Neurocomputing 275 (2018) 2239–2250

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Self-evolving function-link interval type-2 fuzzy neural network for nonlinear system identification and control

Chih-Min Lin[a,*], Tien-Loc Le[a,b], Tuan-Tu Huynh[a,b]

[a] Yuan Ze University, Taoyuan, Taiwan
[b] Department of Electrical Electronic and Mechanical Engineering, Lac Hong University, Bien Hoa, Viet Nam

ARTICLE INFO

Article history:
Received 28 May 2017
Revised 1 August 2017
Accepted 1 November 2017
Available online 9 November 2017

Communicated by Prof. Qi Zhou

Keywords:
System identification
Interval type-2 fuzzy system
Neural network
Self-evolving algorithm

ABSTRACT

Determining a network size for a fuzzy neural network structure is very important, and it is often difficult to obtain the most suitable value. This study develops a self-evolving function-link interval type-2 fuzzy neural network (SEFT2FNN) that autonomously constructs the rule base with the initial empty and the membership functions. The function-link is applied to an interval type-2 fuzzy neural network to give a more accurate approximation of the function. The adaptive laws for the proposed system are derived using the steepest descent gradient approach. The stability of system was guaranteed using Lyapunov function approach. Finally, the performance of the proposed system is verified using the numerical simulations of the nonlinear system identification and the control of time-varying plants.

1. Introduction

In recent years, many studies have combined the fuzzy inference system and neural networks to produce fuzzy neural networks (FNNs) [1–6]. The fuzzy set theory was first introduced by Zadeh in 1965 [7]. It used linguistic variables and membership functions to describe the degrees to which elements belong in fuzzy sets. These system are categorized as type-1 fuzzy logic systems (T1FLS). Since then, flexible features, intuitive knowledge, and easy computation have meant that the T1FLS has been widely applied to a variety of fields [8]. Because T1FLS are precise sets, the process cannot deal with the uncertainty that arises from internal and external disturbances [9]. Therefore, in 1975 Zadeh developed the concept of type-2 fuzzy logic systems (T2FLS), which takes account of the uncertainty in membership functions [10]. Previous studies [11–13] showed that under the same condition, a T2FLS copes better with the uncertainties than a T1FLS. In order to reduce the computational cost, interval type-2 fuzzy logic systems (IT2FLS) were proposed by Liang and Mendel [14]. Since then, the IT2FLS has been the subject of study in various fields, such as control problems, system identification, prediction and classification [15–24]. In 2009, Castro et al. presented a hybrid learning algorithm for a class of interval type-2 fuzzy neural network [20]. In 2009, Hagras and Wagner proposed the interval type-2 fuzzy logic controllers – towards better uncertainty handling in real world applications [21]. In 2013, Castillo et al. introduced the universal approximation of a class of interval type-2 fuzzy neural networks in nonlinear identification [22]. In 2014, Castillo et al. provided the application of interval type-2 fuzzy neural networks in nonlinear identification and time series prediction [23]. Following that, in 2015, Gaxiola et al. presented the optimization of type-2 fuzzy weights in backpropagation learning for neural networks using GAs and PSO [24]. However, because of the fixed structure, it is difficult to determine suitable fuzzy rules of the system. Many studies have proposed a self-organizing algorithm that autonomously determines the network size for a fuzzy neural network [25–28]. In 2008, Juang and Tsao proposed a self-organizing type-2 neural fuzzy system and applied for the non-linear system identification and a truck braking control problem [25]. In 2009, Lin and Chen proposed a self-organizing cerebellar model articulation controller for multi-output multi-input uncertainty nonlinear systems [27]. In 2017, Lin and Le proposed the PSO-self-organizing interval type-2 fuzzy neural network for antilock braking systems [28]. However, the disadvantage is that the initial design of the network significantly affects the control performance, and it requires knowledge about the system to design the initial rules and membership functions [29,30]. A self-evolving algorithm was proposed that autonomously constructs the IT2FNN using the empty rule and membership functions. In 2014, Lin et al. presented a self-evolving compensatory interval type-2 fuzzy neural network for systems

* Corresponding author.
E-mail addresses: cml@saturn.yzu.edu.tw (C.-M. Lin), tienloc@lhu.edu.vn (T.-L. Le), s1038507@mail.yzu.edu.tw, huynhtuantu@gmail.com (T.-T. Huynh).

https://doi.org/10.1016/j.neucom.2017.11.009
0925-2312/

identification and prediction problem [31]. In 2016, Mohammadzadeh et al. proposed a new self-evolving non-singleton type-2 fuzzy neural network to control fractional-order chaotic systems [32]. However, compared with function-link IT2FNN, this structure does not have a sufficient number of adjustable free variables and its performance could be improved.

A functional-link network (FLN) was developed by Pao in 1989, which can enrich the input information by function expansion [33]. It has been successfully applied in many fields, such as prediction, classification, system identification, pattern recognition and control problems [34–39]. Many studies have used the FLN to improve the performance of control systems. The input representation is enhanced using trigonometric functions that are expanded in an extended space [34]. The output from the FLN is generated by a nonlinear combination of the input variables, so approximation of the nonlinear function is more accurate [39].

The main motivation of this study is to construct a neural network controller which does not need to design the initial structure of network in advance, and the on-line learning will self-construct the network to the suitable structure. In this study, a self-evolving algorithm and a FLN are applied to an IT2FNN to enhance the performance of the network. This proposed network structure is referred to as a self-evolving function-link IT2FNN (SEFT2FNN). The main contributions of this study are the development of a SEFT2FNN that incorporates adaptive laws for updating parameters and the design of a self-evolving algorithm that autonomously constructs a network from an empty structure and the empty rule. The convergency of the proposed algorithm is proven by Lyapunov function analysis approach. The design of the function-link network for IT2FNN helps the updating parameter have more adjustable free variables, which improves the accuracy of the system. Comparison with the previous type-2 fuzzy neural networks in [20–24], the proposed SEFT2FNN has some advantages such as it does not request to set the initial structure in advance, the FLN gives more freedom for adjusting the parameters of IT2FNN, and more simple computation. Finally, the numerical simulations of system identifications and the control of time-varying plants are conducted to show the advantages of the proposed method than other methods.

The remainder of the paper is organized as follows. Section 2 presents the structure of the function-link IT2FNN. Section 3 details the self-evolving algorithm and parameter learning for the SEFT2FNN. Section 4 shows the simulation results for non-linear system identifications and the control of time-varying plants. Finally, the conclusions are discussed in Section 5.

2. Function-link IT2FNN control system

Fig. 1 shows the structure of the proposed SEFT2FNN network system with a self-generated structure. The structure of function-link is shown in Section 2.1. The detail steps for constructing the function-link interval type-2 fuzzy neural network is provided in Section 2.2. The parameters are adjusted using the adaptive laws that are designed in Section 3.

2.1. Function-link network

In many studies of fuzzy inference systems, the weights that are used in the consequent part of fuzzy rules are often given by singleton values and are updated using adaptive laws. In this study, the fuzzy weights are determined using the FLN and updating indirectly uses the update laws for the weights of the FLN, so there are more design parameters, which allows a better approximation. Fig. 2 shows the structure of the FLN. This study uses a trigonometric function to expand the function expansion because it is more compact and the orthogonal basis functions (sine and cosine) can be computed more quickly [36]. In the FLN model, an input vector, $\boldsymbol{I} = [i_1,\ i_2,\ \ldots, i_n]^T$, is enhanced as $\boldsymbol{\phi} = [\phi_1,\ \phi_2, \ldots, \phi_m]^T$, where $\boldsymbol{I}$ and $\boldsymbol{\phi}$ are the input and output vector of function expansion block, respectively. m is the number of elements in the function expansion output and n is the number of elements in vector input. For instance, if $\boldsymbol{I} = [i_1,\ i_1, i_3]^T$ then $\boldsymbol{\phi} = [i_1, \sin(\pi i_1), \cos(\pi i_1), i_2, \sin(\pi i_2), \cos(\pi i_2), i_3, \sin(\pi i_3), \cos(\pi i_3), i_1 i_2, i_1 i_3, i_2 i_3, i_1 i_2 i_3]^T$. The jth output of FLN can be expressed as

$$w_j = \ q_{1j}\phi_1 + q_{2j}\phi_2 \ldots + q_{mj}\phi_m = \sum_{i=1}^{m} q_{ij}\phi_i = \boldsymbol{q}_j^T\boldsymbol{\phi} \tag{1}$$

where $\boldsymbol{q}_j = [q_{1j}, q_{2j} \ldots, q_{mj}]^T$, $\phi = [\phi_1, \phi_1 \ldots, \phi_m]^T$, and q_{ij} are the weights for the FLN that connect w_j and ϕ_i. Initially, q_{ij} has an initial value and this is updated using the adaptive law that is presented in the following sections.

2.2. The function-link interval type-2 fuzzy neural network

In the IT2FNN, the relationship between the consequent and the antecedent is explained using an IF-THEN rule. The j^{th} rule has the following form:

IF x_1 is $\tilde{X}_{1j}$ and ... and x_i is $\tilde{X}_{ij}$ and ... and x_n is $\tilde{X}_{nj}$

$$\text{THEN } o_j = \ \tilde{W}_j \tag{2}$$

where $\tilde{X}_{ij}$ is the type-2 fuzzy membership function for the jth rule of the ith input ($j = 1, \ldots, M$ *and* $i = 1, \ldots, n$) and $\tilde{W}_j$ is the type-2 fuzzy membership function for the jth output. All parameters in the consequent and antecedent parts are updated using the adaptive laws that are detailed in Section 3.2.

The structure of the interval type-2 fuzzy neural network is shown in Fig. 1, It has six layers: an input layer, a membership function layer, a firing layer, a weight memory layer, a pre-output layer and a final output layer. They are described below.

