Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks

Ching-Hung Lee and Ching-Cheng Teng

***Abstract*—This paper proposes a recurrent fuzzy neural network (RFNN) structure for identifying and controlling nonlinear dynamic systems. The RFNN is inherently a recurrent multilayered connectionist network for realizing fuzzy inference using dynamic fuzzy rules. Temporal relations are embedded in the network by adding feedback connections in the second layer of the fuzzy neural network (FNN). The RFNN expands the basic ability of the FNN to cope with temporal problems. In addition, results for the FNN-fuzzy inference engine, universal approximation, and convergence analysis are extended to the RFNN. For the control problem, we present the direct and indirect adaptive control approaches using the RFNN. Based on the Lyapunov stability approach, rigorous proofs are presented to guarantee the convergence of the RFNN by choosing appropriate learning rates. Finally, the RFNN is applied in several simulations (time series prediction, identification, and control of nonlinear systems). The results confirm the effectiveness of the RFNN.**

***Index Terms*—Control, fuzzy logic, fuzzy neural network (FNN), identification, neural network.**

I. Introduction

RECENTLY, feedforward neural networks have been shown to obtain successful results in system identification and control [10]. Such neural networks are static input/output mapping schemes that can approximate a continuous function to an arbitrary degree of accuracy. Results have also been extended to recurrent neural networks [6]–[8]. For example, Jin *et al.* [7] studied the approximation of continuous-time dynamic systems using the dynamic recurrent neural network (DRNN) and a Hopfield-type DRNN was presented by Funahashi and Nakamura [6]. Recurrent neural network systems learn and memorize information implicitly with weights embedded in them.

As is widely known, both fuzzy logic systems and neural network systems are aimed at exploiting human-like knowledge processing capability. Moreover, combinations of the two have found extensive applications. This approach involves merging or fusing fuzzy systems and neural networks into an integrated system to reap the benefits of both. For instance, Lin and Lee [9] proposed a general neural network model for a fuzzy logic control and decision system, which is trained to control an unmanned vehicle. In previous literature, we presented a fuzzy neural network (FNN) to establish a model reference control structure and verified that our FNN is a universal approximator [4], [5]. The design process for the FNN in [4] and [5] combined tapped delays with the backpropagation (BP) algorithm to solve the dynamic mapping problems. However, a major drawback of the FNN is that its application domain is limited to static problems due to its feedforward network structure. Processing temporal problems using the FNN is inefficient. Hence, we propose a recurrent fuzzy neural network (RFNN) based on supervised learning, which is a dynamic mapping network and is more suitable for describing dynamic systems than the FNN. Of particular interest is that it can deal with time-varying input or output through its own natural temporal operation [16]. For this ability to temporarily store information, the structure of the network is simplified. That is, fewer nodes are required for system identification.

In this paper, the proposed RFNN, which is a modified version of the FNN, is used to identify and control a nonlinear dynamic system. The RFNN is a recurrent multilayered connectionist network for realizing fuzzy inference and can be constructed from a set of fuzzy rules. The temporal relations embedded in the RFNN are developed by adding feedback connections in the second layer of the FNN. This modification provides the memory elements of the RFNN and expands the basic ability of the FNN to include temporal problems. Since a recurrent neuron has an internal feedback loop, it captures the dynamic response of a system, thus the network model can be simplified. We show that all the characteristics of the FNN—fuzzy inference, universal approximation, and convergence properties—are extended to the RFNN. We also study the proposed RFNNs approximation and dynamics mapping abilities. For the control problem, we present the direct and indirect adaptive control approaches using the RFNN. In addition, to guarantee the convergence of the RFNN, the Lyapunov stability approach is applied to select appropriate learning rates. Finally, the proposed RFNN is applied to some numerical examples: time sequence prediction, identification of nonlinear systems without tapped delays, identification of a chaotic system, and adaptive control of a nonlinear system.

The paper is organized as follows. In Section II, an RFNN structure is developed and the universal approximation of the RFNN is studied. The comparison between the FNN and the RFNN is also described. The training architectures for identification and control and the learning algorithm are presented in Section III. Section IV presents the stability analysis of the RFNN, which is based on the Lyapunov approach to show the convergence of the RFNN. Simulation results are discussed in Section V. Section VI gives the conclusion of this paper. Note

Manuscript received December 2, 1999; revised March 16, 2000. This work was supported by the National Science Council, Taiwan, R.O.C., under contract NSC 89-2213-E009-126.

C.-H. Lee is with the Department of Electronic Engineering, Lunghwa Institute of Technology, Taoyuan 333, Taiwan, R.O.C.

C.-C. Teng is with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C.

Publisher Item Identifier S 1063-6706(00)06585-1.

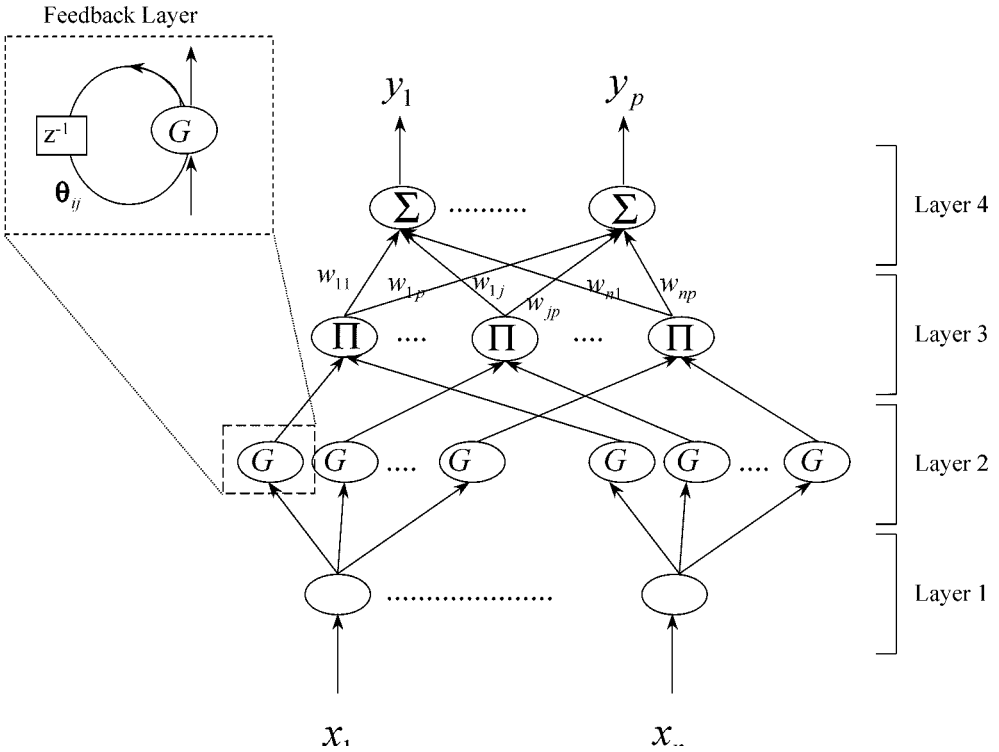

Fig. 1. The configuration of the proposed RFNN.

that all proof of theorems and lemmas are presented in Appendix.

II. Recurrent Fuzzy Neural Networks (RFNNs)

In this section, the proposed recurrent fuzzy neural network (RFNN) is presented to show that the RFNN is a system generalized from the FNN. The key aspects of the RFNN—dynamic mapping capability, temporal information storage, universal approximation, and the fuzzy inference system—are discussed here. The RFNN will be shown to possess the same advantages over recurrent neural networks [8] and extend the application domain of the FNN to temporal problems.

A. Structure of the RFNN

This section presents a fuzzy inference system implemented by using a multilayer recurrent neural network, called a RFNN. A schematic diagram of the proposed RFNN structure is shown in Fig. 1, which is organized into n input variables, m-term nodes for each input variable, p output nodes, and $m \times n$ rule nodes. This RFNN system thus consists of four layers and $n + (n \times m) + m + p$ nodes, where m denotes the rule number. Layer 1 accepts input variables. Its nodes represent input linguistic variables. Layer 2 is used to calculate Gaussian membership values. Nodes in this layer represent the terms of the respective linguistic variables. Nodes at layer 3 represent fuzzy rules. Layer 3 forms the fuzzy rule base. Links before layer 3 represent the preconditions of the rules, and the links after layer 3 represent the consequences of the rule nodes. Layer 4 is the output layer, where each node is for an individual output of the system. The links between layer 3 and layer 4 are connected by the weighting values w_p^i.

B. Layered Operation of the RFNN

Next, we shall indicate the signal propagation and the operation functions of the nodes in each layer. In the following description, u_i^k denotes the ith input of a node in the kth layer; O_i^k denotes the ith node output in layer k.

Layer 1: Input Layer: The nodes in this layer only transmit input values to the next layer directly, i.e.,

$$O_i^1 = u_i^1. \tag{1}$$

From this equation, the link weight at layer 1 (w_i^1) is unity.

Layer 2: Membership Layer: In this layer, each node performs a membership function and acts as a unit of memory. The Gaussian function is adopted here as a membership function. Thus, we have

$$O_{ij}^2 = \exp\left\{-\frac{(u_{ij}^2 - m_{ij})^2}{(\sigma_{ij})^2}\right\} \tag{2}$$

where m_{ij} and σ_{ij} are the center (or mean) and the width (or standard deviation—STD) of the Gaussian membership function. The subscript ij indicates the jth term of the ith input x_i. In addition, the inputs of this layer for discrete time k can be denoted by

$$u_{ij}^2(k) = O_i^1(k) + O_{ij}^f(k) \tag{3}$$

where $O_{ij}^f(k) = O_{ij}^2(k-1) \cdot \theta_{ij}$ and θ_{ij} denotes the link weight of the feedback unit. It is clear that the input of this layer contains the memory terms $O_{ij}^2(k-1)$, which store the past information of the network. This is the apparent difference between the FNN and RFNN. Each node in this layer has three adjustable parameters: m_{ij}, σ_{ij}, and θ_{ij}.

Layer 3: Rule Layer: The nodes in this layer are called rule nodes. The following AND operation is applied to each rule node to integrate these fan-in values, i.e.,

$$O_i^3 = \prod_i u_i^3 = \exp\{-[\mathbf{D}_i(\mathbf{u}_i^2 - \mathbf{m}_i)]^T[\mathbf{D}_i(\mathbf{u}_i^2 - \mathbf{m}_i)]\} \quad (4)$$

where $\mathbf{D}_i = \text{diag}[1/\sigma_{1i}, 1/\sigma_{2i}, \cdots, 1/\sigma_{ni}]$, $\mathbf{u}_i = [u_{1i}, u_{2i}, \cdots, u_{ni}]^T$, and $\mathbf{m}_i = [m_{1i}, m_{2i}, \cdots, m_{ni}]^T$. The output O_i^3 of a rule node represents the "firing strength" of its corresponding rule.

Layer 4: Output Layer: Each node in this layer is called an output linguistic node. This layer performs the defuzzification operation. The node output is a linear combination of the consequences obtained from each rule. That is

$$y_j = O_j^4 = \sum_i^m u_{ij}^4 w_i^4 \quad (5)$$

where $u_i^4 = O_i^3$ and w_{ij}^4 (the link weight) is the output action strength of the jth output associated with the ith rule. The w_{ij}^4 are the tuning factors of this layer.

Finally, the overall representation of input x and the mth output y is

$$\begin{aligned} y_m(k) &= O_m^4(k) \\ &= \sum_{j=1}^m w_{mj} \prod_{i=1}^n \\ &\quad \cdot \exp\left[-\frac{[x_i(k) + O_{ij}^2(k-1)\cdot\theta_{ij} - m_{ij}]^2}{(\sigma_{ij})^2}\right] \end{aligned} \quad (6)$$

where m_{ij}, σ_{ij}, θ_{ij}, and w_{mj} are the tuning parameters and $O_{ij}^2(k-1) = \exp\{-[x(k-1) + O_{ij}^2(k-2)\cdot\theta_{ij} - m_{ij}]^2/(\sigma_{ij})^2\}$. Obviously, using the RFNN, the same inputs at different times yield different outputs. As above, the number of tuning parameters for the RFNN is $(n \times m \times 3) + (m \times p)$.

Recall that the FNN, proposed in [4], has the following input/output representation:

$$\begin{aligned} y_m(k) = O_m^4(k) &= \sum_{j=1}^m w_{mj} \prod_{i=1}^n \\ &\quad \cdot \exp\left[-\frac{(x_i(k) - m_{ij})^2}{(\sigma_{ij})^2}\right]. \end{aligned} \quad (7)$$

Clearly, the RFNN features dynamic mapping with feedback and more tuning parameters than the FNN. In the above formulas, note that if the weights in the feedback unit θ_{ij} are all equal to zero, then the RFNN reduces to an FNN.

