Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 959507, 9 pages
http://dx.doi.org/10.1155/2014/959507

Research Article

Identification of Nonlinear Dynamic Systems Using Hammerstein-Type Neural Network

Hongshan Yu,[1,2] Jinzhu Peng,[3] and Yandong Tang[2]

[1] *College of Electrical and Information Engineering, Hunan University, Changsha 410082, China*
[2] *State Key Laboratory of Robotics, Shenyang 110016, China*
[3] *School of Electrical Engineering, Zhengzhou University, Zhengzhou 450001, China*

Correspondence should be addressed to Hongshan Yu; yuhongshancn@hotmail.com

Received 31 May 2014; Revised 15 September 2014; Accepted 29 September 2014; Published 19 October 2014

Academic Editor: Jun-Juh Yan

Hammerstein model has been popularly applied to identify the nonlinear systems. In this paper, a Hammerstein-type neural network (HTNN) is derived to formulate the well-known Hammerstein model. The HTNN consists of a nonlinear static gain in cascade with a linear dynamic part. First, the Lipschitz criterion for order determination is derived. Second, the backpropagation algorithm for updating the network weights is presented, and the stability analysis is also drawn. Finally, simulation results show that HTNN identification approach demonstrated identification performances.

1. Introduction

Identification of nonlinear dynamic systems has been one of the most interesting research areas in engineering. Similar as the well-known Wiener model, Hammerstein model is also well known and the most widely used for modeling of various processes [1, 2], which comprises of a static nonlinear block preceding a dynamic linear one [3]. This type of model was called block-oriented nonlinear model [4]. Different from black-box models, the block-oriented model was regarded as gray-box model which has clear physical interpretation, and its steady-state part describes the gain of the system [5]. The major drawback of these classic identification techniques is that they could have many constraints for nonlinear systems with time-varying parameters and uncertainties.

It has been proven that neural networks (NN) can generally approximate any smooth nonlinear function well [2, 3]. During the last decade, they have been successfully and efficiently applied to identify and control the nonlinear systems due to high adaptation, self-organization, and fast parallel processing of real-time information. However, for general neural networks, the model structures have no relation with the physical nature of the process and the model parameters have no physical interpretation, similar to black-box models [5]. Therefore, many works were focused on integrating NN with Wiener or Hammerstein models. As a result, the model structures and the model parameters of NN have clear physical interpretation while the well-developed classic identification methodologies would have adaptive abilities.

There are two ways to integrate NN with Wiener or Hammerstein models. One is to use NN to formulate the system static nonlinearities of Wiener or Hammerstein models. Chen et al. [6] used a simple linear model to represent the dynamic part and a neural network to represent the nonlinear static part. The dynamic linear part was replaced by Laguerre filters and the nonlinear static part was described as a neural network [2]. Tötterman and Toivonen [7] used support vector regression to identify nonlinear Wiener systems. The linear block was expanded in terms of Laguerre or Kautz filters, and the static nonlinear block was determined using support vector machine regression. Saha et al. [8] developed a Weiner-type nonlinear black-box model for capturing dynamics of open loop stable MIMO nonlinear systems, where quadratic polynomials and NN were used for constructing the nonlinear output map. The models were then used in nonlinear model predictive control [8]. Al-Duwaish et al. [9] used a hybrid model consisting of a linear autoregressive moving

average (ARMA) model and a NN to, respectively, represent the dynamic linear block and the static nonlinear element of Wiener model. The other one is to formulate the Wiener or Hammerstein model entirely by a multilayer NN; that is, the dynamic linear block and static nonlinear block of Wiener or Hammerstein model are both represented by the NN. Janczak [4] designed a NN to formulate the Hammerstein model, which was composed of one hidden layer with nonlinear nodes and one linear output node. Wu et al. [10] proposed a Hammerstein neural network compensator to identify a dynamic process. Peng and Dubay [11] proposed a Wiener-type neural network to identify nonlinear dynamic processes. However, most of those investigations did not give the system order determination or stability analysis.

In this paper, a multilayer neural network is used to formulate the traditional Hammerstein model, which is called Hammerstein-type neural network (HTNN). The HTNN formulates the Hammerstein model with a nonlinear static block in cascade with a linear dynamic block, the weights in which are corresponding with the parameters of Hammerstein model. Then, an identification methodology based on HTNN is presented and applied to nonlinear SISO systems. To determine the order and weights of HTNN, the Lipschitz criterion and backpropagation (BP) algorithm are derived, respectively. Furthermore, the stability analysis is drawn. Finally, the proposed identification method is tested on several nonlinear plants.

In [2, 6–9], neural networks were used to formulate the system static nonlinearities of Wiener or Hammerstein models, while in our design, a multilayer neural network is used to formulate the Hammerstein model entirely; that is, the dynamic linear block and static nonlinear block of Hammerstein model were both represented by the neural network. In this way, the parameters of Hammerstein model can be obtained by training HTNN using an adequate training algorithm. Moreover, the model order determination and convergence stability are also drawn.

The rest of this paper is organized as follows. In Section 2, Hammerstein model is described. The design of the HTNN for identification is given in Section 3, the Lipschitz criterion is presented for order determination, and the learning algorithm is analyzed. The convergence analysis of HTNN is given in Section 4. The simulation results are given in Section 5. The conclusions are drawn in Section 6.

2. Hammerstein Model

Many industrial processes can be described by Wiener model or Hammerstein models [1]. As shown in Figure 1, a general Hammerstein model consists of a nonlinear static block and a linear dynamic block. The nonlinear static block is given by

$$x(t) = f(u(t)), \tag{1}$$

where $u(t)$ is the input variable; $f(\cdot)$ represents the nonlinear component of the Hammerstein model; $x(t)$ is a nonmeasured intermediate variable that does not necessarily have a physical meaning.

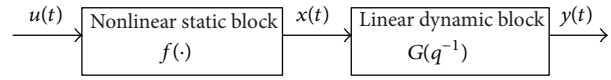

Figure 1: The generally Hammerstein model.

