A Functional-Link-Based Neurofuzzy Network for Nonlinear System Control

Cheng-Hung Chen, *Student Member, IEEE*, Cheng-Jian Lin, *Member, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*—This study presents a functional-link-based neurofuzzy network (FLNFN) structure for nonlinear system control. The proposed FLNFN model uses a functional link neural network (FLNN) to the consequent part of the fuzzy rules. This study uses orthogonal polynomials and linearly independent functions in a functional expansion of the FLNN. Thus, the consequent part of the proposed FLNFN model is a nonlinear combination of input variables. An online learning algorithm, which consists of structure learning and parameter learning, is also presented. The structure learning depends on the entropy measure to determine the number of fuzzy rules. The parameter learning, based on the gradient descent method, can adjust the shape of the membership function and the corresponding weights of the FLNN. Furthermore, results for the universal approximator and a convergence analysis of the FLNFN model are proven. Finally, the FLNFN model is applied in various simulations. Results of this study demonstrate the effectiveness of the proposed FLNFN model.**

***Index Terms*—Entropy, functional link neural networks (FLNNs), neurofuzzy networks (NFNs), nonlinear system control, online learning.**

I. Introduction

NONLINEAR system control is an important tool that is adopted to improve control performance and achieve robust fault-tolerant behavior. Among nonlinear control techniques, those based on artificial neural networks and fuzzy systems have become popular topics of research in recent years [1], [2] because classical control theory usually requires that a mathematical model be used in designing a controller. However, the inaccuracy of the mathematical modeling of plants usually degrades the performance of the controller, especially for nonlinear and complex control problems [3]. On the contrary, both the fuzzy system controller and the artificial neural network controller provide key advantages over traditional adaptive control systems. Although traditional neural networks can learn from data and feedback, the meaning associated with each neuron and each weight in the network is not easily interpreted. Alternatively, the fuzzy systems are easily appreciated, because they use linguistic terms and the structure of IF–THEN rules. However, the learning capacity of fuzzy systems is less than that of neural networks. According to the literature review mentioned before, neurofuzzy networks (NFNs) [4]–[13] provide the advantages of both neural networks and fuzzy systems, unlike pure neural networks or fuzzy systems alone. NFNs bring the low-level learning and computational power of neural networks into fuzzy systems and give the high-level human-like thinking and reasoning of fuzzy systems to neural networks.

Two typical types of NFNs are the Mamdani-type and the Takagi–Sugeno–Kang (TSK)-type. For Mamdani-type NFNs [7]–[9], the minimum fuzzy implication is adopted in fuzzy reasoning. For TSK-type NFNs [10]–[13], the consequence part of each rule is a linear combination of input variables. Many researchers [12], [13] have shown that TSK-type NFNs offer better network size and learning accuracy than Mamdani-type NFNs. In the typical TSK-type NFN, which is a linear polynomial of input variables, the model output is approximated locally by the rule hyperplanes. Nevertheless, the traditional TSK-type NFN does not take full advantage of the mapping capabilities that may be offered by the consequent part. Introducing a nonlinear function, especially a neural structure, to the consequent part of the fuzzy rules has yielded the neural networks designed on approximate reasoning architecture (NARA) [14] and the coactive neurofuzzy inference system (CANFIS) [15] models. These models [14], [15] apply multilayer neural networks to the consequent part of the fuzzy rules. Although the interpretability of the model is reduced, the representational capability of the model is markedly improved. However, the multilayer neural network has such disadvantages as slower convergence and greater computational complexity. Therefore, this study uses the functional link neural network (FLNN) [16], [17] to the consequent part of the fuzzy rules, called a functional-link-based NFN (FLNFN). The consequent part of the proposed FLNFN model is a nonlinear combination of input variables, which differs from the other existing models [8], [12], [13]. The FLNN is a single-layer neural structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries with nonlinear functional expansion. The FLNN [18] was conveniently used for function approximation and pattern classification with faster convergence rate and less computational loading than a multilayer neural network. Moreover, using the functional expansion can effectively increase the dimensionality of the input vector, so the hyperplanes generated by the FLNN will provide a good discrimination capability in input data space.

Manuscript received October 24, 2006; revised July 4, 2007; accepted November 20, 2007. First published April 30, 2008; current version published October 8, 2008. This work was supported in part by the Ministry of Economic Affairs, Taiwan, R.O.C., under Grant 96-EC-17-A-02-S1-032, and in part by the National Science Council, Taiwan, R.O.C., under Grant NSC 95-2221-E-009-180.

C.-H. Chen is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: chchen.ece93g@nctu.edu.tw).

C.-J. Lin is with the Department of Computer Science and Engineering, National Chin-Yi University of Technology, Taichung County, Taiwan 411, R.O.C. (e-mail: cjlin@ncut.edu.tw).

C.-T. Lin is with the Department of Computer Science, and the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. He is also with the Brain Research Center, University System of Taiwan, Hsinchu 300, Taiwan, R.O.C. (e-mail: ctlin@mail.nctu.edu.tw).

Digital Object Identifier 10.1109/TFUZZ.2008.924334

This study presents an FLNFN structure for nonlinear system control. The FLNFN model, which combines an NFN with an FLNN, is designed to improve the accuracy of functional approximation. Each fuzzy rule that corresponds to an FLNN consists of a functional expansion of input variables. The orthogonal polynomials and linearly independent functions are adopted as FLNN bases. An online learning algorithm, consisting of structure learning and parameter learning, is proposed to construct the FLNFN model automatically. The structure learning algorithm determines whether or not to add a new node that satisfies the fuzzy partition of input variables. Initially, the FLNFN model has no rules. The rules are automatically generated from training data by entropy measure. The parameter learning algorithm is based on backpropagation to tune the free parameters in the FLNFN model simultaneously to minimize an output error function. The advantages of the proposed FLNFN model are summarized as follows.

1) The consequent of the fuzzy rules of the proposed model is a nonlinear combination of input variables. This study uses the FLNN to the consequent part of the fuzzy rules. The local properties of the consequent part in the FLNFN model enable a nonlinear combination of input variables to be approximated more effectively.
2) The online learning algorithm can automatically construct the FLNFN model. No rules or memberships exist initially. They are created automatically as learning proceeds, as online incoming training data are received and as structure and parameter learning are performed.
3) The FLNFN model is proven to be a universal approximator by Stone–Weierstrass theorem and its convergence properties are proven by the Lyapunov theorem in the Appendix.
4) As demonstrated in Section IV, the proposed FLNFN model is a more adaptive and effective controller than the other models.

This paper is organized as follows. Section II describes the structure of the suggested model. Section III presents the online structure and parameter learning algorithms. Next, Section IV presents the results of simulations of various problems. Finally, Section V draws conclusions and discusses future works.

II. Structure of Functional-Link-Based Neurofuzzy Network

This section describes the structure of FLNNs and the structure of the FLNFN model. In FLNNs, the input data usually incorporate high-order effects, and thus, artificially increase the dimensions of the input space using a functional expansion. Accordingly, the input representation is enhanced and linear separability is achieved in the extended space. The FLNFN model adopted the FLNN, generating complex nonlinear combinations of input variables to the consequent part of the fuzzy rules. The rest of this section details these structures.

A. Functional Link Neural Networks

The FLNN is a single-layer network in which the need for hidden layers is removed. While the input variables generated by the linear links of neural networks are linearly weighted, the functional link acts on an element of input variables by generating a set of linearly independent functions (i.e., the use of suitable orthogonal polynomials for a functional expansion) and then evaluating these functions with the variables as the arguments. Therefore, the FLNN structure considers trigonometric functions. For example, for a two-dimensional input $\mathbf{X} = [x_1, x_2]^T$, the enhanced input is obtained using trigonometric functions in $\mathbf{\Phi} = [1, x_1, \sin(\pi\, x_1), \cos(\pi\, x_1), \ldots, x_2, \sin(\pi\, x_2), \cos(\pi\, x_2), \ldots]^T$. Thus, the input variables can be separated in the enhanced space [16]. In the FLNN structure with reference to Fig. 1, a set of basis functions $\mathbf{\Phi}$ and a fixed number of weight parameters $\mathbf{W}$ represent $f_{\mathbf{W}}(x)$. The theory behind the FLNN for multidimensional function approximation has been discussed elsewhere [19] and is analyzed later.

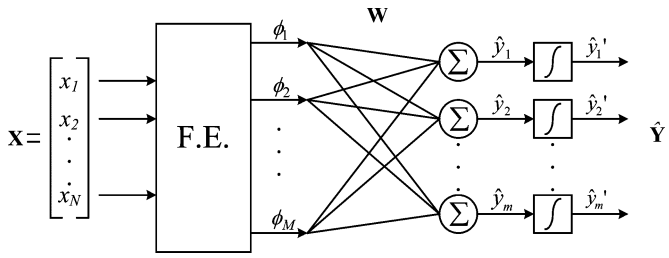


Fig. 1. Structure of an FLNN.

Consider a set of basis functions $\mathbf{B} = \{\phi_k \in \mathbf{\Phi}(A)\}_{k\in\mathbf{K}}$, $\mathbf{K} = \{1, 2, \ldots\}$, with the following properties: 1) $\phi_1 = 1$; 2) the subset $\mathrm{B}_j = \{\phi_k \in \mathbf{B}\}_{k=1}^{M}$ is a linearly independent set, meaning that if $\sum_{k=1}^{M} w_k \phi_k = 0$, then $w_k = 0$ for all $k = 1, 2, \ldots, M$; and 3) $\sup_j [\sum_{k=1}^{j} \|\phi_k\|_A^2]^{1/2} < \infty$.

Let $\mathbf{B} = \{\phi_k\}_{k=1}^{M}$ be a set of basis functions to be considered, as shown in Fig. 1. The FLNN comprises M basis functions $\{\phi_1, \phi_2, \ldots, \phi_M\} \in \mathbf{B}_M$. The linear sum of the jth node is given by

$$\hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k(\mathbf{X}) \tag{1}$$

where $\mathbf{X} \in \mathrm{A} \subset \Re^N, \mathbf{X} = [x_1, x_2, \ldots, x_N]^T$ is the input vector and $\mathbf{W}_j = [w_{j1}, w_{j2}, \ldots, w_{jM}]^T$ is the weight vector associated with the jth output of the FLNN. $\hat{\mathbf{y}}_j$ denotes the local output of the FLNN structure and the consequent part of the jth fuzzy rule in the FLNFN model. Thus, (1) can be expressed in matrix form as $\hat{\mathbf{y}}_j = \mathbf{W}_j \mathbf{\Phi}$, where $\mathbf{\Phi} = [\phi_1(x), \phi_2(x), \ldots, \phi_N(x)]^T$ is the basis function vector, which is the output of the functional expansion block. The m-dimensional linear output may be given by $\hat{\mathbf{y}} = \mathbf{W}\mathbf{\Phi}$, where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]^T$, m denotes the number of functional link bases, which equals the number of fuzzy rules in the FLNFN model, and $\mathbf{W}$ is an $(m \times M)$-dimensional weight matrix of the FLNN given by $\mathbf{W} = [\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_M}]^T$. The jth output of the FLNN is given by $\hat{y}_j{}' = \rho(\hat{y}_j)$, where the nonlinear function $\rho(\cdot) = \tanh(\cdot)$. Thus, the m-dimensional output vector is given by

$$\hat{\mathbf{Y}} = \rho(\hat{y}) = f_{\mathbf{W}}(x) \tag{2}$$

where $\hat{\mathbf{Y}}$ denotes the output of the FLNN. In the FLNFN model, the corresponding weights of functional link bases do not exist in

