NB	− 1.495	− 0.532	− 0.546	− 0.166	− 0.247
NS	− 3.992	− 3.188	− 2.661	− 1.731	− 1.862
ZE	0.479	− 1.363	− 0.214	1.768	0.272
PS	2.131	1.077	2.437	3.615	4.080
PB	0.219	0.251	0.842	0.663	0.979

These figures show that the control structure can track the new reference model quickly. Also, the on-line adaptive ability and robustness of the model reference control structure using the FNN are acceptable.

6. Conclusion

A model reference control structure using a fuzzy neural network has been successfully applied to some difficult learning control problems. The ability of the FNNC and FNNI to learn control rules from experience and to adapt to system changes and rule degradation have been confirmed by simulation results. An approach to finding the bounds on learning rates based on a Lyapunov function was developed. The use of adaptive learning rates guarantees convergence, and the optimal learning rates were found. The FNN-based control system was tested for its on-line adaptive ability, robustness, and interpolation ability. Combining fuzzy logic and neural network computing appears to be a feasible way of dealing with real-time applications.

References

[1] C.W. Anderson, Learning to control an inverted pendulum using neural networks, *IEEE Control Syst. Mag.* **9**(3) (1989) 31–37.

[2] A.G. Barto, R.S. Sutton and C.W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man Cybern.* **13** (1983) 834–846.

[3] Y.C. Chien, Adaptive fuzzy logic controller using neural networks, Master thesis, Chiao-Tung University (1992).

[4] Y.C. Chien, Y.C. Chen and C.C. Teng, Model reference adaptive fuzzy logic controller design using fuzzy neural network, *Proc. First Asian Fuzzy Systems Symp.* (Singapore, November 23–26, 1993) 334–340.

[5] A. Guez and J. Selinsky, A trainable neuromorphic controller, *J. Robotics Syst.* **5**(4) (1988) 363–388.

[6] S. Horikawa, T. Furuhashi, S. Okuma and Y. Uchikawa, A fuzzy controller using a neural network and its capability to learn expert's control rules, *Proc. Int. Conf. Fuzzy logic and Neural Networks* (Iizuka, Japan, July, 1990) 103–106.

[7] K.J. Hunt, D. Sbarbaro, R. Zbikowski and P.J. Gawthrop, Neural networks for control systems – a survey, *Automatica* **28**(6) (1992) 1083–1112.

[8] T. Kohonon, The self-organizing map, *Proc. IEEE* **78**(9) (1990) 1461–1480.

[9] C.C. Ku and K.Y. Lee, Diagonal recurrent neural networks for dynamic systems control, *IEEE Trans. Neural Networks* (1995), to appear.

[10] C.C. Lee, Intelligent control based on fuzzy logic and neural net theory, *Proc. Int. Conf. Fuzzy logic and Neural Networks* (Iizuka, Japan, July, 1990) 759–764.

[11] Y. Li, A.K.C. Wang and F. Yang, Optimal neural network control, *IFAC-INCOM* (1992) 41–46.

[12] C.T. Lin and C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Computers* **40**(12) (1991) 1320–1336.

[13] K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks* **1**(1) (1990) 4–27.

[14] D.H. Nguyen and B. Widrow, Neural networks for self-learning control systems, *Internat. J. Control* **54**(6) (1991) 1439–1451.

[15] M. Sugeno, An introductory survey of fuzzy control, *Information and Sciences*, **36** (1985) 59–83.

[16] M. Sugeno and T. Yasukawa, A Fuzzy-logical-based approach to qualitative modeling, *IEEE Trans. Fuzzy Systems* **1**(1) (1993) 7–31.

[17] G.J. Wang and D.K. Miu, Unsupervised adaptation neural-network control, *IEEE Int. Joint Conf. Neural Networks* **3** (1990) 421–428.

[18] H. Ying, W. Siler and J.J. Buckley, Fuzzy control theory: a nonlinear case, *Automatica* **26**(3) (1990) 513–520.