And the linear dynamic block can be described as

$$y(t) = G\left(q^{-1}\right)x(t) = \frac{A\left(q^{-1}\right)}{B\left(q^{-1}\right)}x(t) \tag{2}$$

with

$$\begin{aligned} A\left(q^{-1}\right) &= a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a} \\ B\left(q^{-1}\right) &= 1 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}, \end{aligned} \tag{3}$$

where $y(t)$ is the output variable; q^{-1} is the unit delay operator; n_a and n_b are the orders of the linear dynamics and generally $n_b \le n_a$.

There are many methods to identify the model parameters of Wiener model or Hammerstein model, such as the least squares method or its recursive version [12], maximum likelihood methods [13], correlation methods [14], linear optimization methods [15], and nonlinear optimization methods [2]. In this study, a multilayer neural network is designed to formulate the Hammerstein model. Therefore, the parameters of the Hammerstein model can be directly obtained through training the multilayer neural network by the BP training algorithm.

3. System Identification Using Hammerstein-Type Neural Network

A general neural network can be regarded as a black-box model [5]. In this section, a Hammerstein-type neural network (HTNN) is designed to formulate the Hammerstein model entirely. To determine the minimal model order, an order determination method based on Lipschitz quotients [16] is utilized. And for weights updating, the BP algorithm is presented.

3.1. Hammerstein-Type Neural Network. As shown in Figure 2, a multilayer neural network called Hammerstein-type neural network is designed, which consists of a nonlinear static element and a single linear node with two tapped delay lines forming the model of the linear dynamic element.

In Figure 2, $u(t)$ is the input variable, $\widehat{y}(t)$ is the output variable, and $\widehat{x}(t)$ is a nonmeasured intermediate variable that does not necessarily have a physical meaning. Λ^i, $i = 1, 2, \ldots, p$, means the order of input u, the parameters $\widehat{a}_i$, $\widehat{b}_j$, and $\widehat{c}_k$ mean the weights of HTNN, and q^{-1} is the unit delay operator. Giving an input signal $u(t)$, initial condition of $x(t)$ and $y(t)$ ($t \le 0$), and initial values of $\widehat{a}_i$, $\widehat{b}_j$, and $\widehat{c}_k$, we can obtain the HTNN output $\widehat{y}(t)$ ($t > 0$).

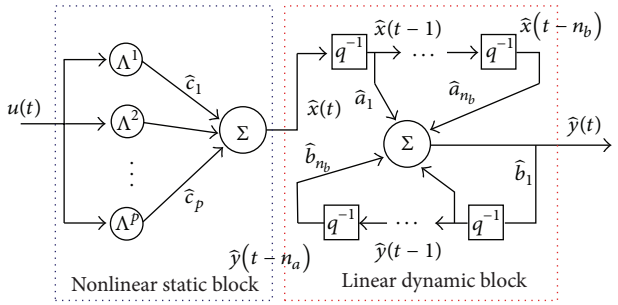

Figure 2: The structure of Hammerstein-type neural network.

As shown in Figure 2, a polynomial function is used in the nonlinear static block, the output of hidden layer $\hat{x}(t)$ can be expressed as

$$\hat{x}(t) = f(u(t)) = \sum_{k=1}^{p} \hat{c}_k u^k(t), \tag{4}$$

and the output $\hat{y}(t)$ can be expressed as

$$\begin{aligned} \hat{y}(t) = & -\hat{b}_1 \hat{y}(t-1) - \cdots - \hat{b}_{n_b} \hat{y}(t-n_b) \\ & + \hat{a}_1 \hat{x}(t-1) + \cdots + \hat{a}_{n_a} \hat{x}(t-n_a) \\ = & -\sum_{j=1}^{n_b} \hat{b}_j \hat{y}(t-j) + \sum_{i=1}^{n_a} \hat{a}_i \hat{x}(t-i), \end{aligned} \tag{5}$$

where $\hat{c}_k$ for $k = 1, \ldots, p$, $\hat{b}_j$ for $j = 1, \ldots, n_b$, and $\hat{a}_i$ for $i = 1, \ldots, n_a$ are the weights of HTNN, which are associated with the parameters of Hammerstein model in (1) and (2). As a result, the parameters of the Hammerstein model can be expressed directly as the weights of the dynamic neural network. Then, the identified Hammerstein model can be obtained by training HTNN using an adequate training algorithm.

3.2. Model Order Determination. Usually, before training a neural network, it is important to determine how many neurons are in the network. In the HTNN, the structure of linear dynamic part is determined by n_a and n_b. As the most popular statistical model selection criterion, Akaike information criterion (AIC) [17] is a common method to determine the model order. However, the AIC for model order determination is usually subjected to a complex optimization process. The Lipschitz quotients criterion, which was proposed by He and Asada [16], can be utilized to determine the model order by analyzing the input-output data. Peng and Dubay [11] introduced the Lipschitz quotients criterion to determine the model order in a Wiener neural network.

Consider a general nonlinear SISO dynamic system, which can be described as

$$y(t) = g(y(t-1), \ldots, y(t-n_b), u(t-1), \ldots, u(t-n_a)), \tag{6}$$

where $y(t)$ and $u(t)$ are the output and input variables of the nonlinear dynamic system, and n_b and n_a are the true orders of the output and input, respectively; $g(\cdot)$ is a nonlinear function assumed to be continuous and smooth.

