1. Introduction

In the past few decades, the fuzzy neural network (FNN) which provides the advantages of both neural network and fuzzy system is successfully applied to nonlinear system identification and control [1–4]. In the FNN, the symmetric and fixed membership functions (MFs) are commonly adopted to simplify the design procedure. However, a large number of fuzzy rules should be used to achieve the specified performance [5, 6]. Thus, an asymmetric fuzzy membership function (AFMF) has been proposed to solve this problem. The AFMF is discussed and analyzed that it can effectively improve the accuracy and reduce the fuzzy rules [7, 8].

Recently, the type-2 fuzzy sets (T2 FSs) are gaining more and more in popularity [9, 10]. The T2 FSs are described by MFs that are characterized by more parameters than the
type-1 fuzzy sets (T1 FSs). Hence, the T2 FSs provide us with more design degree of freedom. Nevertheless, due to the computational complexity of using the T2 FS, most people only the interval type-2 fuzzy sets (IT2 FSs) are proposed [10]. The computations associated with IT2 FSs are very manageable which makes it quite practical [11]. In our previous research [12–15], we proposed a type-2 fuzzy neural network with asymmetric membership function (T2FNN-A) which combines interval type-2 fuzzy logic system with neural network. Then, in order to improve the efficiency of the T2FNN-A, we proposed an interval type-2 fuzzy neural system with AFMFs (IT2FNS-A) which utilizes several enhanced techniques such as Takagi-Sugeno-Kang fuzzy logic system (TSK FLS) and embedded type-reduction network layer [16]. These modifications not only improve the approximation accuracy of the T2FNN-A, but also achieve the specific performance with fewer fuzzy rules. However, a major drawback of the IT2FNS-A is that its application domain is limited to static problem due to its feedforward network structure. Thus, using IT2FNS-A to process dynamic problems is inefficient. Many results have been shown that recurrent system can learn and memorize information to provide better performance [2, 17–20]. In this paper, we propose an interval type-2 recurrent fuzzy neural system with AFMFs (IT2RFNS-A) which provides the memory elements and extends the basic ability of the IT2FNS-A to include the dynamic problems. In addition, since the feedback layer captures the dynamic response of the system, the approximation accuracy of the network is improved.

In training neural networks, the back-propagation (BP) algorithm is widely used. However, the differential information of system is difficult to obtain due to the piece-wise continuous property of triangular AFMFs and since there are many adjustable parameters in the IT2RFNS-A. Herein, we adopt the simultaneous perturbation stochastic approximation (SPSA) algorithm to derive the update laws of the proposed IT2RFNS-A. The SPSA algorithm is based on the approximation to gradient information from the measurements of the objective function [21–23]. Hence, a great deal of computational effort is saved. However, due to the stochastic characteristic of SPSA algorithm, we cannot guarantee that every searching step length is appropriate which may lead to the invalid search. To overcome this situation, we employ the Lyapunov stability analysis to derive the optimal learning step length for guaranteeing the stability of the closed-loop system. In addition, the efficient training is also ensured.

The remainder of this paper is organized as follows. In Section 2, the construction of triangular AFMFs and the SPSA algorithm are introduced. Section 3 illustrates the proposed IT2RFNS-A and the stable SPSA learning algorithm. The simulation results on chaotic system identification and control of the Chua’s chaotic circuit are shown in Section 4. Finally, the conclusion is given.

2. Preliminaries

In this section, we briefly introduce some prerequisite material including the interval type-2 asymmetric fuzzy membership function (IT2 AFMF) and simultaneous perturbation stochastic approximation (SPSA) algorithm.

2.1. Interval Type-2 Asymmetric Fuzzy Membership Function (IT2 AFMF)

The interval type-2 membership function (IT2 MF) is considered to be a special case of a general type-2 membership function (T2 MF) which simplifies the computational effort
significantly [10, 11]. In general, the symmetric MF is used for simplification. However, the symmetric MF only holds either uncertain mean or uncertain variance. In addition, tuning the parameter of MFs symmetrically may result in low precision. The AFMF can treat these problems.

In this paper, the triangular fuzzy MFs are used to construct the interval type-2 asymmetric fuzzy membership functions (IT2 AFMFs) due to their lower computational effort. Each IT2 AFMF consists of upper and lower MFs, as shown in Figure 1. The upper MF is defined as

$$
\overline{\mu}_F(x) = \begin{cases} 
\frac{x - \overline{a}}{\overline{b} - \overline{a}}, & \overline{a} \leq x \leq \overline{b}, \\
\frac{x - \overline{c}}{\overline{c} - \overline{b}}, & \overline{b} \leq x \leq \overline{c}, \\
0, & \text{otherwise},
\end{cases}
$$

(2.1)

where $\overline{a}$, $\overline{b}$, and $\overline{c}$ denote the positions of three corners satisfying $\overline{a} \leq \overline{b} \leq \overline{c}$. Similarly, the lower MF is defined as

$$
\underline{\mu}_F(x) = \begin{cases} 
\lambda \cdot \frac{x - a}{b - a}, & a \leq x \leq b, \\
\lambda \cdot \frac{c - x}{c - b}, & b \leq x \leq c, \\
0, & \text{otherwise},
\end{cases}
$$

(2.2)
where \(a, b,\) and \(c\) denote the positions of three corners satisfying \(a \leq b \leq c\) and \(\lambda\) denotes the magnitude of lower MF which should be limited between 0.5 and 1 to avoid the invalid result (a small firing strength). As above description, the following restrictions should be constrained to avoid unreasonable IT2 AFMFs: \(\overline{a} \leq \overline{b} \leq \overline{c},\; a \leq b \leq c,\; \overline{a} \leq a, c \leq \overline{c},\; \overline{a} + \lambda (\overline{b} - \overline{a}) \leq \overline{b} \leq \overline{c} - \lambda (\overline{c} - \overline{b})\).

