On-line Adaptive Interval Type-2 Fuzzy Controller Design via Stable SPSA Learning Mechanism

Ching-Hung Lee, Feng-Yu Chang, and Chih-Min Lin

Abstract

This paper proposes an interval type-2 Takagi-Sugeno-Kang fuzzy neural system (IT2TFNS) to develop an on-line adaptive controller using stable simultaneous perturbation stochastic approximation (SPSA) algorithm. The proposed IT2TFNS realizes an interval type-2 TSK fuzzy logic system formed by the neural network structure. Differ from the most of interval type-2 fuzzy systems, the type-reduction of the proposed IT2TFNS is embedded in the network by using uncertainty bounds method such that the time-consuming Karnik-Mendel (KM) algorithm is replaced. The proposed stable SPSA algorithm provides the gradient free property and faster convergence. However, the stable SPSA algorithm inherently has the problem for on-line adaptive control. Hence, in order to achieve the on-line result, we utilize the sliding surface to develop a new on-line adaptive control scheme. In addition, the corresponding stable learning is derived by Lyapunov theorem which guarantees the convergence and stability of the closed-loop systems. Simulation and comparison results are shown to demonstrate the performance and effectiveness of our approach.

Keywords: interval type-2 fuzzy neural system, uncertainty bounds, simultaneous perturbation stochastic approximation algorithm, Lyapunov theorem, on-line control.

1. Introduction

Type-2 fuzzy sets (T2 FSs) were introduced by Zadeh in 1975 that characterized by membership functions (MFs) that provide better ability to handle uncertainty than type-1 fuzzy sets [1]. The T2 FSs also have more parameters to provide us with more design degrees of freedom. However, they result in higher computational cost. Therefore, the interval type-2 fuzzy sets (IT2 FSs) were introduced to simplify the computations and made it quite practical [2, 3]. One of the major applications of IT2 FSs is the interval type-2 fuzzy logic system (IT2 FLS). The IT2 FLS has been successfully applied on the system control [4-10]. Recently, fuzzy neural network which provides the both advantages of neural network and fuzzy logic system has attached much attention [11-17]. As the development of the IT2 FLS, the interval type-2 fuzzy neural network (IT2FNN) which realizes the IT2 FLS into neural network structure has gained popularity [18-22]. Unlike the type-1 fuzzy neural networks, an extra type-reduction operation must be processed in IT2FNNs. Karnik and Mendel developed an iterative procedure (KM algorithm) for type-reduction [23]. It is commonly adopted for most of the IT2FNNs [18-22]. Nevertheless, the KM algorithm has the expensive computational cost, especially if the fuzzy rule number is large [24-26]. Instead, Wu and Mendel provided a new approach called uncertainty bounds method to replace the KM algorithm [26]. In this paper, we propose an interval type-2 Takagi-Sugeno-Kang fuzzy neural system (IT2TFNS) which take advantage of the uncertainty bounds for the type-reduction operation. In the IT2TFNS, instead of process iterative KM algorithm, only several equations are needed to obtain type-reduced result. In other words, the IT2TFNS has potential to provide the lower computational complexity than other IT2FNNs.

For training of the fuzzy neural systems, we usually adopt the gradient descent method due to the fast convergence speed and stability analysis. However, the gradient information of objective function of the IT2FNNs is difficult to obtain. Thus, we propose a stable simultaneous perturbation stochastic approximation (SPSA) algorithm to deal with this problem. The original SPSA algorithm has been applied on many applications such as function optimization, system control, system identification, parameter estimation, signal processing, and experimental design [27-29]. It only needs the measurements of objective function to form the gradient information. This simplifies the calculation of gradient term. However, due to the inherently stochastic characteristic of the SPSA algorithm, we cannot guarantee that every searching is appropriate. To overcome this situation, we
employ the Lyapunov stability analysis to derive a time-variant optimal learning step length for guaranteeing the stability of the system and ensuring the efficient training. Moreover, since the SPSA algorithm needs the measurements of error function and perturbed error function to form the approximated gradient information, it has a bottleneck of on-line control. Thus, in the paper, we develop a novel on-line adaptive control scheme for SPSA algorithm by using sliding surface.

The rest of this paper is organized as follows. Section 2 introduces some prerequisite materials of this paper. In Section 3, we introduce the proposed IT2TFNS. The stable SPSA algorithm-based on-line adaptive controller design using IT2TFNS is presented in Section 4. The simulation and comparison results of nonlinear systems are shown in Section 5. Finally, the conclusion is given in Section 6.

2. Preliminaries

In this section, we briefly introduce some prerequisite materials including interval type-2 TSK fuzzy logic, construction of interval type-2 asymmetric fuzzy membership functions, and SPSA algorithm.

A. Interval Type-2 Takagi-Sugeno-Kang Fuzzy Logic System

Considering an interval type-2 Takagi-Sugeno-Kang fuzzy logic system (IT2 TSK FLS) having \( n \) inputs \( x_i \) \((i=1, 2, \ldots, n)\) and \( M \) rules, the \( j \)th fuzzy IF-THEN rule can be expressed as

\[
R^j: \text{IF } x_i = F^j_i \text{ and } \ldots \text{ and } x_n = F^j_n, \text{ THEN } Y_j = C^j_0 + C^j_1 x_1 + C^j_2 x_2 + \ldots + C^j_n x_n
\]

where \( j=1, 2, \ldots, M; \) \( F^j_i \) are interval type-2 antecedent fuzzy sets; \( C^j_0 \) are consequent interval fuzzy sets; \( Y_j \) is the output of the \( j \)th rule. As the above description, the membership function is an interval set which consists the lower and the upper membership functions

\[
\mu_{\hat{F}^j_i}(x_i) = [\mu_{\bar{F}^j_i}(x_i) \quad \mu_{\bar{F}^j_i}(x_i)]
\]

\[
C^j_i = [c^j_i - s^j_i \quad c^j_i + s^j_i]
\]

where \( c^j_i \) denotes the center of \( C^j_i \) and \( s^j_i \) denotes the spread of \( C^j_i \). Hence, in an IT2 TSK FLS with product \( t \)-norm, the firing set of \( j \)th rule is an interval set, that is,

\[
F^j_i(x_i) = [\mu_{\hat{F}^j_i}(x_i) \quad \mu_{\bar{F}^j_i}(x_i)]
\]

