entropy

MDPI

Article

Neural Computing Enhanced Parameter Estimation for Multi-Input and Multi-Output Total Non-Linear Dynamic Models

Longlong Liu [1], Di Ma [1], Ahmad Taher Azar [2,3,*] and Quanmin Zhu [4]

1 School of Mathematical Sciences, Ocean University of China, Qingdao 266000, China; liulonglong@ouc.edu.cn (L.L.); 21181111013@stu.ouc.edu.cn (D.M.)
2 Robotics and Internet-of-Things Lab (RIOTU), Prince Sultan University, Riyadh 11586, Saudi Arabia
3 Faculty of Computers and Artificial Intelligence, Benha University, 13511 Benha, Egypt
4 Department of Engineering Design and Mathematics, University of the West of England, Frenchy Campus Coldharbour Lane, Bristol BS16 1QY, UK; quan.zhu@uwe.ac.uk
* Correspondence: aazar@psu.edu.sa or ahmad.azar@fci.bu.edu.eg

Received: 28 March 2020; Accepted: 26 April 2020; Published: 30 April 2020

Abstract: In this paper, a gradient descent algorithm is proposed for the parameter estimation of multi-input and multi-output (MIMO) total non-linear dynamic models. Firstly, the MIMO total non-linear model is mapped to a non-completely connected feedforward neural network, that is, the parameters of the total non-linear model are mapped to the connection weights of the neural network. Then, based on the minimization of network error, a weight-updating algorithm, that is, an estimation algorithm of model parameters, is proposed with the convergence conditions of a non-completely connected feedforward network. In further determining the variables of the model set, a method of model structure detection is proposed for selecting a group of important items from the whole variable candidate set. In order to verify the usefulness of the parameter identification process, we provide a virtual bench test example for the numerical analysis and user-friendly instructions for potential applications.

Keywords: parameter estimation; total non-linear model; neural networks; neuro-computing; gradient descent algorithm

1. Introduction

Because a total non-linear model can provide a very concise representation for complex non-linear systems and has good extrapolation characteristics, it has attracted the attention of academic research and applications. Compared with the polynomial non-linear auto-regressive moving average with exogenous input (NARMAX) model, the total non-linear model is an extension of the polynomial model, which can be defined as the ratio of two polynomial expressions [1–3]. The introduction of denominator polynomials makes the NARMAX model non-linear in parameters and regression terms. Therefore, compared with the polynomial model, the model identification and the controller design of the total non-linear model are much more challenging [4,5]. In view of the difficulty of parameter estimation of a total non-linear model, using simple and effective algorithm and machine learning should be considered for extracting the information from measurement data.

1.1. Literature Survey

At present, a variety of model structure detection techniques and parameter estimation algorithms are developed for non-linear models, including the orthogonal model structure detection and parameter

estimation program [6], the generalized least square estimator [7,8], the prediction error estimator [9,10], the Kalman filter estimator [11,12], the genetic algorithm estimator [12,13], the artificial neural network estimator [14–17], etc. However, most of these algorithms are parameter estimators for polynomial non-linear models. Zhu and Billings have done a lot of research work on the parameter identification of a total non-linear model [7,8], and they put forward the parameter estimation method of a total non-linear model based on a back-propagation (BP) algorithm in 2003. They discussed the advantages of BP calculation in recognition of the classical model to provide the best combination of classical and neural network methods and provided a powerful tool for analyzing a large number of systems.

In [18], a back-propagation estimation formula based on neuro-computing was presented for estimating the total non-linear model parameters, where a pack of solutions were derived for the problems of parameter initialization, learning rate selection, stop criteria and model structure detection, and the convergence of a back-propagation estimator (BPE). However, Reference [18] only proposed a parameter estimation method for single-input and single-output (SISO) systems, and correspondingly the case studies. Expanding [18], this paper presents solutions for the parameter estimation of a total non-linear multi-input and multi-output (MIMO) model. Due to the complexity of a MIMO system, it is more difficult to estimate the model parameters, but they are more general in academic research and applications. For example, the parameters of a MIMO system are many more than those of a SISO system, and the parameters to be estimated each time will be multiplied, which increases the difficulty of estimation. Moreover, due to the coupling of multiple systems, the parameter values of each system also affect each other. The algorithms to estimate these parameters are not independent but interactive and complex. Because the components of different MIMO systems are different, the total connection neural network structure adopted in [18] is not suitable for estimating the parameters of MIMO systems. When the MIMO system is mapped into a neural network, the network structure is often asymmetric or non-completely connected (the neurons in the hidden layer are not connected with all the neurons in the input layer). That is to say, the network is not a common completely connected feedforward neural network, and the general BP algorithm cannot be directly applied to the estimation of the parameters. Therefore, the learning algorithm of the parameters must be properly derived. Due to the asymmetry of the network, the convergence of the network is also facing challenges. It is necessary to analyze the convergence of the network and give the specific conditions of the network convergence. A MIMO system needs to identify the parameters of a SISO system several times, and a MIMO system can have multiple inputs. In the simulation experiment, the parameters of the system should be estimated under different combinations of multiple inputs, and the performance of the network estimator should be verified. Therefore, the parameter identification of a MIMO system is much more challenging.

1.2. Motivation and Contributions

The authors of [19] presented a thorough analysis that included two kernel components, the SISO model and the orthogonal algorithms are parameter estimators for polynomial non-linear models such as predictive and back propagation computation. Since then, rational model identification has gone to diversified directions, such as more theoretical considerations of a non-linear least squares algorithm [4], a maximum likelihood estimation [3], and a biased compensation recursive least squares algorithm [2]. It has been noted that the MIMO rational model identification has seldom attracted research, probably due to the complexity in algorithm formulation and the coupling effect. However, this MIMO rational model identification should be a research agent now because of recent applications and increasing computing facilities.

The total non-linear system model, which is relatively new, is the alternative name of the NARMAX rational model, which was defined by a survey paper on the rational model identification [19]. The total non-linear model emphasizes the non-linearity in both the parameters and control inputs, and it has been taken as a challenging structure for designing non-linear dynamic control systems [1]. The rational model gives more consideration as expanded polynomials in math, structure detection, and parameter estimation in the field of system identification [2,3]. Therefore, the main contribution of the new study

is to use neural computing algorithms for a MIMO model parameter estimation. The new study is a complement to those classical NAMAX approaches.

The rest of the paper is organized as follows. The total non-linear model is described in Section 2. Section 3 presents the gradient descent calculation of parameter estimation. Next, model structure detection is discussed in Section 4. A convergence analysis of an algorithm is presented in Section 5. Simulation results and discussions are demonstrated in Section 6. Finally, Section 7 includes the paper conclusions and some of the future aspects.

