Dynamical Optimal Training for Interval Type-2 Fuzzy Neural Network (T2FNN)

Chi-Hsu Wang, *Senior Member, IEEE*, Chun-Sheng Cheng, and Tsu-Tian Lee, *Fellow, IEEE*

***Abstract*—Type-2 fuzzy logic system (FLS) cascaded with neural network, type-2 fuzzy neural network (T2FNN), is presented in this paper to handle uncertainty with dynamical optimal learning. A T2FNN consists of a type-2 fuzzy linguistic process as the antecedent part, and the two-layer interval neural network as the consequent part. A general T2FNN is computational-intensive due to the complexity of type 2 to type 1 reduction. Therefore, the interval T2FNN is adopted in this paper to simplify the computational process. The dynamical optimal training algorithm for the two-layer consequent part of interval T2FNN is first developed. The stable and optimal left and right learning rates for the interval neural network, in the sense of maximum error reduction, can be derived for each iteration in the training process (back propagation). It can also be shown both learning rates cannot be both negative. Further, due to variation of the initial MF parameters, i.e., the spread level of uncertain means or deviations of interval Gaussian MFs, the performance of back propagation training process may be affected. To achieve better total performance, a genetic algorithm (GA) is designed to search optimal spread rate for uncertain means and optimal learning for the antecedent part. Several examples are fully illustrated. Excellent results are obtained for the truck backing-up control and the identification of nonlinear system, which yield more improved performance than those using type-1 FNN.**

***Index Terms*—Back propagation, dynamic optimal learning rate, genetic algorithm, interval type-2 FNN.**

I. INTRODUCTION

DURING the past decade, intelligent methodologies have been found to possess the best potential to solve many engineer problems which cannot be solved before. Especially the fuzzy neural network (FNN) has been explored during the past few years by many researchers to equip the intelligent methodologies with better learning capabilities. For instance, the FNN has been applied successfully to control nonlinear, ill-defined systems [1]. In particular, the back propagation (BP) of FNN has been developed to tune the parameters of fuzzy sets and the weighting factors of neural network in [1]. The BP algorithm is applied to minimize the difference (error) between the desired and actual outputs through iterations. For each iteration, the parameters and weighting factors are adjusted by the BP algorithm in order to reduce the error along a descent direction. A reasonable learning rate should be assigned during the BP process. Therefore the dynamic optimization of learning rate for type-1 FNN has been proposed to accelerate the convergence of the BP algorithm [2], [3]. Moreover the analysis of stable and optimal learning rates for type-1 FNN was also discussed rigorously in [3]. However, all of these discussion and analyses are focused on type-1 FNN. To date, type-2 fuzzy sets and fuzzy logic controller have been used in decision making [4], survey processing [5]–[7], time-series forecasting [8], time-varying channel equalization [9], [10], control of mobile robots [11], and preprocessing of data [12]. Further genetic algorithms (GAs) was adopted in [3] to fine-tune the Gaussian MFs in the antecedent part of type-1 FNN. The authors in [13] also applied GAs to search for optimal uncertain means and its extent of interval type-2 Gaussian MFs for the chaotic time-series prediction. Although many reasonable results have been obtained by using BP process or GAs, the discussion of stable and optimal learning rates has not been established in type-2 FNN (T2FNN).

Due to the learning capability of type-1 FNN [14], T2FNN can be similarly defined. We proposed an interval T2FNN that consists of the interval type-2 fuzzy linguistic process as the antecedent part and the two-layer interval NN as the consequent part. The two-layer interval neural network consists of left and right weighting factors which will require left and right learning rates during the learning process. The T2FNN is computational intensive due to the complexity of type 2 to type 1 reduction. Therefore, the interval T2FNN is adopted in this paper to simplify the computational process. The result of type reduction process, called type-reduced set, possesses more important information than a crisp output of type-1 FNN. The stability analysis of the left and right learning rates for this two-layer interval NN will be discussed. A new theorem will be proposed to yield the dynamic optimal learning rates for this two-layer interval NN, which guarantees the maximum error reduction during the BP process. It can also be shown that the left and right learning rates for the interval neural network cannot both be negative. It is not necessary that both learning rates are positive, but they cannot be both negative. For comparison purpose, the dynamical optimal learning rate for type 1 FNN should be positive [3].

Since the variations of parameters setting in the type-2 MFs, i.e., the spread of uncertain means or deviations, will affect total performance during the BP training process. In order to find the optimal settings of uncertain means or deviations in the interval T2FNN, a genetic algorithm is also proposed together with dynamical optimal BP training process to search for optimal spread rate of MFs and optimal learning rate in antecedent part simultaneously. In the meantime, the dynamic optimal learning rate of two-layer neural network of consequent part also can be obtained for each iteration. The well-known ex-

Manuscript received March 3, 2003; revised August 27, 2003. This work was supported by the Ministry of Education, Taiwan, R.O.C., under the Project "Intelligent Transportation System," Grant 91X104EX-91-E-FA06-4-4 (April 2003–2004). This paper was recommended by Associate Editor D. S. Yeung.

The authors are with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan 300, R.O.C. (e-mail: cwang@cn.nctu.edu.tw).

Digital Object Identifier 10.1109/TSMCB.2004.825927

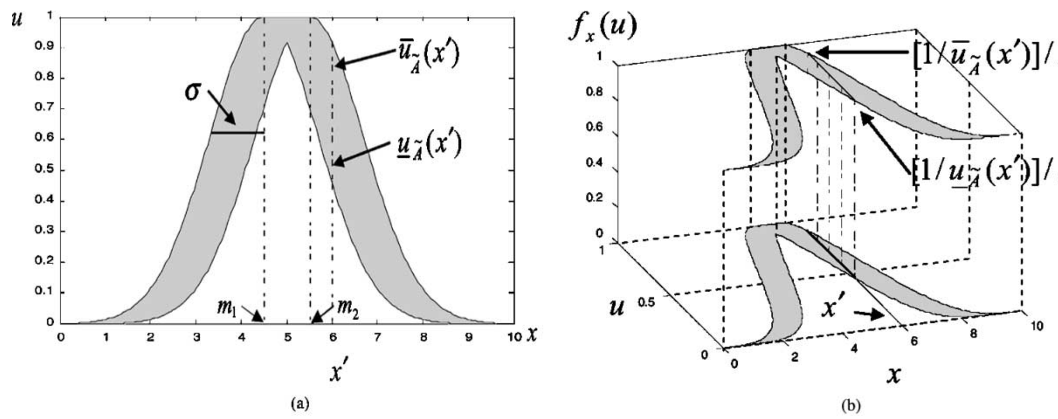


Fig. 1. (a) Interval type-2 fuzzy set with uncertain mean. (b) Three-dimensional membership function for interval type-2 fuzzy set.

amples of truck backing-up and nonlinear system identification will be illustrated via our new optimally trained interval T2FNN with GAs to yield more improved performances than those using type-1 FNN.

This paper is organized as follows. In Section II, a type-2 fuzzy neural network model will be defined. In Section III, the dynamic optimal learning theorem with BP process will be developed to tune the interval T2FNN. Section IV describes how to find optimal spread rate and learning rate via genetic algorithm. Section V shows two applications via dynamic optimal learning theorem with GA. The conclusions and topics for future research are drawn in Section VI.

II. Interval Type-2 Fuzzy Neural Network (T2FNN)

In this section, the interval type-2 fuzzy set and the inference of type-2 fuzzy logic system will be described first. This will lead to interval type-2 fuzzy neural network (T2FNN).

A. Type-2 Fuzzy Logic System (T2FLS)

A type-2 fuzzy set in universal set X is denoted as $\tilde{A}$ which is characterized by a type-2 membership function $u_{\tilde{A}}(x)$ in (1). The $u_{\tilde{A}}(x)$ can be referred as a secondary membership function or also referred as a secondary set, which is a type-1 fuzzy set in [0, 1]. In (1), $f_x(u)$ is a secondary grade, which is the amplitude of a secondary membership function; i.e., $0 \le f_x(u) \le 1$. The domain of a secondary membership function is called the primary membership of x. In (1), J_x is the primary membership of x, where $u \in J_x \subseteq [0,1]$ for $\forall x \in X$; u is a fuzzy set in [0, 1], rather than a crisp point in [0, 1].

$$\tilde{A} = \int_{x\in X} u_{\tilde{A}}(x) \Big/ x = \int_{x\in X} \left[\int_{u \in J_x} f_x(u) \Big/ u \right] \Big/ x \qquad J_x \subseteq [0,1]. \quad (1)$$

When $f_x(u) = 1, \forall u \in J_x \subseteq [0,1]$, then the secondary MFs are interval sets such that $u_{\tilde{A}}(x)$ in (1) can be called an interval type-2 MF [6]. Therefore the type-2 fuzzy set $\tilde{A}$ can be re-expressed as

$$\tilde{A} = \int_{x\in X} u_{\tilde{A}}(x) \Big/ x = \int_{x\in X} \left[\int_{u \in J_x} 1 \Big/ u \right] \Big/ x \qquad J_x \subseteq [0,1]. \quad (2)$$

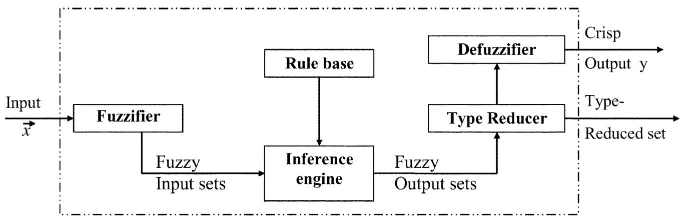


Fig. 2. Type-2 fuzzy logic system.

Also, a Gaussian primary MF with uncertain mean and fixed standard deviation having an interval type-2 secondary MF can be called an interval type-2 Gaussian MF (3). Fig. 1(a) shows a 2-D interval type-2 Gaussian MF with an uncertain mean in $[m_1, m_2]$ and a fixed deviation σ. It can be stated as

$$u_{\tilde{A}}(x) = \exp\left[-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2 \right], \quad m \in [m_1, m_2]. \quad (3)$$

It is obvious that the type-2 fuzzy set is in a region, called a footprint of uncertainty (FOU), and bounded by an upper MF and a lower MF [6], which are denoted as $\bar{u}_{\tilde{A}}(x)$ and $\underline{u}_{\tilde{A}}(x)$, respectively. Both of them are two type-1 MFs. Hence, (2) can be re-stated as

$$\tilde{A} = \int_{x\in X} \left[\int_{u \in [\underline{u}_{\tilde{A}}(x), \bar{u}_{\tilde{A}}(x)]} 1 \Big/ u \right] \Big/ x. \quad (4)$$

We will make great use of upper and lower MFs for type reduction in this section and develop the dynamic optimal learning rate algorithm in next section. Also the interval type-2 Gaussian MF with uniform uncertainty at primary memberships of x in Fig. 1(a)–(b) will be adopted in this paper.

A type-2 FLS in Fig. 2 is constructed by the same structure of type-1 IF-THEN rules, which is still dependent on the knowledge of experts. Expert knowledge, however, is always represented by linguistic terms and implied uncertainty, which leads to the rules of type-2 FLSs having uncertain antecedent part and/or consequent part; then translate into uncertain antecedent or consequent MFs. The structure of rules in the type-2 FLS and its inference engine is similar to those in type-1 FLS. The inference engine combines rules and provides a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets. To achieve this process, we must find unions and intersections of type-2 sets, as well as compositions of type-2 relations. The output of

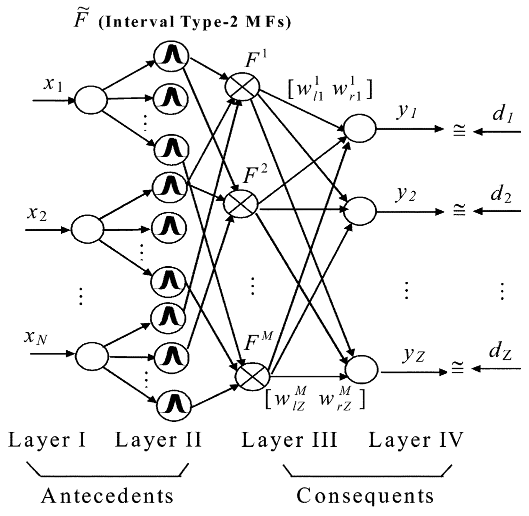


Fig. 3. Interval T2FNN with antecedent part and consequent part.

the type-2 inference engine is a type-2 set. Using Zadeh's extension principle [15], type-1 defuzzification can derive a crisp output from type-1 fuzzy set; similarly, for a higher type set as type-2, this operation derives the type-2 sets to a type-1 set. This process can be so called "type reduction." The complete type-2 fuzzy logic theory with the handling of uncertainties, such as the operations on type-2 fuzzy sets, centroid of a type-2 fuzzy sets, type-reduction, ..., etc., can be found in [16]–[21].

