–Brief Paper–

FINE TUNING OF MEMBERSHIP FUNCTIONS FOR FUZZY NEURAL SYSTEMS

Ching-Hung Lee and Ching-Cheng Teng

ABSTRACT

This paper presents a new method for fine-tuning the Gaussian membership functions of a fuzzy neural network (**FNN**) to improve approximation accuracy. This method results in special shape membership functions without the convex property. We first recall that any continuous function can be represented by a linear combination of Gaussian functions with any standard deviation. Therefore, the Gaussian membership function in the second layer of the **FNN** can be replaced by several small Gaussian functions; the weighting vectors of this new network (called $\mathbf{FNN_5}$) can then be updated using the back-propagation algorithm. The proposed method can adapt proper membership functions for any nonlinear input/output mapping to achieve highly accurate approximation. Convergence analysis shows that the weighting vectors of the $\mathbf{FNN_5}$ eventually converge to the optimal values. Simulation results indicate that (a) this approach improves approximation accuracy, and (b) that the number of rules can be reduced for any given level of accuracy. For the purpose of illustrating the proposed method, the $\mathbf{FNN_5}$ is also applied to tune PI controllers such that gain and phase margins of the closed-loop system achieve the desired specifications.

KeyWords: Function approximation, fuzzy neural network, PI control.

I.INTRODUCTION

With the rapid development of intelligent system engineering, fuzzy system theory, neural networks, and fuzzy neural networks have attracted much attention. In particular, fuzzy neural network (**FNN**) systems (or neuro-fuzzy systems) have been widely applied in various fields, such as model reference control problems, PID controller tuning, signal processing, etc. [1-3,9-16]. **FNN** systems have the advantage that they can be designed merely based on approximation and linguistic information [1,2,13,14]. It is, then, easy to design an **FNN** system to achieve a satisfactory level of accuracy by manipulating the network structure and parameter learning of the **FNN** [1,2,13,14]. However, FNN systems sometimes require more training processes (or computer time) to achieve a higher level of accuracy based on the membership function and the learning algorithm. The Gaussian membership function of an **FNN** is a symmetric function and, thus, is very impractical [1,2,13,14]. Therefore, it is important to develop a practical method for tuning the membership function so as to improve the approximation accuracy.

This paper deals with the function approximation problem by analyzing the relationship between membership functions and approximation accuracy in **FNN** systems. Our objectives are to find a functional expansion of the Gaussian function and to tune the weight so as to modify the shape. That is, we should show that any Gaussian function can be represented by a linear combination of small Gaussian functions (with small standard deviation values and different mean values). The Gaussian membership function in an **FNN** can then be replaced with several small Gaussian functions. The weighting vectors of this new network (called $\mathbf{FNN_5}$) can also be updated using the back-propagation algorithm. This method can adopt proper membership functions for any nonlinear input/output mapping in order to achieve a high degree of approximation accuracy. Convergence analysis will show that the weighting vectors of the $\mathbf{FNN_5}$ eventually converge to the optimal values. Simulation results will further show that (a) this approach improves the approximation accuracy,

Manuscript received April 13, 1999; revised October 20, 1999; accepted September 26, 2000.

Ching-Hung Lee is with Department of Electrical Engineering, Yuan Ze University, Taoyuam 320, Taiwan.

Ching-Cheng Teng is with Department of Electrical & Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan.

(b) the number of rules can be reduced for any given level of accuracy and (c) the **FNN**$_5$ can be used to tune the PI controller so as to efficiently achieve the specified gain and phase margins.

This paper is organized as follows. Section 2 describes the fuzzy neural network (**FNN**) and radial-basis-function (RBF) network employed here. Section 3, in which a new fuzzy neural network **FNN**$_5$ is presented, contains the main results. We show that the **FNN**$_5$ is a universal approximator and introduce the tuning method for the **FNN**$_5$. Stability analysis is presented in this section to guarantee the convergence of the **FNN**$_5$. Simulation results are presented in Section 4 to highlight the effectiveness of the proposed method. In Section 5, we apply the **FNN**$_5$ to tune a PI controller of unstable processes with time delay. Finally, a conclusion is given in Section 6.

II. PRELIMINARIES

This section briefly reviews the fuzzy neural network and the radial-basis-function network.

2.1 Fuzzy neural network

The fuzzy neural network (**FNN**) is one kind of fuzzy inference system [1,2]. A schematic diagram of the four-layered **FNN** is shown in Fig. 1. The input/output representation is denoted as

$$y_p(x) = \sum_{j=1} w_j \prod_{i=1} \exp[-\frac{(x-m_{ij})^2}{\sigma_{ij}^2}], \tag{1}$$

where m_{ij}, σ_{ij}, and w_j are the mean, standard deviation, and weight of the **FNN**, respectively. Nodes in layer one are input nodes representing input linguistic variables. Nodes in layer two are membership nodes. Here, the Gaussian function is used as the membership function. Each membership node is responsible for mapping an input linguistic variable into a possibility distribution for that variable.

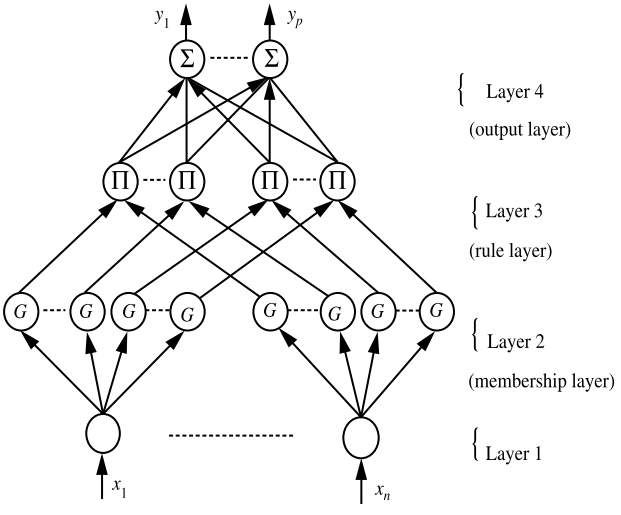

Fig. 1. Structural diagram of the **FNN**.