### 2.2. Simultaneous Perturbation Stochastic Approximation (SPSA) Algorithm

This section introduces the SPSA algorithm briefly. The detailed description can be found in literature [23]. Consider the optimization problem with an objective function \(f(W)\), the SPSA algorithm updates \(W\) by

\[
W(k+1) = W(k) - a_k g(W(k)),
\]

where \(g(\cdot)\) is the estimated gradient result of objective function \(f(\cdot)\) with respect to \(W\), that is, \(\partial f(W) / \partial W \approx g(W)\). \(a_k\) denotes the learning step length which is decreased over iterations with \(a_k = a / (k + A)^{\alpha}\), where \(a, A,\) and \(\alpha\) are positive configuration coefficients [23]. The SPSA approach estimates the gradient, \(g(\cdot)\), using the following method. Assume that the dimension of parameter \(W\) is \(p\). Let \(\Delta_k = [\Delta_{k1} \Delta_{k2} \cdots \Delta_{kp}]\) be a \(p\)-dimensional vector whose element is mutually independent zero-mean random variable. Then, the estimation of the gradient at \(k\)th iteration can be computed by

\[
g(W(k)) = \frac{f(W(k) + c_k \Delta_k) - f(W(k))}{c_k} \left[ \Delta_k^{-1} \Delta_{k2}^{-1} \cdots \Delta_k^{-1} \right]^T,
\]

where \(c_k\) is gain sequence that is also decreased with \(c_k = c / (k + 1)^{\gamma}\), where \(c\) and \(\gamma\) are nonnegative configuration coefficients [23]. Obviously, all elements of \(W\) are perturbed simultaneously and only two measurements of the objective function are needed to estimate the gradient. In addition, \(\Delta_k\) is usually obtained using Bernoulli \(\pm 1\) distribution with equal probability for each value.

In general, the gradient information of neural fuzzy system is not easy to obtain due to the piecewise continuous property of AFMFs and large number of adjustable parameters. Herein, we adopt the SPSA algorithm to derive the stable learning for guaranteeing the convergence and stability of the closed-loop systems.

### 3. Interval Type-2 Recurrent Fuzzy Neural System with Asymmetric Membership Function (IT2RFNS-A)

#### 3.1. Fuzzy Reasoning of IT2RFNS-A

The proposed IT2RFNS-A realizes the fuzzy inference into the network structure. Assume that an IT2RFNS-A system has \(M\) rules and \(n\) inputs, the \(j\)th rule can be expressed as

\[
R^j: \text{if } u_{i1}, \ldots, u_{in} \text{ is } \tilde{F}_{nj}, \text{ then } Y_j = C_{j0} + C_{j1}x_1 + C_{j2}x_2 + \cdots + C_{jn}x_n,
\]
where \( u_{ij} \) is input linguistic variable of the \( j \)th rule, \( \tilde{F}_{ij} \) are interval type-2 antecedent fuzzy sets, \( C_{ji} \) are consequent interval fuzzy set, \( x_i \) is the network input, and \( Y_j \) is the output of the \( j \)th rule. Note that the input linguistic variable \( u_{ij} \) has the system input term and the past information which is introduced in Section 3.2. As the above description of T2 FLSs, the membership grade is an interval valued set which consists of the lower and upper membership grades, that is,

\[
\mu_{\tilde{F}_{ij}}(u_{ij}) = \left[ \mu_{\tilde{F}_{ij}}^l(u_{ij}) \quad \mu_{\tilde{F}_{ij}}^u(u_{ij}) \right]
\]  

and the consequent part is

\[
C_{ji} = [c_{ji} - s_{ji} \quad c_{ji} + s_{ji}],
\]

where \( c_{ji} \) denotes the center of \( C_{ji} \) and \( s_{ji} \) denotes the spread of \( C_{ji} \). Therefore, using the product \( t \)-norm, the firing strength associated with the \( j \)th rule is

\[
F_j(u) = \left[ f_j(u) \quad \bar{f}_j(u) \right],
\]

where \( f_j(u) = \mu_{\tilde{F}_{ij}}(u_{ij}) \times \cdots \times \mu_{\tilde{F}_{nj}}(u_{nj}) \) and \( \bar{f}_j(u) = \mu_{\tilde{F}_{ij}}(u_{ij}) \times \cdots \times \mu_{\tilde{F}_{nj}}(u_{nj}) \). Thus, the consequent part of the \( j \)th rule is

\[
y_j^f = \left( c_{j0} + \sum_{i=1}^n c_{ji}u_{ij} \right) - \left( s_{j0} + \sum_{i=1}^n s_{ji}|u_{ij}| \right),
\]

\[
y_j^r = \left( c_{j0} + \sum_{i=1}^n c_{ji}u_{ij} \right) + \left( s_{j0} + \sum_{i=1}^n s_{ji}|u_{ij}| \right).
\]