The proposed RFNN can be shown to be a universal uniform approximator for continuous functions over compact sets if it satisfies a certain condition. The condition is described as

$$\begin{aligned} &\prod_{i=1}^n \frac{(O_{ij}^2\theta_{ij})^2}{(\sigma_{ij})^2} \prod_{i=1}^n \frac{(O_{ik}^2\theta_{ik})^2}{(\sigma_{ik})^2} \\ &\quad \neq \prod_{i=1}^n \frac{(m_{ik} + O_{ij}^2\theta_{ij} - m_{ij})^2}{(\sigma_{ij})^2} \\ &\quad\quad \cdot \prod_{i=1}^n \frac{(m_{ij} + O_{ik}^2\theta_{ik} - m_{ik})^2}{(\sigma_{ik})^2} \\ &\quad\quad\quad \text{for all } j \neq k. \end{aligned} \quad (8)$$

Then we have the following result.

Theorem 1: Universal approximation theorem—for any real function h: $\mathbb{R}^n \to \mathbb{R}^m$ which is continuous on a compact set $K \subset \mathbb{R}^n$ and for any given $\epsilon > 0$ there is an RFNN system f that satisfies condition (8), such that

$$\sup_{x \in K} \|f(x) - h(x)\| < \epsilon.$$

Here $\|\cdot\|$ can be any norm.

This theorem shows that if the RFNN has a sufficiently large number of fuzzy logical rules (or neurons), then it can approximate any continuous function in $C(\mathbb{R}^n)$ over a compact subset of $\mathbb{R}^n$. For system identification, the theorem means that for any given continuous output trajectory $y(t)$ of any nonlinear dynamic system over any compact time-interval $t \in [t_0, T]$, the output $\hat{y}(t)$ of the RFNN can approximate $y(t)$ uniformly with arbitrarily high precision.

C. Fuzzy Reasoning

For a multi-input single-output RFNN system, let x_i be the ith input linguistic variable and define α_j as the firing strength of rule j, which is obtained by the product of the grades of the membership functions $\mu_{A_{ij}}(x_i)$ in the antecedent. If w_j represents the jth consequence link weight, the inferred value y^* is then obtained by taking the weighted sum of its inputs, i.e., $\sum_j w_j\alpha_j$. This is the so-called area defuzzification process.

The proposed RFNN realizes fuzzy inference as follows:

$$R^j\text{: IF } u_{1j} \text{ is } A_{1j}, \cdots, u_{nj} \text{ is } A_{nj}, \text{then } y = w_j$$

where for $i = 1, \cdots, n$ $u_{ij} = x_i + O_{ij}^2(k-1)\cdot\theta_{ij}$, A_{1j}, A_{nj} are fuzzy sets, w_j is a fuzzy singleton, and n is the number of inputs. That is, the input of each membership function is the network input x_i plus the temporal term $O_{ij}^2\theta_{ij}$. Therefore, a connection structure based on the fuzzy rule can be illustrated as in Fig. 2. This fuzzy system, with its memory terms (feedback units), can be considered a *dynamic fuzzy inference system* and the inferred value is given by

$$y^* = \sum_{j=1}^m \alpha_j w_j$$

where $\alpha_j = \prod_{i=1}^n \mu_{A_{ij}}(u_{ij})$.

From the above description, it is clear that the RFNN is a fuzzy logic system with memory elements. Given that the tuning

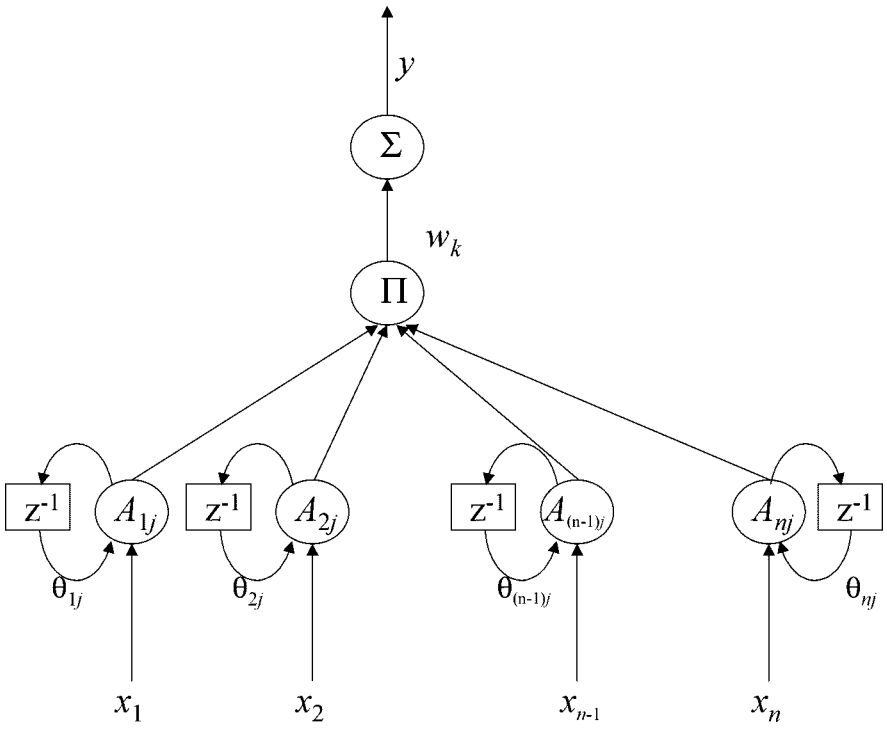

Fig. 2. A connection structure based on the jth fuzzy rule.

parameters of a fuzzy system have clear physical meanings, the memory terms make it possible to incorporate *a priori* knowledge in the selection of initial parameter values and constraints among parameter values. Note that the parameters θ of the feedback units are not set from the human knowledge. According to the requirements of the system, they will be given proper values representing the memorized information.

For initializing system parameters, the on-line method also can be used [1] in the RFNN. The rule number and the initial value of tuning parameters m, σ, w, θ are given, where $\theta_{ij} = 0$ for all i, j. That is, there are no feedback units initially. As for parameter learning, we will develop a recursive learning algorithm based on the gradient method.

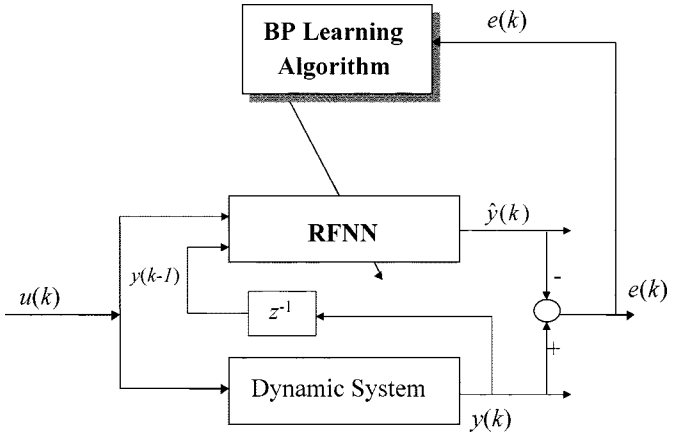

Fig. 3. Dynamical modeling of nonlinear systems using the RFNN.

III. Training for the RFNNs

Training architectures for identification and adaptive control, and a learning algorithm based on the gradient method are presented below.

A. Training Architecture of the RFNNs

1) Architecture for identification: To identify a nonlinear dynamic system, prior studies [4], [5], [10] used the series-parallel model with tapped delay units for training networks. In this paper, we adopt this model to train the RFNN with some modification. The current input and the most recent output of the system are fed into the RFNN and the error $e(k)$ between the actual system output and the RFNN is used to train the RFNN. The modified training model is shown in Fig. 3. The RFNN output will estimate the output trajectories of the nonlinear system. Note that only $y(k-1)$ and $u(k)$ are fed into the identification model even though the system output $y(k+1)$ depends both on its past values $y(k-i)$, $i = 0, 1, \cdots$ as well as the past values of inputs $u(k-j)$, $j = 0, 1, \cdots$. This simplifies the network structure, i.e., reduces the number of neurons.

2) Architecture for control: For system control problems, we focus on the adaptive control of dynamic systems using the RFNN. In [10], Narendra and Parthasarathy used two distinct neural networks to control systems adaptively by direct and indirect control. Subsequently, Chen and Teng and Ku and Lee, proposed control architecture for the model reference adaptive control (MRAC) problem by using FNNs [4], [5], and diagonal neural networks [8]. Indirect control architecture usually requires an identified system model and the controller design is based on the learning algorithm [see Fig. 4(a) and (b)]. Here, we present the direct adaptive control approach using the RFNN. Fig. 4(c) illustrates the block diagram of the RFNN-based control system. Obviously, the inputs of the RFNN are the reference

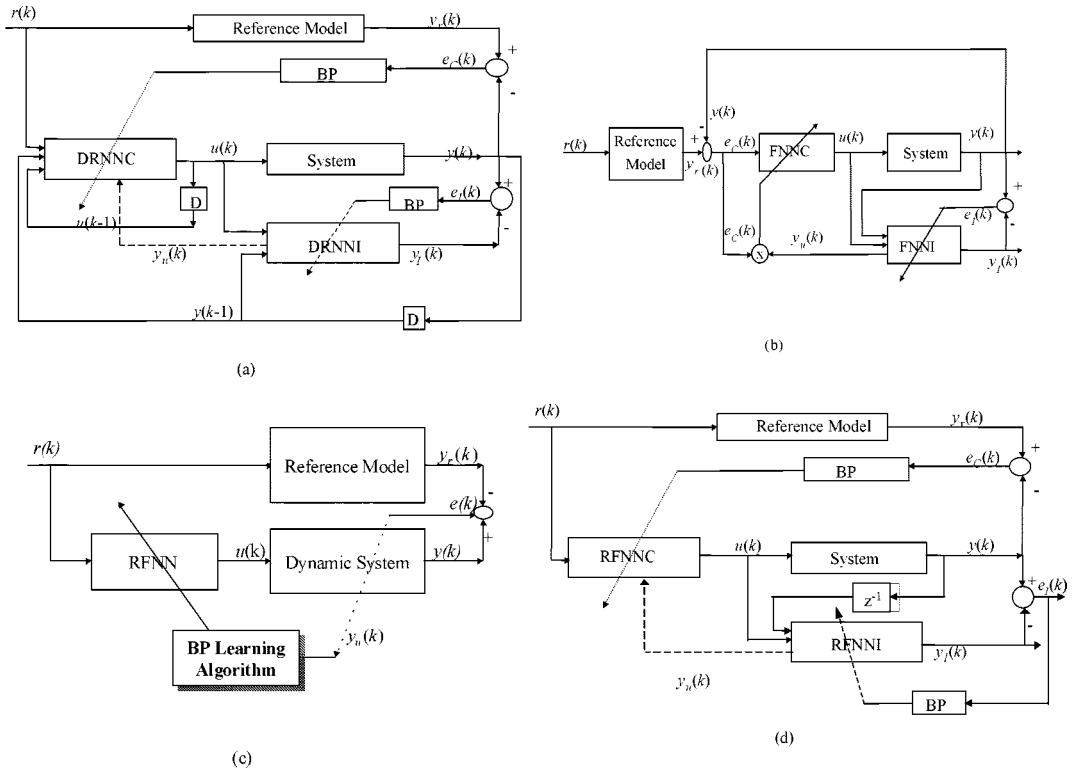

Fig. 4. Block diagram of RFNN-based control system. (a) Indirect control architecture in [4]. (b) Indirect control architecture in [7]. (c) Direct adaptive control architecture using the RFNN. (d) Indirect control architecture using RFNNs.

input, the previous plant output, and the previous control signal, while the output of the RFNN is the control signal.

Remark 1: The RFNN can also be applied to construct the above indirect control architectures. The identifier and controller can be replaced by the RFNN and the tapped-delay unit can be removed [see Fig. 4(d)]. In indirect control, Fig. 4(a), (b), and (d), an unknown system is identified by the identifier (FNNI, DRNNI, or RFNNI), which provides the information about the system to the neural controller (FNNC, DRNNC, or RFNNC). The neural controller generates a control signal to drive the unknown system such that the error between actual system output and desired output is minimized. Herein, both identifier and controller networks are the same network structure (see [4], [5], and [8]).

B. Learning Algorithm

Consider the single-output case for simplicity. Our goal is to minimize the following cost function:

$$E(k) = \tfrac{1}{2}(y(k) - \hat{y}(k))^2 = \tfrac{1}{2} \sum_j (y(k) - O^4(k))^2 \quad (9)$$

where $y(k)$ is the desired output and $\hat{y}(k) = O^4(k)$ is the current output for each discrete time k. In each training cycle, starting at the input nodes, a forward pass is used to compute the activity of all the nodes in the current output $\hat{y}(k)$.

