Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 3219140, 19 pages
https://doi.org/10.1155/2018/3219140

Research Article

MTN Optimal Tracking Control of SISO Nonlinear Time-Varying Discrete-Time Systems without Mechanism Models

Jiao-Jun Zhang[1,2] **and Hong-Sen Yan**[1,2]

[1]*Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education, Southeast University, Nanjing, China*
[2]*School of Automation, Southeast University, Nanjing, China*

Correspondence should be addressed to Hong-Sen Yan; hsyan@seu.edu.cn

Received 3 October 2017; Revised 24 April 2018; Accepted 30 April 2018; Published 3 July 2018

Academic Editor: Javier Moreno-Valenzuela

Nonlinear time-varying systems without mechanism models are common in application. They cannot be controlled directly by the traditional control methods based on precise mathematical models. Intelligent control is unsuitable for real-time control due to its computation complexity. For that sake, a multidimensional Taylor network (MTN) based output tracking control scheme, which consists of two MTNs, one as an identifier and the other as a controller, is proposed for SISO nonlinear time-varying discrete-time systems with no mechanism models. A MTN identifier is constructed to build the offline model of the system, and a set of initial parameters for online learning of the identifier is obtained. Then, an ideal output signal is selected relative to the given reference signal. Based on the system identification model, Pontryagin minimum principle is introduced to obtain the numerical solution of the optimal control law for the system relative to the given ideal output signal, with the corresponding optimal output taken as the desired output signal. A MTN controller is generated automatically to fit the numerical solution of the optimal control law using the conjugate gradient (CG) method, and a set of initial parameters for online learning of the controller is obtained. An adaptive back propagation (BP) algorithm is developed to adjust the parameters of the identifier and controller in real time, and the convergence for the proposed learning algorithm is verified. Simulation results show that the proposed scheme is valid.

1. Introduction

Nonlinear time-varying systems without mechanism models exist in practical engineering applications widely. However, it is difficult to obtain the precise mathematical model of a system due to the limitation of the modeling theory, the influences of its internal structure and parameter variations, and the external environment disturbances. In addition, the state variables are not easy to be determined, and it is inconvenient for state feedback control to be realized physically due to the practical and economic limitations of the measuring equipment in engineering practices. Output feedback control [1–6], which is of great theoretic and realistic significance, is to probe into the problem for nonlinear time-varying systems without mechanism models.

Nonlinear autoregressive moving average with exogenous inputs (NARMAX) model describes an input-output relationship for a nonlinear dynamic system, by which the system output can be represented as a nonlinear functional expansion of its lagged inputs and outputs [7–9]. NARMAX has attracted considerable interest both in its theory and in applications [10–13], especially in the field of black-box nonlinear modeling. It is also referred to as time-varying NARMAX model in [14]. Neural network (NN) has the ability to approximate any continuous function with an arbitrary degree of accuracy over a compact set [15], and various kinds of NN have been used in system identification and control [7, 8, 16–24]. However, to satisfy the approximation requirement of a high-order uncertain system, both the numbers of the hidden layer neurons and the corresponding weight parameters needing online updating are large, which leads to the fact that the learning time tends to be unacceptably long and the real-time control is hardly realizable in practice. In addition, NN can only represent local dynamic characteristics when the state goes without too much change. However, the actual state change can be quite notable or even divergent. Thus NN cannot represent dynamic characteristics in the general sense. In fact, it cannot approximate to the unstable system as the artificial neurons suffer the limitations from the sign function, sigmoid function, and radial basis function, regardless of sample size or weight parameters. Furthermore, NN cannot approximate any continuous function to an arbitrarily degree of accuracy if the sign function or the sigmoid function is removed. From the point of view of the frequency characteristics of the input signals, a signal can be viewed as the superposition of value-high signal with the lowest frequency and value-low signals with high frequencies. Therefore, the value-low signals with high frequencies tend to be limited or restrained by the sign function, sigmoid function, or radial basis function of the artificial neuron, as a result of which the output of NN may fail to track the rapid change of its input. In terms of the neural network function approximation theorem, a three-layer neural network can approximate any nonlinear real-valued continuous function defined on a closed bounded subset. However, it cannot ensure that the nonlinear real-valued continuous function can be well approximated outside the closed bounded subset. Therefore, the general nonlinear dynamic system in the entire state space (i.e., not the bounded subset) is difficult to be approximated with an arbitrary degree of accuracy. As the polynomial function tends to be of infinity, the multidimensional Taylor network (MTN, whose idea was proposed by Hong-Sen Yan in 2010 and realization was done by Bo Zhou) is good at approximating or representing the general nonlinear dynamic system. It is suitable to be used as the identified dynamic model of the controlled plant, as it can represent polynomial dynamic system accurately, being simple with only states and inputs, and can be easily analyzed and solved for optimal control in terms of the minimum principle. However, NN only approximates the polynomial dynamic system instead of representing it. It is too complicated to be analyzed or solved for optimal control [25]. In addition, exponential function is contained in NN, which leads to the computational complexity and poor real-time control performance by a single chip microcomputer (SCM) and embedded system. That makes us resort to MTN, whereby only addition and multiplication are needed, its computation complexity being nearly that of the Taylor expansion of a single neuro in NN.