By applying the Extension principle [24], the output of the FLS is

\[
Y_{\text{TSK}}(u) = \left[ y^f \quad y^r \right] = \frac{\int_{y_i \in [y_i]} \cdots \int_{y_M \in [y_M]} \int_{f_1 \in [f_1]} \cdots \int_{f_M \in [f_M]} f_1 \cdots f_M 1}{\sum_{j=1}^M f_j y_j / \sum_{j=1}^M f_j}.
\]

To compute \( Y_{\text{TSK}}(u) \), we need to compute its two end-points \( y^f \) and \( y^r \) by type-reduction operation. Karnik and Mendel developed an iterative algorithm which is known as KM algorithm to compute these two end-points [24, 25]. A type reducer combines all fired-rule output sets in some way, just like a type-2 defuzzifier combines the type-1 rule output sets, which leads to a T1 FS that is called a type-reduced set. Herein, we need to compute the left-end point \( y^f \) and right-end point \( y^r \) by utilizing the KM algorithm [24, 25]. In the KM algorithm, the left-end and right-end points are represented as

\[
y^f = \frac{\sum_{i=1}^L f_i y_i + \sum_{i=1}^M f_i y_i}{\sum_{i=1}^L f_i + \sum_{i=1}^M f_i}, \quad y^r = \frac{\sum_{i=1}^R f_i y_i + \sum_{i=1}^M f_i y_i}{\sum_{i=1}^R f_i + \sum_{i=1}^M f_i}.
\]
Then, we have to find the proper switch point value $L$ and $R$ by iterative procedure where more detail can be referred to [24, 25]. However, this iterative procedure for finding switch point values is time wasting. Hence, in the proposed IT2RFNS-A, a simple weight-average method is used to approximate the $L$ and $R$ so that the iterative procedure is unnecessary. That is, we can calculate the left-most and right-most point of firing strength by

$$f^l_j = \frac{\omega^l_j f_j + \omega^r_j f_j}{\omega^l_j + \omega^r_j}, \quad f^r_j = \frac{\omega^l_j f_j + \omega^r_j f_j}{\omega^l_j + \omega^r_j},$$

where $\omega^l_j$, $\omega^r_j$, $\omega^l_j$, and $\omega^r_j$ are adjustable weights. Then, we can obtain the following left-end point and right-end point of the output of interval type-2 fuzzy inference system

$$y^l = \frac{\sum_{j=1}^{M} f^l_j y^l_j}{\sum_{j=1}^{M} f^l_j}, \quad y^r = \frac{\sum_{j=1}^{M} f^r_j y^r_j}{\sum_{j=1}^{M} f^r_j}.$$  

Note that the above simplified type-reduction technique is adopted in layer 4 (left-most and right-most layer) and feedback layer of IT2RFNS-A system. This simplifies the computational effort in type-reduction procedure. Finally, we defuzzify the type-reduced set to get a crisp output, that is,

$$y(u) = \frac{y^l + y^r}{2}. \quad (3.10)$$

**3.2. Network Structure of IT2RFNS-A**

The proposed IT2RFNS-A consists of six feed-forward layers and a feedback one. Layer 1 accepts input variables. Layer 2 is used to calculate the IT2 AFMF grade. The feedback layer embedded in Layer 2 is used to store the past information. Layer 3 forms the fuzzy rule base. Layer 4 introduces the simplified type-reduction scheme, called left-most and right-most layer. The TSK-type consequent part is implemented into Layer 5. Layer 6 is the output layer.

Next, we indicate the signal propagation and the operation functions of the node in each layer. For convenience, the multi-input-single-output case is considered here. The schematic diagram of the proposed IT2RFNS-A is shown in Figure 2. In the following description, $O^{(i)}_l$ denotes the $i$th output node in layer $l$.

**Layer 1: Input Layer**

For the $i$th node of layer 1, the net input and output are represented as

$$O^{(1)}_i = x_i. \quad (3.11)$$
where \( x_i \) represents the \( i \)th input to the \( i \)th node. The nodes in this layer only transmit input to the next layer directly.

**Layer 2: Membership Layer**

In this layer, each node performs a triangular IT2 AFMF, that is,

\[
O_{ij}^{(2)} = \mu_{\tilde{F}_{ij}} \left( O_i^{(1)} + O_i^{(f)} \right)
\]

\[
= \left[ O_{ij}^{(2)} \quad O_{ij}^{(2)} \right]^T
\]

\[
= \left[ \mu_{\tilde{F}_{ij}} \left( O_i^{(1)} + O_i^{(f)} \right) \quad \mu_{\tilde{F}_{ij}} \left( O_i^{(1)} + O_i^{(f)} \right) \right]^T,
\]

where the subscript \( ij \) indicates the \( j \)th term of the \( i \)th input and \( \mu_{\tilde{F}_{ij}} \) is an IT2 AFMF as shown in Figure 1 and (3.2). Note that the output of layer 2 and the feedback weight are interval values. According to the above description and the results of literature [26], the type reduction is embedded in the network. Herein, the output of feedback layer is expressed as

\[
O_i^{(f)}(k) = \frac{O_{ij}^{(2)}(k-1) \cdot \theta_{ij} + \bar{O}_{ij}^{(2)}(k-1) \cdot \bar{\theta}_{ij}}{\tilde{\theta}_{ij} + \bar{\theta}_{ij}},
\]

where \( \theta_{ij} \) and \( \bar{\theta}_{ij} \) are the left and right bounds of the feedback weight, respectively.
where $\theta_{ij}$ and $\bar{\theta}_{ij}$ denote the link weight of the feedback layer. Clearly, the input of this layer contains the memory terms $O_{ij}^{(2)}(k-1)$ and $\bar{O}_{ij}^{(2)}(k-1)$ which store the past information of the network.