The rule nodes reside in layer three. The last layer contains the output variable nodes. This is a simple fuzzy logic system implemented by using a multilayer feed-forward neural network. The adjustment of parameters in the **FNN** can be divided into two categories, corresponding to the premise part and the consequence part, of the fuzzy rules. In the premise part, we must initialize the mean and the variance of the Gaussian functions. In the consequence part, the parameters are output singletons. These singletons are initialized with small random values, as in a pure neural network. More details about **FNN**s, convergent theorems and the learning algorithm, can be found in [1,2]. Also, the **FNN** used here has been shown to be a universal approximator. That is, for any given real function $h:\Re^n \to \Re^p$, continuous on a compact set $K \subset \Re^n$, and arbitrary $\varepsilon > 0$, there exists a fuzzy neural network (**FNN**) system $F(\mathbf{x}, W)$, such that $\left\| F(\mathbf{x}, W) - h(\mathbf{x}) \right\| < \varepsilon$ for every x in K.

We emphasize that the used membership functions (gaussian, triangular, trapezoid, etc.) are specific functions [1,2,13,14] within the limit of the convex property. This unreasonable limit should be released. Herein, we propose a method for modifying the membership function so that it has a suitable shape without the convex property.

2.2 Radial-basis-function (RBF) network

In this subsection, the Radial-Basis-Function (RBF) network is introduced. The input/output representation of a RBF network with r inputs and m outputs has the following form:

$$\mathbf{y}(k) = \sum_{i=1}^{M} w_i G\left(\frac{\mathbf{x} - m_i}{\sigma}\right), \tag{2}$$

where $M \in \aleph$, the set of natural numbers, is the number of nodes in the hidden layer, $w_i \in \Re^m$ is the vector of weights

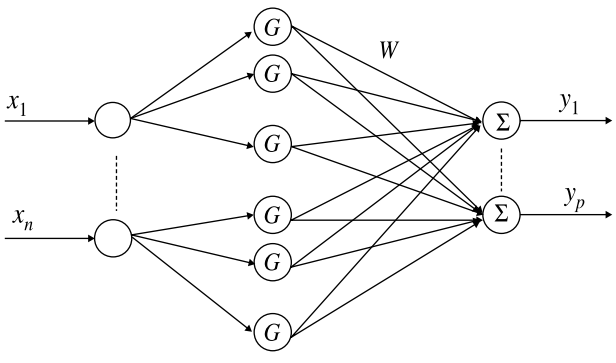

Fig. 2. Structural diagram of the three-layer-network with the Gaussian activation function.

from the *i*th node to the output nodes, **x** is an input vector (an element of $\Re^r$), *G* is a radially symmetric function of a unit in the hidden layer, and m_i and σ_i are the center and width of the *i*th node, respectively. A Gaussian function, $\exp\left(-\frac{(\mathbf{x}-m)^2}{\sigma^2}\right)$, is often used as an activation function, and the width of nodes may be the same or may vary across nodes. In the literature [4,7,17,18], RBF networks having the form (2) have been reported. Some results indicate that, under certain mild conditions on the function *G*, RBF networks represented by (2) (with the same σ_i in each node) are capable of universal approximation.

III. MAIN RESULTS

This section presents the main results, including the approximation theorems and the new fuzzy neural network **FNN**$_5$. The tuning method and convergence analysis of the **FNN**$_5$ are also introduced.

3.1 Gaussian functions

Motivated by the concept of Fourier series, our goal is to obtain a series of Gaussian functions that is the expansion of a membership function. From the previous discussion and literature [4,7,17,18], we can conclude that any Gaussian membership function can be represented by a series of Gaussian functions, i.e., $G(\mathbf{x}; m, \sigma) = \sum_{i=1} w_i G(\mathbf{x}, m_i, \sigma_i)$, where $G(\mathbf{x}, m, \sigma) = \exp[-\frac{(\mathbf{x}-m)^2}{\sigma^2}]$. This result follows from the RBF network's capability of universal approximation. Note that the Gaussian function is a continuous function. Clearly, the RBF network can approximate any Gaussian function. Therefore, any Gaussian function of arbitrary mean and width can be represented by a linear combination of Gaussian functions. In this way, we can tune the membership function to obtain a suitable shape (perhaps not convex) by changing the weighting vector w_i. As in previous studies [1,2,4,7,13,14,17,18] in the literature, the so-called parameter learning of Gaussian function limits change of the center (mean) and width (STD) based on the convex property.

Example 1. Here, we use a series of Gaussian functions to approximate a desired Gaussian function. The interval considered is $[m - 2\sigma, m + 2\sigma]$, and the number of functions in the series, *N*, is set at 17. The back-propagation algorithm is used to update the weighting vector. After 200 epochs, we have the following results (see Fig. 3).

Desired function. $f(x) = e^{-\frac{(x-m)^2}{\sigma^2}}$, where $m = 0$, $\sigma = 2$.

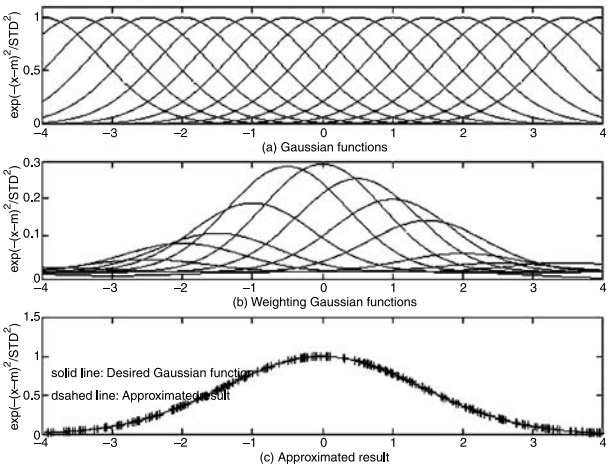

Fig. 3. Gaussian function approximated by several small Gaussian functions.

