Optimum Design of Function-Link Type-2 Fuzzy Asymmetric CMAC Based on Self-Organizing Algorithm and Modified Jaya Algorithm

TIEN-LOC LE	extsuperscript{1,2}, (Member, IEEE), NGUYEN VU QUYNH	extsuperscript{2}, NGO KIM LONG	extsuperscript{2}, AND SUNG KYUNG HONG	extsuperscript{1}

	extsuperscript{1}Faculty of Mechanical and Aerospace, Sejong University, Seoul 05006, South Korea

	extsuperscript{2}Faculty of Mechatronics and Electronics, Lac Hong University, Bien Hoa 810000, Vietnam

Corresponding author: Sung Kyung Hong (skhong@sejong.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT) through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Promotion (IITP), South Korea, under Grant IITP-2020-2018-0-01423, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2020R1A6A1A03038540.

ABSTRACT In this work, a novel self-organizing interval type-2 fuzzy cerebellar model articulation controller design for chaotic systems synchronization is presented. A function-link network and an asymmetric-membership-function are employed to improve the learning ability and flexibility of the proposed network. The gradient-descent method is used to derive the adaptive laws for online updating the network parameters. Suitable learning rates for adaptive laws can be achieved by employing the modified Jaya algorithm. Furthermore, the proposed self-organizing algorithm is used to generate new fuzzy rules or delete less-active fuzzy rules. The control system’s exponential stability is analyzed by applying the Lyapunov theory. Finally, numerical simulations on the synchronization of four-dimensional chaotic systems (considering external disturbances and system uncertainties) are conducted to validate the efficiency and performance of the proposed method.

INDEX TERMS Self-organizing algorithm, type-2 fuzzy CMAC, Jaya algorithm, function-link network, asymmetric membership function.

I. INTRODUCTION

During the last decades, the cerebellar-model articulation controller (CMAC) has attracted increasing attention by many researchers working in various fields [1]–[4]. This controller, which was provided by Albus in 1975, is a special type of neural network [5]. It is inspired by the mammalian cerebellum, and it is classified as a non-fully connected perceptron-like associative-memory network with overlapping receptive fields. The CMAC’s characteristic structure is mapping input space to a group of memory locations. Then, the quantized state value can be obtained from the sum of the data stored in the memory cells associated with the hypercubes covering this input [6]. Several researchers have incorporated fuzzy inference into the CMAC to obtain an architecture known as fuzzy CMAC (FCMAC) that incorporates association layers as fuzzy-membership function layers, and each associative cell performs a fuzzy rule [7]–[9]. However, the FCMAC with type-1 fuzzy-membership functions (T1FMFs) is incapable of dealing well with uncertainties [10]. Many studies in the literature have demonstrated that the performance of type-2 fuzzy-logic systems (T2FLSs), which use type-2 fuzzy-membership functions (T2FMFs), is better than that of type-1 fuzzy-logic systems (T1FLSs), particularly in the presence of internal and external uncertainties [11]–[13]. The high-complexity computation of T2FLSs can be reduced by the interval type-2 fuzzy-logic systems (IT2FLSs), proposed by Liang and Mendel in 2010 [14]. Since this first work, the development of IT2FLSs has been significantly extended and widely applied to various research fields [15]–[17].

Symmetric-membership functions in fuzzy structures capable of facilitating the design of adaptation laws have been presented in the literature [18]. Recently, asymmetric-membership functions have been presented in an effort to improve the network-learning ability [19]–[22]. In this
work, a type-2 asymmetric Gaussian-membership function (T2AGMF), which is constituted by two lower Gaussian-membership functions and two upper Gaussian-membership functions, is employed.

A function-link network (FLN) was first introduced by Pao in 1989 [23]. This network is capable of expanding the input information using function expansion. Generally, the FLN is a type of feed-forward neural network, which can be efficiently employed to approximate nonlinear functions with less computational load and rapid convergence speed [24]. The FLN outputs are generated by trigonometric functions and nonlinear combinations of the input variables [25]. In the past, FLN was applied in network design to achieve better control performance [25]–[28].

The network structure of the CMAC and fuzzy controller significantly affects the control performance. A trial-and-error method to obtain a suitable network structure has often been used in the literature. However, this is a time-consuming method, and the network’s performance needs to be improved [29]. Recently, a self-organizing algorithm has been proposed to automatically construct a network’s structure [30]–[32]. In this work, the self-organizing algorithm is applied to obtain a suitable network structure for the designed network.

In the design of many neural networks and neural-fuzzy systems, the determination of suitable learning rates is very important as it significantly affects the system’s performance. In this work, the modified Jaya algorithm is used to optimize the suitable learning rates for the proposed network. Recently, the Jaya algorithm has been used to solve the optimization problem in various research fields [33]–[36]. It is a simple metaheuristic algorithm proposed by Rao in 2016, and it does not require any tuning of the algorithm-specific parameters [37]. In contrast, most of the well-known metaheuristic algorithms, such as the genetic algorithm, particle swarm optimization, gray wolf optimization, and ant lion optimization, often require tuning of their algorithm-specific parameters to improve their performance. In the Jaya algorithm, only two parameters (population size and number of iterations) are required. In this work, a modified Jaya algorithm (MJA) is proposed. Compared with the original Jaya algorithm in [37], the proposed MJA considers the second and the third of the worst and the best solutions. Therefore, the searchability of the proposed MJA can be enhanced.

Chaotic systems are nonlinear dynamic systems, which exhibit highly-complex dynamics, inner randomness, self-similarity, strange attractor, and sensitivity to initial conditions [38]. The synchronization of chaotic systems is a fundamental problem in nonlinear science that has attracted the attention of many researchers in various fields. Many approaches for synchronizing chaotic systems have been proposed in the past, such as the fuzzy controller, neural network controller, robust controller, and CMAC controller [39]–[41]. Synchronization can be defined as the process where two or more chaotic systems are coupled or when a chaotic system drives another chaotic system [42]. According to the study in [43], the sources of uncertainties in fuzzy logic systems can be listed as the unknown nonlinear characteristics of the systems, the noise, the precision, and the environmental conditions of the measurement devices.

Taking into account the above literature review, a self-organizing interval type-2 fuzzy CMAC using a FLN and an asymmetric-membership function (SFIT2FAC) is presented in this work. The proposed synchronizer can be used to synchronize chaotic systems. Compared with existing research work [44], [45], the proposed system exhibits certain advantages. For example, the network structure can be easily designed, the learning rates can be optimized, and the learning capability and network flexibility can be improved. The major contributions of this work can be summarized as follows: (1) Design of a SFIT2FAC synchronizer with adaptive parameters, which can be tuned online; (2) Design of a FLN to update the weight for the SFIT2FAC network; (3) Design of a type-2 asymmetric Gaussian-membership function to improve the network’s learning ability; (4) Design of a self-organizing algorithm to autonomously construct the proposed network structure. (5) Design of a MJA to obtain the suitable learning rates for the SFIT2FAC network.