MTN, first presented in [26], can reflect the dynamic characteristics of the system without knowing the order or other prior knowledge of the system. It approximates any nonlinear function with an arbitrary high accuracy, thus widely being applied in the study of time series prediction problems successfully [27–32]. The idea of MTN optimal control was proposed by Hong-Sen Yan in 2010 [33]. The optimal adjustment control of SISO nonlinear time-invariant systems has been achieved by introducing control input into MTN [34]. Asymptotic tracking and dynamic regulation of SISO nonlinear system based on discrete multidimensional Taylor network are considered in [35]. However, the system without mechanism model is not considered, nor are the time-varying characteristics of the nonlinear discrete-time system. MTN relies on the polynomial network for identification and control of the nonlinear system [10] in which the system considered is constant and its learning algorithm is based on the gradient descent algorithm with constant learning factor, which leads to its slow learning speed and convergence to local minima.

Due to the uncertainties of the external environment and time-varying characteristics of a controlled plant [36, 37], the identifier and controller parameters need constant updating online in the process of its control, and the adjustments affect not only the control process but the robustness of the controller. Therefore, designing a desirable real-time self-tuning rule for the weight parameters of the identifier and controller is highly wanted. Backpropagation (BP) algorithm [38] is the most widely used learning algorithm in training multilayer neural network. However, it has such drawbacks as slow convergence speed and local optimal point. To raise the convergence speed, the improvement of BP algorithm [39–42] has been focused on, with certain desirable results achieved. However, in the improved learning algorithm, the learning rate and momentum factor are taken as constants in the interval (−1,1) randomly. There have emerged some other evolutionary algorithms developed to adjust the weight parameters for NN, such as genetic algorithm (GA) [43], particle swarm optimization (PSO) algorithm [44], the hybrid of them [45, 46], and the fuzzy logic approach [47], and so on. It has been shown that the coefficients should not remain fixed but should be changed adaptively throughout the entire training process so as to produce better training results, and it leads to the emergence of various schemes for adjusting the learning rate and momentum factor of BP algorithms adaptively [48, 49].

In this paper, a MTN-based output tracking real-time control scheme, which consists of two MTNs, one as the identifier and the other as the controller, is proposed for SISO nonlinear time-varying discrete-time systems without mechanism models. A MTN identifier is developed for offline modeling of the system, and a set of initial parameters for online learning of the identifier is obtained. An ideal output signal is then selected for the given reference signal. Pontryagin minimum principle is employed to obtain the numerical solution of the optimal control law of the system relative to the ideal output signal, with the corresponding optimal output called the desired output signal. A MTN controller is generated automatically to fit the numerical solution of the optimal control law via the CG method, and a set of initial parameters for online learning of the controller is obtained. Based on the above, a novel adaptive BP algorithm for adjusting both the learning rate and the momentum factor in real time is designed to further enhance the learning speed of the identifier and controller. Finally, the convergence of the novel adaptive BP algorithm is analyzed. Simulation results show that the proposed scheme is valid.

This paper is arranged as follows: in Section 2, problem description; in Section 3, identifier design; in Section 4,

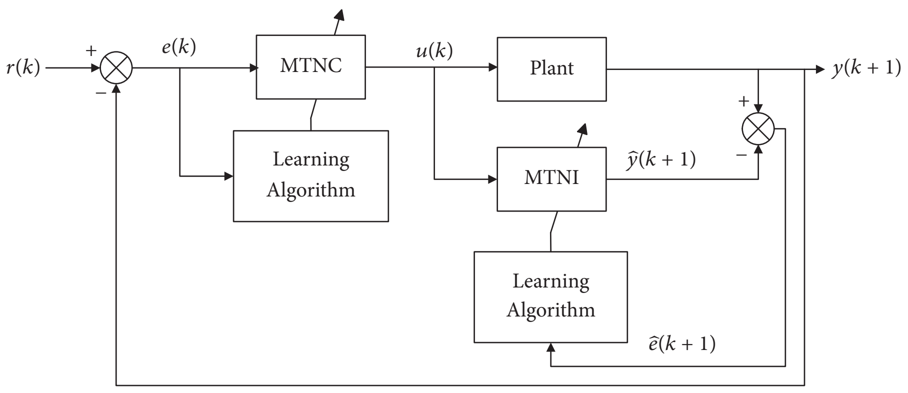

FIGURE 1: Block diagram of control system (1).

automatic generation of the controller; in Section 5, selection of initial value of the online controller parameters; in Section 6, controller parameters self-tuning; The algorithm steps for MTN optimal control scheme are summarized in Section 7; simulation study is mentioned in Section 8; conclusion is in Section 9.