**Layer 3: Rule Layer**

This layer is used for computing firing strength of fuzzy rule. From (3.4), we obtain

$$
\begin{align*}
\bar{F}_j &= \mu_{\bar{F}_{ij}}(O_1^{(1)} + O_1^{(f)}) \times \cdots \times \mu_{\bar{F}_{nj}}(O_n^{(1)} + O_n^{(f)}), \\
\bar{F}_j &= \mu_{\bar{F}_{ij}}(O_1^{(1)} + O_1^{(f)}) \times \cdots \times \mu_{\bar{F}_{nj}}(O_n^{(1)} + O_n^{(f)}),
\end{align*}
$$

(3.14)

where $\mu_{\bar{F}_{ij}}(\cdot)$ and $\bar{\mu}_{\bar{F}_{ij}}(\cdot)$ are the lower and upper membership grades. Therefore, the operation function in this layer is

$$
O_j^{(3)} = [\bar{F}_j \ T_j]^T = \begin{bmatrix} O_j^{(3)} \\ O_j^{(3)} \end{bmatrix}^T = \begin{bmatrix} \prod_{i=1}^{n} O_{ij}^{(2)} \ \prod_{i=1}^{n} \bar{O}_{ij}^{(2)} \end{bmatrix}^T.
$$

(3.15)

**Layer 4: Left-Most & Right-Most layer**

Similar to the feedback layer, the type reduction is integrated into the network structure by calculating their left-most and right-most values. That is, the complicated type-reduction method such as KM algorithm can be reduced as

$$
O_j^{(4)} = \begin{bmatrix} O_j^{(4)} \\ O_j^{(4)} \end{bmatrix}^T = \begin{bmatrix} \omega_j^{\bar{O}_j^{(3)}} + \omega_j^{O_j^{(3)}} \omega_j^{\bar{O}_j^{(3)}} + \omega_j^{O_j^{(3)}} \omega_j^{\bar{O}_j^{(3)}} + \omega_j^{O_j^{(3)}} \omega_j^{\bar{O}_j^{(3)}} + \omega_j^{O_j^{(3)}} \end{bmatrix}^T,
$$

(3.16)

where the link weights are $\omega_j^r = [\omega_1^r \cdots \omega_M^r]^T$, $\omega_j^l = [\omega_1^l \cdots \omega_M^l]^T$, $\omega_j^r = [\omega_1^r \cdots \omega_M^r]^T$, and $\omega_j^l = [\omega_1^l \cdots \omega_M^l]^T$. These link weights are adjusted by the proposed stable SPSA learning algorithm. Note that $\omega_j^l < \omega_j^r$ and $\omega_j^l < \omega_j^r$.

**Layer 5: TSK Layer**

From (3.5), the TSK-type consequent part is

$$
T_j = \begin{bmatrix} T_j^l \\ T_j^r \end{bmatrix}^T = \begin{bmatrix} (c_{j0} + \sum_{i=1}^{n} c_{ji}x_i) - (s_{j0} + \sum_{i=1}^{n} s_{ji|x_i}) \\ (c_{j0} + \sum_{i=1}^{n} c_{ji}x_i) + (s_{j0} + \sum_{i=1}^{n} s_{ji|x_i}) \end{bmatrix}^T.
$$

(3.17)
Then, the output of this layer is

\[
O^{(5)} = \left[ O^{(5)}_l \ O^{(5)}_r \right]^T = \begin{bmatrix} \sum_{j=1}^{M} O^{(4)}_{jl} T_j^l \sum_{j=1}^{M} O^{(4)}_{jr} T_j^r \end{bmatrix}^T.
\] (3.18)

**Layer 6: Output Layer**

Layer 6 is the output layer which is used to implement the defuzzification operation. Therefore, the crisp output is

\[
O^{(6)} = \frac{O^{(5)}_l + O^{(5)}_r}{2}.
\] (3.19)

Obviously, the total design parameters of IT2RFNS-A are \( \overline{a}, \overline{b}, \overline{c}, \ \overline{a}, \overline{b}, \overline{c}, \ \lambda, \ \omega_l^l, \ \omega_l^r, \ \omega_r^l, \ \omega_r^r, \ \omega_l^j, \ \omega_r^j, \ c_{ji}, \) and \( s_{ji}. \) These parameters are adjusted by the proposed stable SPSA learning algorithm which guarantees the convergence of IT2RFNS-A systems.

### 3.3. Training of IT2RFNS-A by the Stable SPSA Algorithm

Consider the nonlinear control problem, our goal is to generate a proper control sequence \( u(k) \) such that the system output \( y(k) \) follows the desired trajectory \( y_r(k) \), where \( k \) is the discrete-time index. The IT2RFNS-A with the stable SPSA algorithm plays the role of controller for nonlinear plant, the adaptive control scheme is shown in Figure 3. For convenience, we consider the single-output case and the tracking error is defined as

\[
e(k) = y_r(k) - y(k).
\] (3.20)