1) The input layer: there is no computation in this layer. All of the input variables from this layer are directly transferred to the next layer.
2) The membership function layer: In this layer, the membership grades are determined using the input variables, x_j, and a type-2 fuzzy membership input function, $\tilde{X}_{ij}$. In this study, $\tilde{X}_{ij}$ is defined as a type-2 Gaussian membership function (T2GMF) that has one mean and an uncertain standard deviation, $\sigma \in [\sigma_1, \sigma_2]$ (see Fig. 3). Using the T2GMF, the membership grades, $\mu_{\tilde{X}_{ij}}$, are described by the upper and lower membership functions (UMF and LMF)

$$\bar{\mu}_{ij} = \exp\left\{-\frac{1}{2}\left(\frac{x_i - m_{ij}}{\bar{\sigma}_{ij}}\right)^2\right\} \tag{3}$$

$$\underline{\mu}_{ij} = \exp\left\{-\frac{1}{2}\left(\frac{x_i - m_{ij}}{\underline{\sigma}_{ij}}\right)^2\right\} \tag{4}$$

3) The firing layer: In this layer, each node is produced using a t-norm operator. The firing strength of the ith rule is an interval value $F^i = [\underline{f}^i,\ \bar{f}^i]$, where $\underline{f}^i$ and $\bar{f}^i$are given by:

$$\bar{f}^i = \prod_{j=1}^{n} \bar{\mu}_{ij} \tag{5}$$

$$\underline{f}^i = \prod_{j=1}^{n} \underline{\mu}_{ij} \tag{6}$$

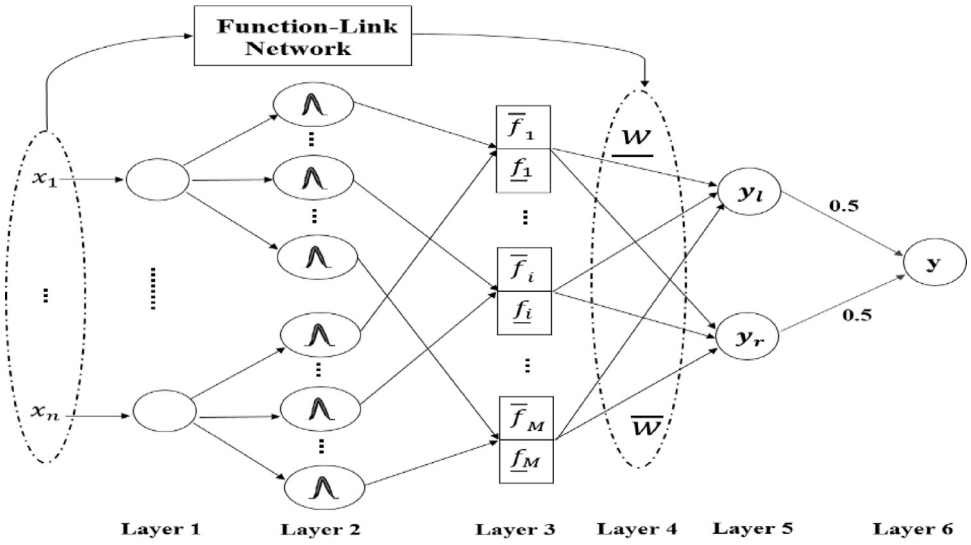

Fig. 1. Structure of the SEFT2FNN.

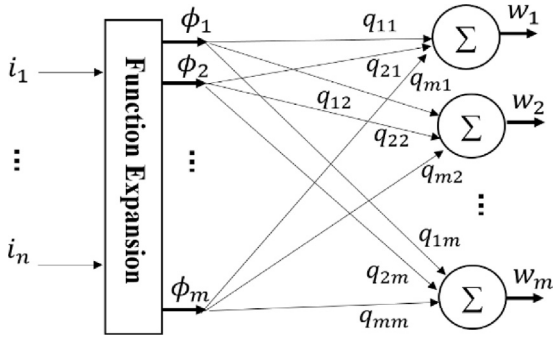

Fig. 2. Structure of the function-link network.

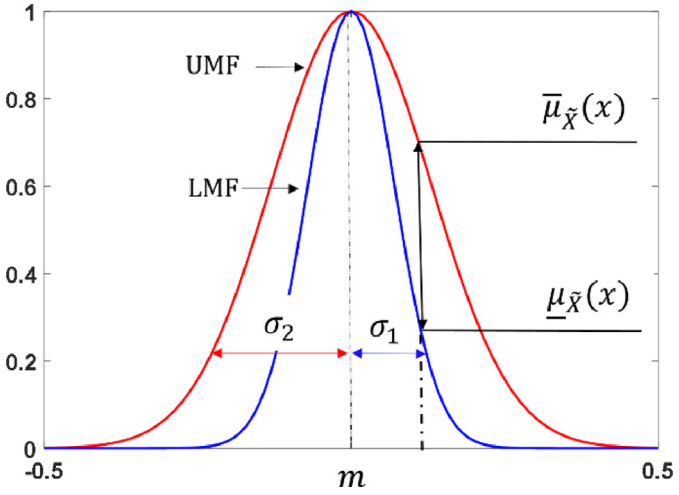

Fig. 3. Interval type-2 Gaussian membership function.

4) The output weight layer: Each node in this layer is the weight of IT2FNN and is determined by the output of the FLN. From Fig. 3 and (1), the output weight, $\boldsymbol{w}$, is derived as:

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} = \begin{bmatrix} q_{11} & \cdots & q_{m1} \\ \vdots & \ddots & \vdots \\ q_{1k} & \cdots & q_{mk} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_m \end{bmatrix} = \mathbf{Q}^T \boldsymbol{\phi} \tag{7}$$

The consequent part of the function-link IT2FNN uses a type-2 fuzzy set to determine the upper bound and the lower bound of $\mathbf{Q}$ as $\bar{\mathbf{Q}}$ and $\underline{\mathbf{Q}}$, respectively. The interval value for the vector output weight $[\bar{\boldsymbol{w}},\ \underline{\boldsymbol{w}}]$ is then derived. For the ith rule, the output weight is an interval value, $w^i = [\underline{w}^i,\ \bar{w}^i]$.

5) The pre-output layer: This layer calculates the left and right output limits by combining the output of layer 3, firing strength $[\underline{f}^i,\ \bar{f}^i]$, and layer 4, output weight $[\underline{w}^i,\ \bar{w}^i]$. The Karnik–Mendel (KM) algorithm is also used to adjust the contribution of the upper and lower values in the firing strength [40]. Therefore, the output $[y_l, y_r]$ is given by:

$$y_l = \frac{\sum_{i=1}^{M} f_l^i \underline{w}^i}{\sum_{i=1}^{M} f_l^i} \tag{8}$$

$$y_r = \frac{\sum_{i=1}^{M} f_r^i \bar{w}^i}{\sum_{i=1}^{M} f_r^i} \tag{9}$$

where the firing strengths, f_l^i and f_r^i, are chosen as

$$f_l^i = \begin{cases} \bar{f}^i, & i \le L \\ \underline{f}^i, & i > L \end{cases} \tag{10}$$

$$f_r^i = \begin{cases} \underline{f}^i, & i \le R \\ \bar{f}^i, & i > R \end{cases} \tag{11}$$

where L and R respectively represent the left and right switch points, which are determined using the KM algorithm. The detail of KM algorithm is shown in Appendices A and B.

6) Output layer: This layer is the final output of the function-link IT2FNN (FT2FNN). The output of the previous layer is an interval set $[y_l, y_r]$. The average operation is used for defuzzification:

$$y = \frac{y_l + y_r}{2} \tag{12}$$

The FT2FNN requires suitable rules, which significantly affect the estimation accuracy of the network. A large number of rules result in a very high computational burden, which is unsuitable for real-time applications. A small number of rules may be inadequate to achieve the desired performance. To overcome this problem, this study uses a self-evolving algorithm to autonomously construct the structure of the FT2FNN, which is presented in the following sections.

3. Self-evolving algorithm and parameter learning

3.1. Self-evolving for function-link interval type-2 fuzzy neural network

In self-evolving algorithm, the architecture of the FT2FNN is automatically generated from an empty structure. In the initial run, the self-evolving algorithm uses the input information to generate the first Gaussian function (GF) for the input membership function and for the weight for the defuzzification operation. When the input changes, the algorithm then determines whether to generate new rules or to delete inappropriate rules or when to update the existing rules (the GFs and the weights).

New rules are generated by comparing the prior threshold and the maximum contribution of GFs to the rule, which is represented by the membership grade. The condition for generating a new rule is: If $(G_I < T_g)$, the new input data is far from the existing membership function, so new rule is generated. T_g is the prior threshold for generating rule and G_I is the maximum membership grade of the Ith input, which is given by

$$G_I = max[\mu_{I1},\ \mu_{I2}, \ldots, \mu_{Ik}] \tag{13}$$

where the interval membership grade is given by:

$$\mu_{Ij} = \frac{1}{2}\left(\bar{\mu}_{Ij} + \underline{\mu}_{Ij}\right) \tag{14}$$

The initial values for the mean and the variance of the new rule are defined as:

$$m_{ij}^{M(k)+1} = \ i_i(k) \tag{15}$$

$$\left[\bar{\sigma}_{ij}^{M(k)+1}, \underline{\sigma}_{ij}^{M(k)+1}\right] = [\sigma_{init} + \Delta\sigma, \quad \sigma_{init} - \Delta\sigma\ ] \tag{16}$$

The vector $\boldsymbol{q}_{new}^{M(k)+1} = [\bar{\boldsymbol{q}}_{new}^{M(k)+1}, \underline{\boldsymbol{q}}_{new}^{M(k)+1}]$ is the weight of FLN and is defined as

$$\boldsymbol{q}_{new}^{M(k)+1} = [q_{init}, q_{init}, \ldots, q_{init}]^T \in \Re^m \tag{17}$$

where $M(k)$ is the total number of rules at the kth step, $\Delta\sigma$ is half of the uncertain variance, σ_{init} and q_{init} are the initial values for the variance and the function-link weight, respectively, m is the number of elements in the function expansion output.

The values for $\Delta\sigma$ and σ_{init} significantly affect the network system. Fig. 3 shows if the values of $\Delta\sigma$ is extremely small, the uncertainty in the GFs is very small and it becomes a type-1 fuzzy set. If the values of $\Delta\sigma$ or σ_{init} are extremely large, the GFs cover all input domains and a smaller number of rules is generated [25]. For extremely small values of σ_{init}, the degree of over-lapping between GFs is small, so a very large number of rules is generated.

The process of deleting existing rules also takes account of the contribution of GFs, which is determined using the minimum membership grades, D_I, and is given by:

$$D_I = arg\ min[\mu_{I1},\ \mu_{I2}, \ldots, \mu_{Ik}] \tag{18}$$

If $(D_I < T_d)$ the minimum membership grade, D_I, of the Ith input is smaller than the prior threshold, T_d, for the deletion of a rule, so the Ith rule is deleted.

Using this automatic generation and pruning method, the proposed SEFT2FNN can determines the optimum number of rules. Fig. 4 shows the flowchart of the structure and parameter learning for the proposed method. The online learning for updating parameters is presented in the following sections.