2. Problem Statement

Consider the following unknown SISO nonlinear time-varying discrete-time system described by the input-output difference equation:

$$\begin{aligned} y(k+1) = f\big(&y(k), y(k-1), \dots, y(k-n_y), u(k), \\ &u(k-1), \dots, u(k-n_u), k\big), \end{aligned} \tag{1}$$

where $f(\cdot)$ is an unknown nonlinear scale function, $y(k)$ and $u(k)$ are the output and input of the system, and n_y and n_u are the corresponding maximum delays.

The goals of the present study are as follows: (a) to design an offline identifier to build the system model based on the input-output data pairs $\{u(k), y(k+1)\}$; (b) to design such real-time controller that allows the output $y(k)$ of the system (1) to track the given reference signal $r(k)$ as closely as possible.

The block diagram of control system (1) is shown in Figure 1. For clarity, the multidimensional Taylor network identifier and controller are abbreviated as MTNI and MTNC, as shown in Figure 1.

3. System Identification

It is known from [26] that MTN provides a good nonlinear function approximation approach, and $f(\cdot)$ in the system (1) can be approximated with arbitrary precision by MTN, using an appropriate learning algorithm. Let $\hat{f}(\cdot)$ be the mapping relationship, and we obtain the MTN model (MTNI) of the system (1) as follows:

$$\begin{aligned} \hat{y}(k+1) = \hat{f}\big(&\hat{k}_1 y(k), \hat{k}_2 y(k-1), \dots, \hat{k}_{n_y+1} y(k-n_y), \\ &\hat{l}_1 u(k), \hat{l}_2 u(k-1), \dots, \hat{l}_{n_u+1} u(k-n_u), \widehat{\mathbf{w}}(k)\big) \end{aligned} \tag{2}$$

where $\hat{y}(k+1)$ is the output of MTNI, $\widehat{\mathbf{w}}(k)$ is the weight coefficient vector of MTNI, and $\hat{l}_i$ and $\hat{k}_j$ are positive constants, $i = 1, 2, \dots, n_u+1$, $j = 1, 2, \dots, n_y+1$.

For convenience and without loss of generality, set $\hat{t} = n_y + n_u + 2$ and we get

$$\begin{aligned} \hat{\mathbf{z}}(k) = \big[&\hat{z}_1(k), \dots, \hat{z}_{n_y+1}(k), \hat{z}_{n_y+2}(k), \dots, \hat{z}_{\hat{t}-1}(k), \\ &\hat{z}_{\hat{t}}(k)\big]^{\mathrm{T}} = \big[\hat{k}_1 y(k), \dots, \hat{k}_{n_y+1} y(k-n_y), \hat{l}_2 u(k-1), \dots, \\ &\hat{l}_{n_u+1} u(k-n_u), \hat{l}_1 u(k)\big]^{\mathrm{T}}. \end{aligned} \tag{3}$$

Setting the weight coefficient vector of MTNI as $\widehat{\mathbf{w}}(k) = [\widehat{w}_1(k), \dots, \widehat{w}_{\widehat{N}(\hat{t},\widehat{m})}(k)]^{\mathrm{T}}$ allows us to rewrite the identification model (2) as

$$\hat{y}(k+1) = \sum_{\hat{p}=1}^{\widehat{N}(\hat{t},\widehat{m})} \widehat{w}_{\hat{p}}(k) \prod_{\hat{q}=1}^{\hat{t}} \hat{z}_{\hat{q}}^{\hat{\lambda}(\hat{p},\hat{q})}(k), \tag{4}$$

where $\widehat{N}(\hat{t},\widehat{m})$ represents the total number of product items of the $\hat{t}$-ary function $\hat{f}(\cdot)$ expanded into the approximate polynomial with $\widehat{m}$ powers, $\widehat{w}_{\hat{p}}(k)$ is the weight coefficient of the $\hat{p}$-th product item in the formula, $\hat{\lambda}(\hat{p},\hat{q})$ denotes the power of the variable $\hat{z}_{\hat{q}}(k)$ in the $\hat{p}$-th product item, and $\sum_{\hat{q}=1}^{\hat{t}} \hat{\lambda}(\hat{p},\hat{q}) \le \widehat{m}$, where $\hat{p} = 1, 2, \dots, \widehat{N}(\hat{t},\widehat{m})$.

The diagram of MTNI is shown in Figure 2.

