processes

MDPI

Article

A T-S Fuzzy Quaternion-Value Neural Network-Based Data-Driven Generalized Predictive Control Scheme for Mecanum Mobile Robot

Congjun Ma, Xiaoying Li, Guofei Xiang * and Songyi Dian

College of Electrical Engineering, Sichuan University, Chengdu 610065, China
* Correspondence: gfxiang@scu.edu.cn

Abstract: Four-mecanum-wheeled omnidirectional mobile robots (FMOMR) are widely used in many practical scenarios because of their high mobility and flexibility. However, the performance of trajectory tracking would be degenerated largely due to various reasons. To deal with this issue, this paper proposes a data-driven algorithm by using the T-S fuzzy quaternion-value neural network (TSFQVNN). TSFQVNN is tailored to obtain the controlled autoregressive integral moving average (CARIMA) model, and then the generalized predictive controller (GPC) is designed based on the CARIMA model. In this way, the spatial relationship between the three-dimensional pose coordinates can be preserved and training times can be reduced. Furthermore, the convergence of the proposed algorithm is verified by the Stone–Weierstrass theorem, and the convergence conditions of the algorithm are discussed . Finally, the proposed control scheme is applied to the three-dimensional (3D) trajectory tracking problem on the arc surface, and the simulation results prove the necessity and feasibility of the algorithm.

Keywords: T-S fuzzy quaternion-value neural network; generalized predictive control; data-driven method; mecanum-wheeled mobile robot

Citation: Ma, C.; Li, X.; Xiang, G.; Dian, S. A T-S Fuzzy Quaternion-Value Neural Network-Based Data-Driven Generalized Predictive Control Scheme for Mecanum Mobile Robot. *Processes* **2022**, *10*, 1964. https://doi.org/10.3390/pr10101964

Academic Editor: Wen-Jer Chang

Received: 31 August 2022
Accepted: 23 September 2022
Published: 29 September 2022

1. Introduction

Nowadays, mobile robots appear in many industrial applications, such as logistics [1], the chemical industry [2,3], shopping malls [4], and other fields. Compared with four-wheel car-like mobile robots, FMOMR , due to their high mobility and flexibility, have been widely used in soccer robots, nursing robots, mobile manipulators, etc. [5,6]. However, in practical applications, FMOMR would inevitably be affected by time-delays, nonlinear frictions, and unknown external disturbances, resulting in great performance degeneration in trajectory tracking. Therefore, there are great challenges to realize accurate trajectory tracking for FMOMR [7,8].

Usually, there are at least two methods to perform the modeling work, such as black box modeling and non-black box modeling. The latter requires a clear physical model or at least structure parameters. In [9–11], researchers created the model for FMOMR from the point of kinematics. However, the kinematics model is not accurate enough for the existence of uncertainty. Black box modeling methods can make up for the shortcomings of mechanism modeling, and they are suitable for nonlinear system modeling, such as neural network [12,13], even though some important information can be wasted or ignored. For example, there is plenty of empirical knowledge, and rule-based information cannot be utilized by neural networks for their structure characteristic. In addition, the tool of fuzzy logic has come in vogue for solving the problem of uncertainty and nonlinearity [14,15]. Generally, the fuzzy logic system does not obtain the ability to learn, but behaves well by combining with the neural networks, such as the fuzzy differential neural network [16], the fuzzy wavelet neural network [17], and the fuzzy recurrent wavelet neural network [18].

However, the huge computation and memory demands prevent the aforementioned fuzzy neural networks from being implemented in practical systems.

In addition, in many practical applications, the ability to move along the spherical or arc surface is required, which requires taking the dynamics of FMOMR in 3D space into account. Under these circumstances, the aforementioned learning-based methods are usually simplified to multiple multi-input–single-output (MISO) systems when they are used to approximate a multi-input–multi-output (MIMO) system, which would miss the spatially structured relations among those 3D pose coordinates [19]. Recently, the quaternion neural network has been regarded as an effective method to identify 3D or four-dimensional (4D) systems [20,21]. The core idea of the quaternion neural network is to use quaternion to represent 3D or 4D input information, so as to capture the inherent structured information through Hamilton product [19]. In the meantime, a quaternion weight linking two quaternion units only has four degrees of freedom, whereas the corresponding weight parameter of standard BP neural network is $4 \times 4 = 16$, i.e., a fourfold memory saving is achieved by using a quaternion neural network [22]. Because of these benefits, the quaternion neural network plays an important role in the field of speech recognition [23], color image compression [20], wind forecasting, etc. [24]. However, it still has the common shortcoming of the neural network, which does not have the ability to deal with uncertain information and cannot make effective use of empirical knowledge. Therefore, we consider a tailored fuzzy quaternion neural network to represent the dynamics of FMOMR with structured spatial information preserved. On the other hand, we can obtain a set of T-S fuzzy models through TSFQVNN. Then, because the consequent of the T-S fuzzy model is very similar to the CARIMA model, we can convert them into the CARIMA model. Finally, for the CARIMA model, a 3D trajectory tracking controller is designed by using multivariable generalized predictive control.

Compared to the existing works, the proposed approach has several advantages. (1) Those existing trajectory tracking control schemes for FMOMR in [7,8,25] could only realize trajectory tracking in a two-dimensional plane. This strategy realizes the trajectory tracking of FMOMR on the arc surface. (2) The proposed system identification method not only has the advantages of the fuzzy logic system and neural network, but also gives full play to the benefits of quaternion representation. In this way, the algorithm can identify 3D or 4D systems with better performance, and the number of weight updates is less than that of [12–18]. (3) The universal approximation of TSFQVNN is proven by the Stone–Weierstrass theorem, and its stability is analyzed by the Lyapunov theory. (4) Different from the existing fuzzy generalized predictive controller [26,27], the proposed CARIMA-based controller preserves the correlation between multi-dimensional pose coordinates. Simulation results show that the designed controller can obtain higher 3D trajectory tracking accuracy.

The rest of the paper is organized as follows. Section 2 presents a formulation of the problem. Section 3 introduces TSFQVNN. Section 4 presents the TSFQVNN-GPC. In Section 5, simulations for FMOMR 3D trajectory tracking based on TSFQVNN-GPC, are provided to show the effectiveness of the proposed method. Finally, Section 6 draws a conclusion.

2. Problem Description

In this paper, we consider the situation that the FMOMR could move on a circular arc or a spherical surface, which makes the FMOMR kinematic model a 4-input-3-output system. The input and output data come from a four rotate speed input for mecanum wheels and position output of the robot body along the x, y, and z axes among the world coordinate system. Figure 1 describes a scene in which FMOMR move on the internal surface of a cylindrical cavity. The robot mainly consists of a mobile robot body, mecanum wheels, and continuum manipulator. There are various functional devices equipped in the mobile robot body, and the four mecanum wheels can move in any direction on the surface. The continuum manipulator installed on the front of the robot body can achieve inspection operation in a spatially complex environment. At the same time, many uncertainties in

FMOMR lead to a complex and imprecise modeling process. This makes it impossible to guarantee that an effective motion controller can be used for FMOMR with different structural designs. Therefore, the "data-based" method can be used to control the motion of the mobile robot.

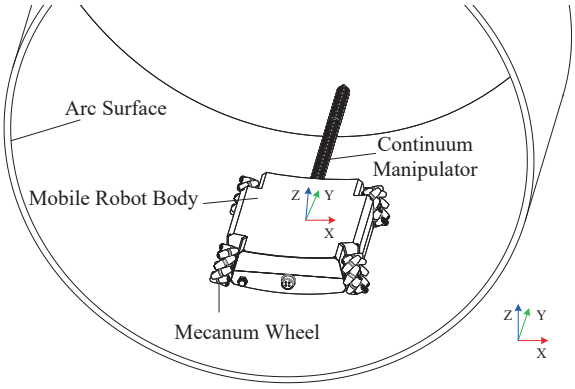

Figure 1. FMOMR in a cylindrical cavity.

First, to compensate for the lack of self-learning ability of the T-S fuzzy logic system, many researchers use the fuzzy neural network for system identification. Figure 2 describes the structure of the T-S fuzzy neural network (TSFNN). For the FMOMR system, the implementation of TSFNN requires the 4-input-3-output system to be divided into three 4-input-1-output systems. Finally, three T-S fuzzy models with 4-input-1-output can be obtained. The characteristics of T-S fuzzy model afterparts are helpful to build the CARIMA model, and then design the GPC controller based on the CARIMA model to realize FMOMR 3D trajectory tracking control.