Then, we define the objective function (error cost function) as

\[
E(k) = \frac{1}{2} e^2(k) = \frac{1}{2} (y_r(k) - y(k))^2.
\] (3.21)
The control objective is to generate the control signal $u(k)$ such that the tracking error approaches to zero, that is, to minimize the objective function $E(k)$. From the well-known gradient descent method, the parameter update law can be written as

$$W(k + 1) = W(k) + \Delta W(k) = W(k) + a_k \left( -\frac{\partial E(k)}{\partial W} \right),$$  \hspace{1cm} (3.22)

where $W$ is the tuning parameters of IT2RFNS-A. Then,

$$\frac{\partial E(k)}{\partial W} = -e(k) \frac{\partial e(k)}{\partial W}. \hspace{1cm} (3.23)$$

Herein, we adopt the SPSA algorithm to reduce the computational complexity. The parameters update laws can be represented as

$$W(k + 1) = W(k) + a_k e(k) g(W(k)), \hspace{1cm} (3.24)$$

where

$$g(W(k)) = \frac{e(W(k) + c_k \Delta_k) - e(W(k))}{c_k} \left[ \Delta_k^{-1} \Delta_k^{-1} \cdots \Delta_k^{-1} \right]^T, \hspace{1cm} (3.25)$$

where $e(W(k) + c_k \Delta_k)$ denotes the tracking error between the desired output and system output resulted by the IT2RFNS-A with perturbed tuning parameters. Note that (3.25) does not calculate the system sensitivity or gradient functions; this simplifies the computational effort.

Note that, in training neural networks, it may not be possible to update all estimated parameters with a single gradient approximation function (3.25). Thus, we should partition the estimated parameters $W$ into several parts, that is, each kind of parameter has its corresponding estimated parameter (e.g., $W^o$ as the estimated link weight of feedback layer). Then, the estimated parameters are updated separately by the SPSA algorithm.

**Remark 3.1.** As the previous results of [1, 2, 4], the adaptive control laws can be obtained by multiplying system sensitivity $\partial y / \partial u$, that is,

$$\frac{\partial E(k)}{\partial W} = e(k) \frac{\partial (y_r(k) - y(k))}{\partial W} = e(k)(-1) \frac{\partial y(k)}{\partial W} = e(k) \cdot (-1) \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial W}, \hspace{1cm} (3.26)$$

where $u$ is the control input produced by the IT2RFNS-A, that is, $u = O^6$. Hence, the gradient of the IT2RFNS-A should be calculated. However, an additional identifier (using a FNN or an IT2RFNS-A) should be developed to find the unknown system’s sensitivity (details can be found in [1, 2]). In this way, the approximation accuracy should be guaranteed for training.
the IT2RFNS-A controller and the computational effort is complex and huge. Therefore, an approximation term of $\frac{\partial E(k)}{\partial W} \approx -(e + \Delta e) \frac{\partial u}{\partial W}$ and the optimal learning step length are derived to enhance the efficiency and guarantee the stability, where $\Delta e$ denotes $\dot{e}$ and $e(k) - e(k - 1)$ for continuous and discrete time case, respectively.

Remark 3.2. Our proposed approach is also valid for the nonlinear system identification. The series-parallel architecture shown in Figure 4 is adopted. Hence, the inputs of IT2RFNS-A are $u$ and $y(k - 1)$ and the IT2RFNS-A output is the estimated output $\hat{y}(k)$. The parameters of IT2RFNS-A are tuned by the proposed stable SPSA algorithm. Thus, the parameter update law is

$$W(k + 1) = W(k) + a_k [y(k) - \hat{y}(k)] \cdot \frac{O^{(6)}(W(k) + c_k \Delta_k) - O^{(6)}(W(k))}{c_k} [\Delta_k^{-1} \Delta_{k-1}^{-1} \cdots \Delta_{k-p}^{-1}]^T,$$

(3.27)

where $O^{(6)}(W(k) + c_k \Delta_k)$ denotes the output of the IT2RFNS-A with perturbed tuning parameters.

### 3.4. Stability Analysis

In this section, the convergence theorem for selecting appropriate learning step length $a_k$ is introduced. The choice of learning step length is very important for convergence. If a small value is given for the learning step length, then the convergence of the IT2RFNS-A is guaranteed. However, the convergent speed may be slow. On the other hand, if a large value is selected, then the system may be unstable. Hence, we employ the Lyapunov stability approach to have the condition for convergence and find the optimal learning step length for IT2RFNS-A.
Theorem 3.3. Let $a_k$ be the learning step length of tuning parameters for the IT2RFNS-A controller. Consider the nonlinear control problem using IT2RFNS-A (shown in Figure 3), the asymptotic convergence of the closed-loop system is guaranteed if the learning step length is chosen satisfying

$$0 < a_k < \frac{2}{|g(k)|^2}, \quad \forall k,$$

(3.28)

where $g(W(k)) = -(e + \Delta e/e) \cdot (O^{(6)}(W(k) + c_k \Delta_k) - O^{(6)}(W(k))/c_k)[\Delta_{k1}^{-1} \Delta_{k2}^{-1} \cdots \Delta_{kp}^{-1}]^T$ is the gradient estimation using SPSA approach. In addition, the faster convergence can be obtained by the following optimal time-varying learning step length

$$a^*_k = \frac{1}{|g(k)|^2}.$$  

(3.29)

Proof. First, we define the discrete-time Lyapunov function as follows:

$$V(k) = E(k) = \frac{1}{2} e^2(k) = \frac{1}{2} (y_r(k) - y(k))^2,$$

(3.30)

where $e(k)$ represents the tracking error. Then, the change of the Lyapunov function is

$$\Delta V(k) = V(k+1) - V(k) = \frac{1}{2} \left(e^2(k+1) - e^2(k)\right).$$

(3.31)