3.2. The SEFT2FNN parameter learning algorithm

For system identification, the input data is used to train the network and the final output from the SEFT2FNN is also the output of the system. An estimation system identification, $\hat{y}_{\text{SEFT2FNN}}(\underline{\hat{\boldsymbol{q}}}, \bar{\hat{\boldsymbol{q}}}, \hat{m}_{ij}, \underline{\hat{\sigma}}_{ij}, \bar{\hat{\sigma}}_{ij})$, allows an online estimate of the desired output, y_d.

The tracking error vector for the system is defined as $\boldsymbol{e}(k) = [e(k),\ \dot{e}(k),\ \ldots, e^{(n-1)}(k)]^T \in \Re^n$ where $e(k)$ is defined as

$$e(k) = y_d(k) - \hat{y}_{SEFT2FNN}(k) \in \Re \tag{19}$$

where $y_d(k)$ is the desired output for the system and $\hat{y}_{SEFT2FNN}(k)$ is the output of the SEIT2FNN.

To ensure accurate identification, this study uses a high-order sliding mode from [41–43]

$$\begin{aligned} s(k) &= \sum_{l=0}^{l-1} \frac{(n-1)!}{l!(n-l-1)!}\left(\frac{\partial}{\partial k}\right)^{n-l-1} \lambda^l e \\ &= e^{(n-1)} + (n-1)\lambda e^{(n-2)} + (n-2)\lambda^2 e^{(n-3)} \ldots + \lambda^{n-1} e \end{aligned} \tag{20}$$

where λ is a positive constant, that defines the slope of the sliding surface.

Taking the derivative of (20)

$$\begin{aligned} \dot{s}(k) &= e^{(n)} + (n-1)\lambda e^{(n-1)} + (n-2)\lambda^2 e^{(n-2)} \ldots + \lambda^{n-1} e^{(1)} \\ &= \ e^{(n)} + \boldsymbol{K}^T \boldsymbol{e} \end{aligned} \tag{21}$$

where $\boldsymbol{K} = [(n-1)\lambda,\ (n-2)\lambda^2, \ldots. \lambda^{n-1}]^T \in \Re^{n-1}$ is the positive gain vector.

If the values for n and λ are selected to correspond to the coefficients of a Hurwitz polynomial, then $\lim_{k\to\infty} e(k) = 0$. The Lyapunov cost function is chosen as $V_1(s(k)) = \frac{1}{2}s^2(k)$, so $\dot{V}_1(s(k)) = s(k)\dot{s}(k)$. Using (19) and (21), yield

$$\dot{V}_1(s(k)) = s(k)[y_d^n - \hat{y}_{SEFT2FNN}^n + \boldsymbol{K}^T \boldsymbol{e}] \tag{22}$$

An online learning gradient descent algorithm is applied to minimize $\dot{V}_1(s(k))$. Therefore, the online tuning laws for the parameters of a type-2 fuzzy system are given by the following equations:

$$\underline{\hat{\boldsymbol{q}}}^i(k+1) = \underline{\hat{\boldsymbol{q}}}^i(k) - \hat{\eta}_q \frac{\partial s(k)\dot{s}(k)}{\partial \underline{\hat{\boldsymbol{q}}}^i} \tag{23}$$

$$\bar{\hat{\boldsymbol{q}}}^i(k+1) = \bar{\hat{\boldsymbol{q}}}^i(k) - \hat{\eta}_q \frac{\partial s(k)\dot{s}(k)}{\partial \bar{\hat{\boldsymbol{q}}}^i} \tag{24}$$

$$\hat{m}_{ij}(k+1) = \hat{m}_j^i(k) - \hat{\eta}_m \frac{\partial s(k)\dot{s}(k)}{\partial \hat{m}_{ij}} \tag{25}$$

$$\bar{\hat{\sigma}}_{ij}(k+1) = \bar{\hat{\sigma}}_{ij}(k) - \hat{\eta}_\sigma \frac{\partial s(k)\dot{s}(k)}{\partial \bar{\hat{\sigma}}_{ij}} \tag{26}$$

$$\underline{\hat{\sigma}}_{ij}(k+1) = \underline{\hat{\sigma}}_{ij}(k) - \hat{\eta}_\sigma \frac{\partial s(k)\dot{s}(k)}{\partial \underline{\hat{\sigma}}_{ij}} \tag{27}$$

where $\hat{\eta}_q,\ \hat{\eta}_m,\ \hat{\eta}_\sigma$ are the learning-rates for updating the function-link weights, the means and the variances, respectively. Applying the chain rule for the derivation term in (23)–(27), gives:

$$\frac{\partial s(k)\dot{s}(k)}{\partial \underline{\hat{\boldsymbol{q}}}^i} = \frac{\partial s(k)\dot{s}(k)}{\partial \hat{y}_{SEFT2FNN}} \frac{\partial \hat{y}_{SEFT2FNN}}{\partial y_l} \frac{\partial y_l}{\partial \underline{\hat{w}}^i} \frac{\partial \underline{\hat{w}}^i}{\partial \underline{\hat{\boldsymbol{q}}}^i} = -\frac{1}{2} s(k) \frac{f_l^i}{\sum_{i=1}^M f_l^i} \boldsymbol{\phi} \tag{28}$$

$$\frac{\partial s(k)\dot{s}(k)}{\partial \bar{\hat{\boldsymbol{q}}}^i} = \frac{\partial s(k)\dot{s}(k)}{\partial \hat{y}_{SEFT2FNN}} \frac{\partial \hat{y}_{SEFT2FNN}}{\partial y_r} \frac{\partial y_r}{\partial \bar{\hat{w}}^i} \frac{\partial \bar{\hat{w}}^i}{\partial \bar{\hat{\boldsymbol{q}}}^i} = -\frac{1}{2} s(k) \frac{f_r^i}{\sum_{i=1}^M f_r^i} \boldsymbol{\phi} \tag{29}$$

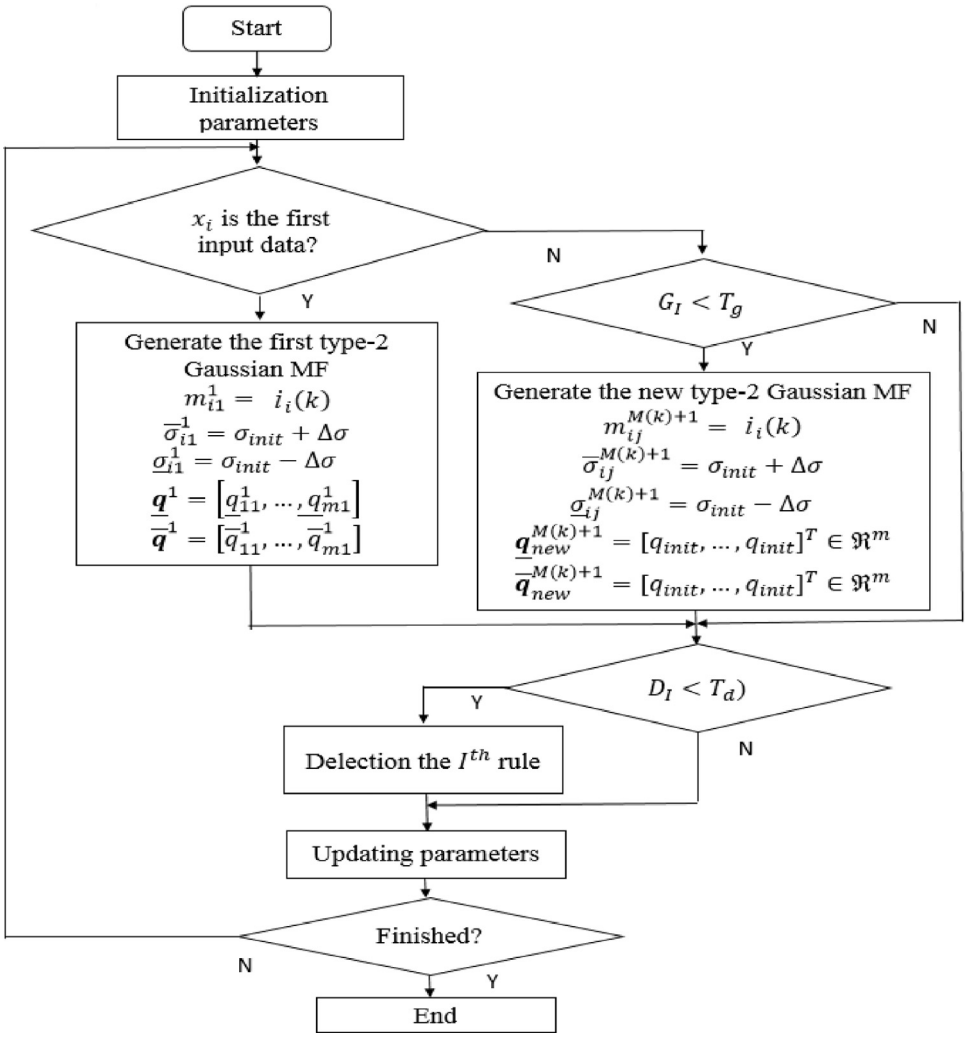

Fig. 4. Flowchart of the structure and parameter learning for the SEFT2FNN.