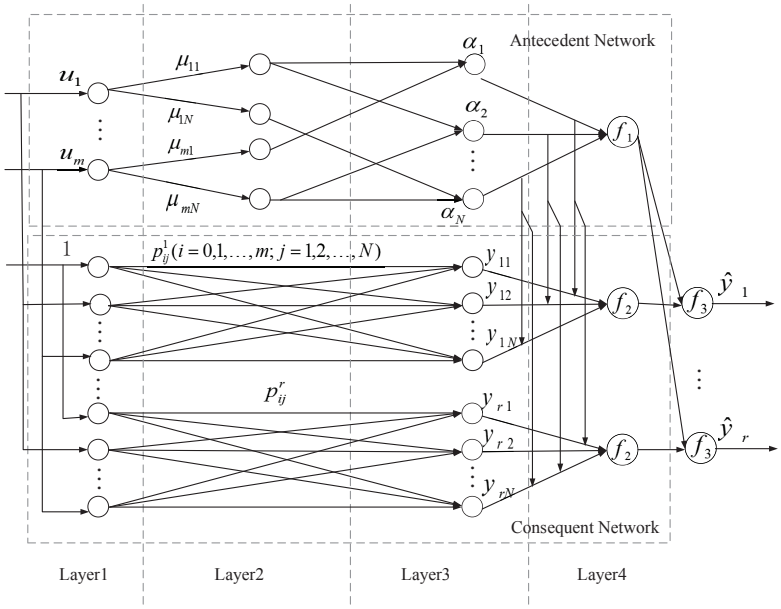

Figure 2. T-S fuzzy neural network.

Three T-S fuzzy models with 4-input-1-output can be obtained from TSFNN above.

$$\mathbb{R}_j^r : if \quad u \quad is \quad \mu_j \quad , Then \quad y_j^r(k+1) = (p_{0j}^r)^T + (p_{ij}^r)^T U^r \quad , \quad j = 1, \cdots, N, r = 1, 2, 3, \tag{1}$$

where $U^r = [y^r(k), y^r(k-1), \ldots, y^r(k-n_a+1), u(k-1), \ldots, u(k-n_b)]$ is the input and output variable. $\mu_j = [A_{1j}, \ldots, A_{n_aj}, B_{1j}, \ldots, B_{n_bj}]$ denotes the *jth* membership function. $(p_{ij}^r)^T = [a_{1j}^r, \ldots, a_{n_aj}^r, b_{1j}^r, \ldots, b_{n_bj}^r]^T$ represents the jth consequent parameter. It can be regarded as the parameter of the jth linear submodel of the rth T-S fuzzy model.

Then the CARIMA model is obtained by weighted summation of the T-S fuzzy model above, and finally, the 3D trajectory tracking controller is designed by GPC.

3. T-S Fuzzy Quaternion-Value Neural Network Identification

In this part, TSFQVNN and its learning algorithm are introduced, and it is proven that the fuzzy neural network model can approximate any nonlinear system with any accuracy. In addition, the stability analysis is achieved by the method of Lyapunov stability analysis.

3.1. T-S Fuzzy Quaternion-Value Neural Network

First of all, a network structure is supposed to be introduced of the TSFQVNN. It has four layers, and each input and output is a quaternion neuron in Figure 3, for the sake of simplicity. Therefore, the network structure looks more concise than the TSFNN structure, and the four-layer structure of TSFQVNN is identified with the TSFNN layers structure. In the antecedent network, we do not need to divide the MIMO system into subsystems according to its dimension of input variable, but adopt the quaternion method. Furthermore, the output of the consequent network can benefit from the quaternion as well. In this way, the traditional TSFNN with quaternion-value can handle with the trajectory tracking problem in 3D space well from the degree of structured information representation and memory saving. The detailed descriptions of the four layers are as follows.

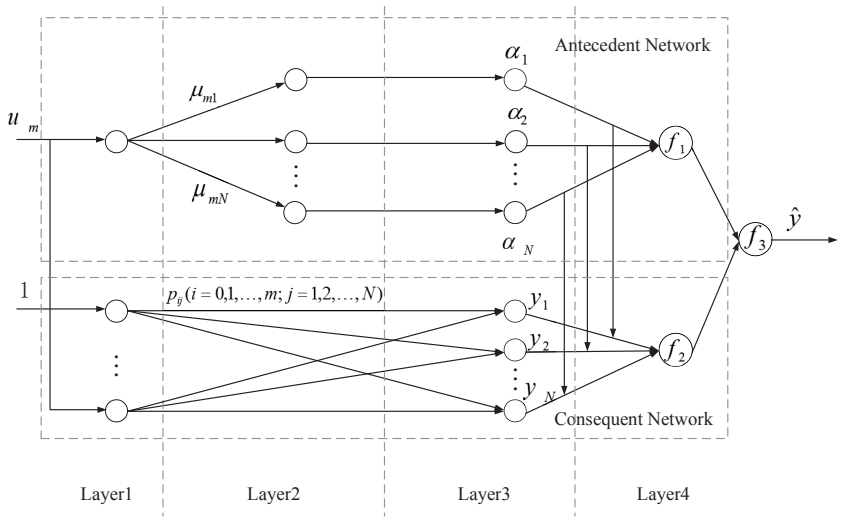

Figure 3. T-S fuzzy quaternion-value neural network.

- *Layer 1:* The first layer is defined by N input variable u_i:

$$u_i(k) = u_i^{Re} + \mathbf{i}u_i^{Im(i)} + \mathbf{j}u_i^{Im(j)} + \mathbf{k}u_i^{Im(k)}, \quad k = 1, \ldots, n, i = 1, \ldots, m, \tag{2}$$

  where m and n are the dimension of input vector and order of the system, respectively.
- *Layer 2:* The second layer fuzzifies the data from the first layer and gets the membership function $\mu_{ij}(j = 1, \ldots, N)$. N denotes the fuzzy partition number of u_i. Then, we use the Gaussian function as the split-activation function because the split-activation function can avoid a large number of singularities in the process of solving [22]. We have

$$\mu_{ij}(u_i) = exp\left[\frac{-((u_i(k)) - (c_{ij})^2)}{(\sigma_{ij}^2)}\right], \tag{3}$$

where the initial values c_{ij}^0 and σ_{ij}^0 of c_{ij} and σ_{ij} are calculated from the fuzzy c-means (FCM) methods [28]. The formula is as follows:

$$c_{ij}^0 = (c_{ij}^0)^{Re} + \mathbf{i}(c_{ij}^0)^{Im(i)} + \mathbf{j}(c_{ij}^0)^{Im(j)} + \mathbf{k}(c_{ij}^0)^{Im(k)}, \tag{4}$$

$$\sigma_{ij}^0 = (\sigma_{ij}^0)^{Re} + \mathbf{i}(\sigma_{ij}^0)^{Im(i)} + \mathbf{j}(\sigma_{ij}^0)^{Im(j)} + \mathbf{k}(\sigma_{ij}^0)^{Im(k)}. \tag{5}$$

The following is the real part for example, and the rest can be obtained in this way:

$$(c_{ij}^0)^{Re} = \frac{\sum_{k=1}^{N}(\mu_{ij}^{Re}(k))^{\varrho} u_i^{Re}(k)}{\sum_{k=1}^{N}(\mu_{ij}^{Re}(k))^{\varrho}}, \tag{6}$$

$$(\sigma_{ij}^0)^{Re} = \sqrt{\frac{\sum_{k=1}^{N}\mu_{ij}^{Re}(k)(u_i^{Re}(k) - (c_{ij}^0)^{Re}(k))^2}{\sum_{k=1}^{N}\mu_{ij}^{Re}(k)}}, \tag{7}$$

where $\varrho \epsilon (1, \infty)$ is a weighted exponent.