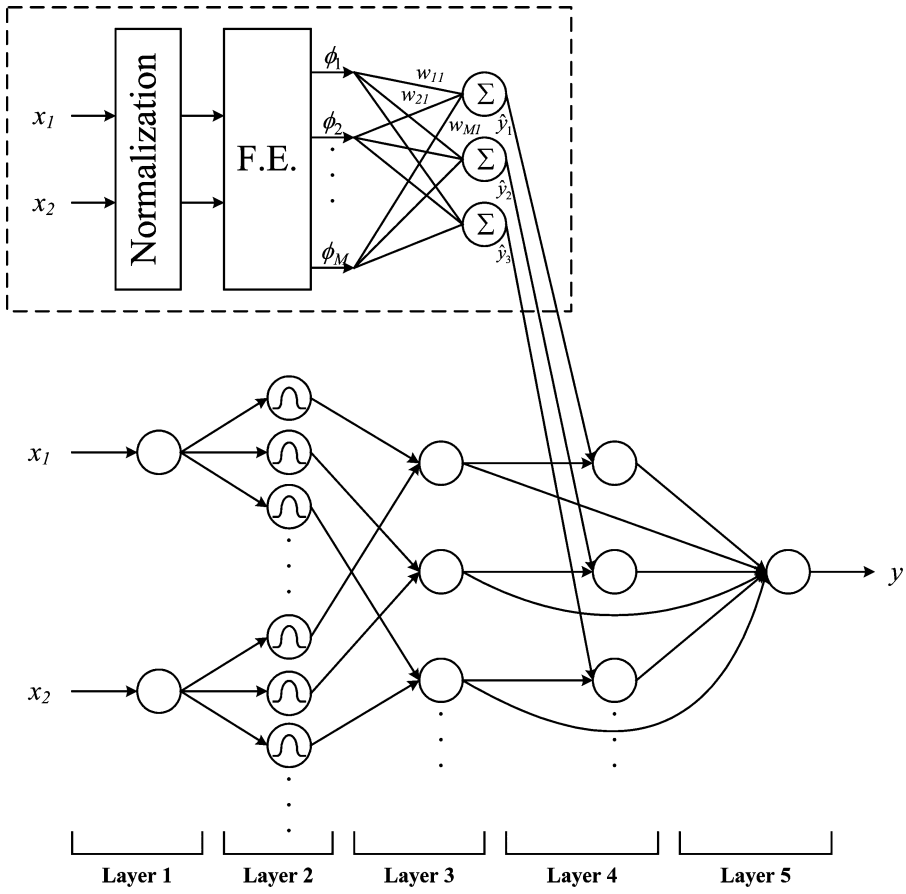


Fig. 2. Structure of proposed FLNFN model.

the initial state, and the amount of the corresponding weights of functional link bases generated by the online learning algorithm is consistent with the number of fuzzy rules. Section III details the online learning algorithm.

B. Structure of FLNFN Model

This subsection describes the FLNFN model, which uses a nonlinear combination of input variables (FLNN). Each fuzzy rule corresponds to a sub-FLNN, comprising a functional link. Fig. 2 presents the structure of the proposed FLNFN model.

The FLNFN model realizes a fuzzy IF–THEN rule in the following form.

Rule j:

IF x_1 is A_{1j} and x_2 is $A_{2j}\cdots$ and x_i is $A_{ij}\cdots$ and x_N is A_{Nj}

$$\text{THEN } \hat{y}_j = \sum_{k=1}^{M} w_{kj}\phi_k = w_{1j}\phi_1 + w_{2j}\phi_2 + \cdots + w_{Mj}\phi_M \tag{3}$$

where x_i and $\hat{y}_j$ are the input and local output variables, respectively; A_{ij} is the linguistic term of the precondition part with Gaussian membership function, N is the number of input variables, w_{kj} is the link weight of the local output, ϕ_k is the basis trigonometric function of input variables, M is the number of basis function, and rule j is the jth fuzzy rule.

The operation functions of the nodes in each layer of the FLNFN model are now described. In the following description, $u^{(l)}$ denotes the output of a node in the lth layer.

No computation is performed in layer 1. Each node in this layer only transmits input values to the next layer directly

$$u_i^{(1)} = x_i. \tag{4}$$

Each fuzzy set A_{ij} is described here by a Gaussian membership function. Therefore, the calculated membership value in layer 2 is

$$u_{ij}^{(2)} = \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right) \tag{5}$$

where m_{ij} and σ_{ij} are the mean and variance of the Gaussian membership function, respectively, of the jth term of the ith input variable x_i.

Nodes in layer 3 receive one-dimensional membership degrees of the associated rule from the nodes of a set in layer 2. Here, the product operator described earlier is adopted to perform the precondition part of the fuzzy rules. As a result, the output function of each inference node is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{6}$$

where the $\prod_i u_{ij}^{(2)}$ of a rule node represents the firing strength of its corresponding rule.

Nodes in layer 4 are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are calculated from the FLNN that has not used the function $\tanh(\cdot)$, as shown in Fig. 2. For such a node

$$u_j^{(4)} = u_j^{(3)} \sum_{k=1}^{M} w_{kj}\phi_k \tag{7}$$

where w_{kj} is the corresponding link weight of the FLNN and ϕ_k is the functional expansion of input variables. The functional expansion uses a trigonometric polynomial basis function, given by $[x_1 \sin(\pi\, x_1) \cos(\pi\, x_1) x_2 \sin(\pi\, x_2) \cos(\pi\, x_2)]$ for two-dimensional input variables. Therefore, M is the number of basis functions, $M = 3 \times N$, where N is the number of input variables. Moreover, the output nodes of the FLNN depend on the number of fuzzy rules of the FLNFN model.

The output node in layer 5 integrates all of the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{R} u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \left(\sum_{k=1}^{M} w_{kj}\phi_k\right)}{\sum_{j=1}^{R} u_j^{(3)}} = \frac{\sum_{j=1}^{R} u_j^{(3)} \hat{y}_j}{\sum_{j=1}^{R} u_j^{(3)}} \tag{8}$$

where R is the number of fuzzy rules and y is the output of the FLNFN model.

As described earlier, the number of tuning parameters for the FLNFN model is known to be *(2+3P)NR*, where N, R, and P denote the number of inputs, existing rules, and outputs, respectively. The proposed FLNFN model can be demonstrated to be a universal uniform approximation by the Stone–Weierstrass theorem [20] for continuous functions over compact sets. The detailed proof is given in the Appendix.

III. Learning Algorithms of the FLNFN Model

This section presents an online learning algorithm for constructing the FLNFN model. The proposed learning algorithm comprises a structure learning phase and a parameter learning phase. Fig. 3 presents flow diagram of the learning scheme for the FLNFN model. Structure learning is based on the entropy measure used to determine whether a new rule should be added to satisfy the fuzzy partitioning of input variables. Parameter learning is based on supervised learning algorithms. The backpropagation algorithm minimizes a given cost function by adjusting the link weights in the consequent part and the parameters of the membership functions. Initially, there are no nodes in the network except the input–output nodes, i.e., there are no nodes in the FLNFN model. The nodes are created automatically as learning proceeds, upon the reception of online incoming training data in the structure and parameter learning processes. The rest of this section details the structure learning phase and the parameter learning phase. Finally, in this section, the stability analysis of the FLNFN model based on the Lyapunov approach is performed to ensure that the convergence property holds.

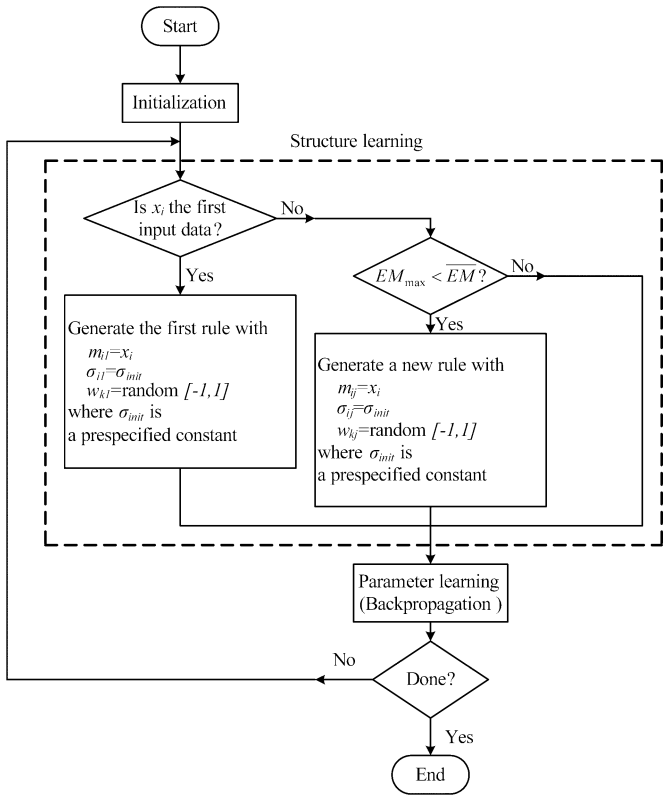


Fig. 3. Flow diagram of the structure/parameter learning for the FLNFN model.

A. Structure Learning Phase

The first step in structure learning is to determine whether a new rule should be extracted from the training data and to determine the number of fuzzy sets in the universe of discourse of each input variable, since one cluster in the input space corresponds to one potential fuzzy logic rule, in which m_{ij} and σ_{ij} represent the mean and variance of that cluster, respectively. For each incoming pattern x_i, the rule firing strength can be regarded as the degree to which the incoming pattern belongs to the corresponding cluster. The entropy measure between each data point and each membership function is calculated based on a similarity measure. A data point of closed mean will have lower entropy. Therefore, the entropy values between data points and current membership functions are calculated to determine whether or not to add a new rule. For computational efficiency, the entropy measure can be calculated using the firing strength from $u_{ij}^{(2)}$ as

$$\mathrm{EM}_j = -\sum_{i=1}^{N} D_{ij} \log_2 D_{ij} \tag{9}$$

where $D_{ij} = \exp(u_{ij}^{(2)^{-1}})$ and $\mathrm{EM}_j \in [0, 1]$. According to (9), the measure is used to generate a new fuzzy rule, and new functional link bases for new incoming data are described as

follows. The maximum entropy measure

$$\mathrm{EM}_{\max} = \max_{1 \le j \le R_{(T)}} \mathrm{EM}_j \tag{10}$$

is determined, where $R(t)$ is the number of existing rules at time t. If $\mathrm{EM}_{\max} \le \overline{\mathrm{EM}}$, then a new rule is generated, where $\overline{\mathrm{EM}} \in [0, 1]$ is a prespecified threshold that decays during the learning process.

In the structure learning phase, the threshold parameter $\overline{\mathrm{EM}}$ is an important parameter. The threshold is set between zero and one. A low threshold leads to the learning of coarse clusters (i.e., fewer rules are generated), whereas a high threshold leads to the learning of fine clusters (i.e., more rules are generated). If the threshold value equals zero, then all the training data belong to the same cluster in the input space. Therefore, the selection of the threshold value $\overline{\mathrm{EM}}$ will critically affect the simulation results. As a result of our extensive experiments and by carefully examining the threshold value $\overline{\mathrm{EM}}$, which uses the range [0, 1], we concluded that there was a relationship between threshold value $\overline{\mathrm{EM}}$ and the number of input variables (N). Accordingly, $\overline{\mathrm{EM}} = \tau N$, where τ belongs to the range [0.26, 0.3].

Once a new rule has been generated, the next step is to assign the initial mean and variance to the new membership function and the corresponding link weight for the consequent part. Since the goal is to minimize an objective function, the mean, variance, and weight are all adjustable later in the parameter learning phase. Hence, the mean, variance, and weight for the new rule are set as

$$m_{ij}^{(R_{(t+1)})} = x_i \tag{11}$$

$$\sigma_{ij}^{(R_{(t+1)})} = \sigma_{\mathrm{init}} \tag{12}$$

$$w_{kj}^{(R_{(t+1)})} = \mathrm{random}[-1, 1] \tag{13}$$

where x_i is the new input and σ_{init} is a prespecified constant. The whole algorithm for the generation of new fuzzy rules and fuzzy sets in each input variable is as follows. No rule is assumed to exist initially.