B. Type-2 Fuzzy Neuro Network

Due to the complexity of type reduction, the general type-2 FLS becomes computationally intensive. An interval type-2 FLS, whose secondary MFs are all unity, make things simpler and easier to compute meet and join operations, which leads finally to simplify type reduction. An interval T2FNN system is shown on Fig. 3, which is an implementation of interval type-2 fuzzy logic system, and some of their parameters and components are presented by fuzzy logic terms. Like type-1 FNN, Fig. 3 is a typical FNN with four layers structure [1]. Input nodes and type-2 fuzzification nodes are drawn on layer I and layer II, respectively. They form the antecedent part of this T2FNN. Consequent parts are drawn on layer III and IV which are constructed from a classical 2-layer NN with fuzzy rule nodes and output nodes. The fuzzifier nodes in layer II will yield type-2 membership grades. Each node at layer III is a fuzzy rule. Layer III nodes consist of the preconditions of the rule, i.e., the firing strength F^i from (6) as shown in the following context. Layer IV nodes define the consequences of the rule nodes. The links between layer III and layer IV consist of interval weighting factors which will decide the actual outputs of this system.

The IF-THEN rule for interval T2FNN can be expressed as

$$\begin{aligned} R^i &: \text{IF } x_1 \text{ is } \tilde{F}_1^i, \text{ and } \ldots, \text{and } x_n \text{ is } \tilde{F}_n^i, \\ &\text{THEN } y_1 \text{ is } [w_{l1}^i w_{r1}^i], \text{ and } \ldots, \text{and } y_Z \text{ is } [w_{lZ}^i w_{rZ}^i] \end{aligned} \quad (5)$$

where $i = 1, 2, \ldots, \mathrm{M}$ is rule number, the $\tilde{F}_n^i$ is the interval type-2 fuzzy sets of antecedent part, and $[w_{lz}^i w_{rz}^i], z = 1, \ldots, Z$, is a centroid set with unity membership grade (interval type-1 fuzzy set), which can be called weighting interval set, derived from interval type-2 fuzzy set in the consequent part [6], [17]. Both w_{lz}^i and w_{rz}^i are treated as weighting factors to fully connect layer III and layer IV in our interval T2FNN structure. In practical use, both w_{lz}^i and w_{rz}^i can also be set at random initially in a reasonable interval. This structure combines type-2 fuzzification in antecedent part with a random weighting interval set in consequent part. It cannot only totally represents a type-2 fuzzy logic relation, but it can also process type-reduction and lead to the development a dynamic optimal training in consequent part which will be shown later in Section III.

C. Type Reduction

In Fig. 3, we only consider singleton input fuzzification throughout this paper. Similar to type-1 FNN, the firing strength F^i in (6) can be obtained by the following inference process:

$$F^i = \coprod_{\vec{x} \in X} \left[\prod_{k=1}^{n} u_{\tilde{F}_k^i}(x_k) \right] \quad (6)$$

where $\prod$ is the meet operation and $\coprod$ is the join operation [6].

For Gaussian interval type-2 fuzzy set as shown in Fig. 1, the upper MF is a subset that has the maximum membership grade and the lower MF is a subset that has the minimum membership grade. The join operation in (6) leads to join the result from above meet operations using supremum (i.e., maximum value), the result F^i can be an interval type-1 set [20] as

$$F^i = \begin{bmatrix} \underline{f}^i \\ \bar{f}^i \end{bmatrix} \quad (7)$$

where

$$\begin{aligned} \underline{f}^i &= \underline{u}_{\tilde{F}_1^i}(x_1) * \cdots * \underline{u}_{\tilde{F}_n^i}(x_n) \quad \text{and} \\ \bar{f}^i &= \bar{u}_{\tilde{F}_1^i}(x_1) * \cdots * \bar{u}_{\tilde{F}_n^i}(x_n). \end{aligned} \quad (8)$$

The center-of-sets type-reduction [6], [20] will be used in this paper. In order to simplify the notation, we consider single output here. Then we have the center-of-sets type reduction method (as shown the bottom of the page) where $y_{\cos}(\vec{x})$ is also an interval type-1 set determined by left and right end points (y_l and y_r), which can be derived from consequent centroid set

$$y_{\cos}(\vec{x}) = [y_l, y_r] = \int_{w^1 \in [w_l^1, w_r^1]} \cdots \int_{w^M \in [w_l^M, w_r^M]} \int_{f^1 \in [\underline{f}^1, \bar{f}^1]} \cdots \int_{f^M \in [\underline{f}^M, \bar{f}^M]} 1 \Bigg/ \frac{\sum_{i=1}^{M} f^i w^i}{\sum_{i=1}^{M} f^i} \quad (9)$$

$[w_l^i \ w_r^i]$ and firing strengths $f^i \in F^i = [\underline{f}^i, \bar{f}^i]$. The interval set $[w_l^i \ w_r^i]$ $(i = 1, \ldots, M)$ should be computed or set first before the computation of $y_{\cos}(\vec{x})$. For any value $y \in y_{\mathrm{COS}}$, y can be expressed as

$$y = \frac{\sum_{i=1}^M f^i w^i}{\sum_{i=1}^M f^i} \tag{10}$$

where y is a monotonic increasing function with respect to w^i. Also, y_l in (9) is the minimum associated only with w_l^i, and y_r in (9) is the maximum associated only with w_r^i. Note that y_l and y_r depend only on mixture of $\underline{f}^i$ or $\bar{f}^i$ values. Hence, left-most point y_l and right-most point y_r can be expressed as [9]

$$y_l = \frac{\sum_{i=1}^M f_l^i w_l^i}{\sum_{i=1}^M f_l^i} \tag{11}$$

and

$$y_r = \frac{\sum_{i=1}^M f_r^i w_r^i}{\sum_{i=1}^M f_r^i}. \tag{12}$$

For illustrative purposes, the type-reduction algorithm for computing y_r from ([6], P. 310–311) is listed below as Algorithm 1.

Algorithm 1. Type Reduction for Interval T2FNN: Without loss of generality, assume the w_r^i's are arranged in ascending order, i.e., $w_r^1 \le w_r^2 \le \cdots \le w_r^M$.

```
[Step 1]: Compute $y_r$ in (12) by initially using $f_r^i = (\underline{f}^i + \bar{f}^i)/2$ for $i = 1, \ldots, M$, where
    $\underline{f}^i$ and $\bar{f}^i$ are pre-computed by (8); and let $y_r' = y_r$.
[Step 2]: Find $R (1 \le R \le M - 1)$ such that $w_r^R \le y_r' \le w_r^{R+1}$.
[Step 3]: Compute $y_r$ in (12) with $f_r^i = \underline{f}^i$ for $i \le R$ and $f_r^i = \bar{f}^i$ for $i > R$, then set
    $y_r'' = y_r$.
[Step 4]: If $y_r'' \ne y_r'$, then go to Step 5. If $y_r'' = y_r'$, then set $y_r = y_r''$ and go to Step 6.
[Step 5]: Let $y_r' = y_r''$ and return to Step 2.
[Step 6]: End.
```

This algorithm decides the point to separate two sides by the number R, one side using lower firing strengths $\underline{f}^i$'s and another side using upper firing strengths $\bar{f}^i$'s. Hence, the y_r in (12) can be re-expressed as

$$\begin{aligned} y_r &= y_r\left(\underline{f}^1, \ldots, \underline{f}^R, \bar{f}^{R+1}, \ldots, \bar{f}^M, w_r^1, \ldots, w_r^M\right) \\ &= \frac{\sum_{i=1}^R \underline{f}^i w_r^i + \sum_{i=R+1}^M \bar{f}^i w_r^i}{\sum_{i=1}^R \underline{f}^i + \sum_{i=R+1}^M \bar{f}^i} \\ &= \sum_{i=1}^R \underline{q}_b^i w_r^i + \sum_{i=R+1}^M \bar{q}_b^i w_r^i \end{aligned} \tag{13}$$

where $\underline{q}_b^i = \underline{f}^i/D_r, \bar{q}_b^i = \bar{f}^i/D_r$ and $D_r = 1/(\sum_{i=1}^R \underline{f}^i + \sum_{i=R+1}^M \bar{f}^i)$.

The procedure to compute y_l is similar to compute y_r. In step 2, it only needs to find $L(1 \le L \le M - 1)$, such that $w_l^L \le y_l' \le w_l^{L+1}$. In step 3, let $f_l^i = \bar{f}^i$ for $i \le L$ and $f_l^i = \underline{f}^i$ for $i > L$. y_l in (11) can be also re-expressed as

$$\begin{aligned} y_l &= y_l\left(\bar{f}^1, \ldots, \bar{f}^L, \underline{f}^{L+1}, \ldots, \underline{f}^M, w_l^1, \ldots, w_l^M\right) \\ &= \frac{\sum_{i=1}^L \bar{f}^i w_l^i + \sum_{i=L+1}^M \underline{f}^i w_l^i}{\sum_{i=1}^L \bar{f}^i + \sum_{i=L+1}^M \underline{f}^i} \\ &= \sum_{i=1}^L \bar{q}_a^i w_l^i + \sum_{i=L+1}^M \underline{q}_a^i w_l^i \end{aligned} \tag{14}$$

where $\bar{q}_a^i = \bar{f}^i/D_l, \underline{q}_a^i = \underline{f}^i/D_l$ and $D_l = 1/(\sum_{i=1}^L \bar{f}^i + \sum_{i=L+1}^M \underline{f}^i)$.

The defuzzified crisp output from an interval type-2 FLS is the average of y_l and y_r, i.e.,

$$y(\vec{x}) = \frac{y_l + y_r}{2}. \tag{15}$$

According to the above analysis, the defuzzified output $y(\vec{x})$, i.e., actual output, is determined only by the upper and lower antecedent MFs and the weighting interval set. Like type-1 FNN, we also can use the BP method to tune all the parameters of type-2 fuzzy MFs in T2FNN. However, the process of tuning the parameters of interval T2FNN is more complicated than those in type-1 FNN. We must first determine the parameters associated wth y_l and y_r. This requires comparing x_k $(\mathrm{k} = 1, \ldots, \mathrm{n})$ to some points associated with parameters of upper and lower antecedent MFs [6]. When the input x_k is located in one segment of domain, then its corresponding MF branch is called active branch. For instance, $x' = 6$ in Fig. 1(a), we have two respective active upper and lower MFs branches. Once these parameters are changed due to tuning, the dependency of y_l and y_r on parameters may also be changed, i.e., the active branches may be changed to the other branches. The tuning of the parameters of these active branches are the same as tuning the parameters in type-1 FNN. For instance, to tune the parameters of the active branches located in the any upper or lower MF, we can have the following details.

By using the back propagation method, for P input-output training data $(\vec{x}^p : d^p), p = 1, \ldots, P$ the following error function should be minimized:

$$e^p = \frac{1}{2}[y(\vec{x}^p) - d^p]^2 \quad p = 1, \ldots, P. \tag{16}$$

To tune the mean m_k^i of Gaussian MF in the ith rule [6] as

$$\begin{aligned} m_k^i(p+1) &= m_k^i(p) - \alpha \left.\frac{\partial e^p}{\partial m_k^i}\right|_p \\ &= m_k^i(p) - \frac{1}{2}\alpha(y(\vec{x}^p) - d^p) \\ &\quad \times \left[\frac{(w_{lr}^i - y_{lr})}{\prod_{k=1}^N u^*_{\tilde{F}_k^i}}\left(x_k - m_k^i\right) \times \right. \\ &\quad \left. N\left(m_k^i, \sigma_k^i; x_k\right) \Big/ \left(\sigma_k^i\right)^2 \right] \end{aligned} \tag{17}$$

where $m_k^i \in [m_k^i, m_{k2}^i]$ and $N(m_k^i, \sigma_k^i; x_k) \equiv [-(1/2)((x_k - m_k^i)/(\sigma_k^i))^2]$.

Similarly, to tune standard deviation σ_k^i and weighting factor w_{lr}^i, we have

$$\sigma_k^i(p+1) = \sigma_k^i(p) - \frac{1}{2}\alpha(y(\vec{x}^p) - d^p) \times \left[\frac{\left(w_{lr}^i - y_{lr}\right)}{\prod_{k=1}^{N} u^*_{\tilde{F}_k^i}} \left(x_k - m_k^i\right)^2 \times N\left(m_k^i, \sigma_k^i; x_k\right) \Big/ \left(\sigma_k^i\right)^3 \right] \quad (18)$$

$$w_{lr}^i(p+1) = w_{lr}^i(p) - \frac{1}{2}\alpha(y(\vec{x}^p) - d^p) \left[\frac{N\left(m_k^i, \sigma_k^i; x_k\right)}{\prod_{k=1}^{N} u^*_{\tilde{F}_k^i}} \right] \quad (19)$$

where α is learning rate for tuning the parameters of MFs. Whereas, w_{lr}^i and y_{lr} can be w_l^i or w_r^i and y_l or y_r respectively, and $u^*_{\tilde{F}_k^i}$ can be $\underline{u}_{\tilde{F}_k^i}$ or $\bar{u}_{\tilde{F}_k^i}$. The weighting factor w_l^i or w_r^i depends on which branch is active in the process calculating left-most point y_l or right-most point y_r (13)–(14). Based on both type-reduction and BP processes, a dynamic optimal learning algorithm for tuning weighting matrices of consequents will be developed to fasten the convergence of back propagation process in the next section. An example to tune the parameters by using back propagation process in (17)–(19) is illustrated as follows.