- *Layer 3:* Each node in the antecedent network represents a fuzzy rule. The fuzzy method of single point fuzzy set is adopted for the input value, namely

$$\alpha_j = \prod_{i=1}^{m}\mu_{ij}^{Re} + \mathbf{i}\prod_{i=1}^{m}\mu_{ij}^{Im(i)} + \mathbf{j}\prod_{i=1}^{m}\mu_{ij}^{Im(j)} + \mathbf{k}\prod_{i=1}^{m}\mu_{ij}^{Im(k)}. \tag{8}$$

$$y_j = y_j^{Re} + \mathbf{i}y_j^{Im(i)} + \mathbf{j}y_j^{Im(j)} + \mathbf{k}y_j^{Im(k)}, \tag{9}$$

where $y_j^{Re} = p_{0j}^{Re} + p_{1j}^{Re}u_1^{Re} + \ldots + p_{mj}^{Re}u_m^{Re}$.

- *Layer 4:* The fourth layer is the output layer and contains three functions, f_1, f_2, f_3. The function f_1 sums the fitness of the fuzzy antecedent, and then the function f_2 and f_3 can calculate the output of the system. We have

$$f_1 = \sum_{j=1}^{N}\alpha_j^{Re} + \mathbf{i}\sum_{j=1}^{N}\alpha_j^{Im(i)} + \mathbf{j}\sum_{j=1}^{N}\alpha_j^{Im(j)} + \mathbf{k}\sum_{j=1}^{N}\alpha_j^{Im(k)}, \tag{10}$$

$$f_2 = \sum_{j=1}^{N}\alpha_j y_j, \tag{11}$$

$$f_3 = \frac{f_2^{Re}}{f_1^{Re}} + \mathbf{i}\frac{f_2^{Im(i)}}{f_1^{Im(i)}} + \mathbf{j}\frac{f_2^{Im(j)}}{f_1^{Im(j)}} + \mathbf{k}\frac{f_2^{Im(k)}}{f_1^{Im(k)}}. \tag{12}$$

Consequently the system forecast output is

$$\begin{aligned}\hat{y} = &\frac{\sum_{j=1}^{N}(\alpha_j)^{Re}(y_j)^{Re}}{\sum_{j=1}^{N}(\alpha_j)^{Re}} + \mathbf{i}\frac{\sum_{j=1}^{N}(\alpha_j)^{Im(i)}(y_j)^{Im(i)}}{\sum_{j=1}^{N}(\alpha_j)^{Im(i)}} \\ &+ \mathbf{j}\frac{\sum_{j=1}^{N}(\alpha_j)^{Im(j)}(y_j)^{Im(j)}}{\sum_{j=1}^{N}(\alpha_j)^{Im(j)}} + \mathbf{k}\frac{\sum_{j=1}^{N}(\alpha_j)^{Im(k)}(y_j)^{Im(k)}}{\sum_{j=1}^{N}(\alpha_j)^{Im(k)}}.\end{aligned} \tag{13}$$

The T-S fuzzy model composed of N rules can be obtained from TSFQVNN above. We have

$$\begin{aligned}\mathbb{R}_j : if \quad u \quad is \quad \mu_j \quad , Then \quad y_j(k+1) = (p_{0j})^T + (p_{ij})^T U \quad , \\ j = 1, \cdots, N,\end{aligned} \tag{14}$$

where $U = [y(k), y(k-1), \ldots, y(k-n_a+1), u(k-1), \ldots, u(k-n_b)]$ is the input and output variable. $\mu_j = [A_{1j}, \ldots, A_{n_aj}, B_{1j}, \ldots, B_{n_bj}]$ is the jth membership function. $(p_{ij})^T = [a_{1j}, \ldots, a_{n_aj}, b_{1j}, \ldots, b_{n_bj}]^T$ is the jth consequent parameter. It can be regarded as the parameter of the jth linear submodel.

Remark 1. *For 3D and 4D systems, TSFNN needs to update the weight three or four times, while TSFQVNN only needs to update the weight once. This makes TSFQVNN identification faster and less computational.*

3.2. The Learning Algorithm

The rules for TSFQVNN learning is defined. Take the error cost function as

$$E = \frac{1}{2}\sum_{l=1}^{M} \mid y_l - \hat{y}_l \mid^2, \tag{15}$$

where y_i is reference output, $\hat{y}_i$ represents forecast output, M denotes the number of output neurons, and $|x| \stackrel{def}{=} \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}$ for a quaternion $x = x_1 + \mathbf{i}x_2 + \mathbf{j}x_3 + \mathbf{k}x_4$.

For a sufficiently small learning constant $\varepsilon > 0$, and the thresholds are supposed to be updated according to the following equations:

$$\Delta p_{0j} \stackrel{def}{=} -\varepsilon\left(\frac{\partial E}{\partial (p_{0j})^{Re}} + \mathbf{i}\frac{\partial E}{\partial p_{0j}^{Im(i)}} + \mathbf{j}\frac{\partial E}{\partial p_{0j}^{Im(j)}} + \mathbf{k}\frac{\partial E}{\partial p_{0j}^{Im(k)}}\right), \tag{16}$$

$$\Delta p_{ij} \stackrel{def}{=} -\varepsilon\left(\frac{\partial E}{\partial p_{ij}^{Re}} + \mathbf{i}\frac{\partial E}{\partial p_{ij}^{Im(i)}} + \mathbf{j}\frac{\partial E}{\partial p_{ij}^{Im(j)}} + \mathbf{k}\frac{\partial E}{\partial p_{ij}^{Im(k)}}\right), \tag{17}$$

where Δ denotes the correction of a parameter, ε is the learning rate, and because of the complicated relationship between its size, speed and accuracy of network convergence, it is necessary to adjust the learning rate through multiple tests.

The above Equations (16) and (17) can be expressed as:

$$\Delta dp_{ij} = \overline{f}_2 \Delta dp_{0j}, \quad \Delta dp_{0j} = \varepsilon e_l (1 - f_3) f_3, \tag{18}$$

where $e_l = e_l^{Re} + \mathbf{i}e_l^{Im(i)} + \mathbf{j}e_l^{Im(j)} + \mathbf{k}e_l^{Im(k)} = y_l - \hat{y}_l$ denotes the error between $\hat{y}_l$ and the target output signal y_l. We have

$$\Delta p_{ij} = \overline{u}_j \Delta p_{0j}, \tag{19}$$

$$\begin{aligned}
\Delta p_{0j} = {} & \left(1 - f_1^{Re}\right) f_1^{Re} \cdot Re\left[\sum_{j=1}^{N} (\Delta dp_{0j} \times \Delta \overline{dp}_{ij})\right] \\
& + \mathbf{i}\left(1 - f_1^{Im(i)}\right) f_1^{Im(i)} \cdot Im(i)\left[\sum_{j=1}^{N} (\Delta dp_{0j} \times \Delta \overline{dp}_{ij})\right] \\
& + \mathbf{j}\left(1 - f_1^{Im(j)}\right) f_1^{Im(j)} \cdot Im(j)\left[\sum_{j=1}^{N} (\Delta dp_{0j} \times \Delta \overline{dp}_{ij})\right] \\
& + \mathbf{k}\left(1 - f_1^{Im(k)}\right) f_1^{Im(k)} \cdot Im(k)\left[\sum_{j=1}^{N} (\Delta dp_{0j} \times \Delta \overline{dp}_{ij})\right],
\end{aligned} \tag{20}$$

where $\overline{x} \stackrel{def}{=} x_1 - \mathbf{i}x_2 - \mathbf{j}x_3 - \mathbf{k}x_4$, $Re[x] \stackrel{def}{=} x_1$, $Im(i)[x] \stackrel{def}{=} x_2$, $Im(j)[x] \stackrel{def}{=} x_3$, $Im(k)[x] \stackrel{def}{=} x_4$ for a quaternion $x = x_1 + \mathbf{i}x_2 + \mathbf{j}x_3 + \mathbf{k}x_4$.