The consequent part of \( j \)th rule is also an interval set, i.e.,

\[
y_j = (c^j_0 + \sum_{i=1}^n c^j_i x_i) - (s^j_0 + \sum_{i=1}^n s^j_i x_i).
\]

Finally, the output of an interval type-2 TSK fuzzy logic system is

\[
Y_{y_{jk}} = \left[y_j \quad y_j^* \right] = \left[\int_{x_1 \in \mathbb{R}} \ldots \int_{x_n \in \mathbb{R}} \overline{T}_j \sum_{j=1}^M f^j y_j \right] \frac{\sum_{j=1}^M f^j y_j}{\sum_{j=1}^M f^j y_j} \quad (7)
\]

where \( \int_{x_1 \in \mathbb{R}} \ldots \int_{x_n \in \mathbb{R}} \overline{T}_j \) and \( y_j \) are computed using (4)-(6).

Obviously, \( Y_{y_{jk}} \) is also an interval set. To compute \( Y_{y_{jk}} \), we therefore need to compute its two end-points \( y_j \) and \( y_j^* \). This process is also called the type-reduction of IT2 FLS [23]. Herein, we briefly introduce the computation procedure of KM algorithm for \( y_j \) and \( y_j^* \). Without loss of generality, we assume that \( y_j^* \) and \( y_j \) are reordered in ascending order, such that

\[
y_{j_1} \leq y_{j_2} \leq \ldots \leq y_{j_M} \quad \text{and} \quad y_{j_1} \leq y_{j_2} \leq \ldots \leq y_{j_M}^* \quad (8)
\]

Then, we can compute \( y_j^* \) by

Step 1: Initialize \( f_j^* \) by setting \( f_j^* = \left[ f_j + \overline{T_j} \right] / 2 \)

\[
\sum_{j=1}^M f^j y_j
\]

compute \( y_j^* = \frac{\sum_{j=1}^M \overline{T_j} y_j^*}{\sum_{j=1}^M f_j} \). Then, \( y_j = y_j^* \).

Step 2: Find \( L \) \((1 \leq L \leq M-1)\) such that \( y_{j_L} \leq y_j^* \leq y_{j_{L+1}} \).

Step 3: Compute \( y_j^* \) by using

\[
y_j = \frac{\sum_{j=1}^L \overline{T}_j y_j + \sum_{j=L+1}^M f_j^* y_j}{\sum_{j=1}^L \overline{T}_j + \sum_{j=L+1}^M f_j}
\]

and \( y_j^* = y_j \).

If \( y_j^* \neq y_j^* \), then go to Step 5. If \( y_j^* = y_j^* \), then stop and set \( y_j^* = y_j \).

Step 4: Set \( y_j^* = y_j^* \) and return to Step 2.

The procedure for computing \( y_j^* \) is similar to the above procedure. It just replaces \( y_j \) by \( y_j \) and find \( R \) in Step 2 such that \( y_{j_R} \leq y_j \leq y_{j_{R+1}} \).

In addition, in Step 3, we compute \( y_j \) by using

\[
y_j = \frac{\sum_{j=1}^L \overline{T}_j y_j + \sum_{j=L+1}^M f_j y_j}{\sum_{j=1}^L \overline{T}_j + \sum_{j=L+1}^M f_j}
\]

This four step iterative procedure has been proven to converge to the exact solution in no more than \( M \) iterations [2]. Finally, the defuzzified output of an IT2 TSK FLS is

\[
y_{y_{jk}} = \left(c^j_0 + \sum_{i=1}^n c^j_i x_i \right) - \left(s^j_0 + \sum_{i=1}^n s^j_i x_i \right).
\]
y(x) = \frac{y_i + y_r}{2} \quad (11)

As above description, the KM algorithm is an iterative procedure which results in complex computation. This paper utilizes the uncertainty bound method to treat the problem and implement the IT2 TSK-type FLS in a network structure.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig1.png}
\caption{Construction of IT2 triangular asymmetric fuzzy membership function.}
\end{figure}

\subsection*{B. Construction of Interval Type-2 Triangular Asymmetric Fuzzy Membership Function}

In general, the symmetric interval type-2 fuzzy membership function (IT2 FMF) is used for simplification. However, the IT2 FMF holds less uncertainty. In addition, tuning the parameter of IT2 FMF symmetrically may result in low precision. Many researches also show that adopting asymmetric fuzzy membership function (AFMF) results in better performance [30, 31]. Usually, the Gaussian membership functions (MFs) are adopted to construct the IT2 AFMF. However, they result in more computational effort for hardware implementations. Therefore, in IT2TFNS, we adopt the triangular AFMFs to construct the IT2 AFMFs, i.e., interval type-2 triangular asymmetric fuzzy membership functions (IT2 triangular AFMFs).