Example 1: The following interval T2FNN has three rules in which each rule has two antecedent parts and two consequent parts (i.e., MIMO—multiple inputs and multiple outputs):

$$R^1: \text{IF } x_1 \text{ is } \tilde{F}_1^1 \text{ and } x_2 \text{ is } \tilde{F}_2^1 \quad \text{THEN } y_1 \text{ is } \left[w_{l1}^1 \quad w_{r1}^1\right] \text{ and } y_2 \text{ is } \left[w_{l2}^1 \quad w_{r2}^1\right] \quad (20)$$

$$R^2: \text{IF } x_1 \text{ is } \tilde{F}_1^2 \text{ and } x_2 \text{ is } \tilde{F}_2^2 \quad \text{THEN } y_1 \text{ is } \left[w_{l1}^2 \quad w_{r1}^2\right] \text{ and } y_2 \text{ is } \left[w_{l2}^2 \quad w_{r2}^2\right] \quad (21)$$

and

$$R^3: \text{IF } x_1 \text{ is } \tilde{F}_1^3 \text{ and } x_2 \text{ is } \tilde{F}_2^3 \quad \text{THEN } y_1 \text{ is } \left[w_{l1}^3 \quad w_{r1}^3\right] \text{ and } y_2 \text{ is } \left[w_{l2}^3 \quad w_{r2}^3\right]. \quad (22)$$

where two antecedents are Gaussian primary MFs with uncertain mean. We extend type-1 Gaussian MFs by its deviation ratio 0.5 to form interval type-2 Gaussian MFs. The original type-1 Gaussian MFs are

$$\begin{bmatrix} m_1^1 \\ m_1^2 \\ m_1^3 \end{bmatrix} = \begin{bmatrix} 5.5 \\ 4.5 \\ 6.2 \end{bmatrix}, \quad \begin{bmatrix} m_2^1 \\ m_2^2 \\ m_2^3 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 6.0 \\ 5.1 \end{bmatrix}$$
$$\begin{bmatrix} \sigma_1^1 \\ \sigma_1^2 \\ \sigma_1^3 \end{bmatrix} = \begin{bmatrix} 1.30 \\ 1.10 \\ 0.80 \end{bmatrix}, \quad \begin{bmatrix} \sigma_2^1 \\ \sigma_2^2 \\ \sigma_2^3 \end{bmatrix} = \begin{bmatrix} 1.20 \\ 1.00 \\ 1.50 \end{bmatrix}.$$

The extended type-2 MFs are: (with same fixed deviations)

$$\begin{bmatrix} m_{11}^1 & m_{12}^1 \\ m_{11}^2 & m_{12}^2 \\ m_{11}^3 & m_{12}^3 \end{bmatrix} = \begin{bmatrix} 4.85, & 6.15 \\ 3.95, & 5.05 \\ 5.80, & 6.60 \end{bmatrix}$$
$$\begin{bmatrix} m_{21}^1 & m_{22}^1 \\ m_{21}^2 & m_{22}^2 \\ m_{21}^3 & m_{22}^3 \end{bmatrix} = \begin{bmatrix} 2.40, & 3.60 \\ 5.50, & 6.50 \\ 4.35, & 5.85 \end{bmatrix}.$$

The weighting matrices of consequent part (20)–(22) are initially at random assumed as

$$\begin{bmatrix} w_{l1}^1 & w_{l2}^1 \\ w_{l1}^2 & w_{l2}^2 \\ w_{l1}^3 & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.4730, & 5.3356 \\ 1.0210, & 0.1271 \\ 4.1400, & 0.3272 \end{bmatrix}$$
$$\begin{bmatrix} w_{r1}^1 & w_{r2}^1 \\ w_{r1}^2 & w_{r2}^2 \\ w_{r1}^3 & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.8062, & 8.9973 \\ 5.3044, & 8.4949 \\ 5.4016, & 1.4851 \end{bmatrix}$$

and given four pairs of training data ($\boldsymbol{X} : \boldsymbol{D}$) as

$$\boldsymbol{X} = \begin{bmatrix} \vec{x}^1 \\ \vec{x}^2 \\ \vec{x}^3 \\ \vec{x}^4 \end{bmatrix} = \begin{bmatrix} 4.7, & 6.0 \\ 6.1, & 3.9 \\ 2.9, & 4.2 \\ 7.0, & 5.5 \end{bmatrix}$$
$$\boldsymbol{D} = \begin{bmatrix} 3.52, & 4.02 \\ 5.43, & 6.23 \\ 4.95, & 5.76 \\ 4.70, & 4.28 \end{bmatrix}.$$

For example, the interval Gaussian type-2 MFs of antecedent part in rule R^1 are shown in Fig. 4(a)–(b). The interval type-1 set of weighing factors $[w_l \ w_r]$ in consequent part for the first output are also shown in Fig. 4(c). We examine the first training pair $\vec{x}^1 = [4.7 \ 6.0]$ and $d_{11} = 3.52$. Given $\vec{x}^1$ into this MIMO interval T2FNN, by using product t-norm, we can obtain three interval type-1 sets of firing strength as

$$\begin{aligned} F^1 &= [\underline{f}^1 \quad \bar{f}^1] = \left[\underline{u}_{\tilde{F}_1^1} * \underline{u}_{\tilde{F}_2^1} \quad \bar{u}_{\tilde{F}_1^1} * \bar{u}_{\tilde{F}_2^1}\right] \\ &= [0.5368 \times 0.0111 \quad 0.9934 \times 0.1353] \\ &= [0.0060 \quad 0.1344] \\ F^2 &= [\underline{f}^2 \quad \bar{f}^2] = \left[\underline{u}_{\tilde{F}_1^2} * \underline{u}_{\tilde{F}_2^2} \quad \bar{u}_{\tilde{F}_1^2} * \bar{u}_{\tilde{F}_2^2}\right] \\ &= [0.7926 \times 0.8825 \quad 1.000 \times 1.000] \\ &= [0.6995 \quad 1.0000] \\ F^3 &= [\underline{f}^3 \quad \bar{f}^3] = \left[\underline{u}_{\tilde{F}_1^3} * \underline{u}_{\tilde{F}_2^3} \quad \bar{u}_{\tilde{F}_1^3} * \bar{u}_{\tilde{F}_2^3}\right] \\ &= [0.0596 \times 0.5461 \quad 0.3886 \times 0.9960] \\ &= [0.0325 \quad 0.3866]. \end{aligned}$$

After firing the consequent part, we have upper and lower interval type-1 sets, which form interval type-2 fuzzy sets. Fig. 4(d) shows the three-dimensional (3-D) view of these interval type-2 fuzzy sets. The type-reduction procedure in Algorithm 1 can be applied to find the right-most point y_r from these fired type-2 interval sets in Fig. 4(d). Similarly the left-most point y_l can also be found. Then the type-reduced set $[y_l \ y_r]$ for this pair training data is $[1.1328 \ 5.8514]$. As a result, we have $y(\vec{x}^1) = 3.4921$ as shown in Fig. 4(e). By using fixed learning rate $\alpha = 0.2$ in (17)–(19), after 15 iterations, we

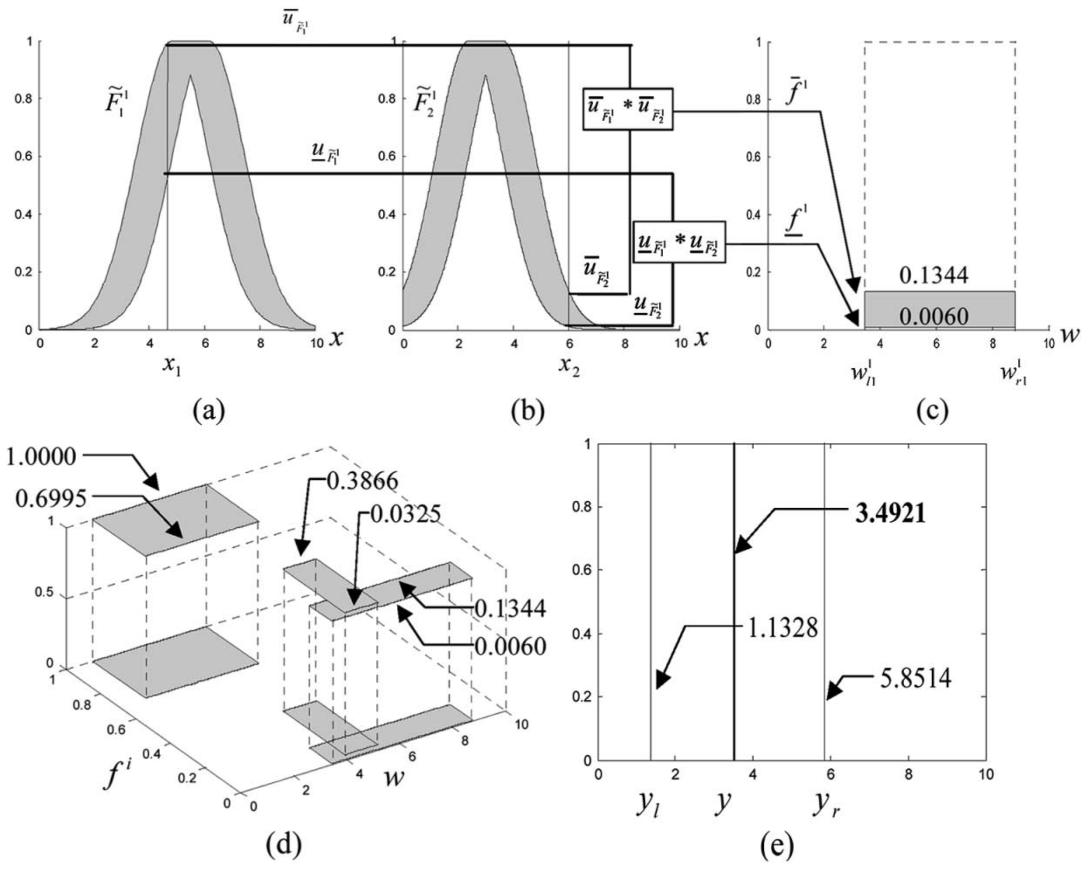


Fig. 4. Two interval type-2 Gaussian MFs of antecedent part for rule R^1 are shown in (a) and (b). The weighing interval sets and corresponding fired interval type-2 sets for first output are shown in (c). The 3-D view of interval type-2 sets is shown in (d). The result $y(\vec{x}^1) = 3.4921$ shown in (e).

have the final means, standard deviations, weighting matrices and actual output as

$$\begin{bmatrix} m_{11}^1 & m_{12}^1 \\ m_{11}^2 & m_{12}^2 \\ m_{11}^3 & m_{12}^3 \end{bmatrix} = \begin{bmatrix} 5.0969, & 6.0139 \\ 4.0881, & 5.1321 \\ 5.7985, & 6.9088 \end{bmatrix}$$

$$\begin{bmatrix} m_{21}^1 & m_{22}^1 \\ m_{21}^2 & m_{22}^2 \\ m_{21}^3 & m_{22}^3 \end{bmatrix} = \begin{bmatrix} 2.7441, & 3.6038 \\ 5.6162, & 6.4427 \\ 4.6381, & 6.2564 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_1^1 \\ \sigma_1^2 \\ \sigma_1^3 \end{bmatrix} = \begin{bmatrix} 1.8127 \\ 1.1720 \\ 0.7055 \end{bmatrix} \quad \begin{bmatrix} \sigma_2^1 \\ \sigma_2^2 \\ \sigma_2^3 \end{bmatrix} = \begin{bmatrix} 1.2716 \\ 0.9567 \\ 0.6086 \end{bmatrix}$$

$$\begin{bmatrix} w_{l1}^1 & w_{l2}^1 \\ w_{l1}^2 & w_{l2}^2 \\ w_{l1}^3 & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.1776, & 5.5488 \\ 0.9235, & -0.1359 \\ 4.1174, & 0.6569 \end{bmatrix}$$

$$\begin{bmatrix} w_{r1}^1 & w_{r2}^1 \\ w_{r1}^2 & w_{r2}^2 \\ w_{r1}^3 & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.3457, & 9.5789 \\ 5.3613, & 8.1266 \\ 5.3898, & 1.5516 \end{bmatrix}$$

$$y(\vec{x}) = \begin{bmatrix} 3.4348, & 4.1999 \\ 5.5248, & 6.2312 \\ 5.0183, & 5.7612 \\ 4.6845, & 4.2916 \end{bmatrix}.$$

Then the trajectory of total squared errors J is plotted in Fig. 5.

We only apply back-propagation algorithm in Example 1 with fixed learning rate. Better training results using dynamic optimal learning rates can be seen in the next section.