$$\frac{\partial s(k)\dot{s}(k)}{\partial \hat{m}_{ij}} = \frac{1}{2}\frac{\partial s(k)\dot{s}(k)}{\partial \hat{y}_{SEFT2FNN}}\left(\frac{\partial y_l}{\partial f_l^i}\frac{\partial f_l^i}{\partial \hat{m}_{ij}} + \frac{\partial y_r}{\partial f_r^i}\frac{\partial f_r^i}{\partial \hat{m}_{ij}}\right) = -\frac{1}{2}s(k)\left(\frac{(\underline{w}^i - y_l)}{\sum_{i=1}^M f_l^i}\frac{\partial f_l^i}{\partial \hat{m}_{ij}} + \frac{(\bar{w}^i - y_r)}{\sum_{i=1}^M f_r^i}\frac{\partial f_r^i}{\partial \hat{m}_{ij}}\right) \quad (30)$$

$$\frac{\partial s(k)\dot{s}(k)}{\partial \hat{\bar{\sigma}}_{ij}} = \frac{1}{2}\frac{\partial s(k)\dot{s}(k)}{\partial \hat{y}_{SEFT2FNN}}\left(\frac{\partial y_l}{\partial f_l^i}\frac{\partial f_l^i}{\partial \hat{\bar{\sigma}}_{ij}} + \frac{\partial y_r}{\partial f_r^i}\frac{\partial f_r^i}{\partial \hat{\bar{\sigma}}_{ij}}\right) = -\frac{1}{2}s(k)\left(\frac{(\underline{w}^i - y_l)}{\sum_{i=1}^M f_l^i}\frac{\partial f_l^i}{\partial \hat{\bar{\sigma}}_{ij}} + \frac{(\bar{w}^i - y_r)}{\sum_{i=1}^M f_r^i}\frac{\partial f_r^i}{\partial \hat{\bar{\sigma}}_{ij}}\right) \quad (31)$$

$$\frac{\partial s(k)\dot{s}(k)}{\partial \hat{\underline{\sigma}}_{ij}} = \frac{1}{2}\frac{\partial s(k)\dot{s}(k)}{\partial \hat{y}_{SEFT2FNN}}\left(\frac{\partial y_l}{\partial f_l^i}\frac{\partial f_l^i}{\partial \hat{\underline{\sigma}}_{ij}} + \frac{\partial y_r}{\partial f_r^i}\frac{\partial f_r^i}{\partial \hat{\underline{\sigma}}_{ij}}\right) = -\frac{1}{2}s(k)\left(\frac{(\underline{w}^i - y_l)}{\sum_{i=1}^M f_l^i}\frac{\partial f_l^i}{\partial \hat{\underline{\sigma}}_{ij}} + \frac{(\bar{w}^i - y_r)}{\sum_{i=1}^M f_r^i}\frac{\partial f_r^i}{\partial \hat{\underline{\sigma}}_{ij}}\right) \quad (32)$$

From (10) and (11), the elements f_l^i and f_r^i in (28)–(32) can be $\underline{f}^i$ or $\bar{f}^i$

$$\frac{\partial \underline{f}^i}{\partial \hat{m}_{ij}} = \frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial \hat{m}_{ij}} = \underline{f}^i\frac{x_j - \hat{m}_{ij}}{(\hat{\underline{\sigma}}_{ij})^2} \quad (33)$$

$$\frac{\partial \bar{f}^i}{\partial \hat{m}_{ij}} = \frac{\partial \bar{f}^i}{\partial \bar{\mu}_j^i}\frac{\partial \bar{\mu}_j^i}{\partial \hat{m}_{ij}} = \bar{f}^i\frac{x_j - \hat{m}_{ij}}{(\hat{\bar{\sigma}}_{ij})^2} \quad (34)$$

$$\frac{\partial \underline{f}^i}{\partial \hat{\bar{\sigma}}_{ij}} = \frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial \hat{\bar{\sigma}}_{ij}} = \underline{f}^i\frac{(x_j - \hat{m}_{ij})^2}{(\hat{\bar{\sigma}}_{ij})^3} \quad (35)$$

$$\frac{\partial \bar{f}^i}{\partial \hat{\bar{\sigma}}_{ij}} = \frac{\partial \bar{f}^i}{\partial \bar{\mu}_j^i}\frac{\partial \bar{\mu}_j^i}{\partial \hat{\bar{\sigma}}_{ij}} = \bar{f}^i\frac{(x_j - \hat{m}_{ij})^2}{(\hat{\bar{\sigma}}_{ij})^3} \quad (36)$$

$$\frac{\partial \underline{f}^i}{\partial \hat{\underline{\sigma}}_{ij}} = \frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial \hat{\underline{\sigma}}_{ij}} = \underline{f}^i\frac{(x_j - \hat{m}_{ij})^2}{(\hat{\underline{\sigma}}_{ij})^3} \quad (37)$$

$$\frac{\partial \bar{f}^i}{\partial \hat{\underline{\sigma}}_{ij}} = \frac{\partial \bar{f}^i}{\partial \bar{\mu}_j^i}\frac{\partial \bar{\mu}_j^i}{\partial \hat{\underline{\sigma}}_{ij}} = \bar{f}^i\frac{(x_j - \hat{m}_{ij})^2}{(\hat{\underline{\sigma}}_{ij})^3} \quad (38)$$

Using the online tuning parameter that is detailed in (23)–(27), the SEFT2FNN identifier is obtained and the system can achieve the desired performance. The stability of system is guaranteed using Lyapunov function approach, and it is proven in Appendix C.

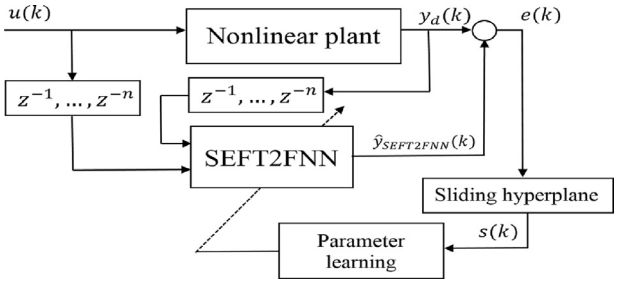

Fig. 5. Identification scheme.

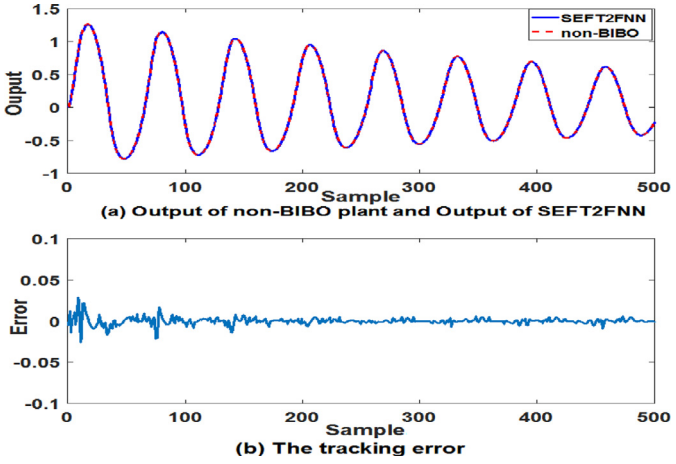

Fig. 6. The result of identification non-BIBO plant after 200 epochs training.

4. Simulation studies

4.1. System identification problems

The scheme for system identification is shown in Fig. 5. The term, z^{-n}, is the step delayed value for signals. For example, the signal $y(k-n)$ represents a signal $y(k)$ that is delayed n steps. The input for the system identifier SEFT2FNN is the nonlinear plant inputs, $u(k)$, $u(k-1),\ldots,$ $u(k-n)$, and the delay in the nonlinear plant output is $y_d(k-1)$, $y_d(k-2)$, $\ldots, y_d(k-n)$. System identification requires that the errors between the nonlinear plant output $y_d(k)$ and the output of $\hat{y}_{SEFT2FNN}$ converge to zero for all input values of $u(k)$. This study uses three inputs, $u(k)$, $y_d(k-1)$ and $y_d(k-2)$, to train the SEFT2FNN for all simulations. All the training data are the same as those used in [16,44]. To limit the cost of computation, the maximum number of T2GMF in each input is limited to 7 MFs.

Example 1: Identification of a non-BIBO nonlinear plant

The non-bounded-input bounded-output nonlinear plant (non-BIBO) used in [43] and [44] is described as

$$y(k+1) = 0.2y^2(k) + 0.2y(k-1) + 0.4\sin(0.5(y(k) + y(k-1)))cos(0.5(y(k)+y(k-1))) + 1.2u(k) \quad (39)$$

where $u(k)$ is the input signal, which is $u(k) = 0.5e^{-0.1kT_0}\sin(5kT_0)$, and $T_0 = 0.001$ is the sampling time.

The SET2FNN is trained for 200 epochs using 500 samples. Fig. 6a shows the output for a non-BIBO system, $y_d(k)$, and the output for the identifier, $\hat{y}_{\text{SEFT2FNN}}(k)$. Fig. 6b shows the tracking error between $y_d(k)$ and $\hat{y}_{\text{SEFT2FNN}}(k)$ during the online identification. Fig. 7 shows the evolution of the root mean square error (RMSE) during 200 epochs of training. Fig. 8 shows the change in the number of input membership functions during 200 epochs (total sample is 500*200). The change in the number membership functions is not shown clearly because the algorithm converges

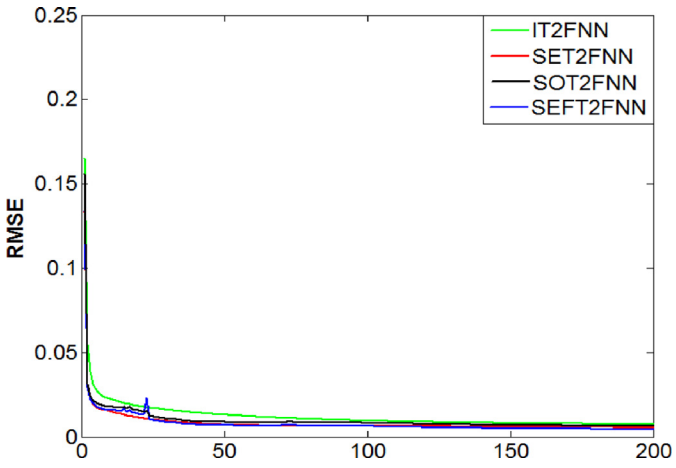

Fig. 7. The RMSE during 200 training epochs.

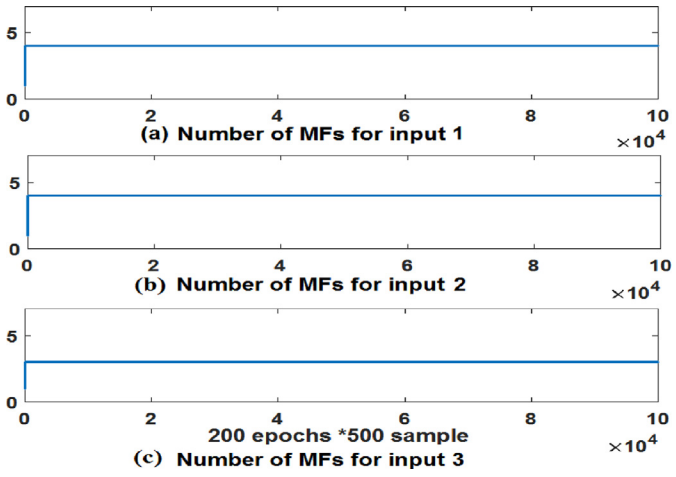

Fig. 8. The number of input MFs during 200 training epochs.