Rewriting (6) in a compact form gives

$$y = g(\chi_1, \chi_2, \ldots, \chi_n), \tag{7}$$

where $n = n_b + n_a$ is the number of input variables; let $\chi = [\chi_1, \chi_2, \ldots, \chi_n]^{\mathrm{T}}$. To reconstruct the nonlinear function $g(\cdot)$ from the input-output data pairs $[\chi(i), y(i)]_{i=1}^{N_{\text{set}}}$, where N_{set} is the number of data sets used for the model order determination. Define the Lipschitz quotient L_{ij} as

$$L_{ij} = \frac{|y(i) - y(j)|}{|\chi(i) - \chi(j)|}, \quad i \neq j, \tag{8}$$

where $|\chi(i) - \chi(j)|$ is the distance of two points in the input space and $|y(i) - y(j)|$ is the difference between $g(\chi(i))$ and $g(\chi(j))$. For data points with a small distance $|\chi(i) - \chi(j)|$ between them, the Lipschitz quotient $L_{ij}^{(n)}$ can be rewritten as

$$L_{ij}^{(n)} = \frac{|\delta y|}{\sqrt{(\delta\chi_1)^2 + (\delta\chi_2)^2 + \cdots + (\delta\chi_n)^2}}, \tag{9}$$

where $\delta y = y(i) - y(j)$ and $\delta\chi_r = \chi_r(i) - \chi_r(j)$ for $r = 1, \ldots, n$ and the subscript n in $L_{ij}^{(n)}$ represents the number of input variables in (7). According to the investigation in He and Asada [16], the values of $L_{ij}^{(n)}$ can be used as indicator when one or more input variables are missing or in the case when one or more redundant input variables are included. For instance, if one of input variables, χ_n, is missing from the input set, the Lipschitz quotient $L_{ij}^{(n-1)}$ will be considerably larger than $L_{ij}^{(n)}$ or even unbounded. On the contrary, when an input variable χ_{n+1} is included and the Lipschitz quotients $L_{ij}^{(n)}$ and $L_{ij}^{(n+1)}$ are calculated, if χ_{n+1} is a redundant input variable, there will be a slight difference between $L_{ij}^{(n)}$ and $L_{ij}^{(n+1)}$ but not significant.

To avoid the effect of measurement noise, the following index [16] is utilized to determine an appropriate order:

$$L^{(n)} = \left(\sqrt{n} \prod_{s=1}^{m} L^{(n)}(s) \right)^{1/m}, \tag{10}$$

where $L^{(n)}(s)$ is the sth largest Lipschitz quotient among all $L_{ij}^{(n)}$ with the n input variables $(\chi_1, \ldots, \chi_n)$, and parameter m is a positive number, usually selected to be $m = 0.01N_{\text{set}} \sim 0.02N_{\text{set}}$. For testing purposes, the stop criterion can be defined as [18]

$$\frac{\left| L^{(n+1)} - L^{(n)} \right|}{\max\left(1, \left| L^{(n)} \right|\right)} < \epsilon, \tag{11}$$

where $\epsilon > 0$ is a prespecified threshold.

It should be noted that the number of nodes in the nonlinear layer, p, is chosen manually. This is because, from empirical experience, for most cases, $3 \leq p \leq 6$ can be chosen [11].

3.3. Learning Algorithm. During the neural network learning procedures, the weights are updated by the BP algorithm. An error function can be defined as

$$J(\mathbf{w}, t) = \frac{1}{2}\left(y(t) - \widehat{y}(t)\right)^2 = \frac{1}{2}e(t)^2, \tag{12}$$

where $e(t) = y(t) - \widehat{y}(t)$ is the identification error; $y(t)$ and $\widehat{y}(t)$ are the actual system output and the neural network output, respectively. Let $\mathbf{w}$ be the weights, which consist of the parameters $\widehat{c}_k$, $\widehat{b}_j$, and $\widehat{a}_i$. Applying the steepest descent method, the optimization target is characterized to minimize the error function equation (12) with respect to the weights of the network. Consider

$$\frac{\partial J}{\partial w} = -e(t)\frac{\partial \widehat{y}(t)}{\partial w}, \tag{13}$$

where w represents the element $\mathbf{w}$.

The general update rule is expressed as

$$\begin{aligned} w(t+1) &= w(t) + \Delta w(t) \\ &= w(t) - \eta\frac{\partial J}{\partial w} \\ &= w(t) + \eta e(k)\frac{\partial \widehat{y}(t)}{\partial w}, \end{aligned} \tag{14}$$

where η is the training rate.

Considering the element of $\widehat{y}(t-j)$ also being function of the weights $\widehat{a}_i$ and $\widehat{b}_j$, the partial derivative of the intermediate variables $\widehat{y}(t)$ to linear dynamic block weights $\widehat{a}_i$, $\widehat{b}_j$ and intermediate variables $\widehat{x}(t-i)$ can be calculated as

$$\frac{\partial \widehat{y}(t)}{\partial \widehat{b}_j} = -\widehat{y}(t-j) - \sum_{m=1}^{n_b}\widehat{b}_m\frac{\partial \widehat{y}(t-m)}{\partial \widehat{b}_j}, \quad j = 1, \ldots, n_b, \tag{15}$$

$$\frac{\partial \widehat{y}(t)}{\partial \widehat{a}_i} = x(t-i) - \sum_{m=1}^{n_b}\widehat{b}_m\frac{\partial \widehat{y}(t-m)}{\partial \widehat{a}_i}, \quad i = 1, \ldots, n_a, \tag{16}$$

$$\frac{\partial \widehat{y}(t)}{\partial \widehat{x}(t-i)} = \widehat{a}_i - \sum_{m=1}^{n_b}\widehat{b}_m\frac{\partial \widehat{y}(t-m)}{\partial \widehat{x}(t-i)}, \quad i = 1, \ldots, n_a. \tag{17}$$

The partial derivative of intermediate variables $\widehat{x}(t)$ to weights $\widehat{c}_k$ can be calculated as

$$\frac{\partial \widehat{x}(t)}{\partial \widehat{c}_k} = u^k(t), \quad k = 1, \ldots, p. \tag{18}$$