Step 1: IF x_i is the first incoming pattern THEN do
{Generate a new rule
with mean $m_{i1} = x_i$, variance $\sigma_{i1} = \sigma_{\mathrm{init}}$,
weight $w_{k1} = \mathrm{random}[-1, 1]$
where σ_{init} is a prespecified constant.
}

Step 2: ELSE for each newly incoming x_i, do
{Find $\mathrm{EM}_{\max} = \max_{1 \le j \le R_{(t)}} \mathrm{EM}_j$
IF $\mathrm{EM}_{\max} \ge \overline{\mathrm{EM}}$
do nothing
ELSE
{$R(t+1) = R(t) + 1$
generate a new rule
with mean $m_{iR_{(t+1)}} = x_i$, variance $\sigma_{iR_{(t+1)}} = \sigma_{\mathrm{init}}$,
weight $w_{kR_{(t+1)}} = \mathrm{random}[-1, 1]$
where σ_{init} is a prespecified constant.}
}

B. Parameter Learning Phase

After the network structure has been adjusted according to the current training data, the network enters the parameter learning phase to adjust the parameters of the network optimally based on the same training data. The learning process involves determining the minimum of a given cost function. The gradient of the cost function is computed and the parameters are adjusted with the negative gradient. The backpropagation algorithm is adopted for this supervised learning method. When the single-output case is considered for clarity, the goal to minimize the cost function E is defined as

$$E(t) = \frac{1}{2}[y(t) - y^d(t)]^2 = \frac{1}{2}e^2(t) \tag{14}$$

where $y^d(t)$ is the desired output and $y(t)$ is the model output for each discrete time t. In each training cycle, starting at the input variables, a forward pass is adopted to calculate the activity of the model output $y(t)$.

When the backpropagation learning algorithm is adopted, the weighting vector of the FLNFN model is adjusted such that the error defined in (14) is less than the desired threshold value after a given number of training cycles. The well-known backpropagation learning algorithm may be written briefly as

$$W(t+1) = W(t) + \Delta W(t) = W(t) + \left(-\eta \frac{\partial E(t)}{\partial W(t)}\right) \tag{15}$$

where, in this case, η and W represent the learning rate and the tuning parameters of the FLNFN model, respectively. Let $\mathbf{W} = [m, \sigma, w]^T$ be the weighting vector of the FLNFN model. Then, the gradient of error $E(\cdot)$ in (14) with respect to an arbitrary weighting vector W is

$$\frac{\partial E(t)}{\partial W} = e(t)\frac{\partial y(t)}{\partial W}. \tag{16}$$

Recursive applications of the chain rule yield the error term for each layer. Then the parameters in the corresponding layers are adjusted. With the FLNFN model and the cost function as defined in (14), the update rule for w_j can be derived as

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t) \tag{17}$$

where

$$\Delta w_{kj}(t) = -\eta_w \frac{\partial E}{\partial w_{kj}} = -\eta_w e \left(\frac{u_j^{(3)}\phi_k}{\sum_{j=1}^{R} u_j^{(3)}}\right).$$

Similarly, the update laws for m_{ij} and σ_{ij} are

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij}(t) \tag{18}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta \sigma_{ij}(t) \tag{19}$$

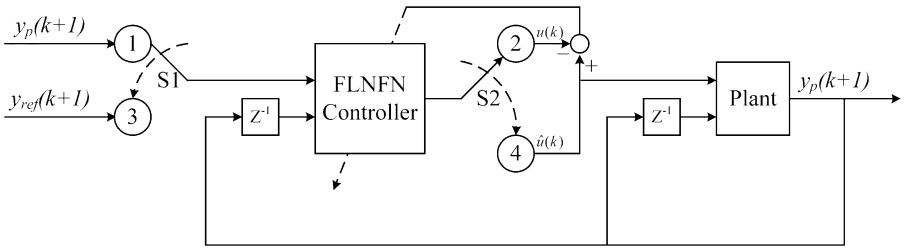


Fig. 4. Conventional online training scheme.

where

$$\Delta m_{ij}(t) = -\eta_m \frac{\partial E}{\partial m_{ij}} = -\eta_m e \left(\frac{u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} \right) \left(\frac{2(u_i^{(1)} - m_{ij})}{\sigma_{ij}^2} \right)$$

$$\Delta \sigma_{ij}(t) = -\eta_\sigma \frac{\partial E}{\partial \sigma_{ij}} = -\eta_\sigma e \left(\frac{u_j^{(4)}}{\sum_{j=1}^{R} u_j^{(3)}} \right) \left(\frac{2(u_i^{(1)} - m_{ij})^2}{\sigma_{ij}^3} \right)$$

where η_w, η_m, and η_σ are the learning rate parameters of the weight, the mean, and the variance, respectively. In this study, both the link weights in the consequent part and the parameters of the membership functions in the precondition part are adjusted by using the backpropagation algorithm. Recently, many researchers [13], [21] tuned the consequent parameters using either LMS or recursive least squares (RLS) algorithms to obtain optimal parameters. However, they still used the backpropagation algorithm to adjust the precondition parameters.

C. Convergence Analysis

The selection of suitable learning rates is very important. If the learning rate is small, convergence will be guaranteed. In this case, the speed of convergence may be slow. However, the learning rate is large, and then the system may become unstable. The Appendix derives varied learning rates, which guarantee convergence of the output error based on the analyses of a discrete Lyapunov function, to train the FLNFN model effectively. The convergence analyses in this study are performed to derive specific learning rate parameters for specific network parameters to ensure the convergence of the output error [22], [23]. Moreover, the guaranteed convergence of output error does not imply the convergence of the learning rate parameters to their optimal values. The following simulation results demonstrate the effectiveness of the online learning FLNFN model based on the proposed delta adaptation law and varied learning rates.

IV. Simulation Results

This study demonstrated the performance of the FLNFN model for nonlinear system control. This section simulates various control examples and compares the performance of the FLNFN model with that of other models. The FLNFN model is adopted to design controllers in four simulations of nonlinear system control problems—a water bath temperature control system [24], control of a bounded-input–bounded-output (BIBO) nonlinear plant [22], control of the ball and beam system [25], and multiinput–multioutput (MIMO) plant control [2].

A. Example 1: Control of Water Bath Temperature System

The goal of this section is to elucidate the control of the temperature of a water bath system according to

$$\frac{dy(t)}{dt} = \frac{u(t)}{C} + \frac{Y_0 - y(t)}{T_R C} \tag{20}$$

where $y(t)$ is the output temperature of the system in degrees Celsius, $u(t)$ is the heat flowing into the system, Y_0 is the room temperature, C is the equivalent thermal capacity of the system, and T_R is the equivalent thermal resistance between the borders of the system and the surroundings.

T_R and C are assumed to be essentially constant, and the system in (20) is rewritten in discrete-time form to some reasonable approximation. The system

$$y(k+1) = e^{-\alpha T s} y(k) + \frac{\delta/\alpha(1 - e^{-\alpha T s})}{1 + e^{0.5y(k)-40}} u(k) + [1 - e^{-\alpha T s}] y_0 \tag{21}$$

is obtained, where α and δ are some constant values of T_R and C. The system parameters used in this example are $\alpha = 1.0015e^{-4}$, $\delta = 8.67973e^{-3}$, and $Y_0 = 25.0\,(^\circ\mathrm{C})$, which were obtained from a real water bath plant considered elsewhere [24]. The input $u(k)$ is limited to 0, and 5 V represents the voltage unit. The sampling period is $T_s = 30$.

The conventional online training scheme is adopted for online training. Fig. 4 presents a block diagram for the conventional online training scheme. This scheme has two phases—the training phase and the control phase. In the training phase, the switches S1 and S2 are connected to nodes 1 and 2, respectively, to form a training loop. In this loop, training data with input vector $I(k) = [y_p(k+1)y_p(k)]$ and desired output $u(k)$ can be defined, where the input vector of the FLNFN controller is the same as that used in the general inverse modeling [26] training scheme. In the control phase, the switches S1 and S2 are connected to nodes 3 and 4, respectively, forming a control loop. In this loop, the control signal $\hat{u}(k)$ is generated according to the input vector $I'(k) = [y_{\mathrm{ref}}(k+1)y_p(k)]$, where y_p is the plant output and y_{ref} is the reference model output.

A sequence of random input signals $u_{\mathrm{rd}}(k)$ limited to 0 and 5 V is injected directly into the simulated system described in (21), using the online training scheme for the FLNFN controller.

The 120 training patterns are selected based on the input–outputs characteristics to cover the entire reference output. The temperature of the water is initially 25 °C, and rises progressively when random input signals are injected. After 10 000 training iterations, four fuzzy rules are generated. The obtained fuzzy rules are as follows.

Rule 1: IF x_1 is $\mu(32.416, 11.615)$ and x_2 is $\mu(27.234, 7.249)$

THEN $\hat{y}_1 = 32.095x_1 + 74.849\sin(\pi\, x_1) - 34.546\cos(\pi\, x_1)$
$-17.026x_2 - 41.799\sin(\pi\, x_2) + 35.204\cos(\pi\, x_2)$.

Rule 2: IF x_1 is $\mu(34.96, 9.627)$ and x_2 is $\mu(46.281, 13.977)$

THEN $\hat{y}_2 = 21.447x_1 + 11.766\sin(\pi\, x_1) - 77.705\cos(\pi\, x_1)$
$-52.923x_2 - 61.827\sin(\pi\, x_2) + 70.946\cos(\pi\, x_2)$.

Rule 3: IF x_1 is $\mu(62.771, 6.910)$ and x_2 is $\mu(62.499, 15.864)$

THEN $\hat{y}_3 = 25.735x_1 - 10.907\sin(\pi\, x_1) - 46.359\cos(\pi\, x_1)$
$-40.322x_2 + 36.752\sin(\pi\, x_2) + 103.33\cos(\pi\, x_2)$.

Rule 4: IF x_1 is $\mu(79.065, 8.769)$ and x_2 is $\mu(64.654, 9.097)$

THEN $\hat{y}_4 = 46.055x_1 - 37.223\sin(\pi\, x_1) - 57.759\cos(\pi\, x_1)$
$-5.8152x_2 + 61.065\sin(\pi\, x_2) + 34.838\cos(\pi\, x_2)$.

This study compares the FLNFN controller to the PID controller [27], the manually designed fuzzy controller [4], the FLNN [17], and the TSK-type NFN [12]. Each of these controllers is applied to the water bath temperature control system. The performance measures include the set points regulation, the influence of impulse noise, and a large parameter variation in the system, and the tracking capability of the controllers.

The first task is to control the simulated system to follow three set points

$$y_{\rm ref}(k) = \begin{cases} 35\,^{\circ}\text{C}, & \text{for } k \le 40 \\ 55\,^{\circ}\text{C}, & \text{for } 40 < k \le 80 \\ 75\,^{\circ}\text{C}, & \text{for } 80 < k \le 120. \end{cases} \tag{22}$$

Fig. 5(a) presents the regulation performance of the FLNFN controller. The regulation performance was also tested using the FLNN controller, and the TSK-type NFN controller. Fig. 5(b) plots the error curves of the FLNFN controller, the FLNN controller, and the TSK-type NFN controller between $k = 81$ and 100. In this figure, the FLNFN controller obtains smaller errors than the other two controllers. To test their regulation performance, a performance index, the sum of absolute error (SAE), is defined by

$$\text{SAE} = \sum_k |y_{\rm ref}(k) - y(k)| \tag{23}$$

where $y_{\rm ref}(k)$ and $y(k)$ are the reference output and the actual output of the simulated system, respectively. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TKS-type NFN controller are 352.8, 418.5, 401.5, 379.2, and 361.9, which are given in the second row of Table I. The proposed FLNFN controller has a much better SAE value of regulation performance than the other controllers.