Gaussian functions. $G_i(x) = e^{-\frac{(x-m_i)^2}{\sigma_i^2}}$,

$$\sigma_i = 1.2,\ m_i = (m - 2\sigma) + \frac{4\sigma}{N-1} i,\ i = 0, \ldots, N-1.$$

Learning rate. 0.1

Mean square error. 9.7×10^{-7}.

Weighting vector.

W = [0.01377 –0.00040 –0.00991 0.03506 0.07881
0.10642 0.18707 0.28713 0.29348 0.25380
0.19719 0.13919 0.05146 0.02494 –0.00039
0.02603 –0.01556].

Figures 3(a) and 3(b) depict the 17 small Gaussian functions and the weighting Gaussian functions, respectively. By summing the weighting Gaussian functions shown in Fig. 3(b), the approximation can be obtained. In Fig. 3 (c), the solid line is the desired function $f(x) = e^{-\frac{(x-l)^2}{2^2}}$, and the plus symbols (+) show the approximated results obtained using Gaussian functions. This simulation result supports our original motivation to use Gaussian functions with adjusted coefficients to form the functional expansion of a membership function. Actually, any continuous function can be represented by a linear combination of Gaussian functions with small standard deviation and different mean values.

3.2 Application in modifying the membership function of the FNN

The main advantage of **FNN** systems is that they can

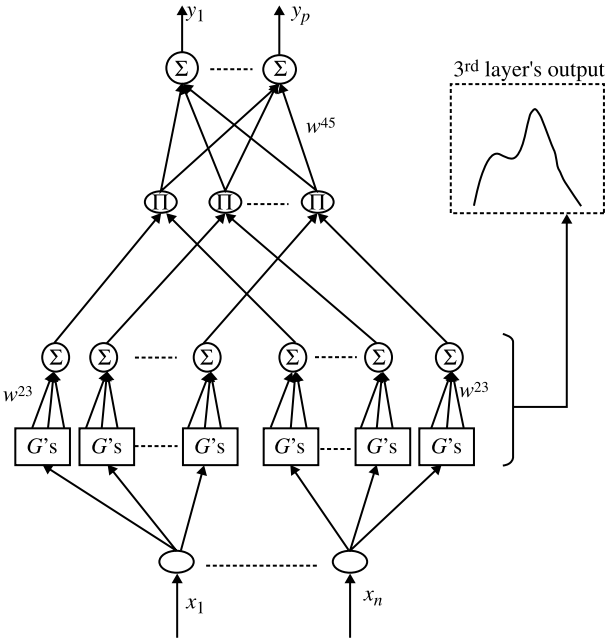

Fig. 4. Structural diagram of the **FNN$_5$** (*G*'s denotes the Gaussian functions).

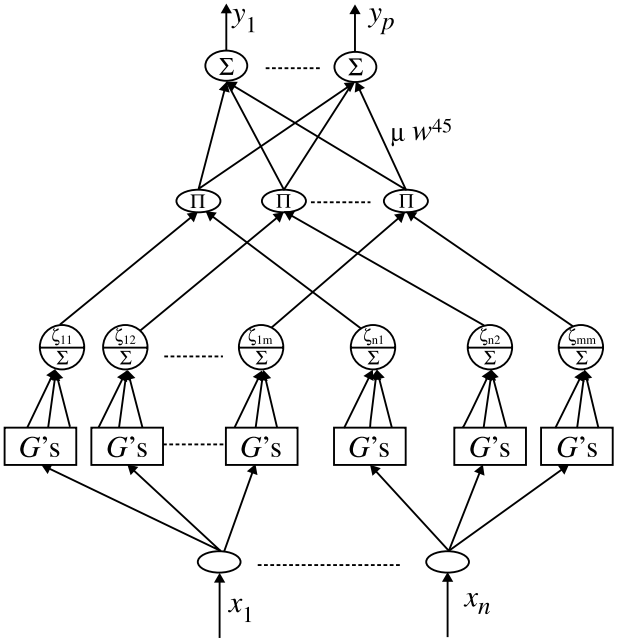

Fig. 5. Structural diagram of the normalized **FNN$_5$** (*G*'s denotes the Gaussian functions)

be designed using only approximation and linguistic information [1,2,9-16]. It is also easy to design an **FNN** system with a satisfactory level of accuracy by choosing the network structure and learning algorithm of the **FNN** appropriately. However, this sometimes requires more training processes (or computer time) involving both the membership function and the learning algorithm. Since the Gaussian membership function is a symmetric function, it is very impractical. Thus, using ideas from the previous discussion and [19], we can proceed to develop a practical method for modifying the membership function so as to improve the approximation accuracy of the **FNN**.

As noted above, any Gaussian function can be represented by a linear combination of Gaussian functions. We, therefore, use these Gaussian functions to replace the second layer membership function of the **FNN** and obtain the following **FNN$_5$** system:

$$y(x(k)) = \sum_{j=1}^{m} w_j^{45} \prod_{i=1}^{m} \sum_{k=1}^{N} w_{ij,k}^{23} \exp\left[-\frac{(x - m_{ij,k})^2}{\sigma_{ij,k}^2}\right]. \quad (3)$$

The architecture of the **FNN$_5$** system is depicted in Fig. 5. This network consists of five layers, in which layers two and three constitute the membership function in the FNN. Note that the **FNN$_5$**, like the **FNN**, is a fuzzy inference system. For the **FNN$_5$**, the adjustable parameters are w^{23} and w^{45}. The previous discussion established that the **FNN** system is a universal approximator, and that the Gaussian membership function in the **FNN** system can be approximated by a series of Gaussian functions. Consequently, we can conclude that the **FNN$_5$** system is also a universal approximator. This is obvious, and a similar proof can be found in [2]. Note that the **FNN$_5$** retains the properties of the FNN-universal approximation and differentiable membership functions.