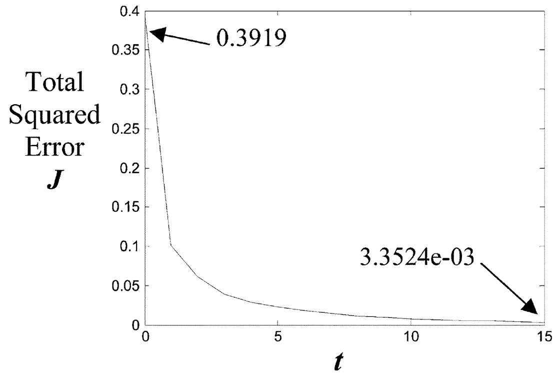


Fig. 5. Total squared error $\boldsymbol{J}$ versus iteration $\boldsymbol{t}$ for a fixed learning rate $\alpha = 0.2$.

III. Dynamic Optimal Learning Rate of Interval Type-2 Fuzzy Neural Network

According to [3], authors have developed the dynamic optimal learning rate theorem to speed up the convergence of tuning weighting factors of consequent part in type-1 FNN. From the type-reduction process in Algorithm 1 with (13)–(14), we have

$$y_l = \sum_{i=1}^{L} \bar{q}_a^i w_l^i + \sum_{i=L+1}^{M} \underline{q}_a^i w_l^i; \quad \text{and}$$

$$y_r = \sum_{i=1}^{R} \underline{q}_b^i w_r^i + \sum_{i=R+1}^{M} \bar{q}_b^i w_r^i.$$

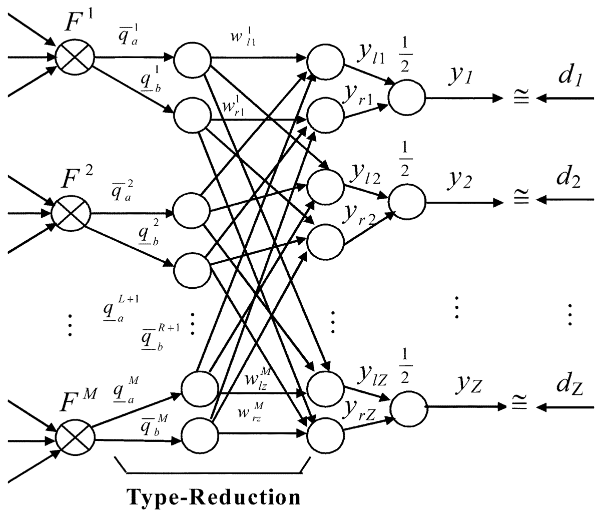


Fig. 6. Detailed look of the consequent part in Fig. 3.

Then it is obvious that we can interpret the above equations as an interval NN which is shown in layers III and IV of Fig. 3 and is presented in more details in Fig. 6. The goal is then to find the dynamic optimal training for tuning weighting interval sets $[w_l^i \;\; w_r^i]$ in Fig. 6. The i th node input of layer III is firing strength F^i from layer II. Once all firing strengths enter into the type-reduction process, $\bar{q}_a^i, \underline{q}_a^i, \underline{q}_b^i$, and $\bar{q}_b^i$ in (13)–(14) can be determined via Algorithm 1 to find y_l and y_r. A dynamic optimal learning algorithm for T2FNN will be developed in this section to guarantee maximum error reduction during the back propagation process in previous section, where

$$F = [F^1, \quad F^2 \quad \cdots \quad F^M] = \begin{bmatrix} \underline{f}^1, & \underline{f}^2 & \cdots & \underline{f}^M \\ \bar{f}^1 & \bar{f}^2 & \cdots & \bar{f}^M \end{bmatrix} \in \Re^{2\times M}$$

the firing strength matrix (23)

$$W = \begin{bmatrix} \vec{w}_{l1}, & \vec{w}_{l2} & \cdots & \vec{w}_{lZ} \\ \vec{w}_{r1}, & \vec{w}_{r2} & \cdots & \vec{w}_{rZ} \end{bmatrix} \in \Re^{2\times M\times Z}$$

the weighting matrix (24)

$$\vec{w}_{lz} = \left[w_{lz}^1, \quad w_{lz}^2 \quad \cdots \quad w_{lz}^M\right]^T \in \Re^M$$

the zth left weighting vector (25)

$$\vec{w}_{rz} = \left[w_{rz}^1, \quad w_{rz}^2 \quad \cdots \quad w_{rz}^M\right]^T \in \Re^M$$

the zth right weighting vector (26)

$$\vec{q}_l = \left[\bar{q}_a^1, \quad \cdots \quad \bar{q}_a^L \quad \underline{q}_a^{L+1} \quad \cdots \quad \underline{q}_a^M\right]^T \in \Re^M$$

the left firing strength vector (27)

$$\vec{q}_r = \left[\underline{q}_b^1, \quad \cdots \quad \underline{q}_b^R \quad \bar{q}_b^{R+1} \quad \cdots \quad \bar{q}_b^M\right]^T \in \Re^M$$

the right firing strength vector (28)

$$\vec{y} = [y_1, \quad y_2 \quad \cdots \quad y_Z]^T \in \Re^Z$$

the actual output vector (29)

$$\vec{d} = [d_1, \quad d_2 \quad \cdots \quad d_Z]^T \in \Re^Z$$

the desired output vector (30)

and "T" denotes matrix transpose, "$\rightarrow$" denotes vector.

To derive the actual output, we need first to compute its left-most (14) and right-most (13) output as

$$y_{lz} = \sum_{i=1}^{L} \bar{q}_a^i w_l^i + \sum_{i=L+1}^{M} \underline{q}_a^i w_l^i = \vec{q}_l^T \vec{w}_{lz} \tag{31}$$

and

$$y_{rz} = \sum_{i=1}^{R} \underline{q}_b^i w_r^i + \sum_{i=R+1}^{M} \bar{q}_b^i w_r^i = \vec{q}_r^T \vec{w}_{rz}. \tag{32}$$

Therefore, the actual output y_z can be obtained as

$$y_z = \frac{1}{2}(y_{lz} + y_{rz}). \tag{33}$$

Then, we have $\vec{y} = [y_1 \;\; y_2 \;\; \cdots \;\; y_Z]^T$ as (29) by union all outputs.

Given P training vectors, its actual output Y, and the desired output D as

$$Q_l = [\vec{q}_l^1 \quad \vec{q}_l^2 \quad \cdots \quad \vec{q}_l^P] \in \Re^{M\times P}$$

the left firing strength matrix (34)

$$Q_r = [\vec{q}_r^1 \quad \vec{q}_r^2 \quad \cdots \quad \vec{q}_r^P] \in \Re^{M\times P}$$

the right firing strength matrix (35)

$$W_l = [\vec{w}_{l1} \quad \vec{w}_{l2} \quad \cdots \quad \vec{w}_{lZ}] \in \Re^{M\times Z}$$

the left weighting factor matrix (36)

$$W_r = [\vec{w}_{r1} \quad \vec{w}_{r2} \quad \cdots \quad \vec{w}_{rZ}] \in \Re^{M\times Z}$$

the right weighting factor matrix (37)

$$Y = [\vec{y}_1 \quad \vec{y}_2 \quad \cdots \quad \vec{y}_P]^T \in \Re^{P\times Z}$$

the actual output matrix (38)

$$D = [\vec{d}_1 \quad \vec{d}_2 \quad \cdots \quad \vec{d}_P]^T \in \Re^{P\times Z}$$

the desired output matrix. (39)

From (33), the actual output matrix Y can be expressed as

$$Y = \frac{1}{2}(Y_l + Y_r) = \frac{1}{2}\left(Q_l^T W_l + Q_r^T W_r\right). \tag{40}$$

The total squared error (16) can be expressed as

$$J = \frac{1}{2P\cdot Z}\sum_{p=1}^{P}\sum_{z=1}^{Z}\left(y_z^p - d_z^p\right)^2. \tag{41}$$

By using matrix notation to re-organize $\boldsymbol{J}$, first we define error function $\boldsymbol{E}$ as

$$\begin{aligned} E = Y - D &= \frac{1}{2}(Y_l + Y_r) - D \\ &= \frac{1}{2}\left(Q_l^T W_l - D + Q_r^T W_r - D\right) \end{aligned} \tag{42}$$

then we have

$$J = \frac{1}{2PZ} Tr(EE^T). \tag{43}$$

To tune weighting factors using chain rule, we have

$$W_{l,t+1} = W_{l,t} - \beta_{l,t}\left.\frac{\partial J}{\partial W_l}\right|_t = W_{l,t} - \beta_{l,t}\frac{1}{2PZ}Q_l E_t \tag{44}$$

$$W_{r,t+1} = W_{r,t} - \beta_{r,t}\left.\frac{\partial J}{\partial W_r}\right|_t = W_{r,t} - \beta_{r,t}\frac{1}{2PZ}Q_r E_t. \tag{45}$$

After training, assuming zero error, we should have $D = Y = (1/2)(Q_l^T W_l + Q_r^T W_r)$. The learning rate for each iteration during the back propagation process is different, i.e., the learning rates are not fixed [3]. To find such the optimal learning rate for β_l and β_r, we have the following theorem.

Theorem 1: The optimal learning rates β_l and β_r defined in (44) and (45) can be found from the minimum of a quadratic polynomial $A\beta_l^2 + B\beta_r^2 + C\beta_l\beta_r + F\beta_l + G\beta_r < 0$, where $A > 0, B > 0$ and $F < 0, G < 0$ can be obtained from the left firing strength Q_l and right firing strength Q_r, desired output D and the weighting factors W_l and W_r.

Proof: First, we must find the stable range for β_l and β_r. To do so, we define the Lyapunov function as

$$V = J^2 \tag{46}$$

where J is defined in (41). The change of the Lyapunov function is $\Delta V = J_{t+1}^2 - J_t^2$. It is well known that if $\Delta V < 0$, the response of the system is guaranteed to be stable. For $\Delta V < 0$, we have

$$J_{t+1} - J_t < 0. \tag{47}$$

Consider all the P firing strengths as $Q_l = [\vec{q}_l^{\,1}\ \vec{q}_l^{\,2}\ \cdots\ \vec{q}_l^{\,P}] \in \Re^{M\times P}$ and $Q_r = [\vec{q}_r^{\,1}\ \vec{q}_r^{\,2}\ \cdots\ \vec{q}_r^{\,P}] \in \Re^{M\times P}$, the firing strengths remain the same but their order may change according to the order of weighting factors during the training process. Then, we have J_{t+1} from (43) as

$$\begin{aligned}
J_{t+1} &= (2PZ)^{-1} Tr\left(E_{t+1}E_{t+1}^T\right)\\
&= (2PZ)^{-1} Tr\left[\left(\frac{1}{2}\left(Q_l^T W_{l,t+1} + Q_r^T W_{r,t+1}\right) - D\right)\right.\\
&\qquad \times \left.\left(\frac{1}{2}\left(Q_l^T W_{l,t+1} + Q_r^T W_{r,t+1}\right) - D\right)^T\right]\\
&= (2PZ)^{-1} Tr\left\{\left[\frac{1}{2}\left(Q_l^T\left(W_{l,t} - \frac{1}{2}\beta_{l,t}\right.\right.\right.\right.\\
&\qquad \times \left.(P\cdot Z)^{-1}Q_l E_t\right)\\
&\qquad + \left.\left.Q_r^T\left(W_{r,t} - \frac{1}{2}\beta_{r,t}(P\cdot Z)^{-1}Q_r E_t\right)\right) - D\right]\\
&\qquad \times \left[\frac{1}{2}\left(Q_l^T\left(W_{l,t} - \frac{1}{2}\beta_{l,t}(P\cdot Z)^{-1}Q_l E_t\right)\right.\right.\\
&\qquad + \left.\left.\left.Q_r^T\left(W_{r,t} - \frac{1}{2}\beta_{r,t}(P\cdot Z)^{-1}Q_r E_t\right)\right) - D\right]^T\right\}\\
&= (2PZ)^{-1} Tr\left\{\left[E_t - \frac{1}{4}\left(\beta_{l,t}(P\cdot Z)^{-1}Q_l^T Q_l E_t\right.\right.\right.\\
&\qquad \left.\left. + \beta_{r,t}(P\cdot Z)^{-1}Q_r^T Q_r E_t\right)\right]\\
&\qquad \times \left[E_t^T - \frac{1}{4}\left(\beta_{l,t}(P\cdot Z)^{-1}E_t^T Q_l^T Q_l\right)\right.\\
&\qquad \left.\left. + \beta_{r,t}(P\cdot Z)^{-1}\ E_t^T Q_r^T Q_r\right)\right]\right\}\\
&= J_t - \frac{1}{4}\beta_{l,t}(PZ)^{-2} Tr\left[Q_l^T Q_l E_t E_t^T\right]\\
&\qquad - \frac{1}{4}\beta_{r,t}(PZ)^{-2} Tr\left[Q_r^T Q_r E_t E_t^T\right]\\
&\qquad + \frac{1}{32}(PZ)^{-3} Tr\left\{\left[\beta_{l,t}^2 Q_l^T Q_l E_t E_t^T Q_l^T Q_l\right.\right.\\
&\qquad \left. + \beta_{r,t}^2 Q_r^T Q_r E_t E_t^T Q_r^T Q_r\right]\\
&\qquad + \left[\beta_{l,t}\beta_{r,t} Q_l^T Q_l E_t E_t^T Q_r^T Q_r\right.\\
&\qquad \left.\left. + \beta_{l,t}\beta_{r,t} Q_r^T Q_r E_t E_t^T Q_l^T Q_l\right]\right\}.
\end{aligned}$$

Hence

$$J_{t+1} - J_t = A\beta_l^2 + B\beta_r^2 + C\beta_l\beta_r + F\beta_l + G\beta_r \tag{48}$$

where

$$A = \frac{1}{32}(PZ)^{-3} Tr\left[Q_l^T Q_l E_t E_t^T Q_l^T Q_l\right] \tag{49}$$

$$B = \frac{1}{32}(PZ)^{-3} Tr\left[Q_r^T Q_r E_t E_t^T Q_r^T Q_r\right] \tag{50}$$

$$C = \frac{1}{16}(PZ)^{-3} Tr\left[Q_l^T Q_l E_t E_t^T Q_r^T Q_r\right] \tag{51}$$

$$F = -\frac{1}{4}(PZ)^{-2} Tr\left[E_t^T Q_l^T Q_l E_t\right] \tag{52}$$

$$G = -\frac{1}{4}(PZ)^{-2} Tr\left[E_t^T Q_r^T Q_r E_t\right]. \tag{53}$$

It is obvious that A and B in (49) and (50) and F and G in (52) and (53) contain quadratic matrices. Therefore the A and B should be positive; F and G should be negative. However, C in (51) can be either positive or negative.