2. Total Non-Linear Model

In mathematics, the dynamic total non-linear model of a MIMO system with error can be defined as

$$\mathrm{y}_i(t) = \hat{\mathrm{y}}_i + e_i(t) = \frac{a_i(t)}{b_i(t)} + e_i(t) = \frac{a_i(u_1, u_2, \ldots, u_J, y_1, y_2, \ldots, y_I, e_1, e_2, \ldots, e_I)}{b_i(u_1, u_2, \ldots, u_J, y_1, y_2, \ldots, y_I, e_1, e_2, \ldots, e_I)} + e_i(t) \mathrm{i} = 1, 2, \ldots, \mathrm{I} \quad (1)$$

$$a_i(t) = \sum_{k=1}^{N} p_k^n(t)\theta_k^n, b_i(t) = \sum_{k=1}^{D} p_k^d(t)\theta_k^d \ \mathrm{i} = 1, 2, \ldots, \mathrm{I} \quad (2)$$

where $\mathbf{y}(\mathbf{t}) = [y_1(t), y_2(t), \ldots, y_I(t)] \in R^I$ and $\hat{\mathbf{y}}(\mathbf{t}) = \left[\hat{y}_1(t), \hat{y}_2(t), \ldots, \hat{y}_I(t)\right] \in R^I$ are the measured output and model output, respectively; $\mathbf{u}(\mathbf{t}) = \left[u_1(t), u_2(t), \ldots, u_J(t)\right] \in R^J$ is the input; $\mathbf{e}(\mathbf{t}) = [e_1(t), e_2(t), \ldots, e_I(t)] \in R^I$ is the model error; and $\mathbf{t} = [1, 2, \ldots, \mathrm{T}] \in \boldsymbol{Z}^{+T}$ is the sampling time index. Numerator $a_i(t) \in \mathrm{R}$ and denominator $b_i(t) \in \mathrm{R}$ as represented by polynomials, regression term $p_k^n(t)$, and $p_k^d(t)$ are products of past inputs, outputs, and errors, such as $u_1(t-1)y_2(t-3)$, $u_1(t-1)e_2(t-2)$, $y_2^3(t-1).\theta^n = \left[\theta_1^n, \theta_2^n, \ldots, \theta_N^n\right] \in R^N$, and $\theta^d = \left[\theta_1^d, \theta_2^d, \ldots, \theta_D^d\right] \in R^D$ are the parameter sets of $a_i(t)$ and $b_i(t)$, respectively.

The task of parameter estimation is to extract the relevant parameter values from the measured input and output data for a given model structure. To form a regression expression for parameter estimation, multiplying $e_i(t)$ of both sides of Formula (1) gives

$$\mathrm{y}_i(t)b_i(t) - a_i(t) = b_i(t)e_i(t) \quad (3)$$

To consider the neuro-computing approach for parameter estimation, a total non-linear model is expressed into a non-completely connected feedforward neural network, as shown in Figure 1.

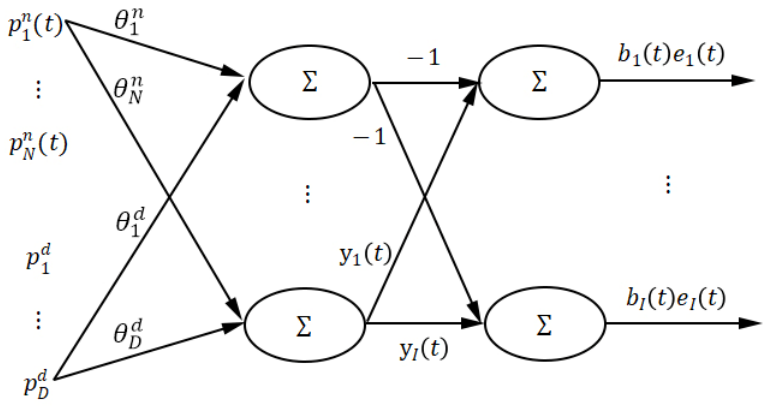

Figure 1. Structure of a neural network corresponding to a total non-linear model.

We define the network with an on both sides, Formula (11) is obtainedinput layer, a hidden layer, and an output layer, where:

(i) The input layer consists of regression terms $p_k^n(t)(k = 1, \ldots, N)$ and $p_k^d(t)$ $(k = 1, \ldots, D)$; here, a neuron in the hidden layer is not connected to all the neurons in the input layer, that is, the network is a non-completely connected feedforward neural network.
(ii) The action function of the neurons in the hidden layer is linear, and the output of the hidden layer neurons is $a_i(t)$ or $b_i(t)$.
(iii) The action function of the output layer neurons is linear, and the output of the ith output layer neuron is $b_i(t)e_i(t)$.
(iv) The connection weights between the input layer neurons and the hidden layer neurons are the parameters θ_k^n and θ_k^d of the model.
(v) The connection weight between the hidden layer neurons and the ith output layer neurons are -1 and the observed output $y_i(t)$.

Leung and Haykin proposed a rational function neural network [20] but did not define a generalized total non-linear model structure or consider the relevant errors. Therefore, their parameter estimation algorithm could not provide an unbiased estimation for noise damaged data, which was essentially a special implementation of Zhu and Billings's [7,8] methods in the case of no noise data. The method proposed in this paper is a further study of the method in Zhu [18]. The characteristics of a total non-linear model (1) are as follows:

(i) By setting parameter $i = 1$, Zhu's [18] model can be a special case of the model in Formula (1).
(ii) The model is non-linear in parameters and regression terms, which was caused by denominator polynomials.
(iii) When the denominator $b_i(t)$ of the model is close to 0, the output deviation would be large. In this paper, considering this point, division operation was avoided in the action function of the neuron when the neural network model was being built.
(iv) The structure of the neural network corresponding to the total non-linear model is a non-completely connected feedforward neural network, or a partially connected feedforward neural network. Therefore, the convergence of the network becomes a big problem, which is the difficulty of this paper.
(v) The model has a wide range of application prospects. In many non-linear system modeling and control applications, the total non-linear model has been gradually adopted. Some non-linear models, such as the exponential model e^x, which describes the change of dynamic rate constant with temperature, cannot be directly used. The exponential model can be firstly transformed into a non-linear model ($e^x = \frac{1-\frac{x}{2}+\frac{x^2}{12}}{1+\frac{x}{2}+\frac{x^2}{12}}$), and then, system identification can be implemented [19,21,22].