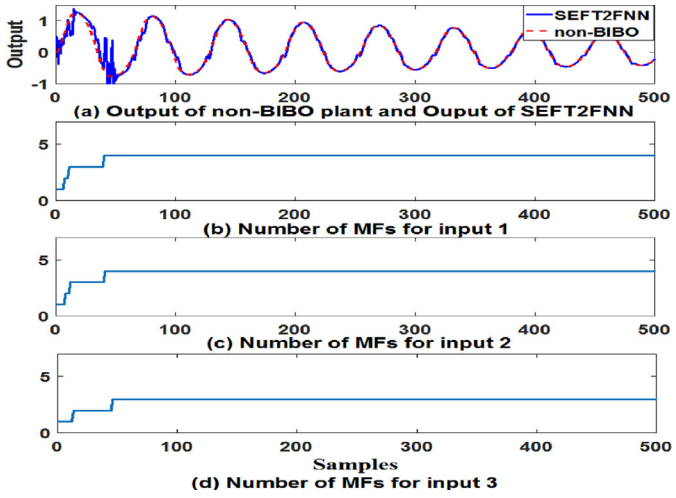

Fig. 9. The result during 1 training epoch.

quickly to a suitable structure after training for a few epochs. Fig. 9 shows the result during training one epoch. It shows that the changes in the input membership functions only occur for the first 50 samples. When the error converges to zero, the number of rules quickly converges to an optimum number. Table 1 compares the RMSE values and computation times for the IT2FNN, a self-organizing IT2FNN (SOT2FNN), a self-evolving IT2FNN (SET2FNN)

Table 1.
Comparison results in RMSE of non-BIBO plant.

	Computation time (s)	RMSE
IT2FNN	0.410	0.0092
SOT2FNN [28]	0.458	0.0083
SET2FNN	0.472	0.0076
SEFT2FNN	0.507	0.0065

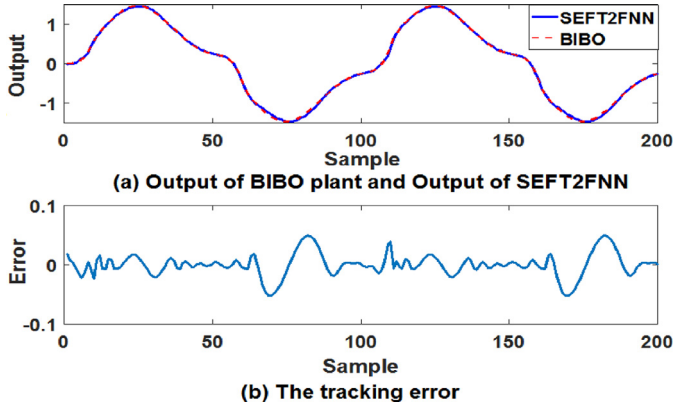

Fig. 10. Identification BIBO nonlinear plant.

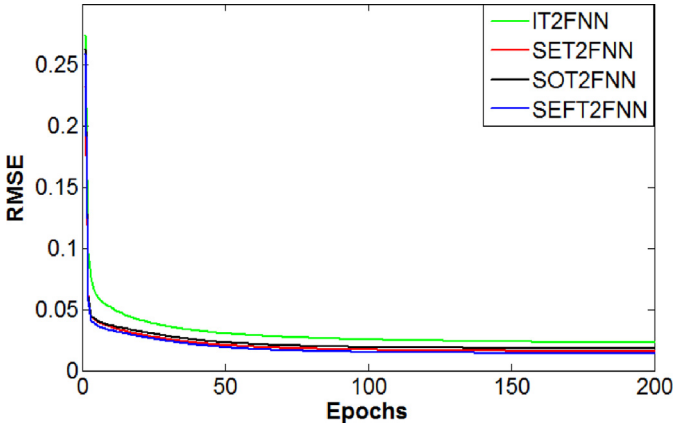

Fig. 11. The RMSE during 200 training epochs.

and Self-evolving function-link IT2NN (the proposed method). This comparison shows that the proposed SEFT2FNN performs better, in terms of identification than the other methods. However, computation time is greater.

Example 2: Identification of a BIBO nonlinear plant

In this example, the nonlinear bounded-input bounded-output (BIBO) nonlinear plant that was used in [15] and [16] is used to evaluate the performance of the SEFT2FNN identifier

$$y(k) = u(k)^3 + \frac{y(k-1)}{1+y(k-1)^2} \tag{40}$$

where the input signal is given as $u(k) = \sin(\frac{2\pi k}{100})$.

After training for 200 epochs, using 200 samples, the SET2FNN identifies the plant with a small error. Theoutput for the BIBO nonlinear plant, $y_d(k)$, and the output of the identifier, $\hat{y}_{\text{SEFT2FNN}}(k)$, are shown in Fig. 10a. The tracking error during the online identification is shown in Fig. 10b. The evolution of the RMSE value over 200 epochs is shown in Fig. 11. Fig. 12 shows the change in the number of input membership functions over 200 epochs (total sample is 200*200). Table 2 compares the RMSE values and the computation times for the proposed method and the other methods. The

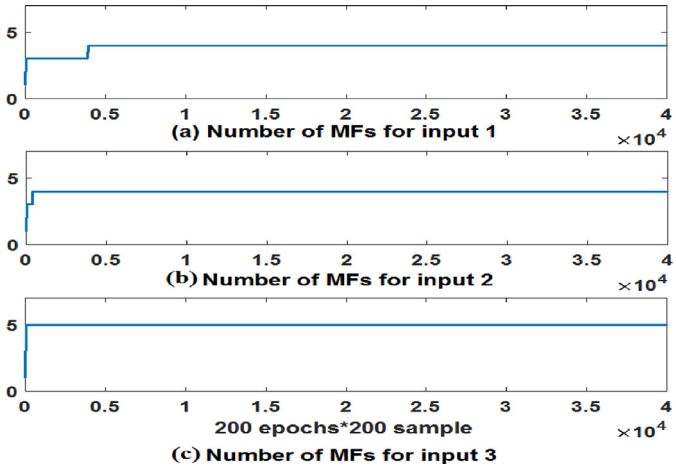

Fig. 12. The number of input MFs during 200 training epochs.

Table 2.
Comparison results in RMSE of BIBO plant.

	Computation time (s)	RMSE
T2 TSK FNS [16]	None	0.032
T2FNN	0.142	0.038
SOT2FNN [28]	0.149	0.035
SET2FNN	0.155	0.029
SEFT2FNN	0.171	0.026

result of this comparison results are similar to those for the previous example.

Example 3: Identification of a second-order nonlinear time-varying plant

The second-order nonlinear time-varying plant that is used in this simulation is described in [44] and [16] by the dynamic equation:

$$y(k) = \frac{x_1 x_2 + x_3}{x_4} \tag{41}$$

where $x_1 = y(k-1)y(k-2)y(k-3)u(k-1)$, $x_2 = y(k-3) - b(k)$, $x_3 = c(k)u(k)$, and $x_4 = a(k) + y(k-2)^2 + y(k-3)^2$, in which $a(k)$, $b(k)$ and $c(k)$ are time-varying parameters that are given by:

$$\begin{aligned} a(k) &= 1.2 - 0.2\cos(2\pi k T_0) \\ b(k) &= 1 - 0.4\sin(2\pi k T_0) \\ c(k) &= 1 + 0.4\sin(2\pi k T_0) \end{aligned} \tag{42}$$

and $u(k)$ is the input signal, which is given as:

$$u(k) = \begin{cases} \sin\left(\frac{\pi k}{25}\right) & k < 250 \\ 1.0 & 250 \le k < 500 \\ -1.0 & 500 \le k < 750 \\ 0.3\sin\left(\frac{\pi k}{25}\right) + 0.1\sin\left(\frac{\pi k}{32}\right) + 0.6\sin\left(\frac{\pi k}{10}\right) & 750 \le k < 1000 \end{cases} \tag{43}$$

The SET2FNN system identification is trained for 100 epochs using 1000 samples. Fig. 13a shows the output for the second-order nonlinear time-varying plant, $y_d(k)$, and the output of the identifier, $\hat{y}_{\text{SEFT2FNN}}(k)$. Fig. 13b shows the tracking error for the system during the online identification. Fig. 14 shows the evolution of the RMSE value over 100 epochs. Fig. 15 shows the change in the number of input membership functions over 100 epochs (total sample is 1000*100). The RMSE values and the computation times for the proposed method and the other methods are compared in Table 3.

From Figs. 7, 11, 14 and the comparison results for the RMSE in Tables 1–3, it is clear that the proposed method gives a faster and better result for the identification problem. The RMSE

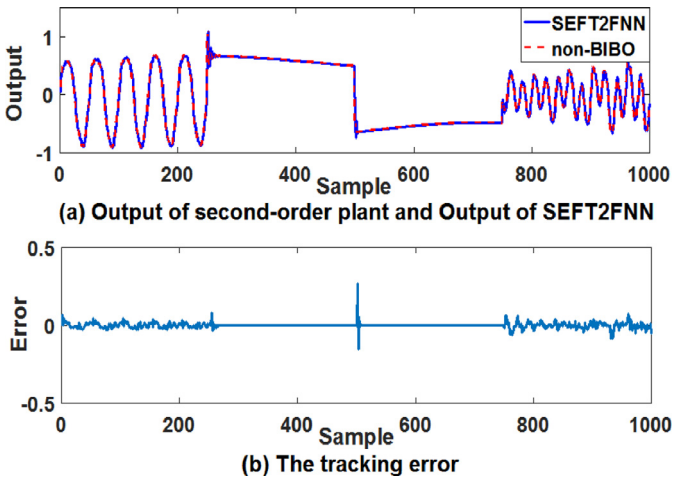

Fig. 13. Identification second-order nonlinear time-varying plant.

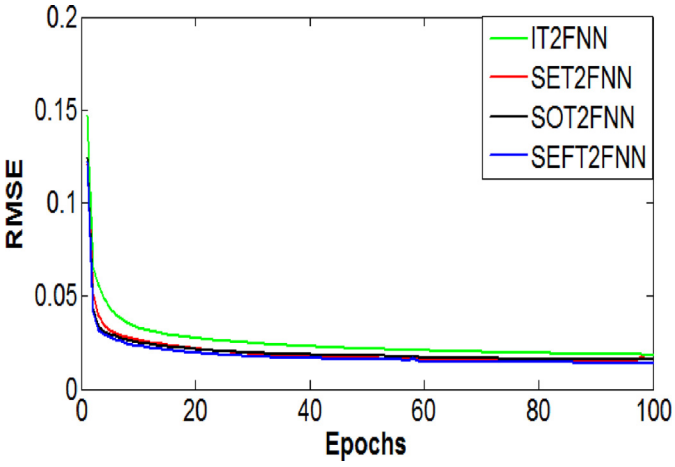

Fig. 14. The RMSE during 100 training epochs.