The organization of this study is presented as follows. Section II presents the problem formulation of chaotic systems. Section III describes the structure of the proposed SFIT2FAC synchronizer. Section IV presents the simulation results of chaotic system synchronization. Section V concludes the work.

**II. PROBLEM FORMULATION**

Consider a four-dimensional Lorenz–Stenflo chaotic system [46], which includes the master and slave systems as follows:

\[ \dot{x}_1(t) = \alpha (x_2(t) - x_1(t)) + \beta x_4 \]
\[ \dot{x}_2(t) = \gamma x_1(t) - x_1(t)x_3(t) - x_2(t) \]
\[ \dot{x}_3(t) = x_1(t)x_2(t) - \lambda x_3(t) \]
\[ \dot{x}_4(t) = -x_1(t) - ax_4, \]

(1)

\[ \dot{y}_1(t) = \alpha (y_2(t) - y_1(t)) + \beta y_4 + \xi_1(t) + \Delta f(y_1) + u_1(t) \]
\[ \dot{y}_2(t) = \gamma y_1(t) - y_1(t)y_3(t) - y_2(t) + \xi_2(t) + \Delta f(y_2) + u_2(t) \]
\[ \dot{y}_3(t) = y_1(t)y_2(t) - \lambda y_3(t) + \xi_3(t) + \Delta f(y_3) + u_3(t) \]
\[ \dot{y}_4(t) = -y_1(t) - ay_4 + \xi_4(t) + \Delta f(y_4) + u_4(t), \]

(2)

where \( x(t) = [x_1(t), x_2(t), x_3(t), x_4(t)] \) is the master chaotic state vector; \( y(t) = [y_1(t), y_2(t), y_3(t), y_4(t)] \) is the slave chaotic state vector; \( u(t) = [u_1(t), u_2(t), u_3(t), u_4(t)] \) is the synchronization control vector; \( \xi(t) = [\xi_1(t), \xi_2(t), \xi_3(t), \xi_4(t)] \) is the external disturbances vector; \( \Delta f(y(t)) = [\Delta f(y_1), \Delta f(y_2), \Delta f(y_3), \Delta f(y_4)] \) is the uncertainties vector; \( \alpha, \beta, \gamma, \lambda \) denote the parameters for defining the chaotic attractor.
The synchronization error vector $\mathbf{e}(t) = [e_1(t), e_2(t), e_3(t), e_4(t)]$ can be obtained as follows:

$$
\begin{align*}
e_1(t) &= y_1(t) - x_1(t) \\
e_2(t) &= y_2(t) - x_2(t) \\
e_3(t) &= y_3(t) - x_3(t) \\
e_4(t) &= y_4(t) - x_4(t).
\end{align*}
$$

Applying (1) and (2) into (3), the dynamic-error vector $\dot{\mathbf{e}}(t) = [\dot{e}_1(t), \dot{e}_2(t), \dot{e}_3(t), \dot{e}_4(t)]$ is obtained as follows:

$$
\begin{align*}
\dot{e}_1(t) &= \alpha (e_2(t) - e_1(t)) + \beta e_4(t) + \xi_1(t) + \Delta f(y_1(t)) + u_1(t) \\
\dot{e}_2(t) &= \gamma e_1(t) - e_2(t) - y_1(t)y_3(t) + x_1(t)x_3(t) + \xi_2(t) + \Delta f(y_2(t)) + u_2(t) \\
\dot{e}_3(t) &= y_1(t)y_2(t) - x_1(t)x_2(t) - \lambda e_3(t) + \xi_3(t) + \Delta f(y_3(t)) + u_3(t) \\
\dot{e}_4(t) &= -e_1(t) - \alpha e_4 + \xi_4(t) + \Delta f(y_4(t)) + u_4(t).
\end{align*}
$$

Eq. (4) can be formulated in the state space as follows:

$$
\dot{\mathbf{e}}(t) = \mathbf{F}\mathbf{e}(t) + \mathbf{G}\mathbf{y}(t) + \mathbf{u}(t),
$$

where

$$
\mathbf{F} = \begin{bmatrix}
-\alpha & \alpha & 0 & \beta \\
(\gamma - y_3(t)) & -1 & -x_1(t) & 0 \\
x_2(t) & y_1(t) & -\lambda & 0 \\
-1 & 0 & 0 & -\alpha
\end{bmatrix}.
$$

In (5), if $\mathbf{F}(t)$ and $\Delta f(y(t))$ are precisely known, the ideal synchronizer control vector can be determined as follows:

$$
\mathbf{u}^*(t) = -\mathbf{F}\mathbf{e}(t) - \mathbf{G}\mathbf{y}(t) - \mathbf{\zeta}(t) - \Delta f(y(t)),
$$

where $\dot{\mathbf{e}}(t) = -\mathbf{G}\mathbf{y}(t)$ and $\mathbf{G} = \text{diag}(g_1, g_2, g_3, g_4)$ is the feedback gain vector.

If the terms $\mathbf{F}(t)$ and $\Delta f(y(t))$ are precisely known, $\mathbf{u}^*(t)$ in (6) can be obtained. However, in practical applications, $\mathbf{F}(t)$ and $\Delta f(y(t))$ cannot be precisely known. Therefore, the proposed SFIT2FAC synchronizer, which does not request $\mathbf{F}(t)$ and $\Delta f(y(t))$ to be known quantities, is designed in the following section to estimate the $\mathbf{u}^*(t)$.

### III. CONTROLLER DESIGN

#### A. THE STRUCTURE OF THE SFIT2FAC

The $p^{th}$ fuzzy inference rule for the SFIT2FAC synchronizer can be described as follows:

$$
\text{Rule } p^{th}: \text{IF } i_1 \text{ is } \tilde{x}_{1jk} \text{ and } i_2 \text{ is } \tilde{x}_{2jk}, \ldots, \text{ and } i_n \text{ is } \tilde{x}_{njk} \text{ THEN } \tilde{\psi}_{jk} = \begin{bmatrix} \psi^q_{jk} & \tilde{\psi}^q_{jk} \end{bmatrix},
$$

for $i = 1, 2, \ldots, n_i; \ j = 1, 2, \ldots, n_j; \ k = 1, 2, \ldots, n_k; \ q = 1, 2, \ldots, n_q.$ \hfill (7)

where $n_i$ and $n_q$ are the input and output dimensions, respectively; $n_j$ is the number of layers for each input dimension; $n_k$ is the number of blocks for each layer; $\tilde{x}_{ijk}$ and $\tilde{\psi}_{jk}$ denote the input and output membership functions, respectively.

The architecture of the proposed SFIT2FAC synchronizer is presented in Fig. 1. The synchronizer consists of five spaces as follows:

1) **Input space:** This space is used to obtain the input vector $\mathbf{i} = [i_1, i_2, \ldots, i_{n_i}]$. Each node in this space corresponds to each input.