By using the BP learning algorithm, the weighting vector of the RFNN is adjusted such that the error defined in (9) is less than a desired threshold value after a given number of training cycles. The well-known BP algorithm may be written briefly as

$$W(k+1) = W(k) + \Delta W(k) = W(k) + \eta \left(-\frac{\partial E(k)}{\partial W} \right) \quad (10)$$

where, in this case, η and W represent the learning rate and tuning parameters of the RFNN. Let $e(k) = y(k) - \hat{y}(k)$ and $W = [m, \sigma, \theta, w]^T$ be the training error and weighting vector of the RFNN, then the gradient of error $E(\cdot)$ in (9) with respect to an arbitrary weighting vector W is

$$\frac{\partial E(k)}{\partial W} = -e(k) \frac{\partial \hat{y}(k)}{\partial W} = -e(k) \frac{\partial O^4(k)}{\partial W}. \quad (11)$$

By recursive applications of the chain rule, the error term for each layer is first calculated, then the parameters in the corresponding layers are adjusted. With the RFNN (6) and the cost function defined in (9), derive the update rule of w_{ij}

$$w_{ij}(k+1) = w_{ij}(k) - \eta^w \frac{\partial E(k)}{\partial w_{ij}} \quad (12)$$

where

$$\frac{\partial E(k)}{\partial w_{ij}} = -e(k) \cdot O_i^3.$$

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

$$m_{ij}(k+1) = m_{ij}(k) - \eta^m \frac{\partial E(k)}{\partial m_{ij}} \quad (13)$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) - \eta^\sigma \frac{\partial E(k)}{\partial \sigma_{ij}} \quad (14)$$

$$\theta_{ij}(k+1) = \theta_{ij}(k) - \eta^\theta \frac{\partial E(k)}{\partial \theta_{ij}} \quad (15)$$

where

$$\frac{\partial E(k)}{\partial m_{ij}} = -\sum_k e(k)w_{ik}\cdot O_k^3 \cdot \frac{2(x_i + O_{ij}^2(k-1)\cdot\theta_{ij} - m_{ij})}{(\sigma_{ij})^2}$$

$$\frac{\partial E(k)}{\partial \sigma_{ij}} = -\sum_k e(k)w_{ik}\cdot O_k^3 \cdot \frac{2(x_i + O_{ij}^2(k-1)\cdot\theta_{ij} - m_{ij})^2}{(\sigma_{ij})^3}$$

$$\frac{\partial E(k)}{\partial \theta_{ij}} = -\sum_k e(k)w_{ik}\cdot O_k^3 \cdot \frac{-2(x_i + O_{ij}^2(k-1)\cdot\theta_{ij} - m_{ij})O_{ij}^2(k-1)}{(\sigma_{ij})^2}.$$

Finally, we have to check (8). In general, this condition usually holds. If (8) does not hold for some j, k, a small value ϵ must be added to the STD value σ such that (8) holds, where $||\epsilon|| \ll 1$. This completes the derivation of the BP learning algorithm.

The BP algorithm is a widely used algorithm for training multilayer networks by means of error propagation via variational calculus [11]. But its success depends upon the quality of the training data. In [2], Chen and Jain proposed a robust BP learning algorithm that was stable under small noise perturbation and robust against gross errors. Since our intent here is to emphasize the universal approximation and dynamic mapping abilities of the proposed RFNN, the effect of the BP algorithm is neglected. Other existing on-line learning algorithms for tuning the weights of recurrent neural networks can also be adopted for tuning the RFNN.

Remark 2: When the control architecture shown in Fig. 4(c) and (d) is used, we must pay attention to the training of the RFNNC. Similarly, let us define the cost function $E_C(k) = \frac{1}{2}[y_r(k) - y(k)]^2 = \frac{1}{2}[e_C(k)]^2$. Next, the gradient of E_C is

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

where $O_C(k)$ is the output of the RFNN for control and $y_u(k) = \partial y(k)/\partial u(k)$ denotes the system sensitivity. Thus, the parameters of the RFNNC can be also adjusted by (10).

Remark 3: Note that the convergence of the RFNN cannot be guaranteed if the system sensitivity is unknown. For an unknown system, we must adopt the indirect control architecture in Fig. 4(d). Obviously, the identifier (RFNNI) can provide the system sensitivity and it can be computed by the chain rule

$$\begin{aligned}\frac{\partial y_j(k)}{\partial u_i(k)} &= \frac{\partial O_{Ij}^4}{\partial u_i} = \sum_{a=1}^{R_I}\left\{\frac{\partial O_{Ij}^4}{\partial O_{Ia}^3}\cdot\frac{\partial O_{Ia}^3}{\partial u_i}\right\}\\ &= \sum_{a=1}^{R_I} w_{Iaj}\cdot\left\{\frac{\partial O_{Ia}^3}{\partial u_i}\right\}\\ &= \sum_{a=1}^{R_I} w_{Iaj}\cdot\left\{\sum_{k=1}^{N_{I_k}}\frac{\partial O_{Ia}^3}{\partial O_{Ika}^2}\cdot\frac{\partial O_{Ika}^2}{\partial u_i}\right\}\\ &= \sum_{a=1}^{R_I} w_{Iaj}\cdot\left\{\sum_{k=1}^{N_{I_k}}\frac{O_{Ia}^3}{O_{Ika}^2}\cdot\frac{\partial O_{Ika}^2}{\partial u_i}\right\}\\ &= \sum_{a=1}^{R_I} w_{Iaj}\cdot\left\{\sum_{k=1}^{N_{I_k}}\frac{O_{Ia}^3}{O_{Ika}^2}\cdot(-2)\cdot\frac{(u_i + O_{Ika}^2(k-1)\cdot\theta_{Ika} - m_{Ika})}{(\sigma_{Ika})^2}\right\}\end{aligned}$$

where m_{Ika} and σ_{Ika} are, respectively, the center and the width of the Gaussian function in the kth term of the ith input linguistic variable u_i. The superscripts denote the layer numbers. The link weight w_{Iaj} is the output action strength of the jth output associated with the ath rule. N_{I_k} is the number of fuzzy sets of the ith input linguistic variable u_i, which satisfies $\prod_{k=1}^{n} N_{I_k} = R_I$. Finally, R_I is the number of rules in the RFNNI.

IV. Stability Analysis of the RFNN

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

A. Stability Analysis for Identification

First, define a discrete Lyapunov function as follows:

$$V_I(k) = E_I(k) = \tfrac{1}{2}[e_I(k)]^2 \tag{16}$$

where $e_I(k)$ represents the identification error in the learning process. The change of the Lyapunov function due to the training process is thus

$$\Delta V_I(k) = V_I(k+1) - V_I(k) = \tfrac{1}{2}[e_I^2(k+1) - e_I^2(k)]. \tag{17}$$

The error difference due to the learning can be represented by [17]

$$\begin{aligned}\Delta e_I(k) = e_I(k+1) - e_I(k) &\approx \left[\frac{\partial e_I(k)}{\partial W_I}\right]^T \Delta W_I\\ &= \left[\frac{\partial e_I(k)}{\partial m_I}\ \frac{\partial e_I(k)}{\partial \sigma_I}\ \frac{\partial e_I(k)}{\partial \theta_I}\ \frac{\partial e_I(k)}{\partial w_I}\right]\\ &\quad\cdot\begin{bmatrix}\Delta m_I\\ \Delta\sigma_I\\ \Delta\theta_I\\ \Delta w_I\end{bmatrix}\end{aligned} \tag{18}$$

where ΔW_I denotes the change in an arbitrary weighting vector. From (9) and (11), we have

$$\Delta W_I \equiv -\underline{\eta}_I e_I(k)\,\frac{\partial e_I(k)}{\partial W_I} = \underline{\eta}_I e_I(k)\,\frac{\partial O_I^4(k)}{\partial W_I} \tag{19}$$

that is

$$\begin{bmatrix}\Delta m_I\\ \Delta\sigma_I\\ \Delta\theta_I\\ \Delta w_I\end{bmatrix} = e_I(k)\begin{bmatrix}\eta_I^m & 0 & 0 & 0\\ 0 & \eta_I^\sigma & 0 & 0\\ 0 & 0 & \eta_I^\theta & 0\\ 0 & 0 & 0 & \eta_I^w\end{bmatrix}\begin{bmatrix}\dfrac{\partial O_I(k)}{\partial m_I}\\ \dfrac{\partial O_I(k)}{\partial\sigma_I}\\ \dfrac{\partial O_I(k)}{\partial\theta_I}\\ \dfrac{\partial O_I(k)}{\partial w_I}\end{bmatrix}$$

where $W_I = [m_I, \sigma_I, \theta_I, w_I]^T$ and $\underline{\eta}_I = [\eta_I^m, \eta_I^\sigma, \eta_I^\theta, \eta_I^w]^T$ are the tuning parameters and the corresponding learning rates in the RFNNI and $O_I(k)$ is the current output of the RFNNI for each discrete time k.

Now we have the following convergence theorem.

Theorem 2: Let $\underline{\boldsymbol{\eta}}_I = [\eta_I^1\ \eta_I^2\ \eta_I^3\ \eta_I^4]^T = [\eta_I^m\ \eta_I^\sigma\ \eta_I^\theta\ \eta_I^w]^T$ be the learning rates for the tuning parameters of the RFNNI and let $P_{I,\max}$ be defined as

$$\begin{aligned}P_{I,\max} &\equiv [P_{I_{1,\max}} \quad P_{I_{2,\max}} \quad P_{I_{3,\max}} \quad P_{I_{4,\max}}]^T\\ &= \left[\max_k\left|\frac{\partial O_I(k)}{\partial m_I}\right| \quad \max_k\left|\frac{\partial O_I(k)}{\partial\sigma_I}\right| \right.\\ &\qquad \left.\max_k\left|\frac{\partial O_I(k)}{\partial\theta_I}\right| \quad \max_k\left|\frac{\partial O_I(k)}{\partial w_I}\right|\right]^T.\end{aligned}$$

Then asymptotic convergence is guaranteed if η_I^i are chosen to satisfy

$$0 < \eta_I^i < \frac{2}{(P_{Ii,\max})^2}, \qquad i = 1, \cdots, 4.$$

Lemma 1: If the learning rates are chosen as $\eta_I = \eta_I^m = \eta_I^\sigma = \eta_I^\theta = \eta_I^w$, then we have the convergence condition

$$0 < \eta_I < \frac{2}{(P_{I,\max})^2}$$

where

$$\begin{aligned}P_{I,\max} &= \max_k \|P(k)\|\\ &= \max_k\left\|\left[\frac{\partial O_I(k)}{\partial m_I} \quad \frac{\partial O_I(k)}{\partial\sigma_I} \quad \frac{\partial O_I(k)}{\partial\theta_I} \quad \frac{\partial O_I(k)}{\partial w_I}\right]^T\right\|\end{aligned}$$

and $\|\cdot\|$ is the usual Euclidean norm. Additionally, the maximum learning rate which guarantees convergence corresponds to $\eta_I^\star = (1/(P_{I,\max})^2)$.

The general convergence Theorem 2 can now be applied to find the specific convergence criterion for each type of parameter.

Theorem 3: Let η_I^w be the learning rate for the RFNNI weights w_I. Then asymptotic convergence is guaranteed if the learning rate satisfies: $0 < \eta_I^w < (2/R_I)$, where R_I is the number of rules in the RFNNI.

Theorem 4: We define η_I^m, η_I^σ and η_I^θ to be the learning rates for the RFNNI parameters m_I, σ_I and θ_I, respectively. Then asymptotic convergence is guaranteed if the learning rates are chosen as follows:

$$\begin{aligned}&0 < \eta_I^m, \eta_I^\sigma,\\ &\eta_I^\theta < \frac{2}{R_I N_{I,\max}}\left[\frac{1}{|w_{I,\max}|\left(\dfrac{2}{\sigma_{I,\min}}\right)}\right]^2\end{aligned}$$

where R_I and $N_{I,\max}$ denote the number of rules in the RFNNI and the maximum of the number of fuzzy sets with respect to the input.