Therefore, by the Formulas (18)–(20), Δp_{0j} and Δp_{ij} can be simply expressed as follows, respectively. We have

$$\Delta p_{0j} = (1 - f_1) f_1 \cdot \varepsilon e_l \sum_{j=1}^{N} (1 - f_3) f_3 \times \Delta \overline{dp}_{ij} = \varepsilon e_l f_4, \tag{21}$$

$$\Delta p_{ij} = \overline{u}_j e_l f_4 = \varepsilon e_l f_5. \tag{22}$$

In addition, the center c_{ij} and width σ_{ij} of membership function are further modified . The gradient descent algorithm is applied to the modification of parameters, namely:

$$\Delta c_{ij} \stackrel{def}{=} -\varepsilon \left(\frac{\partial E}{\partial c_{ij}^{Re}} + \mathbf{i} \frac{\partial E}{\partial c_{ij}^{Im(i)}} + \mathbf{j} \frac{\partial E}{\partial c_{ij}^{Im(j)}} + \mathbf{k} \frac{\partial E}{\partial c_{ij}^{Im(k)}} \right), \tag{23}$$

$$\Delta \sigma_{ij} \stackrel{def}{=} -\varepsilon \left(\frac{\partial E}{\partial \sigma_{ij}^{Re}} + \mathbf{i} \frac{\partial E}{\partial \sigma_{ij}^{Im(i)}} + \mathbf{j} \frac{\partial E}{\partial \sigma_{ij}^{Im(j)}} + \mathbf{k} \frac{\partial E}{\partial \sigma_{ij}^{Im(k)}} \right), \tag{24}$$

where j denotes the jth rule. i represents the ith input vector.

Remark 2. *Compared with the gradient descent method used in TSFNN weight updating, the Hamilton product used in TSFQVNN's gradient descent method can capture these internal latent relations within the features of a quaternion and restore the spatial relations within 3D coordinates.*

3.3. Proof of Global Approximation

Due to the application of fuzzy neural network in modeling work, it is necessary to prove its global approximation property.

Formula (13) can be treated as the set of fuzzy systems, named Y, and then Stone–Weierstrass theorem [29] is used to prove the global approximation ability of the model. First, it is proved that y satisfies the three conditions given in the Stone–Weierstrass theorem.

Lemma 1. (Y, d_∞) *is an algebra.*

Proof. Assume f_{11}, $f_{22} \epsilon Y$, which are described in detail as follows:

$$f_{11}(u) = d_1 + \mathbf{i} d_2 + \mathbf{j} d_3 + \mathbf{k} d_4 \tag{25}$$

$$f_{22}(u) = t_1 + \mathbf{i} t_2 + \mathbf{j} t_3 + \mathbf{k} t_4 \tag{26}$$

where $d_1 = \frac{\sum_{j=1}^{N_1} (\alpha_j^1)^{Re} (y_j^1)^{Re}}{\sum_{j=1}^{N_1} (\alpha_j^1)^{Re}}$, $d_2 = \frac{\sum_{j=1}^{N_1} (\alpha_j^1)^{Im(i)} (y_j^1)^{Im(i)}}{\sum_{j=1}^{N_1} (\alpha_j^1)^{Im(i)}}$, $d_3 = \frac{\sum_{j=1}^{N_1} (\alpha_j^1)^{Im(j)} (y_j^1)^{Im(j)}}{\sum_{j=1}^{N_1} (\alpha_j^1)^{Im(j)}}$,
$d_4 = \frac{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(k)} (y_j^2)^{Im(k)}}{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(k)}}$. $t_1 = \frac{\sum_{j=1}^{N_2} (\alpha_j^2)^{Re} (y_j^2)^{Re}}{\sum_{j=1}^{N_2} (\alpha_j^2)^{Re}}$, $t_2 = \frac{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(i)} (y_j^2)^{Im(i)}}{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(i)}}$,
$t_3 = \frac{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(j)} (y_j^2)^{Im(j)}}{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(j)}}$, $t_4 = \frac{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(k)} (y_j^2)^{Im(k)}}{\sum_{j=1}^{N_2} (\alpha_j^2)^{Im(k)}}$.

Thereupon:

$$\begin{aligned} f_{11}(u) \times f_{22}(u) = {} & (d_1 t_1 - d_2 t_2 - d_3 t_3 - d_4 t_4) + \mathbf{i}(d_2 t_1 + d_1 t_2 + d_3 t_4 - d_4 t_3) \\ & + \mathbf{j}(d_3 t_1 + d_1 t_3 + d_4 t_2 - d_2 t_4) + \mathbf{k}(d_4 t_1 + d_1 t_4 + d_2 t_3 - d_3 t_2). \end{aligned} \tag{27}$$

Because $\mu_{ij}^1(u_i)$ and $\mu_{ij}^2(u_i)$ are of Gaussian type, their product α_j^1 and α_j^2 are also of Gaussian type. Therefore, the form of Equation (27) is completely equivalent to that of Equation (13). It can be seen that

$$f_{11}(u) \times f_{22}(u) \epsilon Y. \tag{28}$$

Furthermore, we can also get $qf_{11}(u)\epsilon Y$ and $f_{11}(u) + f_{22}(u)\epsilon Y$. So far, Lemma 1 is proven.

Lemma 2. *(Y, d_∞) can separate points on U.*

Proof. Construct a f to prove the above conditions, that is, given the number of fuzzy sets defined on U and R, the Gauss membership function parameters, the rules number, and the rules expression, so that the f (in the form of the Formula (13)) obtained has the following characteristics:

(1) For any given u^0, $u1^0 \epsilon U$, when $u^0 \neq u1^0$, there is $f(u^0) \neq f(u1^0)$.

Set $u^0 = (u_1^0, \ldots, u_m^0)$, $u1^0 = (u1_1^0, \ldots, u1_m^0)$. If $u_i^0 \neq u1_i^0$, two fuzzy sets $(A_i^1, \mu_{A_i}^1)$ and $(A_i^2, \mu_{A_i}^2)$ are defined in the ith subspace of U, and their corresponding membership functions are: $\mu_{A_i}^1(u_i) = exp\left(-\frac{\left((u_i)-(u_i^0)\right)^2}{2}\right)$, $\mu_{A_i}^2(u_i) = exp\left(-\frac{\left((u_i)-(u1_i^0)\right)^2}{2}\right)$.

If $u^0 = u1^0$, there are $A_i^1 = A_i^2$ and $\mu_{A_i}^1 = \mu_{A_i}^2$. Two fuzzy sets (B_i^1, μ_B^1) and (B_i^2, μ_B^2) are defined on the output domain R, and their corresponding membership functions are as follows:

$$\mu_B^j(u_i) = exp\left(-\frac{(y - y_j)^2}{2}\right), \tag{29}$$

where $j = 1, 2$, y_j .

Then we get the function f with the form shown in formula (30), and its expression is as follows:

$$\mu_{A_i}^1(u^0) = 1 + \mathbf{i} + \mathbf{j} + \mathbf{k}, \tag{30}$$

$$\mu_{A_i}^1(u1^0) = exp\left(-\frac{(u1^0 - u_i^0)^2}{2}\right). \tag{31}$$

The same can be get $\mu_{A_i}^2(u^0)$ and $\mu_{A_i}^2(u1^0)$. Then:

$$f(u^0) = y_1 + Q(y_2)^{Re} + R(y_2)^{Im(i)} + S(y_2)^{Im(j)} + T(y_2)^{Im(k)}, \tag{32}$$

$$f(u1^0) = Q(y_1)^{Re} + R(y_1)^{Im(i)} + S(y_1)^{Im(j)} + T(y_1)^{Im(k)} + y_2, \tag{33}$$

Suppose $f(u^0) = f(u1^0)$. Then we have

$$\begin{aligned} f(u^0) = f(u1^0) \Rightarrow\ & (1-Q)((y_1)^{Re} - (y_2)^{Re}) + (1-R)((y_1)^{Im(i)} - (y_2)^{Im(i)}) \\ & + (1-S)((y_1)^{Im(j)} - (y_2)^{Im(j)}) + (1-T)((y_1)^{Im(k)} - (y_2)^{Im(k)}) = 0. \end{aligned} \tag{34}$$

Then, we can take $y_1 = 2 + 2\mathbf{i} + 2\mathbf{j} + 2\mathbf{k}$, $y_2 = 1 + \mathbf{i} + \mathbf{j} + \mathbf{k}$, and there must be

$$\begin{aligned} & (1-Q)((y_1)^{Re} - (y_2)^{Re}) + (1-R)((y_1)^{Im(i)} - (y_2)^{Im(i)}) \\ & + (1-S)((y_1)^{Im(j)} - (y_2)^{Im(j)}) + (1-T)((y_1)^{Im(k)} - (y_2)^{Im(k)}) \neq 0, \end{aligned}$$

which contradicts the hypothesis, and $f(u^0) = f(u1^0)$ is proven.

Lemma 2 proves the end.