According to (17) and (18), the partial derivative of the HTNN output $\widehat{y}(t)$ to weights $\widehat{c}_k$ can be calculated as

$$\begin{aligned} \frac{\partial \widehat{y}(t)}{\partial \widehat{c}_k} &= \sum_{i=1}^{n_a}\left(\frac{\partial \widehat{y}(t)}{\partial \widehat{x}(t-i)}\cdot\frac{\partial \widehat{x}(t-i)}{\partial \widehat{c}_k}\right) \\ &= \sum_{i=1}^{n_a}\widehat{a}_i u^k(t-i) - \sum_{m=1}^{n_b}\widehat{b}_m\frac{\partial \widehat{y}(t-m)}{\partial \widehat{c}_k}, \\ &\qquad k = 1, \ldots, p. \end{aligned} \tag{19}$$

According to (14), the update law of $\widehat{c}_k$, $\widehat{a}_i$, and $\widehat{b}_j$ can be calculated as

$$\widehat{a}_i(t+1) = \widehat{a}_i(t) + \eta\cdot\widehat{e}(t)\frac{\partial \widehat{y}(t)}{\partial \widehat{a}_i}, \quad i = 1, \ldots, n_a, \tag{20}$$

$$\widehat{b}_j(t+1) = \widehat{b}_j(t) + \eta\cdot\widehat{e}(t)\frac{\partial \widehat{y}(t)}{\partial \widehat{b}_j}, \quad j = 1, \ldots, n_b, \tag{21}$$

$$\widehat{c}_k(t+1) = \widehat{c}_k(t) + \eta\cdot\widehat{e}(t)\frac{\partial \widehat{y}(t)}{\partial \widehat{c}_k}, \quad k = 1, \ldots, p, \tag{22}$$

where the partial derivatives in (20), (21), and (22) are shown in (16), (15), and (19), respectively. From the above analysis, the structure of the system identification using the HTNN is shown in Figure 3.

4. Convergence Analysis

In the training procedures of HTNN, a proper choice of training rate η is usually required in the update rules of (20)–(22). Too small η guarantees convergence but with slow training speed, while too big η leads to being unstable. In this section, the approach on selecting properly η is developed.

A discrete type Lyapunov function can be defined as [19]

$$V(t) = \frac{1}{2}e^2(t). \tag{23}$$

Therefore, the time change of the Lyapunov function can be obtained as

$$\begin{aligned} \Delta V(t) &= V(t+1) - V(t) \\ &= \frac{1}{2}\left[e^2(t+1) - e^2(t)\right] \\ &= \Delta e(t)\left[e(t) + \frac{\Delta e(t)}{2}\right]. \end{aligned} \tag{24}$$

The error difference can be represented as

$$e(t+1) = e(t) + \Delta e(t) = e(t) + \left(\frac{\partial e(t)}{\partial w}\right)^{\mathrm{T}}\cdot\Delta w. \tag{25}$$

From the update rule of (13), we have

$$\Delta w = -\eta\cdot e(t)\cdot\frac{\partial e(t)}{\partial w} = \eta\cdot e(t)\cdot\frac{\partial \widehat{y}(t)}{\partial w}. \tag{26}$$

Theorem 1. *Assume that η is the learning rate for the weights of HTNN and $\beta_{\max}$ is defined as $\beta_{\max} = \max_t\|\beta(t)\|$, where $\beta(t) = \partial\widehat{y}(t)/\partial w$ and $\|\cdot\|$ is the Euclidean norm. Then, the convergence is guaranteed if η is satisfied as*

$$0 < \eta < \frac{2}{\beta_{\max}^2}. \tag{27}$$

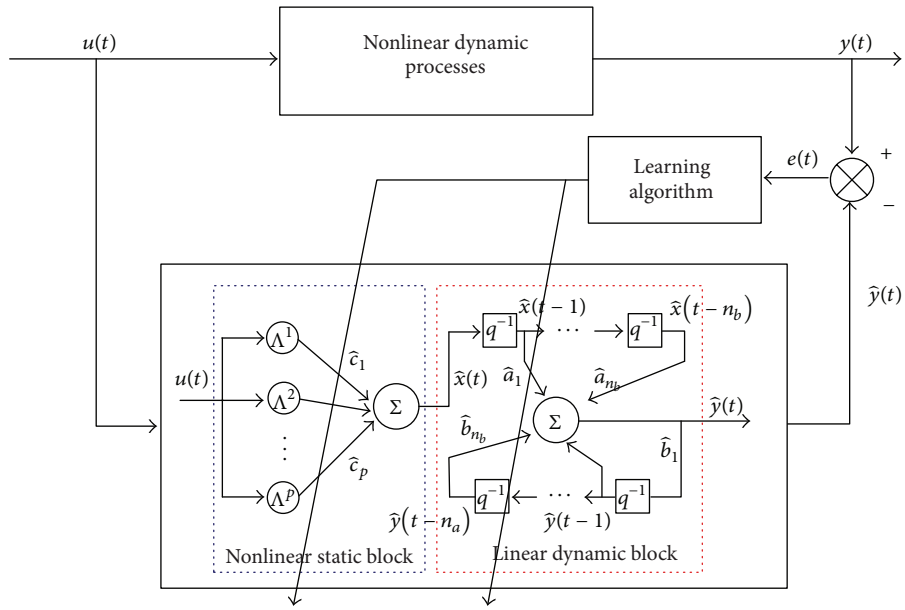

Figure 3: Model identification using HTNN.