Remark 4: From Lemma 1, the optimal learning rates of the RFNNI are

$$\begin{aligned}\eta_I^{w^\star} &= \frac{1}{R_I}\\ \eta_I^{m^\star} &= \eta_I^{\delta^\star} = \eta_I^{\theta^\star}\\ &= \frac{1}{R_I N_{I,\max}}\left[\frac{1}{|w_{I,\max}|\left(\dfrac{2}{\sigma_{I,\min}}\right)}\right]^2.\end{aligned}$$

B. Stability Analysis for Indirect Control

Similar to (16) and (18), we have

$$V_C(k) = \frac{1}{2}e_C^2(k) \quad \text{and}$$
$$\Delta e_C(k) \approx \left[\frac{\partial e_C(k)}{\partial W_C}\right]^T \Delta W_C.$$

Thus,

$$\begin{aligned}\Delta W_C &\equiv -\underline{\eta}_C e_C(k)\,\frac{\partial e_C(k)}{\partial w_C}\\ &= \underline{\eta}_C e_C(k) y_u(k)\,\frac{\partial O_C^4(k)}{\partial w_C}\end{aligned} \tag{20}$$

where $y_u(k)$ is defined in Remark 3.

Now we have the following convergence theorem for the RFNNC.

Theorem 5: Let

$$\underline{\boldsymbol{\eta}}_C = [\eta_C^1 \quad \eta_C^2 \quad \eta_C^3 \quad \eta_C^4]^T = [\eta_C^m \quad \eta_C^\sigma \quad \eta_C^\theta \quad \eta_C^w]^T$$

be the learning rates for the tuning parameters of the RFNNC and let $P_{C,\max}$ be defined as

$$\begin{aligned}P_{C,\max} &\equiv [P_{C_{1,\max}} \quad P_{C_{2,\max}} \quad P_{C_{3,\max}} \quad P_{C_{4,\max}}]^T\\ &= \left[\max_k\left|\frac{\partial O_C(k)}{\partial m_C}\right| \quad \max_k\left|\frac{\partial O_C(k)}{\partial\sigma_C}\right| \right.\\ &\qquad \left.\max_k\left|\frac{\partial O_C(k)}{\partial\theta_C}\right| \quad \max_k\left|\frac{\partial O_C(k)}{\partial w_C}\right|\right]^T.\end{aligned}$$

Then asymptotic convergence is guaranteed if η_C^i, $i = 1, \cdots, 4$ are chosen to satisfy

$$0 < \eta_C^i < \frac{2}{(y_u P_{C_i,\max})^2}.$$

Lemma 2: If the learning rates are chosen as $\eta_C = \eta_C^m = \eta_C^\sigma = \eta_C^\theta = \eta_C^w$, then we have the convergence condition

$$0 < \eta_C < \frac{2}{(y_u(k)P_{C,\max})^2}$$

where

$$P_{C,\max} = \max_k \|P_C(k)\| = \max_k \left\| \left[\frac{\partial O_C(k)}{\partial m_C} \; \frac{\partial O_C(k)}{\partial \sigma_C} \; \frac{\partial O_C(k)}{\partial \theta_C} \; \frac{\partial O_C(k)}{\partial w_C} \right]^T \right\|$$

and $\|\cdot\|$ is the usual Euclidean norm. Additionally, the maximum learning rate which guarantees convergence is

$$\eta_C^\star = \frac{1}{(y_u(k)P_{C,\max})^2}.$$

The general convergence theorem can now be applied to find the specific convergence criterion for each type of parameter.

Theorem 6: Let η_C^w be the learning rate for the RFNNC weights w. Then asymptotic convergence is guaranteed if the learning rates satisfies

$$0 < \eta_C^w < \frac{2}{y_u^2(k)R_C}$$

where R_C is the number of rules in the RFNNC.

Theorem 7: We define η_C^m, η_C^σ and η_C^θ to be the learning rates for the RFNNC parameters m_C, σ_C and θ_C, respectively. Then asymptotic convergence is guaranteed if the learning rates are chosen as follows:

$$0 < \eta_C^m, \eta_C^\sigma$$

$$\eta_C^\theta < \frac{2}{y_u^2(k)R_C N_{C,\max}} \left[\frac{1}{|w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right)} \right]^2$$

where R_C and $N_{C,\max}$ denote the number of rules in the RFNN and the maximum of the number of fuzzy sets with respect to the input, respectively.

Table I shows the conditions on learning rates for identification and control from Theorems 2–7.

Remark 5: In the previous discussion, the system sensitivity $y_u(k)$ is provided by the RFNN identifier (RFNNI). Therefore, in the convergent conditions of Theorems 5–7 (also in Table I), the sensitivity y_u must be replaced by $S_{\max}$, where

$$\begin{aligned} S_{\max} &= \max_k \{y_u(k)\} \\ &= \max \left\{ \sum_{a=1}^{R_I} w_{Iaj} \left[\sum_{k=1}^{N_{I_k}} \frac{O_{Ia}^3}{O_{Ika}^2} \cdot (-2) \cdot \frac{u_i + O_{Ika}^2 \theta_{Ika} - m_{Iak}}{\sigma_{Ika}} \right] \right\} \\ &= 2R_I N_{I,\max} \max \|w_I\| \cdot \max_k \left| \frac{u_i + O_{Ika}^2 \theta_{Ika} - m_{Ika}}{\sigma_{Ika}} \right| \\ &= 2R_I N_{I,\max} \cdot w_{I,\max} \cdot \frac{M + \theta_{I,\max} + m_{I,\max}}{\sigma_{I,\min}} \end{aligned}$$

TABLE I
CONDITIONS FOR LEARNING RATES

	identification
m	$0 < \eta_I^m < \frac{2}{R_I N_{I,\max}} \frac{1}{[\|w_{I,\max}\|(2/\sigma_{I,\min})]^2}$
σ	$0 < \eta_I^\sigma < \frac{2}{R_I N_{I,\max}} \frac{1}{[\|w_{I,\max}\|(2/\sigma_{I,\min})]^2}$
θ	$0 < \eta_I^\theta < \frac{2}{R_I N_{I,\max}} \frac{1}{[\|w_{I,\max}\|(2/\sigma_{I,\min})]^2}$
w	$0 < \eta_I^w < \frac{2}{R_I}$
	Control
m	$0 < \eta_C^m < \frac{2}{R_C y_u^2(k) N_{C,\max}} \frac{1}{[\|w_{C,\max}\|(2/\sigma_{C,\min})]^2}$
σ	$0 < \eta_C^\sigma < \frac{2}{R_C y_u^2(k) N_{C,\max}} \frac{1}{[\|w_{C,\max}\|(2/\sigma_{C,\min})]^2}$
θ	$0 < \eta_C^\theta < \frac{2}{R_C y_u^2(k) N_{C,\max}} \frac{1}{[\|w_{C,\max}\|(2/\sigma_{C,\min})]^2}$
w	$0 < \eta_C^w < \frac{2}{R_C y_u^2(k)}$

and where M is the bound of control input $u(k)$, i.e., $|u(k)| \leq M$, for all k. Therefore, the convergent conditions of tuning parameters, m, σ, θ, for indirect adaptive control must be changed to

$$0 < \eta_C^m, \eta_C^\sigma, \eta_C^\theta < \frac{2}{R_C S_{\max}^2 N_{C,\max}} \frac{1}{[|w_{C,\max}|(2/\sigma_{C,\min})]^2}.$$

V. SIMULATION RESULTS

Several examples and performance comparisons with the FNN are presented in this section to verify the performance of the RFNN for temporal problems, identification, and adaptive control for nonlinear systems. The examples given here include the time-series prediction problem as well as identification and control of nonlinear systems.

Example 1: Time sequence prediction. To clearly verify that the proposed RFNN can learn temporal relationships, a simple sequence prediction problem found in [14] is used as a test in the following example.

The test bed used is shown in Fig. 5(a). This is a "figure eight" shape made up of a series of 12 points to be presented to the network in the order shown. The RFNN is asked to predict the succeeding point for every presented point. Obviously, this task cannot be accomplished by a static network because the point at coordinate $(0, 0)$ has two successors: point 5 and point 11. The RFNN must decide the successor of $(0, 0)$ based on it predecessor: if the predecessor is 3, then the successor is 5; if the predecessor is 9, however, the successor is 11.

Table II shows the parameters used for the RFNN and FNN. In this example, the RFNN contains only two input nodes, which give the two coordinates of the current point, and two output nodes, which represent the predicted point's coordinates. The predicted values are shown in Fig. 5(b) (solid line: desired output; dotted line: RFNN). We also applied the (nonrecurrent) FNN to this time prediction problem (solid line: desired output; dotted line: FNN). The prediction results after training are shown in Fig. 5(c), verifying that a feedforward fuzzy neural network cannot predict successfully. Fig. 5(d) shows the mean square error (MSE) for the FNN and RFNN (solid line: RFNN; dotted line: FNN). From the simulation results shown in Fig. 5(b), we can see that the FNN is inappropriate for time sequence prediction because of its static mapping.

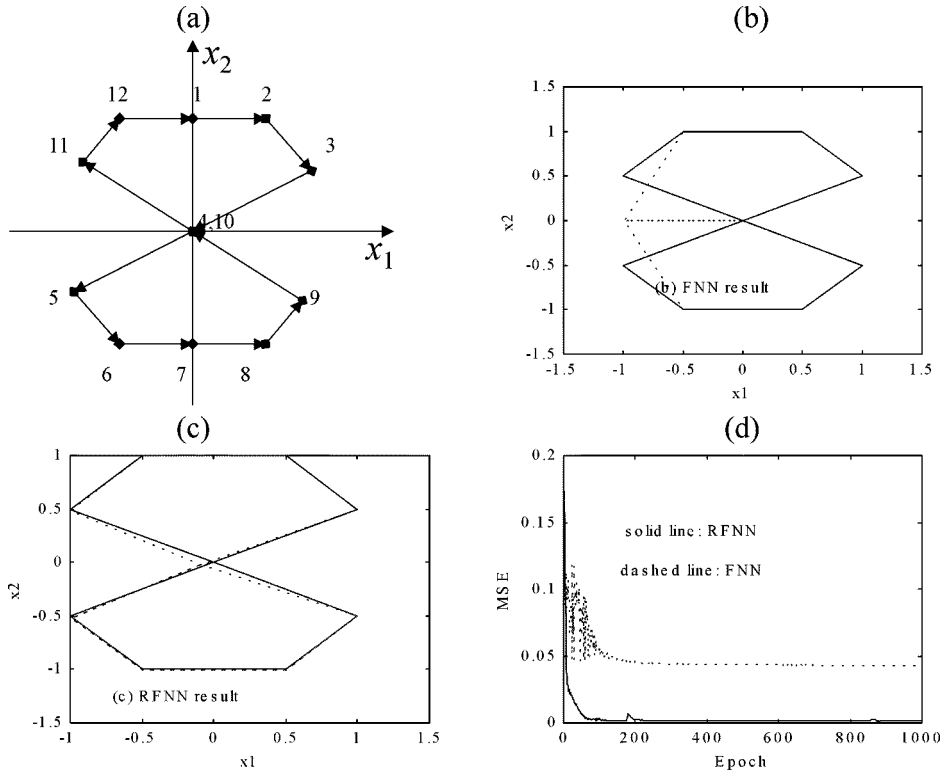

Fig. 5. Simulation results of time-series prediction. (a) Test bed for the next sample prediction experiment in Example 1. (b) Results of prediction using the RFNN after 1000 training epochs. (c) Results of prediction using the FNN after 1000 training epochs. (d) MSE of the RFNN and FNN.

TABLE II
PARAMETERS FOR EXAMPLE 1

	RFNN	FNN
No. of inputs: n	2	4
No. of outputs: p	2	2
Rule number: m	12	12
Training patterns	12	12
nodes	40	66
parameters	84	108
Learning rates	$\eta_I^m = \eta_I^\sigma = \eta_I^\theta = 0.011$, $\eta_I^w = 0.055$	
epochs	1000	

Example 2: Identification of a nonlinear dynamic system. In this example, the nonlinear plant with multiple time-delay is described as [10]

$$y_p(k+1) = f(y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)) \tag{21}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2}.$$

Here, the current output of the plant depends on three previous outputs and two previous inputs. In [4], [5], and [10], the feedforward neural network, with five input nodes for feeding the appropriate past values of y_p and u were used. In this paper, only two values, $y_p(k)$ and $u(k)$, are fed into the RFNN to determine the output $y_p(k)$. In training the RFNN, we used 100 epochs (90 000 time steps). The testing input signal $u(k)$ as the following equation is used to determine the identification results

$$u(k) = \begin{cases} \sin\left(\frac{\pi k}{25}\right), & 0 < k < 250 \\ 1.0, & 250 \le k < 500 \\ -1.0, & 500 \le k < 750 \\ 0.3\sin\left(\frac{\pi k}{25}\right) + 0.1\sin\left(\frac{\pi k}{32}\right) & \\ \quad +0.6\sin\left(\frac{\pi k}{10}\right), & 750 \le k < 1000. \end{cases}$$

Fig. 6(a) shows results using the FNN and RFNN for identification. Fig. 6(b) presents the MSEs of the RFNN and FNN (solid line: the RFNN result; dotted line: the FNN result). The parameters for training the RFNN and FNN are listed in Table III. It is clear that the RFNN results a small network structure and a small number of tuning parameters from Tables II and III.