Lemma 3. *All points on (Y, d_∞) are not zero.*

Proof. Observing formula (13) carefully, just select $y_j > 0, j = 1, 2, \ldots, N$. Any corresponding $f\epsilon Y$ can be used as the required f.

This lemma has been proven.

Theorem 1. *The TSFQVNN shown in Figure 2 is a global approximator.*

Proof. Y is a continuous and real function from Formula (25). It is easy to deduce the global approximation ability through the lemmas mentioned above.

3.4. Stability Analysis

The direct method of Lyapunov stability analysis is a universal and effective method to analyze the stability and convergence. There are two ways to derive it; one is the convergence of the weight vector, and the other is the convergence of the output error. The latter method is used here.

In order to obtain the stability conditions of the learning algorithm, the Lyapunov function is selected as follows:

$$V(k) = \frac{1}{2}\bar{e}_l(k)e_l(k) = \frac{1}{2}\overline{[y_l(k) - \hat{y}_l(k)]}[y_l(k) - \hat{y}_l(k)], \tag{35}$$

where $y_l(k)$ is the lth reference output, $\hat{y}_l(k)$ denotes the lth forecasted output.

The difference of Lyapunov function from k step to $k+1$ step is

$$\Delta V(k) = V(k+1) - V(k) = \frac{1}{2}[\bar{e}_l(k+1)e_l(k+1) - \bar{e}_l(k)e_l(k)]. \tag{36}$$

We assume that the reference output $y(k)$ is constant; then

$$e_l(k) = y_l(k) - \hat{y}_l(k) = y_l(k) - \frac{\sum_{j=1}^{N}[\alpha_j \sum_{j=1}^{N}(p_{0j}(k) + p_{ij}(k)u_i)]}{\sum_{j=1}^{N}\alpha_j}. \tag{37}$$

From Equation (1) minus Equation (2), we have

$$\Delta e_l(k) = e_l(k+1) - e_l(k) = -\varepsilon\frac{\sum_{j=1}^{N}[\alpha_j \sum_{j=1}^{N}(e_l f_4 + e_l f_5 u_i)]}{\sum_{j=1}^{N}\alpha_j}. \tag{38}$$

Consequently, we have

$$\begin{aligned}\Delta V(k) &= \frac{1}{2}[\bar{e}_l(k+1)e_l(k+1) - \bar{e}_l(k)e_l(k)] = \frac{1}{2}[(\bar{e}_l(k) - \Delta\bar{e}_l(k))(e_l(k) + \Delta e_l(k)) - \bar{e}_l(k)e_l(k)] \\ &= -\frac{\varepsilon\bar{e}_l(k)e_l(k)\sum_{j=1}^{N}[\alpha_j \sum_{j=1}^{N}(f_4 + f_5 u_i)]}{\sum_{j=1}^{N}\alpha_j} \cdot \left[1 - \frac{\varepsilon\sum_{j=1}^{N}[\alpha_j \sum_{j=1}^{N}(f_4 + f_5 u_i)]}{2\sum_{j=1}^{N}\alpha_j}\right].\end{aligned} \tag{39}$$

Because $V(k)$ is positive definite, it can be seen from Lyapunov stability theorem that to make the learning process convergent and stable, $\Delta V(k)$ must be positive-definite, that is, $\Delta V(k) > 0$. That is to say, the learning rate ε must meet the following requirements:

$$0 < \varepsilon < \frac{2\sum_{j=1}^{N}\alpha_j}{\sum_{j=1}^{N}[\alpha_j \sum_{j=1}^{N}(f_4 + f_5 u_i)]}. \tag{40}$$

4. Fuzzy Predictive Control Algorithm

The multi-step T-S fuzzy model can be obtained from Formula (14) in section III:

$$y(k+1) = \sum_{t=1}^{n_a}\overline{a}_t(k)y(k-t+1) + \sum_{t=1}^{n_b}\overline{b}_t(k)u(k-t+1) + \overline{p}(k), \tag{41}$$

where, $\overline{a}_t(k) = \frac{\sum_{j=1}^{N}\alpha_j a_{jt}}{\sum_{j=1}^{N}\alpha_j}$, $\overline{b}_t(k) = \frac{\sum_{j=1}^{N}\alpha_j b_{jt}}{\sum_{j=1}^{N}\alpha_j}$, $\overline{p}(k) = \frac{\sum_{j=1}^{N}\alpha_j p_{0j}}{\sum_{j=1}^{N}\alpha_j}$, and each of these elements is a quaternion.

The CARIMA model can be used to obtain

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \overline{p}, \tag{42}$$

where $A(z^{-1})$ and $B(z^{-1})$ are as follows:

$$\begin{cases} A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}, \\ B(z^{-1}) = b_0 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}. \end{cases}$$

Then the control variable of N_u is searched to minimize the error.

According to Section III, the CARIMA model of MIMO system can be obtained, and its objective function can be expressed as

$$J = \sum_{j=1}^{N_p} \|y(k+j) - w(k+j)\|^2 + \sum_{j=1}^{N_u} \|\Delta u(k+j-1)\|_\lambda^2, \tag{43}$$

where $y(k+j)$ represents the output after j step, and $w(k+j)$ denotes the desired trajectory. N_p and N_u are the prediction time domain and the control time domain, respectively. After N_u step, u will not change; λ is the control weighted matrix $n \times n$, namely $\lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Introducing the polynomial equation of diophantine matrix: $I = E_j(z^{-1})A(z^{-1})\Delta(z^{-1}) + z^{-j}F_j(z^{-1})$, $E_j(z^{-1})B(z^{-1}) = G_j(z^{-1}) + z^{-j}H_j(z^{-1})$, where $j = 1, \ldots, N_p$,

$$\begin{cases} E_j(z^{-1}) = E_0 + E_1 z^{-1} + \ldots + E_{j-1} z^{-(j-1)}, \\ F_j(z^{-1}) = F_0^j + F_1^j z^{-1} + \ldots + F_{n_a}^j z^{-(n_a)}, \\ G_j(z^{-1}) = G_0 + G_1 z^{-1} + \ldots + G_{j-1} z^{-(j-1)}, \\ H_j(z^{-1}) = H_0^j + H_1^j z^{-1} + \ldots + H_{n_b-1}^j z^{-(n_b-1)}. \end{cases} \tag{44}$$

Multiplying both sides of Equation (44) by $E_j(z^{-1})$, we use the diophantine equation to get

$$y(k+j) = G_j \Delta u(k+j-1) + H_j \Delta u(k-1) + F_j y(k). \tag{45}$$

The reference track $w(k+j)$, as shown in the following formula:

$$w(k+j) = \delta y(k) + (1-\delta) y_r, \tag{46}$$

where δ is the softening coefficient, from 0 to 1.

The target function can be written as

$$J = \|y - w\|^2 + \|\Delta u\|_\lambda^2. \tag{47}$$

Next, the optimal solution is solved. First, assuming that function $h(\tau)$ satisfies:

- $\lim_{\tau \to 0^+} h(\tau) = +\infty$.
- $h(\tau) > 0, \forall \tau > 0$.
- $h(\tau_1) \geq h(\tau_2), \forall \tau_1 \leq \tau_2$.

where $\tau_i(u) \geq 0$, $i = 1, \ldots, m$.

Taking $h(\tau)$ as the penalty term to construct the interior point penalty function

$$P_\zeta(u) = J(u) + \zeta^{-1} \sum_{i=1}^{m} h(\tau_i(u)). \tag{48}$$

The optimization problems are as follows:

$$\lim_{u \epsilon R^n} P_{\zeta_k}(u) = J(u) + \zeta_k^{-1} \sum_{i=1}^{m} h(\tau_i(u)), \tag{49}$$

where ζ_k^{-1} satisfies $\zeta_1^{-1} > \cdots > \zeta_k^{-1} > \cdots > 0$.

It can be seen that the influence of penalty term is gradually decreasing and tends to 0.

