mathematics

MDPI

Article

Chaotic Synchronization Using a Self-Evolving Recurrent Interval Type-2 Petri Cerebellar Model Articulation Controller †

Tien-Loc Le [1,2], Tuan-Tu Huynh [2,3], Vu-Quynh Nguyen [2], Chih-Min Lin [3] and Sung-Kyung Hong [1,*]

[1] Faculty of Mechanical and Aerospace, Sejong University, Seoul 143-747(05006), Korea; tienloc@sju.ac.kr
[2] Department of Electrical Electronic and Mechanical Engineering, Lac Hong University, Bien Hoa 810000, Vietnam; huynhtuantu@saturn.yzu.edu.tw (T.-T.H.); vuquynh@lhu.edu.vn (V.-Q.N.)
[3] Department of Electrical Engineering, Yuan Ze University, Taoyuan 32003, Taiwan; cml@saturn.yzu.edu.tw
* Correspondence: skhong@sju.ac.kr; Tel.: +82-02-3408-3772
† This paper is an extended version of our paper published in 2019 IEEE International Conference on System Science and Engineering (ICSSE 2019), Dong Hoi, Vietnam, 20–21 July 2019; pp. 420–424.

Received: 2 December 2019; Accepted: 5 February 2020; Published: 9 February 2020

Abstract: In this manuscript, the synchronization of four-dimensional (4D) chaotic systems with uncertain parameters using a self-evolving recurrent interval type-2 Petri cerebellar model articulation controller is studied. The design of the synchronization control system is comprised of a recurrent interval type-2 Petri cerebellar model articulation controller and a fuzzy compensation controller. The proposed network structure can automatically generate new rules or delete unnecessary rules based on the self-evolving algorithm. Furthermore, the gradient-descent method is applied to adjust the proposed network parameters. Through Lyapunov stability analysis, bounded system stability is guaranteed. Finally, the effectiveness of the proposed controller is illustrated using numerical simulations of 4D chaotic systems.

Keywords: chaotic systems; self-evolving algorithm; interval type-2 fuzzy system; Petri nets; cerebellar model articulation controller

1. Introduction

Recently, chaotic synchronization has attracted academic attention due to its nonlinear phenomena characteristic. Recent studies have demonstrated that chaotic synchronization can be applied to various disciplines, such as economics and chemistry as well as mechanical systems, information systems, and electronic and communication systems [1]. In recent years, a number of real-life applications have been studied by [2–6]. In 2016, Naderi and Kheiri proposed a secure-communication method using the exponential synchronization of a chaotic system [2]. In 2017, Pappu et al. presented an electronic implementation of Lorenz chaotic-oscillator synchronization for bistatic-radar applications [3]. In 2019, Jayaprasath et al. introduced secure optical communication using chaotic semiconductor lasers [4]. In addition, in 2019, Mandal and Das established chaos-based color image encryption using microcontrollers [5]. In recent decades, various control methods have been reported to synchronize master-slave chaotic systems, such as adaptive control [7], fuzzy control [8], fuzzy-brain emotional-learning networks [9], sliding-mode control [10], and cerebellar model articulation control [11]. However, the majority of these methods are complex, and the controlling performance requires improvement.

The cerebellar model articulation controller (CMAC) is a type of neural network based on a model of the mammalian cerebellum (associative memory), which was proposed by Albus [12]. Compared

with other neural networks, CMAC is advantageous insofar that it has fast learning properties, simple computations, and good generalization capabilities [13]. In the past decade, CMAC has been applied to various fields, such as control systems [14–17], classification systems [18–20], signal processing [21–23], and image processing [24,25]. Due to the work of Zadeh [26], fuzzy modeling and fuzzy control have attracted many researchers since said methods can be used to convert problems into simple human terms. The recent progress of fuzzy-control systems has eventuated in many novel results [27–33]. Similar to the type-1 fuzzy system, the CMAC with type-1 membership functions (T1MFs) cannot effectively deal with the uncertainty associated with system internalities and externalities [34]. To address these uncertainties, type-2 membership functions (T2MFs) were introduced by Zadel [35]. Recent studies have proven the superior effectiveness of T2MFs over T1MFs [36–38]. To reduce the computational complexity of type-2 fuzzy logic systems (T2FLS), interval type-2 fuzzy logic systems (IT2FLS) were established in 2000 by Liang and Mendel [39]. Recently, by combining the advantages of CMAC and IT2FLS, the interval type-2 fuzzy CMAC (IT2CMAC) was developed and applied to various fields [40–43].

Due to the work of Peterson and Looney [44,45], Petri nets (PNs) and fuzzy PNs (FPNs) have been widely investigated in various fields [46–50]. A PN is a directed, weighted, and bipartite graph in which each node is either a place or a transition. The transition nodes are enabled when the value of the inputs connected to a transition that is greater than, or equal to, the threshold value [51]. In 2019, Rosdi et al. proposed the speech intelligibility detection of children using an FPN-based classification method [46]. In 2018, Zhu et al. presented model-based fault identification using PNs [47]. In 2018, Hansen et al. introduced a FPN for soccer-ball recognition and distance prediction [50]. As a special kind of PN, FPNs have some advantages, such as simple in computation, intuitive and easy to understand [46].

The recurrent neural network (RNN) is a special kind of neural network that naturally comprises feedback connections used as internal memories [52]. Many studies have used RNNs in their control network design [53–57] due to their advantages of simple architecture and dynamic characteristics. In 2018, Yen et al. proposed robust adaptive sliding-mode control using recurrent fuzzy wavelet neural networks [54]. In 2016, Lin et al. introduced a piezo-flexural nanopositioning stage using a RNN and intelligent integral backstepping sliding-mode control [55]. In 2016, Sharma et al. presented a robotic manipulator using a RNN and an adaptive controller similar to proportional–integral–derivative controllers [56]. In 2016, Wang et al. proposed a switched-reluctance motor-drive system using adaptive recurrent CMAC [57].

To improve the work of [58], this paper incorporates the advantages of CMAC, IT2FLS, RNN, and FPNs to propose a recurrent interval type-2 Petri cerebellar model articulation controller (RIT2PC). However, similar to other neural networks, it is difficult to determine a suitable network size for the RIT2PC to achieve the desired performance. The majority of studies used the trial-and-error approach to obtain network size, but this method is not time-effective, and its performance requires improvement. In the past, studies have provided self-organizing and self-evolving algorithms to construct network structures autonomously [59–65]. In 2017, Lin et al. introduced a self-evolving function-link interval type-2 fuzzy neural network for nonlinear system identification and control [60]. In addition, in 2017, Rong et al. proposed a self-evolving fuzzy model controller for hypersonic vehicles [63]. In 2018, Ge and Zeng provided a self-evolving fuzzy system that can independently learn dynamic threshold parameters [64]. Besides being able to automatically construct networks to achieve optimal structure, the self-evolving algorithm also has disadvantages; for instance, choosing the threshold to generate and delete rules significantly affects system performance [65]. This study applies a self-evolving algorithm to establish the RIT2PC structure. Thus, the proposed controller has the advantages of the aforementioned networks, but it has a better control performance. The main contributions of this study include the following: successful development of a self-evolving RIT2PC (SRIT2PC) control system; the online learning-parameter adaptation laws are obtained using the gradient-descent method; the Lyapunov stability function is used to prove the stability of the proposed

synchronization system; the effectiveness of the proposed control method is illustrated using numerical experiments of four-dimensional (4D) chaotic systems.

This study is organized as follows: system description is given in Section 2; the architecture of the proposed SRIT2PC is provided in Section 3; the illustrative examples are given in Section 4; finally, conclusions are drawn in Section 5.

2. System Description

Consider the 4D Lorenz–Stenflo chaotic system, which was provided by Stenflo [66] as follows:

$$\begin{aligned} \dot{x}_1(t) &= \alpha(y_1(t) - x_1(t)) + \gamma w_1 \\ \dot{y}_1(t) &= \tau x_1(t) - x_1(t) z_1(t) - \lambda y_1(t) \\ \dot{z}_1(t) &= x_1(t) y_1(t) - \varphi z_1(t) \\ \dot{w}_1(t) &= -x_1(t) - \alpha w_1 \end{aligned} \tag{1}$$

where x_1, y_1, z_1, and w_1 are the master chaotic state variables; α, τ, λ, φ, and γ are the parameters for defining the chaotic attractor:

$$\begin{aligned} \alpha &= (25\theta + 1) \\ \tau &= (26 - 35\theta) \\ \lambda &= (1 - 29\theta) \\ \varphi &= \left(\tfrac{2.1+\theta}{3}\right) \\ \gamma &= (\theta + 1.5) \end{aligned} \tag{2}$$

where θ used to define the feature of the chaotic system.

When the system uncertainties, external disturbances, and control inputs are under consideration, Equation (1) can be rewritten as

$$\begin{aligned} \dot{x}_2(t) &= \alpha(y_2(t) - x_2(t)) + \gamma w_2 + d_x(t) + \Delta f(x_2) + u_x(t) \\ \dot{y}_2(t) &= \tau x_2(t) - x_2(t) z_2(t) - \lambda y_2(t) + d_y(t) + \Delta f(y_2) + u_y(t) \\ \dot{z}_2(t) &= x_2(t) y_2(t) - \varphi z_2(t) + d_z(t) + \Delta f(z_2) + u_z(t) \\ \dot{w}_2(t) &= -x_2(t) - \alpha w_2 + d_w(t) + \Delta f(w_2) + u_w(t) \end{aligned} \tag{3}$$

where, x_2, y_2, z_2, and w_2 are the slave chaotic state variables; $d_x(t)$, $d_y(t)$, $d_z(t)$, and $d_w(t)$ denote the external disturbances; $\Delta f(x_2)$, $\Delta f(y_2)$, $\Delta f(z_2)$, and $\Delta f(w_2)$ denote the system uncertainties; $u_x(t)$, $u_y(t)$, $u_z(t)$, and $u_w(t)$ denote the active control functions. The goal of the control system is to generate the control signal, which can force the slave system, represented by Equation (3), to synchronize with the master system, represented by Equation (1).

The tracking errors of synchronization between Equations (1) and (3) can be defined as

$$\begin{aligned} e_x(t) &= x_2(t) - x_1(t) \\ e_y(t) &= y_2(t) - y_1(t) \\ e_z(t) &= z_2(t) - z_1(t) \\ e_w(t) &= w_2(t) - w_1(t) \end{aligned} \tag{4}$$

Thus, subtracting Equation (3) from Equation (1), yields

$$\begin{aligned} \dot{e}_x(t) &= \alpha\big(e_y(t) - e_x(t)\big) + \gamma e_w + d_x(t) + \Delta f(x_2) + u_x(t) \\ \dot{e}_y(t) &= \tau e_x(t) - \lambda e_y(t) - x_2(t) z_2(t) + x_1(t) z_1(t) + d_y(t) + \Delta f(y_2) + u_y(t) \\ \dot{e}_z(t) &= x_2(t) y_2(t) - x_1(t) y_1(t) - \varphi e_z(t) + d_z(t) + \Delta f(z_2) + u_z(t) \\ \dot{e}_w(t) &= -e_x(t) - \alpha e_w(t) + d_w(t) + \Delta f(w_2) + u_w(t) \end{aligned} \tag{5}$$

Equation (5) can be rewritten as

$$\dot{\boldsymbol{e}}(t) = \boldsymbol{A}\boldsymbol{e}(t) + \boldsymbol{d}(t) + \Delta\boldsymbol{f}(t) + \boldsymbol{u}(t) \tag{6}$$

where $\boldsymbol{e}(t) = \left[e_x(t), e_y(t), e_z(t), e_w(t)\right]^T$; $\boldsymbol{A} = \begin{bmatrix} -\alpha & \alpha & 0 & \gamma \\ (\tau - z_1(t)) & -1 & -x_2(t) & 0 \\ y_1(t) & x_2(t) & -\varphi & 0 \\ -1 & 0 & 0 & -\alpha \end{bmatrix}$

If the system dynamics and the external disturbance can be obtained, the design of the ideal controller can be given by

$$\boldsymbol{u}^*(t) = -\boldsymbol{A}\boldsymbol{e}(t) - \boldsymbol{K}\boldsymbol{e}(t) - \boldsymbol{d}(t) - \Delta\boldsymbol{f}(t) \tag{7}$$

where $\dot{\boldsymbol{e}}(t) = -\boldsymbol{K}\boldsymbol{e}(t)$ and $\boldsymbol{K} = diag(k_1, k_2, k_3, k_4)$ is the feedback gain vector.

If $\boldsymbol{K}$ is selected to correspond to the coefficients of the Hurwitz polynomial, then $\lim_{t\to\infty} e(t) \to 0$. However, the ideal controller, which is represented by Equation (7), is generally unobtainable because the external disturbance and system dynamics cannot be precisely known in practical applications. Therefore, in this paper, an SRIT2PC is proposed to achieve the desired synchronization performance.

3. Architecture of SRIT2PC

The control scheme of the proposed SRIT2PC for the chaotic synchronization system is shown in Figure 1. It consists of an SRIT2PC main controller and a fuzzy compensation controller. The high-order sliding surface is applied to guarantee system stability and to achieve satisfactory control performance.

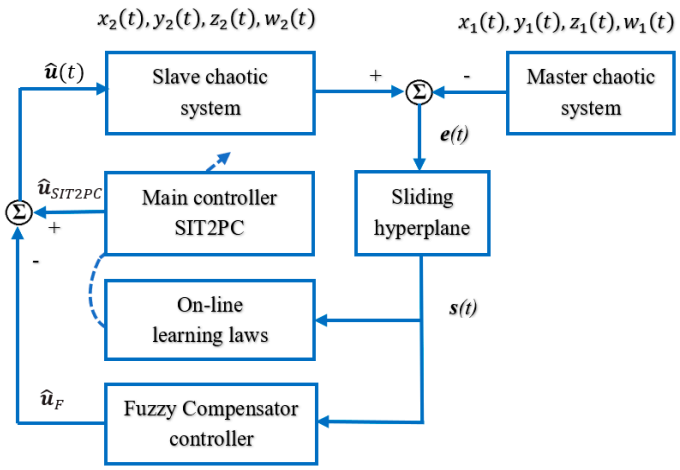

Figure 1. Block diagram of self-evolving recurrent interval type-2 Petri cerebellar model articulation controller (SRIT2PC) synchronization system.

3.1. Recurrent Interval Type-2 Petri CMAC

The fuzzy inference rules of the novel SRIT2PC are given as

$$\begin{aligned} Rule\ \lambda: \quad & IF\ x_1\ is\ \widetilde{\mu}_{1jk}\ and\ x_2\ is\ \widetilde{\mu}_{2jk}, \ldots, and\ x_{n_i}\ is\ \widetilde{\mu}_{n_i jk} \\ & Then\ \widetilde{w}_{jk} = \left[\underset{jk}{w}\ \overline{w}_{jk}\right] \\ & for\ \ i = 1, 2, \ldots, n_i; \quad j = 1, 2, \ldots, n_j; \\ & \quad k = 1, 2, \ldots, n_k; \quad \lambda = 1, 2, \ldots, n_\lambda; \end{aligned} \tag{8}$$

where n_i, n_j and n_k denote the input dimension, the number of layers, and the number of blocks in each layer, respectively; n_λ denotes the total number of fuzzy rules, which is given by $n_\lambda = n_j * n_k$; $\widetilde{\mu}_{ijk}$ denotes the input membership function; $\widetilde{w}_{jk}$ denotes the output weight in the consequent part.