According to the Lyapunov stability theorem, if the change of the positive definite Lyapunov function, denoted $\Delta V(k)$, satisfies the condition $\Delta V(k) < 0$, for all $k$, then the asymptotical stability is guaranteed [1, 2, 27]. Hence, our objective is to select the proper learning step length such that $\Delta V(k) < 0$, for all $k$. This implies that $V(k)$ will converge to zero when $k$ approaches to infinity. By [1, 2, 28], the error difference can be represented as

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

(3.32)

where $\Delta W$ denotes the change of $W$. From (3.22) and (3.24), we obtain

$$\Delta W = -a_k e(k) \frac{\partial e(k)}{\partial W} \equiv a_k e(k) g(k).$$

(3.33)
Thus,

\[
\Delta V(k) = \frac{1}{2} \left[ e^2(k + 1) - e^2(k) \right]
\]
\[
= \frac{1}{2} [e(k + 1) - e(k)] \cdot [e(k + 1) + e(k)]
\]
\[
= \frac{1}{2} \Delta e(k) \cdot [2e(k) + \Delta e(k)]
\]
\[
= \Delta e(k) \cdot \left( e(k) + \frac{1}{2} \Delta e(k) \right)
\]
\[
= \left[ \frac{\partial e(k)}{\partial W} \right]^T a_k e(k) g(k) \cdot \left\{ e(k) + \frac{1}{2} \left[ \frac{\partial e(k)}{\partial W} \right]^T a_k e(k) g(k) \right\}
\]
\[
= -a_k e^2(k) |g(k)|^2 \left( 1 - \frac{1}{2} a_k |g(k)|^2 \right).
\]

Note that \( P(k) \) is positive for all \( k > 0 \), and let \( P(k) = a_k |g(k)|^2 \). Thus,

\[
\Delta V(k) = -e^2(k) P(k) \left( 1 - \frac{1}{2} P(k) \right). \tag{3.35}
\]

Recall that the asymptotical stability of the IT2RFNS-A is guaranteed if \( \Delta V(k) < 0 \), for all \( k > 0 \). Thus, \( (1 - (1/2) P(k)) \) should be positive such that \( \Delta V(k) < 0 \), for all \( k \). Therefore, we obtain the stability condition for \( a_k \):

\[
0 < a_k < \frac{2}{|g(k)|^2}. \tag{3.36}
\]

The asymptotic stability is guaranteed if \( a_k \) is chosen to satisfy (3.36). In addition, we would like to find a condition for \( a_k \) that guarantees the fast convergence. From (3.34) and (3.35), we have

\[
e^2(k + 1) = e^2(k) - e^2(k) P(k) (2 - P(k))
\]
\[
= e^2(k) \left[ 1 - 2P(k) + P^2(k) \right]
\]
\[
= e^2(k) |P(k) - 1|^2. \tag{3.37}
\]
The minimum of $e^2(k + 1)$ is achieved when $P(k) = 1$. Hence, the time-varying optimal learning step length is

$$a_k^* = \frac{1}{|g(k)|^2}. \quad (3.38)$$

For the system identification problem, a similar convergence theorem can be obtained.

**Theorem 3.4.** Let $a_{Ik}$ be the learning step length of tuning parameters for the IT2RFNS-A. Consider the nonlinear identification problem by series-parallel architecture using IT2RFNS-A (shown in Figure 4), and the parameters update laws are shown in (3.27). The asymptotic convergence of the IT2RFNS-A system is guaranteed if the chosen learning step length is satisfied:

$$0 < a_{Ik} < \frac{2}{|g_I(k)|^2}, \quad \forall k, \quad (3.39)$$

where $g_I(k) = (O^{(6)}(W(k) + c_k \Delta_k) - O^{(6)}(W(k))/c_k)[\Delta^{-1}_{k1} \Delta^{-1}_{k2} \cdots \Delta^{-1}_{kp}]^T$ is the gradient estimation of IT2RFNS-A by SPSA approach. In addition, the faster convergence can be obtained by using the following optimal time-varying learning step length:

$$a_{Ik}^* = \frac{1}{|g_I(k)|^2}. \quad (3.40)$$

**Proof.** Herein, we omitted it due to that the proof of Theorem 3.4 is similar to the proof of Theorem 3.3. Only the express of the estimated gradient function $g(k)$ is replaced by $g_I(k)$.

**Remark 3.5.** As above description, the SPSA algorithm has the stochastic property in the estimation of gradient functions $g(k)$ and $g_I(k)$ due to the random values $c_k$ and $\Delta_k$. Therefore, the stability conditions (3.28) and (3.39) are important for training. In addition, the time-varying optimal learning step lengths $a^*_{ik}$ and $a^*_{Ik}$, shown in (3.29) and (3.40), have the ability of guaranteeing the high-speed convergence. This reduces the effect of random values. Simulation results in Section 4 demonstrate the performance of our approach.

**4. Simulation Results**

Simulation results and comparisons, including the nonlinear system identification and the control of nonlinear system, are introduced to verify the performance of our proposed approach.