$J(u)$, $h(\tau_i(u))$ and $P_{\zeta_k}(u)$ is supposed to convex function, and then the Newton step size of problem (49) is

$$d_k = -[\nabla^2 P_{\zeta_k}(u_k)]^{-1} \nabla P_{\zeta_k}(u_k). \tag{50}$$

The penalty items is $\zeta^{-1} \sum_{i=1}^{m} \frac{1}{\tau_i(u)}$. Making $\frac{\partial P}{\partial u} = 0$, we can get

$$G^T G u + f G^T - \omega G^T + \lambda u - \zeta^{-1} \sum_{i=1}^{m} \frac{1}{(\tau_i(u))^2} = 0, \tag{51}$$

where $f = H\Delta u(k-1) + Fy$, $\zeta^{-1} = [1, 0.1, 0.01, \ldots]$

Then we get the control law for the multi-variable GPC:

$$u = (G^T G + I\lambda)^{-1}(D + \zeta G^T - f G^T), \tag{52}$$

where $D = -\zeta^{-1} \sum_{i=1}^{m} \frac{1}{(\tau_i(u))^2}$.

Let us take the first row of $(G^T G + I\lambda)^{-1}$ and call it $P^T = [P_1, \ldots, P_{N_p}]$. Then, the generalized predictive control law can be written as follows:

$$\Delta u(k) = P^T(D + \omega G^T - f G^T), \tag{53}$$

$$u(k+1) = u(k) + \Delta u(k). \tag{54}$$

5. Simulation Results

In this section, the system identification of the FMOMR and the 3D trajectory tracking control simulation results are discussed in detail. Figure 4 shows the control flow of the algorithm proposed in paper. Firstly, the training data pairs $y(t)$ and $u(t)$ of FMOMR are sent to the TSFQVNN for training the network. Both $y(t)$ and $u(t)$ are consisted of quaternion value data. Then, the T-S fuzzy model is constructed by the TSFQVNN. Thirdly, the weighted sum of the T-S fuzzy model is used to get the CARIMA model. Finally, the GPC is designed and the objective function is optimized respectively.

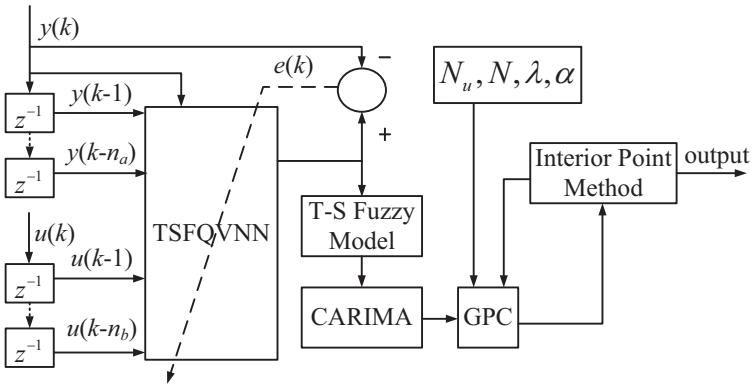

Figure 4. The control flow diagram.

5.1. System Identification

First, this paper adopts the Lorenz chaotic time series proposed by Edward Lorenz [30], and Chen's chaotic time series proposed by Chen [31]. Chaotic time series have non-periodicity and randomness, which can effectively verify the feasibility and test the performance of TSFQVNN.

The Lorenz equation is given below [30]:

$$\frac{dx(t)}{dt} = \phi_1[y(t) - x(t)], \frac{dy(t)}{dt} = x(t)[\varphi_1 - z(t)] - y(t), \frac{dz(t)}{dt} = x(t)y(t) - \psi_1 z(t). \tag{55}$$

Chen's equation is given below [31]:

$$\frac{dx(t)}{dt} = \phi_2[y(t) - x(t)], \frac{dy(t)}{dt} = x(t)[\psi_2 - \phi_2 - z(t)] + \psi_2 y(t), \frac{dz(t)}{dt} = x(t)y(t) - \varphi_2 z(t), \quad (56)$$

where ϕ_1, φ_1, ψ_1, ϕ_2, φ_2, ψ_2 are the dimensionless parameters, and their typical values are used where $\phi_1 = 10$, $\varphi_1 = 28$, $\psi_1 = 8/3$, $\phi_2 = 35$, $\varphi_2 = 3$, $\psi_2 = 28$.

All the three coordinates are selected, and 3000 samples are obtained. Samples from No. 1 to No. 1840 are selected as the training data and data from No. 1841 to No. 2840 is chosen as the test set. The TSQVNN model predicts at the same time the three chaotic outputs $x(t+1)$, $y(t+1)$, and $z(t+1)$ using one quaternion's input that contains $x(t)$, $y(t)$, and $z(t)$.

The T-S fuzzy logic system (TSFLS) model and TSFNN model are used for comparing the difference of output with TSFQVNN model. The root mean square error (RMSE), the symmetric mean absolute percentage error (SMAPE), and the normalized RMSE (NRMSE) are applied for evaluating the prediction model performance [32]. From the results of Tables 1 and 2, it can be concluded that the proposed TSFQVNN performs better under those evaluation functions. In addition, in TSFQVNN, when the number of input/output is reduced to 1/1, only one weight update is needed. In other strategies, the number of input/output is 3/3. When it is converted to three 3/1 systems, three weight updates are needed, which requires more calculation.

Table 1. Comparison of performance on Lorenz series prediction

Series	Method		RMSE	SMAPE	NRMSE
Lorenz −x(t+1)	TSFLS	ave.	0.1308	0.0143	2.6706×10^{-4}
		std.	2.9804×10^{-4}	4.9069×10^{-5}	1.2154×10^{-6}
	TSFNN	ave.	0.7136	0.0465	0.0079
		std.	0.0707	0.0046	7.8721×10^{-4}
	TSFQVNN	ave.	0.0632	0.0037	7.6470×10^{-5}
		std.	7.0648×10^{-4}	7.6684×10^{-5}	8.3994×10^{-8}
Lorenz −y(t+1)	TSFLS	ave.	0.2796	0.0330	0.0010
		std.	0.0081	9.8812×10^{-4}	6.1224×10^{-5}
	TSFNN	ave.	0.9233	0.0543	0.0112
		std.	0.0915	0.0054	0.0011
	TSFQVNN	ave.	0.1008	0.0064	1.6670×10^{-4}
		std.	0.0022	1.9541×10^{-4}	3.6935×10^{-7}
Lorenz −z(t+1)	TSFLS	ave.	0.4901	0.0140	0.0033
		std.	0.0081	0.0021	0.0018
	TSFNN	ave.	0.4959	0.0083	0.0031
		std.	0.0492	8.2122×10^{-4}	3.1080×10^{-4}
	TSFQVNN	ave.	0.1119	0.0016	1.7078×10^{-4}
		std.	0.0022	1.6465×10^{-4}	1.9180×10^{-5}

Figure 5 is the prediction curve and error curve of the Lorenz series and Chen's series generated by TSFLS, TSFNN, and TSFQVNN. The smaller error in the curve proves that the TSFQVNN can obtain 3D and 4D system information better than other methods.