2) **Association memory space:** This space is used to compute the membership grade. Each node in this space includes a type-2 asymmetric membership function (T2AMF), which
is presented in Fig. 2. This function is given as follows:

\[
\bar{\chi}_{ijk} = \begin{cases} 
\exp \left\{ \frac{-(i_j - \bar{\rho}_{ijk})^2}{2 (\bar{\sigma}_{ijk})^2} \right\}, & i_j \leq \bar{\rho}_{ijk} \\
1, & \bar{\rho}_{ijk} \leq i_j \leq \bar{\rho}_{ijk} \\
\exp \left\{ \frac{-(i_j - \bar{\rho}_{ijk})^2}{2 (\bar{\sigma}_{ijk})^2} \right\}, & \bar{\rho}_{ijk} \leq i_j,
\end{cases} \tag{8}
\]

\[
\underline{\chi}_{ijk} = \begin{cases} 
\sigma \times \exp \left\{ \frac{-(i_j - \underline{\rho}_{ijk})^2}{2 (\underline{\sigma}_{ijk})^2} \right\}, & i_j \leq \underline{\rho}_{ijk} \\
\sigma, & \underline{\rho}_{ijk} \leq i_j \leq \underline{\rho}_{ijk} \\
\sigma \times \exp \left\{ \frac{-(i_j - \underline{\rho}_{ijk})^2}{2 (\underline{\sigma}_{ijk})^2} \right\}, & \underline{\rho}_{ijk} \leq i_j,
\end{cases} \tag{9}
\]

where \( \bar{\chi}_{ijk} \) and \( \underline{\chi}_{ijk} \) are the upper and lower membership functions; \( \bar{\rho}_{ijk}, \bar{\sigma}_{ijk} \) and \( \underline{\rho}_{ijk}, \underline{\sigma}_{ijk} \) are the means and variances of the two upper Gaussian membership functions, respectively; \( \bar{\rho}'_{ijk}, \bar{\sigma}'_{ijk} \) and \( \underline{\rho}'_{ijk}, \underline{\sigma}'_{ijk} \) are the means and variances of the two lower Gaussian membership functions, respectively.

The following condition is applied to ensure a reasonable T2AMF:

\[
\begin{align*}
\bar{\rho}_{ijk} &\leq \bar{\rho}'_{ijk} \leq \rho_{ijk} \\
\bar{\sigma}_{ijk} &\leq \bar{\sigma}'_{ijk} \\
0.5 &\leq \sigma \leq 1.
\end{align*} \tag{10}
\]

3) Receptive-field space: This space is used to compute the multidimensional receptive-field function, which is defined as follows:

\[
\bar{\tau}_{jk} = \prod_{i=1}^{n_j} \bar{\chi}_{ijk} \quad \text{and} \quad \underline{\tau}_{jk} = \prod_{i=1}^{n_j} \underline{\chi}_{ijk}, \tag{11}
\]

where \( \bar{\tau}_{jk} \) and \( \underline{\tau}_{jk} \) are the upper and lower firing strengths, which are associated with the \( j^{th} \) layer and \( k^{th} \) block.

This space can be expressed in vector form as follows:

\[
\bar{\tau} = \left[ \bar{\tau}_{11}, \ldots, \bar{\tau}_{1n_k}, \bar{\tau}_{21}, \ldots, \bar{\tau}_{2n_k}, \ldots, \bar{\tau}_{n_j1}, \ldots, \bar{\tau}_{njn_k} \right]^T \in \mathbb{R}^{n_jn_k}
\]

\[
\underline{\tau} = \left[ \underline{\tau}_{11}, \ldots, \underline{\tau}_{1n_k}, \underline{\tau}_{21}, \ldots, \underline{\tau}_{2n_k}, \ldots, \underline{\tau}_{nj1}, \ldots, \underline{\tau}_{njn_k} \right]^T \in \mathbb{R}^{n_jn_k}, \tag{12}
\]

where \( n_j = n_jn_k \) is the number of fuzzy rules.

4) Weight memory space: This space is used to compute the fuzzy memory weights, which are used to connect the receptive-field space with the output space.

Initially, the FLN with a trigonometric function is applied to expand the inputs, as shown in Fig. 3. The function expansion vector is defined as follows:

\[
\varphi = \left[ i_1, \sin (\pi i_1), \ldots, \sin (\pi i_{n_j}), \cos (\pi i_1), \ldots, \cos (\pi i_{n_j}) \right]^T
\]

\[
= \left[ \varphi_1, \varphi_2, \ldots, \varphi_{n_k} \right]^T \in \mathbb{R}^{n_k}, \tag{13}
\]

where \( n_k \) is the number of function expansion outputs.

Then, the fuzzy memory weights for the \( q^{th} \) output are given as follows:

\[
\tilde{\theta}^q = \begin{bmatrix} \tilde{\theta}^q_{11} & \ldots & \tilde{\theta}^q_{1n_k} \\ \tilde{\theta}^q_{21} & \ldots & \tilde{\theta}^q_{2n_k} \\ \vdots & \ddots & \vdots \\ \tilde{\theta}^q_{n_j1} & \ldots & \tilde{\theta}^q_{njn_k} \end{bmatrix} = \begin{bmatrix} \bar{\varphi}_1 \\ \bar{\varphi}_2 \\ \vdots \\ \bar{\varphi}_{n_k} \end{bmatrix} \tag{15}
\]
T.-L. Le et al.: Optimum Design of Function-Link Type-2 Fuzzy Asymmetric CMAC

B. PARAMETER LEARNING ALGORITHM

Assume that an optimal controller $u^*_{\text{SFIT2FAC}}$, which is an approach to the ideal controller $u^*(t)$ in (6), exists, such that

$$u^*(t) = u^*_{\text{SFIT2FAC}}(\hat{\theta}^s, \hat{\theta}^l, \hat{\rho}^l, \hat{\rho}^r, \hat{\nu}^l, \hat{\nu}^r) + \epsilon(t). \quad (19)$$

where $\hat{\theta}^s, \hat{\theta}^l, \hat{\rho}^l, \hat{\rho}^r, \hat{\nu}^l, \hat{\nu}^r$ are the estimation parameters for $\theta^s, \theta^l, \rho^l, \rho^r, \nu^l, \nu^r$; $\epsilon(t)$ is the approximation error.