The IT2 triangular AFMFs consist of two triangular MFs: upper MF and lower MF, shown in Fig. 1, where \( \bar{p}_i \) and \( p_i \) (i=1, 2, 3) denote the vertex position of upper MF and lower MF, respectively; \( \lambda \) denotes the magnitude of lower MF which should be limited between 0.5 and 1 to avoid the invalid result, e.g., a small firing strength or unreasonable IT2 triangular AFMF. Besides, the following restrictions should be constrained to avoid unreasonable IT2 triangular AFMFs, i.e., \( \bar{p}_i \leq \bar{p}_2 \leq \bar{p}_3 \), \( p_2 \leq p_3 \leq p_1 \), \( \bar{p}_i \leq \bar{p}_1 \), \( p_i \leq p_3 \), and \( \bar{p}_i + \lambda(\bar{p}_3 - \bar{p}_i) \leq p_2 \leq \bar{p}_3 - \lambda(\bar{p}_1 - \bar{p}_2) \).

Comparing with the IT2 Gaussian AFMF [31], the IT2 triangular AFMF has fewer designable parameters and is less complicated. Moreover, the computation of IT2 triangular AFMF is simple because the linear operation is needed. We can also observe that the IT2 triangular AFMF can hold both uncertain width and uncertain center at the same time. This provides more uncertainty and design flexibility than symmetric one.

\subsection*{C. Simultaneous Perturbation Stochastic Approximation (SPSA) Algorithm}

The SPSA algorithm is a gradient-free algorithm which means that we do not have to derive the gradient information of the objective function such that a great deal of human effort is saved. Consider a optimization problem with an objective function \( f(W) \), the SPSA algorithm updates adjustable parameters \( W \) by

\[ W_{k+1} = W_k - \Delta g(W_k) \quad (12) \]

where \( g(.) \) is the estimated gradient result of the objective function \( f(.) \), i.e., \( \frac{\partial f(W)}{\partial W} \approx g(W) \); \( \Delta \) denotes the learning step length which is decreased over iterations by [29]

\[ \Delta_k = \frac{a}{(k + A)^\gamma} \quad (13) \]

where \( a, A, \) and \( \gamma \) are positive configuration coefficients. We can observe that (12) is similar to the parameter update law by gradient descent method. However, the gradient part is replaced by the approximation of gradient. In the SPSA algorithm, we only use the measurements of objective function to form the approximated gradient. Assume that the dimension of parameter \( W \) is \( D \). Let \( \Delta = [\Delta_1, \Delta_2, \ldots, \Delta_D] \) be \( D \)-dimension whose elements are mutually independent zero-mean random variable. Then, the estimation of the gradient at \( k \)th iteration can be computed by

\[ g(W_k) = \frac{f(W_k + c_k \Delta_k) - f(W_k)}{c_k} \Delta_1 \Delta_2 \ldots \Delta_D \quad (14) \]

where \( c_k \) is the gain sequence that is also decreased over iteration by

\[ c_k = \frac{c}{(k + 1)^\gamma} \quad (15) \]

c and \( \gamma \) are positive configuration coefficients. Equations (13) and (15) provide the convergence of optimization [29]. Observe that all elements of \( W \) are perturbed simultaneously and only two measurement 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 interval type-2 fuzzy neural system is not easy to obtain due to the piecewise continuous property of IT2 triangular AFMFs and large number of adjustable parameters. Herein, we adopt the SPSA algorithm to treat these problems.
3. Structure of Interval Type-2 Takagi-Sugeno-Kang Fuzzy Neural Systems (IT2TFNSs)

In this section, 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. Hence, we assume that the IT2TFNS with \( n \) inputs and \( M \) rules. The IT2TFNS is depicted in Fig. 2.

Layer 1: Input layer

The nodes in this layer only receive the input and transmit to the next layer directly, i.e.,
\[
O^{(1)} = x_i.
\]

Layer 2: Membership layer

In this layer, each node performs the IT2 triangular AFMF shown in Fig. 1, i.e.,
\[
O^{(2)} = \left[ \frac{\mu_{\tilde{A}_{ij}}(O^{(1)}) - \mu_{\tilde{B}_{ij}}(O^{(1)})}{\bar{\mu}_{\tilde{A}_{ij}}(O^{(1)}) - \bar{\mu}_{\tilde{B}_{ij}}(O^{(1)})} \right]^T.
\]

Layer 3: Rule layer

This layer is used for computing firing strength of the rule. According to (4), we obtain
\[
\tilde{f}_j = \mu_{\tilde{A}_{ij}}(O^{(1)}) \times \cdots \times \mu_{\tilde{B}_{ij}}(O^{(1)})
\]
\[
\overline{f}_j = \bar{\mu}_{\tilde{A}_{ij}}(O^{(1)}) \times \cdots \times \bar{\mu}_{\tilde{B}_{ij}}(O^{(1)})
\]
Therefore, the output of this layer is
\[
O^{(3)} = \left[ O^{(3)} - \overline{O}^{(3)} \right] = \left[ \prod_{i=1}^{n} O^{(2)} - \prod_{i=1}^{n} \overline{O}^{(2)} \right]^T.
\]

Layer 4: TSK layer

This layer is used to form the TSK type consequent, thus, the output of this layer is
\[
O^{(4)} = \left[ T^{i}_j T^{r}_j \right]^T = \left[ (c_{j0} + \mu_{\tilde{s}_{ij}})(s_{j0} + \mu_{\tilde{r}_{ij}}) \right]^T
\]

Layer 5: Type-reduction layer

In this layer, we calculate four uncertainty bounds defined in [26]. Hence, the output of this layer is
\[
O^{(5)} = \left\{ O^{(3)}, \overline{O}^{(3)}, \bar{O}^{(3)}, \overline{\bar{O}}^{(3)} \right\}
\]