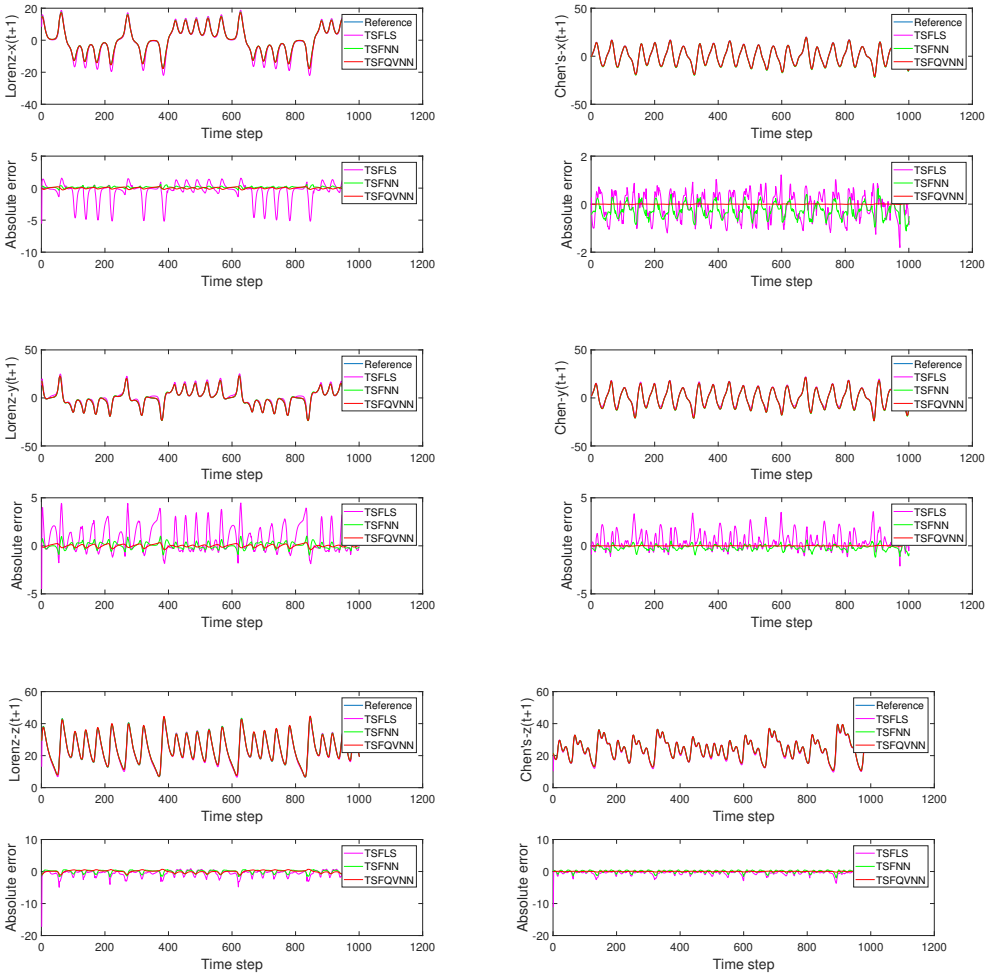

Figure 5. Prediction curves and error curves produced by the TSFLS, TSFNN, and TSFQVNN for Lorenz-x(t + 1), y(t + 1), and z(t + 1) and Chen's-x(t + 1), y(t + 1), and z(t + 1), respectively.

Finally, it is not difficult to conclude that the TSFQVNN generates smaller averages error and standard deviation. We tend to attribute the good performance of TSFQVNN to the advantage of quaternion representation of multidimensional input and output data and the effective characteristics obtained by Hamilton product. It is specially designed for 3D and 4D system identification.

We used a virtual robot experimentation platform (CoppeliaSim) software to design an FMOMR. In addition, a narrow horizontal pipe space with three cylinders parallel to the pipe length is simulated, and an obstacle is designed at the bottom of the pipe. The purpose of the experiment is to let FMOMR bypass the obstacle and pass through the whole pipe.

Table 2. Comparison of performance on Chen series prediction

Series	Method		RMSE	SMAPE	NRMSE
Chen $-x(t+1)$	TSFLS	ave.	0.1396	0.0064	2.4755×10^{-4}
		std.	0.0032	1.7751×10^{-4}	1.1356×10^{-5}
	TSFNN	ave.	1.0863	0.0606	0.0150
		std.	0.0029	1.8568×10^{-4}	7.9359×10^{-6}
	TSFQVNN	ave.	0.0037	1.7592×10^{-4}	1.8488×10^{-7}
		std.	0.0016	7.6684×10^{-5}	1.9952×10^{-7}
Chen $-y(t+1)$	TSFLS	ave.	0.1690	0.0065	3.0051×10^{-4}
		std.	0.0023	3.9201×10^{-5}	8.2060×10^{-6}
	TSFNN	ave.	1.4721	0.0711	0.0228
		std.	0.0046	2.2327×10^{-4}	1.4459×10^{-4}
	TSFQVNN	ave.	0.0081	3.5710×10^{-4}	7.2862×10^{-7}
		std.	0.0463	0.0014	1.1083×10^{-8}
Chen $-z(t+1)$	TSFLS	ave.	0.4379	0.0110	0.0020
		std.	0.0269	1.9368×10^{-6}	3.0995×10^{-7}
	TSFNN	ave.	0.4379	0.0065	0.0024
		std.	0.0046	5.9560×10^{-5}	7.2265×10^{-5}
	TSFQVNN	ave.	0.2644	0.0043	0.0012
		std.	0.0463	0.0028	0.0013

5.2. Trajectory Tracking in 3D Space

First, we get a set of motion data pairs of FMOMR from the FMOMR simulation model established by CoppeliaSim software, and then use the method described in Section 3 to fit the nonlinear system. The identification results are shown in Figure 6. As you can see from Figure 6, the fitting performance is satisfactory. The errors of fitting for the x-axis and y-axis are in close proximity to zero extremely. The forecasted curve of z-axis is approaching to the reference curve as well. The comparison between forecasted and reference spatial trajectory indicate that the system identification work has been accomplished well. Figure 7 depicts a picture of robot simulation on the internal surface of a cylindrical cavity in case that the FMOMR is caught into the hole. The cylindrical cavity flats on the floor and displays in the transparent mode. The trajectory of FMOMR is shown clearly by a red solid line. Therefore, The FMOMR simulation scene can be treated as a platform for verification of the algorithm and multidimensional data generator at the same time.

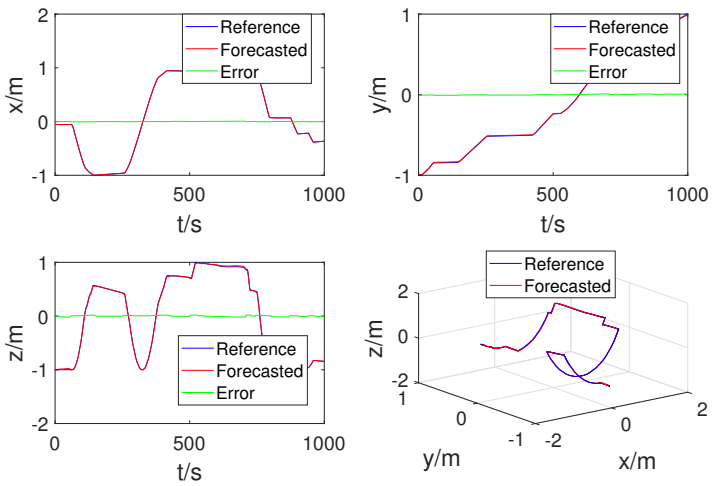

Figure 6. System identification of FMOMR by TSFQVNN.

The obtained T-S fuzzy model is weighted and summed to obtain the CARIMA model, and then the generalized predictive control algorithm is used for trajectory tracking

control. We make the control time-domain N_u set to 4, the prediction time-domain N_p to 8, the softening factor to 0.4, the control weighting coefficient to 1, and the target trajectory is a 3D rectangle. Figure 8 compares the tracking performance of TSFQVNN GPC and TSFNN GPC. The orange trajectories in Figure 8 are first substituted into the trajectory data obtained in CoppeliaSim by substituting the angular velocities of the four mecanum obtained after Matlab simulation, and the data is then substituted into Matlab and plotted. We can observe that TSFQVNN GPC performs better than TSFNN GPC, and it also works well in CoppeliaSim.

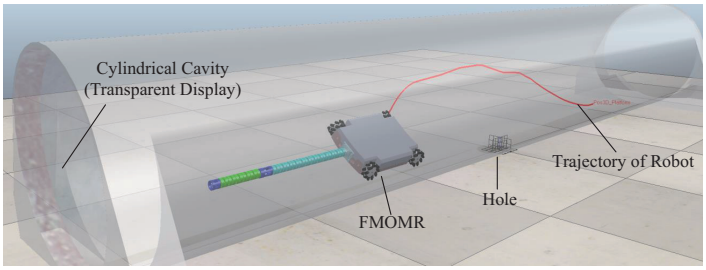

Figure 7. CoppeliaSim scene for four mecanum wheels omnidirectional mobile robot.

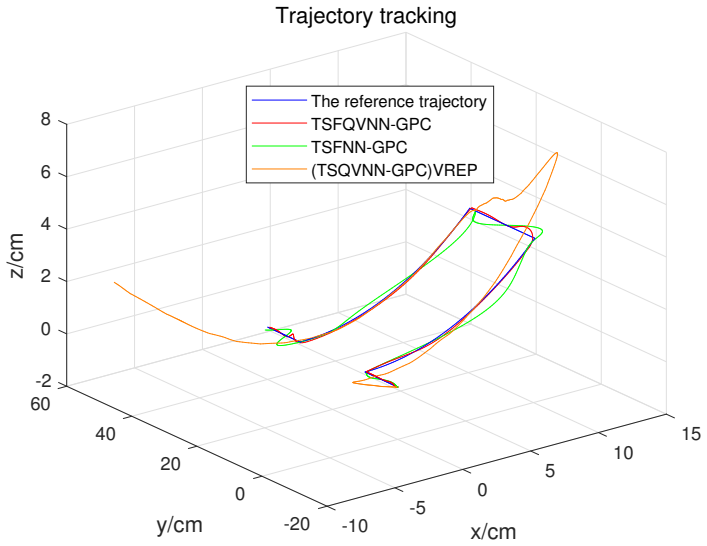

Figure 8. Executed pathes of four mecanum wheels omnidirectional mobile robot.