3. Gradient Descent Calculation of Parameter Estimation

For the convenience of the following derivations, set the output of neuron i in the output layer of the neural network as $f_i(t)$.

$$f_i(t) = b_i(t)e_i(t) \tag{4}$$

Define the error measure function of one iteration of network as:

$$E(t) = \frac{1}{2}\sum (y_i(t) - \hat{y}_i(t))^2 = \frac{1}{2}\sum (e_i(t))^2 \tag{5}$$

The Lyapunov method is often used to analyze the stability of a neural network [23]; similarly, the network parameters are estimated by minimizing the network error based on the Lyapunov method. It should be noted that when the total non-linear model is represented in the neural network structure of Figure 1, the parameter estimation of the model can be described as the training of neural network weight by minimizing the error $E(t)$ in Formula (5).

In order to train the weights of the network, the learning algorithm based on the gradient descent is given by Formulas (6) and (7):

$$\Delta\theta_k^n = -\eta_n \frac{\partial \mathrm{E}}{\partial \theta_k^n} = -\eta_n e_i(t) \frac{\partial e_i(t)}{\partial \theta_k^n} \tag{6}$$

$$\Delta\theta_k^d = -\eta_{\mathrm{d}} \frac{\partial \mathrm{E}}{\partial \theta_k^d} = -\eta_d e_i(t) \frac{\partial e_i(t)}{\partial \theta_k^d} \tag{7}$$

where η_n and η_{d} are learning rates.

By deriving Formula (4) from θ_k^n on both sides, Formula (8) is obtained:

$$\begin{gathered} \frac{\partial f_i(t)}{\partial \theta_k^n} = \frac{\partial b_i(t)}{\partial \theta_k^n} e_i(t) + b_i(t) \frac{\partial e_i(t)}{\partial \theta_k^n} \\ \frac{\partial e_i(t)}{\partial \theta_k^n} = \frac{1}{b_i(t)} \left(\frac{\partial f_i(t)}{\partial \theta_k^n} - \frac{\partial b_i(t)}{\partial \theta_k^n} e_i(t) \right) = \frac{1}{b_i(t)} \frac{\partial f_i(t)}{\partial \theta_k^n} = \frac{1}{b_i(t)} \frac{\partial (y_i(t) b_i(t) - a_i(t))}{\partial \theta_k^n} \\ = -\frac{1}{b_i(t)} \frac{\partial a_i(t)}{\partial \theta_k^n} = -\frac{p_k^n(t)}{b_i(t)} \end{gathered} \tag{8}$$

Substituting Formula (8) into Formula (6) to get Formula (9), we can then get Formula (10):

$$\Delta\theta_k^n = -\eta_n e_i(t) \frac{\partial e_i(t)}{\partial \theta_k^n} = \eta_n e_i(t) \frac{p_k^n(t)}{b_i(t)} \tag{9}$$

$$\theta_k^n(\mathrm{t}+1) = \theta_k^n(\mathrm{t}) + \Delta\theta_k^n = \theta_k^n(\mathrm{t}) + \eta_n e_i(t) \frac{p_k^n(t)}{b_i(t)} \tag{10}$$

By deriving Formula (4) from θ_k^d on both sides, Formula (11) is obtained:

$$\begin{gathered} \frac{\partial f_i(t)}{\partial \theta_k^d} = \frac{\partial b_i(t)}{\partial \theta_k^d} e_i(t) + b_i(t) \frac{\partial e_i(t)}{\partial \theta_k^d} \\ \frac{\partial e_i(t)}{\partial \theta_k^d} = \frac{1}{b_i(t)} \left(\frac{\partial f_i(t)}{\partial \theta_k^d} - \frac{\partial b_i(t)}{\partial \theta_k^d} e_i(t) \right) \\ = \frac{1}{b_i(t)} \left(\frac{\partial (y_i(t) b_i(t) - a_i(t))}{\partial \theta_k^d} - \frac{\partial b_i(t)}{\partial \theta_k^d} e_i(t) \right) = \frac{1}{b_i(t)} (y_i(t) p_k^d(t) - p_k^d(t) e_i(t)) \\ = \frac{1}{b_i(t)} (y_i(t) - e_i(t)) p_k^d(t) = \frac{1}{b_i(t)} \frac{a_i(t)}{b_i(t)} p_k^d(t) = \frac{a_i(t)}{b_i^2(t)} p_k^d(t) \end{gathered} \tag{11}$$

Substituting Formula (11) into Formula (8) to get Formula (12), we then get Formula (13):

$$\Delta\theta_k^d = -\eta_d e_i(t) \frac{\partial e_i(t)}{\partial \theta_k^d} = -\eta_d e_i(t) \frac{a_i(t)}{b_i^2(t)} p_k^d(t) \tag{12}$$

$$\theta_k^d(t+1) = \theta_k^d(t) + \Delta\theta_k^d = \theta_k^d(t) - \eta_d e_i(t) \frac{a_i(t)}{b_i^2(t)} p_k^d(t) \tag{13}$$

The gradient descent algorithm for parameter estimation of a total non-linear model is summarized in Algorithm 1.

Algorithm 1. Gradient Descent Algorithm

1: Initialization: The weights of the neural network (parameters of a total non-linear model) are set as random little numbers with uniform distribution; the average value is zero, and the variance is small. Set the maximum number of iterations T, the minimum error ε, and the maximum number of samples P.
2: Generate training sample set {**X**, **Y**} of the neural network according to Formula (1), where
$\mathbf{X} = \{X_1, X_2, \ldots, X_I\}$, $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_I\}$,
$X_i \ni \{p_1^n(t), p_2^n(t), \ldots, p_N^n(t), p_1^d, p_2^d, \ldots, p_D^d\}$, $Y_i = \{y_i(t)\}$.
3: Input a training sample p to the neural network.
4: Calculate the output value $a_i(\mathrm{k})$,$y_i(t)e_i(t)$ and $f_i(t)$ of the neurons in the hidden layer and the output layer according to Formulas (2), (3), and (4), respectively.
5: Adjust the weight of the neural network according to Formulas (10) and (13).
6: Calculate the error $E(t)$ according to Formula (4) and calculate the total error according to Formula (14).

$$E = \sum E(t) \tag{14}$$

7: $p = p + 1$
8: If $p > P$, then t = t + 1; otherwise, run step 3.
9: If $E < \varepsilon$ or $t > T$, stop training; otherwise, run step 3.