This simulation demonstrates that the RFNN has the smaller network structure for identification. In addition, we observe that the identification error of the RFNN is less than that of the FNN after 90 000 time steps.

Example 3: Identification of a chaotic system. The discrete-time Henon system is frequently used in the study of chaotic dynamics and is not overly simple in the sense that it is of second order with one delay and two parameters [3]. This chaotic system is described by

$$y(k+1) = -P \cdot y^2(k) + Q \cdot y(k-1) + 1.0, \quad k = 1, 2, \cdots$$

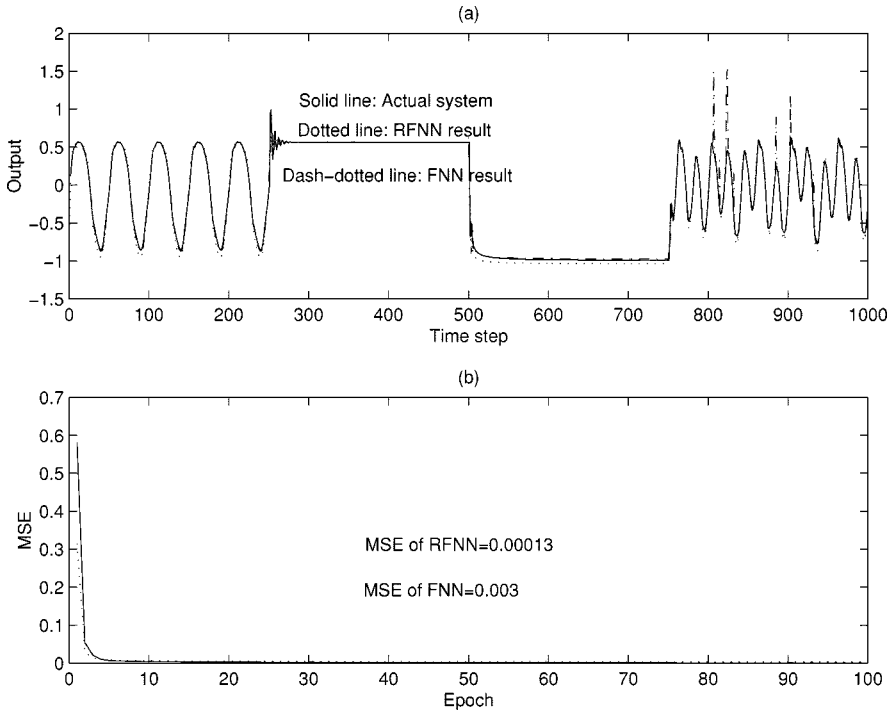

Fig. 6. Simulation results for nonlinear system identification. (a) Comparison of the RFNN (dotted line), FNN with tapped delay (dash-dotted line), and actual system output (solid line). (b) MSEs of the RFNN (solid line) and FNN with tapped delay (dotted line).

TABLE III
PARAMETERS FOR EXAMPLE 2

	RFNN	FNN
No. of inputs: n	2	5
No. of outputs: p	1	1
Rule number: m	16	16
Training pattern	900	900
nodes	51	112
parameters	112	176
Learning rates	$\eta_I^m = \eta_I^\sigma = \eta_I^\theta = 0.01$ $\eta_I^w = 0.1$	
epochs	100	100

which, with $P = 1.4$ and $Q = 0.3$, produces a chaotic strange attractor as shown in [3]. For this study, the input of the RFNN is $y(k-1)$ and the output is $y(k)$. We first randomly choose the training data (1000 pairs) from system over the interval $[-1.5, 1.5]$. Then, the RFNN is used to approximate the chaotic system (the parameters in the RFNN being updated by the BP algorithm). Fig. 7 shows the phase plane of this chaotic system after training (100 epochs). Here the initial point is $[y(1), y(0)]^T = [0.4, 0.4]^T$ and the MSE is 0.0136 less than was achieved in [3] (0.0186).

Example 4: Adaptive control of nonlinear systems. The indirect adaptive control method is used in this example. We consider here the problem of controlling a nonlinear system which was considered in [10]. A brief description is as follows (details can be found in [10]). The system model is

$$y(k+1) = f[y(k), y(k-1)] + u(k)$$

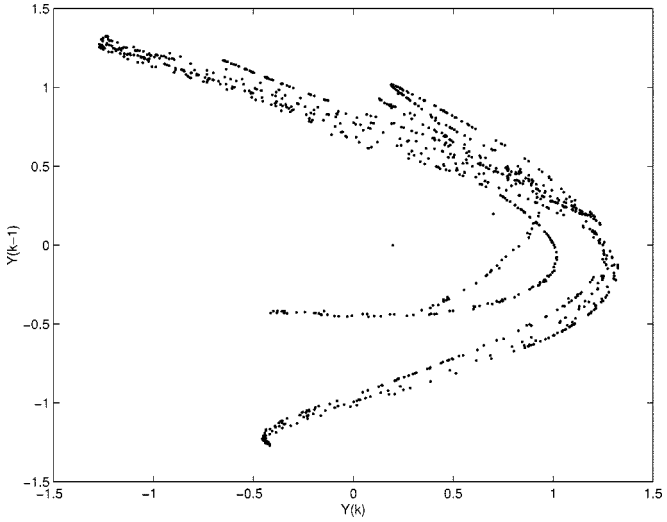

Fig. 7. Result of the phase plot for the chaotic system.

where

$$f[y(k), y(k-1)] = \frac{y(k)y(k-1)\,(y(k)+2.5)}{1+y^2(k)+y^2(k-1)}$$

is assumed to be unknown. A reference model is described as

$$y_m(k+1) = 0.6y_m(k) + 0.2y_m(k-1) + r(k)$$

where $r(k)$ is a bounded reference input. However, since function $f[\cdot]$ is unknown, it is estimated on-line as $\hat{f}$. Then, the control input is

$$u(k) = -\hat{f}(y(k), y(k-1)) + 0.6y(k) + 0.2y(k-1) + r(k) \tag{22}$$

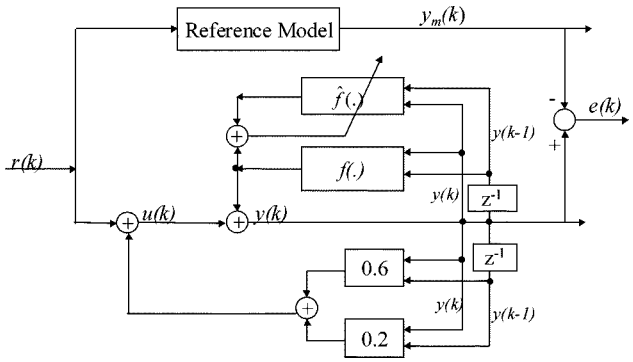

Fig. 8. Control architecture for Example 4.

where $r(k) = \sin(2\pi k/25)$. The control architecture is shown in Fig. 9. In this example, the unknown system is identified off-line using random inputs. Then, the control law (22) can generate the control input. Finally, both identification and control are implemented simultaneously. Figs. 8 and 9 give the control architecture and simulation result. In this example, the RFNN successfully approximated the continuous function $f(y(k), y(k-1))$. This suggests that the RFNN is also a static mapping network. In fact, if we assign the values to listed below parameter θ:

$$\boldsymbol{\theta} = [0.000 \quad \cdots \quad 0.000]_{1\times R} \approx \mathbf{0}$$

where R is the rule number, then the RFNN reduces to an FNN. This confirms that the RFNN is also a static network.

Example 5: Direct adaptive control. The model reference control problem for a nonlinear system with linear input [8] is consider below. The nonlinear system is described by

$$\begin{aligned} y(k+1) = {} & 0.2y^2(k) + 0.2y(k-1) \\ & + 0.4\sin[0.5(y(k)+y(k-1))] \\ & \cdot \cos[0.5(y(k)+y(k-1))] + 1.2u(k). \end{aligned}$$

The reference model is described by the difference equation

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

where $r(k) = \beta\sin(2\pi j/100)$. The system model output diverges when the step input $u(k)=0.83, \forall k \geq 0$ is applied to the system. This implies that the reference model signal needs to be restricted such that $\beta \leq 0.9$. In this example, we adopt $\beta=0.2$.

In this example, the system sensitivity $y_u(k) = (\partial y(k)/\partial u(k))=1.2$, the inputs of the RFNN are $[r(k)\ u(k-1)\ y(k-1)]$, and the learning rates are $\eta_w = \eta_m = \eta_\sigma = \eta_\theta = 0.07$. Fig. 10 shows the final system response.

Remark 6: The previous results can be summarized as follows.

1) The RFNN provides a new recurrent fuzzy neural network constructed from dynamic fuzzy if-then rules and shares the advantages of the FNN.
2) The RFNN has small network structure and a small number of tuning parameters in application.
3) The RFNN is capable of dynamic mapping for solving the temporal problems. In constrast, we see from Example 1 that the FNN cannot predict the time-series successfully. That is, the FNN is not a dynamic mapping system.
4) The RFNN is also a static network when the parameter θ is set to zero since this reduces the RFNN to an FNN. We can conclude that the RFNN is a generalized FNN system, which combines dynamic characteristics and the advantages of FNN.
5) The RFNN-based control system was tested for its on-line adapting ability and found to perform well.

VI. Conclusion

This paper has proposed an RFNN that is a generalization FNN. This RFNN is a recurrent multilayered connectionist network for realizing fuzzy inference using the dynamic fuzzy rules. The network consists of four layers including two hidden layers and a feedback layer. The temporal relations embedded in the network were built by adding feedback connections to the FNN, where the feedback units act as memory elements. A simple comparison showed that this modification simplifies the network structure. Moreover, we have successfully extended the results for the FNN to the RFNN:

1) the RFNN was proven to be a universal approximator;
2) its (dynamic) fuzzy inference system was presented;
3) using the Lyapunov approach, convergence theorems for the RFNN were proven and the optimal adaptive learning rates were also established.

The RFNNs capability to temporarily store information allowed us to extend the application domain to include temporal problems. Finally, the proposed RFNN was applied to identify nonlinear dynamic systems.

We proposed direct and indirect adaptive control architectures for nonlinear systems using RFNNs. Simulation results show that the RFNN has the following advantages:

1) RFNN has the capabilities of attractor dynamics and temporary information storage;
2) in application, the RFNN has smaller network structure and a small number of tuning parameters than the FNN;
3) RFNN successfully solved the temporal problems and can approximate a dynamic system mapping as accurately as desired;
4) RFNN is also a static network (an FNN) when the parameters $\theta = 0$; we conclude that the RFNN is a generalization system of the FNN, which combines dynamic characteristics with the advantages of the FNN;
5) RFNN has on-line adapting ability for dynamic system control.

Appendix

A. Proof of the Universal Approximation Theorem

Theorem 1 will be proven by using the Stone–Weierstrass theorem. We shall begin with the single-output case and then extend it to the multiple-output case.

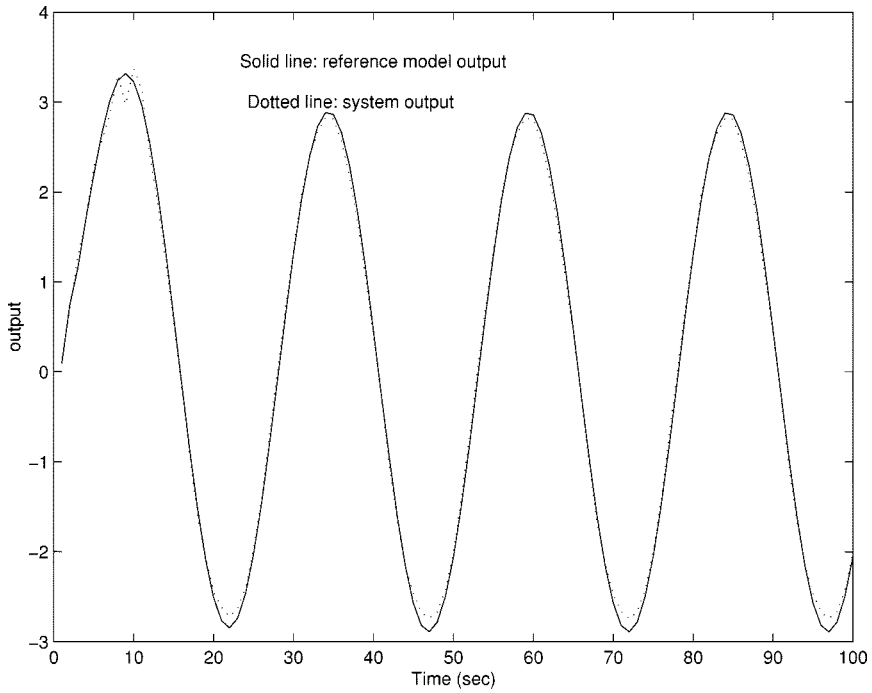

Fig. 9. Response for $r(k) = \sin(2\pi k/25)$ with control (solid line: reference model output $y_m(k)$; dashed line: system output $y(k)$.