To calculate $\widehat{N}(\hat{t},\widehat{m})$ and $\hat{\lambda}(\hat{p},\hat{q})$, the product items in (4) are rearranged as illustrated in Figure 3, i.e., storing the product items of the expansion according to their powers, respectively. We use the symbol $(\hat{i},\hat{j})$ to denote the $\hat{i}$-th rectangle in which the product items with $\hat{j}$-th power are stored and store the product items with $\hat{j}$-th power which are got by adding 1 on the power of the $\hat{i}$-th element $\hat{z}_{\hat{i}}(k)$ from the $\hat{i}$-th rectangle to the $\hat{t}$-th rectangle with $\hat{j}-1$-th power

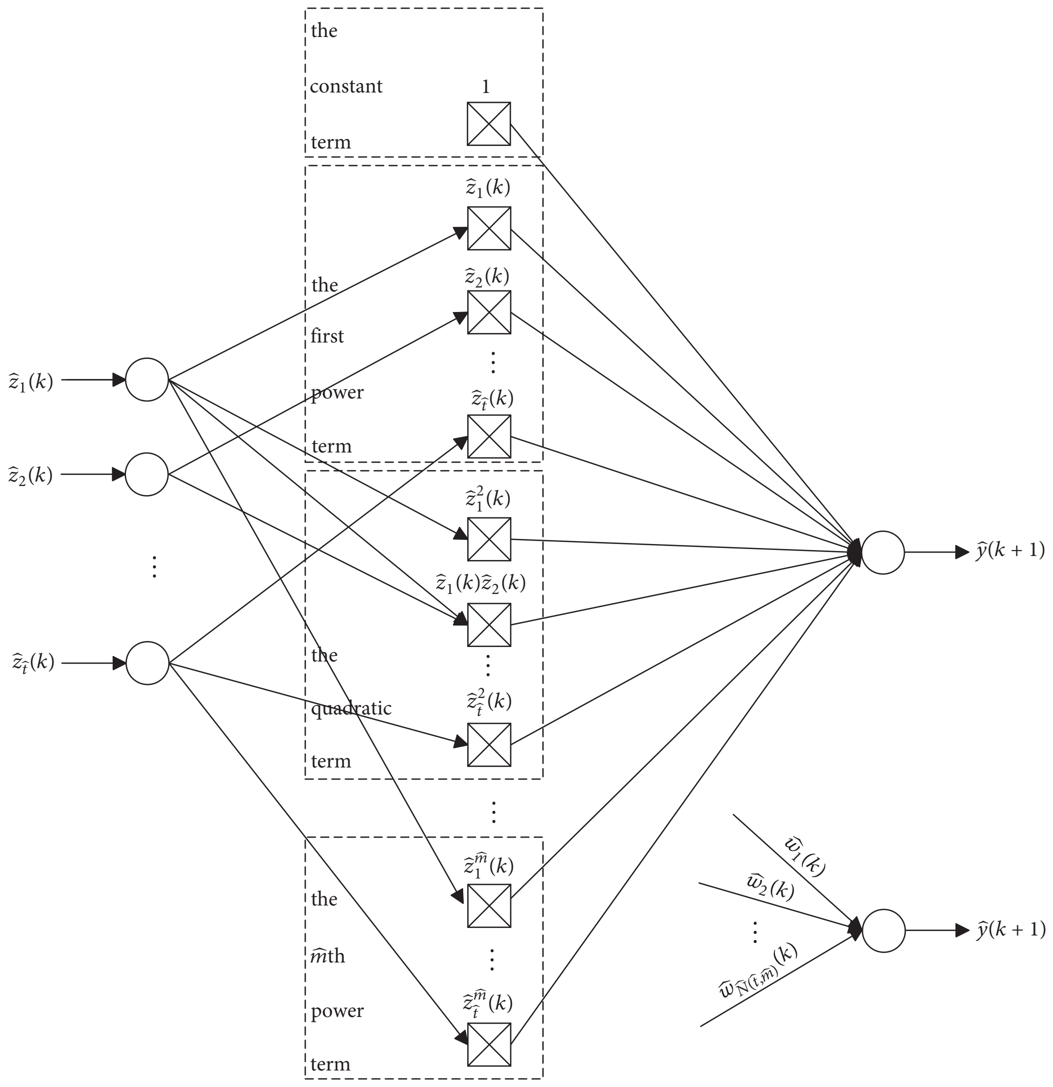

Figure 2: The diagram of MTNI.

into $(\hat{i},\hat{j})$, and so on, until storing the product items with $\hat{m}$-th power which are obtained by adding 1 on the power of the $\hat{t}$-th element $\hat{z}_{\hat{t}}(k)$ in $(\hat{t},\hat{m}-1)$ into $(\hat{t},\hat{m})$, where $\hat{i}=1,2,\dots,\hat{t}$ and $\hat{j}=2,3,\dots,\hat{m}$.

The calculation of $\widehat{N}(\hat{t},\hat{m})$ and $\hat{\lambda}(\hat{p},\hat{q})$ goes as follows.