**4.1. Example 1: Nonlinear System Identification of Chaotic System**

Consider a nonlinear system described as

$$y(k) = f(y(k-1), \ldots, y(k-n); u(k-1), \ldots, u(k-m)), \quad (4.1)$$
where \( u \) denotes the input, \( y \) is the output, \( m \) and \( n \) are positive integers, and \( f(\cdot) \) is the unknown function \( f : \mathbb{R}^{m+n} \to \mathbb{R} \). Our purpose is to identify the nonlinear system (4.1) using IT2RFNS-A with the proposed stable SPSA algorithm. From literature [2], the chaotic system is described as

\[
y(k+1) = -P \cdot y^2(k) + Q \cdot y(k-1) + 1.0,
\]

where \( P = 1.4 \) and \( Q = 0.3 \) and produces a chaotic strange attractor as shown in Figure 5(a). We first choose 1000 pairs training data randomly from the system over the interval \([-1.5, 1.5]\) and the initial point is set to \([y(1) y(0)] = [0.4 0.4]\). Then, the IT2RFNS-A is used to approximate the chaotic system using the stable SPSA algorithm with the following coefficients listed in Table 1. After training for 100 epochs, the phase plane of the chaotic system is shown in Figure 5(b). In order to show the superiority of using optimal learning step length, the same simulation is done without using optimal learning step length for comparison. In addition, we also compare it with other neural networks such as fuzzy neural network (FNN) [1], recurrent fuzzy neural network (RFNN) [2], and interval type-2 fuzzy logic system with asymmetric membership function (IT2FNS-A) [16]. For statistical analysis, the learning process is repeated for 10 independent runs. The descending trend of the mean square error (MSE) is shown in Figure 6, solid-black line: IT2RFNS-A with the optimal learning step length \( a_{ik}^* \); solid-red line: IT2RFNS-A without the optimal learning step length; dashed line: IT2FNS-A; dotted line: RFNN; dash-dotted line: FNN. The worst, average, and best MSE that each neural network achieved are shown in Table 2. Obviously, the IT2RFNS-A with optimal learning step length achieves the best performance and if we focus on using optimal learning step length or not, then we can find that there is a great improvement. From Figure 6, we can observe that using optimal learning step length results in better performance in convergence. Hence, we accumulate the number of epochs that MSE decreased in 100 epochs and listed in Table 3. The result, evidently, shows that the optimal learning step length can guarantee the efficient training of IT2RFNS-A.
Table 1: Configuration coefficients for Example 1: system identification.

<table>
<thead>
<tr>
<th></th>
<th>FNN</th>
<th>RFNN</th>
<th>IT2FNS-A</th>
<th>IT2RFNS-A</th>
<th>IT2RFNS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.01</td>
<td>0.01</td>
<td>optimal learning step length (3.40)</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>0.602</td>
<td>0.602</td>
<td>0.602</td>
<td>0.602</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.101</td>
<td>0.101</td>
<td>0.101</td>
<td>0.101</td>
<td>0.101</td>
</tr>
</tbody>
</table>

Neural network coefficient rule number 8 8 2 2 2

epochs 100

Table 2: Comparison results in MSE for Example 1.

<table>
<thead>
<tr>
<th></th>
<th>FNN</th>
<th>RFNN</th>
<th>IT2FNS-A</th>
<th>IT2RFNS-A</th>
<th>IT2RFNS-A (optimal $a_{ik}^*$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>0.0048</td>
<td>0.0017</td>
<td>6.9579 $\times 10^{-5}$</td>
<td>1.9464 $\times 10^{-5}$</td>
<td>5.0267 $\times 10^{-6}$</td>
</tr>
<tr>
<td>Average</td>
<td>0.0028</td>
<td>6.6921 $\times 10^{-4}$</td>
<td>3.4154 $\times 10^{-5}$</td>
<td>4.8760 $\times 10^{-6}$</td>
<td>1.3319 $\times 10^{-5}$</td>
</tr>
<tr>
<td>Best</td>
<td>0.0018</td>
<td>2.2731 $\times 10^{-4}$</td>
<td>1.1265 $\times 10^{-5}$</td>
<td>6.0900 $\times 10^{-8}$</td>
<td>1.0091 $\times 10^{-27}$</td>
</tr>
</tbody>
</table>

4.2. Nonlinear System Control of Chua’s Chaotic Circuit

Consider an $n$th-order nonlinear dynamic system in the companion form or controllability canonical from

$$x^{(n)} = f(x) + g(x)u + d,$$

$$y = x,$$

where $u$ and $y$ are the control input and output of the nonlinear system, $f(\cdot)$ and $g(\cdot)$ are unknown nonlinear and continuous functions, and $d$ is the external disturbance or system uncertainty. Our purpose is to develop an IT2RFNS-A controller to generate the proper control signal such that the system output $y$ can follow a given reference trajectory $y_r$.

In this paper, we consider the typical Chua’s chaotic circuit which consists of one inductor, two capacitors, one linear resistor, and one piecewise-linear resistor [29, 30]. It has been shown that this circuit holds very rich nonlinear dynamics such as chaos and bifurcations. The dynamic equation of Chua’s circuit are described as

$$\frac{dv_{C_1}}{dt} = (C_1)^{-1}\left(\frac{v_{C_1} - v_{C_2}}{R} - \gamma\right),$$

$$\frac{dv_{C_2}}{dt} = (C_2)^{-1}\left(\frac{v_{C_1} - v_{C_2}}{R} - i_L\right),$$

$$\frac{di_L}{dt} = (L)^{-1}(-v_{C_1} - R_0i_L),$$

$$\frac{dv_{C_1}}{dt} = (C_1)^{-1}\left(\frac{v_{C_1} - v_{C_2}}{R} - \gamma\right),$$

$$\frac{dv_{C_2}}{dt} = (C_2)^{-1}\left(\frac{v_{C_1} - v_{C_2}}{R} - i_L\right),$$

$$\frac{di_L}{dt} = (L)^{-1}(-v_{C_1} - R_0i_L),$$

(4.3)
Table 3: Comparison results of using optimal learning step length (number of epochs that MSE decreases in 100 epochs, 10 independent runs).