The architecture of the SRIT2PC is composed of seven spaces, shown in Figure 2. The operation in each space is outlined below.

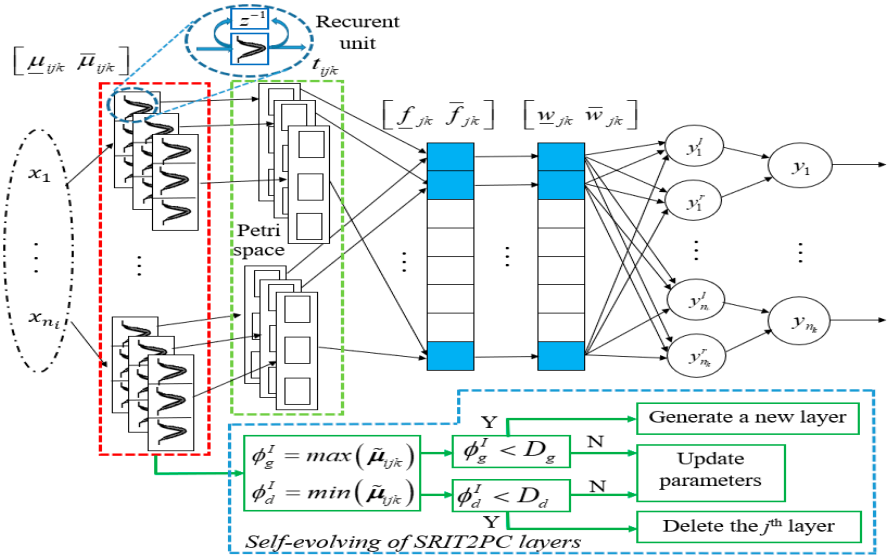

Figure 2. Structure of the SRIT2PC control system.

(1). Input space: The input signal is given as $\boldsymbol{X} = [x_i, x_i, \ldots x_i, \ldots, x_i]^T \epsilon\ \Re^{n_i}$. Herein, each input state variable, x_i, is directly propagated to the association memory space.

(2). Association memory space: Several elements can be accumulated as a block, with each block performing a type-2 Gaussian membership function (T2GMF). Applying the x_i signal from the input space into the T2GMF, the membership grade can be given as

$$\underline{\mu}_{ijk} = \exp\left\{\frac{-\left(I_{ri} - m_{ijk}\right)^2}{2\sigma_{ijk}^2}\right\};\ \overline{\mu}_{ijk} = \exp\left\{\frac{-\left(\overline{I}_{ri} - m_{ijk}\right)^2}{2\overline{\sigma}_{ijk}^2}\right\} \tag{9}$$

$$\underline{I}_{ri}(t) = x_i(t) + r_{ijk}\underline{\mu}_{ijk}(t-1);\ \overline{I}_{ri}(t) = x_i(t) + r_{ijk}\overline{\mu}_{ijk}(t-1) \tag{10}$$

where $\underline{\mu}_{ijk}$ and $\overline{\mu}_{ijk}$ denote the lower and upper membership functions (MFs), respectively; the mean of the T2GMF is denoted by m_{ijk}; $\underline{\sigma}_{ijk}$ and $\overline{\sigma}_{ijk}$ denote the lower and upper variance, respectively; $\underline{I}_{ri}$ and $\overline{I}_{ri}$ denote the lower and upper recurrent inputs, respectively.

(3). Petri space: Each node acts as a transition operation to produce the tokens, which are then used to select suitable fuzzy laws. This can be described as

$$t_{ijk} = \begin{cases} 1, & \mu_{ijk} \geq g_{th} \\ 0, & \mu_{ijk} < g_{th} \end{cases} \tag{11}$$

where t_{ijk} denotes the transition nodes; μ_{ijk} denotes the average value of $\underline{\mu}_{ijk}$ and $\overline{\mu}_{ijk}$; g_{th} denotes the dynamic threshold value, which is given as

$$g_{th} = \frac{\varphi \exp(-\psi E)}{1 + \exp(-\psi E)} \tag{12}$$

where φ and ψ denote the positive constants for adjusting the Petri threshold; E denotes the energy function, which can be described as $E = \frac{1}{2}e^2$, in which the tracking error is denoted by e.

As shown in Equation (11), the transition node, t_{ijk}, is enabled when the value of μ_{ijk} is at least equal to the dynamic threshold value, g_{th}. The operation of simple PN is illustrated in Figure 3.

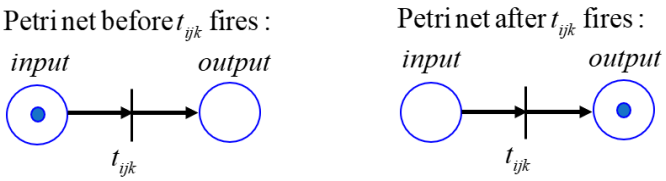

Figure 3. The operation of simple Petri net (PN).

(4). Receptive-field space: Each node acts as a t-norm operation. The illustrative mechanism for mapping two-dimensional inputs is shown in Figure 4. The multi-dimensional receptive-field function is given by

$$\begin{aligned} \boldsymbol{f}_{jk} &= \left[f_{11}, \ldots f_{1n_k}, \ldots, f_{n_j1}, \ldots, f_{n_jn_k} \right] \\ \overline{\boldsymbol{f}}_{jk} &= \left[\overline{f}_{11}, \ldots \overline{f}_{1n_k}, \ldots, \overline{f}_{n_j1}, \ldots, \overline{f}_{n_jn_k} \right] \end{aligned} \tag{13}$$

where

$$f_{jk} = \prod_{i=1}^{n_i} \mu_{ijk} \text{ and } \overline{f}_{jk} = \prod_{i=1}^{n_i} \overline{\mu}_{ijk} \tag{14}$$

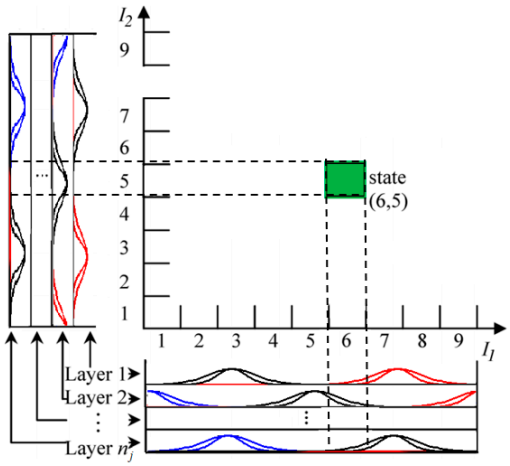

Figure 4. Mechanism for mapping two-dimensional inputs in the association memory space.

(5). Weight memory space: Each location $\widetilde{f}_{jk} = \left[f_{jk} \; \overline{f}_{jk} \right]$ corresponds to a particular adjustable value in the output weight space, $\widetilde{w}_{jk} = \left[w_{jk} \; \overline{w}_{jk} \right]$, which can be described as

$$\begin{aligned} \boldsymbol{w}_{jk} &= \left[w_{11}, \ldots w_{1n_k}, \ldots, w_{n_j1}, \ldots, w_{n_jn_k} \right] \in \Re^{n_jn_k} \\ \overline{\boldsymbol{w}}_{jk} &= \left[\overline{w}_{11}, \ldots \overline{w}_{1n_k}, \ldots, \overline{w}_{n_j1}, \ldots, \overline{w}_{n_jn_k} \right] \in \Re^{n_jn_k} \end{aligned} \tag{15}$$

where w_{jk} denotes the connecting weight between the pre-output space and the receptive-field space; the adaptive laws for the online adjusting of the weight memory space are given in Section 3.3.

(6). Pre-output space: Each node performs defuzzification to obtain the left and right-most point values of the type reduction for the SRIT2PC. The output of this space is given by

$$y_k^l = \frac{\sum_{j=1}^{n_j} f_{jk} w_{jk}}{\sum_{j=1}^{n_j} f_{jk}} \text{ and } y_k^r = \frac{\sum_{j=1}^{n_j} \overline{f}_{jk} \overline{w}_{jk}}{\sum_{j=1}^{n_j} \overline{f}_{jk}} \tag{16}$$

(7). Output layer: The output of this space, which is the final output of the SRIT2PC is given by the algebraic sum of the left and right most point values in the pre-output space:

$$\hat{u}_{SRIT2PC}^k = u_k = \frac{y_k^l + y_k^r}{2} \tag{17}$$

The control signal, $\hat{u}_{SRIT2PC}^k$, is then applied to estimate the ideal controller in Equation (7).

3.2. Self-Evolving Algorithm

In designing the network structure for the RIT2PC, choosing the number of layers greatly affects the control system. If the number of layers is large, huge computation times will follow; however, a few numbers of layers may not cover all cases, especially when the input changes across a wide range of values. To overcome this problem, this study presents the self-evolving algorithm to construct the layers of the proposed network autonomously. The flowchart of the self-evolving algorithm is shown in Figure 5.

The condition for generating new layers can be described as follows:

$$If \ \left(\phi_g^I < D_g\right) \quad Then \quad \{Generating\, a\, new\, layer\} \tag{18}$$

$$\phi_g^I = max\left[\mu_{i11}, \ldots, \mu_{i1n_k}, \mu_{i21}, \ldots, \mu_{i2n_k}, \ldots, \mu_{in_j1}, \ldots, \mu_{in_jn_k}\right] \tag{19}$$

$$\mu_{ijk} = \frac{\mu_{ijk} + \overline{\mu}_{ijk}}{2} \tag{20}$$

where ϕ_g^i and D_g denote the maximum membership grades and the generating threshold, respectively.

The T2GMF for a new layer is given as

$$m_{ijk}^{M(t)+1} = x_i(t) \tag{21}$$

$$\sigma_{ijk}^{M(t)+1} = v_{init} - \Delta v \text{ and } \overline{\sigma}_{ijk}^{M(t)+1} = \sigma_{init} + \Delta\sigma \tag{22}$$

where M(t) denotes the number of the existing rules at time t; σ_{init} denotes the initial value of the variance; $\Delta\sigma$ denotes the half of the variance uncertain.

The condition for deleting unnecessary layers can be described as

$$If \ \left(\phi_d^I < D_d\right) \quad Then \quad \left\{deleting\, the\, I^{th}\, layer\right\} \tag{23}$$

$$\phi_d^I = min\left[\mu_{i11}, \ldots, \mu_{i1n_k}, \mu_{i21}, \ldots, \mu_{i2n_k}, \ldots, \mu_{in_j1}, \ldots, \mu_{in_jn_k}\right] \tag{24}$$

where ϕ_d^i and D_d denote the minimum membership grades and the deleting threshold, respectively.

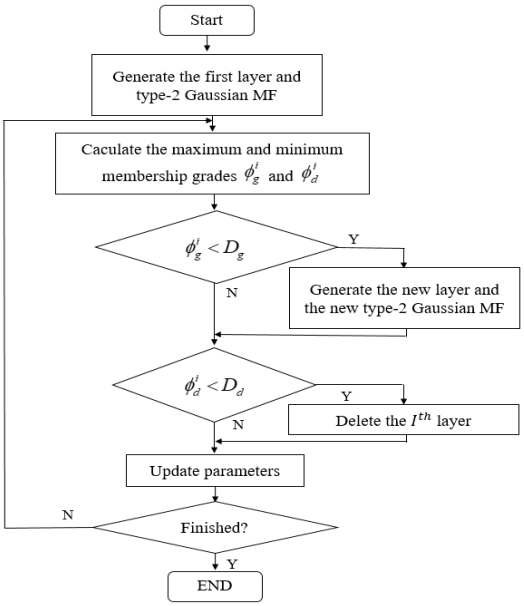

Figure 5. Self-evolving flowchart for RIT2PC.

3.3. Parameter Learning For SRIT2PC

Herein, we can assume there exists an optimal controller $\boldsymbol{u}^*_{SRIT2PC}$ such that

$$\boldsymbol{u}^*(t) = \boldsymbol{u}^*_{SRIT2PC}(w^*, \overline{w}^*, m^*, \sigma^*, \overline{\sigma}^*, r^*, \overline{r}^*, t) - \xi(t) \tag{25}$$

where $\xi(t)$ denotes the approximation error; $w^*, \overline{w}^*, m^*, \sigma^*, \overline{\sigma}^*, r^*, \overline{r}^*$ denote the optimal parameters for $w, \overline{w}, m, \sigma, \overline{\sigma}, r, \overline{r}$, respectively.

Since $\boldsymbol{u}^*_{SRIT2PC}$ cannot be determined, an online estimation controller, $\hat{\boldsymbol{u}}_{SRIT2PC}$, is used to estimate $\boldsymbol{u}^*_{SRIT2PC}$. Thus, the control input is denoted as

$$\hat{\boldsymbol{u}}(t) = \hat{\boldsymbol{u}}_{SRIT2PC}(\hat{w}, \hat{\overline{w}}, \hat{m}, \hat{v}, \hat{\overline{v}}, \hat{r}, \hat{\overline{r}}, t) - \hat{\boldsymbol{u}}_F(t) \tag{26}$$

where $\hat{w}, \hat{\overline{w}}, \hat{m}, \hat{v}, \hat{\overline{v}}, \hat{r}, \hat{\overline{r}}$ denote the estimation of $w^*, \overline{w}^*, m^*, v^*, \overline{v}^*, r^*, \overline{r}^*$, respectively; $\hat{u}_F$ denotes the estimation of fuzzy compensator controller.