Proof. According to (23)-(24), $\Delta V(t)$ can be represented as

$$\begin{aligned}\Delta V(t) &= \Delta e(t)\left[e(t)+\frac{\Delta e(t)}{2}\right]\\ &= \left(\frac{\partial e(t)}{\partial w}\right)^{\mathrm{T}}\eta e(t)\frac{\partial \hat{y}(t)}{\partial w}\\ &\quad\cdot\left[e(t)+\frac{1}{2}\left(\frac{\partial e(t)}{\partial w}\right)^{\mathrm{T}}\eta e(t)\frac{\partial \hat{y}(t)}{\partial w}\right].\end{aligned} \tag{28}$$

Substituting the equation $\partial e(t)/\partial w = -\partial \hat{y}(t)/\partial w$ into (28) yields

$$\Delta V(t) = -\eta e^2(t)\left\|\frac{\partial \hat{y}(k)}{\partial w}\right\|^2 + \frac{1}{2}\eta^2 e^2(t)\left\|\frac{\partial \hat{y}(t)}{\partial w}\right\|^4. \tag{29}$$

Define $\beta_{\max} = \max_t\|\beta(t)\|$, where $\beta(t) = \partial \hat{y}(t)/\partial w$; we obtain

$$\begin{aligned}\Delta V(t) &= -\frac{1}{2}\|\beta(t)\|^2\eta\left(2-\eta\|\beta(t)\|^2\right)e^2(t)\\ &\le -\frac{1}{2}\|\beta(t)\|^2\eta\left(2-\eta\beta_{\max}^2\right)e^2(t).\end{aligned} \tag{30}$$

Since $V(t) > 0$ for all time t, the convergence of training algorithm means $\Delta V(t) < 0$. Referring to (30), it implies that (27) is satisfied. It should be pointed that optimal convergence via maximum learning rate corresponds to $\eta^* = 1/\beta_{\max}^2$, which is the upper half of the limit in (27). □

5. Simulation Examples

In this section, three examples are utilized to illustrate the HTNN identifier. The first is a Hammerstein process which is utilized to demonstrate the HTNN of fitting the Hammerstein model. The second and the third are a nonlinear dynamic system and an industry process, which are utilized to demonstrate the HTNN can be used to identify nonlinear dynamic systems and processes, respectively.

The implementation procedure of the HTNN for nonlinear system identification is itemized as follows.

Step 1. Select the input-output variables $u(t)$ and $y(t)$ and the structures (n_a and n_b) of the HTNN according to Section 3.2.

Step 2. Select the order p of polynomial function; note that the value range of integer p is $3 \le p \le 6$.

Step 3. Model the nonlinear system using HTNN as in (4) and (5), and train the HTNN using the training datasets according to (20)–(22) to obtain the weights $\hat{a}_i$, $\hat{b}_j$, and $\hat{c}_k$.

Step 4. Calculate the error function J in (12). If J is less than a limiting value within a given number of training iterations, it means that the model can be accepted; otherwise go to Step 2 to reselect the value of integer p.

Example 1. Nonlinear Hammerstein model identification is as follows.

The following Hammerstein model is described as

$$\begin{aligned}x(t) &= u(t) - 0.4u(t)^2 + 0.1u(t)^3;\\ y(t) &= 1.4138y(t-1) - 0.6065y(t-2) + 0.1044x(t-1)\\ &\quad + 0.0883x(t-2).\end{aligned} \tag{31}$$

A total of 1000 data pairs of an *i.i.d.* uniform sequence within the limits $[-0.5, 0.5]$ were generated to train the HTNN. In the order determination procedure, firstly, it gives the input $\chi_1 = y(t-1)$ and computes the Lipschitz quotient

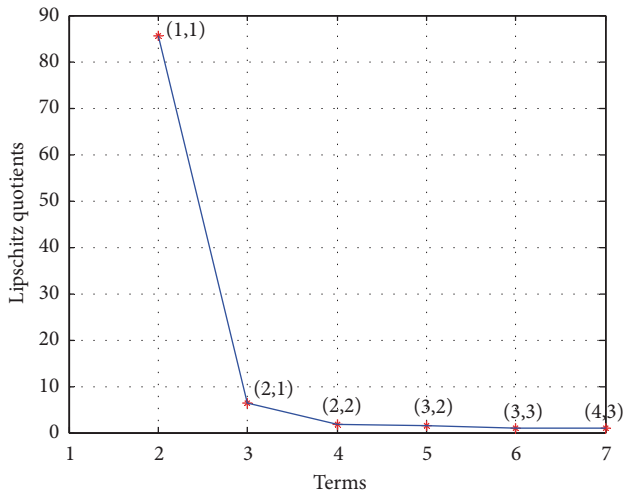

Figure 4: Values of the order determination.

$L^{(1,0)} = +\infty$ and then sets $\chi_2 = u(t-1)$; the Lipschitz quotient $L^{(1,1)} = 85.55$. For $\chi_3 = y(t-2)$ and $\chi_4 = u(t-2)$, the corresponding Lipschitz quotients $L^{(2,1)} = 6.312$ and $L^{(2,2)} = 1.853$ are decreased significantly. Then it sets $\chi_5 = y(t-3)$, $\chi_6 = u(t-3)$, and $\chi_7 = y(t-4)$, the corresponding Lipschitz quotients $L^{(3,2)} = 1.452$, $L^{(3,3)} = 1.002$, and $L^{(4,3)} = 0.9799$ are not significantly different from $L^{(2,2)}$, and the stop criterion equation (11) is satisfied, where the threshold is chosen as $\epsilon = 0.5$. Figure 4 shows the values of the Lipschitz quotients for different order; the best order of the system is (2,2); that is, $n_a = 2$ and $n_b = 2$. From (31), the true order of the system is (2,2).

In this case, the number of neurons in nonlinear static block is chosen as $p = 3$. According to the convergence analysis, $\beta_{\max} = 2.534$; therefore, the learning rate can be chosen as $\eta = 0.16$. The initial parameters in (21)–(23) are $y(t) = 0$, $x(t) = 0$, $\partial y(t)/\partial \widehat{a}_i = 0$, $\partial y(t)/\partial \widehat{b}_j = 0$, and $\partial y(t)/\partial \widehat{c}_k = 0$ as $t \le 0$. Then, the testing input signal $u(t) = 0.5\sin(t)$ is used to verify the identification performance of the HTNN. Figure 5 illustrates the output of the plant and the HTNN. And the mean square error (MSE) is 1.497×10^{-3} with 7 tunable parameters using the proposed HTNN.

Example 2. Nonlinear dynamic system identification is as follows.