The structure of the proposed RFNN is illustrated in Fig. 1. The single-output of the RFNN can be expressed as

$$y(x) = \sum_{j=1}^{m} w_j \cdot \alpha_j(I_j(x)) \tag{23}$$

where $x \in \mathbb{R}^n$ is the input variable of the RFNN

$$\alpha_j(I_j) = O_j^3 = \prod_{i=1}^{n} \mu_{A_{ij}}(I_{ij}) = \prod_{i=1}^{n} \exp\left[-\frac{(I_{ij} - m_{ij})^2}{(\sigma_{ij})^2}\right]$$

is a function of the input $I_j = [I_{1j}, I_{2j}, \cdots, I_{nj}]^T$, and the link weight w_j is the output action strength. For $i = 1, \cdots, n$, $I_{ij}(k) = u_{ij}^2(k) = x_i(k) + O_{ij}^2(k-1) \cdot \theta_{ij}$ denotes the input of layer 2. Let Φ be a set of functions that have the form

$$\prod_{i=1}^{n} \exp\left(-\left(\frac{I_{ij} - b}{a}\right)^2\right)$$

where $a, b \in \mathbb{R}$, and F^n is the family of function y: $\mathbb{R}^n \to \mathbb{R}$. As in the previous description, in the RFNN, the value of a and b are σ_{ij} and m_{ij}, respectively. The output y is therefore written

$$y(x) = \sum_{j=1}^{m} w_j \cdot \alpha_j, \text{for } w_j \in \mathbb{R}, \{\alpha_j\} \in \Phi, x \in \mathbb{R}^n \quad j = 1, 2, \cdots, m. \tag{24}$$

In order to prove the universal approximation theorem, the following definitions and theorem which are quoted from [12] are necessary.

Definition 1:

A.1 A real functions family $\mathcal{A}$ defined on a set $\mathcal{K}$ is an *algebra* if: (i) $f + g \in \mathcal{K}$, (ii) $fg \in \mathcal{K}$, and (iii) $cf \in \mathcal{K}$ are satisfied, where $f \in \mathcal{A}$, $g \in \mathcal{K}$, and c is a complex constant, i.e., $\mathcal{K}$ is closed under addition, multiplication, and scalar multiplication. For example, the set of all polynomials is an algebra.

A.2 A family $\mathcal{K}$ is uniformly closed if $f \in \mathcal{K}$ whenever $f_n \in \mathcal{A}$, $n = 1, 2, \cdots$ and $f_n \to f$ uniformly on $\mathcal{K}$.

A.3 The uniform closure of $\mathcal{K}$, denoted by $\mathcal{B}$, is the set of all functions that are limits of uniformly convergent sequences of members of $\mathcal{K}$. By the Stone–Weierstrass theorem, it is known that the set of continuous functions on $[a, b]$ is the uniform closure of the set of polynomials on $[a, b]$.

A.4 $\mathcal{K}$ *separates points* on a set $\mathcal{K}$ if for every x, y in $\mathcal{K}$, $x \neq y$, there is a function f in $\mathcal{K}$ such that $f(x) \neq f(y)$.

A.5 $\mathcal{K}$ *vanishes at no point of* $\mathcal{K}$ if for each x in $\mathcal{K}$, there is a function f in A such that $f(x) \neq 0$.

Theorem 8: (Stone–Weierstrass Theorem) [12] Let $\mathcal{K}$ be a set of real continuous functions on a compact set $\mathcal{K}$. If (1) $\mathcal{K}$ is an algebra, (2) $\mathcal{K}$ separates points on $\mathcal{K}$, and (3) $\mathcal{K}$ vanishes at no point of $\mathcal{K}$, then the uniform closure of $\mathcal{K}$ consists of all real continuous functions on $\mathcal{K}$.

Now we are ready to prove the universal approximation theorem. The proof is divided into four parts as follows:

Lemma 3: Let F^n be the family of y defined in (24), then $F^n \subset \mathcal{K}$, where $\mathcal{K}$ is a compact set.

Proof: Here, the membership function

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

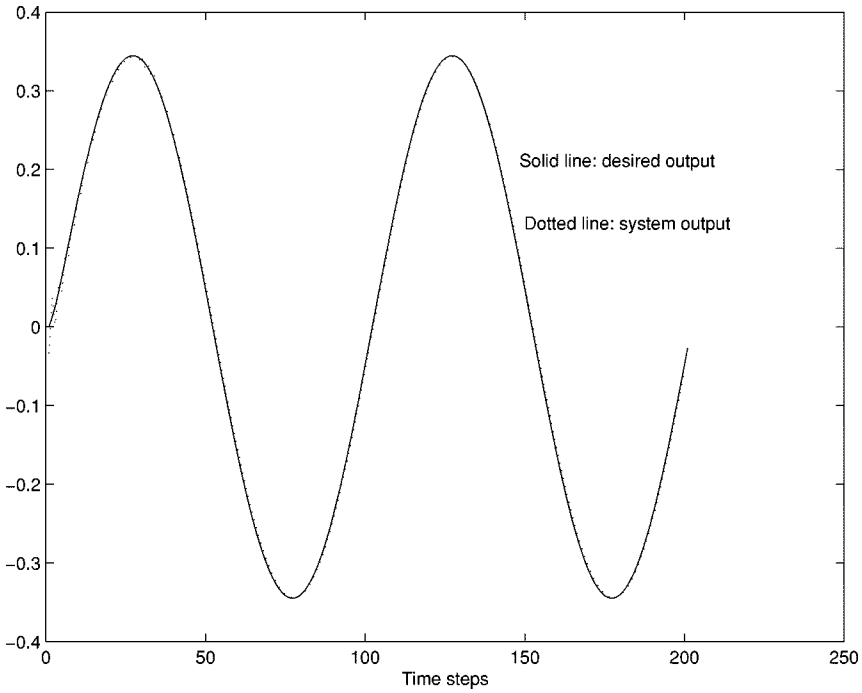

Fig. 10. Final system response for Example 5 (dashed line: reference signal; solid line: system output).

and, therefore, the continuous function $\alpha_j(I_j(x))$ is closed and bounded for all $x \in \mathbb{R}^n$. That is, $F^n \subset \mathcal{K}$. ■

Lemma 4: F^n is an algebra (if $f, g \in F^n$ and $c \in \mathbb{R}$, then $f+g \in F^n$, $f \cdot g \in F^n$, and $cf \in F^n$).

Proof: Let $f, g \in F^n$ as shown in (24). We can write them as

$$f(\mathbf{x}) = \sum_{j=1}^{k_1} w_j^1 \alpha_j^1(I_{ij}(x)) = \sum_{j=1}^{k_1} w_j^1 \prod_{i=1}^{n} \exp\left[-\frac{(I_{ij}-m_{ij}^1)^2}{(\sigma_{ij}^1)^2}\right] \tag{25}$$

$$g(\mathbf{x}) = \sum_{j=1}^{k_2} w_j^2 \alpha_j^2(I_{ij}(x)) = \sum_{j=1}^{k_2} w_j^2 \prod_{i=1}^{n} \exp\left[-\frac{(I_{ij}-m_{ij}^2)^2}{(\sigma_{ij}^2)^2}\right] \tag{26}$$

where w_j^1 and $w_j^2 \in \mathbb{R}$, $\forall j$, and I_{ij} is a time sequence of x_i, $i = 1, \cdots, n$. That is

$$\begin{aligned} I_{ij}(1) &= x_i(1) \\ &\vdots \\ I_{ij}(k) &= x_i(k) + O_{ij}^2(k-1) \cdot \theta_{ij} \\ &= x_i(k) + \exp\left[-\left(\frac{I_{ij}(k-1)-m_{ij}}{\sigma_{ij}}\right)^2\right] \cdot \theta_{ij}. \end{aligned}$$

1) Hence,

$$\begin{aligned} f+g &= \sum_{j=1}^{k_1} w_j^1 \alpha_j^1(I_j^1(x)) + \sum_{j=1}^{k_2} w_j^2 \alpha_j^2(I_j^2(x)) \\ &= \sum_{j=1}^{k_1+k_2} w_j \alpha_j(I_j(x)) = W \cdot \overline{\alpha} \end{aligned}$$

where $W = [w_1^1 \ w_2^1 \cdots w_{k_1}^1 \ w_1^2 \cdots w_{k_2}^2]^T$ and $\overline{\alpha} = [\alpha_1^1 \ \alpha_2^1 \cdots \alpha_{k_1}^1 \ \alpha_1^2 \cdots \alpha_{k_2}^2]^T$. Since $w_j \in \mathbb{R}$, and $\alpha_j \in \alpha$, then $f+g \in F^n$. That is, the linear combination of Gaussian functions is also a Gaussian function. So, F^n is closed under addition.

2) Similarly,

$$\begin{aligned} f \cdot g &= \sum_{j=1}^{k_1} w_j^1 \alpha_j^1(I_j^1(x)) \sum_{j=1}^{k_2} w_j^2 \alpha_j^2(I_j^2(x)) \\ &= W_1^T \hat{\alpha}_1 W_2^T \hat{\alpha}_2 = W_1^T (\hat{\alpha}_1 \cdot \hat{\alpha}_2^T) W_2 \end{aligned} \tag{27}$$

where

$$\begin{aligned} W_1 &= [w_1^1 \quad \cdots \quad w_{k_1}^1]^T \in \mathbb{R}^{k_1} \\ W_2 &= [w_1^2 \quad \cdots \quad w_{k_2}^2]^T \in \mathbb{R}^{k_2} \end{aligned}$$

and

$$\hat{\alpha}_1 \cdot \hat{\alpha}_2 = \begin{bmatrix} \alpha_1^1\alpha_1^2 & \cdots & & \cdots & \alpha_1^1\alpha_{k_2}^2 \\ \vdots & \ddots & & \cdots & \vdots \\ & & \alpha_i^1\alpha_j^2 & & \\ \vdots & & & \ddots & \\ \alpha_{k_1}^1\alpha_1^2 & \cdots & & \cdots & \alpha_{k_1}^1\alpha_{k_2}^2 \end{bmatrix}.$$

Since, for $i = 1, \cdots k_1$, $j = 1, \cdots, k_2$, $\hat{\alpha}_i^1$ and $\hat{\alpha}_j^2$, are Gaussian functions, their product $\alpha_i^1\alpha_j^2$ is also

a Gaussian function. This can be verified by straightforward algebraic operations. Thus, (27) has the same form as (23). That is, $f \cdot g \in F^n$ (F^n is closed under multiplication).

3) Finally, for arbitrary $c \in \mathbb{R}$

$$c \cdot f = \sum_{j=1}^{k_1} (c \cdot w_j^1)\alpha_j^1(I_j(x)) = \sum_{j=1}^{k_1} w_j^c \alpha_j^1(I_j(x))$$

where $w_j^c = c \cdot w_j^1 \in \mathbb{R}, \forall j$. That is, $cf(\cdot) \in F^n$ (F^n is closed under scalar multiplication).

By (1)–(3), we conclude that F^n is an algebra. ■

Lemma 5: F^n separates points on $\mathcal{K}$ (for $x, y \in F^n$, $x \neq y$, there is an $f \in F^n$ such that $f(x) \neq f(y)$).

Proof: We prove this lemma by constructing a function f. That is, we specify the number of fuzzy sets defined in $\mathcal{K}$, the parameters of the Gaussian membership functions, and the number of fuzzy rules, such that the resulting f (in the form of (24)) has the property $f(\underline{x}^0) \neq f(\underline{y}^0)$, for arbitrarily given $\underline{x}^0, \underline{y}^0 \in \mathcal{K}$ with $\underline{x}^0 \neq \underline{y}^0$).