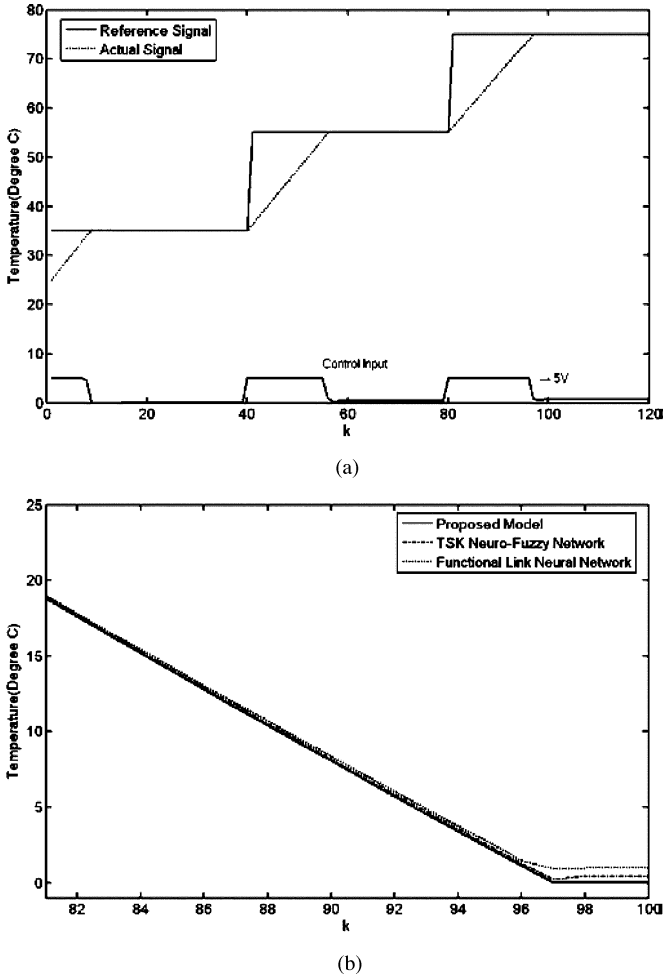


Fig. 5. (a) Final regulation performance of FLNFN controller in water bath system. (b) Error curves of the FLNFN controller, TSK-type NFN controller, and FLNN controller between $k = 81$ and $k = 100$.

The second set of simulations is performed to elucidate the noise rejection ability of the five controllers when some unknown impulse noise is imposed on the process. One impulse noise value of –5 °C is added to the plant output at the 60th sampling instant. A set point of 50 °C is adopted in this set of simulations. For the FLNFN controller, the same training scheme, training data, and learning parameters were used as in the first set of simulations. Fig. 6(a) and (b) presents the behaviors of the FLNFN controller under the influence of impulse noise and the corresponding errors, respectively. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 270.4, 311.5, 275.8, 324.51, and 274.75, which are shown in the third row of Table I. The FLNFN controller performs quite well. It recovers very quickly and steadily after the occurrence of the impulse noise.

One common characteristic of many industrial-control processes is that their parameters tend to change in an unpredictable way. The value of $0.7u(k-2)$ is added to the plant input after the 60th sample in the third set of simulations to test the robustness of the five controllers. A set point of $50\,^{\circ}$C is adopted in this

TABLE I
COMPARISON OF PERFORMANCE OF VARIOUS CONTROLLERS IN EXAMPLE 1

$SAE=\sum_{k=1}^{120}\lvert y_{ref}(k)-y(k)\rvert$	**FLNFN Controller**	PID Controller [26]	Fuzzy Controller [3]	FLNN Controller [17]	TSK-type NFN Controller [12]
Regulation Performance	**352.84**	418.5	401.5	379.22	361.96
Influence of Impulse Noise	**270.41**	311.5	275.8	324.51	274.75
Effect of Change in Plant Dynamics	**263.35**	322.2	273.5	311.54	265.48
Tracking Performance	**44.28**	100.6	88.1	98.43	54.28

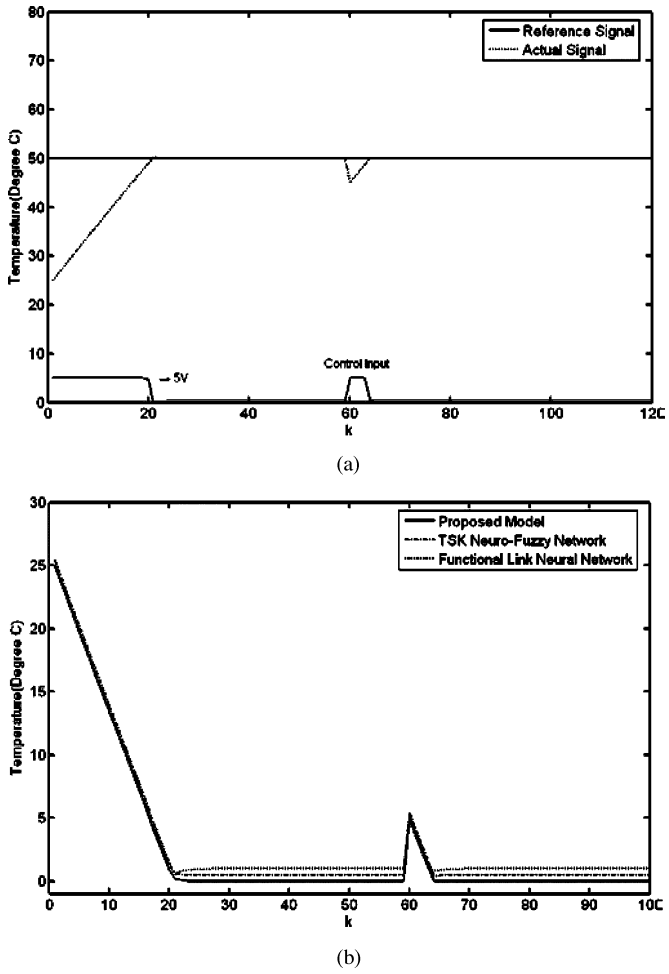


Fig. 6. (a) Behavior of FLNFN controller under impulse noise in water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller, and FLNN controller.

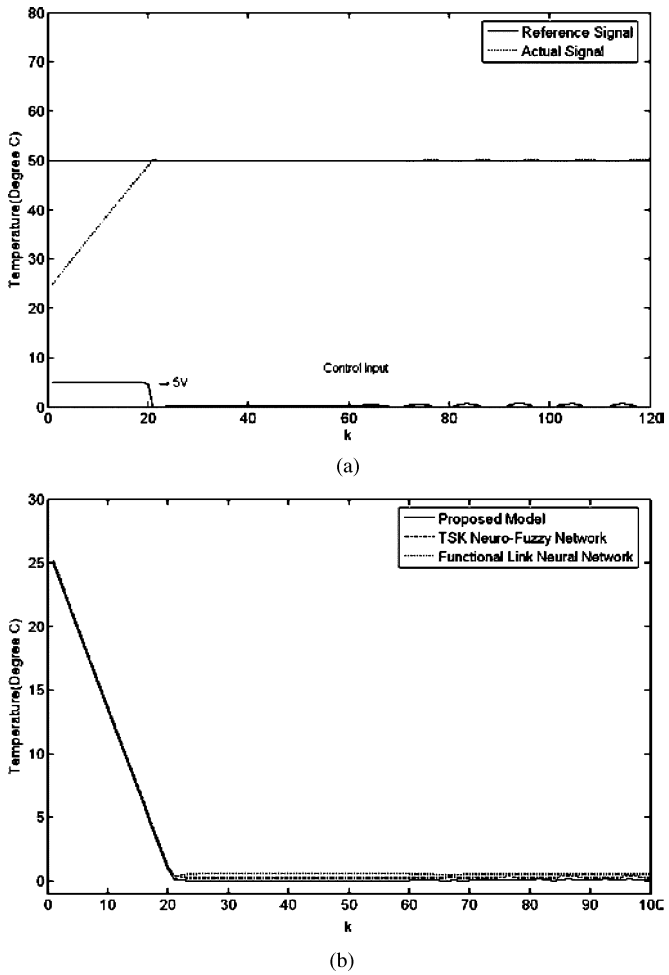
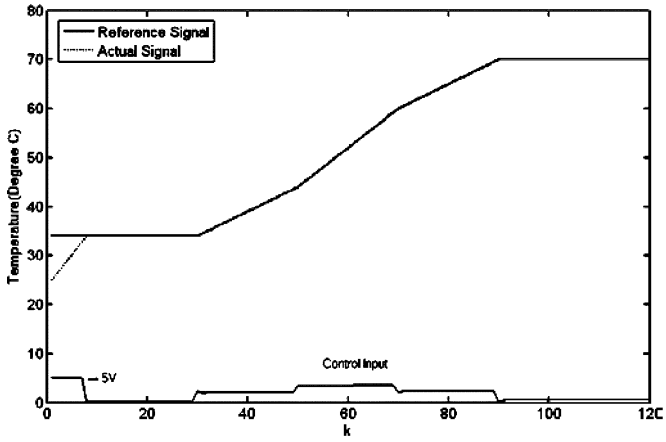


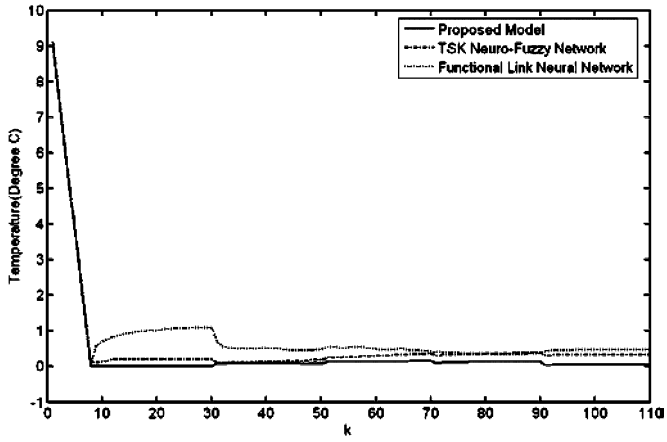
Fig. 7. (a) Behavior of FLNFN controller when a change occurs in the water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller, and FLNN controller.

set of simulations. Fig. 7(a) presents the behaviors of the FLNFN controller when the plant dynamics change. Fig. 7(b) presents the corresponding errors of the FLNFN controller, the FLNN controller and the TSK-type NFN controller. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 263.3, 322.2, 273.5, 311.5, and 265.4, which are shown in the fourth row of Table I. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN controller in the water bath system.

In the final set of simulations, the tracking capability of the FLNFN controller with respect to ramp-reference signals is



(a)



(b)

Fig. 8. (a) Tracking of FLNFN controller when a change occurs in the water bath system. (b) Error curves of FLNFN controller, TSK-type NFN controller, and FLNN controller.

studied. Define

$$y_{\mathrm{ref}}(k) = \begin{cases} 34\,^{\circ}\mathrm{C}, & \text{for } k \le 30 \\ (34 + 0.5(k-30))\,^{\circ}\mathrm{C}, & \text{for } 30 < k \le 50 \\ (44 + 0.8(k-50))\,^{\circ}\mathrm{C}, & \text{for } 50 < k \le 70 \\ (60 + 0.5(k-70))\,^{\circ}\mathrm{C}, & \text{for } 70 < k \le 90 \\ 70\,^{\circ}\mathrm{C}, & \text{for } 90 < k \le 120 \end{cases} \quad (24)$$

Fig. 8(a) presents the tracking performance of the FLNFN controller. Fig. 8(b) presents the corresponding errors of the FLNFN controller, the FLNN controller, and the TSK-type NFN controller. The SAE values of the FLNFN controller, the PID controller, the fuzzy controller, the FLNN controller, and the TSK-type NFN controller are 44.2, 100.6, 88.1, 98.4, and 54.2, which are shown in the fifth row of Table I. The results present the favorable control and disturbance rejection capabilities of the trained FLNFN controller in the water bath system. The aforementioned simulation results, presented in Table I, demonstrate that the proposed FLNFN controller outperforms other controllers.

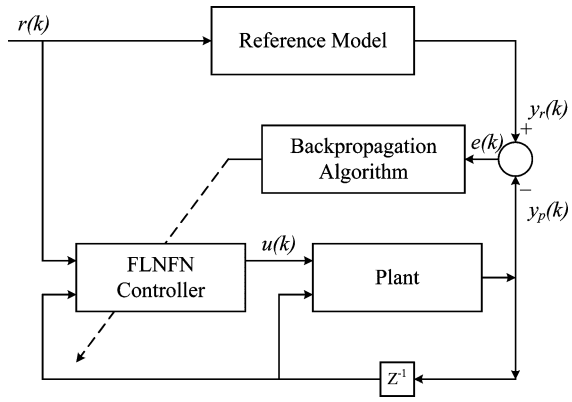


Fig. 9. Block diagram of FLNFN controller-based control system.