4. Model Structure Detection

Model structure detection is to select important items from a rather large model set (usually called the whole item set) and determine the sub-model with important items [18]. Because of the powerful self-learning and associative memory function of an artificial neural network [24], it is the first-choice tool to identify the model structure. When identifying systems with unknown structures, it is important to avoid losing these important items in the final model. For the structure detection of a total non-linear model, the connection weight estimation in the neural network, that is, the parameter estimation of the total non-linear model, could be used to select the significant terms.

For the important and unimportant items in the whole model item set, the knock-out algorithm is adopted. First, remove the items that lead to the increase of network error, and then knock out the items with lighter weight according to the requirements of significance. Finally, test the error of the non-linear model composed of the remaining items. The specific algorithm is summarized in Algorithm 2.

Algorithm 2. Knock-Out Algorithm

1: Using the network structure shown in Figure 1, all the items contained in the whole items set are taken as the input of the network.
2: The algorithm in Section 3 is used to train the network, and network error E_1 is obtained.
3: A new network structure is obtained by randomly removing a network input. The algorithm in Section 3 is used to train the new network, and network error E_2 is obtained. If $E_2 \leq E_1$, then $E_1 = E_2$. Otherwise, this operation should be invalid (the input is reserved).
4: Another input is selected, and step 3 is executed again until all the input items are executed once.
5: The N connection weights between the input layer and the hidden layer are sorted in descending order. The first n weights are selected to make the significance reach 95%. Meanwhile, Formulas (15) and (16) are met, and the network input items corresponding to the first n weights are retained.

$$\frac{\sum_{i=1}^{n} |w_i|}{\sum_{i=1}^{N} |w_i|} \geq 0.95 \tag{15}$$

$$\frac{\sum_{i=1}^{n-1} |w_i|}{\sum_{i=1}^{N} |w_i|} < 0.95 \tag{16}$$

In the above process, the neural network is not only used to estimate the parameters of the model but also to detect the structure of the model and analyze the significance of the regression term.

5. Convergence Analysis of the Algorithm

Convergence proof:

Assuming that a connection weight of the neural network shown in Figure 1 is changed, this weight can take any value. When the weight θ_k^n corresponds to the regression term parameter of the numerator of the total non-linear model, the resulting network error changes as follows (remove the lower corner marks in Formula (2) for the convenience of proof):

$$\mathrm{y(t)} = \frac{a(t)}{b(t)} + e(t) \tag{17}$$

Substitute Formula (2) into Formula (1) to get Formula (18):

$$\mathrm{y(t)} - \frac{\sum_{\mathrm{n}=1}^{N} p_k^n(t)\theta_k^n}{b(t)} = e(t) \tag{18}$$

When θ_k^n is updated, (18) becomes (19):

$$\mathrm{y(t)} - \frac{\sum_{\mathrm{n}=1,n\neq j}^{N} p_k^n(t)\theta_k^n + p_k^n(t)(\theta_k^n + \Delta\theta_k^n)}{b(t)} = \widetilde{e}(t) \tag{19}$$

$\widetilde{e}(t)$ is the new error of the neural network after the weight has been updated. Subtract Formula (18) from Formula (19) to get Formulas (20) and (21):

$$\begin{gathered} \widetilde{e}(t) - e(t) = -\frac{p_k^n(t)\Delta\theta_k^n}{b(t)} = -\eta_n\left(\frac{p_k^n(t)}{b(t)}\right)^2 \mathrm{e}(t) \\ \widetilde{e}(t) = \left(1 - \eta_n\left(\frac{p_k^n(t)}{b(t)}\right)^2\right) \mathrm{e}(t) \end{gathered} \tag{20}$$

$$\widetilde{e}(t)^2 = \left(1 - \eta_n\left(\frac{p_k^n(t)}{b(t)}\right)^2\right)^2 \mathrm{e}(t)^2 \tag{21}$$

In order to ensure $\widetilde{e}(t)^2 \leq \mathrm{e}(t)^2$, $-1 \leq 1 - \eta_n\left(\frac{p_k^n(t)}{b(t)}\right)^2 \leq 1$, namely:

$$\begin{cases} \eta_n\left(\frac{p_k^n(t)}{b(t)}\right)^2 \leq 2 \\ \eta_n\left(\frac{p_k^n(t)}{b(t)}\right)^2 \geq 0 \end{cases} \tag{22}$$

Solving Formula (22) gives:

$$0 \leq \eta_n \leq \frac{2\mathrm{b}(t)^2}{p_k^n(t)^2} \tag{23}$$

When the changed weight θ_k^d corresponds to the regression parameter of the denominator of the total non-linear model, the resulting network error change is as follows:

$$\mathrm{y(t)} - \frac{a(t)}{\sum_{\mathrm{d}=1}^{D} p_k^d(t)\theta_k^d} = e(t) \tag{24}$$

$$\mathrm{y(t)} - \frac{a(t)}{\sum_{\mathrm{d}=1,d\neq j}^{D} p_k^d(t) + p_k^d(t)(\theta_k^d + \Delta\theta_k^d)} = \widetilde{e}(t) \tag{25}$$

Subtracting Formula (24) from Formula (25) gives $\widetilde{b}(t)$ as the new denominator of the neural network after the weight has been updated.

$$\widetilde{e}(t) - e(t) = \frac{(\widetilde{b}(t) - b(t))a(t)}{\widetilde{b}(t)b(t)} = p_k^d(t)\Delta\theta_k^d \frac{a(t)}{\widetilde{b}(t)b(t)} = -\eta_d e(t)\frac{\partial e(t)}{\partial \theta_k^d} p_k^d(t)\frac{a(t)}{\widetilde{b}(t)b(t)} = -\eta_d e(t)\frac{a(t)^2}{\widetilde{b}(t)b(t)^3} p_k^d(t)^2 \tag{26}$$

$$\widetilde{e}(t)^2 = (1 - \eta_d \frac{a(t)^2}{\widetilde{b}(t)b(t)^3} p_k^d(t)^2)^{\ 2}\ e(t)^2 \tag{27}$$

In order to satisfy $\widetilde{e}(t)^2 \leq \mathrm{e}(t)^2$, namely, $-1 \leq 1 - \eta_d \frac{a(t)^2}{\widetilde{b}(t)b(t)^3} p_k^d(t)^2 \leq 1$, that is:

$$\begin{cases} \eta_d \frac{a(t)^2}{\widetilde{b}(t)b(t)^3} p_k^d(t)^2 \leq 2 \\ \eta_d \frac{a(t)^2}{\widetilde{b}(t)b(t)^3} p_k^d(t)^2 \geq 0 \end{cases} \tag{28}$$

Because the learning coefficient is too large, the training effect of the network is not effective; accordingly, we take $0 \leq \eta_n \leq 1$ to get $\widetilde{b}(t)b(t) > 0$, and thus, it has:

$$0 \leq \eta_d \leq \frac{2\widetilde{b}(t)b(t)^3}{a(t)^2 p_k^d(t)^2} \tag{29}$$

To sum up, the network is convergent when the following conditions are met:

$$1.\ 0 \leq \eta_n \leq \frac{2\mathrm{b}(t)^2}{p_k^n(t)^2}$$

$$2.\ 0 \leq \eta_d \leq \frac{2\widetilde{b}(t)b(t)^3}{a(t)^2 p_k^d(t)^2}$$

Under these two conditions, this algorithm provides a convergence estimate for the parameters of the total non-linear model.