Let $\widehat{P}(\hat{i},\hat{j})$ represent the number of product items in $(\hat{i},\hat{j})$, and we get

$$\widehat{N}(\hat{t},\hat{m})=\sum_{\hat{j}=1}^{\hat{m}}\sum_{\hat{i}=1}^{\hat{t}}\widehat{P}(\hat{i},\hat{j})+1 \tag{5}$$

where

$$\widehat{P}(\hat{i},\hat{j})=\begin{cases}\displaystyle\sum_{\hat{s}=\hat{i}}^{\hat{t}}\widehat{P}(\hat{s},\hat{j}-1), & \hat{j}=2,3,\dots,\hat{m},\\ 1, & \hat{j}=1.\end{cases} \tag{6}$$

Suppose that, in (4), from the 2-th item, the $\hat{p}$-th product item corresponds to the $\hat{r}$-th product item in $(\hat{i},\hat{j})$ of Figure 3. For clarity, the power of the element $\hat{z}_{\hat{q}}(k)$ is termed as $\hat{\lambda}_{\hat{i},\hat{j}}(\hat{r},\hat{q})$, and $\widehat{Q}_{\hat{j}}(\hat{a},\hat{b})$ represent the number of product items with the $\hat{j}$-th power from the $\hat{a}$-th to the $\hat{b}$-th rectangle. From Figure 3, it is known that

$$\begin{aligned}&\hat{\lambda}_{\hat{i},\hat{j}}(\hat{r},\hat{q})\\ &=\begin{cases}\hat{\lambda}_{\hat{i}+\hat{d},\hat{j}-1}\left(\hat{r}-\widehat{Q}_{\hat{j}-1}\left(\hat{i},\hat{i}+\hat{d}-1\right),\hat{q}\right)+1, & \hat{q}=\hat{i},\\ \hat{\lambda}_{\hat{i}+\hat{d},\hat{j}-1}\left(\hat{r}-\widehat{Q}_{\hat{j}-1}\left(\hat{i},\hat{i}+\hat{d}-1\right),\hat{q}\right), & \hat{q}\neq\hat{i},\end{cases}\\ &\qquad\qquad\left(\hat{j}=2,3,\dots,\hat{m}\right)\end{aligned} \tag{7}$$

$$\widehat{Q}_{\hat{j}}(\hat{a},\hat{b})=\begin{cases}\displaystyle\sum_{\hat{s}=\hat{a}}^{\hat{b}}\widehat{P}(\hat{s},\hat{j}), & \hat{a}\le\hat{b},\\ 0, & \hat{a}>\hat{b},\end{cases}\quad\left(\hat{j}=2,3,\dots,\hat{m}\right).$$