Since the optimal controller $u^*_{\text{SFIT2FAC}}$ described in (19) cannot be directly obtained, the estimation controller $\hat{u}_{\text{SFIT2FAC}}$ described in (18) and the compensator controller can be used to estimate it. Then the estimation control signal can be expressed as follows:

$$\hat{u}(t) = \hat{u}_{\text{SFIT2FAC}}(\hat{\theta}, \hat{\rho}^l, \hat{\nu}^l, \hat{\rho}^r, \hat{\nu}^r) + \hat{u}_{\text{RB}}(t), \quad (20)$$

where $\hat{\theta}, \hat{\rho}^l, \hat{\nu}^l, \hat{\rho}^r, \hat{\nu}^r$ are the estimation of $\theta^s, \theta^l, \rho^l, \rho^r, \nu^l, \nu^r$; $\hat{u}_{\text{RB}}$ is the robust compensator controller, which is defined as

$$\hat{u}_{\text{RB}}(t) = \Omega(t)\text{sgn}(s(t)) \quad \text{for} \quad \hat{\Omega}(t) = \eta_2 |s(t)|, \quad (21)$$

where $\hat{\Omega}(t)$ is the estimated value of $\Omega(t)$; $\Omega$ is the uncertainty boundary, which is assumed to be an upper limit for the approximation error ($0 \leq \epsilon(t) \leq \Omega$); $s(t)$ denotes the high-order sliding surface, which is given as follows:

$$s(t) = e^{(n-1)} + g_1 e^{(n-2)} + \ldots + g_n \int_0^t e(\tau)d\tau, \quad (22)$$

where $G = [g_n, \ldots, g_2, g_1]^T$ is the positive gain matrix, $g_i = \text{diag}(g_{i1}, g_{i2}, \ldots, g_{in})$ is a positive gain vector; $n$ is the order of the sliding surface.
The sliding surface can be expressed in vector form as \( s(t) = [s_1(t), s_2(t), \ldots, s_q(t)]^T \). Consider a Lyapunov function as follows:

\[
V_1(s(t)) = \frac{1}{2} s^T(t)s(t).
\] (23)

Taking the time-derivative of (22), yields the following:

\[
\dot{s}(t) = e^{(\alpha)} + G^T E,
\] (24)

where \( E = [e_1, e_2, \ldots, e^{(n-1)}] \) is the error matrix; \( e = [e_1, e_2, \ldots, e_n] \) is the synchronization error vector in (3).

Taking the time-derivative of (23) and using (5), (20), and (24), yields the following:

\[
\dot{V}_1(s(t)) = s^T(t) \dot{s}(t) = s^T(t) \left[ e^{(\alpha)} + G^T E \right] = s^T(t) \left[ Fe(t) + \dot{\xi}(t) + \Delta f(y(t)) + \hat{u}_{SFIT2FAC} \right]
\] (25)

Using the chain rule and the gradient descent approach, the online parameter learning algorithms for \( \hat{u}_{SFIT2FAC} \) can be obtained as follows:

\[
\dot{\hat{\eta}}_{lq}^{j} (t + 1) = \dot{\hat{\eta}}_{lq}^{j} (t) - \hat{\eta}_{lq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\eta}_{lq}^{j}} + \frac{1}{2} \tilde{\eta}_{lq} s_k(t) \omega \tau \psi_{lq},
\] (26)

\[
\dot{\hat{\eta}}_{hq}^{j} (t + 1) = \dot{\hat{\eta}}_{hq}^{j} (t) - \hat{\eta}_{hq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\eta}_{hq}^{j}} + \frac{1}{2} \tilde{\eta}_{hq} s_k(t) (1 - \omega) \tau \psi_{hq},
\] (27)

\[
\dot{\hat{\eta}}_{lq}^{j} (t + 1) = \dot{\hat{\eta}}_{lq}^{j} (t) - \hat{\eta}_{lq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\eta}_{lq}^{j}} + \frac{1}{2} \tilde{\eta}_{lq} s_k(t) \omega \tau \psi_{lq},
\] (28)

\[
\dot{\hat{\eta}}_{hq}^{j} (t + 1) = \dot{\hat{\eta}}_{hq}^{j} (t) - \hat{\eta}_{hq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\eta}_{hq}^{j}} + \frac{1}{2} \tilde{\eta}_{hq} s_k(t) (1 - \omega) \tau \psi_{hq},
\] (29)

\[
\dot{\hat{\rho}}_{lq}^{j} (t + 1) = \dot{\hat{\rho}}_{lq}^{j} (t) - \hat{\rho}_{lq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\rho}_{lq}^{j}} + \frac{1}{2} \tilde{\rho}_{lq} s_k(t) \omega \psi_{lq} \tau \psi_{lq},
\] (30)

\[
\dot{\hat{\rho}}_{hq}^{j} (t + 1) = \dot{\hat{\rho}}_{hq}^{j} (t) - \hat{\rho}_{hq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\rho}_{hq}^{j}} + \frac{1}{2} \tilde{\rho}_{hq} s_k(t) (1 - \omega) \psi_{hq} \tau \psi_{hq},
\] (31)

\[
\dot{\hat{\gamma}}_{lq}^{j} (t + 1) = \dot{\hat{\gamma}}_{lq}^{j} (t) - \hat{\gamma}_{lq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\gamma}_{lq}^{j}} + \frac{1}{2} \tilde{\gamma}_{lq} s_k(t) \omega \tau \chi_{lq},
\] (32)

\[
\dot{\hat{\gamma}}_{hq}^{j} (t + 1) = \dot{\hat{\gamma}}_{hq}^{j} (t) - \hat{\gamma}_{hq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\gamma}_{hq}^{j}} + \frac{1}{2} \tilde{\gamma}_{hq} s_k(t) (1 - \omega) \tau \chi_{hq},
\] (33)

\[
\dot{\hat{\beta}}_{lq}^{j} (t + 1) = \dot{\hat{\beta}}_{lq}^{j} (t) - \hat{\beta}_{lq} \frac{\partial \hat{V}_1(t)}{\partial \hat{\beta}_{lq}^{j}} + \frac{1}{2} \tilde{\beta}_{lq} s_k(t) \omega \tau \chi_{lq},
\] (34)
TABLE 1. The derivative terms for updating the lower Gaussian parameters.

| Region I         | Region II          | Region III         |
|------------------|--------------------|--------------------|
| $i_i \leq \hat{\rho}_g^{l}$ | $\hat{\rho}_g^{l} \leq i_i \leq \hat{\rho}_g^{r}$ | $\hat{\rho}_g^{r} \leq i_i$ |
| \[
\frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\rho}_g^{l}} = \frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\theta}_g^{l}} \left( \frac{i_i - \hat{\rho}_g^{l}}{\hat{\theta}_g^{l}} \right)^2 \cdot \frac{\partial \hat{\theta}_g^{l}}{\partial \hat{\rho}_g^{l}} = 0 ;
\]
| \[
\frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\theta}_g^{l}} = \frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\rho}_g^{l}} = 0 ;
\]
| \[
\frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\rho}_g^{l}} = \frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\theta}_g^{l}} = \frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\rho}_g^{l}} = 0 ;
\]
| \[
\frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\theta}_g^{l}} = \frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\rho}_g^{l}} = \frac{\partial \hat{\chi}_{g}^{l}}{\partial \hat{\rho}_g^{l}} = 0 ;
\]

TABLE 2. The derivative terms for updating the upper Gaussian parameters.