Layer 6: Output layer

This layer is used to implement the defuzzification operation. Therefore, the output of this layer is
\[
O^{(6)} = \frac{1}{2} \left( \frac{y_i + \overline{y}_i}{2} + \frac{y_i + \bar{y}_i}{2} \right) - \frac{1}{2} \left( \frac{O^{(5)} + \overline{O}^{(5)}}{2} + \frac{O^{(5)} + \bar{O}^{(5)}}{2} \right)
\]

As above description, the adjustable parameters of IT2TFNS are \( \beta_i, \tilde{\beta}_i, \tilde{\beta}_i, \tilde{p}_i, \tilde{p}_i, \lambda, c, \) and \( s \). Notice that the type-reduction operation is embedded in the Layer 5. Only several equations are needed to ob-
tain the type-reduced result. That is the reason IT2TFNS provides lower computational effort than IT2TFNNs. An example for comparison of computational effort is introduced to illustrate the advance in Section 5.

4. On-line Design of IT2TFNS Adaptive Controller via Stable SPSA Algorithm

Consider an nth-order nonlinear dynamic system

\[ x^{(n)} = F(x) + G(x) \cdot u + d \tag{28} \]

where \( X = [x_1 \ x_2 \ldots x_n]^T = [x \ \dot{x} \ldots x^{(n)}] \) is the state variable which is assumed to be measurable; \( u \) and \( y \) are the control input and output of the nonlinear system, respectively; \( F(.) \) and \( G(.) \) are unknown nonlinear continuous functions; \( d \) is the external disturbance or system uncertainty. For tracking control problem, we can define the tracking error

\[ e = x_r - x \tag{29} \]

where \( x_r = [x_{r1} x_{r2} \ldots x_{rn}]^T = [x_r \ \dot{x}_r \ldots x_r^{(n-1)}] \) are reference trajectories satisfying bounded and differentiable assumptions. Our control objective is to design an on-line adaptive control scheme to generate proper control sequence such that system output can track the reference trajectory. Herein, we adopt the sliding surface concept and define as

\[ S = \Phi \dot{e} = \phi_1 e_1 + \phi_2 e_2 + \cdots + \phi_n e_n. \tag{30} \]

Note that \( \Phi = [\phi_1 \ \phi_2 \ldots 1]^T \) should be chosen properly to satisfy that all roots are located in the left-half plane when \( S = 0 \), then the tracking error will approach to zero with the time when \( S \) approaches to zero. Thus, our objective is transferred to generate the appropriate control sequence to drive the sliding surface approaches to zero. Herein, we propose an adaptive control scheme based on IT2TFNS and SPSA algorithm as shown in Fig. 3. To introduce the control architecture, we first define an error function as

\[ E_k = \frac{1}{2} S_k^2. \tag{31} \]

According to the gradient descent method, the update law of the IT2TFNS parameters is

\[ W_{k+1} = W_k + \Delta W_k = W_k + a_k \left( -\frac{\partial E_k}{\partial W} \right) \tag{32} \]

where \( a_k \) denotes the time-varying learning step length. By the chain rule, the gradient information can be re-written as

\[ \frac{\partial E_k}{\partial W} = S_k \frac{\partial S_k}{\partial W} \tag{33} \]

Herein, we adopt the stable SPSA algorithm to approximate \( \frac{\partial S_k}{\partial W} \) instead of deriving the gradient term of \( S \) with respect to \( W \). Hence,

\[ \frac{\partial S_k}{\partial W} \approx \frac{S_{k+1} - S_k}{c_i} \begin{bmatrix} \Delta_{a_1}^{-1} \ \\ \Delta_{a_2}^{-1} \ \\ \vdots \ \\ \Delta_{a_d}^{-1} \end{bmatrix} \tag{34} \]

where \( S_k(k) \) denotes the sliding surface resulted by perturbed parameters. From (34), we can easily find that there is a limitation for applying SPSA algorithm to on-line control since we should have \( S_{k+1} \) and \( S_k \) to form the approximated gradient. To obtain \( S_{k+1} \) and \( S_k \), we should feed two different control signals into the system at the same time. However, this leads to the invalid on-line control. To overcome this problem, we have

\[ S_k = \phi_1 e_1 + \phi_2 e_2 + \cdots + \{x_r - [x_{r1} + F(x) + G(x) \cdot u_k + d]\} \tag{35} \]

\[ S_{k+1} = \phi_1 e_1 + \phi_2 e_2 + \cdots + \{x_r - [x_{r1} + F(x) + G(x) \cdot u_{k+1} + d]\} \tag{36} \]

where \( u_{k+1} \) is the control input resulted by perturbed parameters. Therefore, we have

\[ S_{k+1} - S_k = G(x) \cdot (u_k - u_{k+1}). \tag{37} \]

Finally, from (32)-(34), and (37), we can obtain parameter update law of IT2TFNS for on-line adaptive control

\[ W_{k+1} = W_k + a_k \frac{G(x)(u_k - u_{k+1})}{c_i} \begin{bmatrix} \Delta_{a_1}^{-1} \ \\ \Delta_{a_2}^{-1} \ \\ \vdots \ \\ \Delta_{a_d}^{-1} \end{bmatrix} \tag{38} \]

From (38), we can observe that only \( u_k \) and \( u_{k+1} \) are needed to update the parameters of IT2TFNS. In other words, we are able to achieve the purpose of on-line control by the proposed method. Figure summaries the above description of the on-line adaptive control approach by IT2TFNS and stable SPSA algorithm.