Therefore we have

$$J_{t+1} - J_t = A\beta_l^2 + B\beta_r^2 + C\beta_l\beta_r + F\beta_l + G\beta_r < 0.$$

In type-1 FNN [3], authors similarly defined $J_{t+1} - J_t < 0$ to guarantee the NN to be stable and the one-variable quadratic polynomial $J_{t+1} - J_t = A_1\beta^2 + F_1\beta < 0$ was derived for $A_1 > 0$ and $F_1 < 0$. Therefore, the optimal learning rate $\beta_{\text{opt}} = -F_1/2A_1$ can be derived to make $A_1\beta_{\text{opt}}^2 + F_1\beta_{\text{opt}}$ at its minimum in type-1 FNN. Similarly the determination of the minimum values of two-variable quadratic function $H = J_{t+1} - J_t$ can be found as follows:

Let $H = J_{t+1} - J_t < 0$, we have the first order partial derivatives of H as

$$\frac{\partial H}{\partial \beta_l} = 2A\beta_l + C\beta_r + F = 0 \tag{54}$$

$$\frac{\partial H}{\partial \beta_r} = 2B\beta_r + C\beta_l + G = 0. \tag{55}$$

The second partial derivatives of H are

$$\frac{\partial^2 H}{\partial\beta_l\partial\beta_l} = 2A, \quad \frac{\partial^2 H}{\partial\beta_l\partial\beta_r} = C \quad \text{and}$$
$$\frac{\partial^2 H}{\partial\beta_r\partial\beta_r} = 2B. \tag{56}$$

From partial derivatives theorem [22] and (56), if $C^2 < 4AB$ and $A > 0, B > 0$ then there forms a quadratic parabolic function and has a local minimum value at a critical point $(\beta_l,\ \beta_r)$. Therefore, by solving (54) and (55), we can find the left-end and right-end optimal learning rate as

$$\beta_{l,\text{opt}} = (CG - 2BF)/(4AB - C^2) \tag{57}$$

and

$$\beta_{r,\text{opt}} = (CF - 2AG)/(4AB - C^2). \tag{58}$$

To prove $C^2 < 4AB$, we first let $Q_l^T Q_l E_t = [L_{ij}], Q_r^T Q_r E_t = [R_{ij}]$. From (49)–(51), we have

$$A = \frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}^2 \tag{59}$$

$$B = \frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} R_{ij}^2 \tag{60}$$

$$C = \frac{1}{16}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}R_{ij}. \tag{61}$$

According to Cauchy inequality [22] we have

$$\left(\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}R_{ij}\right)^2 < \left(\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}^2\right)\left(\sum_{i=1}^{P}\sum_{j=1}^{Z} R_{ij}^2\right). \tag{62}$$

Therefore, by using (59)–(62), we can obtain

$$\begin{aligned} C^2 &= \left(\frac{1}{16}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}R_{ij}\right)^2 \\ &< 4\left(\frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}^2\right) \\ &\quad \times \left(\frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} R_{ij}^2\right) = 4AB. \end{aligned} \tag{63}$$

Q.E.D.

By inspecting (42) and (48)–(53), it is obvious that the stable range of (β_l, β_r) is a function of Q_l, Q_r, D, W_l and W_r. In comparison with the positive optimal learning rate in type-1 FNN, the following Theorem 2 shows that the stable optimal learning rates in interval T2FNN can be positive or negative, but cannot be negative simultaneously.

Theorem 2: For the two-layer NN of consequent part, both stable optimal learning rate $\beta_{l,\mathrm{opt}}$ and $\beta_{r,\mathrm{opt}}$ cannot be negative simultaneously.

Proof: Suppose both learning rates in (57)–(58) are both negative, i.e.,

$$\beta_{l,\mathrm{opt}} = (CG - 2BF)/(4AB - C^2) < 0$$

and

$$\beta_{r,\mathrm{opt}} = (CF - 2AG)/(4AB - C^2) < 0.$$

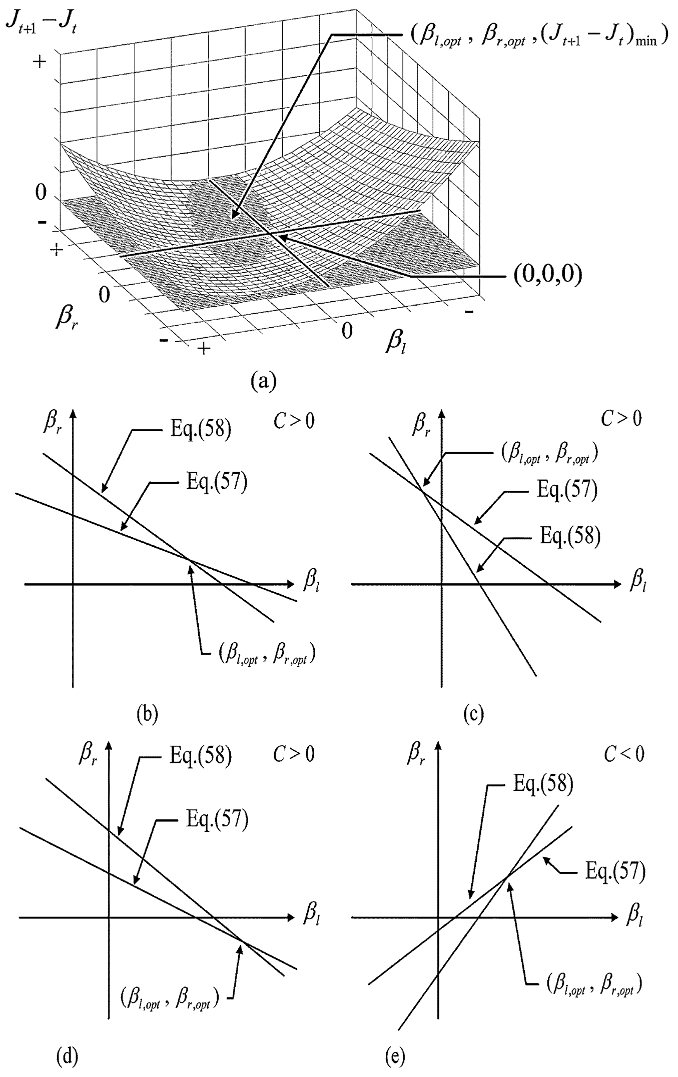
Since $(4AB - C^2) > 0$, therefore $CG - 2BF < 0$ and $CF - 2AG < 0$. Therefore we have

$$CG < 2BF \tag{64}$$

and

$$CF < 2AG. \tag{65}$$

In Theorem 1, we know that $A > 0, B > 0, F < 0$ and $G < 0$, but C can be positive or negative. If C is negative then CG in (64) will be positive. So (64) cannot be true due to $2BF < 0$. Similarly, (65) cannot be true if C is negative. As a result, both learning rates are all positive if $C < 0$.



$J_{t+1} - J_t$
$(\beta_{l,opt}, \beta_{r,opt}, (J_{t+1} - J_t)_{\min})$
(0,0,0)
β_r
β_l
(a)
Eq.(58)
Eq.(57)
$C>0$
$(\beta_{l,opt}, \beta_{r,opt})$
(b)
(c)
$C>0$
(d)
$C<0$
(e)

Fig. 7. Quadratic parabolic trajectories of $J_{t+1} - J_t$ versus β_l and β_r in (a). Three cases of $\beta_{l,\mathrm{opt}}$ and $\beta_{r,\mathrm{opt}}$ are shown on (b), (c), and (d) while $C > 0$. (b) Shows both $\beta_{l,\mathrm{opt}}$ and $\beta_{r,\mathrm{opt}}$ are >0. (c) Shows $\beta_{l,\mathrm{opt}} < 0$ but $\beta_{r,\mathrm{opt}} > 0$. (d) Shows the case when $\beta_{l,\mathrm{opt}} > 0$ and $\beta_{r,\mathrm{opt}} < 0$. (e) Shows both $\beta_{l,\mathrm{opt}}$ and $\beta_{r,\mathrm{opt}}$ are >0 when $C < 0$.

If C is positive, multiply (64) by (F/G), we have

$$CF < 2BF^2/G < 0 \tag{66}$$

and

$$CF < 2AG < 0. \tag{67}$$

Combining (66) and (67), we have

$$C^2F^2 > 4ABF^2. \tag{68}$$

Deleting F^2 from (68), we can yield $C^2 > 4AB$. This violates (63), i.e., $C^2 < 4AB$. Therefore we know that both optimal learning rates should not be negative simultaneously. Q.E.D.

According to the Theorem 2, it is obvious that the two lines (on the plane of [β_l versus β_r]) defined in (54) and (55) may have an intersecting point located on one of the first, second and fourth quadrant respectively. Fig. 7(a) shows a 3-D view of the intersections. Fig. 7(b) and (e) show the cases when both optimal learning rates $\beta_{l,\mathrm{opt}} > 0$ and $\beta_{r,\mathrm{opt}} > 0$. Fig. 7(b) shows the case for $C > 0$ and Fig. 7(e) is for $C < 0$. Fig. 7(c) shows the case when $\beta_{l,\mathrm{opt}} < 0$ and $\beta_{r,\mathrm{opt}} > 0$. Fig. 7(d) is for $\beta_{l,\mathrm{opt}} > 0$ and $\beta_{r,\mathrm{opt}} < 0$.

Consequently, the algorithm of tuning process in this two-layer interval NN is stated as the following Algorithm 2.

Algorithm 2. Dynamic Optimal Learning Rates for Consequent Part of T2FNN:

[Step 1]: Given the initial weighting matrices W_{l0} and W_{r0}, firing matrices Q_l and Q_r, and desired output D, find the initial actual output Y_0 (40) and optimal learning rate $\beta_{l0,\mathrm{opt}}$ and $\beta_{r0,\mathrm{opt}}$. Then, set initial iteration $t = 0$ and start the back propagation training process.

[Step 2]: Check if the desired output D and actual output Y_t are close enough (i.e., threshold limit) or if the maximize number of iteration is achieved? If Yes, Go to Step 6.

[Step 3]: Update the weighting matrices to obtain $[W_{l,t+1} W_{r,t+1}]$ (44), (45).

[Step 4]: Find the optimal learning rate $\beta_{l,\mathrm{opt},t+1}$ and $\beta_{r,\mathrm{opt},t+1}$ for the next iteration.

[Step]5: Set $t = t + 1$. Go to step 2.

[Step 6]: End.

Given the same case of Example 1, the following example illustrates the major concept in this section.