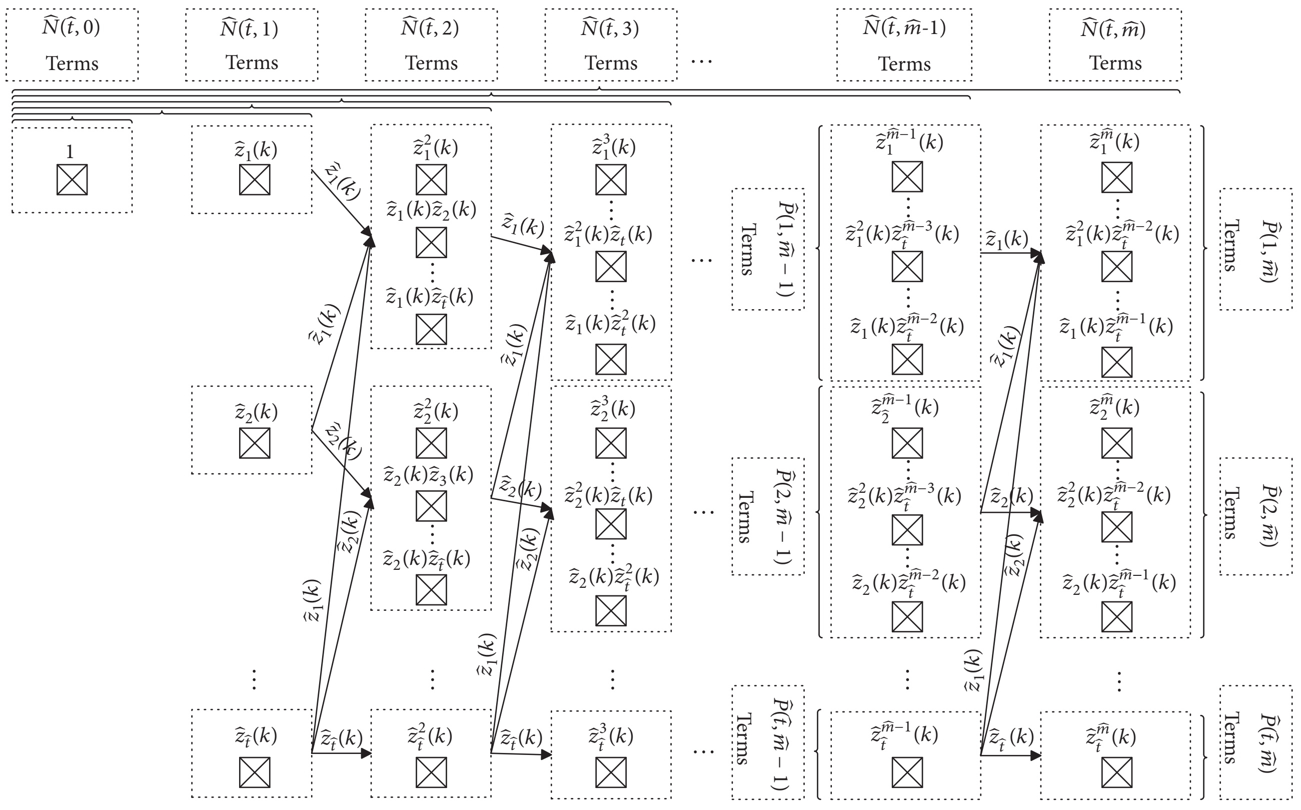
$\hat N(\hat t,0)$ Terms
$\hat N(\hat t,1)$ Terms
$\hat N(\hat t,2)$ Terms
$\hat N(\hat t,3)$ Terms
$\hat N(\hat t,\hat m$-1) Terms
$\hat N(\hat t,\hat m)$ Terms
1
$\hat z_1(k)$
$\hat z_2(k)$
$\hat z_{\hat t}(k)$
$\hat z_1^2(k)$
$\hat z_1(k)\hat z_2(k)$
$\hat z_1(k)\hat z_{\hat t}(k)$
$\hat z_2^2(k)$
$\hat z_2(k)\hat z_3(k)$
$\hat z_2(k)\hat z_{\hat t}(k)$
$\hat z_{\hat t}^2(k)$
$\hat z_1^3(k)$
$\hat z_1^2(k)\hat z_t(k)$
$\hat z_1(k)\hat z_t^2(k)$
$\hat z_2^3(k)$
$\hat z_2^2(k)\hat z_t(k)$
$\hat z_2(k)\hat z_t^2(k)$
$\hat z_{\hat t}^3(k)$
$\hat P(1,\hat m-1)$ Terms
$\hat P(2,\hat m-1)$ Terms
$\hat P(\hat t,\hat m-1)$ Terms
$\hat z_1^{\hat m-1}(k)$
$\hat z_1^2(k)\hat z_{\hat t}^{\hat m-3}(k)$
$\hat z_1(k)\hat z_{\hat t}^{\hat m-2}(k)$
$\hat z_2^{\hat m-1}(k)$
$\hat z_2^2(k)\hat z_{\hat t}^{\hat m-3}(k)$
$\hat z_2(k)\hat z_{\hat t}^{\hat m-2}(k)$
$\hat z_{\hat t}^{\hat m-1}(k)$
$\hat z_1^{\hat m}(k)$
$\hat z_1^2(k)\hat z_{\hat t}^{\hat m-2}(k)$
$\hat z_1(k)\hat z_{\hat t}^{\hat m-1}(k)$
$\hat z_2^{\hat m}(k)$
$\hat z_2^2(k)\hat z_{\hat t}^{\hat m-2}(k)$
$\hat z_2(k)\hat z_{\hat t}^{\hat m-1}(k)$
$\hat z_{\hat t}^{\hat m}(k)$
$\hat P(1,\hat m)$ Terms
$\hat P(2,\hat m)$ Terms
$\hat P(\hat t,\hat m)$ Terms

FIGURE 3: The new arrangement of product items for MTNI.

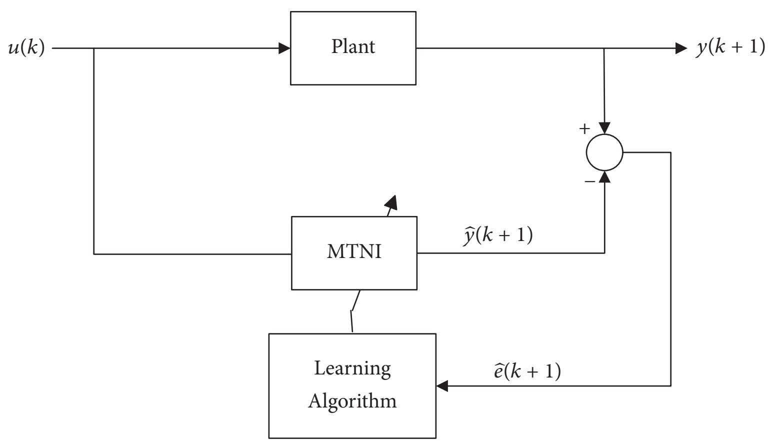

FIGURE 4: The diagram of system identification.

The initial values are set as follows:

$$\hat\lambda_{\hat i,\hat j}(\hat r,\hat q)=\begin{cases}1, & \hat q=\hat i,\\ 0, & \hat q\neq\hat i,\end{cases}\quad(\hat j=1),$$

$$\hat Q_{\hat j}(\hat a,\hat b)=\begin{cases}\hat b-\hat a+1, & \hat a\le\hat b,\\ 0, & \hat a>\hat b,\end{cases}\quad(\hat j=1), \tag{8}$$

where $\hat Q_{\hat j-1}(\hat i,\hat i+\hat d-1)<\hat r\le\hat Q_{\hat j-1}(\hat i,\hat i+\hat d)$, $\hat d=0,1,\dots,\hat t-\hat i$ and $\mathbf{y}=(y(1),y(2),\dots,y(\mathrm{N}))^{\mathrm{T}}$.