B. Example 2: Control of BIBO Nonlinear Plant

In this case, the plant is described by the difference equation

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k). \quad (25)$$

The reference model is described by the difference equation

$$y_r(k+1) = 0.6y_r(k) + r(k) \quad (26)$$

where $r(k) = \sin(2\pi k/10) + \sin(2\pi k/25)$. Fig. 9 presents the block diagram of the FLNFN-based control system. The inputs to the FLNFN controller are the reference input, the previous plant output, and the previous control signal; the output of the FLNFN controller is the control signal to the plant. The online algorithm developed in this study is adopted to adjust the structure and the parameters of the FLNFN controller such that the error between the output of the plant and the desired output from a reference model approaches a small value after some training cycles.

After 500 training iterations, six fuzzy rules are generated. In this example, the proposed FLNFN controller is compared to the FLNN controller [17] and the TSK-type NFN controller [12]. Each of the controllers is applied to control the BIBO nonlinear plant. In the following four cases, the FLNFN controller is demonstrated to outperform the other models.

In the first case, the reference input is given by (26) and the final result is shown in Fig. 10(a). Fig. 10(b) presents the error curves of the FLNFN controller and the TSK-type NFN controller. In this figure, the FLNFN controller yields smaller errors than the TSK-type NFN controller. In the second case, after 100 epochs, the reference input is changed to $r(k) = \sin(2\pi k/25)$. Fig. 11(a) and (b) plots the result of the FLNFN controller and the corresponding errors of the FLNFN controller and the TSK-type NFN controller. In the third case, after 100 epochs, the reference input is changed to an impulse signal. Fig. 12(a) presents the simulation result. Fig. 12(b) presents the corresponding errors of the FLNFN controller, the FLNN controller, and the TSK-type NFN controllers. In the fourth case, a disturbance of 2.0 is added to the system between the 100th and the 150th epochs. In this case, the FLNFN-based control system can recover from the disturbance

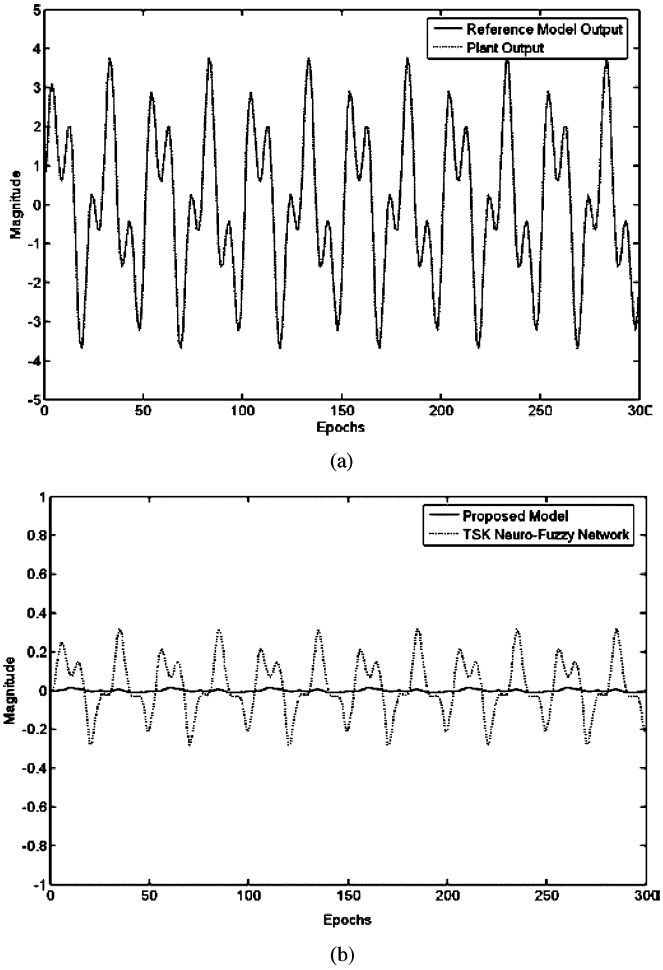


Fig. 10. Final system response in first case of Example 2. (a) Dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.

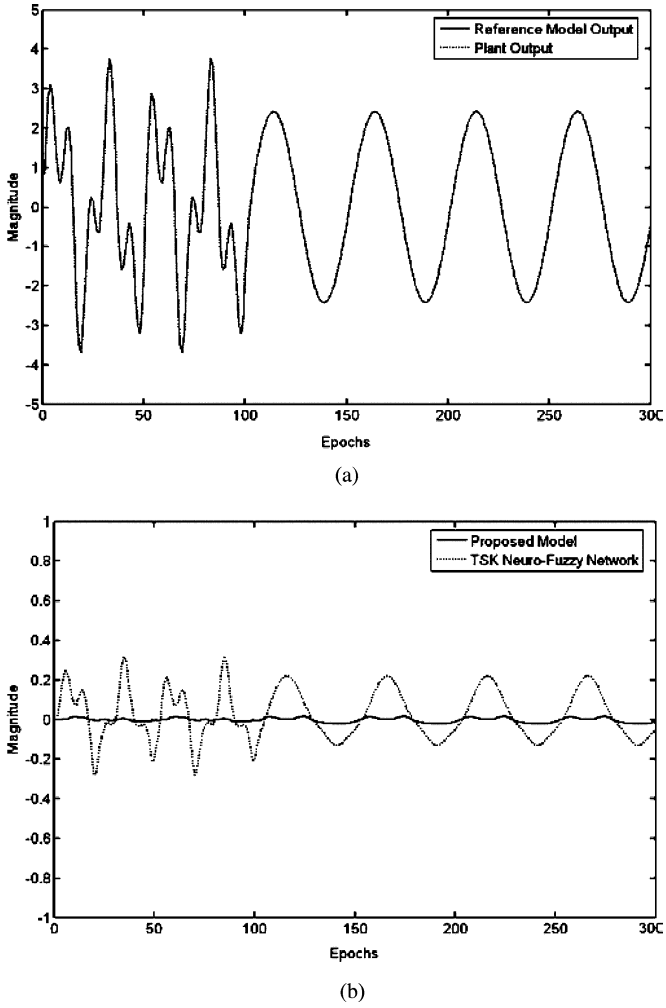


Fig. 11. Final system response in second case of Example 2. (a) Dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.

quickly, as shown in Fig. 13. The rms error is adopted to evaluate the performance. Table II presents the rms errors of the FLNFN controller, the FLNN controller, and the TSK-type NFN controller. Table II shows that, according to the simulation results, the proposed FLNFN controller outperforms the other models.

C. Example 3: Control of Ball and Beam System

Fig. 14 presents the ball and beam system [25]. The beam is made to rotate in the vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. The ball must remain in contact with the beam.

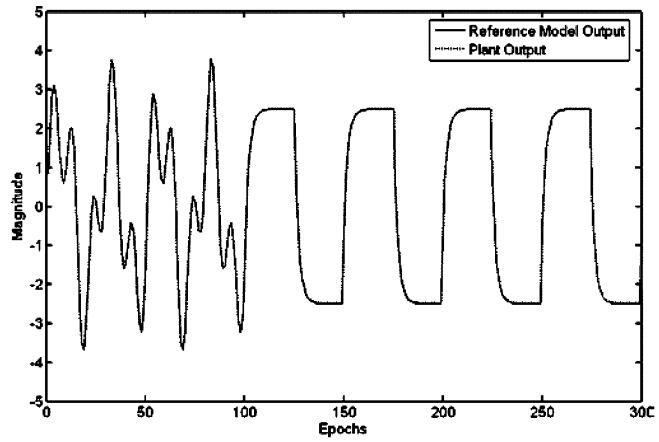
The ball and beam system can be written in state space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B\left(x_1 x_4^2 - G\ \sin\ x_3\right) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u, \quad y = x_1 \qquad (27)$$

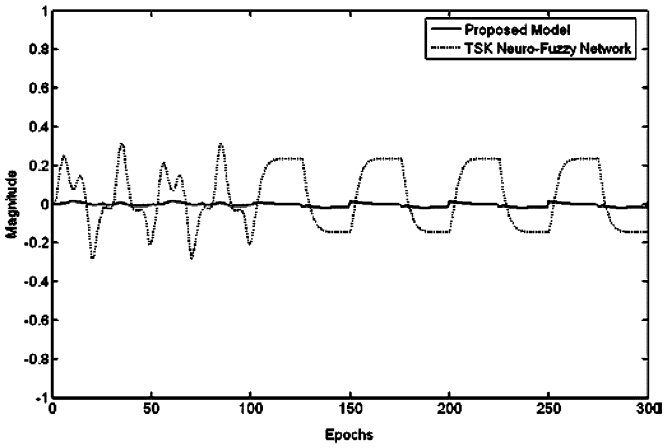
where $x = (x_1, x_2, x_3, x_4)^T \equiv (r, \dot{r}, \theta, \dot{\theta})^T$ is the state of the system and $y = x_1 \equiv r$ is the output of the system. The control u is the angular acceleration ($\ddot{\theta}$) and the parameters $B = 0.7143$ and $G = 9.81$ are set in this system. The purpose of control is to determine $u(x)$ such that the closed-loop system output y converges to zero from different initial conditions.

According to the input/output linearization algorithm [25], the control law $u(x)$ is determined as follows: for state x, compute $v(x) = -\alpha_3\phi_4(x) - \alpha_2\phi_3(x) - \alpha_1\phi_2(x) - \alpha_0\phi_1(x)$, where $\phi_1(x) = x_1$, $\phi_2(x) = x_2$, $\phi_3(x) = -BG\ \sin\ x_3$, $\phi_4(x) = -BGx_4\ \cos\ x_3$, and α_i's are chosen such that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is a Hurwitz polynomial. Compute $a(x) = -BG\ \cos\ x_3$ and $b(x) = BGx_4^2\ \sin\ x_3$; then $u(x) = [v(x) - b(x)]/a(x)$.

In this simulation, the differential equations are solved using the second-/third-order Runge–Kutta method. The FLNFN model is trained to approximate the aforementioned conventional controller of a ball and beam system. $u(x) = [v(x) - b(x)]/a(x)$ is adopted to generate the input/output training pairs with x obtained by randomly sampling 200 points in



(a)



(b)

Fig. 12. Final system response in third case of Example 2. (a) Dashed line represents plant output and the solid line represents the reference model. (b) Error curves of FLNFN controller and TSK-type NFN controller.

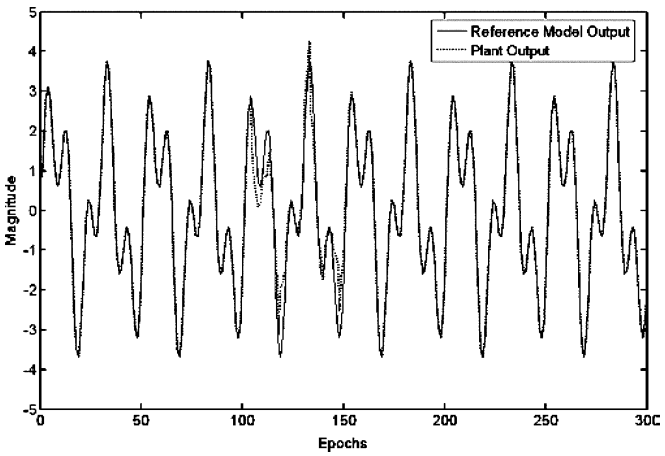


Fig. 13. Final system response in fourth case of Example 2. The dashed line represents plant output and the solid line represents the reference model.

the region $U = [-5, 5] \times [-3, 3] \times [-1, 1] \times [-2, 2]$. After online structure-parameter learning, 14 fuzzy rules are generated. The controller after learning was tested under the following four initial conditions: $x(0) = [2.4, -0.1, 0.6, 0.1]^T$, $[1.6,\ 0.05,\ -0.5,\ -0.05]^T$, $[-1.6,\ -0.05,\ 0.5,\ 0.05]^T$, and $[-2.4, 0.1, -0.6, -0.1]^T$. Fig. 15 plots the output responses of the closed-loop ball and beam system controlled by the FLNFN model and the TSK-type NFN model. These responses approximate those of the original controller under the four initial conditions. In this figure, the curves of the FLNFN model quickly tend to stabilize. Fig. 16 also presents the behavior of the four states of the ball and beam system, starting at the initial condition $[-2.4, 0.1, -0.6, -0.1]^T$. In this figure, the four states of the system gradually decay to zero. The results demonstrate the perfect control capability of the trained FLNFN model. The performance of the FLNFN controller is compared with that of the FALCON controller [8], the FLNN controller [17], and the TSK-type NFN controller [12]. Table III presents the comparison results. The results demonstrate that the proposed FLNFN controller outperforms other controllers.