Subsequently, we develop the stable theorem for selecting appropriate learning step length \( a_k \). We employ the Lyapunov stability approach to have the condition for convergence and the optimal learning step length for IT2TFNS which guarantee the faster convergence.

**Theorem 1**: Let \( a_k \) be the learning step length of tuning parameters for the IT2TFNS controller. Consider the nonlinear control problem using IT2TFNS (shown in Fig. 3), the asymptotic convergence of the closed-loop system is guarantee if the learning step length is chosen satisfying

\[ 0 < a_k < \frac{2}{|g_i|}, \text{ for all } k \tag{39} \]

where \( g_i = \frac{G(x) \cdot (u_k - u_{k+1})}{c_i} \begin{bmatrix} \Delta_{a_1}^{-1} \ \\ \Delta_{a_2}^{-1} \ \\ \vdots \ \\ \Delta_{a_d}^{-1} \end{bmatrix} \) is the estimated gradient using SPSA approach. In addition, the faster convergence can be obtained by the following optimal time-varying learning step length

\[ a'_k = \frac{1}{|g_i|}. \tag{40} \]
**Proof:** In the beginning, we define the discrete-time Lyapunov function as follows

$$V_k = E_k = \frac{1}{2} S_k^2.$$  \hspace{1cm} (41)

The change of the Lyapunov function is

$$\Delta V_k = V_{k+1} - V_k = \frac{1}{2} (S_{k+1}^2 - S_k^2)$$  \hspace{1cm} (42)

According to the Lyapunov stability theorem, if the change of the positive definite Lyapunov function, denoted by $\Delta V_k$, satisfies the condition $\Delta V_k < 0$, $\forall k$, then the asymptotic stability is guaranteed [16, 32]. Hence, our objective is to select the proper learning step length such that $\Delta V_k < 0$, $\forall k$. This implies $V_k$ will converge to zero when $k$ approaches to infinity. From previous results [16], the error difference can be represented as

$$\Delta W = -a_k S_k \frac{\partial S_k}{\partial W} \approx -a_k S_k g_k.$$  \hspace{1cm} (43)

where $g_k$ is the approximated gradient obtained by SPSA approach. Therefore,

$$\Delta V_k = \frac{1}{2} \left[ S_{k+1}^2 - S_k^2 \right]$$

$$= \frac{1}{2} \left[ S_{k+1} S_k + S_k + S_k + S_k \right]$$

$$= \frac{1}{2} \Delta S_k \left[ 2S_k + \Delta S_k \right]$$

$$= \Delta S_k \left( S_k + \frac{1}{2} \Delta S_k \right)$$

$$= \left[ \frac{\partial S_k}{\partial W} \right] a_k g_k \cdot \left( S_k - \frac{1}{2} \left[ \frac{\partial S_k}{\partial W} \right] a_k g_k \right)$$

$$= -a_k S_k \left| g_k \right|^2 \left( 1 - \frac{1}{2} a_k \left| g_k \right|^2 \right).$$  \hspace{1cm} (45)

Let $P_k = a_k \left| g_k \right|^2$. Notice that $P_k > 0$ for all $k > 0$. Thus,

$$\Delta V_k = -e^2 P_k \left( 1 - \frac{1}{2} P_k \right).$$  \hspace{1cm} (46)

The asymptotic stability of the closed-loop system is guaranteed when $\Delta V_k < 0$, $\forall k$. Hence, $(1 - P_k/2)$ should be positive to satisfy $\Delta V_k < 0$, $\forall k$. Therefore, we can obtain the stability condition for $a_k$

$$0 < a_k < \frac{2}{\left| g_k \right|^2}.$$  \hspace{1cm} (47)

The asymptotic stability is guaranteed if $a_k$ is chosen to satisfy (47). Moreover, we can find a condition to guarantee the fast convergence. From (45) and (46), we have

$$S_{k+1}^2 = S_k^2 - e^2 P_k (2 - P_k)$$

$$= S_k^2 \left[ 1 - 2P_k + P_k^2 \right]$$

$$= S_k^2 \left[ P_k - 1 \right]^2.$$  \hspace{1cm} (48)

From (48), the minimum of $S_{k+1}^2$ is achieved when $P_k = 1$. Finally, the time-varying optimal learning step length is

$$a_k^* = \frac{1}{\left| g_k \right|^2}.$$  \hspace{1cm} (49)

This completes the proof. 

As above description, the proposed on-line adaptive control scheme can provide the following advantages:

- The interval type-2 fuzzy neural systems have many adjustable parameters and complicated network structure which make them very hard to obtain the gradient information. We adopt the stable SPSA algorithm to overcome this problem.
- The piece-wise continuous property of triangular-shaped MFs is also solved by stable SPSA algorithm.
The SPSA algorithm inherently has the bottleneck for on-line control. In the proposed control scheme, we utilize the sliding surface to conquer this bottleneck. By using the sliding surface, the input number of controller is reduced. This implies that the computational complexity of the proposed control scheme is low which makes it very practical.

Table 1. Comparison result of IT2TFNS and traditional IT2TFNN.

<table>
<thead>
<tr>
<th>Time (sec.)</th>
<th>Number of rules</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional IT2TFNN</td>
<td>11.954560</td>
<td>19.799836</td>
<td>29.378339</td>
<td>36.624335</td>
<td></td>
</tr>
<tr>
<td>Proposed IT2TFNS</td>
<td>10.774855</td>
<td>11.739968</td>
<td>12.809940</td>
<td>13.598060</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. The used configuration coefficients of Example 2.