Now, let $\underline{x}^0 = (x_1^0, x_2^0, \cdots, x_n^0)$ and $\underline{y}^0 = (y_1^0, y_2^0, \cdots, y_n^0)$, and choose two fuzzy rules as the fuzzy rule base. Furthermore, let the Gaussian membership functions be

$$\mu_{A_{i1}}(x_i) = \exp\left(-\frac{(I_{i1} - x_i^0)^2}{(\sigma)^2}\right) = \exp\left(-\frac{(x_i + O_{i1}\theta_{i1} - x_i^0)^2}{(\sigma)^2}\right)$$

$$\mu_{A_{i2}}(x_i) = \exp\left(-\frac{(I_{i2} - y_i^0)^2}{(\sigma)^2}\right) = \exp\left(-\frac{(x_i + O_{i2}\theta_{i2} - y_i^0)^2}{(\sigma)^2}\right).$$

Then f can be expressed as

$$f = w_1 \cdot \prod_{i=1}^{n} \exp\left(-\frac{(x_i + O_{i1}\theta_{i1} - x_i^0)^2}{(\sigma)^2}\right) + w_2 \cdot \prod_{i=1}^{n} \exp\left(-\frac{(x_i + O_{i2}\theta_{i2} - y_i^0)^2}{(\sigma)^2}\right) \quad (28)$$

where w_1, w_2 are the link weights. With this system, we have

$$f(\underline{x}^0) = w_1 \cdot \prod_{i=1}^{n} \exp\left(-\frac{(O_{i1}\theta_{i1})^2}{(\sigma)^2}\right) + w_2 \cdot \prod_{i=1}^{n} \exp\left(-\frac{(x_i^0 + O_{i2}\theta_{i2} - y_i^0)^2}{(\sigma)^2}\right)$$

$$f(\underline{y}^0) = w_1 \cdot \prod_{i=1}^{n} \exp\left(-\frac{(y_i^0 + O_{i1}\theta_{i1} - x_i^0)^2}{(\sigma)^2}\right) + w_2 \cdot \prod_{i=1}^{n} \exp\left(-\frac{(O_{i2}\theta_{i2})^2}{(\sigma)^2}\right).$$

Because (8) holds in training processes, it is easy to verify that $f(\underline{x}^0) \neq f(\underline{y}^0)$ if $\underline{x}^0 \neq \underline{y}^0$.

Lemma 6: F^n vanishes at no point of $\mathcal{K}$.

Proof: From (24), there are α_is that are constant and not equal to zero. That is, for all $x \in \mathbb{R}^n$, $\alpha_j(x) > 0$. If we choose $w_j > 0\ (j = 1, 2, \cdots, m)$, then $y > 0$ for any $x \in \mathbb{R}^n$. That is, any $y \in F^n$ with $w_j > 0$ can serve as the required f.

Therefore, the single output RFNN is a universal approximator from the Stone–Weierstrass theorem and Lemmas 3–6.

From the previous proofs, we can conclude that given a real function $h: \mathbb{R}^n \to \mathbb{R}$, continuous on $\mathcal{K}$, and $\epsilon > 0$, there exists an RFNN system $y \in \mathcal{B}$ satisfying condition (8), where $\mathcal{B}$ is the uniform closure of F^n, such that $\sup_{x \in \mathcal{K}} |y(x) - h(x)| < \epsilon$, for every x in $\mathcal{K}$. That is, an RFNN system with an arbitrarily large number of fuzzy logical rules can approximate any real continuous function in $C(\mathbb{R}^n)$ over a compact subset of $\mathbb{R}^n$.

Extension to Multiple Output Case: The following subsection will extend the previous results to the case of an RFNN with multiple outputs. Consider the following example.

Suppose f_1 and f_2 are functions of x and let functions f_1 and f_2 be approximated by m rules and p rules, respectively. This structure is shown in Fig. 11(a) and (b).

It is easy to combine these two single-output RFNNs into a new RFNN with two outputs, which is shown in Fig. 12. This new structure has $m + p$ rules; the first m rules are constructed from f_1 and the last p rules from f_2. In this structure, the consequence weights of the last p rules in the first output (f_1) are set to zero and the consequence weights of the first m rules in the second output (f_2) are also set to zero.

Based on the previous discussion, since the RFNN with a single output can perform universal approximation, there must be an RFNN with multiple outputs with an arbitrarily large number of fuzzy logical rules that can perform universal approximation on each output. This completes the proof. □

B. Proof of Convergence Theorems

To prove our results, the following facts are needed.

Fact 1: Let $g(y) = ye^{(-y^2)}$. Then $|g(y)| < 1, \forall y \in \mathbb{R}$.

Fact 2: Let $f(y) = y^2 e^{(-y^2)}$. Then $|f(y)| < 1, \forall y \in \mathbb{R}$.

Proof of Theorem 2: The Lyapunov function (16) is a time-invariant positive definite function. Thus, from (18) and (19), the change in the Lyapunov function is

$$\begin{aligned}
\Delta V_I(k) &= \frac{1}{2}\left[e_I^2(k+1) - e_I^2(k)\right] \\
&= \Delta e_I(k)\left[e_I(k) + \frac{1}{2}\Delta e_I(k)\right] \\
&= \left[\frac{\partial e_I(k)}{\partial W_I}\right]^T \underline{\eta}_I e_I(k) \frac{\partial O_I(k)}{\partial W_I} \\
&\quad \cdot \left\{e_I(k) + \frac{1}{2}\left[\frac{\partial e_I(k)}{\partial W_I}\right]^T \underline{\eta}_I e_I(k) \frac{\partial O_I(k)}{\partial W_I}\right\} \\
&= -\left[\frac{\partial O_I(k)}{\partial W_I}\right]^T \underline{\eta}_I e_I(k) \frac{\partial O_I(k)}{\partial W_I} \\
&\quad \cdot \left\{e_I(k) - \frac{1}{2}\left[\frac{\partial O_I(k)}{\partial W_I}\right]^T \underline{\eta}_I e_I(k) \frac{\partial O_I(k)}{\partial W_I}\right\} \\
&= -e_I^2(k) \sum_i^4 \left[\frac{1}{2}\eta_i \left(\frac{\partial O_I(k)}{\partial W_{I_i}}\right)^2 \right. \\
&\quad \left. \cdot \left(2 - \eta_I^i \left(\frac{\partial O_I(k)}{\partial W_{I_i}}\right)^2\right)\right] \\
&\equiv -\lambda_1 e_I^2(k)
\end{aligned}$$

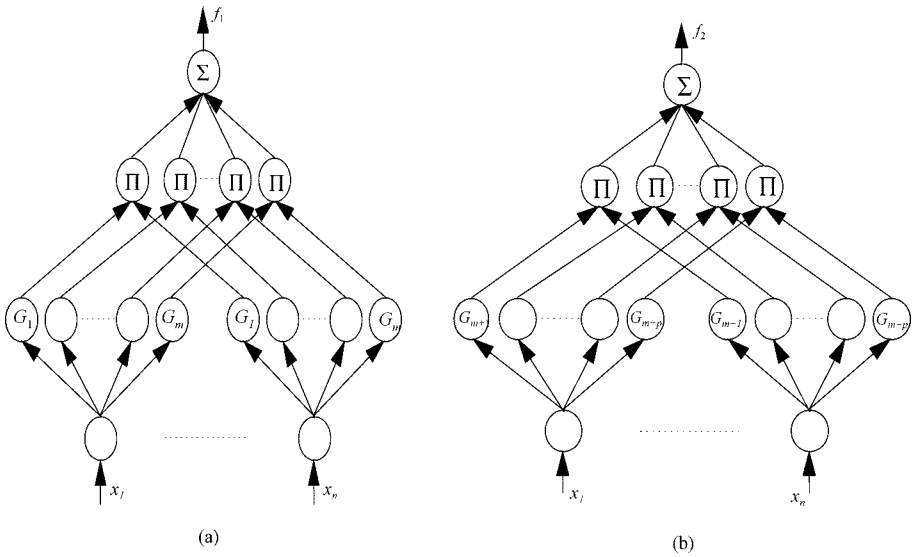

Fig. 11. Structural diagrams of f_1 and f_2.

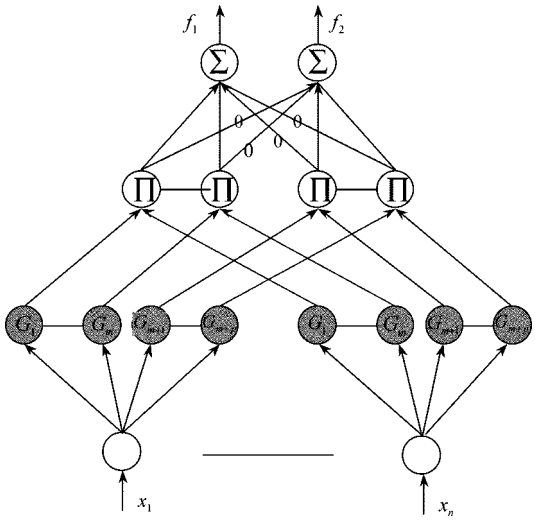

Fig. 12. Structural diagram of a multiple output system.

where

$$\lambda_1 = \sum_i^4 \left[\frac{1}{2}\eta_i \left(\frac{\partial O_I(k)}{\partial W_{I_i}}\right)^2 \left(2 - \eta_I^i \left(\frac{\partial O_I(k)}{\partial W_{I_i}}\right)^2\right)\right].$$

Because

$$\Delta V_I(k) = \tfrac{1}{2}[e_I^2(k+1) - e_I^2(k)] = -\lambda_1 e_I^2(k).$$

The convergence of the RFNN is guaranteed if $0 < \lambda_1$. We obtain

$$0 < \eta_I^i < \frac{2}{(P_{Ii,\max})^2}, \qquad i = 1, \cdots, 4.$$

This completes the proof. ■

Proof of Lemma 1: Since

$$\lambda_1 = \frac{1}{2}\eta_I \left\|\frac{\partial O_I(k)}{\partial W_I}\right\|^2 \left(2 - \eta_I \left\|\frac{\partial O_I(k)}{\partial W_I}\right\|^2\right)$$
$$= \frac{1}{2}\|P_I(k)\|^2 \eta_I (2 - \eta_I \|P_I(k)\|^2) > 0$$

we obtain $0 < \eta_I < (2/(P_{I,\max})^2)$ that guarantees convergence. However, the maximum learning rate which guarantees convergence is $\eta_I^\star = (1/P_{I,\max})^2)$. ■

Proof of Theorem 3: Let

$$P_{I_4}(k) = \frac{\partial O_I(k)}{\partial w_I} = Z_I(k)$$

where

$$Z_I = [Z_1, Z_2, \cdots, Z_{R_I}]^T$$

in which Z_j is the output value of the third layer of the RFNN and R_I is the number of rules in the RFNN. Then we have $Z_j \leq 1$ for all j, $|P_4(k)| \leq \sqrt{R_I}$, and $(P_{I_{4,\max}})^2 = \max_k |P_I(k)|^2 = R_I$. From Theorem 3, we find that $0 < \eta_I^w < (2/R_I)$. ■

Proof of Theorem 4:

1) Mean m_I:

From the previous discussion, we have the following result by chain rule:

$$P_{I_1}(k) = \frac{\partial O_I(k)}{\partial m_I} = \sum_{i=1}^{R_I} w_{I_i} \left\{\sum_{j=1}^{N_{I_j}} \frac{O_{Ii}^3}{O_{I_{ij}}^2} \cdot \frac{\partial O_{I_{ij}}^2}{\partial m_I}\right\}$$
$$< \sum_{i=1}^{R_I} w_{I_i} \left\{\sum_{j=1}^{N_{I_j}} \max\left(\frac{\partial O_{I_{ij}}^2}{\partial m_I}\right)\right\}$$
$$= \sum_{i=1}^{R_I} w_{I_i} \left\{\sum_{j=1}^{N_{I_j}} \max\left(\left(\frac{2}{\sigma_I}\right)\right.\right.$$
$$\cdot \left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)$$
$$\left.\left. \cdot \exp\left[-\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2\right]\right)\right\}.$$

Fact 1 gives

$$\left|\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)\exp\left[-\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2\right]\right| < 1.$$

Then

$$P_{I_1}(k) < \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \max\left(\frac{2}{\sigma_I}\right) \right\} < \sum_{i=1}^{R_I} w_{I_i} \sqrt{N_{I,\max}} \left\{ \max\left(\frac{2}{\sigma_I}\right) \right\} = \sum_{i=1}^{R_I} w_{I_i} \sqrt{N_{I,\max}} \left\{ \left(\frac{2}{\sigma_{I,\min}}\right) \right\}. \quad (29)$$

Thus

$$|P_{I_1}(k)| < \sqrt{R_I N_{I,\max}} |w_{I,\max}| \left(\frac{2}{\sigma_{I,\min}}\right).$$

Theorem 2 gives

$$0 < \eta_I^m < \frac{2}{P_{I_{1,\max}}^2} = \frac{2}{R_I N_{I,\max}} \left[\frac{1}{|w_{I,\max}| \left(\frac{2}{\sigma_{I,\min}}\right)} \right]^2.$$

2) STD: σ_I.