TABLE II
COMPARISON OF PERFORMANCE OF VARIOUS MODELS IN EXAMPLE 2

Method	**FLNFN Controller**	FLNN Controller [17]	TSK-type NFN Controller [12]
Training Steps	**500**	1000	500
Parameter Numbers	**6 rules/ 60 parameters**	79 parameters	9 rules/ 63 parameters
RMS error of case1	**0.0004**	0.0211	0.0084
RMS error of case2	**0.0006**	0.0208	0.0075
RMS error of case3	**0.0007**	0.0303	0.0095

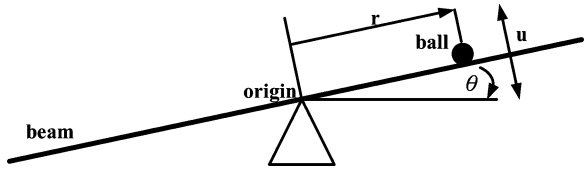


Fig. 14. Ball and beam system.

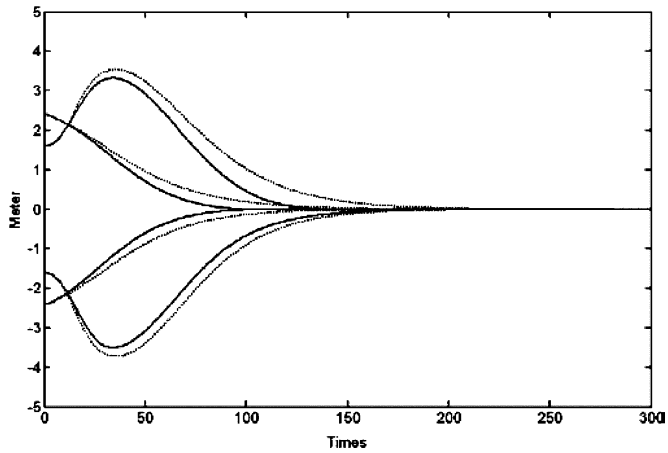


Fig. 15. Responses of ball and beam system controlled by FLNFN model (solid curves) and TSK-type NFN model (dotted curves) under four initial conditions.

TABLE III
COMPARISON OF PERFORMANCE OF EXISTING MODELS IN EXAMPLE 3

Method	**FLNFN Controller**	FALCON Controller [8]	FLNN Controller [17]	TSK-type NFN Controller [12]
Training Steps	**500**	50000	1000	500
Parameter Numbers	**14 rules/ 280 parameters**	28 rules/ 280 parameters	317 parameters	22 rules/ 286 parameters
RMS errors	**0.056**	0.2	0.153	0.079

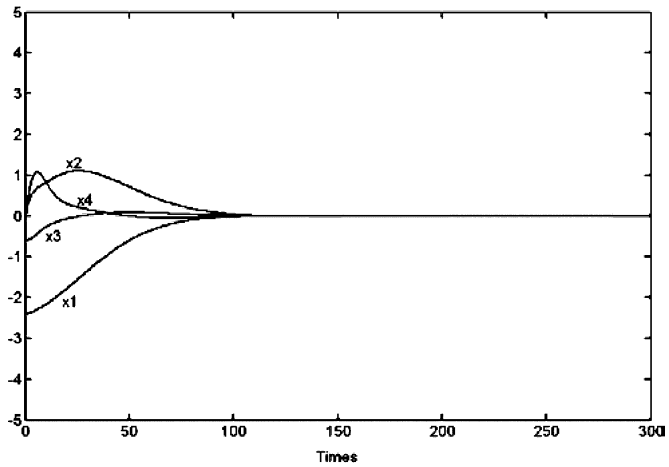


Fig. 16. Responses of four states of ball and beam system under the control of the trained FLNFN controller.

D. Example 4: Control of MIMO Plant

In this example, the MIMO plants [2] to be controlled are described by the equations

$$\begin{bmatrix} y_{p1}(k+1) \\ y_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.5\dfrac{y_{p1}(k)}{1+y_{p2}^2(k)} \\ 0.5\dfrac{y_{p1}(k)y_{p2}(k)}{1+y_{p2}^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \quad (28)$$

The controlled outputs should follow the desired output y_{r1} and y_{r2} as specified by the following 250 pieces of data

$$\begin{bmatrix} y_{r1}(k) \\ y_{r2}(k) \end{bmatrix} = \begin{bmatrix} \sin(k\pi/45) \\ \cos(k\pi/45) \end{bmatrix}. \quad (29)$$

The inputs of the FLNFN are $y_{p1}(k)$, $y_{p2}(k)$, $y_{r1}(k)$, and $y_{r2}(k)$, and the outputs are $u_1(k)$ and $u_2(k)$. After 500 training iterations, four fuzzy rules are generated. In this example, the proposed FLNFN controller is compared to the FLNN controller [17] and the TSK-type NFN controller [12]. Each of the controllers is applied to control the MIMO plant. To demonstrate the performance of the proposed controller, Fig. 17(a) and (b) plots the control results of the desired output and the model output using FLNFN controller. Fig. 17(c) and (d) shows the error curves of the FLNFN controller and the TSK-type NFN controller. Table IV presents the rms errors of the FLNFN controller, the FLNN controller, and the TSK-type NFN controller. Table IV shows that, according to the simulation

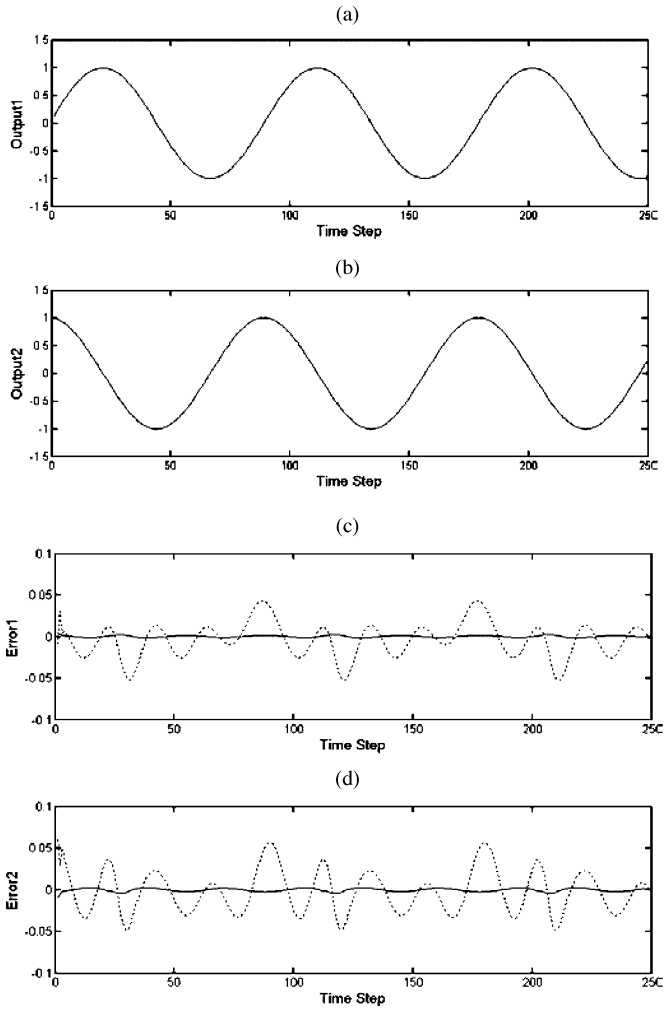


Fig. 17. Desired output (solid line) and model output using FLNFN controller (dotted line). (a) Output 1. (b) Output 2 in Example 4. Error curves of FLNFN controller (solid line) and TSK-type NFN controller (dotted line). (c) Output 1. (d) Output 2.

TABLE IV
COMPARISON OF PERFORMANCE OF EXISTING MODELS IN EXAMPLE 4

Method	**FLNFN Controller**	FLNN Controller [17]	TSK-type NFN Controller [12]
Training Steps	**500**	1000	500
Parameter Numbers	**4 rules/ 128 parameters**	161 parameters	10 rules/ 140 parameters
RMS errors	**0.0002**	0.0738	0.0084

results, the proposed FLNFN controller is better than the other controllers.

V. Conclusion and Future Works

This study proposes an FLNFN structure for nonlinear system control. The FLNFN model uses an FLNN to the consequent part of the fuzzy rules. The FLNFN model can automatically construct and adjust free parameters by performing online structure/parameter learning schemes concurrently. The FLNFN model was proven to be a universal approximator and convergence-stable. Finally, the proposed FLNFN model yields better simulation results than other existing models under some circumstances.

Three advanced topics on the proposed FLNFN model should be addressed in future research. First, the FLNFN model will tend to apply high-order nonlinear or overly complex systems if it can suitably adopt the consequent part of a nonlinear combination of input variables, and a functional expansion of multiple trigonometric polynomials. Therefore, it should be analyzed to determine how many trigonometric polynomials for functional expansion should be used in the future. Second, the backpropagation technique is slow for many practical applications. To speed up the convergence and make the algorithm more practical, some variations are necessary. For example, the heuristic techniques include such ideas as varying the learning rate and using momentum [31], and the standard numerical optimization techniques in the BP procedure. In the standard numerical optimization techniques, the conjugate gradient algorithm [32] and the Levenberg–Marquardt algorithm [33] have been applied to the training of neural networks and shown a faster convergence than the basic BP algorithm. On the other hand, since the backpropagation technique is used to minimize the error function, the results may reach the local minima solution. In future work, we will adopt genetic algorithms (GAs) [28], [29] to solve the local minima problem. A GA is a global search technique. Because it can simultaneously evaluate many points in the search space, it is more likely to converge toward the global solution. Third, it would be better if the FLNFN model has the ability to delete unnecessary or redundant rules. The fuzzy similarity measure [30] determines the similarity between two fuzzy sets in order to prevent existing membership functions from being too similar.

Appendix

A. Proof of the Universal Approximator Theorem

The Stone–Weierstrass theorem [15] is adopted to prove the universal approximator theorem. For a clear description in the FLNFN model, only the multiinput–single-output (MISO) function $f : x \in \Re^N \rightarrow y \in \Re$ is considered. The FLNFN is expressed as

$$y(x) = \frac{\sum_{j=1}^{R} \hat{y}_j u_j^{(3)}(x)}{\sum_{j=1}^{R} u_j^{(3)}(x)}. \tag{A1}$$

Theorem A1 (Stone–Weierstrass Theorem): Let A be a set of real continuous functions on a compact set U. 1) If U is an algebra such that if $f_1, f_2 \in \mathrm{A}$, and $c \in \mathrm{R}$, then $f_1 + f_2 \in \mathrm{A}$, $f_1 \cdot f_2 \in \mathrm{A}$, and $cf_1 \in \mathrm{A}$; 2) A separates points on U, meaning that for $x, y \in \mathrm{U}$, $x \neq y$, there exists $f_1 \in \mathrm{A}$ such that $f_1(x) \neq f_2(y)$; and 3) A vanishes at no point of U, meaning that for each $x \in \mathrm{U}$, there exists $f_1 \in \mathrm{A}$ such that $f_1(x) \neq 0$. Then the uniform closure of A consists of all real continuous functions on U.

Lemma A1: Let Y be the family of function $y : \Re^N \rightarrow \Re$ defined in (A1); then $\mathrm{Y} \rightarrow \mathrm{U}$, where U is a compact set.

Proof of Lemma A1: Here, the membership function is

$$0 < \mu_{A_{ij}}(x) = \exp\left[-\frac{(x_i - m_{ij})^2}{\sigma_{ij}^2}\right] \leq 1.$$

Therefore, the continuous function $u_j^{(3)}$ is closed and bounded for all $x \in \Re^N$. That is, $\mathrm{Y} \subset \mathrm{U}$.

Proof of Theorem A1: First, we prove that Y is algebra. Let $f_1, f_2 \in \mathrm{Y}$, such that they can be written as (A2) and (A3) shown at the bottom of the page, where $\hat{y}1_j$ and $\hat{y}2_j \in \Re\ \forall j$.