A high-order sliding surface can be defined as

$$\boldsymbol{s}(t) = \boldsymbol{e}^{(n-1)} + k_1 \boldsymbol{e}^{(n-2)} \ldots + k_n \int_0^t \boldsymbol{e}(\tau) d\tau \tag{27}$$

Taking the derivative of Equation (27) and using Equation (6), the following can be obtained:

$$\dot{\boldsymbol{s}}(t) = \boldsymbol{e}^{(n)} + \boldsymbol{K}^T \boldsymbol{e} = \boldsymbol{A}\boldsymbol{e}(t) + \boldsymbol{d}(t) + \Delta \boldsymbol{f}(t) + \boldsymbol{u}(t) + \boldsymbol{K}^T \boldsymbol{e} \tag{28}$$

The Lyapunov function can be described as

$$V_1(\boldsymbol{s}(t)) = \frac{1}{2}\boldsymbol{s}^2(t) \tag{29}$$

Taking the derivative of Equation (29) and using Equations (26) and (28), the following can be obtained:

$$\begin{aligned}\dot{V}_1(t) &= \boldsymbol{s}(t)\dot{\boldsymbol{s}}(t) = \boldsymbol{s}(t)\left[\boldsymbol{e}^{(n)} + \boldsymbol{K}^T\boldsymbol{e}\right] \\ &= \boldsymbol{s}(t)[\boldsymbol{A}\boldsymbol{e}(t) + \boldsymbol{d}(t) + \Delta\boldsymbol{f}(t) + \left(\hat{\boldsymbol{u}}_{SRIT2PC}(\hat{w}, \hat{\overline{w}}, \hat{m}, \hat{v}, \hat{\overline{v}}, \hat{r}, \hat{\overline{r}}, t) - \hat{\boldsymbol{u}}_F(t)\right) + \boldsymbol{K}^T\boldsymbol{e}\Big]\end{aligned} \tag{30}$$

Using the gradient descent method, the parameter-updating laws for SRIT2PC can be obtained as follows:

$$\hat{w}_{jk}(t+1) = \hat{w}_{jk}(t) - \hat{\eta}_w\frac{\partial\dot{V}_1(t)}{\partial\hat{w}_{jk}} = \hat{w}_{jk}(t) - \hat{\eta}_w\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial y^l_k}\frac{\partial y^l_k}{\partial\hat{w}_{jk}} = \hat{w}_{jk}(t) - \frac{1}{2}\hat{\eta}_w s(t) f_{jk} \tag{31}$$

$$\hat{\overline{w}}_{jk}(t+1) = \hat{\overline{w}}_{jk}(t) - \hat{\eta}_w\frac{\partial\dot{V}_1(t)}{\partial\hat{\overline{w}}_{jk}} = \hat{\overline{w}}_{jk}(t) - \hat{\eta}_w\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial y^r_k}\frac{\partial y^r_k}{\partial\hat{\overline{w}}_{jk}} = \hat{\overline{w}}_{jk}(t) - \frac{1}{2}\hat{\eta}_w s(t)\overline{f}_{jk} \tag{32}$$

$$\begin{aligned}\hat{m}_{ijk}(t+1) &= \hat{m}_{ijk}(t) - \hat{\eta}_m\frac{\partial\dot{V}_1(t)}{\partial\hat{m}_{ijk}} \\ &= \hat{m}_{ijk}(t) - \hat{\eta}_m\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\left(\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial y^l_k}\frac{\partial y^l_k}{\partial f_{jk}}\frac{\partial f_{jk}}{\partial\mu_{ijk}}\frac{\partial\mu_{ijk}}{\partial\hat{m}_{ijk}} + \frac{\partial\hat{u}_{SRIT2PC}}{\partial y^r_k}\frac{\partial y^r_k}{\partial\overline{\phi}_{jk}}\frac{\partial\overline{f}_{jk}}{\partial\overline{\mu}_{ijk}}\frac{\partial\overline{\mu}_{ijk}}{\partial\hat{m}_{ijk}}\right)\end{aligned} \tag{33}$$

$$\begin{aligned}\hat{\sigma}_{ijk}(t+1) &= \hat{\sigma}_{ijk}(t) - \hat{\eta}_\sigma\frac{\partial\dot{V}_1(t)}{\partial\hat{\sigma}_{ijk}} \\ &= \hat{\sigma}_{ijk}(t) - \hat{\eta}_\sigma\left(\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial\hat{y}^l_k}\frac{\partial\hat{y}^l_k}{\partial f_{jk}}\frac{\partial f_{jk}}{\partial\mu_{ijk}}\frac{\partial\mu_{ijk}}{\partial\hat{\sigma}_{ijk}}\right) = \hat{\sigma}_{ijk}(t) - \hat{\eta}_\sigma s(t) f_{jk} w_{jk}\frac{\left(I_i - \hat{m}_{ijk}\right)^2}{\hat{\sigma}^3_{ijk}}\end{aligned} \tag{34}$$

$$\begin{aligned}\hat{\overline{\sigma}}_{ijk}(t+1) &= \hat{\overline{\sigma}}_{ijk}(t) - \hat{\eta}_\sigma\frac{\partial\dot{V}_1(t)}{\partial\hat{\overline{\sigma}}_{ijk}} \\ &= \hat{\overline{\sigma}}_{ijk}(t) - \hat{\eta}_\sigma\left(\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial\hat{y}^r_k}\frac{\partial\hat{y}^r_k}{\partial\overline{f}_{jk}}\frac{\partial\overline{f}_{jk}}{\partial\overline{\mu}_{ijk}}\frac{\partial\overline{\mu}_{ijk}}{\partial\hat{\overline{\sigma}}_{ijk}}\right) = \hat{\overline{\sigma}}_{ijk}(t) - \hat{\eta}_\sigma s(t)\overline{f}_{jk}\overline{w}_{jk}\frac{\left(I_i - \hat{m}_{ijk}\right)^2}{\hat{\overline{\sigma}}^3_{ijk}}\end{aligned} \tag{35}$$

$$\begin{aligned}\hat{r}_{ijk}(t+1) &= \hat{r}_{ijk}(t) - \hat{\eta}_r\frac{\partial\dot{V}_1(t)}{\partial\hat{r}_{ijk}} \\ &= \hat{r}_{ijk}(t) - \hat{\eta}_\sigma\left(\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial\hat{y}^l_k}\frac{\partial\hat{y}^l_k}{\partial f_{jk}}\frac{\partial f_{jk}}{\partial\mu_{ijk}}\frac{\partial\mu_{ijk}}{\partial\hat{I}_{ri}}\frac{\partial\hat{I}_{ri}}{\partial\hat{r}_{ijk}}\right) = \hat{r}_{ijk}(t) + \tfrac{1}{2}\hat{\eta}_r s(t) f_{jk} w_{jk}\mu_{ijk}\frac{\left(I_i - \hat{m}_{ijk}\right)}{\hat{\sigma}^2_{ijk}}\end{aligned} \tag{36}$$

$$\begin{aligned}\hat{\overline{r}}_{ijk}(t+1) &= \hat{\overline{r}}_{ijk}(t) - \hat{\eta}_r\frac{\partial\dot{V}_1(t)}{\partial\hat{\overline{r}}_{ijk}} \\ &= \hat{\overline{r}}_{ijk}(t) - \hat{\eta}_\sigma\left(\frac{\partial\dot{V}_1(t)}{\partial\hat{u}^k_{SRIT2PC}}\frac{\partial\hat{u}^k_{SRIT2PC}}{\partial\hat{y}^r_k}\frac{\partial\hat{y}^r_k}{\partial\overline{f}_{jk}}\frac{\partial\overline{f}_{jk}}{\partial\overline{\mu}_{ijk}}\frac{\partial\overline{\mu}_{ijk}}{\partial\hat{\overline{I}}_{ri}}\frac{\partial\hat{\overline{I}}_{ri}}{\partial\hat{\overline{r}}_{ijk}}\right) = \hat{\overline{r}}_{ijk}(t) + \tfrac{1}{2}\hat{\eta}_r s(t)\overline{f}_{jk}\overline{w}_{jk}\overline{\mu}_{ijk}\frac{\left(I_i - \hat{m}_{ijk}\right)}{\hat{\overline{\sigma}}^2_{ijk}}\end{aligned} \tag{37}$$

where the positive learning-rates are denoted by $\hat{\eta}_w, \hat{\eta}_m, \hat{\eta}_\sigma, \hat{\eta}_r$.

3.4. Compensator Controller

To address the approximation error, a simple fuzzy compensator controller can be proposed as follows:

$$\begin{aligned}&R^1: \ If\ s_i\ is\ POS,\ \ then\ u^i_F\ is\ FP \\ &R^2: \ If\ s_i\ is\ ZE,\ \ then\ u^i_F\ is\ FZ \\ &R^3: \ If\ s_i\ is\ NEG,\ then\ u^i_F\ is\ FN\end{aligned} \tag{38}$$

where POS, ZE, and NEG denote the positive, zero, and negative inputs of the MFs, respectively; FP, FZ, and FN denote the positive, zero, and negative outputs the MFs, respectively; s_i and u^i_F denote the control input and control output, respectively.

Figure 6 shows the input and output MFs of the fuzzy compensator controller. Using the center-of-gravity method, the control output is given by

$$u_F^i = \frac{\sum_{a=1}^{3} \alpha_a^i \beta_a^i}{\sum_{a=1}^{3} \beta_a^i} = \alpha_1^i \beta_1^i + \alpha_2^i \beta_2^i + \alpha_3^i \beta_3^i \tag{39}$$

where $\boldsymbol{\alpha}^i = \left[\alpha_1^i, \alpha_2^i, \alpha_3^i\right]$ denotes the weight vector of the fuzzy rules and $\boldsymbol{\beta}^i = \left[\beta_1^i, \beta_2^i, \beta_3^i\right]$ denotes the firing-strengths vector of the fuzzy rules, which is given by

$$\begin{aligned}
&Case\,1: \quad (s_i \le -\vartheta)\\
&\qquad \beta_1^i = 0;\ \beta_2^i = 0;\ \beta_3^i = 1;\\
&Case\,2: \quad (-\vartheta \le s_i \le 0)\\
&\qquad \beta_1^i = 0;\ \beta_2^i = (s_i + \vartheta)/\vartheta;\ \beta_3^i = 1 - \beta_2^i;\\
&Case\,3: \quad (0 \le s_i \le \vartheta)\\
&\qquad \beta_1^i = 1 - \beta_2^i;\ \beta_2^i = (\vartheta - s_i)/\vartheta;\ \beta_3^i = 0;\\
&Case\,4: \quad (s_i > \vartheta)\\
&\qquad \beta_1^i = 1;\ \beta_2^i = 0;\ \beta_3^i = 0;
\end{aligned} \tag{40}$$

where ϑ is the parameter for defining the firing strengths.

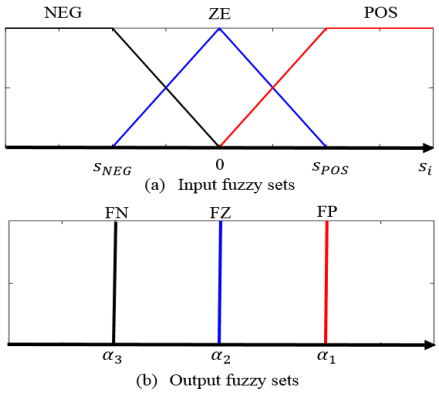

Figure 6. The input and output membership functions (MFs).

By choosing the triangular membership function for the input shown in Figure 6, we can obtain $\beta_1^i + \beta_2^i + \beta_3^i = 1$. Using a singleton membership function for the output, letting $\alpha_1^i = \hat{\alpha}_i,\ \alpha_2^i = 0,\ \alpha_3^i = -\hat{\alpha}_i$, and rewriting Equation (39) using $\boldsymbol{\alpha}^i = [\hat{\alpha}_i,\ \ 0,\ -\hat{\alpha}_i]$, the following can be obtained:

$$u_F^i = \hat{\alpha}_i\left(\beta_1^i - \beta_3^i\right) \tag{41}$$

Rewriting Equation (30) and using Equations (25), (26), and (41), the following can be obtained:

$$\begin{aligned}
\dot{V}_1 &= \sum_{i=1}^{m}\left[s_i(t)\xi_i(t) - s_i(t)\hat{\alpha}_i\left(\beta_1^i - \beta_3^i\right)\right]\\
&\le \sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\xi_i(t)\right| - s_i(t)\hat{\alpha}_i\left(\beta_1^i - \beta_3^i\right)\right]\\
&= \sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\xi_i(t)\right| - \hat{\alpha}_i\left|s_i(t)\right|.\left|\beta_1^i - \beta_3^i\right|\right]\\
&= -\sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\beta_1^i - \beta_3^i\right|\right]\left(\hat{\alpha}_i - \frac{\left|\xi_i(t)\right|}{\left|\beta_1^i - \beta_3^i\right|}\right)
\end{aligned} \tag{42}$$

where m denotes the dimension of vector $\boldsymbol{s}_i$. In Equation (42), if there exists an estimated value $\hat{\alpha}_i > \frac{|\xi_i(t)|}{|\beta_1^i - \beta_3^i|}$, then $\dot{V} \leq 0$ is satisfied. An optimal value α_i^* can be defined to achieve a minimum value of $\hat{\alpha}_i$ with the following equation:

$$\alpha_i^* = \frac{\left|\xi_i(t)\right|}{\left|\beta_1^i - \beta_3^i\right|} + \Omega_i \tag{43}$$

where Ω_i denotes a positive constant.

The estimation-error vector can be described as $\widetilde{\boldsymbol{\alpha}}_i = [\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_i, \ldots, \widetilde{\alpha}_m]^T$, where $\widetilde{\alpha}_i$ is given as

$$\widetilde{\alpha}_i = \alpha_i^* - \hat{\alpha}_i \tag{44}$$

Accordingly, the Lyapunov function can be defined as

$$V_2(\boldsymbol{s}(t)) = \frac{1}{2}\boldsymbol{s}^T(t)\boldsymbol{s}(t) + \frac{1}{2}\widetilde{\boldsymbol{\alpha}}^T\widetilde{\boldsymbol{\alpha}} \tag{45}$$

Taking the derivative of Equation (45) and using Equations (7), (25), (30), and (41), the following can be obtained:

$$\begin{aligned}
&= \sum_{i=1}^{m}\left[s_i(t)\left[-u_i^*(t)\right] + s_i(t)\left[\hat{u}_{SRIT2PC}^i(t) - \hat{u}_F^i(t)\right] + \widetilde{\alpha}_i\dot{\widetilde{\alpha}}_i\right] \\
&= \sum_{i=1}^{m}\left[s_i(t)\left[\xi(t) - \hat{u}_{SRIT2PC}^i(t)\right] + s_i(t)\left[\hat{u}_{SRIT2PC}^i(t) - \hat{u}_F^i(t)\right] + \widetilde{\alpha}_i\dot{\widetilde{\alpha}}_i\right] \\
&= \sum_{i=1}^{m}\left[s_i(t)\xi_i(t) - \hat{\alpha}_i s_i(t)\left(\beta_1^i - \beta_3^i\right) + \widetilde{\alpha}_i\dot{\widetilde{\alpha}}_i\right] \\
&\leq \sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\xi_i(t)\right| + \widetilde{\alpha}_i s_i(t)\left(\beta_1^i - \beta_3^i\right) - \alpha_i^*\left|s_i(t)\right|.\left|\beta_1^i - \beta_3^i\right| + \widetilde{\alpha}_i\dot{\widetilde{\alpha}}_i\right] \\
&= \sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\xi_i(t)\right| + \widetilde{\alpha}_i\left[s_i(t)\left(\beta_1^i - \beta_3^i\right) + \dot{\widetilde{\alpha}}_i\right] - \alpha_i^*\left|s_i(t)\right|.\left|\beta_1^i - \beta_3^i\right|\right]
\end{aligned} \tag{46}$$

The estimation laws can be described as

$$\dot{\hat{\alpha}}_i = -\dot{\widetilde{\alpha}}_i = s_i(t)\left(\beta_1^i - \beta_3^i\right) \tag{47}$$

Accordingly, rewriting Equation (46) and using Equation (43), the following can be obtained:

$$\begin{aligned}
\dot{V}_2(\boldsymbol{s}(t)) &\leq \sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\xi_i(t)\right| - \left|s_i(t)\right|\left|\beta_1^i - \beta_3^i\right|\left(\frac{\left|\xi_i(t)\right|}{\left|\beta_1^i - \beta_3^i\right|} + \Omega_i\right)\right] \\
&= \sum_{i=1}^{m}\left[\left|s_i(t)\right|.\left|\xi_i(t)\right| - \left|s_i(t)\right|\left(\left|\xi_i(t)\right| + \Omega_i\left|\beta_1^i - \beta_3^i\right|\right)\right] \\
&= -\sum_{i=1}^{m}\Omega_i\left|s_i(t)\right|\left|\beta_1^i - \beta_3^i\right|
\end{aligned} \tag{48}$$

Since $\dot{V}_2(\boldsymbol{s}(t))$ is a negative semidefinite, the stability of the proposed SRIT2PC control system can be guaranteed by the Lyapunov stability theorem.