<table>
<thead>
<tr>
<th>Configuration coefficients of SPSA algorithm</th>
<th>a</th>
<th>(a)</th>
<th>A</th>
<th>(c)</th>
<th>(\gamma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rules</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network structure</td>
<td>1-4-4-1-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter number</td>
<td>44</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Simulation Results

In this section, we introduce three examples to demonstrate the performance of the proposed approach. Herein, we claim that the proposed IT2TFNS has potential to provide lower computational complexity. Therefore, in the first example, we compare the computational cost between IT2TFNS and traditional IT2TFNN. In the second example, we perform a tracking control of Chua’s chaotic circuit to demonstrate the feasibility and the performance of the proposed method. Finally, we perform the control of water bath temperature control which is a real world application. In addition, the comparison results are also included.

Example 1: Comparison of computational complexity for type-reduction

In order to show the low computational effort of the IT2TFNS, we compare with the traditional IT2TFNN having the same structure as IT2TFNS except that the KM algorithm is adopted for type-reduction. In this example, we feed a random value into each fuzzy neural network for \(10^5\) times and accumulate the cost time. The single-input-single-output case is considered here for convenience. In addition, we compare with different rule number to identify the influence of size of rule base. This simulation is done by MATLAB. The specification of the computer is Intel i5 661 3.33 GHz with 3 GB RAM. Table 1 shows the comparison result. From Table 1, we can observe that the IT2TFNS with uncertainty bounds results in less computational complexity. Especially if the number of fuzzy rules is large, the influence will be huge. Hence, we conclude that the proposed IT2TFNS can provide the lower computational complexity. Besides, the uncertainty bounds method only need several equations to obtain the type-reduced result. This makes IT2TFNS is much easier for implementation.

Example 2: Nonlinear control of Chua’s chaotic circuit

According to [33], the Chua’s chaotic circuit is described as

\[
\begin{align*}
\dot{x} &= \frac{14}{1805} x - \frac{168}{9025} x_2 + \frac{1}{38} \ddot{x} \\
&\quad - \frac{2}{45} \left( \frac{28}{361} x + \frac{7}{95} \ddot{x} + \dot{x} \right) + u + d \\
y &= x
\end{align*}
\]

where \(d\) is the external disturbance assumed to be a square-wave with amplitude \(\pm 0.1\) and period \(2\pi\). Define \(x = [x_1 \ x_2 \ x_3]^T = [x \ \dot{x} \ \ddot{x}]^T\). Herein, the objective is to track the reference trajectory \(y_r = 1.5 \sin(t)\). We choose the initial state value as \(x(0) = [0 \ 0 \ 1]^T\) and sampling time as 0.01 second. With the control scheme shown in Fig. 3, we use IT2TFNS to generate the proper control sequence and track the reference trajectory. The used configuration coefficients of this example are listed in Table 2. After 20 seconds, the state...
trajectories $x_1$, $x_2$, and $x_3$ are shown in Fig. 4(a) (reference trajectory $x_1$: black-dashed; IT2TFNS: red-solid; IT2TFNN: blue-solid; TSK-FNN with 4 rules: Red-dashed; TSK-FNN with 8 rules: red-dotted). From Fig. 4(a), we find that the actual system states successfully track the reference trajectory. The IT2TFNN results similar results with more computational effort (shown in Example 1). In other words, the proposed control scheme is valid. The control effort generated by the IT2TFNS controller is depicted in Fig. 4(b). The final fuzzy rules are

\[ R^1: \text{IF } S \text{ is } \tilde{F}_{11}, \]
\[ \text{THEN } Y_1 = [2.3107 5.3899] + [2.1725 5.1376]x_i \]
\[ R^2: \text{IF } S \text{ is } \tilde{F}_{12}, \]
\[ \text{THEN } Y_1 = [-4.5655 1.2902] + [-0.1650 1.5030]x_i \]
\[ R^3: \text{IF } S \text{ is } \tilde{F}_{13}, \]
\[ \text{THEN } Y_1 = [-1.1003 2.2501] + [-0.3660 6.7873]x_i \]
\[ R^4: \text{IF } S \text{ is } \tilde{F}_{14}, \]
\[ \text{THEN } Y_1 = [-5.3135 0.4121] + [1.6657 6.2384]x_i \]

The corresponding MFs are depicted in Fig. 5.

![Final constructed MFs of Example 2](image)

Figure 5. Final constructed MFs of Example 2: (a) MF of $x_1$ rule 1, (b) MF of $x_1$ rule 2, (c) MF of $x_1$ rule 3, and (d) MF of $x_1$ rule 4.

Subsequently, in order to demonstrate the performance of the IT2TFNS controller, we compare with other fuzzy neural networks including type-1 TSK fuzzy neural network (TSK-FNN) [14] and traditional IT2TFNN. In addition, we also analyze the affection of different rule number using IT2TFNS. Herein, we adopt mean square error (MSE) for the performance index. For each case, we have 10 independent runs for statistical analysis. The comparison results are shown below.

**Discussion for different fuzzy neural systems:** Table 3 shows the comparison results including worse, average, best MSE and average computational cost. From Table 3, the worst, average, and the best MSE of traditional IT2TFNN are 0.1180, 0.0555, and 0.0116, respectively. The average computational time is 1.8095 (ms). The worst, average, and the best MSE of proposed IT2TFNS are 0.1243, 0.0564, and 0.0132, respectively. The average computational time is 1.5079 (ms). There are two cases in the result of TSK-FNN which are 4 rules and 8 rules. The case of 4 rules is same as the number of rules in the results of traditional IT2TFNN and IT2TFNS. Next, we increase the number of rules to 8 in order to reach the same number of parameter as traditional IT2TFNN and IT2TFNS. In the case of same number of rule (number of rule is 4), the worst, average, and the best MSE of TSK-FNN are 0.2308, 0.1501, and 0.0902, respectively. The average computational time is 0.9239 (ms). Although, the computational time of TSK-FNN is only 0.9239 (ms), we can easily observe that the traditional IT2TFNN and IT2TFNS perform much better than TSK-FNN. On the other hand, in the case of same number of parameter (number of parameter is 44), the worst, average, and the best MSE of TSK-FNN are 0.1111, 0.0844, and 0.0516, respectively. The average computational time is 1.1528 (ms). The traditional IT2TFNN and IT2TFNS still perform better than TSK-FNN. As above description, we conclude that the both interval type-2 fuzzy neural networks can provide better performance than type-1 fuzzy neural network.