Tuning the FNN$_5$

In this paper, we use the back-propagation algorithm to tune the weighting vectors $W = [w^{23}\ w^{45}]$. To simplify the description, we will consider the single-output case. Our goal is to minimize the following error function:

$$E(k) = \frac{1}{2}[r(k) - y(k)]^2, \quad (4)$$

where *r* and *y* are outputs of the desired-function and **FNN$_5$** system, respectively. Using the back-propagation algorithm, we can obtain the update laws of *W*:

$$W(k+1) = W(k) + \Delta W(k) \approx W(k) + \eta\left(-\frac{\partial E(k)}{\partial W}\right). \quad (5)$$

From equation (4), the gradients of error function *E* (*k*) with respect to w^{23} and w^{45} are

$$\frac{\partial E}{\partial w^{23}} = [r(k) - y(k)]\frac{\partial e(k)}{\partial w^{23}} = -e(k)\frac{\partial y(k)}{\partial w^{23}}, \quad (6)$$

$$\frac{\partial E}{\partial w^{45}} = [r(k) - y(k)]\frac{\partial e(k)}{\partial w^{45}} = -e(k)\frac{\partial y(k)}{\partial w^{45}}, \quad (7)$$

where $e(k) = r(k) - y(k)$. By the chain-rule, we have the following result:

$$\frac{\partial y(k)}{\partial w_j^{45}} = \prod_i \sum_k w_{ij,k}^{23} \exp\left(-\frac{(x-m_{ij,k})^2}{\sigma^2}\right) = O_j^4(k), \quad \forall j \tag{8}$$

$$\frac{\partial y(k)}{\partial w_{ij,k}^{23}} = w_j^{45} \prod_{i \neq j} \sum w_{ij,k}^{23} \exp\left(-\frac{(x-m_{ij,k})^2}{\sigma^2}\right)$$

$$= w_j^{45} \prod_{i \neq j} O_{ij}^3(k) \cdot O_{ij,k}^2(k), \quad \forall i,j,k. \tag{9}$$

Thus, we obtain the update laws for the **FNN$_5$**:

$$w_{ij,k}^{23}(k+1) = w_{ij,k}^{23}(k) + \eta_{w_{23}} e(k) w_j^{45} \prod_{i \neq j} O_{ij}^3 \cdot O_{ij,k}^2(k), \tag{10}$$

$$w_j^{45}(k+1) = w_j^{45}(k) + \eta_{w_{45}} e(k) O_j^4(k). \tag{11}$$

This completes tuning of the **FNN$_5$**.

Convergence analysis

Define the discrete Lyapunov function

$$V(k) = \frac{1}{2}[r(k) - y(k)]^2. \tag{12}$$

Let $e(k) = r(k) - y(k)$ be the approximation error. From [8], we know that the difference in error can be approximated as

$$\Delta e(k) \approx \frac{\partial e(k)}{\partial w^{23}} \Delta w^{23} + \frac{\partial e(k)}{\partial w^{45}} \Delta w^{45}, \tag{13}$$

where

$$\Delta w_{ij,k}^{23} = -\eta_{w23} e(k) \frac{\partial e(k)}{\partial w_{ij,k}^{23}} = \eta_{w23} e(k) w_j^{45} \prod_{i \neq j} O_{ij}^3 \cdot O_{ij,k}^2(k),$$

$$\Delta w_j^{45} = -\eta_{w45} e(k) \frac{\partial e(k)}{\partial w_j^{45}} = \eta_{w45} e(k) O_j^4(k).$$

Therefore,

$$\Delta e(k) \approx \frac{\partial e(k)}{\partial w^{23}} \eta_{w23} e(k) \frac{\partial y(k)}{\partial w^{23}} + \frac{\partial e(k)}{\partial w^{45}} \eta_{w45} e(k) \frac{\partial y(k)}{\partial w^{45}}$$

$$= -\eta_{w23} e(k) \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 - \eta_{w45} e(k) \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2. \tag{14}$$

Theorem 1. Let $\eta_{w^{23}}$ and $\eta_{w^{45}}$ be the learning rates for w^{23} and w^{45}, respectively. The asymptotical convergence of the **FNN$_5$** system is guaranteed if the following inequality holds:

$$\eta_{w^{23}} \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 + \eta_{w^{45}} \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2 < 2. \tag{15}$$

Proof. We rewrite the Lyapunov function as

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

The gradients of the Lyapunov function with respect to the weighting vectors w^{23} and w^{45} are

$$\frac{\partial V(k)}{\partial w^{23}} = e(k) \frac{\partial e(k)}{\partial w^{23}} - e(k) \frac{\partial y(k)}{\partial w^{23}}, \tag{16}$$

$$\frac{\partial V(k)}{\partial w^{45}} = e(k) \frac{\partial e(k)}{\partial w^{45}} - e(k) \frac{\partial y(k)}{\partial w^{45}}. \tag{17}$$

The change in the Lyapunov function is, then,

$$\Delta V = \frac{1}{2}[e^2(k+1) - e^2(k)] = \Delta e(k)[e(k) + \frac{1}{2}\Delta e(k)]$$

$$= -\left[\eta_{w23} e(k) \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 + \eta_{w45} e(k) \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2\right]$$

$$\cdot \left\{ e(k) - \frac{1}{2}\left[\eta_{w23} e(k) \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 + \eta_{w45} e(k) \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2\right] \right\}$$

$$= -\frac{1}{2} e^2(k) \left[\eta_{w23} e(k) \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 + \eta_{w45} e(k) \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2\right]$$

$$\cdot \left\{ 2 - \left[\eta_{w23} \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 + \eta_{w45} \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2\right] \right\}$$

$$\equiv -\lambda e^2(k),$$

where

$$\lambda = \frac{1}{2}\left[\eta_{w23} \left|\frac{\partial y(k)}{\partial w^{23}}\right|^2 + \eta_{w45} \left|\frac{\partial y(k)}{\partial w^{45}}\right|^2\right]$$

$$\cdot \left\{ 2 - \left[\eta_{w^{23}} \left| \frac{\partial y(k)}{\partial w^{23}} \right|^2 + \eta_{w^{45}} \left| \frac{\partial y(k)}{\partial w^{45}} \right|^2 \right] \right\}. \quad (18)$$