4. Illustrative Examples

To verify the feasibility and effectiveness of the proposed controller, an illustrative example is used to describe the Lorenz–Stenflo chaotic system. Using the proposed parameter-adaptive laws, the control signals, $\hat{u}_x(t)$, $\hat{u}_y(t)$, $\hat{u}_z(t)$, $\hat{u}_w(t)$, can be obtained, after which point the synchronization of the slave and the master chaotic can be obtained. The initial positions for the chaotic system are $[x_1, y_1, z_1, w_1] = [0.028,\ 0.02,\ 0.03,\ 0.048]^T$ and $[x_2, y_2, z_2, w_2] = [0.01,\ 0.037,\ 0.029,\ 0.008]^T$. The system uncertainties are $[\Delta f(x_2), \Delta f(y_2), \Delta f(z_2), \Delta f(w_2)] = rd(.)[0.2x_2,\ 0.2y_2,\ 0.2z_2, 0.2w_2]^T$. The external disturbances are $\left[d_x, d_y, d_z, d_w\right] = [0.2\cos\pi t, 0.5\cos\pi t, 0.3\cos\pi t, 0.4\cos\pi t]^T$, where

$rd(.)$ denotes the random values in the range [0, 1]. The performance of the synchronization system can be calculated using the root mean square error (RMSE):

$$RMSE = \sqrt{\frac{1}{n_h}\sum_{h=1}^{n_h}\left((e_{xh})^2 + \left(e_{yh}\right)^2 + (e_{zh})^2 + (e_{wh})^2\right)} for \ \ h = 1, 2, \ldots n_h \tag{49}$$

where n_h denotes the number of samples; $e_{xh}, e_{yh}, e_{zh}, e_{wh}$ denote the tracking error for the h^{th} sample.

The parameters of the proposed controller consist of the following: $\sigma_{init} = 0.4$, $\Delta\sigma = 0.05$, $n_i = 3$, $n_j = 4$, $n_k = 1$, $D_g = 0.2$, $D_d = 0.02$, and $\vartheta = 0.04$; the sliding surface order is $n = 2$; the adaptive-learning rates are $\hat{\eta}_w = 0.01$, $\hat{\eta}_m = 0.001$, $\hat{\eta}_\sigma = 0.001$, and $\hat{\eta}_r = 0.001$. To limit the system computation burden, the maximum number of membership functions for each input is limited to seven MFs and the minimum number of MFs in each input is limited to one MF. The comparison results in RMSE for the proposed method and the other methods are given in Table 1, from which it is evident that the proposed SRIT2PC is superior over the wavelet CMAC controller (WCMAC) [13], the interval type-2 Petri CMAC (IT2PCMAC) [58], and the type-2 fuzzy-brain emotional-learning controller (T2FBELC) [59].

Case 1: $\theta = 0$

Using Equation (2), the parameters for defining the Lorenz–Stenflo chaotic-system attractor are $\alpha = 1$, $\tau = 26$, $\lambda = 1$, $\varphi = 0.7$, and $\gamma = 1.5$. The synchronization results of the 4D Lorenz–Stenflo chaotic system using the SRIT2PC are depicted in Figure 7; Figure 8 shows the trajectory signals, $\boldsymbol{x}_1(t)$, $\boldsymbol{y}_1(t)$, $z_1(t)$, and $\boldsymbol{w}_1(t)$, and the synchronization outputs, $\boldsymbol{x}_2(t)$, $\boldsymbol{y}_2(t)$, $z_2(t)$, and $\boldsymbol{w}_2(t)$; Figure 9 shows the control signals, $\boldsymbol{u}_x(t)$, $\boldsymbol{u}_y(t)$, $\boldsymbol{u}_z(t)$, and $\boldsymbol{u}_w(t)$; Figure 10 shows the tracking errors, $\boldsymbol{e}_x(t)$, $\boldsymbol{e}_y(t)$, $\boldsymbol{e}_z(t)$, and $\boldsymbol{e}_w(t)$. The number of layers of the SRIT2PC using the self-evolving algorithm are shown in Figure 11. In this case, the simulation results suggest that the proposed SRIT2PC can effectively synchronize the slave chaotic system with the master system.

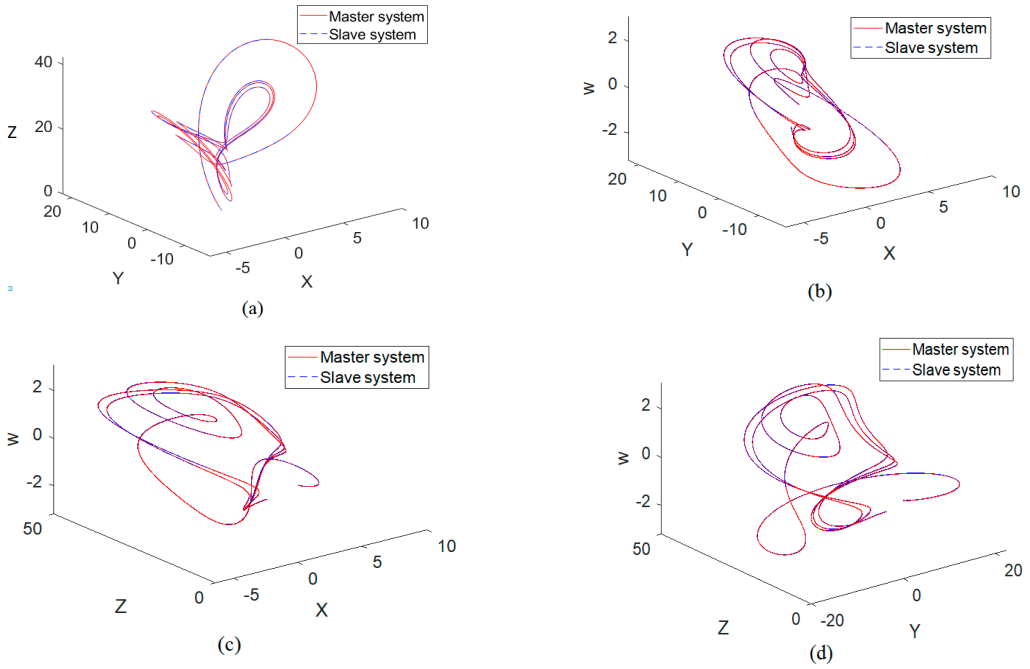

Figure 7. Synchronization of 4D Lorenz–Stenflo chaotic system using the SRIT2PC for Case 1: (**a**) x–y–z space, (**b**) x–y–w space, (**c**) x–z–w space, and (**d**) y–z–w space.

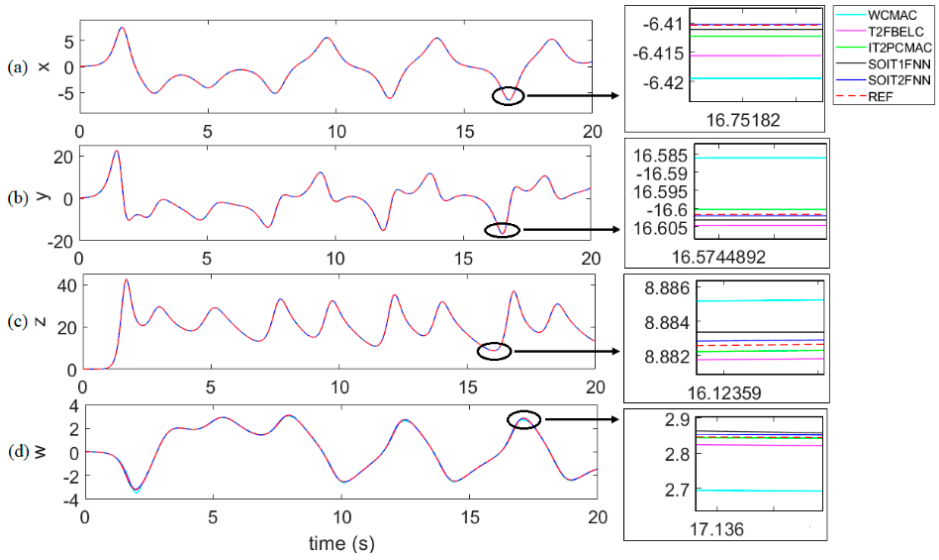

Figure 8. System outputs between the proposed SRIT2PC and other synchronization methods for Case 1: (**a**) x_1, x_2, (**b**) y_1, y_2, (**c**) z_1, z_2, and (**d**) w_1, w_2.

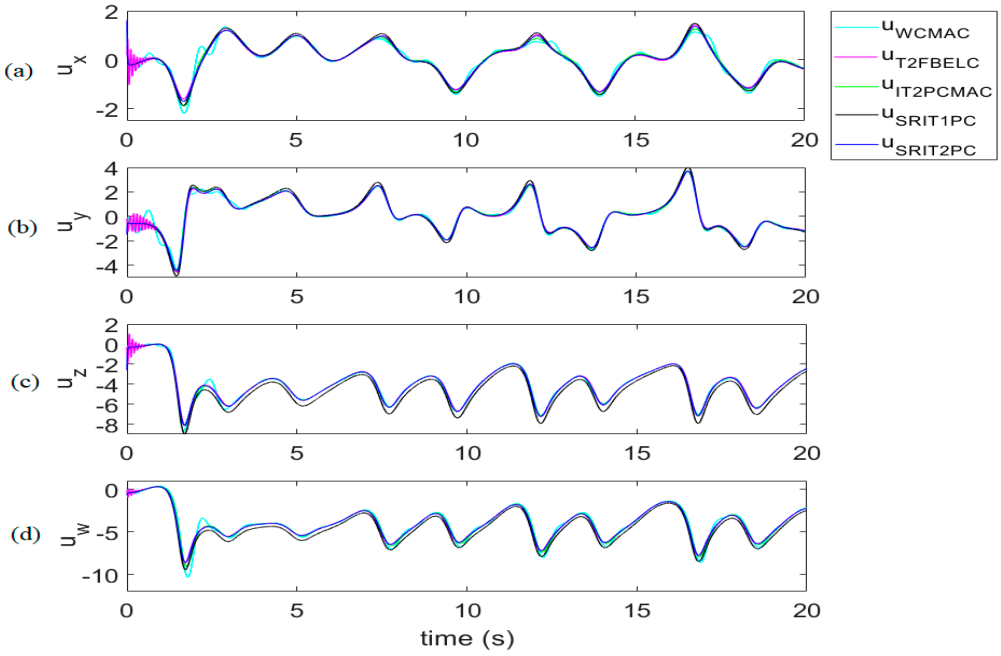

Figure 9. Control signals between the proposed SRIT2PC and other synchronization methods for Case 1: (**a**) u_x, (**b**) u_y, (**c**) u_z, and (**d**) u_w.

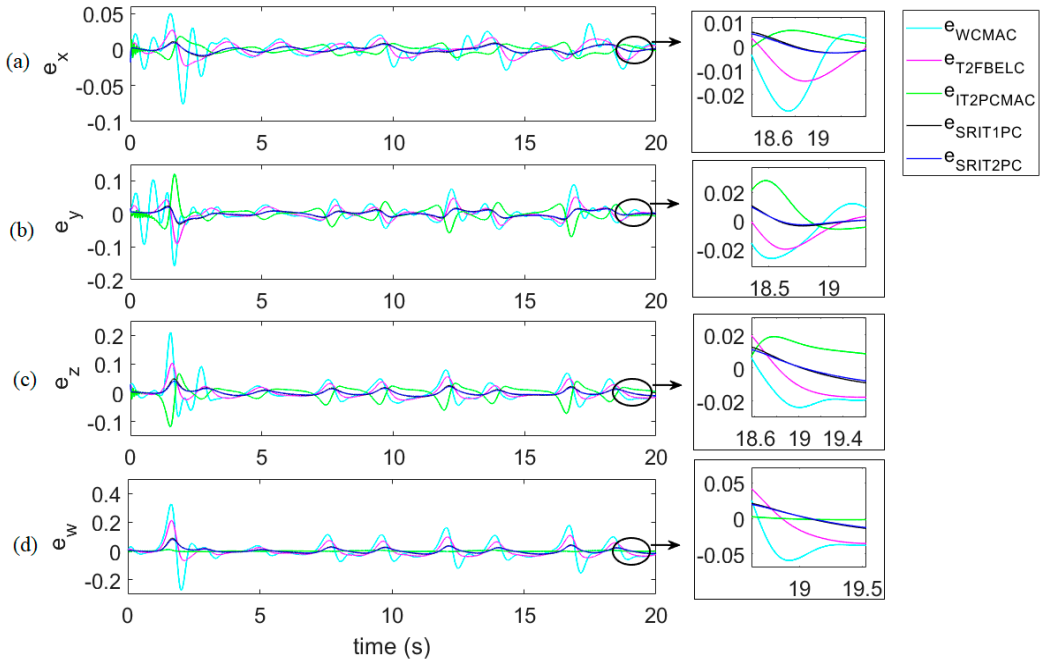

Figure 10. Tracking errors between the proposed SRIT2PC and other synchronization methods for Case 1: (**a**) e_x, (**b**) e_y, (**c**) e_z, and (**d**) e_w.

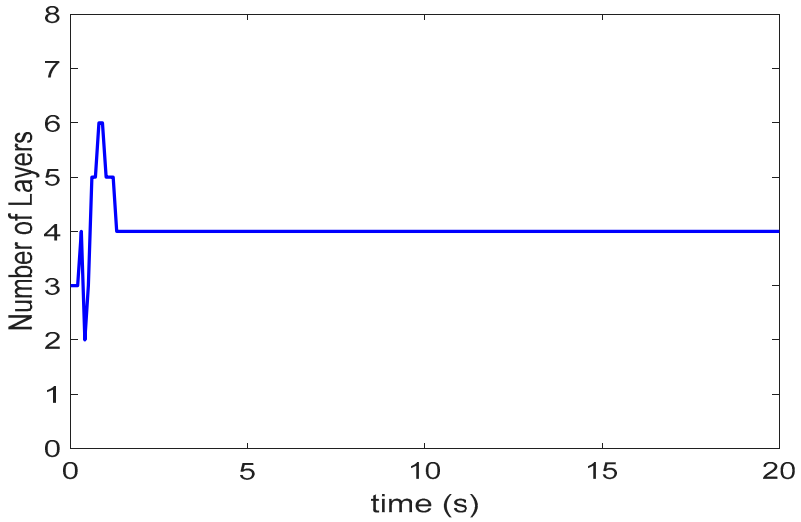

Figure 11. Number of layers using self-evolving algorithm for Case 1.