6. Conclusions

In this paper, TSFQVNN is used to identify 3D and 4D systems. The universal approximation of TSFQVNN and the condition of approximation error convergence are proven. The simulation results show that TSFQVNN can be used in 3D and 4D system identification to get a better identification effect. In addition, TSFQVNN and GPC are combined to design a data-driven FMOMR 3D trajectory tracking controller. The simulation results prove that the controller has higher tracking accuracy, which provides a good premise for the research of FMOMR trajectory tracking in 3D space.

Author Contributions: Conceptualization, S.D.; methodology, writing, visualization, C.M.; investigation, X.L.; supervision, G.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Fundamental Research Funds for the Central Universities under Grant 2022SCU12004.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Jiang, B.H.; Qi, Q. The design of logistics handling robot based on mcu. *Appl. Mech. Mater.* **2013**, *336–338*, 1124–1128. [CrossRef]
2. Wu, H.; Tian, G.H.; Li, Y.; Zhou, F.Y.; Duan, P. Spatial semantic hybrid map building and application of mobile service robot. *Robot. Auton. Syst.* **2014**, *62*, 923–941. [CrossRef]
3. Gao, X.; Acar, L. Using a mobile robot with interpolation and extrapolation method for chemical source localization in dynamic advection-diffusion environment. *Int. Conf. Robot. Autom.* **2016**, *5*, 87–97. [CrossRef]
4. Doering, N.; Poeschl, S.; Gross, H.; Bley, A.; Martin, C.; Boehme, H.J. User-centered design and evaluation of a mobile shopping robot. *Int. J. Soc. Robot.* **2015**, *7*, 203–225. [CrossRef]
5. Takemura, Y.; Yu, O.; Nassiraei, A.A.F.; Sanada, A.; Miyamoto, H. A system design concept based on omnidirectional mobility, safety and modularity for an autonomous mobile soccer robot. *J. Bionic Eng.* **2008**, *5*, 121–129. [CrossRef]
6. de Villiers, M.; Tlale, N.S. Development of a control model for a four wheel mecanum vehicle. *J. Dyn. Syst. Meas. Control* **2012**, *134*, 011007. [CrossRef]
7. Wang, C.; Liu, X.; Yang, X.; Hu, F.; Jiang, A.; Yang, C. Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy. *Appl. Sci.* **2018**, *5*, 231. [CrossRef]
8. Alakshendra, V.; Chiddarwar, S.S. Adaptive robust control of mecanum-wheeled mobile robot with uncertainties. *Nonlinear Dyn.* **2017**, *87*, 2147–2169. [CrossRef]
9. Conceio, A.S.; Moreira, A.P.; Costa, P.J. Practical approach of modeling and parameters estimation for omnidirectional mobile robots. *IEEE/ASME Trans. Mechatronics* **2009**, *14*, 377–381. [CrossRef]
10. Salih, J.E.M.; Rizon, M.; Yaacob, S.; Adom, A.H.; Mamat, M.R. Designing Omni-Directional Mobile Robot with Mecanum Wheel. *Am. J. Appl. Sci.* **2006**, *3*, 1831–1835. [CrossRef]
11. Lin, L.C.; Shih, H.Y. Modeling and adaptive control of an omnimecanum-wheeled robot. *Intell. Control Autom.* **2013**, *2*, 166–179. [CrossRef]
12. Deng, J. Dynamic neural networks with hybrid structures for nonlinear system identification. *Eng. Appl. Artif. Intell.* **2013**, *26*, 281–292. [CrossRef]
13. Bagherzadeh, S.A. Nonlinear aircraft system identification using artificial neural networks enhanced by empirical mode decomposition. *Aerosp. Sci. Technol.* **2018**, *75*, 155–171. [CrossRef]
14. Rong, H.J.; Sundararajan, N.; Huang, G.B.; Saratchandran, P. Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction. *Fuzzy Sets Syst.* **2006**, *157*, 1260–1275. [CrossRef]
15. Wang, S.; Dou, J.; Liu, Y.; Liu, F. Prediction of chaotic time series based on interval type-2 T-S fuzzy system. *J. Comput. Inf. Syst.* **2014**, *10*, 5403–5412.
16. Cervantes, J.; Yu, W.; Salazar, S.; Chairez, I. Takagi-Sugeno dynamic neuro-fuzzy controller of uncertain nonlinear systems. *IEEE Trans. Fuzzy Syst.* **2017**, *25*, 1601–1615. [CrossRef]
17. Yilmaz, Y.; Oysal, S. Fuzzy wavelet neural network models for prediction and identification of dynamical systems. *IEEE Trans. Neural Netw.* **2010**, *21*, 1599–1609. [CrossRef]
18. Abiyev, R.H.; Kaynak, O.; Kayacan, E. A type-2 fuzzy wavelet neural network for system identification and control. *J. Frankl. Inst.* **2013**, *15*, 1658–1685. [CrossRef]
19. Matsui, N.; Isokawa, T.; Kusamichi, H.; Peper, F.; Nishimura, H. Quaternion neural network with geometrical operators. *J. Intell. Fuzzy Syst.* **2004**, *15*, 149–164.
20. Bei, J.C.; Hua, Z.S.; Gang, C.; Ge, J. Color face recognition based on quaternion zernike moment invariants and quaternion bp neural network. *Appl. Mech. Mater.* **2011**, *446*, 1034–1039.
21. Oyama, K.; Hirose, A. Phasor quaternion neural networks for singular point compensation in polarimetric-interferometric synthetic aperture radar. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 2510–2519. [CrossRef]
22. Parcollet, T.; Morchid, M.; Linares, G. A survey of quaternion neural networks. *Artif. Intell. Rev.* **2019**, *53*, 2957–2982. [CrossRef]
23. Takahashi, K. Remarks on feedforward-feedback controller using simple recurrent quaternion neural network. In Proceedings of the 2018 IEEE Conference on Control Technology and Applications (CCTA), Copenhagen, Denmark, 21–24 August 2018; pp. 1246–1251.
24. Saoud, L.S.; Ghorbani, R.; Rahmoune, F. Cognitive quaternion valued neural network and some applications. *Neurocomputing* **2017**, *221*, 85–93. [CrossRef]
25. Yuan, Z.; Tian, Y.; Yin, Y.; Wang, S.; Liu, J.; Wu, L. Trajectory tracking control of a four mecanum wheeled mobile platform: An extended state observer-based sliding mode approach. *IET Control Theory Appl.* **2020**, *14*, 415–426. [CrossRef]
26. Zhou, J.; Zhang, N.; Li, C.; Zhang, Y.; Lai, X. An adaptive Takagi-Sugeno fuzzy model-based generalized predictive controller for pumpedstorage unit. *IEEE Access* **2019**, *99*, 1–13. [CrossRef]
27. Liu, X.; Liu, J.; Guan, P. Neuro-fuzzy generalized predictive control of boiler steam temperature. *J. Control Theory Appl.* **2007**, *5*, 83–88. [CrossRef]
28. Bezdek, J.C.; Ehrlich, R.; Full, W. FCM: The fuzzy c-means clustering algorithm. *Comput. Geosci.* **1984**, *10*, 191–203. [CrossRef]

29. Wang, L.X.; Mendel, J.M. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Trans. Neural Netw.* **1992**, *3*, 807–814. [CrossRef]
30. Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **1963**, *20*, 130–141. [CrossRef]
31. Chen, G.; Ueta, T. Yet another chaotic attractor. *Int. J. Bifurc. Chaos* **1999**, *9*, 1465–1466. [CrossRef]
32. Xu, M.; Han, M.; Chen, C.L.P.; Qiu, T. Recurrent broad learning systems for time series prediction *IEEE Trans. Cybern.* **2020**, *50*, 1405–1417.