Therefore, if we choose proper learning rates $\eta_{w^{23}}$, $\eta_{w^{45}}$ such that $\lambda > 0$, then .

$$\Delta V(k) = -\lambda e^2(k) \le 0.$$

This means that convergence of the system is guaranteed if

$$\eta_{w^{23}} \left| \frac{\partial y(k)}{\partial w^{23}} \right|^2 + \eta_{w^{45}} \left| \frac{\partial y(k)}{\partial w^{45}} \right|^2 < 2, \text{ for all } k.$$

■

The general convergence condition (15) can now be applied to find the specific convergence criterion for each learning rate.

Theorem 2. Let $\eta = \eta_{w^{23}} = \eta_{w^{45}}$ be the learning rates for w^{23} and w^{45}. Asymptotical convergence of the **FNN$_5$** system is guaranteed if the following inequality holds:

$$0 < \eta < \frac{2}{\max\limits_j w_j^{45} \cdot \left(\max\limits_{i,j,k} w_{ij,k}^{23} \cdot M \right)^{m-1} + \left(\max\limits_{i,j,k} w_{ij,k}^{23} \cdot M \right)^m}, \quad (19)$$

where M and m denote layer numbers, two and four, respectively.

Proof. From Theorem 1, we have the following condition:

$$\eta < \frac{2}{\left| \frac{\partial y(k)}{\partial w^{23}} \right|^2 + \left| \frac{\partial y(k)}{\partial w^{45}} \right|^2} = \frac{2}{\left(w_j^{45} \prod\limits_{i \ne j} O_{ij}^3 \cdot O_{ij,k}^2 \right)^2 + \left(O_j^4 \right)^2}. \quad (20)$$

The representation of **FNN$_5$** ($O^4 = \prod\limits_i O_i^3$, and $O^3 = \sum\limits_i w_i^{23} O_i^2$) gives

$$w_j^{45} \prod_{i \ne j} O_{ij}^3 \cdot O_{ij,k}^2 \le \max_j w_j^{45} \cdot \left(\max_{i,j,k} w_{ij,k}^{23} \cdot M \right)^{m-1}. \quad (21)$$

Thus,

$$0 < \eta < \frac{2}{\max\limits_j w_j^{45} \cdot \left(\max\limits_{i,j,k} w_{ij,k}^{23} \cdot M \right)^{m-1} + \left(\max\limits_{i,j,k} w_{ij,k}^{23} \cdot M \right)^m}.$$

■

Theorems 1 and 2 imply that any learning rate $\eta_{w^{23}} \left| \frac{\partial y(k)}{\partial w^{23}} \right|^2 + \eta_{w^{45}} \left| \frac{\partial y(k)}{\partial w^{45}} \right|^2 < 2$ guarantees convergence. However, the maximum learning rate, which guarantees the most rapid or optimal convergence, satisfies

$$\eta^* \left(\left| \frac{\partial y(k)}{\partial w^{23}} \right|^2 + \left| \frac{\partial y(k)}{\partial w^{45}} \right|^2 \right) = 1,$$

which is half of the upper limit in equation (19) [8].

Normalization of membership function

It is well known that the **FNN** is a network with a fuzzy inference engine [1,2]. However, the **FNN$_5$** may not be a fuzzy inference engine because after the tuning process, the membership output may not be a membership function; i.e., the output may be greater than one or negative. Therefore, the output signal of the third layer must be normalized to [0,1]. For each membership function output, we first find the maximum and minimum of the membership output over the interval $[m - 2\sigma, m + 2\sigma]$. Next, the shifting value ζ_{ij} is selected so as to move the minimum to the origin ($\zeta_{ij} = -\min$). Then, a scaling value μ_j is chosen for normalization ($\mu_j = 1/(\max - \min)$). Therefore, the normalization process contains both scaling and shifting. However, the scaling factor μ_j can be transferred to w_j^{45}, and the shifting value ξ_{ij} can be described as the activation function of nodes in the third layer. Therefore, the **FNN$_5$** is represented by means of the following equation (22) and Fig. 5:

$$y(x(k)) = \sum_{j=1}^{n} \mu_j \cdot w_j^{45} \prod_{i=1}^{m} \left(\zeta_{ij} + \sum_{k=1}^{N} \bar{w}_{ij,k}^{23} \exp\left[-\frac{(x_i - m_{ij,k})^2}{\sigma_{ij,k}^2} \right] \right). \quad (22)$$

Obviously, (22) is still a neural network with a fuzzy inference engine; i.e., (22) is a fuzzy neural network. Note that, though the network structure of **FNN** was modified to obtain **FNN$_5$**, it still has the properties of fuzzy inference, universal approximation, and differentiable membership functions. Also, the learning algorithm and convergence theorems have been successfully extended to the **FNN$_5$**.

IV. SIMULATION RESULTS

This section presents two examples of approximating a desired function. The results demonstrate the effectiveness of the proposed method. To simplify the network structure, we adopt nine Gaussian functions to replace each membership function in layer two. Adaptive learning rates are used, starting from the initial rates $\eta_{w^{23}} = 0.1$ and

$\eta_{w^{45}} = 0.1$. Both learning rates are adjusted according to the criterion for η^* developed in Section 3. Simulation results show that these Gaussian functions help improve the approximation accuracy.