Case 2: $\theta = 0.8$

Using Equation (2), the parameters for defining the Lorenz–Stenflo chaotic-system attractor are $\alpha = 21$, $\tau = -2$, $\lambda = -22.2$, $\varphi = 0.9667$, and $\gamma = 2.3$. The synchronization results of the 4D Lorenz–Stenflo chaotic system using the SRIT2PC are depicted in Figure 12; Figure 13 shows the trajectory signals, $\boldsymbol{x}_1(t)$, $\boldsymbol{y}_1(t)$, $\boldsymbol{z}_1(t)$, and $\boldsymbol{w}_1(t)$, and the synchronization outputs, $\boldsymbol{x}_2(t)$, $\boldsymbol{y}_2(t)$, $\boldsymbol{z}_2(t)$, and $\boldsymbol{w}_2(t)$; Figure 14 shows the control signals, $\boldsymbol{u}_x(t)$, $\boldsymbol{u}_y(t)$, $\boldsymbol{u}_z(t)$, and $\boldsymbol{u}_w(t)$; Figure 15 shows the tracking errors, $\boldsymbol{e}_x(t)$, $\boldsymbol{e}_y(t)$, $\boldsymbol{e}_z(t)$, and $\boldsymbol{e}_w(t)$. The number of layers of the SRIT2PC using the self-evolving algorithm is shown in Figure 16. In this case, the simulation results suggest that the proposed SRIT2PC can effectively synchronize the slave chaotic system with the master system.

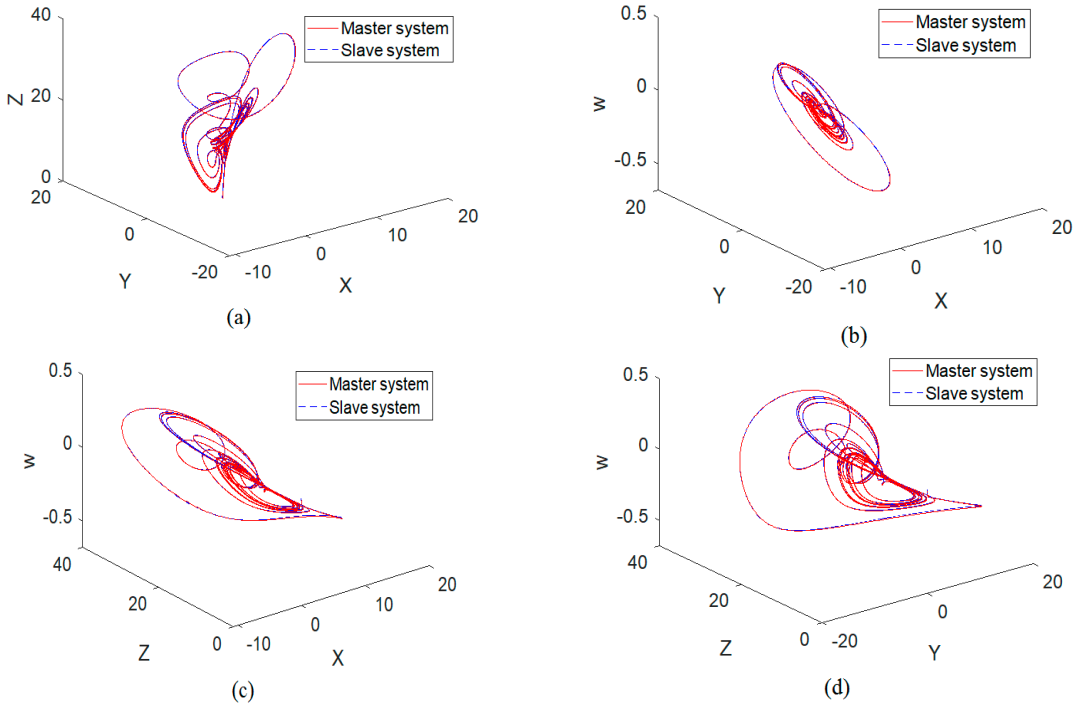

Figure 12. Synchronization of the 4D Lorenz–Stenflo chaotic system using the SRIT2PC for Case 2: (**a**) x–y–z space, (**b**) x–y–w space, (**c**) x–z–w space, and (**d**) y–z–w space.

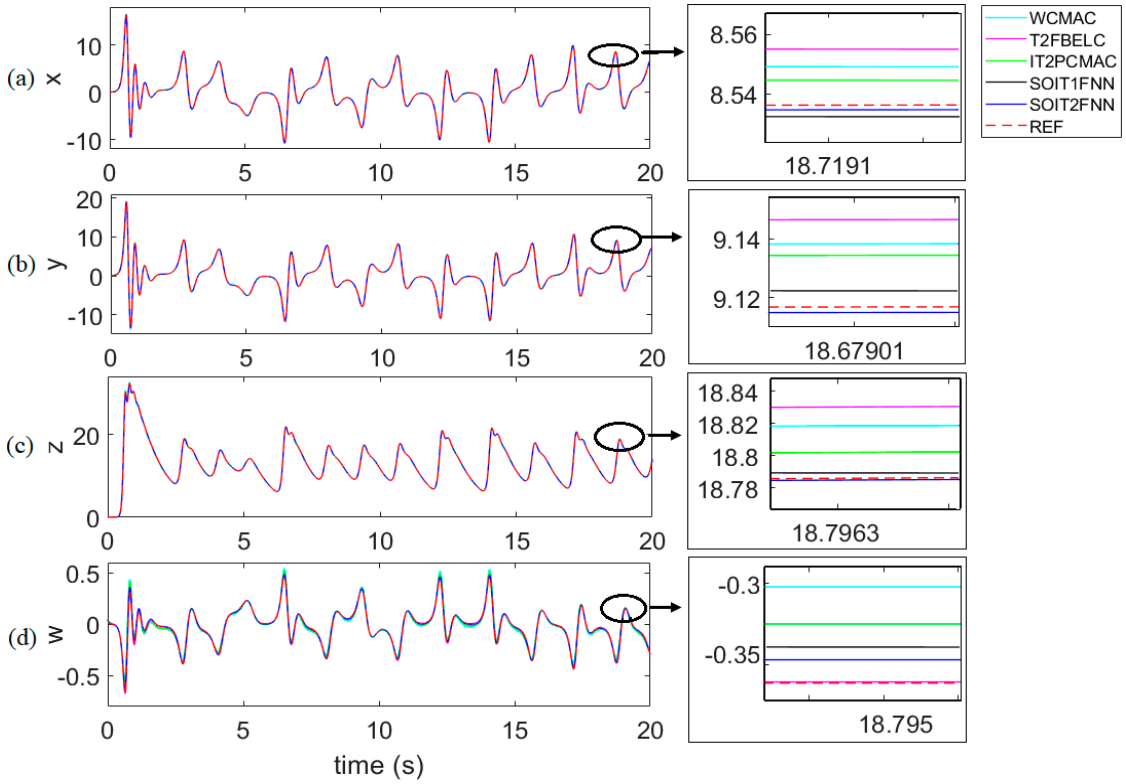

Figure 13. System outputs between the proposed SRIT2PC and other synchronization methods for Case 2: (**a**) x_1, x_2, (**b**) y_1, y_2, (**c**) z_1, z_2, and (**d**) w_1, w_2.

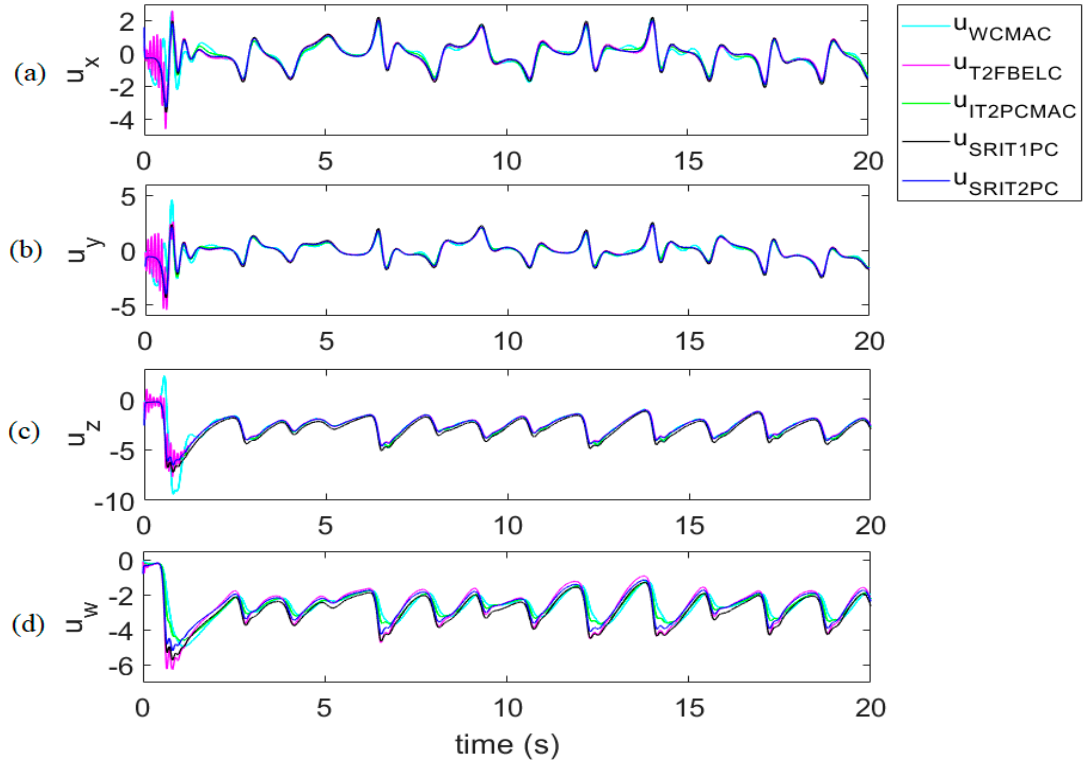

Figure 14. Control signals between the proposed SRIT2PC and other synchronization methods for Case 2: (**a**) u_x, (**b**) u_y, (**c**) u_z, and (**d**) u_w.

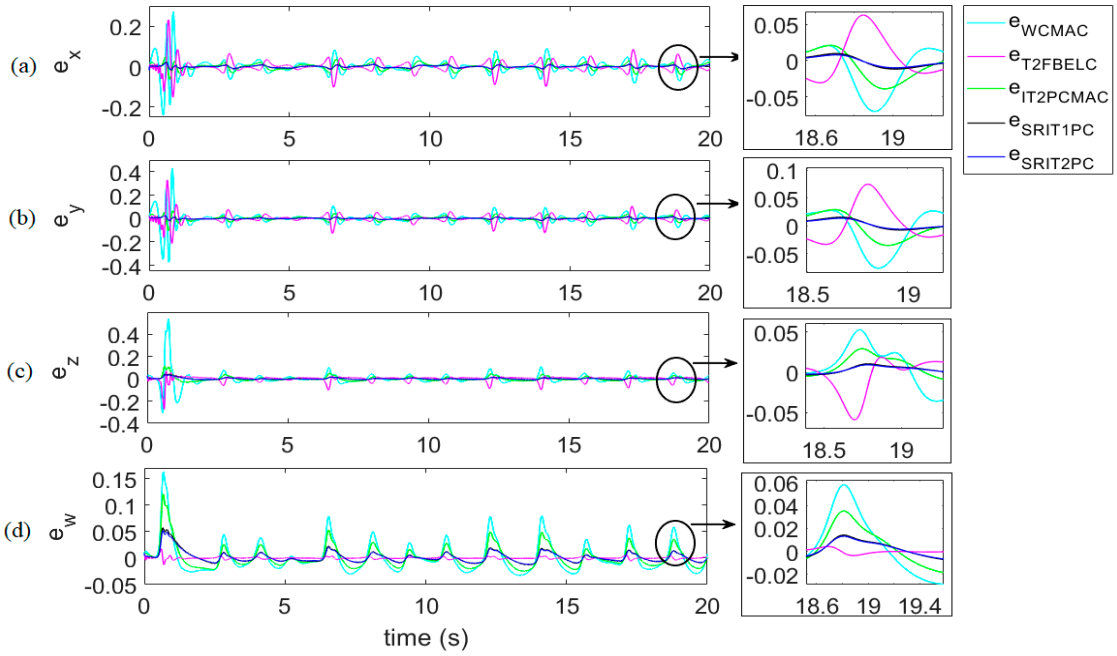

Figure 15. Tracking errors between the proposed SRIT2PC and other synchronization methods for Case 2: (**a**) e_x, (**b**) e_y, (**c**) e_z, and (**d**) e_w.

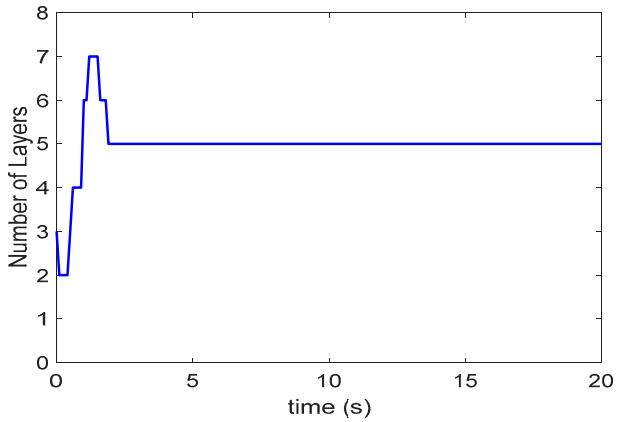

Figure 16. Number of layers using self-evolving algorithm for Case 2.

Case 3: $\theta = 1.0$

Using Equation (2), the parameters for defining the Lorenz–Stenflo chaotic-system attractor are $\alpha = 26$, $\tau = -9$, $\lambda = -28$, $\varphi = 1.033$, and $\gamma = 2.5$. The synchronization results of the 4D Lorenz–Stenflo chaotic system using the SRIT2PC are depicted in Figure 17; Figure 18 shows the trajectory signals, $\boldsymbol{x}_1(t)$, $\boldsymbol{y}_1(t)$, $\boldsymbol{z}_1(t)$, and $\boldsymbol{w}_1(t)$, and the synchronization outputs, $\boldsymbol{x}_2(t)$, $\boldsymbol{y}_2(t)$, $\boldsymbol{z}_2(t)$, and $\boldsymbol{w}_2(t)$; Figure 19 shows the control signals, $\boldsymbol{u}_x(t)$, $\boldsymbol{u}_y(t)$, $\boldsymbol{u}_z(t)$, and $\boldsymbol{u}_w(t)$; Figure 20 shows the tracking errors, $\boldsymbol{e}_x(t)$, $\boldsymbol{e}_y(t)$, $\boldsymbol{e}_z(t)$, and $\boldsymbol{e}_w(t)$. The number of layers of the SRIT2PC using the self-evolving algorithm is shown in Figure 21. In this case, the simulation results suggest that the proposed SRIT2PC can effectively synchronize the slave chaotic system with the master system.