**Discussion for different type-reduction methods:** In this discussion, we focus on the results of the traditional IT2TFNN and the IT2TFNS. We can find they have similar performance. However, the performance of IT2TFNS is slightly worse. This phenomenon is normal. In the traditional IT2TFNN, the KM algorithm is adopted for type-reduction (finding the left-end point and right-end point of centroid). As the result of Example 1, we know that the KM algorithm has higher computational dimension. Therefore, in the proposed IT2TFNS, we want to avoid using the KM algorithm for type-reduction. The uncertainty bounds method is an alternative way for type-reduction. The main idea of the uncertainty bounds method is to find the upper bound and lower bound of the two end points. In the sequel, we use these four bounds (called uncertainty bounds) to approximate the end points. In the procedure of finding these four bounds, only several direct equations are needed. Hence, the computational cost is lower. Nevertheless, the uncertainty bounds method is based on the approximation of the end-points. It is the reason why the performance will be slightly worse than traditional IT2TFNN. To sum up, our purpose is to sacrifices only very slight performance which might be negligible to provide lower computational cost. This makes our IT2TFNS controller more efficient and higher degree of realization.
Table 3. Comparison results of different fuzzy neural systems for Example 2.

<table>
<thead>
<tr>
<th></th>
<th>Number of rules</th>
<th>Number of parameter</th>
<th>Worse MSE</th>
<th>Average MSE</th>
<th>Best MSE</th>
<th>Average Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK-FNN</td>
<td>4</td>
<td>20</td>
<td>0.2308</td>
<td>0.1501</td>
<td>0.0902</td>
<td>923.9</td>
</tr>
<tr>
<td>Traditional IT2TFNN</td>
<td>8</td>
<td>44</td>
<td>0.1111</td>
<td>0.0844</td>
<td>0.0516</td>
<td>1152.8</td>
</tr>
<tr>
<td>Proposed IT2TFNS</td>
<td>4</td>
<td>44</td>
<td>0.1180</td>
<td>0.0555</td>
<td>0.0116</td>
<td>1809.5</td>
</tr>
</tbody>
</table>

Table 4. Comparison result of different rule numbers using IT2TFNS for Example 2.

<table>
<thead>
<tr>
<th>Number of rule</th>
<th>Number of parameter</th>
<th>Worse MSE</th>
<th>Average MSE</th>
<th>Best MSE</th>
<th>Average time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>22</td>
<td>0.2645</td>
<td>0.1058</td>
<td>0.0221</td>
<td>1494.1</td>
</tr>
<tr>
<td>4</td>
<td>44</td>
<td>0.1243</td>
<td>0.0564</td>
<td>0.0132</td>
<td>1507.9</td>
</tr>
<tr>
<td>6</td>
<td>66</td>
<td>0.1967</td>
<td>0.1034</td>
<td>0.0334</td>
<td>1565.8</td>
</tr>
<tr>
<td>8</td>
<td>88</td>
<td>0.4204</td>
<td>0.1484</td>
<td>0.0393</td>
<td>1651.2</td>
</tr>
</tbody>
</table>

Discussion for the affection of different rule number using IT2TFNS: In Table 4, we have four groups of result including 2 rules (decrease the number of rule), 4 rules (original result), 6 rules, and 8 rules (increase the number of rule). From Table 4, we can easily find that the performance became worse if we increase or decrease the number of rule. With only 2 rules, the results show that the number of fuzzy rule is not enough to achieve better performance. With 6 rules, the performance became worse but better than result of 2 rules. If we increase the number of rule to 8 fuzzy rules, we find that the performance become worse. We can observe that the more number of rules enhances the performance; however, too many fuzzy rules results in worse performance. Therefore, the number of rule should be chosen properly. We recommend that the number of rule should be chosen as 4 to obtain the best performance.

Example 3: Water bath temperature control

In the Example 3, we present a more practical control problem. The control of the water bath temperature system is given by

\[
y_{k+1} = e^{-\alpha t}y_k + \frac{\beta}{1 + e^{(15y_k - 40)}}u_k + [1 - e^{-\alpha t}]y_0
\]  

(51)

The system parameters used here are \( \alpha = 1.00151 \times 10^{-4} \), \( \beta = 8.67973 \times 10^{-3} \), and \( Y_0 = 25 \) (°C), which were obtained from a real world water bath manufacturer. The plant input \( u_k \) is limited between 0V and 5V and the sampling period is \( T_s = 30s \). The objective is to use the IT2TFNS controller with the control scheme shown in Fig. 3 to generate the proper control input to the water bath system and follow the reference temperature \( y_r \):

\[
y_r = \begin{cases} 
35' \text{C} & \text{for } k \leq 40, \\
55' \text{C} & \text{for } 40 < k \leq 80, \\
75' \text{C} & \text{for } 80 < k \leq 120.
\end{cases}
\]  

(52)

The used configuration coefficients of this example are the same as Example 2 (listed in Table 2). In this simulation, we choose the number of rule to 4 due to the previous result. The results are shown in Fig. 6. Figure 6(a) shows the actual system temperature (solid line) and reference temperature (dashed line). We can observe that the actual system temperature follows the reference temperature, i.e. the IT2TFNS controller successfully controls the water bath temperature system. The control input made by IT2TFNS controller is depicted in Fig. 6(b). The final fuzzy rules are

\[
R_{11}^i : \text{IF } S \text{ is } E_{11}^i, \\
\text{THEN } Y_i = [-3.8714\ 1.9876] + [0.6836\ 2.9295]x_i
\]
The corresponding MFs are depicted in Fig. 7.