Based on the above, Figure 4 gives the diagram of system identification.

3.1. Offline System Identification. The identification error can be defined as

$$\hat e(k+1)=\hat y(k+1)-y(k+1), \tag{9}$$

where $k=0,1,\dots,\mathrm{N}-1$.

The corresponding mean square error is

$$\hat E=\frac{1}{\mathrm{N}}\sum_{k=0}^{\mathrm{N}-1}\hat e^2(k+1). \tag{10}$$

Substituting (4) and (9) into (10) yields

$$\widehat{E}=\frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{\widehat{p}=1}^{\widehat{N}(\widehat{t},\widehat{m})}\widehat{w}_{\widehat{p}}\prod_{\widehat{q}=1}^{\widehat{t}}\widehat{z}_{\widehat{q}}^{\widehat{\lambda}(\widehat{p},\widehat{q})}(k)-y(k+1)\right)^2. \quad (11)$$

Assume

$$\widehat{\boldsymbol{\alpha}}(k)=\left(\prod_{\widehat{q}=1}^{\widehat{t}}\widehat{z}_{\widehat{q}}^{\widehat{\lambda}(1,\widehat{q})}(k),\prod_{\widehat{q}=1}^{\widehat{t}}\widehat{z}_{\widehat{q}}^{\widehat{\lambda}(2,\widehat{q})}(k),\ldots,\prod_{\widehat{q}=1}^{\widehat{t}}\widehat{z}_{\widehat{q}}^{\widehat{\lambda}(\widehat{N}(\widehat{t},\widehat{m}),\widehat{q})}(k)\right)^{\mathrm{T}} \quad (12)$$

$$\widehat{\mathbf{A}}=(\widehat{\boldsymbol{\alpha}}(0),\widehat{\boldsymbol{\alpha}}(1),\ldots,\widehat{\boldsymbol{\alpha}}(N-1))$$

Equation (11) can be rewritten into

$$\begin{aligned}\widehat{E}&=\frac{1}{N}\sum_{k=0}^{N-1}\left(\widehat{\mathbf{w}}^{\mathrm{T}}\widehat{\boldsymbol{\alpha}}(k)-y(k+1)\right)^2\\&=\frac{1}{N}\widehat{\mathbf{w}}^{\mathrm{T}}\sum_{k=0}^{N-1}\left(\widehat{\boldsymbol{\alpha}}(k)\widehat{\boldsymbol{\alpha}}^{\mathrm{T}}(k)\right)\widehat{\mathbf{w}}\\&\quad-\frac{2}{N}\left(\sum_{k=0}^{N-1}y(k+1)\widehat{\boldsymbol{\alpha}}^{\mathrm{T}}(k)\right)\widehat{\mathbf{w}}\\&\quad+\frac{1}{N}\sum_{k=0}^{N-1}y^2(k+1).\end{aligned} \quad (13)$$

Calculate the partial derivative of $\widehat{E}$ with respect to the weight coefficient vector $\widehat{\mathbf{w}}$, i.e.,

$$\begin{aligned}\frac{\partial\widehat{E}}{\partial\widehat{\mathbf{w}}}&=\frac{2}{N}\sum_{k=0}^{N-1}\left(\widehat{\boldsymbol{\alpha}}(k)\widehat{\boldsymbol{\alpha}}^{\mathrm{T}}(k)\right)\widehat{\mathbf{w}}\\&\quad-\frac{2}{N}\sum_{k=0}^{N-1}y(k+1)\widehat{\boldsymbol{\alpha}}(k).\end{aligned} \quad (14)$$

Setting $\widehat{\mathbf{g}}=\partial\widehat{E}/\partial\widehat{\mathbf{w}}$, $\widehat{\mathbf{Q}}=(2/N)\sum_{k=0}^{N-1}(\widehat{\boldsymbol{\alpha}}(k)\widehat{\boldsymbol{\alpha}}^{\mathrm{T}}(k))$ and $\widehat{\mathbf{b}}=-(2/N)\sum_{k=0}^{N-1}y(k+1)\widehat{\boldsymbol{\alpha}}(k)$ gives

$$\widehat{\mathbf{g}}=\widehat{\mathbf{Q}}\widehat{\mathbf{w}}+\widehat{\mathbf{b}}. \quad (15)$$

Letting $\mathbf{y}=(y(1),y(2),\ldots,y(N))^{\mathrm{T}}$, the vector form of $\widehat{\mathbf{Q}}$, $\widehat{\mathbf{b}}$ and $\widehat{\mathbf{g}}$ can be rewritten as

$$\begin{aligned}\widehat{\mathbf{Q}}&=\frac{2}{N}\widehat{\mathbf{A}}\widehat{\mathbf{A}}^{\mathrm{T}},\\\widehat{\mathbf{b}}&=-\frac{2}{N}\widehat{\mathbf{A}}\mathbf{y},\\\widehat{\mathbf{g}}&=\frac{2}{N}\widehat{\mathbf{A}}\left(\widehat{\mathbf{A}}^{\mathrm{T}}\widehat{\mathbf{w}}-\mathbf{y}\right).\end{aligned} \quad (16)$$