Example 2: The weighting factors of the consequent part in Example 1 will be tuned by using dynamic optimal learning algorithm as stated in Algorithm 2. We also allow 15 iterations for this dynamical optimal tuning so that we can compare the tuning results with those in Example 1. The firing strength matrices F from this MIMO T2FNN (4 inputs and two outputs $(P = 4, Z = 2, M = 3)$) can be found from Example 1 as

$$F = \left[\begin{bmatrix} 0.0060, & 0.6695 & 0.0325 \\ 0.1344, & 1.0000 & 0.3866 \end{bmatrix}_{p=1} \begin{bmatrix} 0.2884, & 0.0050 & 0.3533 \\ 0.9692, & 0.1763 & 0.9560 \end{bmatrix}_{p=2} \begin{bmatrix} 0.0143, & 0.0105 & 0.0000 \\ 0.2865, & 0.2724 & 0.0014 \end{bmatrix}_{p=3} \begin{bmatrix} 0.0091, & 0.0130 & 0.2420 \\ 0.2306, & 0.2078 & 0.8825 \end{bmatrix}_{p=4} \right].$$

During the dynamical optimal training process, the weighting intervals will be tuned to reduce maximum error after each iteration. The initial J in (41) can be obtained as 0.3919. The optimal learning rates are used to update the weighting matrix W. The trajectory of total squared error J is listed in Table I and shown in Fig. 8(a). Fig. 8(a) also shows the trajectory J for fixed learning rate $\alpha = 0.2$ as shown in Fig. 5. It is obvious that the total squared error J is decreased as expected in decent direction and converged faster than that in Fig. 5. Fig. 8(b) shows that both $\beta_{l,\mathrm{opt}}$ and $\beta_{r,\mathrm{opt}}$ cannot be negative simultaneously, which is in accordance with Theorem 2. Note that the tunings in Example 1 include the mean, standard deviation and weighting factors, all

TABLE I
OPTIMAL LEARNING RATES AND TOTAL SQUARED ERROR

t	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	4.2601	16.6882	0.083449
2	16.4054	22.7456	0.023489
3	26.5350	-2.7640	0.007165
⋮	⋮	⋮	⋮
13	-26.5641	39.4360	5.0038e-4
14	20.8115	38.6545	4.7584e-4
15	-25.2773	38.2233	4.5096e-4

using fixed learning rate $\alpha = 0.2$. After 15 iterations, we have the final weighting matrices W_l, W_r and output as

$$W_l = \begin{bmatrix} w_{l1}^1, & w_{l2}^1 \\ w_{l1}^2, & w_{l2}^2 \\ w_{l1}^3, & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.3994, & 5.8865 \\ 1.0985, & -0.5992 \\ 3.7988, & 2.6076 \end{bmatrix}$$
$$W_r = \begin{bmatrix} w_{r1}^1, & w_{r2}^1 \\ w_{r1}^2, & w_{r2}^2 \\ w_{r1}^3, & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.8049, & 11.1911 \\ 5.3116, & 8.0365 \\ 5.0397, & 2.6247 \end{bmatrix}$$
$$y(\vec{x}) = \begin{bmatrix} 3.5199, & 3.9927 \\ 5.4328, & 6.1792 \\ 4.9471, & 5.7872 \\ 4.7015, & 4.3360 \end{bmatrix}.$$

IV. TUNNING INTERVAL T2FNN USING A GENETIC ALGORITHM

Authors in [20], [23] used a reasonable spread rate by standard deviation ratios, $-\lambda\sigma$ and $\lambda\sigma$, to set the uncertain means as

$$\left[m_{k1}^i, m_{k2}^i\right] = \left[m_k^i - \lambda\sigma_k^i, m_k^i + \lambda\sigma_k^i\right] \tag{69}$$

where both the mean m_k^i and the standard deviation σ_k^i are parameters of the kth primary MF in the ith fuzzy rules in (5); and λ is a spread rate. By doing so, all interval type-2 fuzzy sets can be constructed and yield better performance than that in type-1 FLS. However the optimal selection of spread rate λ for type-2 FLS can be done via GA-based approach in [13]. For the overall tunings of T2FNN, we need to find the optimal learning rate α_{opt} in (17) and (18) (for means and standard deviations), the optimal spread rate λ_{opt}, and the optimal weighting matrices for consequent part. However, due to the dynamical optimal training algorithm derived in previous section, it is obvious that we do not have to rely on the genetic search algorithm to find the optimal weighting matrices. We only have to design a GA-based algorithm to search the optimal spread rate λ_{opt} and optimal learning rate α_{opt} for means and standard deviations. The fitness function $\varphi(J)$ (J is the total squared error) in the GA-based search algorithm can be defined as [24]

$$\varphi(J) = \varphi(\psi 10^\gamma) = \begin{cases} -\gamma + 1 - \dfrac{\psi}{10}, & \text{if } \gamma < 0 \\ 10^{-(\gamma+1)} + \dfrac{1 - \psi/10}{10^{(\gamma+1)}}, & \text{if } \gamma \geq 0 \end{cases} \tag{70}$$

where $J = \psi 10^\gamma, 1 < \psi < 10$. The above (70) finds a larger fitness value for smaller J. We randomly encode the parameters λ

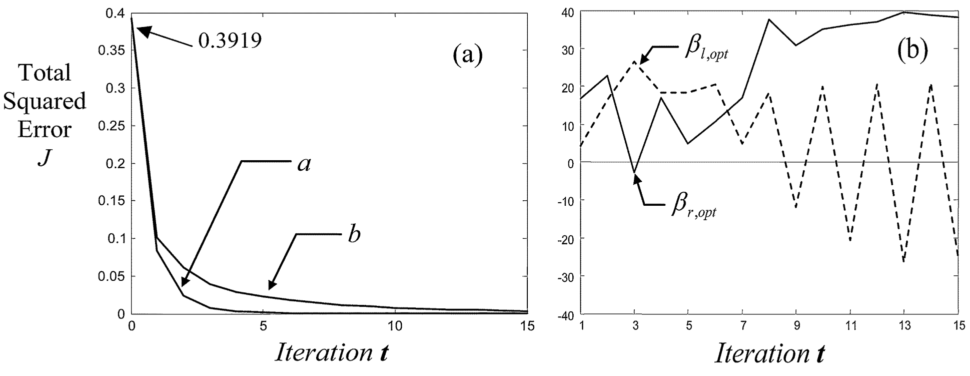


Fig. 8. Case a: Optimal learning for consequent part only. Case b: *Example 1*, $\alpha = 0.2$. (b) $\beta_{l,\mathrm{opt}}$ and $\beta_{r,\mathrm{opt}}$ cannot be negative simultaneously.

TABLE II
PARAMETERS FOR GA AND THE RESULTS FOR λ_{opt} AND α_{opt}

Pop_size	Max_gen	Chromosome size (bits)	$[\lambda_l,\lambda_u]$	$[\alpha_l,\alpha_u]$	Threshold	λ_{opt}	α_{opt}
14	16	32	[0.01 1]	[0.01 1]	0.46	0.7258	0.2966

and α to form a chromosome in a population for each iteration. The chromosome with larger fitness value has a larger probability of selection. Then, a new population is formed by selecting the better-fit chromosomes. Some members of the new population undergo transformation by means of genetic operators to form new solutions. The crossover operation combines the features of two parent chromosomes to form two similar children by swapping corresponding segments of their parents. The parameters defining the crossover operation are the probability of crossover P_c and the crossover position. Mutation is the process of occasional alternation of some gene values in a chromosome by a random change with a probability less than the mutation rate P_m.

Therefore, based on this GA design, we can combine the dynamic optimal learning algorithm (Algorithm 2) to perform some desired iterations under the back propagation training process. Because every new population consists of better-fit chromosomes, the search then can be continued to obtain the optimal spread rate λ_{opt} and the optimal learning rate α_{opt} such that the total squared error J is a minimum. The overall search algorithm, which summarizes the whole concept, is listed as follows.

Algorithm 3. Tuning FNN Via Genetic Algorithm With Optimal Learning: Given P input-output training sample pairs $(\vec{x}^p : d^p), p = 1, \ldots, P$, we wish to tune the all parameters in antecedent and consequent part so that (46) can be minimized for T iterations.

```
Step 1: Initialize weighting matrices $W_{l0}$ and $W_{r0}$ randomly. Define range intervals for
        spread rate $[\lambda_l, \lambda_u]$ and learning rate $[\alpha_l, \alpha_u]$. Set $Pop\_size, Max\_gen, Iteration$ and
        $Threshold$.
Step 2: Initialize population Pop $= \{\lambda_j, \alpha_j\}, \lambda_j \in (\lambda_l, \lambda_u), \alpha_j \in (\alpha_l, \alpha_u), j = 1, \ldots, Pop\_size$.
   For $generation = 1 : Max\_gen$
     For $j = 1 : Pop\_size$
       Get $i$th $\lambda$ and $\alpha$. Use $\lambda$ to initialize uncertain means
       For $t = 1 : Iteration$
         For $p = 1 : P$
           Apply training sample $x^p$, and compute the total firing strength for each
             rule in (8).
           Compute $y_l, y_r$ and its defuzzified output $(y_l + y_r)/2$ in (15).
           Use back propagation to tune the parameters of active branches.
         End
         Compute new firing strength matrices $Q_l$ and $Q_r$.
         While $|J_{t,\min} - J_{t-2,\min}|/J_{t-2,\min} > Threshold$
           Compute matrix $E$ in (42), and find $\beta_{l,\mathrm{opt},t}$ and $\beta_{r,\mathrm{opt},t}$ (Theorem 1).
           Compute matrices $W_{l,t+1}$ and $W_{r,t+1}$ in (44)–(45), and $J_{t+1,\min}$ in (43).
         End
       End
       Put $j$th $J_{t+1,\min}$ (and/or any other CPI factors) into fitness vector.
     End
     Perform selection, crossover, and mutation for next generation.
   End
   Optimal spread rate $\lambda_{\mathrm{opt}}$ and optimal learning rate $\alpha_{\mathrm{opt}}$ are found.
   For antecedent part: uncertain means $[m_{k1}^i, m_{k2}^i]$ and deviation $\delta_k^i$ are found
   For consequent part: $W_{l,t+1}, W_{r,t+1}, \beta_{l,\mathrm{opt}}, \beta_{r,\mathrm{opt}}$ and $J_{t+1,\min}$ are found.
```

Example 3: Given the same Example 1, we extend the same primary type-1 Gaussian MFs by using the spread rate λ in (69). Then we use the Algorithm 3 to tune this MIMO interval T2FNN. Table II shows all parameters in the GAs process and the final results for optimal spread rate λ_{opt} and optimal learning rate α_{opt}. To increase the efficiency, we define mutation rate P_m and crossover rate P_c [3] as

$$P_m = exp^{(0.05k/\mathrm{Max\_gen})} - 1 \tag{71}$$

TABLE III
OPTIMAL LEARNING RATE & TOTAL SQUARED ERROR

t	Optimal rate λ_{opt} and α_{opt}			Fixed rate (λ=0.5, α=0.2)		
	$\beta_{l,opt}$	$\beta_{r,opt}$	J	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	7.8055	13.7388	0.0039257	27.1541	-6.4351	0.0176966
2	11.4009	3.8154	1.7371e-8	-1.1137	24.9637	4.9332e-3
3	16.6235	39.3156	1.5162e-8	9.4266	112.2790	6.9830e-4
4	4.4307	18.0355	4.7117e-9	-13.0602	26.2662	5.7282e-5
5	11.6018	40.9221	3.0365e-9	6.5622	90.5961	1.9310e-6

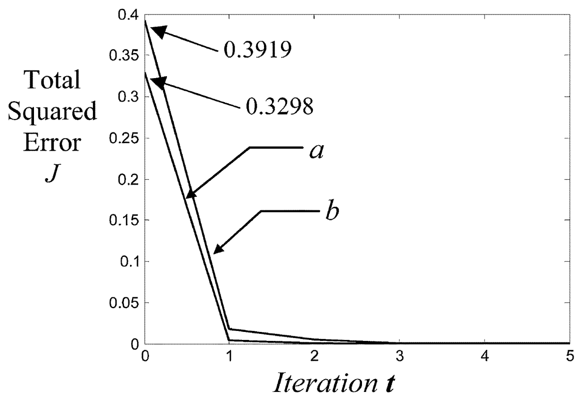


Fig. 9. Performance comparisons. Case a: Optimal spread rate λ_{opt} and learning rate α_{opt} with dynamical optimal learning rates $\beta_{l,\text{opt}}$ and $\beta_{r,\text{opt}}$. Case b: $\lambda = 0.5, \alpha = 0.2$ with dynamical optimal learning rates $\beta_{l,\text{opt}}$ and $\beta_{r,\text{opt}}$.

$$P_c = exp^{(-k/\text{Max\_gen})} \tag{72}$$

where k denotes the kth generation. After 5 iterations, we have the final tuned weighting factors

$$W_l = \begin{bmatrix} w_{l1}^1, & w_{l2}^1 \\ w_{l1}^2, & w_{l2}^2 \\ w_{l1}^3, & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.2758, & 5.0332 \\ 1.2275, & -0.5806 \\ 3.9290, & 0.4600 \end{bmatrix}$$
$$W_r = \begin{bmatrix} w_{r1}^1, & w_{r2}^1 \\ w_{r1}^2, & w_{r2}^2 \\ w_{r1}^3, & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.5219, & 11.649 \\ 4.6719, & 7.4445 \\ 5.0558, & 3.7278 \end{bmatrix}.$$

In comparison with fixed spread rate and optimal learning rate, we also combine Example 1 with Algorithm 2 to tune the overall T2FNN. Therefore we have the output results as

$$y(\vec{x})|_{\lambda_{\text{opt}},\alpha_{\text{opt}}} = \begin{bmatrix} 3.5200, & 4.0200 \\ 5.4301, & 6.2300 \\ 4.9499, & 5.7599 \\ 4.7001, & 4.2801 \end{bmatrix} \quad \text{and}$$
$$y(\vec{x})|_{\lambda=0.5,\alpha=0.2} = \begin{bmatrix} 3.5206, & 4.0215 \\ 5.4324, & 6.2328 \\ 4.9496, & 5.7591 \\ 4.7002, & 4.2764 \end{bmatrix}.$$

The total squared errors vs. iterations are listed in Table III, and the results are plotted in Fig. 9. Fig. 9 also shows the trajectory J for the tuning results of using fixed rates in antecedent part (i.e., $\lambda = 0.5$ and $\alpha = 0.2$), and optimal learning rates $\beta_{l,\text{opt}}$ and $\beta_{r,\text{opt}}$ in consequent part. By inspecting the results in this example, it is obvious that the actual output of GA-based approach is closer to desired output and convergence is much faster than those results from fixed rate, or even Examples 1 and 2.