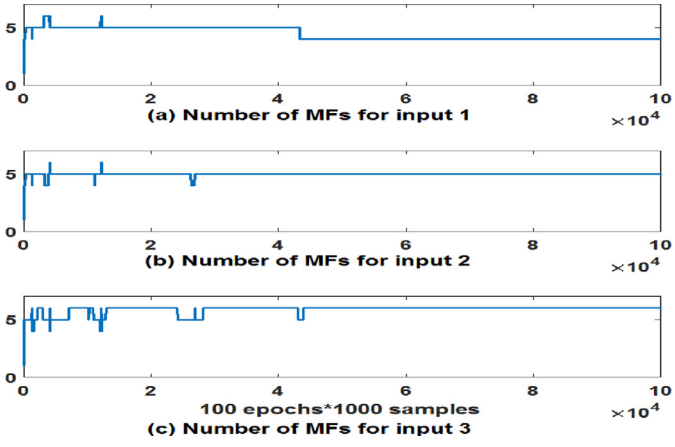

Fig. 15. The number of input MFs during 100 training epochs.

Table 3.
Comparison results in RMSE of second-order nonlinear time-varying plant.

	Computation time (s)	RMSE
SMC-based learning [44]	0.853	0.028
T2FNN	0.874	0.037
SOT2FNN [28]	0.972	0.034
SET2FNN	0.981	0.031
SEFT2FNN	1.040	0.023

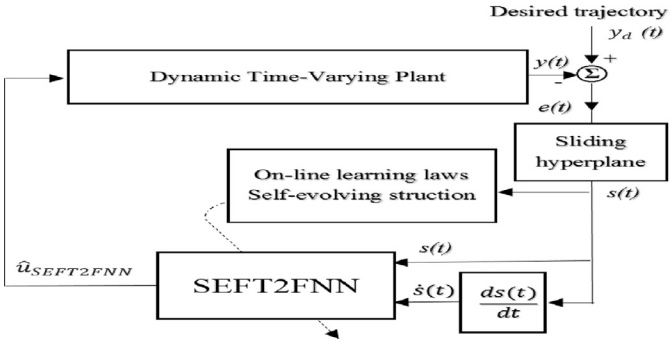

Fig. 16. The control system scheme for dynamic time-varying plant.

values converge faster during training epochs than the other methods. Because the processing time of the self-evolving algorithm and the function-link network, the computation time of the proposed method is a little longer, but it is acceptable. Compared with the proposed method, at the beginning of the processing, the RMSE of SOT2FNN has been decreased more regularly. However, when the SEFT2FNN converges to the suitable structure, it can obtain the smaller value in RMSE.

4.2. Control problems (Dynamic time-varying plants)

The scheme for the control system is shown in Fig. 16. The input for the SEFT2FNN control system is the output of the sliding hyperplane and its derivative, $s(t)$ and $\dot{s}(t)$. The goal of control system is generates the control signal $\hat{u}_{SEFT2FNN}(t)$, which can force the output of the dynamic time-varying plant $y(t)$ to track the reference signal $y_d(t)$. The same with examples in identification problem, the maximum number of T2GMF in each input is limited to 7 MFs.

Example 4: Control the dynamic time-varying plant borrowed from [45]

$$y(t) = \frac{y(t-1)y(t-2)(y(t-1)+2.5)}{\left(1+y(t-1)^2+y(t-2)^2\right)} + u(t) \tag{44}$$

where $y(t)$ and $u(t)$ are output of the plant and control signal, respectively. $y(t-1)$ and $y(t-2)$ denote the one-step and two-step delayed of $y(t)$. The reference signal $y_d(t)$ is given by the stepwise changes as:

$$y_d(t) = \begin{cases} 10, & 0 < k \le 50 \\ 15, & 50 < k \le 100 \\ 10, & 100 < k \le 150 \\ 15, & 150 < k \le 200 \end{cases} \tag{45}$$

Fig. 17 shows the control result of the time-varying plant in Example 4. Fig. 17a includes the reference signal and the output of control system, Fig. 17b show control signal, and Fig. 17c is the tracking error of control system. It is obvious that the SEFT2FNN can quickly generate the rules and control the time-varying plant very well. The RMSE of the control system over 200 s is 0.1359. After about 0.05 s, the number of MFs are converged to 3 MFs for input 1, and 2 MFs for input 2. The change of MFs for input 1 and input 2 during 200 s is shown in Fig. 18.

Example 5. *Control the dynamic time-varying plant borrowed from [46]*

$$y(t) = b_0(t)u(t) \tag{46}$$

where

$$b_0(t) = -\frac{t^2}{1+a_1(t)t+a_2(t)t^2}$$

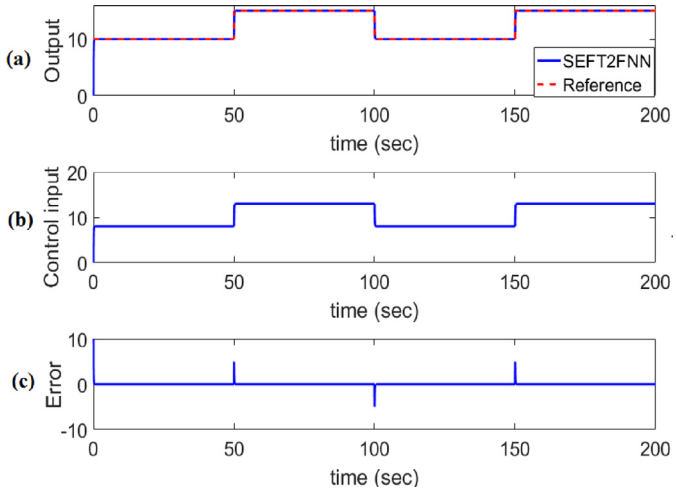

Fig. 17. Control the time-varying plant in Example 4. (a) The reference signal and the output of control system. (b) The control signal. (c) The tracking error of control system.

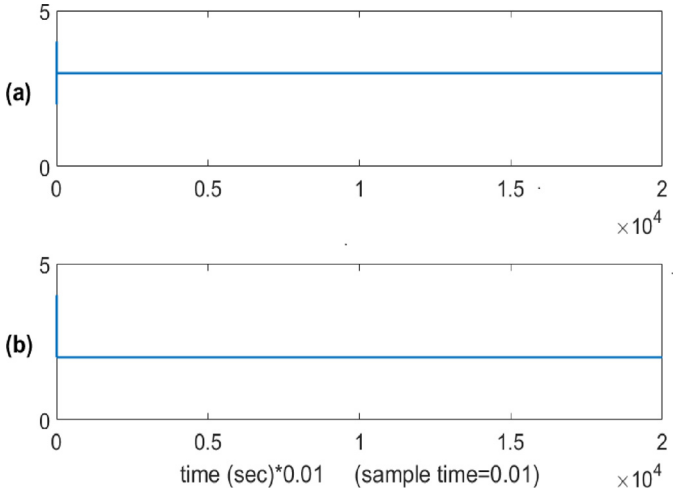

Fig. 18. The number of MFs during 200 s in Example 4. (a) The number of MFs for input 1. (b) The number of MFs for input 2.

$$b_1(t) = \frac{2 + a_1(t)t}{1 + a_1(t)t + a_2(t)t^2}$$

$$b_2(t) = -\frac{1}{1 + a_1(t)t + a_2(t)t^2}$$

Here, $a_1(t)$ and $a_2(t)$ are the time-varying plant parameter given by

$$a_1(t) = \frac{0.1t}{t+1} \qquad a_2(t) = \begin{cases} 0.3, & 0 \le k < 40 \\ 0.1, & 40 \le k < 60 \\ 0.6, & 60 \le k < 85 \\ 0.3, & k > 85 \end{cases} \tag{47}$$

The reference signal is given by

$$y_d(t) = \begin{cases} 10, & 0 < k \le 25 \\ 15, & 25 < k \le 50 \\ 10, & 50 < k \le 75 \\ 15, & 75 < k \le 100 \end{cases} \tag{48}$$

The change of time-varying parameter *a1* and *a2* during 100 s in Example 5 are shown in Fig. 19. The reference signal and the output of control system are shown in Fig. 20a, and the control signal and the tracking error of control system are shown in Fig. 20b and Fig. 20c, respectively. Fig. 21 shows the change of number MFs during 100 s. From the simulation results, it can be observed that the SEFT2FNN controller can control the time-varying plant follow the reference signal very well. The structure of the proposes controller can quickly converge to the suitable structure by self-evolving algorithm. In this example, the number MFs of input 1 and input 2 are converged to 2 MFs. The RMSE in this example is 0.7255. The comparison RMSE of the proposed method with the other methods are shown in Table 4.

In all examples, the structure of the SEFT2FNN does not need to design in advance, because it can self-evolving to the suitable network structure. The parameter in the sliding hyperplane is chosen as $n = 3$ and $\lambda = 0.05$. The initial parameter for the consequent and the antecedent in IT2FNN also does not need to be designed in advance, and the rules can be auto generated using the input signal and all parameters can be updated based on the adaptive

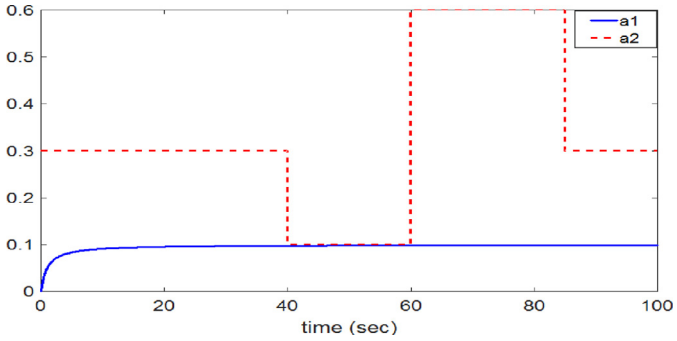

Fig. 19. The change of time-varying parameter a1 and a2 during 100 s in Example 5.