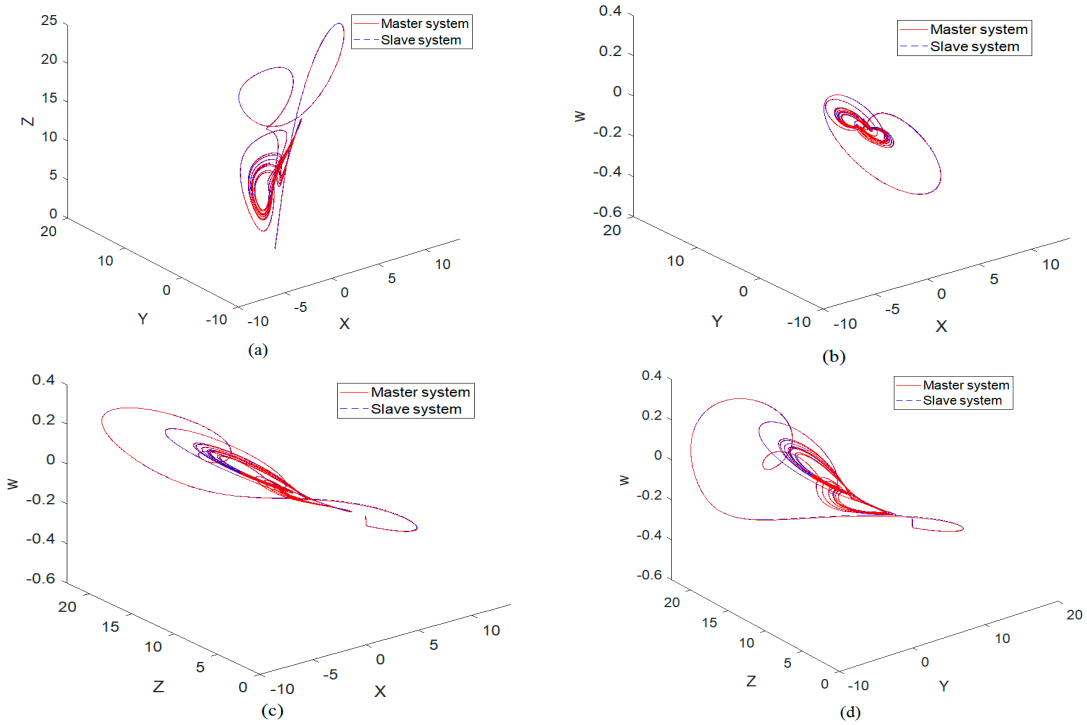

Figure 17. Synchronization of the 4D Lorenz–Stenflo chaotic system using the SRIT2PC for Case 3: (**a**) x–y–z space, (**b**) x–y–w space, (**c**) x–z–w space, and (**d**) y–z–w space.

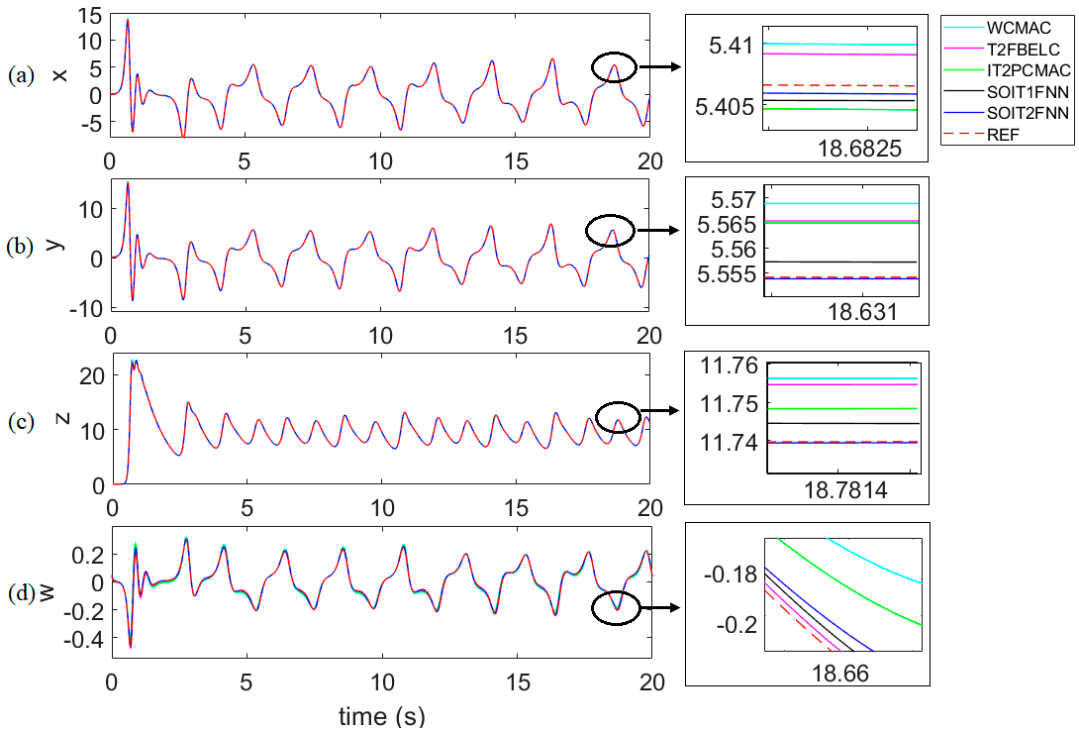

Figure 18. System outputs between the proposed SRIT2PC and other synchronization methods for Case 3: (**a**) x_1, x_2, (**b**) y_1, y_2, (**c**) z_1, z_2, and (**d**) w_1, w_2.

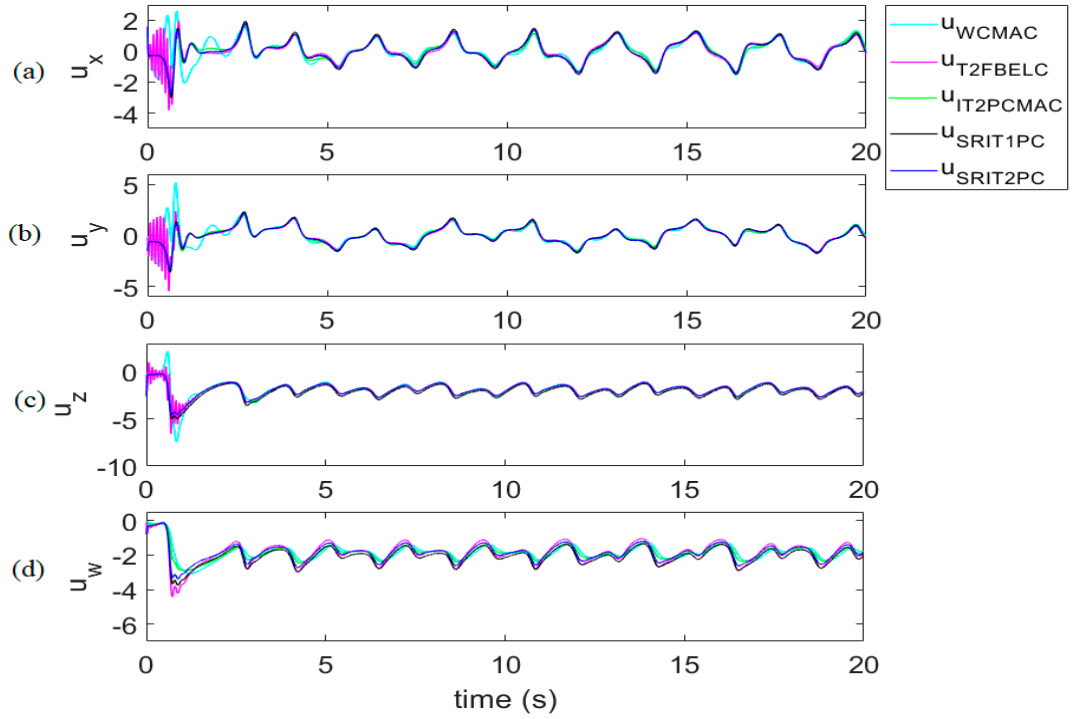

Figure 19. Control signals between the proposed SRIT2PC and other synchronization methods for Case 3: (**a**) u_x, (**b**) u_y, (**c**) u_z, and (**d**) u_w.

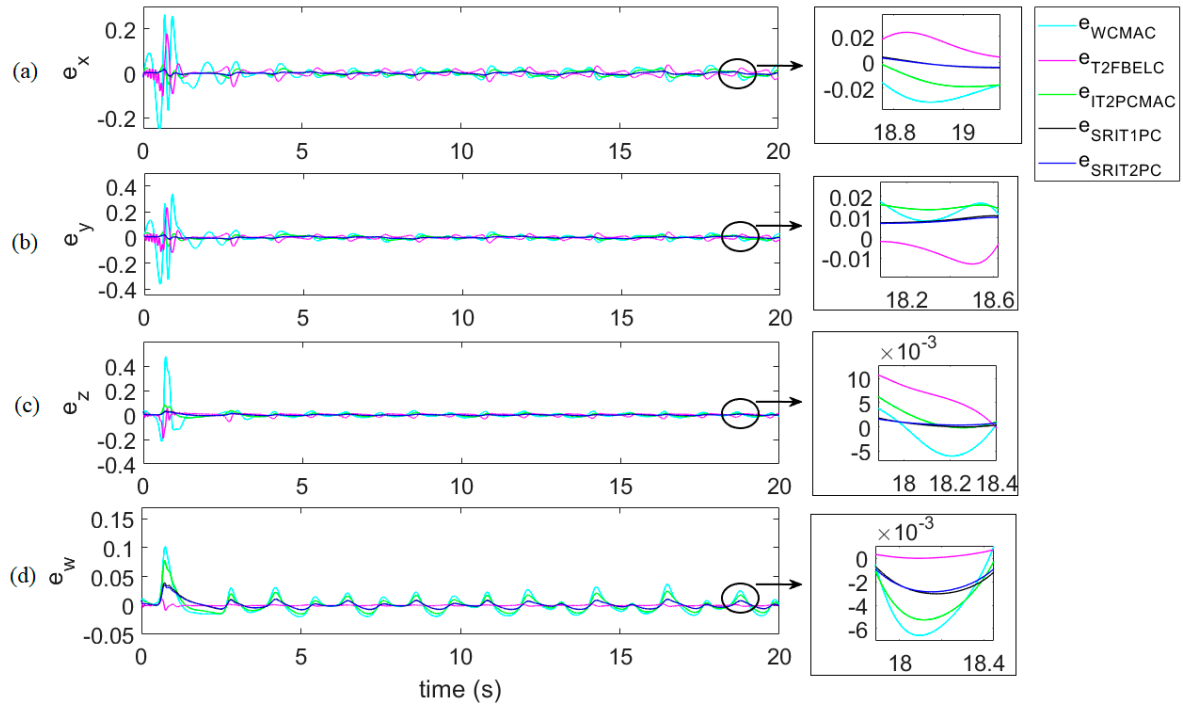

Figure 20. Tracking errors between the proposed SRIT2PC and other synchronization methods for Case 3: (**a**) e_x, (**b**) e_y, (**c**) e_z, and (**d**) e_w.

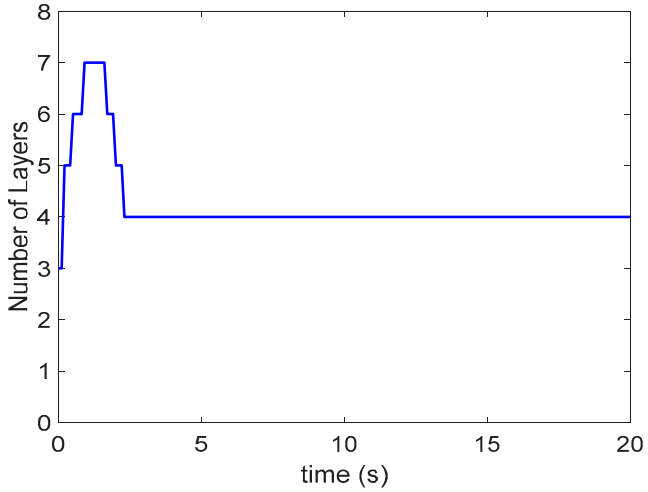

Figure 21. Number of layers using self-evolving algorithm for Case 3.

Case 4:

In this case, the parameter for defining the feature of the Lorenz–Stenflo chaotic-system attractor, θ, is given as a time-varying parameter ranging from zero to one during the control process. Therefore, the parameters α, τ, λ, φ, and γ are also time-varying parameters. The synchronization results of the 4D Lorenz–Stenflo chaotic system using the SRIT2PC are depicted in Figure 22; Figure 23 shows the trajectory signals, $\boldsymbol{x}_1(t)$, $\boldsymbol{y}_1(t)$, $\boldsymbol{z}_1(t)$, and $\boldsymbol{w}_1(t)$, and the synchronization outputs, $\boldsymbol{x}_2(t)$, $\boldsymbol{y}_2(t)$, $\boldsymbol{z}_2(t)$, and $\boldsymbol{w}_2(t)$; Figure 24 shows the control signals, $\boldsymbol{u}_x(t)$, $\boldsymbol{u}_y(t)$, $\boldsymbol{u}_z(t)$, and $\boldsymbol{u}_w(t)$; Figure 25 shows the tracking errors, $\boldsymbol{e}_x(t)$, $\boldsymbol{e}_y(t)$, $\boldsymbol{e}_z(t)$, and $\boldsymbol{e}_w(t)$. The number of layers of the SRIT2PC using the self-evolving algorithm is shown in Figure 26. In this case, the simulation results suggest that the proposed SRIT2PC controller can effectively synchronize the slave chaotic system with the master system.