Therefore,

$$f_1 + f_2(x) = \frac{\sum_{j1=1}^{R1}\sum_{j2=1}^{R2}(\hat{y}1_{j1} + \hat{y}2_{j2}) \cdot (\prod_{i=1}^{N} u1_{j1}^{(3)} u2_{j2}^{(3)})}{\sum_{j1=1}^{R1}\sum_{j2=1}^{R2}(\prod_{i=1}^{N} u1_{j1}^{(3)} u2_{j2}^{(3)})}. \tag{A4}$$

$$\begin{aligned} f_1(x) &= \frac{\sum_{j1=1}^{R1} \hat{y}1_{j1} u1_{j1}^{(3)}}{\sum_{j1=1}^{R1} u1_{j1}^{(3)}} \\ &= \frac{\sum_{j1=1}^{R1} (w1_{1,j1}\phi 1_1 + \cdots + w1_{M,j1}\phi 1_M) \cdot \left[\prod_{i1=1}^{N1} \exp\left[-(x_{i1} - m1_{i1,j1})^2 / \sigma 1_{i1,j1}^2\right]\right]}{\sum_{j1=1}^{R1}\left[\prod_{i1=1}^{N1} \exp\left[-(x_{i1} - m1_{i1,j1})^2 / \sigma 1_{i1,j1}^2\right]\right]} \end{aligned} \tag{A2}$$

$$\begin{aligned} f_2(x) &= \frac{\sum_{j2=1}^{R2} \hat{y}2_{j2} u2_{j2}^{(3)}}{\sum_{j2=1}^{R2} u2_{j2}^{(3)}} \\ &= \frac{\sum_{j2=1}^{R2} (w2_{1,j2}\phi 2_1 + \cdots + w2_{M,j2}\phi 2_M). \left[\prod_{i2=1}^{N2} \exp\left[-(x_{i2} - m2_{i2,j2})^2 / \sigma 2_{i2,j2}^2\right]\right]}{\sum_{j2=1}^{R2}\left[\prod_{i2=1}^{N2} \exp\left[-(x_{i2} - m2_{i2,j2})^2 / \sigma 2_{i2,j2}^2\right]\right]} \end{aligned} \tag{A3}$$

Since $u1_j^{(3)}$ and $u2_j^{(3)}$ are Gaussian in form, i.e., this can be verified by straightforward algebraic operations, hence, (A4) is in the same form as (A1), so that $f_1 + f_2 \in \mathrm{Y}$. Similarly, we have

$$f_1 f_2(x) = \frac{\sum_{j1=1}^{R1}\sum_{j2=1}^{R2}(\hat{y}1_{j1}\cdot\hat{y}2_{j2})(\prod_{i=1}^{N} u1_{j1}^{(3)}u2_{j2}^{(3)})}{\sum_{j1=1}^{R1}\sum_{j2=1}^{R2}(\prod_{i=1}^{N} u1_{j1}^{(3)}u2_{j2}^{(3)})} \tag{A5}$$

which is also in the same form as (A1); hence, $f_1 f_2 \in \mathrm{Y}$, Finally, for arbitrary $c \in \Re$

$$cf_1(x) = \frac{\sum_{j1=1}^{R1}(c\cdot\hat{y}1_{j1})(\prod_{i=1}^{N} u1_{j1}^{(3)})}{\sum_{j1=1}^{R1}(\prod_{i=1}^{N} u1_{j1}^{(3)})} \tag{A6}$$

which is again in the form of (A1); hence, $cf_1 \in \mathrm{Y}$. Therefore, Y is an algebra.

Next, Y is proven to separate points on U by constructing a required f; $f \in \mathrm{Y}$ is specified such that $f(x') \neq f(y')$ for arbitrarily given $x', y' \in \mathrm{U}$ with $x' \neq y'$. Two fuzzy rules in the form of (3) are chosen for the fuzzy rule base.

Let $\underline{x}' = (x_1', x_2', \ldots, x_N')$ and $\underline{y}' = (y_1', y_2', \ldots, y_N')$. If $x_i' \neq y_i'$, then two fuzzy rules can be chosen as the fuzzy rule base. Furthermore, let the Gaussian membership functions be

$$\mu_{A_{i1}}(x_i) = \exp\left[-\frac{(x_i - x_i')^2}{\sigma^2}\right] \tag{A7}$$

$$\mu_{A_{i2}}(x_i) = \exp\left[-\frac{(x_i - y_i')^2}{\sigma^2}\right]. \tag{A8}$$

Then, f can be expressed as (A9), shown at the bottom of the page, where $\hat{y}1$ and $\hat{y}2$ are outputs of the local FLNN model calculated for output y and rule 1, rule 2, and $\hat{y}_j \in \Re\ \forall j$. With this system

$$f(\underline{x}') = \frac{\hat{y}1 + \hat{y}2\left[\prod_{i=1}^{N}\exp\left[-(x_i' - y_i')^2/\sigma_{i2}^2\right]\right]}{1 + \left[\prod_{i=1}^{N}\exp\left[-(x_i' - y_i')^2/\sigma_{i2}^2\right]\right]} \tag{A10}$$

$$f(\underline{y}') = \frac{\hat{y}2 + \hat{y}1\left[\prod_{i=1}^{N}\exp\left[-(y_i' - x_i')^2/\sigma_{i1}^2\right]\right]}{1 + \left[\prod_{i=1}^{N}\exp\left[-(y_i' - x_i')^2/\sigma_{i1}^2\right]\right]} \tag{A11}$$

Since $\underline{x}' \neq \underline{y}'$, some i must exist such that $x_i' \neq y_i'$; hence, $f(\underline{x}') \neq f(\underline{y}')$. Therefore, Y separates points on U.

Finally, Y is proven to vanish at no point of U. By (A1), $u_j^{(3)}(x)$ is constant and does not equal zero. That is, for all $x \in \Re^N$, $u_j^{(3)}(x) > 0$. If $u_j^{(3)}(x) > 0,\ (j = 1, 2, \ldots, R)$, then $y > 0$ for any $x \in \Re^N$. That is, any $y \in \mathrm{Y}$ with $u_j^{(3)}(x) > 0$ can serve as the required f.

In summary, the FLNFN model is a universal approximator, and using the Stone–Weierstrass theorem and the fact that Y is a continuous real set on U proves the theorem.

B. Proof of Convergence Theorem

Theorem B1: Let η_w be the learning rate parameter of the FLNFN weight, and let $P_{w\max}$ be defined as $P_{w\max} \equiv \max_k \|P_w(k)\|$, where $P_w(k) = \partial y/\partial w_{kj}$ and $\|\cdot\|$ is the Euclidean norm in $\Re^N$. The convergence is guaranteed if η_w is chosen as $\eta_w = \lambda/(P_{w\max})^2 = \lambda/R$, in which λ is a positive constant gain, and R is the number of rules in the FLNFN model.

Proof of Theorem B1: Since

$$P_w(k) = \frac{\partial y}{\partial w_{kj}} = \frac{u_j^{(3)}\phi_k}{\sum_{j=1}^{R} u_j^{(3)}} \tag{B1}$$

and $u_j^{(3)}\phi_k/\sum_{j=1}^{R} u_j^{(3)} \leq 1$, the following result holds

$$\|P_w(k)\| \leq \sqrt{R}. \tag{B2}$$

Then, a discrete Lyapunov function is selected as

$$V(k) = \frac{1}{2}e^2(k). \tag{B3}$$

The change in the Lyapunov function is obtained as

$$\begin{aligned}\Delta V(k) &= V(k+1) - V(k)\\ &= \frac{1}{2}\left[e^2(k+1) - e^2(k)\right].\end{aligned} \tag{B4}$$

The error difference can be represented as [16]

$$\begin{aligned}e(k+1) &= e(k) + \Delta e(k)\\ &= e(k) + \left[\frac{\partial e(k)}{\partial w_{kj}}\right]^T \Delta w_{kj}\end{aligned} \tag{B5}$$

where Δe and Δw_k represent the output error change and the weight change in the output layer, respectively. Equations (17) and (B5) yield

$$\frac{\partial e(k)}{\partial w_{kj}} = \frac{\partial e(k)}{\partial y}\frac{\partial y}{\partial w_{kj}} = P_w(k) \tag{B6}$$

$$e(k+1) = e(k) - P_w^T(k)\eta_w e(k) P_w(k). \tag{B7}$$

Then,

$$\begin{aligned}\|e(k+1)\| &= \left\|e(k)\left[1 - \eta_w P_w^T(k)P_w(k)\right]\right\|\\ &\leq \|e(k)\|\cdot\left\|1 - \eta_w P_w^T(k)P_w(k)\right\|\end{aligned} \tag{B8}$$

is true. If $\eta_w = \lambda/(P_{w\max}^2) = \lambda/R$ is chosen, then the term $\left\|1 - \eta_w P_w^T(k)P_w(k)\right\|$ in (B8) is less than 1. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ is guaranteed. The output error between the reference model and actual plant converges to zero as $t \to \infty$. This fact completes the proof of the theorem.

$$f = \frac{\hat{y}1\left[\prod_{i=1}^{N}\exp\left[-(x_i - x_i')^2/\sigma_{i1}^2\right]\right] + \hat{y}2\left[\prod_{i=1}^{N}\exp\left[-(x_i - y_i')^2/\sigma_{i2}^2\right]\right]}{\left[\prod_{i=1}^{N}\exp\left[-(x_i - x_i')^2/\sigma_{i1}^2\right]\right] + \left[\prod_{i=1}^{N}\exp\left[-(x_i - y_i')^2/\sigma_{i2}^2\right]\right]} \tag{A9}$$

The following lemmas [17] are used to prove Theorem 2.

Lemma B1: Let $g(h) = h\exp(-h^2)$, then $|g(h)| < 1$ $\forall h \in \Re$.

Lemma B2: Let $f(h) = h^2\exp(-h^2)$, then $|f(h)| < 1$ $\forall h \in \Re$.

Theorem B2: Let η_m and η_σ be the learning rate parameters of the mean and standard deviation of the Gaussian function for the FLNFN; let $P_{m\max}$ be defined as $P_{m\max} \equiv \max_k \|P_m(k)\|$, where $P_m(k) = \partial y/\partial m_{ij}$; let $P_{\sigma\max}$ be defined as $P_{\sigma\max} \equiv \max_k \|P_\sigma(k)\|$, where $P_\sigma(k) = \partial y/\partial \sigma_{ij}$. The convergence is guaranteed if η_m and η_σ are chosen as $\eta_m = \eta_\sigma = [\eta_w/M]\left[|w_{kj}|_{\max}\left(2/|\sigma_{ij}|_{\min}\right)\right]^{-2}$, in which $|w_{kj}|_{\max} = \max_k |w_{kj}(k)|$, $|\sigma_{ij}|_{\min} = \min_k |\sigma_{ij}(k)|$, and $|\cdot|$ is the absolute value.