6. Simulation Results and Discussions

Consider a representative example of a total non-linear model:

$$y_1(t) = \frac{0.5y_1(t-1) + 0.8y_2^3(t-2) + u_1(t-1)}{1 + y_1^2(t-1) + u_2^2(t-1)} + \mathrm{r}_1(t) = \frac{\theta_1 y_1(t-1) + \theta_2 y_2^3(t-2) + \theta_3 u_1(t-1)}{1 + \theta_4 y_1^2(t-1) + \theta_5 u_2^2(t-1)} + \mathrm{r}_1(t) \tag{30}$$

$$y_2(t) = \frac{0.2y_2(t-1) - 0.5y_1^2(t-2) + u_2(t-1)}{1 + y_2^2(t-1) + u_2^2(t-1)} + r_2(t) = \frac{\theta_6 y_2(t-1) - \theta_7 y_1^2(t-2) + \theta_8 u_2(t-1)}{1 + \theta_9 y_2^2(t-1) + \theta_{10} u_2^2(t-1)} + r_2(t) \tag{31}$$

Because the disturbance of input data will cause interference to the estimation of parameters [25], in this section, the parameter estimation for different inputs was selected. Firstly, for the simulation system without noise, 2000 pairs of input/output data were used as data sets for uniform sampling in 20 cycles, and the learning rate was designed as a linear attenuation sequence (in 50 iterations, the learning rate decreases from $\eta_0 = 0.5$ to $\eta_{\mathrm{end}} = 0.02$). The algorithm in this paper was used to estimate 10 parameters at the same time. Table 1, after 50 iterations, shows the estimated values and mean square deviation of parameters.

Table 1. Parameter estimation of a noiseless system.

$u_1(t)$	$u_2(t)$	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	**MSE**
sine	**sine**	0.5002	0.8025	1.0003	1.0034	1.0000	0.2006	0.5010	1.0004	1.0018	0.9991	2.351E-06
sine	**square**	0.5000	0.8000	1.0000	1.0000	1.0000	0.1996	0.4982	1.0182	0.9677	1.0473	0.0003
square	**square**	0.4973	0.8760	1.0110	1.0031	1.0153	0.2013	0.5072	1.0354	0.9744	1.0840	0.0015

The inputs $u_1(t)$ and $u_2(t)$ of the system are either a sine wave or square wave with an amplitude of 2. Figure 2 shows the difference between the measured value $y_1(t)$ (the real output value of Formula (6.1)) and the output value $\hat{y}_1(t)$ obtained using the parameter estimator when inputs $u_1(t)$ and $u_2(t)$ are both sine waves. In the same way, Figure 3 shows the difference between measured value $y_2(t)$ and output value $\hat{y}_2(t)$ when inputs $u_1(t)$ and $u_2(t)$ are both sine waves. Figure 4 shows the difference between measured value $y_1(t)$ and output value $\hat{y}_1(t)$ when input $u_1(t)$ is a sine wave and $u_2(t)$ is a square wave.

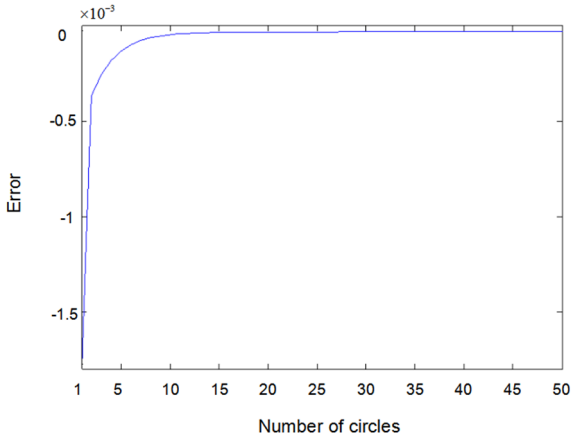

Figure 2. Error of $y_1(t)$ with sine–sine input.

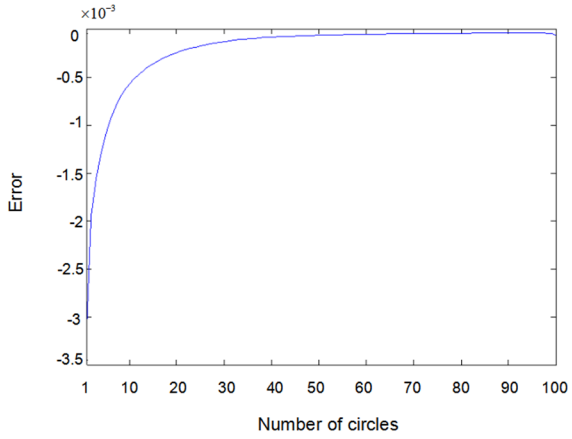

Figure 3. Error of $y_2(t)$ with sine–sine input.

Figure 5 shows the difference between measured value $y_2(t)$ and output value $\hat{y}_2(t)$ when input $u_1(t)$ is a sine wave and $u_2(t)$ is square wave. Figure 6 shows the difference between measured value $y_1(t)$ and output value $\hat{y}_1(t)$ when input $u_1(t)$ and $u_2(t)$ are both square waves. Figure 7 shows the difference between measured value $y_2(t)$ and output value $\hat{y}_2(t)$ when input $u_1(t)$ and $u_2(t)$ are both square waves. It can be seen that when the inputs are both sine waves, the accuracy of parameter estimation is the highest, while when the inputs are square waves, the estimation accuracy of the parameters is relatively lower. This is because when the inputs are square waves, the output of the system has an overshoot, that is to say, the observation value of the system itself has an error. Training

the network with error data will certainly lead to an error of parameter estimation; especially the estimation error of θ_2 is the highest. Here, θ_2 is the parameter of $y_2^3(t-2)$ because $y_2^3(t-2)$ is the third power of the output, which further amplifies the error. Substituting the value of $y_2^3(t-2)$ into the update of θ_2 would inevitably lead to an estimation error.