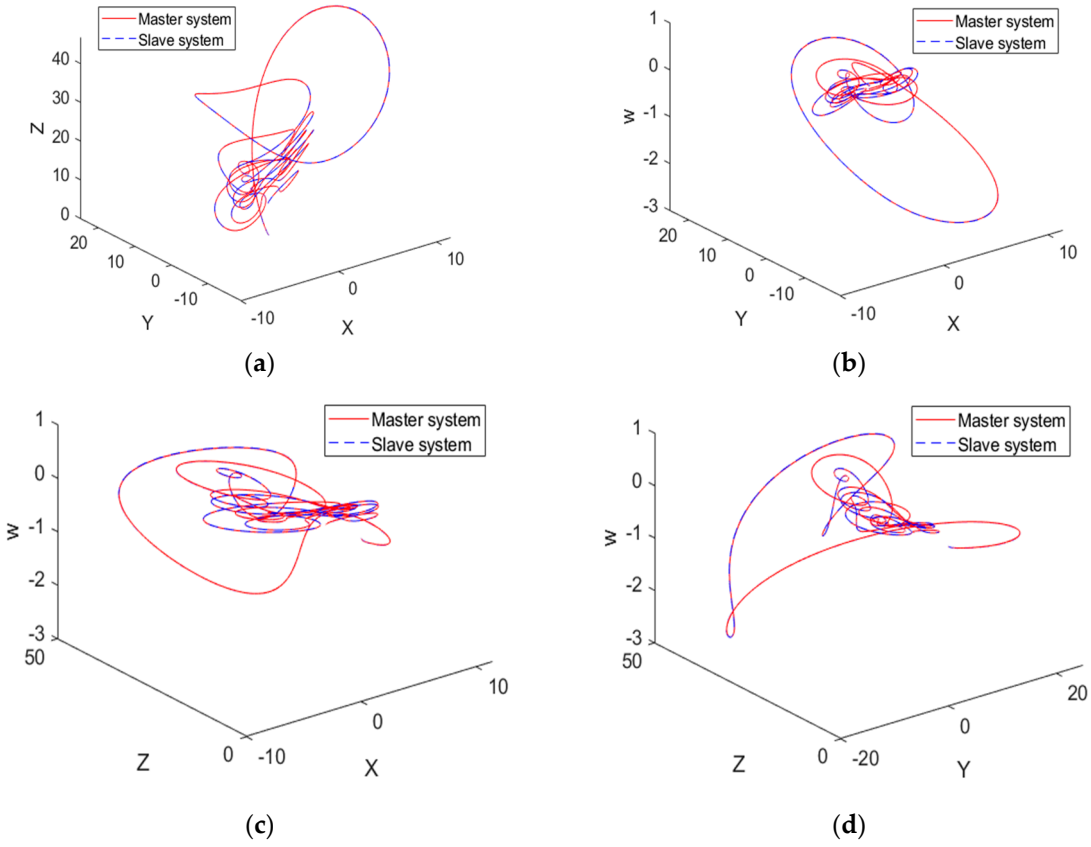

Figure 22. Synchronization of 4D Lorenz–Stenflo chaotic system using the SRIT2PC for Case 4 (**a**) x-y-z space, (**b**) x-y-w space, (**c**) x-z-w space, (**d**) y-z-w space.

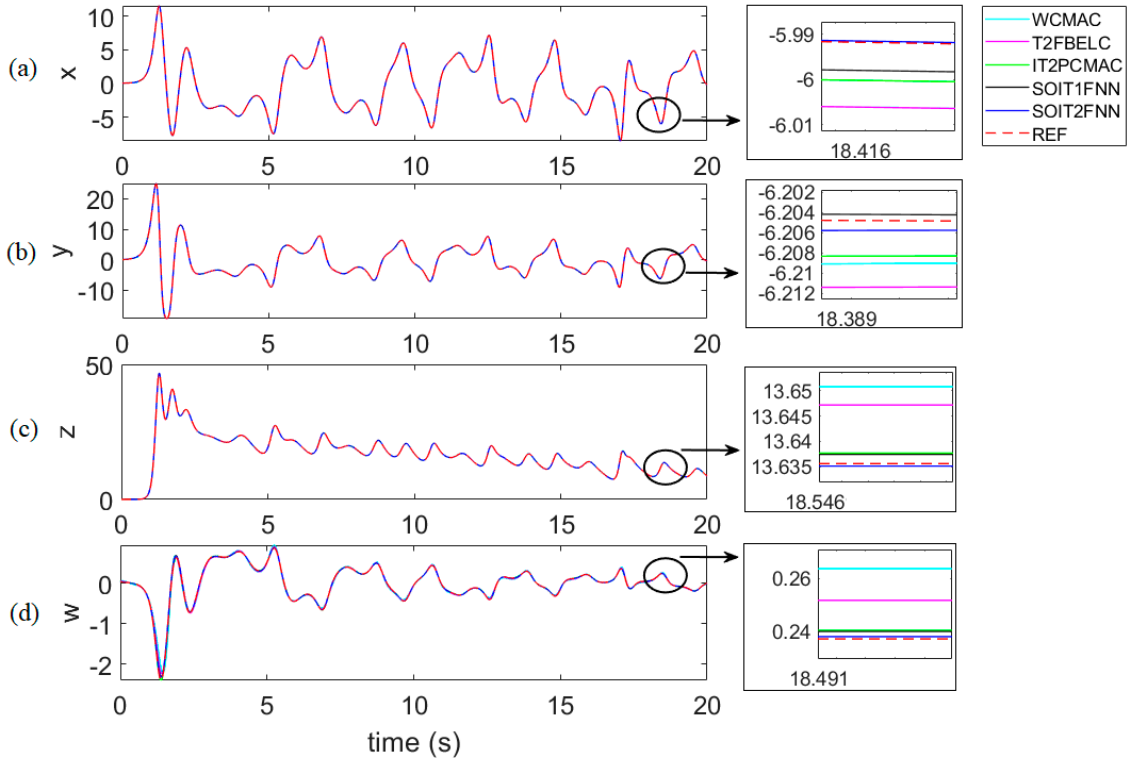

Figure 23. System outputs between the proposed SRIT2PC and other synchronization methods for Case 4: (**a**) x_1, x_2, (**b**) y_1, y_2, (**c**) z_1, z_2, and (**d**) w_1, w_2.

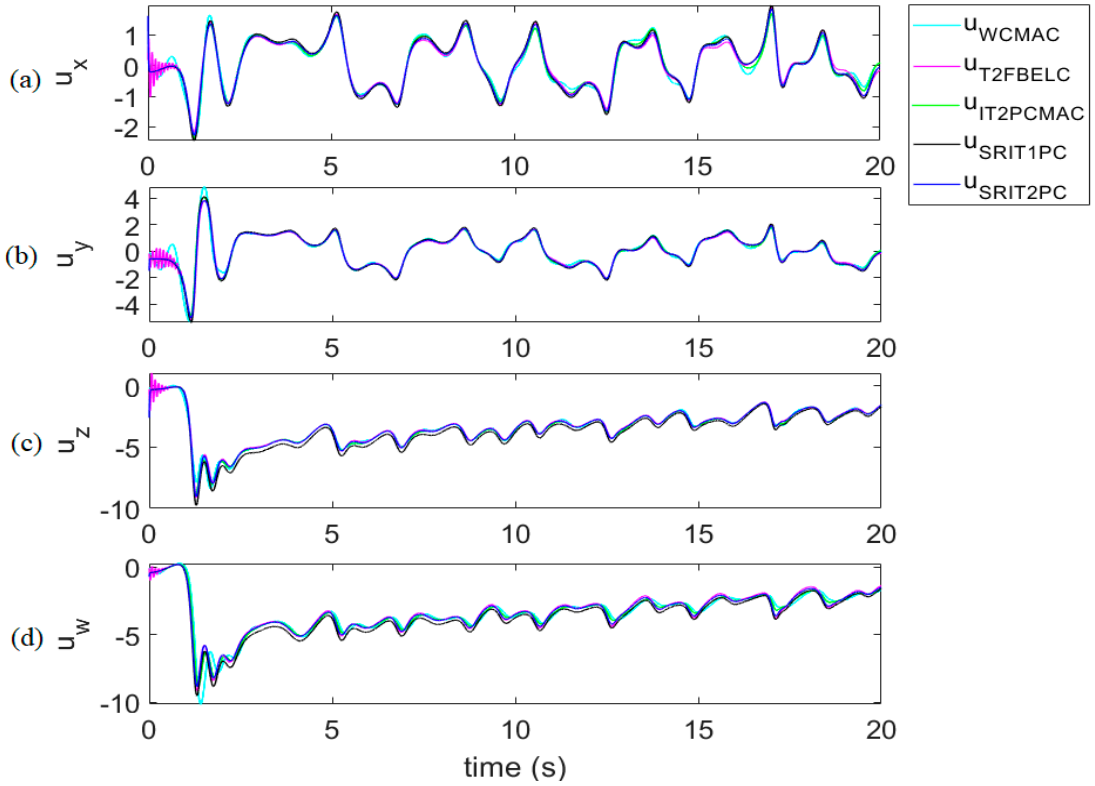

Figure 24. Control signals between the proposed SRIT2PC and other synchronization methods for Case 4: (**a**) u_x, (**b**) u_y, (**c**) u_z, and (**d**) u_w.

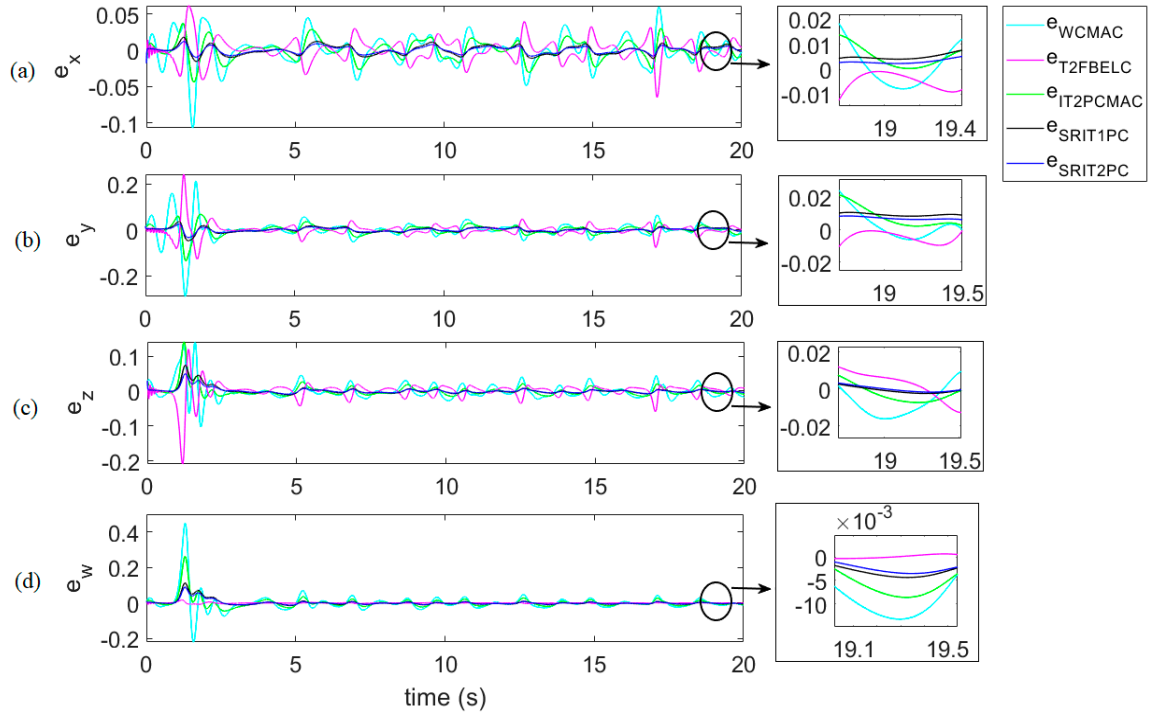

Figure 25. Tracking errors between the proposed SRIT2PC and other synchronization methods for Case 4: (**a**) e_x, (**b**) e_y, (**c**) e_z, and (**d**) e_w.

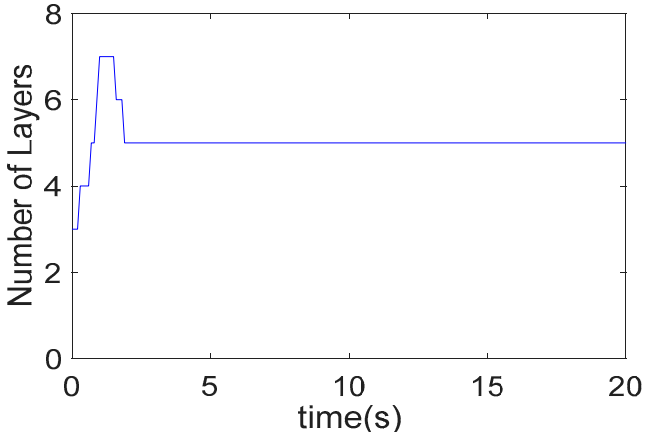

Figure 26. Number of layers using self-evolving algorithm for Case 4.

Figure 11, Figure 16, Figure 21, and Figure 26 show that, at the beginning of the control process, the structure of the proposed controller is in an adjusting period, after which point it quickly converges to a suitable number of layers. The simulation results for Case 4 suggest that, by using the online adaptive laws, the proposed controller can synchronize the chaotic systems effectively, even when θ is a time-varying parameter. In all cases studied, the proposed controller is superior for the synchronization of the 4D Lorenz–Stenflo chaotic system, since it has the fasted response and smallest RMSE tracking errors, even when faced with external disturbances and system uncertainties. Indeed, obtaining the appropriate threshold to generate and delete rules affects control-system performance. For instance, a small generating threshold generates a large number of rules and, contrarily, a large generating threshold will not generate many rules. The same can be said for the deleting threshold: if it is too small, minimal rules are removed and, contrarily, if it is too large, too many rules are removed. In this study, we used the trial-and-error method to obtain these thresholds.

Table 1. Comparison results in root mean square error (RMSE) of synchronization the 4-D Lorenz-Stenflo chaotic system.

Control Method	Computation Time (s)	Case 1 θ=0	Case 2 θ=0.8	Case 3 θ=1.0	Case 4 Time-Varying θ
WCMAC	0.0147	0.1481	0.1804	0.1498	0.1379
T2FBELC	0.0183	0.0902	0.0955	0.0602	0.0797
IT2PCMAC	0.0172	0.0524	0.0716	0.0486	0.0704
SRIT1PC	0.0145	0.0507	0.0422	0.0347	0.0431
SRIT2PC (proposed controller)	0.0196	0.0476	0.0366	0.0299	0.0322

5. Conclusions

In this paper, an adaptive SRIT2PC controller is proposed for the synchronization of 4D Lorenz–Stenflo chaotic systems. In doing so, we presented a new controller that can automatically update the parameters and structure based on the tracking error and the contribution of rules. The proposed controller has the following advantages: a dynamic threshold of PN, autonomous network constructing due to the self-evolving algorithm, type-2 fuzzy membership function, and recurrent-CMAC learning properties. The online adaptive laws of the control system were derived using the gradient-descent method; system stability was guaranteed using Lyapunov stability theory. Indeed, the numerical simulation results suggest that the proposed control system is highly effective. In the future, the estimation method will be applied to estimate the generating and deleting thresholds to achieve optimal control performance.