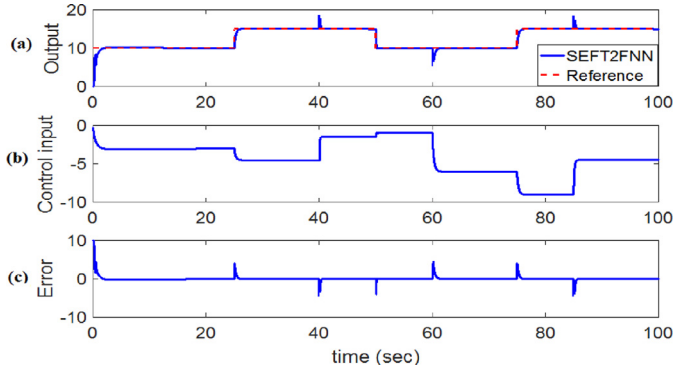

Fig. 20. Control the time-varying plant in Example 5. (a) The reference signal and the output of control system. (b) The control signal. (c) The tracking error of control system.

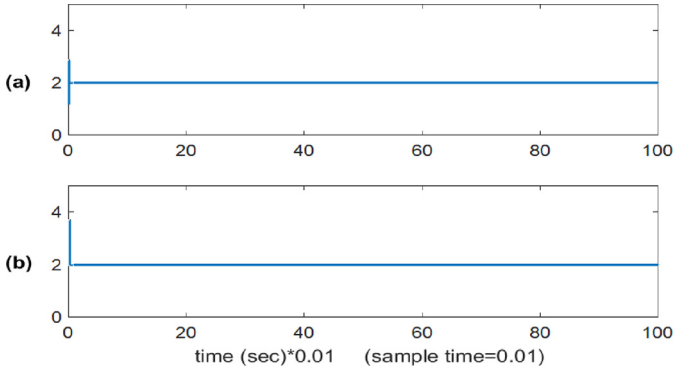

Fig. 21. The number of MFs during 100 s in Example 5. (a) The number of MFs for input 1. (b) The number of MFs for input 2.

Table 4.
Comparison results in RMSE of control time-varying plant.

	Example 4	Example 5
Type-2 TSK FNS [16]	0.1469	0.7395
T2FNN	0.2471	0.8439
SOT2FNN [28]	0.1427	0.7825
SET2FNN	0.1583	0.7540
SEFT2FNN	0.1359	0.7255

law. The initial parameters for generating the variance are chosen as $\sigma_{init} = 0.5$ and $\Delta\sigma = 0.1$. The learning-rates are chosen as, $\hat{\eta}_m = 0.01$ and $\hat{\eta}_\sigma = 0.01$. The prior threshold for generating and deleting rule are chosen as $T_g = 0.1$ and $T_d = 0.05$. The sample time is 0.01 s.

5. Conclusion

This paper proposes a SEFT2FNN for system identification and the control problem. The proposed method is suitable for many fields, such as control problems, system identification, classification, and prediction. The major contributions of this study are (1) the development of a SEFT2FNN with an adaptive law for updating parameters, (2) a self-evolving algorithm that allows the network to automatically achieve optimum construction from empty rules, so there is no need to design the structure of the SEFT2FNN in advance, (3) the convergence of the proposed algorithm is proven by Lyapunov function analysis approach and (4) the function-link network is combined to improve the accuracy with which the nonlinear function is approximated. The numerical simulation results for an identification problem and the control of time-varying plants show the superiority of the proposed method over other methods. Choosing the learning rates for the adaptive laws and the thresholds for generating and deleting the rules significantly affect the system performance. Therefore, the future studies will apply the optimal algorithm to optimize the learning rates and the thresholds, so the RMSE can be quickly converged and the performance of the system can be further improved.

Acknowledgments

The authors appreciate the financial support in part from the Nation Science Council of Republic of China under Grant NSC 101-2221-E-155-026-MY3.

Appendix A. [47]

The detail of KM algorithm for finding the right switch point, R.

Step 1: Sort $\bar{w}^i$ $(i = 1, 2, \ldots, M)$ such that $\bar{w}^1 \le \bar{w}^2 \le \cdots \le \bar{w}^M$
Step 2: Computing f_r^i and y

$$f_r^i = \frac{\bar{f}^i + \underline{f}^i}{2} \text{ and } y = \frac{\sum_{i=1}^{M} f_r^i \bar{w}^i}{\sum_{i=1}^{M} f_r^i}$$

Step 3: Find the point k $(1 \le k \le M-1)$ where $\bar{w}^k \le y \le \bar{w}^{k+1}$
Step 4: Set $f_r^i = \begin{cases} \underline{f}^i, & i \le k \\ \bar{f}^i, & i > k \end{cases}$ and compute $y' = \frac{\sum_{i=1}^{M} f_r^i \bar{w}^i}{\sum_{i=1}^{M} f_r^i}$
Step 5: If $y' \ne y$ Then set $y = y'$ and go to Step 3

If $y' = y$ Then set $y_r = y$; $R = k$ and Stop KM algorithm

Appendix B. [47]

The detail of KM algorithm for finding the left switch point, L.

Step 1: Sort $\underline{w}^i$ $(i = 1, 2, \ldots, M)$ such that $\underline{w}^1 \le \underline{w}^2 \le \cdots \le \underline{w}^M$

Step 2: Computing f_l^i and y

$$f_l^i = \frac{\bar{f}^i + \underline{f}^i}{2} \text{ and } y = \frac{\sum_{i=1}^{M} f_l^i \underline{w}^i}{\sum_{i=1}^{M} f_l^i}$$

Step 3: Find the point k $(1 \le k \le M-1)$ where $\underline{w}^k \le y \le \underline{w}^{k+1}$
Step 4: Set $f_l^i = \begin{cases} \bar{f}^i, & i \le k \\ \underline{f}^i, & i > k \end{cases}$ and compute $y' = \frac{\sum_{i=1}^{M} f_l^i \underline{w}^i}{\sum_{i=1}^{M} f_l^i}$
Step 5: If $y' \ne y$ Then set $y = y'$ and go to Step 3

If $y' = y$ Then set $y_l = y$; $L = k$ and Stop KM algorithm

Appendix C

Proof. The Lyapunov function is defined as:

$$V(s(k)) = \frac{1}{2} s^2(k) \tag{C1}$$

$$\dot{V}(s(k)) = s(k)\dot{s}(k) \tag{C2}$$

The change in (C2) can be obtained

$$\text{Defined} \quad \boldsymbol{P}_x(k) = \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \boldsymbol{x}}, \qquad \text{for} \quad \boldsymbol{x} = \underline{\hat{\boldsymbol{q}}}, \bar{\hat{\boldsymbol{q}}}, \hat{\boldsymbol{m}}, \underline{\hat{\sigma}}, \bar{\hat{\sigma}} \tag{C3}$$

where

$$\boldsymbol{P}_{\underline{\hat{q}}}(k) = \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\boldsymbol{q}}}} = \left[\frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{q}}_{11}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{q}}_{1n_j}}, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{q}}_{21}}, \right.$$
$$\left. \times \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{q}}_{2n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{q}}_{n_i n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{q}}_{n_i n_j}} \right]$$

$$\boldsymbol{P}_{\bar{\hat{q}}}(k) = \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\boldsymbol{q}}}} = \left[\frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{q}}_{11}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{q}}_{1n_j}}, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{q}}_{21}}, \right.$$
$$\left. \times \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{q}}_{2n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{q}}_{n_i n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{q}}_{n_i n_j}} \right]$$

$$\boldsymbol{P}_{\hat{m}}(k) = \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{\boldsymbol{m}}} = \left[\frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{m}_{11}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{m}_{1n_j}}, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{m}_{21}}, \right.$$
$$\left. \times \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{m}_{2n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{m}_{n_i n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \hat{m}_{n_i n_j}} \right]$$

$$\boldsymbol{P}_{\underline{\hat{\sigma}}}(k) = \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\boldsymbol{\sigma}}}} = \left[\frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\sigma}}_{11}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\sigma}}_{1n_j}}, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\sigma}}_{21}}, \right.$$
$$\left. \times \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\sigma}}_{2n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\sigma}}_{n_i n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \underline{\hat{\sigma}}_{n_i n_j}} \right]$$

$$\boldsymbol{P}_{\bar{\hat{\sigma}}}(k) = \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\boldsymbol{\sigma}}}} = \left[\frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\sigma}}_{11}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\sigma}}_{1n_j}}, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\sigma}}_{21}}, \right.$$
$$\left. \times \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\sigma}}_{2n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\sigma}}_{n_i n_j}}, \ldots, \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \bar{\hat{\sigma}}_{n_i n_j}} \right]$$

Apply the gradient descent method, C(2) can be represented by

$$\dot{V}(s(k+1)) = \dot{V}(s(k)) + \Delta\dot{V}(s(k)) \cong \dot{V}(s(k)) + \left[\frac{\partial \dot{V}(s(k))}{\partial \boldsymbol{x}} \right]^T \Delta\boldsymbol{x} \tag{C4}$$

where $\Delta\dot{V}(s(k))$ is the change in $\dot{V}(s(k))$ and $\Delta\boldsymbol{x}$ denotes the change in $\boldsymbol{x}$.

Using the chain rule and obtain

$$\frac{\partial \dot{V}(s(k))}{\partial \boldsymbol{x}} = \frac{\partial \dot{V}(s(k))}{\partial \hat{y}_{SEFT2FNN}} \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \boldsymbol{x}} = \frac{\partial s(k)\dot{s}(k)}{\partial \hat{y}_{SEFT2FNN}} \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \boldsymbol{x}} \quad \text{(C5)}$$

Using (27) and (C4) yields

$$\frac{\partial \dot{V}(s(k))}{\partial \boldsymbol{x}} = -s(k) \frac{\partial \hat{y}_{SEFT2FNN}}{\partial \boldsymbol{x}} = -s(k)\boldsymbol{P}_x(k) \quad \text{(C6)}$$

From (28)–(32), we have

$$\Delta \boldsymbol{x} = -\hat{\eta}_x \frac{\partial s(k)\dot{s}(k)}{\partial \boldsymbol{x}} = \hat{\eta}_x s(k)\boldsymbol{P}_x(k) \quad \text{(C7)}$$

Substituting (C6), (C7) into (C4)

$$\begin{aligned} \Delta \dot{V}(s(k)) &= \left[\frac{\partial \dot{V}(s(k))}{\partial \boldsymbol{x}}\right]^T \Delta \boldsymbol{x} = [-s(k)\boldsymbol{P}_x(k)]^T * \hat{\eta}_x s(k)\boldsymbol{P}_x(k) \\ &= -s^2(k)\hat{\eta}_x \boldsymbol{P}_x(k) \end{aligned} \quad \text{(C8)}$$

From (C8) if $\hat{\eta}_x$ is chosen as $\hat{\eta}_x > 0$ then $\Delta \dot{V}(s(k)) < 0$. Therefore, the convergence of the updating algorithm is guaranteed by the Lyapunov stability theorem.