The following process is a nonlinear dynamic process formulated in a discrete form as [20]

$$y(t) = 0.5\left(y(t-1) + u(t-1)\, e^{-3|y(t-1)|}\right). \quad (32)$$

A total of 500 data pairs which are generated by inputs with a sinusoidal function $1.05\sin(\pi t/45)$ are utilized to train the network.

Similar as Example 1, the values of the Lipschitz quotients are $L^{(1,0)} = +\infty$, and $L^{(1,1)} = 41.45$, $L^{(2,1)} = 2.761$, $L^{(3,1)} = 1.596$, $L^{(2,2)} = 1.389$, $L^{(3,2)} = 1.157$, $L^{(3,3)} = 0.9093$, $L^{(4,3)} = 0.6388$. The stop criterion ends at $L^{(3,1)}$; it implies that the best order of the system is (3,1). Figure 6 shows the values of the Lipschitz quotients for different orders.

Table 1: Comparison among several neural networks for identification.

Network	Parameters	MSE
HTNN	7	1.828×10^{-3}
CReNN	51	4.784×10^{-2}
DFNN	39	1.325×10^{-3}
SHM-LS	7	0.2625

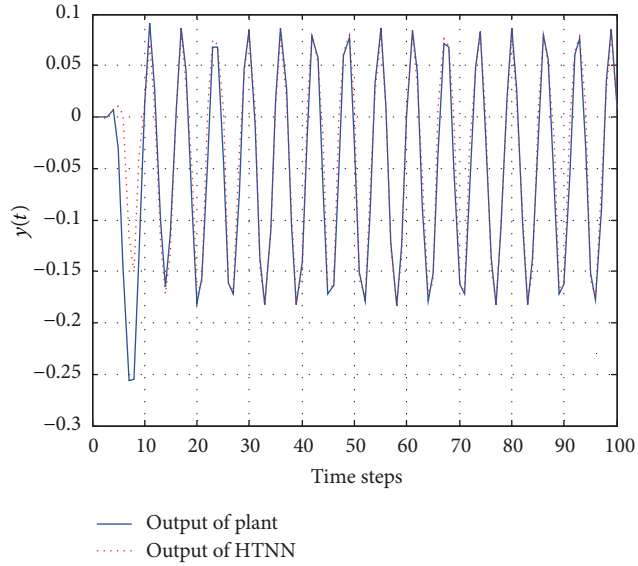

Figure 5: Outputs of the plant (solid) and HTNN (dotted).

In this case, the number of neurons in linear dynamic block is chosen as $n_a = 3$, $n_b = 1$ and the number of neurons in nonlinear static block is chosen as $p = 3$. According to the convergence analysis, $\beta_{\max} = 2.108$; therefore, the learning rate can be chosen as $\eta = 0.22$. The testing input signal $u(t) = \sin(2\pi t)/10$ is used to verify the identification performance of the HTNN.

The proposed HTNN was compared to several neural network based identification methods, the Controllable-Canonical-Form-Based Recurrent Neurofuzzy Network (CReNN) [21] and the Dynamic Fuzzy Neural Network (DFNN) [22]. As for the DFNN model, it is a fuzzy model of three rules, where the two-dimensional input space was partitioned to three clusters and a Gaussian membership function was assigned to each cluster. Also, using least square (LS) algorithm, a standard Hammerstein model (SHM) with $n_a = 3$, $n_b = 1$, and $p = 3$, which is shown in (1) and (2), was applied to identify the nonlinear system for comparison. As shown in Table 1, the results illustrate that the proposed HTNN has the least number of parameters with lower MSE value. Figure 7 shows the output of the plant, the standard Hammerstein model with least square algorithm (SHM-LS), and the HTNN identifier. It can be seen that the standard Hammerstein model with least square algorithm (SHM-LS) can not identify the complex nonlinear system well.

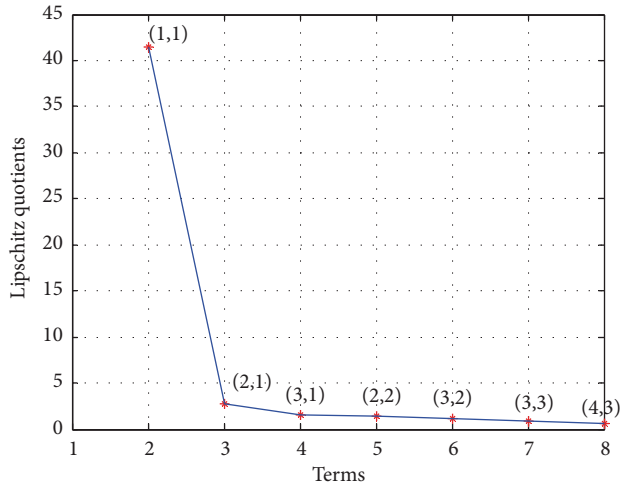

Figure 6: Values of the order determination.

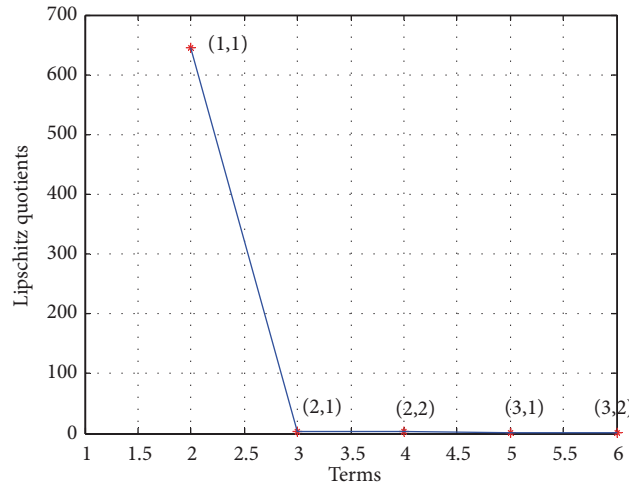

Figure 8: Values of the order determination.