Author Contributions: Conceptualization, T.-L.L.; Data curation, T.-L.L., T.-T.H. and V.-Q.N.; Formal analysis, T.-L.L.; Investigation, T.-L.L.; Methodology, T.-L.L. and T.-T.H.; Project administration, S.-K.H.; Resources, T.-T.H.;

Software, V.-Q.N.; Supervision, C.-M.L. and S.-K.H.; Validation, T.-T.H. and V.-Q.N.; Writing—original draft, T.-L.L. and T.-T.H.; Writing—review & editing, T.-L.L., C.-M.L. and S.-K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This research was supported by the Competency Development Program for Industry Specialists of the Korean Ministry of Trade, Industry and Energy (MOTIE), operated by Korea Institute for Advancement of Technology (KIAT) (No. N0002431) and the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2019-0-01423) supervised by the IITP (Institute for Information and Communications Technology Promotion).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sadaoui, D.; Boukabou, A.; Merabtine, N.; Benslama, M. Predictive synchronization of chaotic satellites systems. *Expert Syst. Appl.* **2011**, *38*, 9041–9045. [CrossRef]
2. Naderi, B.; Kheiri, H. Exponential synchronization of chaotic system and application in secure communication. *Opt. Int. J. Light Electron Opt.* **2016**, *127*, 2407–2412. [CrossRef]
3. Pappu, C.S.; Flores, B.C.; Debroux, P.S.; Boehm, J.E. An electronic implementation of lorenz chaotic oscillator synchronization for bistatic radar applications. *IEEE Trans. Aerosp. Electron. Syst.* **2017**, *53*, 2001–2013. [CrossRef]
4. Jayaprasath, E.; Wu, Z.M.; Sivaprakasam, S.; Hou, Y.S.; Tang, X.; Lin, X.D.; Deng, T.; Xia, G.Q. Investigation of the Effect of Intra-Cavity Propagation Delay in Secure Optical Communication Using Chaotic Semiconductor Lasers. *Photonics* **2019**, *6*, 49. [CrossRef]
5. Mandal, M.K.; Das, A.K. Chaos-Based Colour Image Encryption Using Microcontroller ATMEGA 32. In *Nanoelectronics, Circuits and Communication Systems*; Lecture Notes in Electrical Engineering; Nath, V., Mandal, J., Eds.; Springer: Singapore, 2019; Volume 2019, p. 511.
6. Ohtsubo, J. Chaos Synchronization in Semiconductor Lasers. In *Semiconductor Lasers*; Springer Series in Optical Sciences; Springer: Cham, Switzerland, 2017; p. 111.
7. Xu, L.; Ma, H.; Xiao, S. Exponential Synchronization of Chaotic Lur'e Systems Using an Adaptive Event-Triggered Mechanism. *IEEE Access* **2018**, *6*, 61295–61304. [CrossRef]
8. Boulkroune, A.; Bouzeriba, A.; Bouden, T. Fuzzy generalized projective synchronization of incommensurate fractional-order chaotic systems. *Neurocomputing* **2016**, *173*, 606–614. [CrossRef]
9. Zhou, Q.; Chao, F.; Lin, C.M. A functional-link-based fuzzy brain emotional learning network for breast tumor classification and chaotic system synchronization. *Int. J. Fuzzy Syst.* **2018**, *20*, 349–365. [CrossRef]
10. Mufti, M.R.; Afzal, H.; Rehman, F.U.; Butt, Q.R.; Qureshi, M.I. Synchronization and antisynchronization between two non-identical Chua oscillators via sliding mode control. *IEEE Access* **2018**, *6*, 45270–45280. [CrossRef]
11. Mohammadzadeh, A.; Ghaemi, S. Optimal synchronization of fractional-order chaotic systems subject to unknown fractional order, input nonlinearities and uncertain dynamic using type-2 fuzzy CMAC. *Nonlinear Dyn.* **2017**, *88*, 2993–3002. [CrossRef]
12. Albus, J.S. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *J. Dyn. Syst. Meas. Control* **1975**, *97*, 220–227. [CrossRef]
13. Lin, C.M.; Le, T.L. WCMAC-based control system design for nonlinear systems using PSO. *J. Intell. Fuzzy Syst.* **2017**, *33*, 807–818. [CrossRef]
14. Lu, H.C.; Chuang, C.Y. Robust parametric CMAC with self-generating design for uncertain nonlinear systems. *Neurocomputing* **2011**, *74*, 549–562. [CrossRef]
15. Lin, C.M.; Li, H.Y. Self-organizing adaptive wavelet CMAC backstepping control system design for nonlinear chaotic systems. *Nonlinear Anal. Real World Appl.* **2013**, *14*, 206–223. [CrossRef]
16. Lin, C.M.; Huynh, T.T.; Le, T.L. Adaptive TOPSIS fuzzy CMAC back-stepping control system design for nonlinear systems. *Soft Comput.* **2019**, *23*, 6947–6966. [CrossRef]
17. Fang, W.; Chao, F.; Yang, L.; Lin, C.M.; Shang, C.; Zhou, C.; Shen, Q. A recurrent emotional CMAC neural network controller for vision-based mobile robots. *Neurocomputing* **2019**, *334*, 227–238. [CrossRef]

18. Wang, J.G.; Tai, S.C.; Lin, C.J. Medical diagnosis applications using a novel interactively recurrent self-evolving fuzzy CMAC model. In Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, 6–11 July 2014; pp. 4092–4098.
19. Chung, C.C.; Chen, T.S.; Lin, L.H.; Lin, Y.C.; Lin, C.M. Bankruptcy prediction using cerebellar model neural networks. *Int. J. Fuzzy Syst.* **2016**, *18*, 160–167. [CrossRef]
20. Guan, J.S.; Lin, L.Y.; Ji, G.L.; Lin, C.M.; Le, T.L.; Rudas, I.J. Breast tumor computer-aided diagnosis using self-validating cerebellar model neural networks. *Acta Polytech. Hung.* **2016**, *13*, 39–52.
21. Tsao, Y.; Chu, H.C.; Fang, S.H.; Lee, J.; Lin, C.M. Adaptive noise cancellation using deep cerebellar model articulation controller. *IEEE Access* **2018**, *6*, 37395–37402. [CrossRef]
22. Zhao, J.; Lin, C.M. Wavelet-TSK-type fuzzy cerebellar model neural network for uncertain nonlinear systems. *IEEE Trans. Fuzzy Syst.* **2018**, *27*, 549–558. [CrossRef]
23. Lin, C.M.; Yang, M.S.; Chao, F.; Hu, X.M.; Zhang, J. Adaptive filter design using type-2 fuzzy cerebellar model articulation controller. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 2084–2094. [CrossRef]
24. Wang, J.G.; Tai, S.C.; Lin, C.J. The application of an interactively recurrent self-evolving fuzzy CMAC classifier on face detection in color images. *Neural Comput. Appl.* **2018**, *29*, 201–213. [CrossRef]
25. Lin, C.M.; Hou, Y.L.; Chen, T.Y.; Chen, K.H. Breast nodules computer-aided diagnostic system design using fuzzy cerebellar model neural networks. *IEEE Trans. Fuzzy Syst.* **2013**, *22*, 693–699. [CrossRef]
26. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
27. Zhang, L.; Yang, G.H. Observer-based fuzzy adaptive sensor fault compensation for uncertain nonlinear strict-feedback systems. *IEEE Trans. Fuzzy Syst.* **2017**, *26*, 2301–2310. [CrossRef]
28. Zhang, L.; Yang, G.H. Low-computation Adaptive Fuzzy Tracking Control for Nonlinear Systems via Switching-Type Adaptive Laws. *IEEE Trans. Fuzzy Syst.* **2019**, *27*, 1931–1942. [CrossRef]
29. Wang, H.; Liu, P.X.; Zhao, X.; Liu, X. Adaptive Fuzzy Finite-Time Control of Nonlinear Systems with Actuator Faults. Available online: https://ieeexplore.ieee.org/abstract/document/8709959 (accessed on 2 November 2019).
30. Zhao, X.; Wang, X.; Zhang, S.; Zong, G. Adaptive neural backstepping control design for a class of nonsmooth nonlinear systems. *IEEE Trans. Syst. Man Cybern.* **2019**, *49*, 178–183. [CrossRef]
31. Lin, Y.C.; Wang, Y.C.; Chen, T.C.T.; Lin, H.F. Evaluating the Suitability of a Smart Technology Application for Fall Detection Using a Fuzzy Collaborative Intelligence Approach. *Mathematics* **2019**, *7*, 1097. [CrossRef]
32. Djeddi, A.; Dib, D.; Azar, A.T.; Abdelmalek, S. Fractional Order Unknown Inputs Fuzzy Observer for Takagi–Sugeno Systems with Unmeasurable Premise Variables. *Mathematics* **2019**, *7*, 984. [CrossRef]
33. Salamat, N.; Mustahsan, M.; Missen, M.M.S. Switching Point Solution of Second-Order Fuzzy Differential Equations Using Differential Transformation Method. *Mathematics* **2019**, *7*, 231. [CrossRef]
34. Shiev, K.; Ahmed, S.; Shakev, N.; Topalov, A.V. Trajectory control of manipulators using an adaptive parametric type-2 fuzzy cmac friction and disturbance compensator. In *Novel Applications of Intelligent Systems*; Springer: Berlin, Germany, 2016; pp. 63–82.
35. Zadeh, L.A. The concept of a linguistic variable and its application to approximate reasoning—I. *Inf. Sci.* **1975**, *8*, 199–249. [CrossRef]
36. Mendel, J.M. Type-2 fuzzy sets. In *Uncertain Rule-Based Fuzzy Systems*; Springer: Berlin, Germany, 2017; pp. 259–306.
37. Oh, S.K.; Jang, H.J.; Pedrycz, W. A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization. *Expert Syst. Appl.* **2011**, *38*, 11217–11229. [CrossRef]
38. Castillo, O.; Marroquín, R.M.; Melin, P.; Valdez, F.; Soria, J. 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.* **2012**, *192*, 19–38. [CrossRef]
39. Liang, Q.; Mendel, J.M. Interval type-2 fuzzy logic systems: Theory and design. *IEEE Trans. Fuzzy Syst.* **2000**, *8*, 535–550. [CrossRef]
40. Lee, C.H.; Chang, F.Y.; Lin, C.M. An efficient interval type-2 fuzzy CMAC for chaos time-series prediction and synchronization. *IEEE Trans. Cybern.* **2014**, *44*, 329–341. [CrossRef]

41. Chang, C.W.; Xiao, W.R.; Hsiao, C.C.; Chen, S.S.; Tao, C.W. A simplified interval type-2 fuzzy CMAC. In Proceedings of the Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), Otsu, Japan, 27–30 June 2017; pp. 1–4.
42. Lin, C.M.; La, V.H.; Le, T.L. DC–DC converters design using a type-2 wavelet fuzzy cerebellar model articulation controller. *Neural Comput. Appl.* **2018**. [CrossRef]
43. Zhao, T.; Ping, L.; Cao, J. Self-organising interval type-2 fuzzy neural network with asymmetric membership functions and its application. *Soft Comput.* **2019**, *23*, 7215–7228. [CrossRef]
44. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*; Prentice-Hall: Upper Saddle River, NJ, USA, 1981.
45. Looney, C.G. Fuzzy Petri nets for rule-based decisionmaking. *IEEE Trans. Syst. Man Cybern.* **1988**, *18*, 178–183. [CrossRef]
46. Rosdi, F.S.; Salim, S.; Mustafa, M.B. An FPN-based classification method for speech intelligibility detection of children with speech impairments. *Soft Comput.* **2019**, *23*, 2391–2408. [CrossRef]
47. Zhu, G.; Li, Z.; Wu, N. Model-based fault identification of discrete event systems using partially observed Petri nets. *Automatica* **2018**, *96*, 201–212. [CrossRef]
48. Lin, C.M.; Li, H.Y. Dynamic petri fuzzy cerebellar model articulation controller design for a magnetic levitation system and a two-axis linear piezoelectric ceramic motor drive system. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 693–699. [CrossRef]
49. Bibi, Y.; Bouhali, O.; Bouktir, T. Petri type 2 fuzzy neural networks approximator for adaptive control of uncertain non-linear systems. *IET Control Theory Appl.* **2017**, *11*, 3130–3136. [CrossRef]
50. Hansen, P.; Franco, P.; Kim, S.Y. Soccer ball recognition and distance prediction using fuzzy petri nets. In Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 7–9 July 2018; pp. 315–322.
51. Mejía, G.; Niño, K.; Montoya, C.; Sánchez, M.A.; Palacios, J.; Amodeo, L. A Petri Net-based framework for realistic project management and scheduling: An application in animation and videogames. *Comput. Oper. Res.* **2016**, *66*, 190–198. [CrossRef]
52. Juang, C.F.; Lin, C.T. A recurrent self-organizing neural fuzzy inference network. *IEEE Trans. Neural Netw.* **1999**, *10*, 828–845. [CrossRef]
53. Hsu, C.F.; Cheng, K.H. Recurrent fuzzy-neural approach for nonlinear control using dynamic structure learning scheme. *Neurocomputing* **2008**, *71*, 3447–3459. [CrossRef]
54. Yen, V.T.; Nan, W.Y.; Cuong, P.V. Recurrent fuzzy wavelet neural networks based on robust adaptive sliding mode control for industrial robot manipulators. *Neural Comput. Appl.* **2018**. [CrossRef]
55. Lin, F.J.; Lee, S.Y.; Chou, P.H. Intelligent integral backstepping sliding-mode control using recurrent neural network for piezo-flexural nanopositioning stage. *Asian J. Control* **2016**, *18*, 456–472. [CrossRef]
56. Sharma, R.; Kumar, V.; Gaur, P.; Mittal, A. An adaptive PID like controller using mix locally recurrent neural network for robotic manipulator with variable payload. *Isa Trans.* **2016**, *62*, 258–267. [CrossRef]
57. Wang, S.Y.; Liu, F.Y.; Chou, J.H. Applications on adaptive recurrent cerebellar model articulation controller for switched reluctance motor drive systems. In Proceedings of the International Symposium on Computer, Consumer and Control (IS3C), Xi'an, China, 4–6 July 2016; pp. 6–9.
58. Le, T.L.; Lin, C.M.; Huynh, T.T. Interval Type-2 Petri CMAC Design for 4D Chaotic System. Available online: https://ieeexplore.ieee.org/abstract/document/8823251 (accessed on 2 November 2019).
59. Le, T.L.; Lin, C.M.; Huynh, T.T. Self-evolving type-2 fuzzy brain emotional learning control design for chaotic systems using PSO. *Appl. Soft Comput.* **2018**, *73*, 418–433. [CrossRef]
60. Lin, C.M.; Le, T.L.; Huynh, T.T. Self-evolving function-link interval type-2 fuzzy neural network for nonlinear system identification and control. *Neurocomputing* **2018**, *275*, 2239–2250. [CrossRef]
61. Le, T.L. Self-organizing recurrent interval type-2 Petri fuzzy design for time-varying delay systems. *IEEE Access* **2018**, *7*, 10505–10514. [CrossRef]
62. Lin, C.M.; Le, T.L. PSO-self-organizing interval type-2 fuzzy neural network for antilock braking systems. *Int. J. Fuzzy Syst.* **2017**, *19*, 1362–1374. [CrossRef]
63. Rong, H.J.; Yang, Z.X.; Wong, P.K.; Vong, C.M.; Zhao, G.S. Self-evolving fuzzy model-based controller with online structure and parameter learning for hypersonic vehicle. *Aerosp. Sci. Technol.* **2017**, *64*, 1–15. [CrossRef]

64. Ge, D.; Zeng, X.J. A self-evolving fuzzy system which learns dynamic threshold parameter by itself. *Ieee Trans. Fuzzy Syst.* **2018**, *27*, 1625–1637. [CrossRef]
65. Le, T.L. Intelligent fuzzy controller design for antilock braking systems. *J. Intell. Fuzzy Syst.* **2019**, *36*, 3303–3315. [CrossRef]
66. Vincent, U. Synchronization of identical and non-identical 4-D chaotic systems using active control. *Chaossolitons Fractals* **2008**, *37*, 1065–1075. [CrossRef]