| Region I         | Region II          | Region III         |
|------------------|--------------------|--------------------|
| $i_i \leq \hat{\rho}_g^{r}$ | $\hat{\rho}_g^{r} \leq i_i \leq \hat{\rho}_g^{r}$ | $\hat{\rho}_g^{r} \leq i_i$ |
| \[
\frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\rho}_g^{r}} = \frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\theta}_g^{r}} \left( \frac{i_i - \hat{\rho}_g^{r}}{\hat{\theta}_g^{r}} \right)^2 \cdot \frac{\partial \hat{\theta}_g^{r}}{\partial \hat{\rho}_g^{r}} = 0 ;
\]
| \[
\frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\theta}_g^{r}} = \frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\rho}_g^{r}} = 0 ;
\]
| \[
\frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\rho}_g^{r}} = \frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\theta}_g^{r}} = \frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\rho}_g^{r}} = 0 ;
\]
| \[
\frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\theta}_g^{r}} = \frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\rho}_g^{r}} = \frac{\partial \hat{\chi}_{g}^{r}}{\partial \hat{\rho}_g^{r}} = 0 ;
\]

From (26)-(35), yield the following:

\[\Delta \Xi = -\eta \Xi \frac{\partial V_1(t)}{\partial \Xi} = \eta \Xi \frac{\partial V_1(t)}{\partial s} \Xi s(t)F_\Xi(t), \quad (38)\]

where $F_\Xi(t)$ is defined in (38) is defined as follows:

\[F_\Xi(t) = \frac{\partial \Xi}{\partial \Xi} \Xi s(t)F_\Xi(t)\]

(39)

where $F_\Xi(t)$ is defined as follows:

\[F_\Xi(t) = \frac{\partial \Xi}{\partial \Xi} \Xi s(t)F_\Xi(t)\]

(39)

where $F_\Xi(t)$ is defined as follows:

\[F_\Xi(t) = \frac{\partial \Xi}{\partial \Xi} \Xi s(t)F_\Xi(t)\]

(39)

(39)
\( F_{\delta\zeta}(t) = \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}} \)

\[
= \left[ \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}^{i11}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}^{i1n_k}}, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}^{i21}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}^{i2n_k}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}^{in_1}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\zeta}^{in_{n_k}}} \right].
\]

(41)

\( F_{\delta\upsilon}(t) = \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}} \)

\[
= \left[ \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}^{i11}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}^{i1n_k}}, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}^{i21}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}^{i2n_k}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}^{in_1}}, \ldots, \frac{\partial u_{SFIT2FAC}^q}{\partial \delta_{\upsilon}^{in_{n_k}}} \right].
\]

(42)

Applying the chain rule in (37), yields the following:

\( \frac{\partial s(t)}{\partial \xi} = \frac{\partial s(t)}{\partial u_{SFIT2FAC}^q} \cdot \frac{\partial u_{SFIT2FAC}^q}{\partial \xi} = F_{\xi}(t). \)

(49)

Using (38) and (49), (36) becomes as follows:

\[
\Delta V_1(t) = \Delta s(t) \left[ \frac{1}{2} \Delta s(t) + s(t) \right] = \left[ \left( \left[ \frac{\partial s(t)}{\partial \xi} \right]^T \Delta \xi \right) + \frac{1}{2} \left( \left[ \frac{\partial e(t)}{\partial \xi} \right]^T \Delta \xi \right) + s(t) \right] = \left[ F_{\xi}(t) \right]^T \eta \xi s(t) F_{\xi}(t) + \frac{1}{2} \eta \xi^2(s(t)) \| F_{\xi}(t) \|^2 \eta \xi \xi^2(s(t)) + \frac{1}{2} \eta \xi^2(s(t)) \| F_{\xi}(t) \|^2 - 2. \]

(50)

From (50), if \( \eta \xi \) is chosen to satisfy \( 0 < \eta \xi < \frac{2}{\| F_{\xi}(t) \|^2} \), then \( \Delta V_1(t) < 0 \). Consequently, the stability of the proposed SFIT2FAC synchronization system can be ensured.

\section*{C. STRUCTURE LEARNING ALGORITHM}

The proposed methodology for the self-evolution of the SFIT2FAC synchronization structure is described in this section. A flowchart of the self-organizing algorithm is described in Fig. 4.

The generation criterion for adding a new layer and a new T2AGMF is based on the current membership grade in each input as follows:

\[
\chi_{\max}^i < G_r, \quad (51)
\]

\[
\chi_{\max}^i = \max \left( \text{max} \left( \chi_{i11}, \chi_{i12}, \ldots, \chi_{i1n_k}; \chi_{i21}, \chi_{i22}, \ldots, \chi_{i2n_k}; \ldots; \chi_{in_1}, \chi_{in_2}, \ldots, \chi_{in_{n_k}} \right) \right), \quad (52)
\]

\[
\chi_{ijk} = \frac{\chi_{ijk} + \chi_{ijk}}{2}, \quad (53)
\]

where \( G_r \) and \( \chi_{\max}^i \) respectively are the generating threshold and the maximum membership grade in \( i \)th input; \( \chi_{ijk} \) is the average of the upper and lower membership function.
If (51) is satisfied, the new layer is generated, and the initial parameters for new T2AGMF are given as:

\[
\begin{align*}
\rho_{ijk}^l, \ldots, \rho_{ijk}^r, \bar{\rho}_{ijk}^r \\
= [\bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j, \bar{\mu}_j], \\
\end{align*}
\]

where \( k \) and \( \theta \) are the adjusting parameters of T2AGMF; \( \mu_{init} \) is the initial variance.

The pruning criterion for deleting an inappropriate layer is based on the current membership grade in each input as:

\[
\begin{align*}
\chi_{min}^{\prime} &< G_d, \\
\chi_{max}^{\prime} &< G_d, \\
\end{align*}
\]

where \( G_d \) and \( \chi_{min}^{\prime} \) respectively are the pruning threshold and the minimum membership grade in \( i \)th input.

If (56) is satisfied, the inappropriate layer and its corresponding T2AGMF will be deleted.

### D. MODIFIED JAYA ALGORITHM

The selection of the learning rates, \( \hat{\eta}_{\theta}, \hat{\eta}_{\rho}, \hat{\eta}_{\psi} \), for the proposed SFIT2FAC synchronizer significantly affects the system’s performance. If too large or too small learning rates are selected, the gradient-descent algorithm may be trapped into a local minimum or even not converge. In this section, an MJA is proposed for optimizing these learning rates. The optimal mechanism of the Jaya algorithm is to approach the best solution and avoid the worst solution. In this algorithm, the best and worst solutions are calculated in each iteration based on the defined fitness function. The Jaya algorithm is a simple optimization algorithm, which does not require the tuning of any algorithm-specific parameters. The parameters required by the Jaya algorithm are only the population size and the number of iterations. The performance of the original Jaya algorithm can be improved by the proposed MJA, which provides a new updating equation. This equation also considers the second and third of the worst and best solutions, as shown in (59). A flowchart of the proposed MJA is presented in Fig. 5.