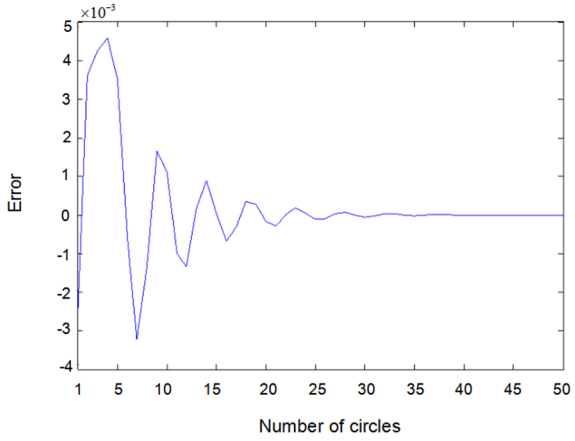

Figure 4. Error of $y_1(t)$ with sine–square input.

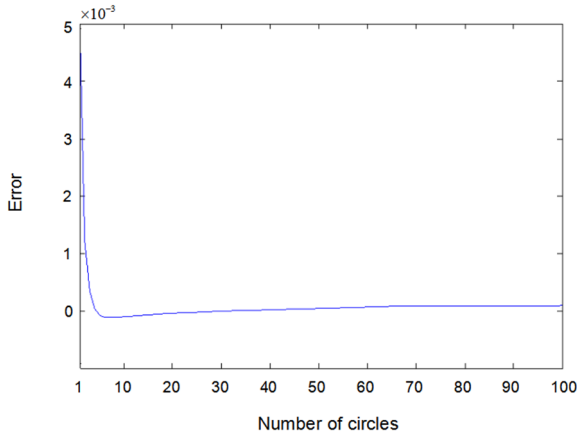

Figure 5. Error of $y_2(t)$ with sine–square input.

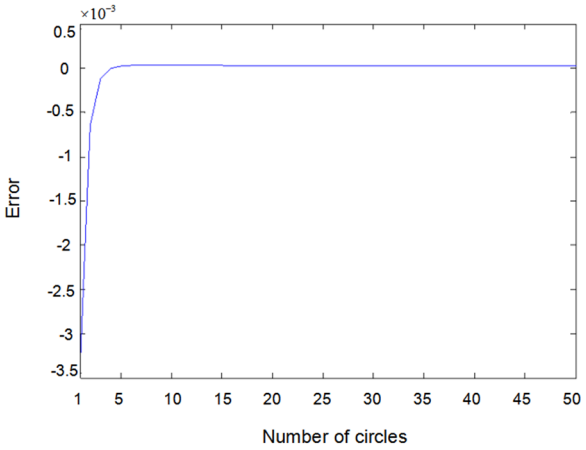

Figure 6. Error of $y_1(t)$ with square–square input.

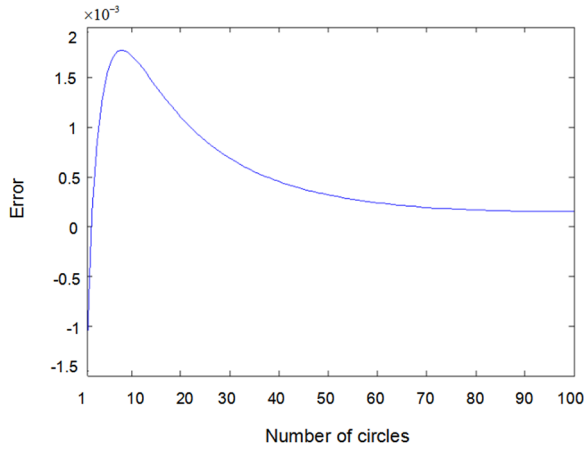

Figure 7. Error of $y_2(t)$ with square–square input.

For a system with noise interference, it is more difficult to estimate its parameters because of the error of the measured value itself [26,27]. In order to further verify the performance of the estimator, a system with noise was selected for parameter estimation, that is, by adding a noise signal to the original system. The noises $r_1(t)$ and $r_2(t)$ were random uniform sequences, where each mean value was zero, each variance was 0.1, and each signal-to-noise ratio was 10. We repeated the previous experiment for the system with noise, and the experimental results are shown in Table 2. The difference between the measured value and the estimated value of the system output is shown in Figures 8–13.

In order to detect the structure of the model, 20 items including 10 items in Formula (6.1) were used as the whole items set of models. The newly added numerator term is of order 1, the denominator term is of order 2, and the input lag and output lag are both of order 1. Using the knock-out algorithm in Section 4, the final 10 items of the model are in good agreement with those in Formula (6.1).

From the above experimental results, the estimation accuracy of the algorithm proposed in this paper is acceptable, and the mean square deviations are all less than 0.003. This level of error is acceptable.

Table 2. Parameter estimation of a noisy system.

$u_1(t)$	$u_2(t)$	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	*MSE*
sine	**sine**	0.5003	0.8041	1.0005	1.0054	1.0001	0.2008	0.5014	1.0005	1.0016	0.9987	5.342E-06
sine	**square**	0.5000	0.8001	1.0000	1.0001	1.0000	0.2045	0.5019	1.073	1.1364	1.0898	0.0032
square	**square**	0.4953	0.8765	1.0085	1.0327	1.0095	0.2969	0.7030	0.9971	1.0007	0.9953	0.0058

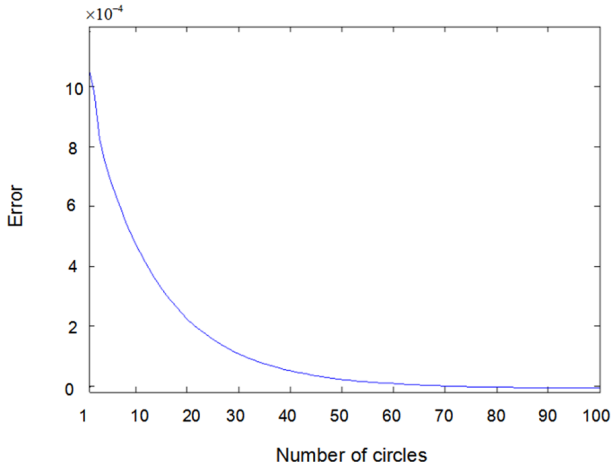

Figure 8. Error of $y_1(t)$ with noise and sine–sine input.