Similar to the previous discussion, we have

$$P_{I_2}(k) = \frac{\partial O_I(k)}{\partial \sigma_I} = \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \frac{O_{Ii}^3}{O_{I_{ij}}^2} \cdot \frac{\partial O_{I_{ij}}^2}{\partial \sigma_I} \right\} < \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \max\left(\frac{\partial O_{I_{ij}}^2}{\partial \sigma_I}\right) \right\} = \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \max\left(\left(\frac{2}{\sigma_I}\right) \cdot \left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2 \cdot \exp\left[-\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2 \right] \right) \right\}.$$

Fact 2 gives

$$\left| \left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2 \exp\left[-\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2 \right] \right| < 1.$$

Then

$$P_{I_2}(k) < \sum_{i=1}^{R_I} w_{I_i} \sqrt{N_{I,\max}} \left\{ \left(\frac{2}{\sigma_{I,\min}}\right) \right\}. \quad (30)$$

Thus

$$|P_{I_2}(k)| < \sqrt{R_I N_{I,\max}} |w_{I,\max}| \left(\frac{2}{\sigma_{I,\min}}\right).$$

Therefore, from Theorem 2 we find that

$$0 < \eta_I^\sigma < \frac{2}{P_{I_{3,\max}}^2} = \frac{2}{R_I N_{I,\max}} \left[\frac{1}{|w_{I,\max}| \left(\frac{2}{\sigma_{I,\min}}\right)} \right]^2.$$

3) Feedback layer's weight θ_I.

Similar to the previous discussion, we have

$$P_{I_3}(k) = \frac{\partial O_I(k)}{\partial \theta_I} = \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \frac{O_{Ii}^3}{O_{I_{ij}}^2} \cdot \frac{\partial O_{I_{ij}}^2}{\partial \theta_I} \right\} < \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \max\left(\frac{\partial O_{I_{ij}}^2}{\partial \theta_I}\right) \right\} = \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \max\left(\left(\frac{2O_I^2(k-1)}{\sigma_I}\right) \cdot \left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right) \cdot \exp\left[-\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2 \right] \right) \right\}.$$

Fact 2 gives

$$\left| \left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right) \exp\left[-\left(\frac{x + O_I^2\theta_I - m_I}{\sigma_I}\right)^2 \right] \right| < 1.$$

Then

$$P_{I_3}(k) < \sum_{i=1}^{R_I} w_{I_i} \left\{ \sum_{j=1}^{N_{I_j}} \max\left(\frac{2O_I^2(k-1)}{\sigma_I}\right) \right\} < \sum_{i=1}^{R_I} w_{I_i} \left\{ \sqrt{N_{I,\max}} \left(\frac{2}{\sigma_{I,\min}}\right) \right\}. \quad (31)$$

Thus

$$|P_{I_3}(k)| < \sqrt{R_I N_{I,\max}} |w_{I,\max}| \left(\frac{2}{\sigma_{I,\min}}\right).$$

Drawing on Theorem 2, we obtain

$$0 < \eta_I^\theta < \frac{2}{P_{I_{3,\max}}^2} = \frac{2}{R_I N_{I,\max}} \left[\frac{1}{|w_{I,\max}| \left(\frac{2}{\sigma_{I,\min}}\right)} \right]^2.$$

This completes the proof. ■

Proof of Theorem 5: The Lyapunov function $V_C(e_C) = \frac{1}{2}e_C^2$ is a time-invariant positive definite function. Thus, from (18) and (20), the change in the Lyapunov function is

$$\Delta V_C(k) = -e_C^2(k) \sum_i^4 \left[\frac{1}{2} y_u^2(k) \eta_C^i \left(\frac{\partial O_C}{\partial W_{C_i}}\right)^2 \cdot \left(2 - \eta_C^i y_u^2(k) \left(\frac{\partial O_C}{\partial w_{C_i}}\right)^2\right) \right] \equiv -\lambda_2 e_C^2(k)$$

where

$$\lambda_2 = \sum_i^4 \left[\frac{1}{2} y_u^2(k) \eta_C^i \left(\frac{\partial O_C}{\partial W_{C_i}}\right)^2 \cdot \left(2 - \eta_C^i y_u^2(k) \left(\frac{\partial O_C}{\partial w_{C_i}}\right)^2\right) \right].$$

Similarly, the convergence of the RFNNC is guaranteed if $0 < \lambda_2$. We obtain

$$0 < \eta_I^i < \frac{2}{(y_u(k)P_{Ci,\max})^2}, \qquad i = 1, \cdots, 4.$$

This completes the proof. ■

Proof of Lemma 2: Since

$$\lambda_2 = \tfrac{1}{2}\,||P_C(k)||^2 \eta_C y_u^2(k)\,(2 - \eta_C y_u^2(k)||P_C(k)||^2) > 0 \tag{32}$$

we obtain

$$0 < \eta_C < \frac{2}{(y_u(k)P_{C,\max})^2}$$

that guarantees convergence. However, the maximum learning rate for control which guarantees convergence is $\eta_C^\star = (1/(y_u(k)P_{C,\max})^2)$. ■

Proof of Theorem 6: Let

$$P_{C_4}(k) = \frac{\partial O_C(k)}{\partial w_C} = Z_C(k)$$

where

$$Z_C = [Z_1, Z_2, \cdots, Z_{R_C}]^T$$

in which Z_j is the output value of the third layer of the RFNN and R_C is the number of rules in the RFNN. Then we have $Z_j \le 1$ for all j, $|P_{C_4}(k)| \le \sqrt{R_C}$, and $(P_{C_{4,\max}})^2 = \max_k |P_C(k)|^2 = R_C$. From Theorem 3, we find that $0 < \eta_C^w < (2/R_C y_u^2(k))$. ■

Proof of Theorem 7:

1) Mean: m_C.

From the previous discussion, we obtain

$$P_{C_1}(k) = \frac{\partial O_C(k)}{\partial m_C} < \sum_{i=1}^{R_C} w_{C_i} \left\{ \sum_{j=1}^{N_{C_j}} \max \left(\left(\frac{2}{\sigma_C} \right) \cdot \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right) \cdot \exp \left[- \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \right] \right) \right\}.$$

Fact 1 gives

$$\left| \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right) \exp \left[- \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \right] \right| < 1.$$

Then

$$P_{C_1}(k) < \sum_{i=1}^{R_C} w_{C_i} \sqrt{N_{C,\max}} \left\{ \left(\frac{2}{\sigma_{C,\min}} \right) \right\}. \tag{33}$$

Thus

$$|P_{C_1}(k)| < \sqrt{R_C N_{C,\max}} |w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right).$$

Theorem 5 gives

$$0 < \eta_C^m < \frac{2}{y_u^2(k) P_{C_{1,\max}}^2} = \frac{2}{R_C N_{C,\max} y_u^2(k)} \cdot \left[\frac{1}{|w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right)} \right]^2.$$

2) STD: σ_C.

From the previous discussion, we obtain

$$P_{C_2}(k) = \frac{\partial O_C(k)}{\partial \sigma_C} < \sum_{i=1}^{R_C} w_{C_i} \left\{ \sum_{j=1}^{N_{C_j}} \max \left(\frac{\partial O_{C_{ij}}^2}{\partial \sigma_C} \right) \right\}$$
$$= \sum_{i=1}^{R_C} w_{C_i} \left\{ \sum_{j=1}^{N_{C_j}} \max \left(\left(\frac{2}{\sigma_C} \right) \cdot \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \cdot \exp \left[- \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \right] \right) \right\}.$$

Fact 2 gives

$$\left| \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \cdot \exp \left[- \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \right] \right| < 1.$$

Then

$$P_{C_2}(k) < \sum_{i=1}^{R_C} w_{C_i} \left\{ \sqrt{N_{C,\max}} \left(\frac{2}{\sigma_{C,\min}} \right) \right\}. \tag{34}$$

Thus

$$|P_{C_2}(k)| < \sqrt{R_C N_{C,\max}} |w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right).$$

Therefore, from Theorem 5 we find that

$$0 < \eta_C^\sigma < \frac{2}{y_u^2(k) P_{C_{3,\max}}^2} = \frac{2}{R_C N_{C,\max} y_u^2(k)} \cdot \left[\frac{1}{|w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right)} \right]^2.$$

3) Feedback layer's weight θ_C.

From the previous discussion, we obtain

$$P_{C_3}(k) = \frac{\partial O_C(k)}{\partial \theta_C} < \sum_{i=1}^{R_C} w_{C_i} \left\{ \sum_{j=1}^{N_{C_j}} \max \left(\frac{\partial O_{C_{ij}}^2}{\partial \theta_C} \right) \right\}$$
$$= \sum_{i=1}^{R_C} w_{C_i} \left\{ \sum_{j=1}^{N_{C_j}} \max \left(\left(\frac{2O_C^2(k-1)}{\sigma_C} \right) \cdot \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right) \cdot \exp \left[- \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \right] \right) \right\}.$$

Fact 2 gives

$$\left| \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right) \cdot \exp \left[- \left(\frac{x + O_C^2 \theta_C - m_C}{\sigma_C} \right)^2 \right] \right| < 1.$$

Then

$$P_{C_3}(k) < \sum_{i=1}^{R_C} w_{C_i} \left\{ \sqrt{N_{C,\max}} \left(\frac{2}{\sigma_{C,\min}} \right) \right\}. \tag{35}$$

Thus

$$|P_{C_3}(k)| < \sqrt{R_C N_{C,\max}} |w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right).$$

Drawing on Theorem 5, we obtain

$$0 < \eta_C^{\theta} < \frac{2}{y_u^2(k) P_{C_{3,\max}}^2} = \frac{2}{y_u^2(k) R_C N_{C,\max}} \cdot \left[\frac{1}{|w_{C,\max}| \left(\frac{2}{\sigma_{C,\min}} \right)} \right]^2.$$

This completes the proof. ■

ACKNOWLEDGMENT

The authors would like to thank the associate editor and the anonymous referees for their valuable comments and suggestions and their many stimulating and constructive discussions for this paper.

REFERENCES

[1] C. T. Chao, Y. J. Chen, and C. C. Teng, "A similarity measure for fuzzy rule in a fuzzy neural network," *IEEE Trans. Syst., Man, Cybern.*, pt. B, vol. 26, pp. 244–254, Feb. 1996.

[2] D. S. Chen and R. C. Jain, "A robust back propagation learning algorithm for function approximation," *IEEE Trans. Neural Networks*, vol. 5, pp. 467–479, May 1994.

[3] G. Chen, Y. Chen, and H. Ogmen, "Identifying chaotic system via a wiener-type cascade model," *IEEE Trans. Contr. Syst.*, pp. 29–36, Oct. 1997.

[4] 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.

[5] ——, "Fuzzy neural network systems in model reference control systems," in *Neural Network Systems, Techniques and Applications*, C. T. Leondes, Ed. New York: Academic, 1998, vol. 6, pp. 285–313.

[6] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous-time recurrent neural network," *Neural Networks*, vol. 6, pp. 801–806, 1993.

[7] L. Jin, P. N. Nikiforuk, and M. Gupta, "Approximation of discrete-time state-space trajectories usding dynamic recurrent neural networks," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 1266–1270, July 1995.

[8] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Networks*, vol. 6, pp. 144–156, Jan. 1995.

[9] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, pp. 1320–1336, Dec. 1991.

[10] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical system using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, Jan. 1990.

[11] D. E. Rummelhart, G. E. Hinton, and R. J. Williams, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press, 1986, vol. 1, ch. 8.

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

[13] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan, "Memory neural networks for identification and control of dynamic systems," *IEEE Trans. Neural Networks*, vol. 5, pp. 306–319, Feb. 1994.

[14] S. Santini, A. D. Bimbo, and R. Jain, "Block-structured recurrent neural networks," *Neural Networks*, vol. 8, no. 1, pp. 135–147, 1995.

[15] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[16] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computat.*, vol. 1, pp. 270–280, 1989.

[17] T. Yabuta and T. Yamada, "Learning control using neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, Sacramento, CA, Apr. 1991, pp. 740–745.

Ching-Hung Lee was born in Taiwan, R.O.C., in 1969. He received the B.S. and M.S. degree in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1992 and 1994, respectively, and the Ph.D. degree in electrical and control engineering from the same university, in 2000.

He is currently an Assistant Professor in the Department of Electronic Engineering at Lunghwa Institute of Technology, Taoyuan, Taiwan. His main research interests are fuzzy neural systems, fuzzy logic control, neural network, signal processing, nonlinear control systems, and robotics control.

Ching-Cheng Teng was born in Taiwan, R.O.C., in 1938. He received the B.S. degree in electrical engineering from the National Cheng Kung University, Taiwan, in 1961.

From 1991 to 1998, he was the Chairman of the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan. He is currently a Professor in the Department of Electrical and Control Engineering at the same university. His research interests include H^{∞} optimal control, signal processing, and fuzzy neural systems.