Example 2. Approximation of a Step Function (SISO Case)

Mean square error after 1000 epochs

FNN with 9 fuzzy rules	2.2×10^{-3}
FNN with 16 fuzzy rules	4.5×10^{-4}
$\mathbf{FNN_5}$ with 9 fuzzy rules	1.7×10^{-4}

Figures 6(a) and 6(b) show simulation results and modification of the membership functions, respectively.

Example 3. Approximation of a 2-D Function

Consider a 2-D function given by

$$f(x_1, x_2) = \frac{\sin \pi \sqrt{2(x_1^2 + x_2^2) + 1}}{2}.$$

Mean square error after 1000 epochs

FNN with 9 fuzzy rules	8.1×10^{-3}
FNN with 16 fuzzy rules	2.1×10^{-3}
$\mathbf{FNN_5}$ with 9 fuzzy rules	3.1×10^{-4}

The adaptive learning rates of $\eta^*_{w^{23}}$ and $\eta^*_{w^{45}}$ for both Examples 2 and 3 are shown in Fig. 7.

Remark 1. Examples 2 and 3 show improvement in the approximation accuracy of the $\mathbf{FNN_5}$ relative to that of the **FNN**. Compared with the **FNN**, the $\mathbf{FNN_5}$ can also reduce the number of fuzzy inference rules for the same approximated specification. However, this approach uses more Gaussian nodes than the **FNN**, thus enlarging the network structure.

V. APPLICATION: TUNING A PI CONTROLLER FOR UNSTABLE PROCESSES

Consider the *n*th-order unstable process with time-delay

$$G_p(s) = \frac{K_p(1 + w_{n1}s)^{n1}(1 + w_{n2}s)^{n2} \cdots (1 + w_{nq}s)^{nq}}{(1 + w_{d1}s)^{d1}(1 + w_{d2}s)^{d2} \cdots (1 + w_{dp}s)^{dp}} e^{-Ls}, \tag{23}$$

where at least one w_{di} is negative and $n = \sum_{i=1}^{p} di$. The open-loop step response of the process is unbounded since it has a pole in the right half plane. To have a stable closed-loop system, the PI controller, given by

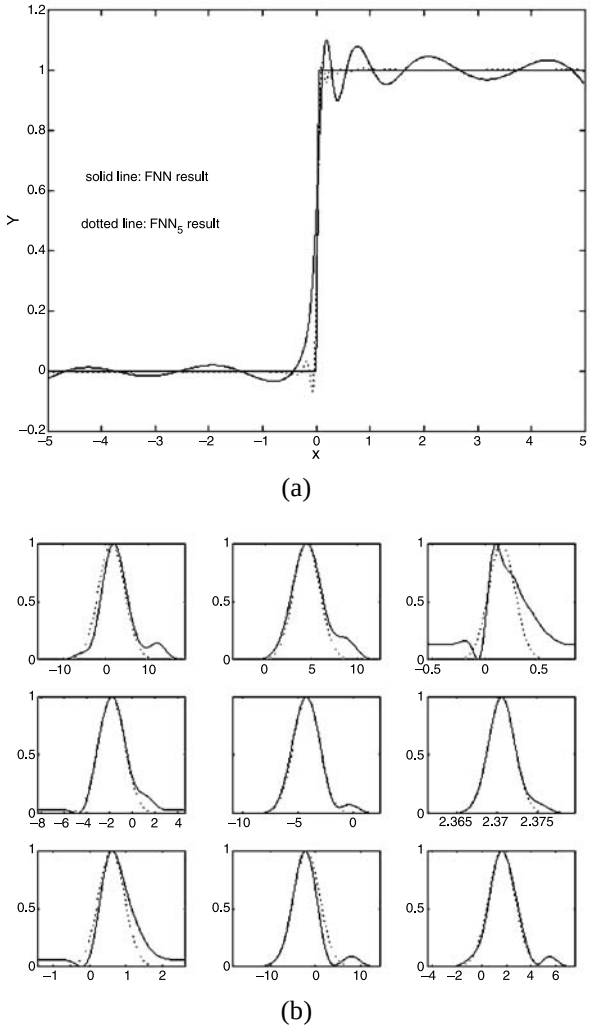

Fig. 6. (a): Simulation results for Example 2 (b): Modification of membership functions (dotted line: **FNN**, solid line: $\mathbf{FNN_5}$).

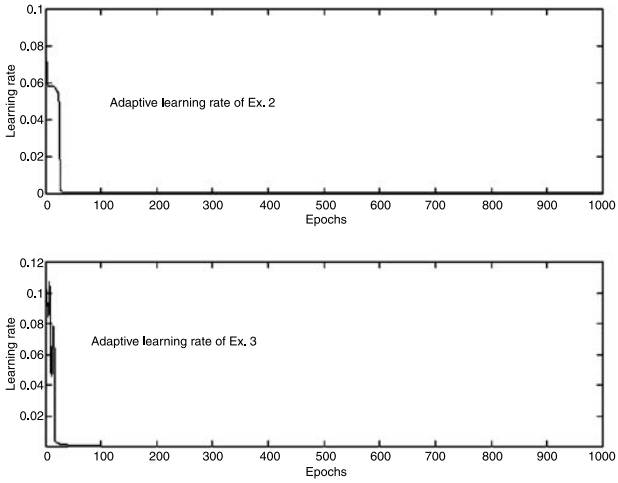

Fig. 7. Adaptive learning rates for Examples 2 and 3.