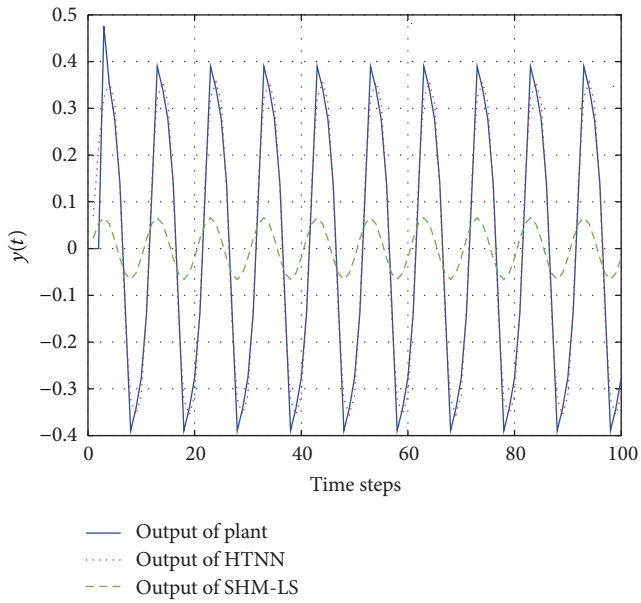

Figure 7: Outputs of the plant (solid), HTNN (dotted), and SHM-LS (dash).

Example 3. Identification for an industry process is as follows.

The proposed HTNN was also used to identify a typical industry process, a continuous stirred tank reactor (CSTR). An irreversible first-order reaction is part of the CSTR, which has the dimensionless mass and energy balances. The system is described by the following equation [23, 24]:

$$\frac{dC_A}{dt} = \frac{Q}{V}\left(C_{Af} - C_A\right) - k_0 C_A e^{-E/RT}$$
$$\frac{dT}{dt} = \frac{Q}{V}\left(T_f - T\right) - \frac{\Delta H}{\rho C_p} k_0 C_A e^{-E/RT} + \frac{UA}{V\rho C_p}\left(T_c - T\right), \quad (33)$$

Table 2: Model parameters of CSTR process.

Variables	Description	Values
Q	Volumetric flow rate	10 (L·min^{-1})
V	Reactor volume	10 (L)
ρ	Density of reaction mixture	1000 (g·L^{-1})
C_{Af}	Feed concentration	1.0 (mol·L^{-1})
T_f	Feed temperature	350 (K)
C_p	Specific heat capacity	1.0 (J·g^{-1} K^{-1})
ΔH	Heat of reaction	-1.0×10^5 (J·mol^{-1})
k_0	Arrhenius preexponential constant	5.33685×10^7 (min^{-1})
E/R	Activation energy/gas law constant	6000 (K)
UA	Heat transfer term	5000 (J·min^{-1} K^{-1})

where the description and value of each variable are given in Table 2. Two state variables of the model are the reactant concentration C_A and the reactor temperature T. The control objective is to control the reactant concentration C_A, through the manipulation of the coolant temperature T_c. Note that the reactor temperature T is not controlled for this simulation. Therefore, the output variable and manipulated variable are given by $y(t) = C_A$ and $u(t) = T_c$, respectively. For practicability, the coolant temperature T_c is constrained to the ranges [273.0, 373.0].

The above model is used to generate a series of input-output time-series data. The sampling time of the process measurements is set to 0.1 min. From Figure 8, the stop criterion ends at $L^{(2,1)} = 2.333$, indicating that the best order of the system is (2,1); that is, $n_a = 2$ and $n_b = 1$. The number of neurons in nonlinear static block $p = 4$. According to the convergence analysis, $\beta_{\max} = 2.856$; therefore, the learning rate can be chosen as $\eta = 0.15$, and the initial parameters in (21)–(23) are the same as Example 1. The nonlinear part of HTNN is sensitive to the data between −1 and 1; however,

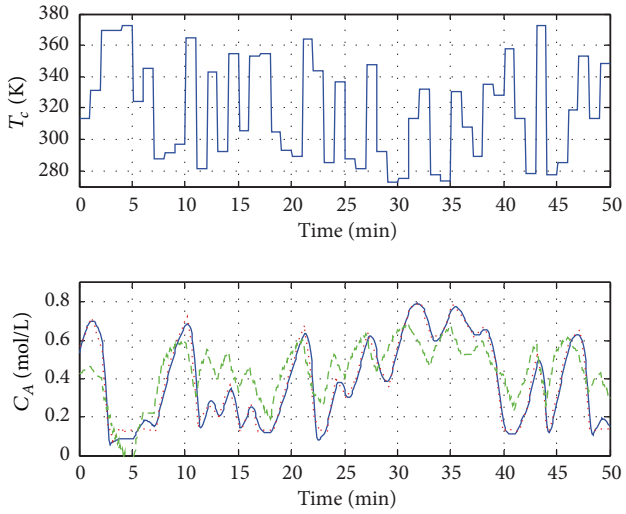

FIGURE 9: Training results, outputs of the plant (solid curve) and HTNN (dotted curve) and multilayer neural network (dash curve).

the input values (T_c) are within [273.0, 373.0]. Therefore, it is necessary to normalize the input signals as follows:

$$u(t) = \frac{T_c(t) - 273.0}{373.0 - 273.0} - \frac{373.0 - T_c(t)}{373.0 - 273.0}. \quad (34)$$

The normalized data is then used to train the HTNN; the training results are shown in Figure 9. The models were then tested on a set of data produced from an input with random amplitude; a common multilayer neural network with 5 hidden neurons [25] was also developed for comparison. As shown in Figure 9, the HTNN gives a good fit to this data.