In this example, we also compare with different fuzzy neural systems to demonstrate the performance. As shown in Fig. 6(a), there are a number of error during the temperature raises to each setting point. Therefore, in order to clarify the error resulted by different control performance, we adopt sum of absolute error (SAE) for the performance index. The comparison results are shown below.

**Discussion for different fuzzy neural systems:** The comparison results including worse, average, and best SAE are shown in Table 5. From Table 5, the worst, average, and the best SAE of the TSK-FNN with 4 rules are 1017.3, 596.3, and 456.4, respectively. The average computational time is 0.0445 (ms). This reveals some invalid controls happened in this case. As for the FNN with 8 rules, they are 624.9, 530.1, and 451.5, respectively. The average computational time is 0.0587 (ms). The worst, average, and the best SAE of traditional IT2TFNN are 414.7, 399.8, and 393.8, respectively while the proposed IT2TFNS are 411.1, 401.6, and 395.3, respectively. The average computational times are 0.1061 and 0.0884 (ms). In this example, we again demonstrate that the interval type-2 fuzzy neural network can provide better performance than type-1 one.

**Discussion for different type-reduction methods:** Focusing on the results of the traditional IT2TFNN and the IT2TFNS, we can observe the result similar to Example 2. Although the performance of IT2TFNS is slightly worse, the IT2TFNS has lower computational complexity. We again demonstrate that the IT2TFNS controller is more efficient than traditional IT2TFNN.

<table>
<thead>
<tr>
<th>Number of rules</th>
<th>Number of Parameter</th>
<th>Worse SAE</th>
<th>Average SAE</th>
<th>Best SAE</th>
<th>Average Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSK-FNN</td>
<td>4</td>
<td>20</td>
<td>1017.3</td>
<td>596.3</td>
<td>456.4</td>
</tr>
<tr>
<td>Traditional IT2TFNN</td>
<td>8</td>
<td>44</td>
<td>624.9</td>
<td>530.1</td>
<td>451.5</td>
</tr>
<tr>
<td>Proposed IT2TFNS</td>
<td>4</td>
<td>44</td>
<td>411.1</td>
<td>399.8</td>
<td>393.8</td>
</tr>
</tbody>
</table>

6. Conclusion

In this paper, we have proposed a novel interval type-2 Takagi-Sugeno-Kang fuzzy neural system (IT2TFNS) and its stable SPSA algorithm-based on-line adaptive control scheme for nonlinear systems control. In the most of the interval type-2 fuzzy neural systems, the KM algorithm is adopted to process type-reduction. However, the KM algorithm is an iterative algorithm which has the bottleneck of computational complexity and is not easy for implementation, especially if the number of fuzzy rule is large. Therefore, in the IT2TFNS, we utilize the uncertainty bounds to process type-reduction. Instead of iterative searching the switching values, only several equation computations are needed for type-reduction. On the other hands, we proposed a stable SPSA algorithm-based on-line adaptive control scheme which copes with the difficulty that the SPSA algorithm inherently has a bottleneck of on-line control. The stable SPSA algorithm also solved the problems that the gradient information of interval type-2 fuzzy neural system is difficult to obtain and the piece-wise continuous property of triangular-shaped MFs. In addition, the corresponding stable learning was derived by Lyapunov theorem which guarantees the convergence and stability of the closed-loop systems. The simulations including three examples were done and the results shown that the proposed approach is not only valid but also efficient.

**Acknowledgment**

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

References


Ching-Hung Lee is currently an Associate Professor of the Department of Mechanical Engineering at National Chung Hsing University. Dr. Lee received the Wu Ta-Yu Medal and Young Researcher Award in 2008 from the National Science Council, R.O.C. He also received the 2009 Youth Automatic Control Engineering Award from Chinese Automatic Control Society. His main research interests are fuzzy neural systems, fuzzy logic control, neural network, signal processing, nonlinear control systems, image processing, and robotics control.

Feng-Yu Chang was born in Taiwan, R.O.C., in 1987. He received the M.S. degree in Electrical Engineering in 2011 and is currently pursuing a Ph.D. degree in Electrical Engineering both in Yuan Ze University, Taiwan. His current research interests include intelligent control, adaptive control, fuzzy logic systems, and neural network applications.

Chih-Min Lin is currently a Chair Professor and the Dean of the College of Electrical and Communication Engineering, Yuan Ze University, Taiwan. He is also an Honorary Professor of Obuda University, Hungary. His research interests include fuzzy system, neural network, cerebellar model articulation controller, automatic control system, and robotics. He has published 123 journal papers and 155 conference papers. He is an Associate Editor of IEEE Transaction on Systems, Man, and Cybernetics, Part B, Asian Journal of Control, International Journal of Fuzzy Systems, and International Journal of Machine Learning and Cybernetics. In addition, he is an IEEE Fellow and IET Fellow; he also serves as a Member of Board of Governors of IEEE Systems, Man, and Cybernetics Society.