References

[1] M. Farahani, S. Ganjefar, An online trained fuzzy neural network controller to improve stability of power systems, Neurocomputing 162 (2015) 245–255.
[2] Y.C. Liu, S.Y. Liu, N. Wang, Fully-tuned fuzzy neural network based robust adaptive tracking control of unmanned underwater vehicle with thruster dynamics, Neurocomputing 196 (2016) 1–13.
[3] A. Rubaai, P. Young, Hardware/software implementation of fuzzy-neural-network self-learning control methods for brushless DC motor drives, IEEE Trans. Ind. Appl. 52 (1) (2016) 414–424.
[4] R.J. Wai, Y.F. Lin, Y.K. Liu, Design of adaptive fuzzy-neural-network control for a single-stage boost inverter, IEEE Trans. Power Electron. 30 (12) (2015) 7282–7298.
[5] S. Tong, Y. Li, Adaptive fuzzy output feedback tracking backstepping control of strict-feedback nonlinear systems with unknown dead zones, IEEE Trans. Fuzzy Syst. 20 (1) (2012) 160–180.
[6] Y. Li, S. Tong, L. Liu, G. Feng, Adaptive output-feedback control design with prescribed performance for switched nonlinear systems, Automatica 80 (2017) 225–231.
[7] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (3) (1965) 338–353.
[8] T. Tao, S.F. Su, Moment adaptive fuzzy control and residue compensation, IEEE Trans. Fuzzy Syst. 22 (4) (2014) 803–816.
[9] Y.Y. Lin, S.H. Liao, J.Y. Chang, C.T. Lin, Simplified interval type-2 fuzzy neural networks, IEEE Trans. Neural Netw. Learn. Syst. 25 (5) (2014) 959–969.
[10] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning, Inf. Sci. 8 (3) (1975) 199–249.
[11] O. Castillo, R. Martínez-Marroquín, P. Melin, F. Valdez, J. Soria, Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot, Inf. Sci. 192 (2012) 19–38.
[12] J.M. Mendel, A quantitative comparison of interval type-2 and type-1 fuzzy logic systems: first results, in: Proceedings of IEEE International Conference on Fuzzy Systems, 2010, pp. 1–8.
[13] S.K. Oh, H.J. Jang, W. Pedrycz, A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization, Expert Syst. Appl. 38 (9) (2011) 11217–11229.
[14] Q. Liang, J.M. Mendel, Interval type-2 fuzzy logic systems: theory and design, IEEE Trans. Fuzzy Syst. 8 (5) (2000) 535–550.
[15] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990) 4–27.
[16] R.H. Abiyev, O. Kaynak, Type 2 fuzzy neural structure for identification and control of time-varying plants, IEEE Trans. Ind. Electron. 57 (12) (2010) 4147–4159.
[17] R. Martínez, O. Castillo, L.T. Aguilar, Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms, Inf. Sci. 179 (13) (2009) 2158–2174.
[18] M.A. Khanesar, M. Teshnehlab, E. Kayacan, O. Kaynak, A novel type-2 fuzzy membership function: application to the prediction of noisy data, in: Proceedings of IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2010, pp. 128–133.
[19] P. Melin, O. Castillo, A review on type-2 fuzzy logic applications in clustering, classification and pattern recognition, Appl. Soft Comput. 21 (2014) 568–577.
[20] J.R. Castro, O. Castillo, P. Melin, A. Rodríguez-Díaz, A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks, Inf. Sci. 179 (13) (2009) 2175–2193.
[21] H. Hagras, C. Wagner, Introduction to interval type-2 fuzzy logic controllers—towards better uncertainty handling in real world applications, IEEE Syst. Man Cybern. 27 (2009) eNewsletter.
[22] O. Castillo, J.R. Castro, P. Melin, A. Rodríguez-Díaz, Universal approximation of a class of interval type-2 fuzzy neural networks in nonlinear identification, Adv. Fuzzy Syst. 2013 (2013) 1–16.
[23] O. Castillo, J.R. Castro, P. Melin, A. Rodríguez-Díaz, Application of interval type-2 fuzzy neural networks in non-linear identification and time series prediction, Soft Comput. 18 (6) (2014) 1213–1224.
[24] F. Gaxiola, P. Melin, F. Valdez, J.R. Castro, O. Castillo, Optimization of type-2 fuzzy weights in backpropagation learning for neural networks using GAs and PSO, Appl. Soft Comput. 38 (2016) 860–871.
[25] C.F. Juang, Y.W. Tsao, A type-2 self-organizing neural fuzzy system and its FPGA implementation, IEEE Trans. Syst. Man Cybern. Part B (Cybern.) 38 (6) (2008) 1537–1548.
[26] C. Li, C.Y. Lee, Self-organizing neuro-fuzzy system for control of unknown plants, IEEE Trans. Fuzzy Syst. 11 (1) (2003) 135–150.
[27] C.M. Lin, T.Y. Chen, Self-organizing CMAC control for a class of MIMO uncertain nonlinear systems, IEEE Trans. Neural Netw. 20 (9) (2009) 1377–1384.
[28] C.M. Lin, T.L. Le, PSO-self-organizing interval type-2 fuzzy neural network for antilock braking systems, Int. J. Fuzzy Syst. 19 (2017) 1362–1374.
[29] C.F. Juang, R.B. Huang, Y.Y. Lin, A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing, IEEE Trans. Fuzzy Syst. 17 (5) (2009) 1092–1105.
[30] C.F. Juang, Y.W. Tsao, A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1411–1424.
[31] Y.Y. Lin, J.Y. Chang, C.T. Lin, A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications, IEEE Trans. Ind. Electron. 61 (1) (2014) 447–459.
[32] A. Mohammadzadeh, S. Ghaemi, O. Kaynak, S. Khanmohammadi, Robust H∞ based synchronization of the fractional order chaotic systems by using new self-evolving non-singleton type-2 fuzzy neural networks, IEEE Trans. Fuzzy Syst. 24 (6) (2016) 1544–1554.
[33] Y.H. Pao, Functional link nets: removing hidden layers, AI Expert 4 (4) (1989) 60–68.
[34] C.M. Lin, Y.L. Liu, H.Y. Li, SoPC-based function-link cerebellar model articulation control system design for magnetic ball levitation systems, IEEE Trans. Ind. Elect. 61 (8) (2014) 4265–4273.
[35] Y.C. Hu, F.M. Tseng, Functional-link net with fuzzy integral for bankruptcy prediction, Neurocomputing 70 (16) (2007) 2959–2968.
[36] F.J. Lin, L.T. Teng, J.W. Lin, S.Y. Chen, Recurrent functional-link-based fuzzy-neural-network-controlled induction-generator system using improved particle swarm optimization, IEEE Trans. Ind. Electron. 56 (5) (2009) 1557–1577.
[37] F.J. Lin, S.Y. Chen, L.T. Teng, H. Chu, Recurrent functional-link-based fuzzy neural network controller with improved particle swarm optimization for a linear synchronous motor drive, IEEE Trans. Magn. 45 (8) (2009) 3151–3165.
[38] C.M. Lin, H.Y. Li, Intelligent hybrid control system design for antilock braking systems using self-organizing function-link fuzzy cerebellar model articulation controller, IEEE Trans. Fuzzy Syst. 21 (6) (2013) 1044–1055.
[39] K.A. Toh, W.Y. Yau, Fingerprint and speaker verification decisions fusion using a functional link network, IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) 35 (3) (2005) 357–370.
[40] J.M. Mendel, Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions, Prentice Hall, Upper Saddle River, 2001.
[41] M. Manceur, N. Essounbouli, A. Hamzaoui, Second-order sliding fuzzy interval type-2 control for an uncertain system with real application, IEEE Trans. Fuzzy Syst. 20 (2) (2012) 262–275.
[42] J.J.E. Slotine, W. Li, Applied Nonlinear Control, Prentice-Hall, Englewood Cliffs, NJ, 1991.
[43] S. Ahmed, N. Shakev, A. Topalov, K. Shiev, O. Kaynak, Sliding mode incremental learning algorithm for interval type-2 Takagi–Sugeno–Kang fuzzy neural networks, Evol. Syst 3 (3) (2012) 179–188.
[44] E. Kayacan, E. Kayacan, M.A. Khanesar, Identification of nonlinear dynamic systems using type-2 fuzzy neural networks—a novel learning algorithm and a comparative study, IEEE Trans. Ind. Electron. 62 (3) (2015) 1716–1724.
[45] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990) 4–27.
[46] C.J. Zhang, C. Shao, T.Y. Chai, Indirect adaptive control for a class of linear time-varying plants, IEE Proc. – Control Theory Appl. 145 (2) (1998) 141–149.
[47] D. Wu, M. Nie, Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems, in: Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ), 2011, pp. 2131–2138.

Chih-Min Lin was born in Taiwan, in 1959. He received the B.S. and M.S. degrees from Department of Control Engineering and the Ph.D. degree from Institute of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 1981, 1983 and 1986, respectively. He is currently a Chair Professor and the Vice President of Yuan Ze University, Chung-Li, Taiwan. His current research interests include fuzzy neural network, cerebellar model articulation controller, intelligent control systems and signal processing. He has published more than 180 journal papers. Dr. Lin was the recipient of an Honor Research Fellow at the University of Auckland, Auckland, New Zealand, from 1997 to 1998. He also serves as an Associate Editor of *IEEE Transactions on Cybernetics* and *IEEE Transactions on Fuzzy Systems*. He is an IEEE Fellow and IET Fellow.

Tien-Loc Le was born in Vietnam, in 1985. He received the B.S. degree in electronics and telecommunication engineering from the LacHong University, Vietnam in 2009, where he is currently a lecturer, and M.S. degree in electrical engineering from Hochiminh University of Technical Education, Vietnam, in 2012. He is curently working toward the Ph.D. degree with Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan. His research interest include intelligent control systems, fuzzy neural network and cerebellar model articulation controller.

Tuan-Tu Huynh was born in Ho Chi Minh City, Viet Nam, 1982. He received the B.S. degree from Department of Electrical & Electronics Engineering, Ho Chi Minh University of Technology and Education, Ho Chi Minh, in 2005, and M.S degree in automation from Ho Chi Minh City University of Transport, Ho Chi Minh, Viet Nam, in 2010. He is currently a PhD student at Yuan Ze University, ZhongLi, Taiwan; and a lecturer at LacHong University, Viet Nam. His research interests include fuzzy logic control, cerebellar model articulation controller, and intelligent control systems.