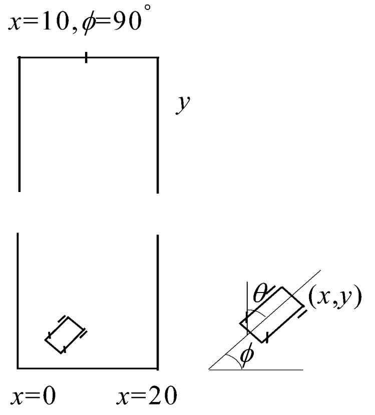


Fig. 10. Diagram of simulated truck and loading zone.

V. EXAMPLES

Based on the above GAs design for the interval T2FNN, two popular examples will be fully illustrated in this section. Example 4 is the truck back up control problem. Example 5 is nonlinear system identification.

Example 4: Truck Back Up Control Problem: The well-known nonlinear problem of backing up a truck into a loading dock via the FNN controller [25], [26], will be controlled by using interval T2FNN. Fig. 10 shows the truck and loading zone. The truck position is located by three state variables x, y, and ϕ, where ϕ is the angle of the truck with the horizontal axis x. Steering angle θ is used to control the truck. The truck moves backward by a fixed unit distance every step. We assume enough clearance between the truck and the loading zone, therefore y does not have to be considered. Hence the system has two inputs, i.e., $-115^\circ \le \phi \le 295^\circ$ & $0 \le x \le 20$, and one output θ within $[-40^\circ, 40^\circ]$. The final states (x_f, ϕ_f) will be equal or close to $(10, 90^\circ)$. In this simulation, steering angle θ will be normalized into [0, 1].

A reasonable numbers of training data pairs must be first generated as desired input-output pairs so that it can cover whole situation. The following 14 initial states are used to generate such pairs: $(x_0, \phi_0) = (1, 0^\circ), (1, 90^\circ), (1, -90^\circ), (7, 0^\circ), (7, 90^\circ), (7, 180^\circ), (7, -90^\circ), (13, 0^\circ), (13, 90^\circ), (13, 180^\circ), (13, 270^\circ), (19, 90^\circ), (19, 180^\circ)$, and $(19, 270^\circ)$. The following approximate kinematics is used to simulate the truck path as

$$x_{t+1} = x_t + \cos(\phi_t + \theta_t) + \sin(\phi_t)\sin(\theta_t) \tag{73}$$

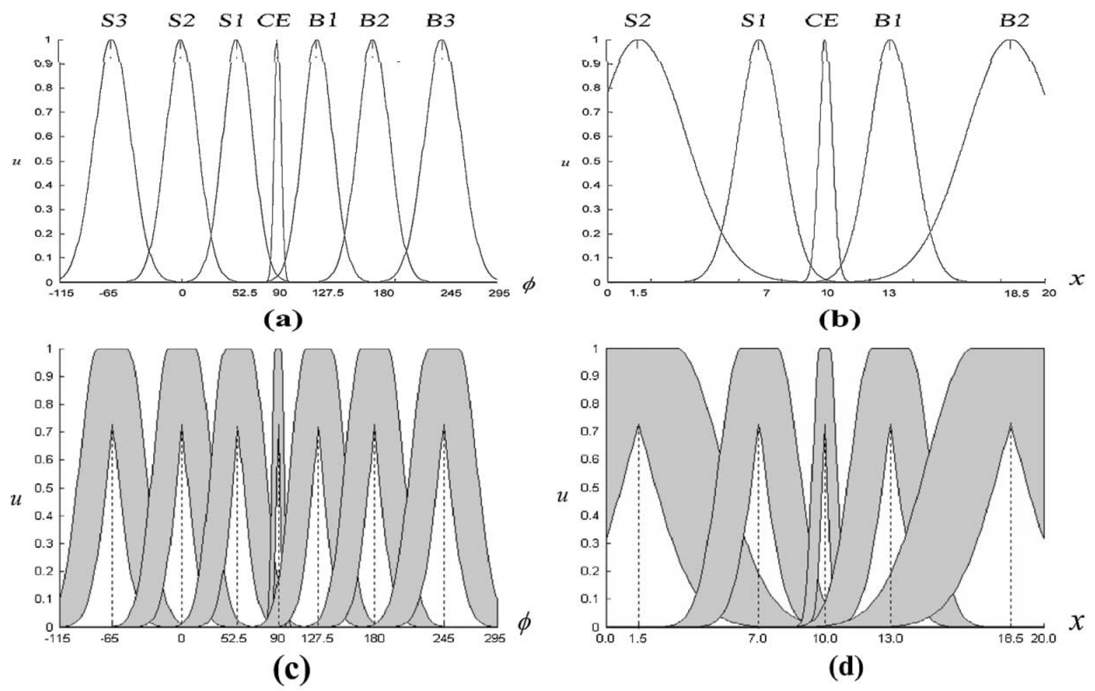


Fig. 11. Two antecedents initial MFs of T1FNN in (a) and (b), and the two corresponding antecedents initial MFs of interval T2FNN in (c) and (d).

$$y_{t+1} = y_t + \sin(\phi_t + \theta_t) - \sin(\phi_t)\sin(\theta_t) \quad (74)$$

$$\phi_{t+1} = \phi_t - \sin^{-1}\left(\frac{2\sin(\theta_t)}{l}\right) \quad (75)$$

where l is the length of the truck, we assume $l = 4$. Equations (73)–(75) will be used to derive the next state when present state and control are given. Since y is not considered, (74) will be discarded.

In this application, we compare the performances by using two different FNNs, i.e., singleton type-1 FNN (T1FNN) and interval T2FNN. We use a single rate to identify the spread of uncertain means of interval Gaussian MF by its deviation ratio in (69), i.e., $-\lambda\sigma$ and $+\lambda\sigma$; then all parameters of interval T2FNN can be obtained. The weighting factors w_l and w_r can also be set randomly in [0, 1].

Figs. 11(a) and (b) show the two antecedents initial type-1 MFs of T1FNN, where Sn means small level, CE means center and Bn means big level. By using spread rate to extend from type-1 MFs, we have two corresponding antecedents initial type-2 MFs of interval T2FNN shown in Figs. 11(c) and (d). For steering angle θ, we let the T's be the its fuzzy MF. The centers of T1, T2, T3, T4, T5, T6, and T7 are $-40°$, $-20°$, $-7°$, $0°$, $7°$, $20°$, and $40°$, respectively.

Each FNN system has 35 rules which come from 7 MFs in the first antecedent by five MFs in the second antecedent. We use 32 bits to form the chromosome, 16 bits for spread rate λ and 16 bits for optimal learning rate α. The chromosome will be mapped into real values in the ranges of $[\lambda_l, \lambda_u]$ and $[\alpha_i, \alpha_u]$, respectively. The mutation rate P_m and crossover rate P_c are defined as in Example 3. To guarantee the performance of control process, the term of settling time (or settling steps) is also taken as a control performance criteria (CPI) for deriving optimal spread and learning rate in GA searching process. The results of simulation show that the smaller settling time (or settling steps) and smaller squared error can lead better performance in this interval type-2 FNN case. The best settling steps from two initial states (0, 0°) and (20, 180°) are 19 and 28 steps, respectively.

Given the same training pairs, we also train the type-1 FNN with dynamic optimal learning for consequent part. For comparison purpose, the following TSK model in [27] and [28]

$$R^i : \text{IF } x_1 \text{ is } F_1^i, \text{and } x_2 \text{ is } F_2^i, \qquad \text{THEN } y_i = c_0^i + c_1^i x_1 + c_2^i x_2 \quad (76)$$

where $i = 1, \ldots, M(=35)$. With more trainable parameters in the consequent part will be tuned to compare their performance with T1FNN and T2FNN. The TSK model, in fact, can be treated as T1FNN (T1FNN-TSK) and trained with optimal learning algorithm [3].

Each case under this application is run for five iterations. Fig. 12(a) shows the performance comparsion with T1FNN, T1FNN-TSK and interval T2FNN. Fig. 12(b) shows the times for the truck to arrive target position in ten different initial conditions, and their first five trajectories are plotted in Fig. 13. Table IV shows the number of design parameters for these three models. The total squared errors of all cases are shown in Table V. From Figs. 12 and 13, it is obvious that the performances of interval T2FNN is better than those in T1FNN. It not only takes less steps to reach target position using interval T2FNN, but it also shows smoother trajectories. However, due to more parameters included, the performances of T1FNN-TSK is still better than T1FNN in Fig. 12.

The TSK model was also similarly defined as a *Type-3 fuzzy reasoning* in adaptive network-based inference system (ANFIS) in [29]. The T2FNN with dynamic optimal training and more trainable parameters can achieve desired performance in fewer iterations, whereas ANFIS with least square estimation needs

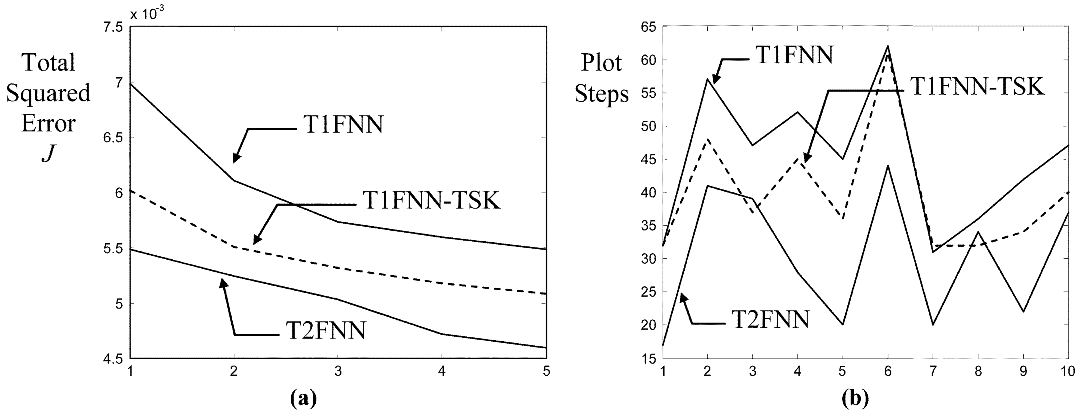


Fig. 12. (a) Shows the results of total squared errors in T1FNN and T2FNN. (b) Shows the total steps to arrive target position by ten different cases.

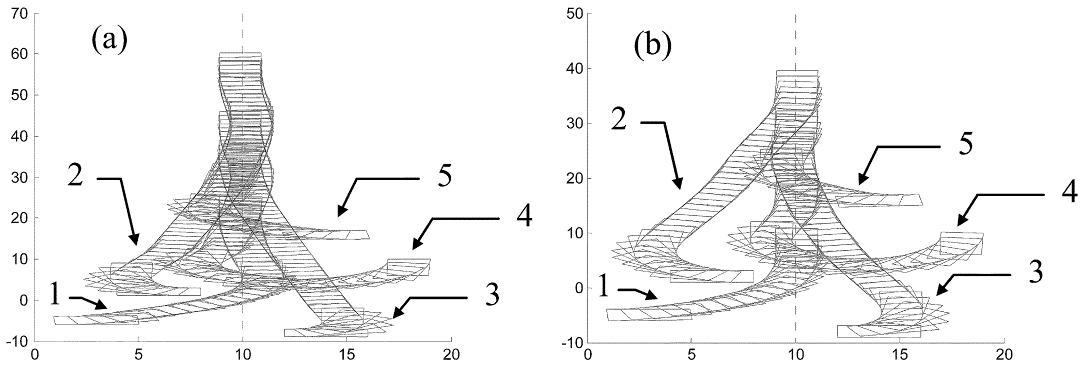


Fig. 13. (a) Shows truck trajectories via T1FNN. (b) Shows via interval T2FNN, all from different initial conditions 1–5.