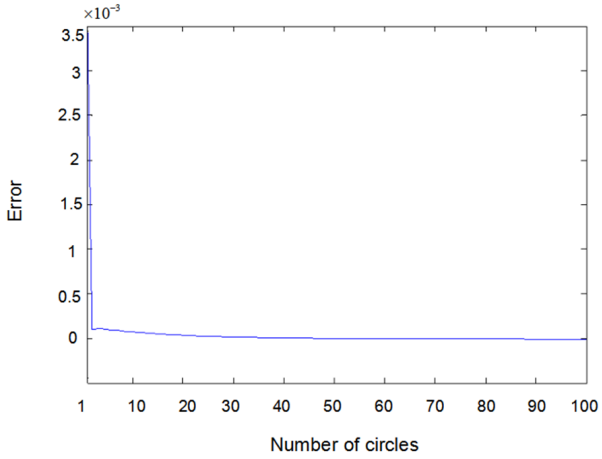

Figure 9. Error of $y_2(t)$ with noise and sine–sine input.

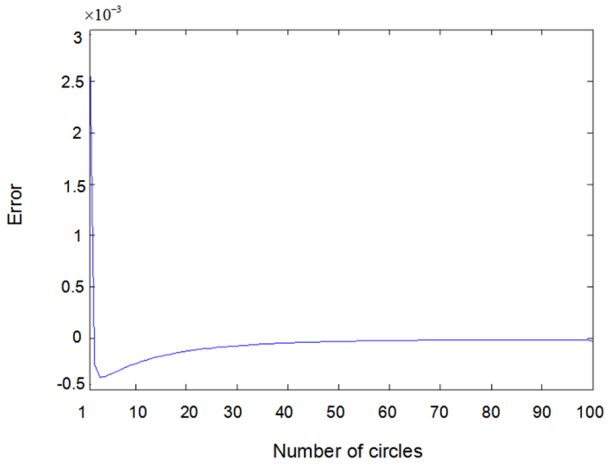

Figure 10. Error of $y_1(t)$ with noise and sine–square input.

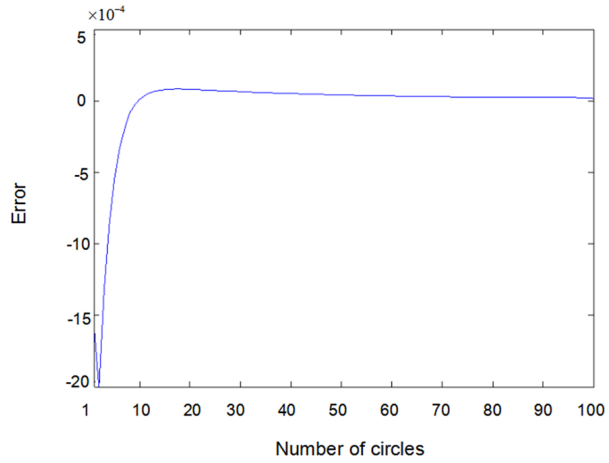

Figure 11. Error of $y_2(t)$ with noise and sine–square input.

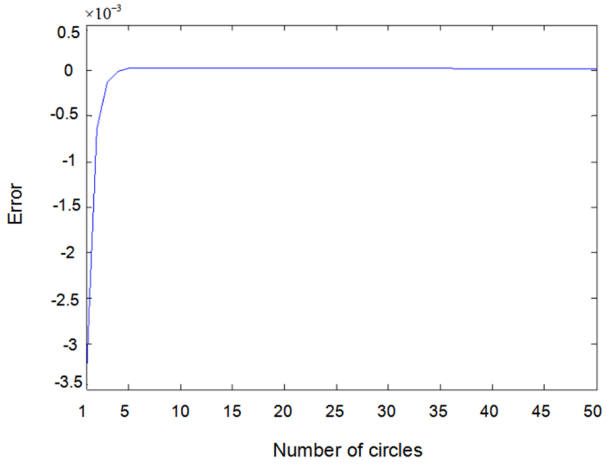

Figure 12. Error of $y_1(t)$ with noise and square–square input.

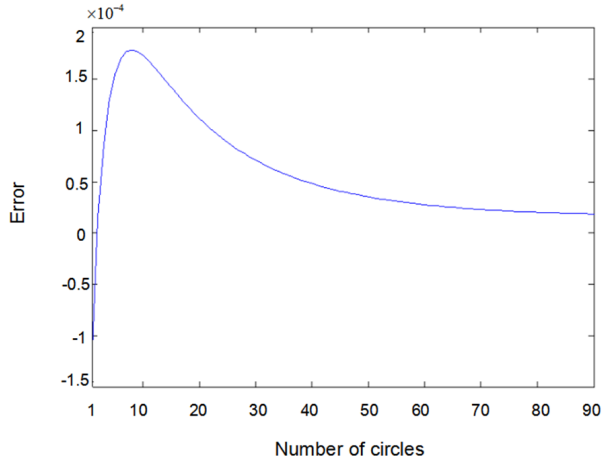

Figure 13. Error of $y_2(t)$ with noise and square–square input.

7. Conclusions

In this paper, the parameter estimation of a SISO rational model was extended to that of a MIMO total non-linear model. A method of parameter estimation of a MIMO non-linear rational model based on a gradient descent algorithm was proposed, and the convergence condition was proposed for the asymmetry of the network. It was proven that the estimator is properly effective by mathematical derivation and simulation. This estimation method has a strong generalization property and could be widely used in many fields, such as non-linear system modeling and control applications. Some systems that could not directly use this method, such as the exponential model describing the change of the kinetic rate constant with the temperature, could first be converted into a rational model and then use the developed estimation method. Some of the future work could be foreseen as (1) estimating the parameters of the state space model based on an artificial neural network, (2) estimating the parameters of a MIMO state space model, (3) estimating the parameters of the non-linear state space model, and (4) estimating the parameters of total non-linear spatial state models.

Author Contributions: Conceptualization, L.L., D.M., Q.Z.; Formal analysis L.L., D.M., Q.Z., A.T.A.; Funding acquisition, A.T.A.; Investigation, D.M., A.T.A., Q.Z; Methodology, L.L., D.M., Q.Z., A.T.A.; Resources, L.L., Q.Z., A.T.A.; Software, L.L., D.M.; Supervision Q.Z., A.T.A.; Visualization, L.L., D.M., A.T.A; Writing – original draft, L.L., D.M., Q.Z.; Writing – review & editing, L.L., D.M., Q.Z., A.T.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by Prince Sultan University, Riyadh, Kingdom of Saudi Arabia. Special acknowledgement to Robotics and Internet-of-Things Lab (RIOTU), Prince Sultan University, Riyadh, Saudi Arabia. We would like to show our gratitude to Prince Sultan University, Riyadh, Saudi Arabia.