The fitness function can be selected as follows:

\[
f_n = (e_1(t))^2 + (e_2(t))^2 + (e_3(t))^2 + (e_4(t))^2.
\]

The learning rates are updated as follows:

\[
\eta_{\hat{\psi}}(t + 1) = \eta_{\hat{\psi}}(t) + r_1 \left[ \eta_{\hat{\psi}}^{best}(t) - \eta_{\hat{\psi}}^p(t) \right] - r_2 \left[ \eta_{\hat{\psi}}^{worst}(t) - \eta_{\hat{\psi}}^p(t) \right],
\]

where \( \Theta \) is replaced by \( \theta, \rho, \) and \( \psi; \eta_{\hat{\psi}}^{best}(t) \) and \( \eta_{\hat{\psi}}^{worst}(t + 1) \) are the current solution and the updated solution, respectively; \( r_1 \) and \( r_2 \) are two random numbers in the range of [0, 1]; \( \eta_{\hat{\psi}}^{best}(1), \eta_{\hat{\psi}}^{best}(2), \) and \( \eta_{\hat{\psi}}^{best}(3) \) are the solution learning rates with the lowest, the second-lowest and the third-lowest fitness function values, respectively; \( \eta_{\hat{\psi}}^{worst}(1), \eta_{\hat{\psi}}^{worst}(2), \) and \( \eta_{\hat{\psi}}^{worst}(3) \) are the solution learning rates with the highest, the second-highest and the third-highest fitness function values, respectively.

**Pseudocode for the MJA**

**Input:** Population size: \( n_p \); Number of iterations: \( n_g \);

**Output:** Learning rates \( \hat{\eta}_{\theta}, \hat{\eta}_{\rho}, \hat{\eta}_{\psi} \);

for \( p := 1 \) to \( n_p \) do

    Initialize \( \hat{\eta}_{\theta}^\ast, \hat{\eta}_{\rho}^\ast, \hat{\eta}_{\psi}^\ast \);

end

repeat

for \( g := 1 \) to \( n_g \) do

    for \( p := 1 \) to \( n_p \) do

        Run simulation with \( \hat{\eta}_{\theta}^p, \hat{\eta}_{\rho}^p, \hat{\eta}_{\psi}^p \);

        Calculate fitness function using Eq. (58)

        Obtain \( \hat{\eta}_{\theta}^{best(id)} = \eta_{\hat{\theta}}^{best(id)} \), \( \hat{\eta}_{\rho}^{best(id)} = \eta_{\hat{\rho}}^{best(id)} \) and \( \hat{\eta}_{\psi}^{best(id)} = \eta_{\hat{\psi}}^{best(id)} \) using Eq. (58)

        end

    for \( p := 1 \) to \( n_p \) do

        Update solutions \( \hat{\eta}_{\theta}^p, \hat{\eta}_{\rho}^p, \hat{\eta}_{\psi}^p \) using Eq. (59)

        Run simulation with updated solutions

        Calculate fitness function using Eq. (58)

    if (updated solutions better than previous) then

        Accept updated solutions;

    else

        Keep previous solutions;

    end

end

until termination criterion satisfied;

**IV. RESULTS AND DISCUSSION**

In this section, the efficiency of the SFIT2FAC is verified by conducting several simulations on the synchronization of
Chaotic systems. A block diagram of the proposed synchronizer’s chaotic systems is shown in Fig. 6. The initial conditions for the master and slave systems are respectively chosen as 
\[ x_1(t), x_2(t), x_3(t), x_4(t) = [0.028, 0.02, 0.03, 0.048] \] and 
\[ y_1(t), y_2(t), y_3(t), y_4(t) = [0.01, 0.037, 0.029, 0.008] \]. The system’s uncertainties and external disturbances are selected as 
\[ 1f(y_1), 1f(y_2), 1f(y_3), 1f(y_4) = r_{rd}[0.1y_1, 0.1y_2, 0.1y_3, 0.1y_4] \] and 
\[ [\xi_1(t), \xi_2(t), \xi_3(t), \xi_4(t)] = [\cos 0.2 \pi t, \cos 0.5 \pi t, 0.3 \cos 0.3 \pi t, 0.4 \cos 0.3 \pi t] \], respectively; where 
\[ r_{rd} \in [0, 1] \] is the random uncertainty value. The root mean square error (RMSE) can be used to evaluate the system’s performance as follows:

\[
RMSE = \sqrt{\frac{1}{n_r} \sum_{r=1}^{n_r} \left( (e'_r)^2 + (e'_2)^2 + (e'_3)^2 + (e'_4)^2 \right)},
\]

where 
\[ n_r \] is the number of samples; 
\[ e'_r, e'_2, e'_3, e'_4 \] are the tracking errors for the \( r^{th} \) sample.

The overall steps required to achieve the synchronization of the proposed chaotic systems are given as follows:

**Step 1:** Optimize the learning rates using the proposed MJA.

**Step 2:** Calculate the synchronization error and its high-order sliding surface using Eqs. (3) and (22).

**Step 3:** Generate the control signal using Eqs. (17), (20), and (21).

**Step 4:** Update network parameters using Eqs. (26)-(35).

**Step 5:** Construct the network using the self-organizing algorithm.

**Step 6:** Repeat Step 1 until the synchronization time is expired.

**Case 1:** In this case, the parameters for defining chaotic system’s attractor are defined as 
\[ \alpha = 0.5, \beta = 20, \gamma = 30, \lambda = 8/3 \]. The chaotic attractor of the four-dimensional Lorenz–Stenflo chaotic system is presented in Fig. 7 in a three-dimensional view. The state trajectories of the chaotic systems’ synchronization using the proposed SFIT2FAC synchronizer are shown in Fig. 8. The time history of the control signal and the tracking errors are shown in Figs. 9 and 10, respectively. From Fig. 10, it can be observed that the proposed SFIT2FAC synchronizer is capable of synchronizing the four-dimensional Lorenz–Stenflo chaotic system well, even when the system uncertainties and external disturbances are considered. The change in the number of layers using the proposed self-organizing algorithm is shown.
in Fig. 11, whereas the change in the learning rates using the proposed MJA is shown in Fig. 12.

**Case 2:** In this case, the parameters for defining the chaotic attractor of the chaotic system are taken as $\alpha = 10$, $\beta = 30$, $\gamma = 40$, $\lambda = 8/3$. The chaotic attractor of the four-dimensional Lorenz–Stenflo chaotic system is presented in Fig. 13 in a three-dimensional view. The state trajectories of the chaotic systems’ synchronization using the proposed SFIT2FAC synchronizer are shown in Fig. 14. The time history of the control signal and the tracking errors are shown in Figs. 15 and 16, respectively. From Fig. 16, it can be observed that the proposed SFIT2FAC synchronizer is capable of synchronizing the four-dimensional Lorenz–Stenflo chaotic system well, even when the system uncertainties and external disturbances are considered. The change in the number of layers using the proposed self-organizing algorithm is shown in Fig. 17, whereas the change in the learning rates using the proposed MJA is shown in Fig. 18.