TABLE IV
NUMBER OF DESIGN PARAMETERS FOR DIFFERENT MODEL, WHERE $n = 2, M = 35$

	T1FNN	T1FNN-TSK	T2FNN
Number of parameters	$(2n+1)M$=175	$(2n+3)M$=245	$(3n+2)M$=280

TABLE V
TOTAL SQUARED ERROR J FOR FIVE ITERATIONS

Iter.	T1FNN		T1FNN-TSK		T2FNN		
	β_{opt}	J	β_{opt}	J	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	40.9681	0.006984	53.6074	0.006023	179.8414	60.6627	0.005491
2	63.0143	0.006108	51.2676	0.005514	303.9327	73.1634	0.005250
3	32.0096	0.005738	40.3804	0.005318	197.6248	39.5050	0.005036
4	36.6070	0.005597	41.8190	0.005185	569.5179	161.1721	0.004722
5	37.6215	0.005489	42.9906	0.005090	316.1715	25.2515	0.004597

more iterations in its proposed model [29]. It is obvious that the improvement of dynamic optimal training in Theorem 1 of T2FNN and T1FNN [3] can yield faster convergence.

Example 5: Nonlinear System Identification Second Order System: The plant to be identified is described by the following second-order difference:

$$y(k+1) = g[y(k), y(k-1)] + u(k) \tag{77}$$

where

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

A series-parallel FNN identifier [25] described by the following equation

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1)] + u(k) \tag{78}$$

will be adopted, where $\hat{f}[y(k), y(k-1)]$ is the form of (10) with two fuzzy variables $y(k)$ and $y(k-1)$. Training data of 500 pairs are generated from plant model, assuming a random input signal $u(k)$ uniformly distributed in $[-2, 2]$. The data are used to build fuzzy model for $\hat{y}$. We follow (69) to allocate type-1 MFs for these two variakbes $y(k)$ and $y(k-1)$ as $m_{k1} = [-1\ 0\ 1.5\ 3], m_{k2} = [-1\ 0\ 1.5\ 3]$, and $\delta_{k1} = [0.5\ 0.3\ 0.4\ 0.6], \delta_{k2} = [0.5\ 0.3\ 0.4\ 0.6]$. Then we extend above MFs to type-2 interval MFs by using spread rate, where its optimal value will be found through Algorithm 3. The

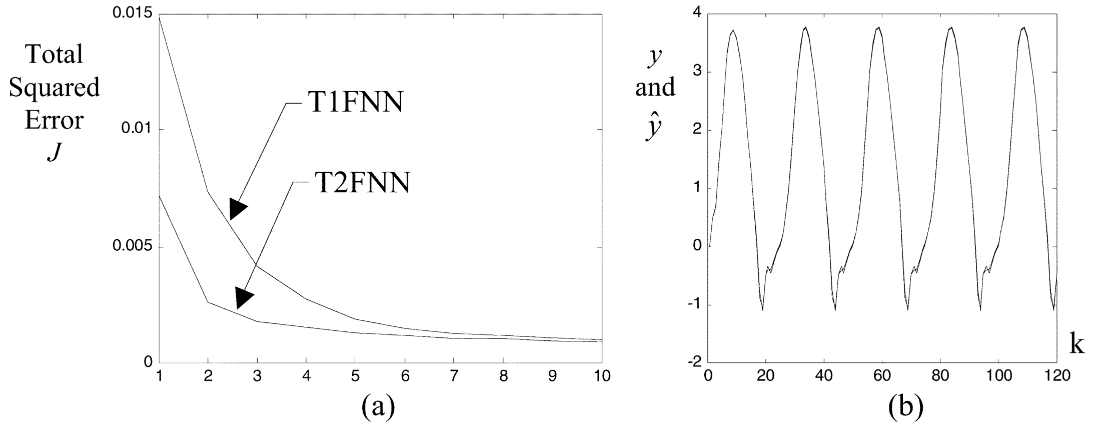


Fig. 14. (a) Shows the results of total squared error vs iterations in T1FNN (dashed line) and T2FNN (solid line). (b) Shows outputs of the plant y (solid line) and the identification model $\hat{y}$ (dashed line).

TABLE VI
TOTAL SQUARED ERROR J FOR TEN ITERATIONS

Iter.	T1FNN		T2FNN		
	β_{opt}	J	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	12.0116	0.014861	14.6116	8.3728	0.007153
2	6.8172	0.007348	83.2264	52.9716	0.002609
3	12.5323	0.004160	-29.5359	53.0753	0.001778
⋮	⋮	⋮	⋮	⋮	⋮
8	39.2183	1.1420e-3	36.5806	1.0250	1.0098e-3
9	33.8319	1.0595e-3	35.2905	-4.4031	9.5119e-4
10	20.8053	1.0063e-3	32.1227	-8.6364	8.9176e-4

mutation rate P_m and crossover rate P_c are defined the same as in Example 3. The initial value of J is 1.8803. Fig. 14(a) shows the performance comparsion with T1FNN and interval T2FNN. It is obvious that faster convergence is also obtained via interval T2FNN. After the training process is finished, the FNN model is tested by applying a sinusoidal input signal $u(k) = \sin(2\pi k/25)$. Fig. 14(b) shows the outputs of both FNN model and actual model. The total squared error J using 120 testing data items is 0.0020. This example shows excellent results are obtained via interval T2FNN. Table VI shows total squared error J for 10 iterations.

VI. CONCLUSION

The interval type-2 FLS with type reduction was extended with interval neural network to construct an interval T2FNN in this paper. The consequent part of this interval T2FNN is also an interval neural network. The dynamical optimal training for this interval neural network is also developed to guarantee maximum error reduction during the training process. A multi-input, multi-output FNN model is adopted to illustrate all the properties of this T2FNN with dynamical optimal training in its consequent part. This dynamical optimal training algorithm can be combined into a GA-based approach to find the optimal spread rate and learning rate for antecedent part. The optimal weighting factors in the consequent part of this T2FNN can be directly found from the dynamical optimal training algorithm with global searching. This interval T2FNN with dynamic optimal learning algorithm is applied to control the truck backing-up system and nonlinear system identification. All the simulation results by using the interval T2FNN show better performances than those using T1FNN.

REFERENCES

[1] C. H. Wang, W. Y. Wang, T. T. Lee, and P. S. Tseng, "Fuzzy B-spline membership function (BMF) and its applications in fuzzy-neural control," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 841–851, May 1995.
[2] X. H. Yu *et al.*, "Dynamic learning rate optimization of the back propagation algorithm," *IEEE Trans. Neural Networks*, vol. 6, pp. 669–677, May 1995.
[3] C. H. Wang, H. L. Liu, and C. T. Lin, "Dynamic optimal learning rate of a certain class of fuzzy neural networks and its applications with genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 467–475, June 2001.
[4] R. R. Yager, "Fuzzy subsets of type II in decisions," *J. Cybern.*, vol. 10, pp. 137–159, 1980.
[5] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems: handling the uncertainty associated with surveys," in *Proc. IEEE FUZZ Conf.*, Seoul, Korea, Aug. 1999, pp. 1546–1551.
[6] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[7] S. Auephanwiriyakul, A. Adrian, and J. M. Keller, "Type 2 fuzzy set analysis in management surveys," in *Proc. FUZZ-IEEE Conf.*, May 2002, pp. 1321–1325.
[8] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems to forecasting of time-series," *Inform. Sci.*, vol. 120, pp. 89–111, 1999.
[9] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 643–658, Dec. 1999.
[10] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzy adaptive filters," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 551–563, Oct. 2000.
[11] K. C. Wu, "Fuzzy interval control of mobile robots," *Comput. Elect. Eng.*, vol. 22, no. 3, pp. 211–229, 1996.
[12] R. I. John, P. R. Innocent, and M. R. Barnes, "Neuro-fuzzy clustering of radiographic tibia images using type 2 fuzzy sets," *Inform. Sci.*, vol. 125, pp. 65–82, 2000.

[13] S. Park and H. Lee-Kwang, "A designing method for type-2 fuzzy logic systems using genetic algorithms," in *Proc. Joint 9th IFSA World Congress 20th NAFIPS Int. Conf.*, Vancouver, BC, Canada, July 2001, pp. 2567–2572.
[14] C. T. Lin and C. S. G. Lee, *Neural Fuzzy System*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[15] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—I," *Inform. Sci.*, vol. 8, pp. 199–249, 1975.
[16] N. N. Karnik and J. M. Mendel, "Operations on type-2 fuzzy sets," *Fuzzy Sets Syst.*, vol. 122, pp. 327–348, 2001.
[17] ——, "Centroid of a type-2 fuzzy set," *Inform. Sci.*, vol. 132, pp. 195–220, 2001.
[18] ——, "Introduction to type-2 fuzzy logic system," in *Proc. IEEE FUZZ Conf.*, Anchorage, AK, May 1998, pp. 915–920.
[19] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 117–127, Apr. 2002.
[20] Q. Liang and J. M. Mendel, "Interval type-2 logic systems: Theory and design," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 535–550, Oct. 2000.
[21] N. N. Karnik and J. M. Mendel, "Type-2 fuzzy logic systems: Type-reduction," in *Proc. IEEE Syst., Man, Cybern. Conf.*, San Diego, CA, Oct. 1998, pp. 2046–2051.
[22] S. I. Grossman, *Multivariable Calculus, Linear Algebra, and Differential Equations*. Orlando, FL: Academic, 1986.
[23] J. M. Mendel, "Uncertainty, fuzzy logic, and signal processing," *Signal Process.*, vol. 80, pp. 913–933, 2000.
[24] C.-C. Hsu *et al.*, "Digital redesign of continuous systems which improved suitability using genetic algorithms," *Electron. Lett.*, vol. 33, no. 15, pp. 1345–1347, July 1997.
[25] L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[26] B. Kosko, *Neural Network and Fuzzy System*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[27] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, 1985.
[28] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, vol. 28, pp. 15–33, 1988.
[29] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–684, May/June 1993.

Chi-Hsu Wang (M'92–SM'93) was born in Tainan, Taiwan, R.O.C, in 1954. He received the B.S. degree in control engineering from National Chiao-Tung University (NCTU), Hsinchu, the M.S. degree in computer science from the National Tsing-Hua University, Hsinchu, and the Ph.D. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 1976, 1978, and 1986, respectively.

He was appointed Associate Professor in 1986, and Professor in 1990, in the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan. He is currently Professor in the Department of Electrical and Control Engineering, NCTU. His current research interests and publications are in the areas of digital control, fuzzy-neural-network, intelligent control, adaptive control, and robotics.

Dr. Wang is currently Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART–B and Webmaster for the IEEE Systems, Man, and Cybernetics Society.

Chun-Sheng Cheng was born in Tainan, Taiwan, R.O.C., in 1957. He received the B.S. degree in communication engineering from National Chiao-Tung University, Hsinchu, Taiwan, and the M.S. degree in microelectronic engineering, Griffith University, Brisbane, Australia, in 1980 and 2003, respectively.

He was a Process Control System Engineer at China Steel Corporation, Kaohsiung, Taiwan, from 1983 to 1992. He is currently a Developer for the e-commerce data base, point of sales system, and duty free system in Data Control Pty. Ltd., Brisbane. His current research interests and publications are in the areas of type-2 fuzzy-neural-network, and adaptive control.

Tsu-Tian Lee (M'87–SM'89–F'97) was born in Taipei, Taiwan, R.O.C., in 1949. He received the B.S. degree in control engineering from the National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1970, and the M.S., and Ph.D. degrees in electrical engineering from the University of Oklahoma, Norman, in 1972 and 1975, respectively.

In 1975, he was appointed Associate Professor and in 1978 Professor and Chairman of the Department of Control Engineering at NCTU. In 1981, he became Professor and Director of the Institute of Control Engineering, NCTU. In 1986, he was a Visiting Professor and in 1987, a Full Professor of electrical engineering at the University of Kentucky, Lexington. In 1990, he was a Professor and Chairman of the Department of Electrical Engineering, National Taiwan University of Science and Technology (NTUST). In 1998, he became the Professor and Dean of the Office of Research and Development, NTUST. Since 2000, he has been with the Department of Electrical and Control Engineering, NCTU, where he is now a Chair Professor. Since 2004, he has been with the National Taipei University of Technology (NTUT), where he is now President. He has published more than 200 refereed journal and conference papers in the areas of automatic control, robotics, fuzzy systems, and neural networks. His current research involves motion planning, fuzzy and neural control, optimal control theory and application, and walking machines.

Prof. Lee received the Distinguished Research Award from the National Science Council, R.O.C., in 1991–1998, and the Academic Achievement Award in Engineering and Applied Science from the Ministry of Education, R.O.C., in 1997, the National Endow Chair from the Ministry of Education, R.O.C., in 2003, and the TECO Science and Technology Award from TECO Technology Foundation in 2003. He was elected to the grade of IEE Fellow in 2000, respectively. He became a Fellow of New York Academy of Sciences (NYAS) in 2002. His professional activities include serving on the Advisory Board of Division of Engineering and Applied Science, National Science Council, serving as the Program Director, Automatic Control Research Program, National Science Council, and serving as an Advisor of Ministry of Education, Taiwan, and numerous consulting positions. He has been actively involved in many IEEE activities. He has served as Member of Technical Program Committee and Member of Advisory Committee for many IEEE sponsored international conferences. He is now the Vice President of Membership, a member of the Board of Governors, and the Newsletter Editor for the IEEE Systems, Man, and Cybernetics Society.