<table>
<thead>
<tr>
<th>Number of epochs</th>
<th>IT2RFNS-A</th>
<th>IT2RFNS-A with optimal $a^*_l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>38</td>
<td>68</td>
</tr>
<tr>
<td>Average</td>
<td>36</td>
<td>89</td>
</tr>
<tr>
<td>Best</td>
<td>55</td>
<td>98</td>
</tr>
<tr>
<td>Total epochs</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 6: Comparison results in MSE of Example 1.

where voltages $v_{C_1}$, $v_{C_2}$ and current $i_L$ are state variables, $R_O$ is a constant, and $\gamma$ denotes the nonlinear resistor which is a function of voltage across two terminals of $C_1$. $\gamma$ is defined as a cubic function as

$$\gamma = aV_{C_1} + c(v_{C_1})^3 \quad (a > 0, \ c < 0).$$

(4.5)

According to [30], (4.4) can be transferred into the canonical form as

$$\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
1
\end{bmatrix} \cdot (f + g \cdot u + d),$$

(4.6)

$$y = \begin{bmatrix}
1 & 0 & 0
\end{bmatrix} \cdot \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}.$$
Table 4: The selected coefficients for Example 2.

<table>
<thead>
<tr>
<th></th>
<th>FNN</th>
<th>RFNN</th>
<th>IT2FNS-A</th>
<th>IT2RFNS-A</th>
<th>IT2RFNS-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>optimal learning step length (3.29)</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.602</td>
<td>0.602</td>
<td>0.602</td>
<td>0.602</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.101</td>
<td>0.101</td>
<td>0.101</td>
<td>0.101</td>
<td>0.101</td>
</tr>
<tr>
<td>Neural network coefficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rule number</td>
<td>12</td>
<td>12</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Time (sec.)</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Comparison results of control performance for Example 2.

<table>
<thead>
<tr>
<th></th>
<th>FNN</th>
<th>RFNN</th>
<th>IT2FNS-A</th>
<th>IT2RFNS-A</th>
<th>IT2RFNS-A (optimal $a_k$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>2.6046</td>
<td>2.5321</td>
<td>0.3906</td>
<td>0.7264</td>
<td>0.0682</td>
</tr>
<tr>
<td>Average</td>
<td>1.2580</td>
<td>1.2005</td>
<td>0.3146</td>
<td>0.2768</td>
<td>0.0476</td>
</tr>
<tr>
<td>Best</td>
<td>0.7599</td>
<td>0.6532</td>
<td>0.2605</td>
<td>0.0713</td>
<td>0.0307</td>
</tr>
</tbody>
</table>

Simulation results of trajectories are shown in Figure 7. Figure 7(a) shows the phase plant trajectory after being controlled; Figures 7(b)–7(d) shows the state trajectories $x_1$, $x_2$, and $x_3$, respectively (solid line: actual system output; dashed line: reference trajectory). Figure 8 depicts the control effort of the IT2RFNS-A controller. We can observe that the proposed method is valid to control a nonlinear dynamic system. The comparison results are shown in Table 5 which is obtained by 10 independent runs. From Table 5, we can easily find that the IT2RFNS-A performs better than others. This result shows that the IT2RFNS-A has ability to cope with dynamic problems. Observe the performance of the case with optimal learning step length from Table 5, we again demonstrate that adopting optimal learning step length guarantees the efficient training of IT2RFNS-A.

5. Conclusion

In this paper, we have proposed an interval type-2 recurrent fuzzy neural system with triangular asymmetric membership functions (IT2RFNS-A) and its training scheme using the proposed stable SPSA algorithm. We adopt the Lyapunov theorem to derive the appropriate range of learning step length for SPSA to guarantee the stability of the closed-loop system for nonlinear control and guarantee the convergence of IT2RFNS-A for system identification. In addition, we also obtain the optimal learning step length that ensures the efficient training for IT2RFNS-A. The feasibility and the effectiveness of the proposed method have been demonstrated by two illustration examples. The simulations on the chaotic system identification and the control of the Chua’s chaotic circuit are done, and both of them show the following advantages of proposed approach, (a) the IT2RFNS-A having few rules can achieve the specific performance or even better; (b) the IT2RFNS-A can catch the dynamic response of the system; that is, it is capable of coping with dynamic problems; (c) by using the proposed stable SPSA algorithm, the gradient information is unnecessary. In other words, a great deal of computational effort is saved; (d) for control problems, the proposed method
Figure 7: Simulation results of Example 2: (a) phase plant trajectory of IT2RFNS-A, (b) state $x_1$ and reference trajectory $y_r$, (c) state $x_2$ and reference trajectory $\dot{y}_r$, and (d) state $x_3$ and reference trajectory $\ddot{y}_r$. 
avoids approximating the system sensitivity which leads to inaccuracy; (e) adopting optimal learning step length improves the performance of training IT2RFNS-A.

Acknowledgments

The authors would like to thank the Associate Editor and anonymous reviewers for their insightful comments and valuable suggestions. This work was partially supported by the National Science Council, Taiwan, under contract no: NSC-97-2221-E-155-033-MY3.

References