In both cases, the proposed SFIT2FAC synchronizer exhibits synchronization efficiency in the chaotic system. Figures. 8 and 14 confirm that the proposed synchronizer is capable of synchronizing well the chaotic system, even when the system’s uncertainties and external disturbances are considered. Figs. 9 and 15 show that the control signals promptly respond to adapt to the changes that occur during the synchronization process. Figs. 10 and 16...
show that tracking errors quickly converge to small values. Figs. 11 and 17 indicate that using the proposed self-organizing algorithm, the number of layers promptly changes to adapt to changes in the input signals. Figs. 12 and 18 show that using the proposed MJA, the learning rates quickly change to achieve the optimal values. Finally, comparison results in the RMSE among the CMAC, the interval type-2 fuzzy neural network (IT2FNN), and the proposed synchronizer with and without the MJA algorithm are shown in Table 3.

Remark 1: By applying the MJA algorithm, the proposed network is capable of quickly obtaining suitable learning rates. Therefore, it is capable of achieving the smallest RMSE values. However, its computation time is somewhat longer than that of other methods.

Remark 2: By applying the self-organizing algorithm, the proposed network is capable of quickly obtaining a suitable network structure. The new fuzzy rules can be generated if the entire current rules cannot cover well the input changes.

Remark 3: The selection of the generating and pruning thresholds for the self-organizing algorithm significantly affects the system’s performance. A high-generating threshold hardly leads the algorithm to generate new rules. In contrast, a low-generating threshold leads to the generation of
huge rules and requires a huge computation time. A high-pruning threshold leads the algorithm to rarely delete the unused rules. In contrast, a low-pruning threshold leads to the deletion of huge rules and may not cover well the input changes. In this work, trial-and-error was used to obtain suitable thresholds. Further studies should investigate methods to optimize the threshold values.

V. CONCLUSION

In this paper, an SFIT2FAC synchronizer was proposed for the synchronization of a four-dimensional Lorenz–Stenflo chaotic system. The adaptive laws were derived to online update the network parameters based on the gradient-descent approach. It was found that the network’s learning ability can be improved by the proposed asymmetric membership function. A FLN was applied to adjust the lower and upper weights of the type-2 asymmetric Gaussian membership function. Then, a self-organizing algorithm was employed to autonomously construct the SFIT2FAC network. Finally, numerical simulation results on the synchronization of four-dimensional chaotic systems were presented to verify the validity of the proposed method.

REFERENCES

[1] P.-Y. Chen, K.-H. Chao, and Y.-C. Tseung, “A motor fault diagnosis system based on cerebellar model articulation controller,” IEEE Access, vol. 7, pp. 120326–120336, 2019.

[2] T.-L. Le, T.-T. Huynh, V.-Q. Nguyen, C.-M. Lin, and S.-K. Hong, “Chaotic synchronization using a self-evolving recurrent interval type-2 Petri cerebellar model articulation controller,” Mathematics, vol. 8, no. 2, p. 219, Feb. 2020.

[3] T.-L. Le, T.-T. Huynh, S.-K. Hong, and C.-M. Lin, “Hybrid neural network cerebellar model articulation controller design for non-linear dynamic time-varying plants,” Frontiers Neurosci., vol. 14, p. 695, Jul. 2020.

[4] C.-M. Lin, V.-H. La, and T.-L. Le, “DC–DC converters design using a type-2 wavelet fuzzy cerebellar model articulation controller,” Neural Comput. Appl., vol. 32, no. 7, pp. 2217–2229, Apr. 2020.

[5] J. S. Albus, “A new approach to manipulator control: The cerebellar model articulation controller (CMAC),” J. Dyn. Syst., Meas., Control, vol. 97, no. 3, pp. 220–227, Sep. 1975.

[6] C. Ching-Tsan and L. Chun-Shin, “CMAC with general basis functions,” Neural Netw., vol. 9, no. 7, pp. 1199–1211, Oct. 1996.

[7] C.-W. Chang, W.-R. Xiao, C.-C. Hsiao, S.-S. Chen, and C.-W. Yao, “A simplified interval type-2 fuzzy CMAC,” in Proc. Joint 17th World Cong. Int. Fuzzy Syst. Assoc. 9th Int. Conf. Soft Comput. Intell. Syst. (IFSA-SCIS), Jun. 2017, pp. 1–4.

[8] H. Wu, H. An, Q. Wei, and H. Ma, “Fuzzy CMAC-based adaptive scale force control of body weight support exoskeletons,” IEEE Access, vol. 8, pp. 147286–147294, 2020.

[9] C.-M. Lin and M.-S. Yang, “Type-2 fuzzy cerebellar model articulation control system design for MIMO uncertain nonlinear systems,” Int. J. Mach. Learn. Cybern., vol. 11, no. 2, pp. 269–286, Feb. 2020.

[10] T.-L. Le, T.-T. Huynh, and S. K. Hong, “A modified grey wolf optimizer for optimum parameters of multilayer Type-2 asymmetric fuzzy controller,” IEEE Access, vol. 8, pp. 121611–121629, 2020.

[11] O. Castillo, R. Martínez-Marroquín, P. Melin, F. Valdez, and 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., vol. 192, pp. 19–38, Jun. 2012.

[12] J. M. Mendel, “A quantitative comparison of interval type-2 and type-1 fuzzy logic systems: First results,” in Proc. Int. Conf. Fuzzy Syst., Jul. 2010, pp. 1–8.

[13] S.-K. Oh, H.-J. Jang, and 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., vol. 38, no. 9, pp. 11217–11229, Sep. 2011.

[14] Q.-L. Jiang and J. M. Mendel, “Interval type-2 fuzzy logic systems: Theory and design,” IEEE Trans. Fuzzy Syst., vol. 8, no. 5, pp. 535–550, Oct. 2000.

[15] R. Kavikumar, R. Saktivheil, O.-M. Kwon, and B. Kaviarasan, “Robust model reference tracking control for interval type-2 fuzzy stochastic systems,” IET Control Theory Appl., vol. 14, no. 9, pp. 1123–1134, Jun. 2020.

[16] R. Kavikumar, R. Saktivheil, O.-M. Kwon, and B. Kaviarasan, “Finite-time boundedness of interval type-2 fuzzy systems with time delay and actuator faults,” J. Franklin Inst., vol. 356, no. 15, pp. 8296–8324, Oct. 2019.

[17] T.-L. Le, “Intelligent fuzzy controller design for antilock braking systems,” Int. J. Fuzzy Syst., vol. 36, no. 4, pp. 3303–3315, Apr. 2019.

[18] T. Zhao, P. Li, and J. Cao, “Self-organising interval type-2 fuzzy neural network with asymmetric membership functions and its application,” Soft Comput., vol. 23, no. 16, pp. 7215–7228, Aug. 2019.