6. Conclusions

In this paper, by formulated Hammerstein model, a Hammerstein-type neural network was developed for identifying nonlinear SISO systems, where the weights are corresponding with the parameters of Hammerstein model. To determine the model order and the parameters of the Hammerstein model, the Lipschitz quotients and the backpropagation training algorithm were used to determine the model order and update the weights in the network, respectively, and the stability was also analyzed. The HTNN was tested on several nonlinear systems to demonstrate the identification performances.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to acknowledge the funding received from the National Natural Science Foundation of China (no. 61005068), Open Foundation of State Key Laboratory of Robotics of China (no. 2013O09), Specialized Research Fund for the Doctoral Program of Higher Education of China (no. 20124101120001), and Key Project for Science and Technology of the Education Department of Henan Province (no. 14A413009) to conduct this research investigation.

References

[1] M. M. Arefi, A. Montazeri, J. Poshtan, and M. R. Jahed-Motlagh, "Wiener-neural identification and predictive control of a more realistic plug-flow tubular reactor," *Chemical Engineering Journal*, vol. 138, no. 1–3, pp. 274–282, 2008.

[2] V. Arto, P. Hannu, and A. Halme, "Modeling of chromatographic separation process with Wiener-MLP representation," *Journal of Process Control*, vol. 11, no. 5, pp. 443–458, 2001.

[3] S. J. Norquay, A. Palazoglu, and J. A. Romagnoli, "Model predictive control based on Wiener models," *Chemical Engineering Science*, vol. 53, no. 1, pp. 75–84, 1998.

[4] A. Janczak, *Identification of Nonlinear Systems Using Neural Networks and Polynomial Models: A Block-Oriented Approach*, Springer, New York, NY, USA, 2004.

[5] M. Ławryńczuk, "Computationally efficient nonlinear predictive control based on neural Wiener models," *Neurocomputing*, vol. 74, no. 1–3, pp. 401–417, 2010.

[6] G. Chen, Y. Chen, and H. Ogmen, "Identifying chaotic systems via a Wiener-type cascade model," *IEEE Control Systems Magazine*, vol. 17, no. 5, pp. 29–36, 1997.

[7] S. Tötterman and H. T. Toivonen, "Support vector method for identification of Wiener models," *Journal of Process Control*, vol. 19, no. 7, pp. 1174–1181, 2009.

[8] P. Saha, S. H. Krishnan, V. S. R. Rao, and S. C. Patwardhan, "Modeling and predictive control of MIMO nonlinear systems using Wiener-Laguerre models," *Chemical Engineering Communications*, vol. 191, no. 8, pp. 1083–1119, 2004.

[9] H. Al-Duwaish, M. N. Karim, and V. Chandrasekar, "Use of multilayer feedforward neural networks in identification and control of Wiener model," *IEE Proceedings-Control Theory and Applications*, vol. 143, pp. 255–258, 1996.

[10] D. Wu, S. Huang, W. Zhao, and J. Xin, "Infrared thermometer sensor dynamic error compensation using Hammerstein neural network," *Sensors and Actuators A: Physical*, vol. 149, no. 1, pp. 152–158, 2009.

[11] J. Peng and R. Dubay, "Wiener-type neural network for nonlinear system identification," in *Proceedings of the 22nd IASTED International Symposia on Modelling and Simulation*, Calgary, Alberta, Canada, July 2011.

[12] M. Boutayeb and M. Darouach, "Recursive identification method for MISO Wiener-Hammerstein model," *IEEE Transactions on Automatic Control*, vol. 40, no. 2, pp. 287–291, 1995.

[13] A. Hagenblad, L. Ljung, and A. Wills, "Maximum likelihood identification of Wiener models," *Automatica*, vol. 44, no. 11, pp. 2697–2705, 2008.

[14] S. A. Billings and S. Y. Fakhouri, "Identification of systems containing linear dynamic and static nonlinear elements," *Automatica*, vol. 18, no. 1, pp. 15–26, 1982.

[15] A. Kalafatis, N. Arifin, L. Wang, and W. R. Cluett, "A new approach to the identification of pH processes based on the Wiener model," *Chemical Engineering Science*, vol. 50, no. 23, pp. 3693–3701, 1995.

[16] X. He and H. Asada, "A new method for identifying orders of input-output models for nonlinear dynamic systems," in

Proceedings of the American Control Conference (ACC '93), pp. 2520–2523, San Francisco, Calif, USA, June 1993.

[17] T. Bengtsson and J. E. Cavanaugh, "An improved Akaike information criterion for state-space model selection," *Computational Statistics and Data Analysis*, vol. 50, no. 10, pp. 2635–2654, 2006.

[18] J.-S. Wang and Y.-P. Chen, "A fully automated recurrent neural network for unknown dynamic system identification and control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 6, pp. 1363–1372, 2006.

[19] J. Peng and R. Dubay, "Identification and adaptive neural network control of a DC motor system with dead-zone characteristics," *ISA Transactions*, vol. 50, no. 4, pp. 588–598, 2011.

[20] Y.-L. Hsu and J.-S. Wang, "A Wiener-type recurrent neural network and its control strategy for nonlinear dynamic applications," *Journal of Process Control*, vol. 19, no. 6, pp. 942–953, 2009.

[21] M. A. Gonzalez-Olvera and Y. Tang, "A new recurrent neurofuzzy network for identification of dynamic systems," *Fuzzy Sets and Systems*, vol. 158, no. 10, pp. 1023–1035, 2007.

[22] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 32, no. 2, pp. 176–190, 2002.

[23] J. Peng, R. Dubay, J. M. Hernandez, and M. Abu-Ayyad, "A wiener neural network-based identification and adaptive generalized predictive control for nonlinear SISO systems," *Industrial and Engineering Chemistry Research*, vol. 50, no. 12, pp. 7388–7397, 2011.

[24] M. A. Hussain and P. Y. Ho, "Adaptive sliding mode control with neural network based hybrid models," *Journal of Process Control*, vol. 14, no. 2, pp. 157–176, 2004.

[25] J. Peng, Y. Wang, W. Sun et al., "A neural network sliding mode controller with application to robotic manipulator," in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA '06)*, pp. 2101–2105, Dalian, China, June 2006.