Proof of Theorem B2: According to Lemma B1, $|[(x_i - m_{ij})/\sigma_{ij}]\exp\{-[(x_i - m_{ij})/\sigma_{ij}]^2\}| < 1$. The upper bounds on $P_m(k)$ can be derived as

$$\begin{aligned} P_m(k) &= \frac{\partial y}{\partial m_{ij}} \\ &= \left(\frac{\partial y}{\partial u_j^{(4)}}\right)\left(\frac{\partial u_j^{(4)}}{\partial u_j^{(3)}}\right)\left(\frac{\partial u_j^{(3)}}{\partial u_{ij}^{(2)}}\right)\left(\frac{\partial u_{ij}^{(2)}}{\partial m_{ij}}\right) \\ &< \left|\sum_{j=1}^{R}\sum_{k=1}^{M} w_{kj}\phi_k\right|\left|\left(\frac{2}{\sigma_{ij}}\right)\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)\right. \\ &\quad \left.\times \exp\left[-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right]\right| \\ &< \left|\sum_{j=1}^{R}\sum_{k=1}^{M} w_{kj}\phi_k\right|\left|\frac{2}{\sigma_{ij}}\right| \\ &< \sqrt{RM}\,|w_{kj}|_{\max}\left(\frac{2}{|\sigma_{ij}|_{\min}}\right) \end{aligned} \tag{B9}$$

where $\phi_k \in [0,1]$, for $k = 1,\ 2,\ \ldots,\ M$. Thus,

$$\|P_m(k)\| < \sqrt{RM}\,|w_{kj}|_{\max}\left(\frac{2}{|\sigma_{ij}|_{\min}}\right). \tag{B10}$$

The error difference can also be represented as [16]

$$\begin{aligned} e(k+1) &= e(k) + \Delta e(k) \\ &= e(k) + \left[\frac{\partial e(k)}{\partial m_{ij}}\right]^T \Delta m_{ij} \end{aligned} \tag{B11}$$

where Δm_{ij} represents the change of the mean of the Gaussian function in the membership function layer. Equation (18) and (B11) yield

$$\frac{\partial e(k)}{\partial m_{ij}} = \frac{\partial e(k)}{\partial y}\frac{\partial y}{\partial m_{ij}} = P_m(k) \tag{B12}$$

$$e(k+1) = e(k) - P_m^T(k)\eta_m e(k) P_m(k). \tag{B13}$$

Then,

$$\begin{aligned} \|e(k+1)\| &= \left\|e(k)\left[1 - \eta_m P_m^T(k)P_m(k)\right]\right\| \\ &\le \|e(k)\| \cdot \left\|1 - \eta_m P_m^T(k)P_m(k)\right\| \end{aligned} \tag{B14}$$

is true. If $\eta_m = \lambda/(P_{m\max})^2 = [\eta_w/M][|w_{kj}|_{\max}(2/|\sigma_{ij}|_{\min})]^{-2}$ is chosen, then the term $\left\|1 - \eta_m P_m^T(k)P_m(k)\right\|$ in (B14) is less than 1. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ given by (B3) and (B4) is guaranteed. The output error between the reference model and actual plant converges to zero as $t \to \infty$.

According to Lemma B2, $|[(x_i - m_{ij})/\sigma_{ij}]^2\ \exp\{-[(x_i - m_{ij})/\sigma_{ij}]^2\}| < 1$. The upper bounds on $P_\sigma(k)$ can be derived as

$$\begin{aligned} P_\sigma(k) &= \frac{\partial y}{\partial \sigma_{ij}} \\ &= \left(\frac{\partial y}{\partial u_j^{(4)}}\right)\left(\frac{\partial u_j^{(4)}}{\partial u_j^{(3)}}\right)\left(\frac{\partial u_j^{(3)}}{\partial u_{ij}^{(2)}}\right)\left(\frac{\partial u_{ij}^{(2)}}{\partial \sigma_{ij}}\right) \\ &< \left|\sum_{j=1}^{R}\sum_{k=1}^{M} w_{kj}\phi_k\right|\left|\left(\frac{2}{\sigma_{ij}}\right)\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right. \\ &\quad \left.\times \exp\left[-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right]\right| \\ &< \left|\sum_{j=1}^{R}\sum_{k=1}^{M} w_{kj}\phi_k\right|\left|\frac{2}{\sigma_{ij}}\right| \\ &< \sqrt{RM}\,|w_{kj}|_{\max}\left(\frac{2}{|\sigma_{ij}|_{\min}}\right) \end{aligned} \tag{B15}$$

where $\phi_k \in [0,1]$, for $k = 1,\ 2,\ \ldots,\ M$. Thus,

$$\|P_\sigma(k)\| < \sqrt{RM}\,|w_{kj}|_{\max}\left(\frac{2}{|\sigma_{ij}|_{\min}}\right). \tag{B16}$$

The error difference can be represented as

$$\begin{aligned} e(k+1) &= e(k) + \Delta e(k) \\ &= e(k) + \left[\frac{\partial e(k)}{\partial \sigma_{ij}}\right]^T \Delta\sigma_{ij} \end{aligned} \tag{B17}$$

where $\Delta\sigma_{ij}$ represents the change of the variance of the Gaussian function in the membership function layer. Equations (19) and (B17) yield

$$\frac{\partial e(k)}{\partial \sigma_{ij}} = \frac{\partial e(k)}{\partial y}\frac{\partial y}{\partial \sigma_{ij}} = P_\sigma(k) \tag{B18}$$

$$e(k+1) = e(k) - P_\sigma^T(k)\eta_\sigma e(k) P_\sigma(k). \tag{B19}$$

Then,

$$\begin{aligned} \|e(k+1)\| &= \left\|e(k)\left[1 - \eta_\sigma P_\sigma^T(k)P_\sigma(k)\right]\right\| \\ &\le \|e(k)\| \cdot \left\|1 - \eta_\sigma P_\sigma^T(k)P_\sigma(k)\right\| \end{aligned} \tag{B20}$$

is true. If $\eta_\sigma = \lambda/(P_{\sigma\max})^2 = [\eta_w/M][|w_{kj}|_{\max}(2/|\sigma_{ij}|_{\min})]^{-2}$ is chosen, then the term $\left\|1 - \eta_\sigma P_\sigma^T(k)P_\sigma(k)\right\|$ in (B20) is less than 1. Therefore, the Lyapunov stability of $V > 0$ and $\Delta V < 0$ given by (B3) and (B4) is guaranteed. The output error between the reference model and actual plant converges to zero as $t \to \infty$. This fact completes the proof of the theorem.

REFERENCES

[1] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Parts I and II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 404–435, Mar. 1990.

[2] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[3] K. J. Astrom and B. Wittenmark, *Adaptive Control*. Reading, MA: Addison-Wesley, 1989.

[4] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[5] S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: Survey in soft computing framework," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 748–768, May 2000.

[6] F. Sun, Z. Sun, L. Li, and H. X. Li, "Neuro-fuzzy adaptive control based on dynamic inversion for robotic manipulators," *Fuzzy Sets Syst.*, vol. 134, no. 1, pp. 117–133, Feb. 2003.

[7] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.

[8] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, Nov. 1997.

[9] W. S. Lin, C. H. Tsai, and J. S. Liu, "Robust neuro-fuzzy control of multivariable systems by tuning consequent membership functions," *Fuzzy Sets Syst.*, vol. 124, no. 2, pp. 181–195, Dec. 2001.

[10] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985.

[11] C. Li and C. Y. Lee, "Self-organizing neuro-fuzzy system for control of unknown plants," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 1, pp. 135–150, Feb. 2003.

[12] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.

[13] C. F. Juang and C. T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–31, Feb. 1998.

[14] H. Takagi, N. Suzuki, T. Koda, and Y. Kojima, "Neural networks designed on approximate reasoning architecture and their application," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 752–759, Sep. 1992.

[15] E. Mizutani and J.-S. R. Jang, "Coactive neural fuzzy modeling," in *Proc. Int. Conf. Neural Netw.*, Perth, WA, Australia, 1995, vol. 2, pp. 760–765.

[16] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[17] J. C. Patra, R. N. Pal, B. N. Chatterji, and G. Panda, "Identification of nonlinear dynamic systems using functional link artificial neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, no. 2, pp. 254–262, Apr. 1999.

[18] Y. H. Pao, S. M. Phillips, and D. J. Sobajic, "Neural-net computing and intelligent control systems," *Int. J. Control*, vol. 56, no. 2, pp. 263–289, 1992.

[19] J. C. Patra and R. N. Pal, "A functional link artificial neural network for adaptive channel equalization," *Signal Process.*, vol. 43, pp. 181–195, May 1995.

[20] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.

[21] L. X. Wang and J. M. Mendel, "Fuzzy adaptive filters, with application to nonlinear channel equalization," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 3, pp. 161–170, Aug. 1993.

[22] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 144–156, Jan. 1995.

[23] Y. C. Chen and C. C. Teng, "A model reference control structure using a fuzzy neural network," *Fuzzy Sets Syst.*, vol. 73, pp. 291–312, 1995.

[24] J. Tanomaru and S. Omatu, "Process control by online trained neural controllers," *IEEE Trans. Ind. Electron.*, vol. 39, no. 6, pp. 511–521, Dec. 1992.

[25] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear control via approximate input–output linearization: The ball and beam example," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 392–398, Mar. 1992.

[26] D. Psaltis, A. Sideris, and A. Yamamura, "A multilayered neural network controller," *IEEE Control Syst.*, vol. 8, no. 2, pp. 17–21, Apr. 1988.

[27] C. L. Phillips and H. T. Nagle, *Digital Control System Analysis and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[28] C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 290–302, Apr. 2000.

[29] C. J. Lin, H. C. Chuang, and Y. J. Xu, "Face detection in color images using efficient genetic algorithms," *Opt. Eng.*, vol. 45, no. 4, pp. 047201-1–047201-12, Apr. 2006.

[30] C. J. Lin and W. H. Ho, "An asymmetry-similarity-measure-based neural fuzzy inference system," *Fuzzy Sets Syst.*, vol. 152, no. 3, pp. 535–551, Jun. 2005.

[31] R. Rumelhart, G. Hinton, and J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D. Rumelhart and J. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[32] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *Proc. IEEE*, vol. 139, no. 3, pp. 301–310, Jun. 1992.

[33] M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

Cheng-Hung Chen (S'07) was born in Kaohsiung, Taiwan, R.O.C., in 1979. He received the B.S. and M.S. degrees in computer science and information engineering from Chaoyang University of Technology, Taichung, Taiwan, in 2002 and 2004, respectively. He is currently working toward the Ph.D. degree in electrical and control engineering at the National Chiao-Tung University, Hsinchu, Taiwan.

His current research interests include neural networks, fuzzy systems, evolutionary algorithms, intelligent control, and pattern recognition.

Cheng-Jian Lin (S'93–M'95) received the B.S. degree in electrical engineering from Ta-Tung University, Taipei, Taiwan, R.O.C., in 1986, and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1991 and 1996, respectively.

From April 1996 to July 1999, he was an Associate Professor in the Department of Electronic Engineering, Nan-Kai College, Nantou, Taiwan. From August 1999 to January 2005, he was an Associate Professor in the Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taiwan, where during 2001–2005, he was the Chairman, and from February 2005 to July 2007, he was a Full Professor. During 2005–2007, he was the Library Director of Poding Memorial Library, Chaoyang University of Technology. He is currently a Full Professor in the Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung, Taiwan. From 2002 to 2005, he was an Associate Editor of the *International Journal of Applied Science and Engineering*. His current research interests include soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and field-programmable gate array (FPGA) design.

Dr. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the Institute of Electronics, Information, and Communication Engineers (IEICE), and the IEEE Computational Intelligence Society. He is an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI).

Chin-Teng Lin (S'88–M'91–SM'99–F'05) received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, R.O.C., in 1996, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

From 2005 to 2007, he was the Founding Dean of the Computer Science College, NCTU, where, since August 1992, he has been with the College of Electrical Engineering and Computer Science, NCTU, and is currently a Provost of academic affairs and the Chair Professor of electrical and control engineering. He is the author or coauthor of more than 110 papers published in international journals, including about 80 IEEE transaction papers, and the books *Neural Fuzzy Systems* (Prentice Hall, 1996) and *Neural Fuzzy Control Systems With Structure and Parameter Learning* (World Scientific, 1994). His current research interests include intelligent technology, soft computing, brain–computer interface, intelligent transportation systems, robotics and intelligent sensing, and nanobioinformation technologies and cognitive science (NBIC).

Dr. Lin is a member of the Tau Beta Pi, the Eta Kappa Nu, and the Phi Kappa Phi honorary societies. He was a member of the Board of Governors (BoG) of the IEEE Systems, Man, and Cybernetics Society (SMCS) during 2003–2005, and is currently the BoG member of the IEEE Circuits and Systems Society (CASS). From 2003 to 2005, he was the IEEE Distinguished Lecturer. He is currently the Deputy Editor-in-Chief (EIC) of the IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS—PART II. He was the Program Chair of the 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei. From 2004 to 2005, he was the President of the Board of Government of the Asia Pacific Neural Networks Assembly (APNNA). He has been the recipient of the Outstanding Research Award by the National Science Council (NSC), Taiwan, since 1997. He has also received the Outstanding Professor Award from the Chinese Institute of Engineering (CIE) in 2000 and the 2002 Taiwan Outstanding Information-Technology Expert Award. He was elected to be one of 38th Ten Outstanding Rising Stars in Taiwan in 2000.