[19] S.-Y. Chen and T.-S. Liu, “Intelligent tracking control of a PMLSM using self-evolving probabilistic fuzzy neural network,” IET Electric Power Appl., vol. 11, no. 6, pp. 1043–1054, Jul. 2017.

[20] F.-J. Lin, K.-H. Tan, Y.-K. Lai, and W.-C. Luo, “Intelligent PV power system with unbalanced current compensation using CFNN-AMF,” IEEE Trans. Power Electron., vol. 34, no. 9, pp. 8588–8598, Sep. 2019.

[21] K.-H. Tan, F.-J. Lin, and J.-H. Chen, “DC-link voltage regulation using RPFPNN-AMF for three-phase active power filter,” IEEE Access, vol. 6, pp. 37454–37463, Jun. 2018.

[22] T.-L. Le, T.-T. Huynh, and S.-K. Hong, “Self-organizing interval type-2 fuzzy asymmetric CMAC design to synchronize chaotic satellite systems using a modified grey wolf optimizer,” IEEE Access, vol. 8, pp. 53697–53709, 2020.

[23] Y.-H. Pao, “Functional link nets: Removing hidden layers,” AI Expert, vol. 4, no. 4, pp. 60–68, 1989.

[24] F.-J. Lin, L.-T. Teng, J.-W. Lin, and S.-Y. Chen, “Recurrent functional-link-based fuzzy-neural-network-controlled inductive-generator system using improved particle swarm optimization,” IEEE Trans. Ind. Electron., vol. 56, no. 5, pp. 1557–1577, May 2009.

[25] C.-M. Lin, T.-L. Le, and T.-T. Huynh, “Self-evolving function-link interval type-2 fuzzy neural network for nonlinear system identification and control,” Neurocomputing, vol. 275, pp. 2239–2250, Jan. 2018.

[26] S. Scardapane, D. Wang, and A. Uncini, “Bayesian random vector functional-link networks for robust data modeling,” IEEE Trans. Cybern., vol. 48, no. 7, pp. 2049–2059, Jul. 2018.
[27] E. Lotfi and A. A. Rezaee, “A competitive functional link artificial neural network as a universal approximator,” Soft Comput., vol. 22, no. 14, pp. 4613–4625, Jul. 2018.

[28] T.-T. Huynh, C.-M. Lin, T.-T. Pham, H.-Y. Cho, and T.-L. Le, “A modified function-link fuzzy cerebellar model articulation controller using a PI-type learning algorithm for nonlinear system synchronization and control,” Chaos, Solitons Fractals, vol. 118, pp. 65–82, Jan. 2019.

[29] T.-L. Le, C.-M. Lin, and T.-T. Huynh, “Self-evolving type-2 fuzzy brain emotional learning control design for chaotic systems using PSO,” Appl. Soft Comput., vol. 73, pp. 418–433, Dec. 2018.

[30] H.-G. Han, X.-L. Wu, Z. Liu, and J.-F. Qiao, “Design of self-organizing intelligent controller using fuzzy neural network,” IEEE Trans. Fuzzy Syst., vol. 26, no. 5, pp. 3097–3111, Oct. 2018.

[31] T.-T. Le, “Self-organizing recurrent interval Type-2 Petri fuzzy design for time-varying delay systems,” IEEE Access, vol. 7, pp. 10505–10514, 2019.

[32] T.-T. Huynh, T.-L. Le, and C.-M. Lin, “Self-organizing recurrent wavelet fuzzy neural network-based control system design for MIMO uncertain nonlinear systems using TOPSIS method,” Int. J. Fuzzy Syst., vol. 21, no. 2, pp. 468–487, Mar. 2019.

[33] R. V. Rao, A. Saroj, P. Oclon, J. Taler, and J. Lakshmi, “A posteriori multi-objective self-adaptive multipopulation jaya algorithm for optimization of thermal devices and cycles,” IEEE Access, vol. 7, pp. 4113–4134, 2019.

[34] K. K. Ingle and D. R. K. Jatoth, “An efficient JAYA algorithm with Lévy flight for non-linear channel equalization,” Expert Syst. Appl., vol. 145, May 2020, Art. no. 112970.

[35] J.-T. Yu, C.-H. Kim, A. Wadood, T. Kharshaid, and S.-B. Rhee, “Jaya algorithm with self-adaptive multi-population and Lévy flights for solving economic load dispatch problems,” IEEE Access, vol. 7, pp. 21372–21384, 2019.

[36] C. Chen, D. Zou, and C. Li, “Improved jaya algorithm for economic dispatch considering valve-point effect and multi-fuel options,” IEEE Access, vol. 8, pp. 84981–84995, 2020.

[37] R. Venkata Rao, “Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems,” Int. J. Ind. Eng. Comput., vol. 7, no. 1, pp. 19–34, 2016.

[38] S. H. Hosseini, A. Marandi, A. Ranjbar N., M. Mahmoudian, and S. Momani, “Sliding mode synchronization of an uncertain fractional order chaotic system,” Comput. Math. Appl., vol. 59, no. 10, pp. 1637–1643, May 2010.

[39] S. Shao, M. Chen, and X. Yan, “Prescribed performance synchronization for uncertain chaotic systems with input saturation based on neural networks,” Neural Comput. Appl., vol. 29, no. 12, pp. 1349–1361, Jun. 2018.

[40] Z. Xu, D. Peng, and X. Li, “Synchronization of chaotic neural networks with time delay via distributed delayed impulsive control,” Neural Netw., vol. 118, pp. 332–337, Oct. 2019.

[41] K. Tian, H.-P. Ren, and C. Bai, “Synchronization of hyperchaos with time delay using impulse control,” IEEE Access, vol. 8, pp. 72570–72576, 2020.

[42] E.-J. Hwang, C.-H. Hyun, E. Kim, and M. Park, “Fuzzy model based adaptive synchronization of uncertain chaotic systems: Robust tracking control approach,” Phys. Lett. A, vol. 373, no. 22, pp. 1935–1939, May 2009.

[43] M. A. Khanesar, E. Kayacan, M. Teshnehlab, and O. Kaynak, “Analysis of the noise reduction property of Type-2 fuzzy logic systems using a novel Type-2 membership function,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 41, no. 5, pp. 1395–1406, Oct. 2011.

[44] A. Mohammadzadeh and S. Ghaemi, “Optimal synchronization of fractional-order chaotic systems subject to unknown fractional order, input nonlinearities and uncertain dynamic using type-2 fuzzy CMAC,” Nonlinear Dyn., vol. 88, no. 4, pp. 2993–3002, Jun. 2017.

[45] C.-M. Lin, T.-T. Huynh, and T.-L. Le, “Adaptive TOPSIS fuzzy CMAC back-stepping control system design for nonlinear systems,” Soft Comput., vol. 23, no. 16, pp. 6947–6966, Aug. 2019.

[46] U. E. Vincent, “Synchronization of identical and non-identical 4-D chaotic systems using active control,” Chaos, Solitons Fractals, vol. 37, no. 4, pp. 1065–1075, Aug. 2008.