$$G_c(s) = K_c(1 + \frac{1}{sT_I}), \tag{24}$$

must be used to satisfy the Nyquist criterion. The frequency w_p, at which the Nyquist curve has a phase of $-\pi$, is known as the phase-crossover frequency; the frequency w_g, at which the Nyquist curve has an amplitude of 1, is the gain-crossover frequency. Let the specified gain and phase margins be denoted by A_m and ϕ_m, respectively. The formulas for the gain margin and phase margin are as follows:

$$\arg[G_c(jw_p)G_p(jw_p)] = -\pi, \tag{25}$$

$$A_m = \frac{1}{\left| G_c(jw_p)G_p(jw_p)] \right|}, \tag{26}$$

$$\left| G_c(jw_g)G_p(jw_g)] \right| = 1, \tag{27}$$

$$\phi_m = \arg[G_c(jw_g)G_p(jw_g)] + \pi, \tag{28}$$

where the gain margin is defined by Eqs. (25) and (26), and the phase margin by Eqs. (27) and (28). The loop-transfer function is obtained from

$$G_c(s)G_p(s) = \frac{K_cK_p(1+sT_I)(1+w_{n1}s)^{n1}(1+w_{n2}s)^{n2}\cdots(1+w_{nq}s)^{nq}}{sT_I(1+w_{d1}s)^{d1}(1+w_{d2}s)^{d2}\cdots(1+w_{dp}s)^{dp}}e^{-Ls}.$$

Substituting the above equation into equations (25)-(28), we have

$$\begin{aligned}&\frac{1}{2}\pi + \tan^{-1}(w_pT_I) + n_1\tan^{-1}(w_pw_{n1}) + \cdots + n_q\tan^{-1}(w_pw_{nq})\\ &\quad - w_pL - d_1\tan^{-1}(w_pw_{d1}) - d_2\tan^{-1}(w_pw_{d2})\\ &\quad - \cdots - d_p\tan^{-1}(w_pw_{dp}) = 0,\end{aligned} \tag{29}$$

$$A_mK_cK_p = w_pT_I\frac{\sqrt{1+w_p^2T_I^2}\sqrt{(1+w_p^2w_{n1}^2)^{n1}}\cdots\sqrt{(1+w_p^2w_{nq}^2)^{nq}}}{\sqrt{(1+w_p^2w_{d1}^2)^{d1}}\sqrt{(1+w_p^2w_{d2}^2)^{d2}}\cdots\sqrt{(1+w_p^2w_{dp}^2)^{dp}}}, \tag{30}$$

$$K_cK_p = w_gT_I\frac{\sqrt{(1+w_g^2w_{d1}^2)^{d1}}\sqrt{(1+w_g^2w_{d2}^2)^{d2}}\cdots\sqrt{(1+w_g^2w_{dp}^2)^{dp}}}{\sqrt{(1+w_g^2T_I^2)}\sqrt{(1+w_g^2w_{n1}^2)^{n1}}\cdots\sqrt{(1+w_g^2w_{nq}^2)^{nq}}}, \tag{31}$$

$$\begin{aligned}\phi_m = &\frac{1}{2}\pi + \tan^{-1}(w_gT_I) + n_1\tan^{-1}(w_gw_{n1}) + \cdots\\ &+ n_q\tan^{-1}(w_gw_{nq}) - w_gL - d_1\tan^{-1}(w_gw_{d1})\\ &- d_2\tan^{-1}(w_gw_{d2}) - \cdots - d_p\tan^{-1}(w_gw_{dp}).\end{aligned} \tag{32}$$

For a given process (K_p, w_{n1}, …, w_{nq}, w_{d1}, …, w_{dp}, L) and specifications (A_m, ϕ_m), Eqs. (29)-(32) can be solved for the PI controller parameters (K_c, T_I) and crossover frequencies (w_g, w_p) numerically but not analytically because of the presence of the arctan function. In 1995, Ho *et al*. first proposed a tuning method for PID controllers based on gain margin and phase margin (GPM) specifications [5]. Later, in 1998, Ho and Xu presented an approach to tuning unstable processes using GPM specifications [6]. They adopted linear equations to approximate the arctan function in the gain-phase margin formulas. The disadvantage of their method [5,6] is that the transfer function of the controlled process is restricted to the first-order-plus-time-delay type. This restriction was relaxed by Chu and Teng [3].

In our final simulation, we use the **FNN$_5$** to tune a PI controller for unstable processes. Figure 8 shows the block diagram of the function mapping of Eqs. (29)-(32) using **FNN$_5$** for an unstable process (first order with time delay). If we are given (A_m, ϕ_m) and have R^i(i = 1, …, n^2) implications, then the value of $y \in \{K_c, T_I\}$ is determined by the **FNN$_5$**.

Here we will consider two examples (first-order and 2nd-order unstable processes with time delay), which are as follows:

$$\text{First order: } G_p(s) = \frac{e^{-0.2s}}{s-1}; \tag{33}$$

$$\text{Second order: } G_p(s) = \frac{100e^{-0.05s}}{s^2+10s-5}. \tag{34}$$

Various gain and phase margins are specified for models (33) and (34) in Table 1. The **FNN$_5$** yields errors of less than 3.98% and 2.73% from the desired gain and phase margin specifications. Note that, since the process in (34) is not first-order, the approximation method used in [6] cannot be applied to (34). Summarizing our simulation results for the two examples, we conclude that the **FNN$_5$** may be used to automatically tune the PID controller parameters for different GPM specifications, and that neither numerical methods nor graphical methods need be used. The simulation results show that the **FNN$_5$** can achieve the specified values efficiently. Note that this

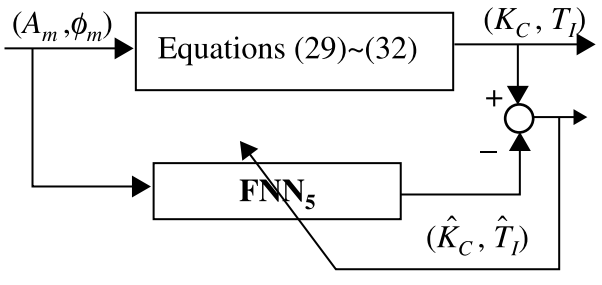

Fig. 8. Block diagram of function mapping using **FNN$_5$**.

Table 1. Simulation results: PI controllers for different models.