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

References

1. Billings, S.A.; Chen, S. Identification of non-linear rational systems using a prediction-error estimation algorithm. *Int. J. Syst. Sci.* **1989**, *20*, 467–494. [CrossRef]
2. Billings, S.A.; Zhu, Q.M. Rational model identification using an extended least-squares algorithm. *Int. J. Control* **1991**, *54*, 529–546. [CrossRef]
3. Sontag, E.D. *Polynomial Response Maps. Lecture Notes in Control & Information Sciences*; Springer: Berlin/Heidelberg, Germany, 1979; Volume 13.
4. Narendra, K.S.; Parthasarathy, K. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Netw.* **2002**, *1*, 4–27. [CrossRef] [PubMed]
5. Zhu, Q.M.; Ma, Z.; Warwick, K. Neural network enhanced generalised minimum variance self-tuning controller for nonlinear discrete-time systems. *IEE Proc. Control Theory Appl.* **1999**, *146*, 319–326. [CrossRef]
6. Billings, S.A.; Zhu, Q.M. A structure detection algorithm for nonlinear dynamic rational models. *Int. J. Control* **1994**, *59*, 1439–1463. [CrossRef]
7. Zhu, Q.M.; Billings, S.A. Recursive parameter estimation for nonlinear rational models. *J. Syst. Eng.* **1991**, *1*, 63–67.
8. Zhu, Q.M.; Billings, S.A. Parameter estimation for stochastic nonlinear rational models. *Int. J. Control* **1993**, *57*, 309–333. [CrossRef]
9. Aguirre, L.A.; Barbosa, B.H.G.; Braga, A.P. Prediction and simulation errors in parameter estimation for nonlinear systems. *Mech. Syst. Signal Process.* **2010**, *24*, 2855–2867. [CrossRef]
10. Huo, M.; Duan, H.; Luo, D.; Wang, Y. Parameter Estimation for a VTOL UAV Using Mutant Pigeon Inspired Optimization Algorithm with Dynamic OBL Strategy. In Proceedings of the 2019 IEEE 15th International Conference on Control and Automation (ICCA), Edinburgh, UK, 16–19 July 2019; pp. 669–674.
11. Zhu, Q.M.; Yu, D.; Zhao, D. An Enhanced Linear Kalman Filter (EnLKF) algorithm for parameter estimation of nonlinear rational models. *Int. J. Syst. Sci.* **2016**, *48*, 451–461. [CrossRef]
12. Türksen, Ö.; Babacan, E.K. Parameter Estimation of Nonlinear Response Surface Models by Using Genetic Algorithm and Unscented Kalman Filter. In *Chaos, Complexity and Leadership 2014*; Erçetin, S., Ed.; Springer Proceedings in Complexity; Springer: Cham, Switzerland, 2016.
13. Billings, S.A.; Mao, K.Z. Structure detection for nonlinear rational models using genetic algorithms. *Int. J. Syst. Sci.* **1998**, *29*, 223–231. [CrossRef]
14. Plakias, S.; Boutalis, Y.S. Lyapunov Theory Based Fusion Neural Networks for the Identification of Dynamic Nonlinear Systems. *Int. J. Neural Syst.* **2019**, *29*, 1950015. [CrossRef] [PubMed]
15. Kumar, R.; Srivastava, S.; Gupta, J.R.P.; Mohindru, A. Diagonal recurrent neural network based identification of nonlinear dynamical systems with lyapunov stability based adaptive learning rates. *Neurocomputing* **2018**, *287*, 102–117. [CrossRef]
16. Chen, S.; Liu, Y. Robust Distributed Parameter Estimation of Nonlinear Systems with Missing Data over Networks. *IEEE Trans. Aerosp. Electron. Syst.* **2019**. [CrossRef]
17. Zhu, Q.M. An implicit least squares algorithm for nonlinear rational model parameter estimation. *Appl. Math. Model.* **2005**, *29*, 673–689. [CrossRef]
18. Zhu, Q.M. A back propagation algorithm to estimate the parameters of non-linear dynamic rational models. *Appl. Math. Model.* **2003**, *27*, 169–187. [CrossRef]
19. Zhu, Q.M.; Wang, Y.; Zhao, D.; Li, S.; Billings, S.A. Review of rational (total) nonlinear dynamic system modelling, identification, and control. *Int. J. Syst. Sci.* **2015**, *46*, 2122–2133. [CrossRef]
20. Leung, H.; Haykin, S. Rational function neural network. *Neural Comput.* **1993**, *5*, 928–938. [CrossRef]
21. Jain, R.; Narasimhan, S.; Bhatt, N.P. A priori parameter identifiability in models with non-rational functions. *Automatica* **2019**, *109*, 108513. [CrossRef]
22. Kambhampati, C.; Mason, J.D.; Warwick, K. A stable one-step-ahead predictive control of nonlinear systems. *Automatica* **2000**, *36*, 485–495. [CrossRef]
23. Kumar, R.; Srivastava, S.; Gupta, J.R.P. Lyapunov stability-based control and identification of nonlinear dynamical systems using adaptive dynamic programming. *Soft Comput.* **2017**, *21*, 4465–4480. [CrossRef]

24. Ge, H.W.; Du, W.L.; Qian, F.; Liang, Y.C. Identification and control of nonlinear systems by a time-delay recurrent neural network. *Neurocomputing* **2009**, *72*, 2857–2864. [CrossRef]
25. Verdière, N.; Zhu, S.; Denis-Vidal, L. A distribution input–output polynomial approach for estimating parameters in nonlinear models. Application to a chikungunya model. *J. Comput. Appl. Math.* **2018**, *331*, 104–118. [CrossRef]
26. Li, F.; Jia, L. Parameter estimation of hammerstein-wiener nonlinear system with noise using special test signals. *Neurocomputing* **2019**, *344*, 37–48. [CrossRef]
27. Chen, C.Y.; Gui, W.H.; Guan, Z.H.; Wang, R.L.; Zhou, S.W. Adaptive neural control for a class of stochastic nonlinear systems with unknown parameters, unknown nonlinear functions and stochastic disturbances. *Neurocomputing* **2017**, *226*, 101–108. [CrossRef]