Model	Specifications		Result						Error	
	A_m	ϕ_m^*	K_c	T_I	A_m^*	ϕ_m^*	w_g	w_p	Error of A_m	Error of ϕ_m^*
$\frac{e^{-0.2s}}{s-1}$	2	30	3.3957	3.7473	2.0796	30.6933	6.9041	3.2548	3.98%	2.31%
	3	35	2.3110	4.4956	3.0725	34.2976	0.5429	2.0960	2.41%	2.00%
	4	30	1.7853	3.8733	3.9598	29.2811	0.5883	1.5068	1.00%	2.39%
$\frac{100e^{-0.05s}}{s^2+10s-5}$	2	25	1.0387	6.3869	2.0468	25.6844	12.7372	7.8934	2.34%	2.73%
	3.2	40	0.6878	10.9548	3.1121	39.0761	12.7996	5.7311	2.74%	2.30%
	4	45	0.5546	7.9801	3.8459	44.3897	12.7671	4.7867	3.85%	1.85%

approach is applicable even if the controlled process is stable, unstable and/or of higher order.

VI. CONCLUSION

This paper has presented a new method for fine-tuning the membership function of the fuzzy neural network (**FNN**) to improve the approximation accuracy. This method generates special shape membership functions without the convex property. Our motivation is based on functional expansion of a Gaussian membership function. We conclude that any Gaussian function can be represented by a linear combination of small Gaussian functions. Therefore, the Gaussian membership functions in the second layer of the **FNN** are replaced by several small Gaussian functions. The weighting vectors of the **FNN** can then be updated using the back-propagation algorithm. Using this approach, the membership functions of the **FNN** are transformed into proper (non-symmetric) functions, thus improving the approximation accuracy. The Lyapunov stability approach has benn used in convergence analysis that guarantees the stability of the $\mathbf{FNN}_5$, i. e., that the weighting vectors converge to the optimal values. Simulation results show that (a) the approximation accuracy is higher with this approach, and that (b) the number of rules can be reduced for any given degree of accuracy. The simulation results also show that the $\mathbf{FNN}_5$, when used to tune the PI controller of a stable or unstable process, efficiently achieves the specified gain and phase margins.

ACKNOWLEDGEMENT

The authors wish to thank Prof. Li-Chen Fu and the reviewers for their constructive remarks and suggestions. This work is supported by the National Science Council, Taiwan, R.O.C. under Grant NSC89-2213-E009-21.

REFERENCES

1. Chen, Y.C. and C.C. Teng, "A Model Reference Control Structure Using a Fuzzy Neural Network," *Fuzzy Sets Syst.*, Vol. 73, pp. 291-312 (1995).
2. Chen, Y.C. and C.C. Teng, "Fuzzy Neural Network Systems in Model Reference Control Systems," in *Neural Network Systems: Technique and Applications*, Vol. 6, Ed. C.T. Leondes, Academic Press, Inc., pp. 285-313 (1998).
3. Chu, S.Y. and C.C. Teng, "Tuning of PID Controllers Based on Gain and Phase Margin Specifications Using Fuzzy Neural Network," *Fuzzy Sets Syst.*, Vol. 101, pp. 21-30 (1999).
4. Hartman, E.J., J.D. Keeler and J.M. Kowalski, "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations," *Neural Comput.*, Vol. 2, pp. 210-215 (1990).
5. Ho, W.K., C.C. Hang and L.S. Cao, "Tuning of PID Controllers Based on Gain and Phase Margin Specification," *Automatica*, Vol. 31, pp. 497-502 (1995).
6. Ho, W.K. and W. Xu, "PID Tuning for Unstable Processes Based on Gain and Phase-margin Specifications," *IEE Proc. Contr. Theory Appl.*, Vol. 145, pp. 392-396 (1998).
7. Hornik, K.M., M. Stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366 (1989).
8. Ku, C.C. and K.Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," *IEEE Trans. Neural Networks*, Vol. 6, No. 1, pp. 144-156 (1995).
9. Lee, C.H. and C.C. Teng, "Control of Nonlinear Systems Using Fuzzy Neural Networks," *Proc. Asian Fuzzy Syst. Symp.*, Kenting, Taiwan, pp. 49-54 (1996).
10. Lee, C.H. and C.C. Teng, "Identification and Control of Hammerstein Systems Using a Fuzzy Neural Network," *J. Signal Process.*, Vol. 4, No. 2, pp. 149-157 (2000).
11. Lee, C.H. and C.C. Teng, "A Novel Method to Identify Nonlinear Dynamic Systems," *Eur. Contr. Conf.*, Karlsruhe, Germany (1999).
12. Lee, C.H. and C.C. Teng, "Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks," *IEEE Trans. Fuzzy Syst.*, Vol. 8, No. 4, pp. 349-366 (2000).
13. Lin, C.T., "A Neural Fuzzy Control System with

structure and Parameter Learning," *Fuzzy Sets Syst.*, Vol. 70, pp. 183-212 (1995).
14. Lin, C.T. and C.S.G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice-Hall International, Inc. (1996).
15. Ma, C.W. and C.C. Teng, "Tracking a Near-field Moving Target Using Fuzzy Neural Networks," *Fuzzy Sets Syst.*, accepted (1998).
16. Ma, C.W. and C.C. Teng, "A Fuzzy Neural Network Approach for 2-D Direction Finding in Multipath Environments," *IEE Proc. Radar Sonar Navig.*, Vol. 146, No. 2, pp. 78-83 (1999).
17. Park, J. and I.W. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Comput.*, Vol. 3, pp. 246-257 (1991).
18. Poggio, T. and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, Vol. 78, No.9, pp. 1481-1497 (1990).
19. Zeng, X.J. and M.G. Singh, "A Relationship Between Membership Functions and Approximation Accuracy in Fuzzy Systems," *IEEE Trans. Syst., Man Cybern.-Part B: Cybern.*, Vol. 26, No. 1, pp.176-180 (1996).