To obtain a precise model of system (1), the weight coefficient vector $\widehat{\mathbf{w}}$ should be updated over and over again by observation of the input-output data pairs $\{u(k),y(k+1)\}$. A number of classical weight update laws have been proposed in the literatures, such as least squares algorithms, various gradient-type algorithms [50, 51], least-mean-square (LMS) algorithm [52], etc. The gradient method is commonly adopted for parameter adjustment; that is, $\widehat{\mathbf{w}}$ can be updated once in the negative gradient direction after each offline learning. Let $\widehat{\mathbf{w}}_{\widehat{\tau}}$ represent the value of $\widehat{\mathbf{w}}$ after the $\widehat{\tau}$ th training, and we obtain $\widehat{\mathbf{w}}_{\widehat{\tau}+1}=\widehat{\mathbf{w}}_{\widehat{\tau}}-\widehat{\mu}_{\widehat{\tau}}\widehat{\mathbf{g}}_{\widehat{\tau}}$, where $\widehat{\mu}_{\widehat{\tau}}=\widehat{\mathbf{g}}_{\widehat{\tau}}^{\mathrm{T}}\widehat{\mathbf{g}}_{\widehat{\tau}}/\widehat{\mathbf{g}}_{\widehat{\tau}}^{\mathrm{T}}\widehat{\mathbf{Q}}\widehat{\mathbf{g}}_{\widehat{\tau}}$. However, as the gradient path to the minimum point is zigzag, the search direction remains vertical to the last. Fortunately, the problem can be solved effectively by employing the CG method, whereby the weight can be updated as follows:

$$\widehat{\mathbf{w}}_{\widehat{\tau}+1}=\widehat{\mathbf{w}}_{\widehat{\tau}}+\widehat{\mu}_{\widehat{\tau}}\widehat{\mathbf{p}}_{\widehat{\tau}}. \quad (17)$$

where $\widehat{\mathbf{p}}_{\widehat{\tau}}=-\widehat{\mathbf{g}}_{\widehat{\tau}}+\widehat{a}_{\widehat{\tau}-1}\widehat{\mathbf{p}}_{\widehat{\tau}-1}$, $\widehat{\mu}_{\widehat{\tau}}=\widehat{\mathbf{g}}_{\widehat{\tau}}^{\mathrm{T}}\widehat{\mathbf{g}}_{\widehat{\tau}}/\widehat{\mathbf{p}}_{\widehat{\tau}}^{\mathrm{T}}\widehat{\mathbf{Q}}\widehat{\mathbf{p}}_{\widehat{\tau}}$, $\widehat{a}_{\widehat{\tau}-1}=\widehat{\mathbf{g}}_{\widehat{\tau}}^{\mathrm{T}}\widehat{\mathbf{Q}}\widehat{\mathbf{p}}_{\widehat{\tau}-1}/\widehat{\mathbf{p}}_{\widehat{\tau}-1}^{\mathrm{T}}\widehat{\mathbf{Q}}\widehat{\mathbf{p}}_{\widehat{\tau}-1}$, and the initial value is $\widehat{\mathbf{p}}_1=-\widehat{\mathbf{g}}_1$.

The block diagram of offline learning of MTNI (4) is shown in Figure 5.

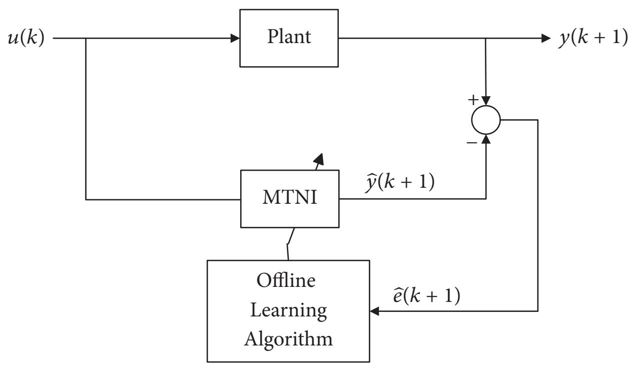

Figure 5: Block diagram of offline learning of MTNI.

3.2. Real-Time Learning for the Weights of MTNI. For an unknown system, system identification is the mathematical modeling process by observation of the input-output data pairs. Nonlinear time-varying system identification based on MTNI is to take the connection weight coefficients of MTNI as time-varying parameters to be estimated and trained online by suitable learning algorithm, with the same outputs of the plant and the model for the same set of inputs. The weight coefficients need to be adjusted online for desirable real-time identification effect.

Set the performance function for MTNI as

$$\widehat{E}(\widehat{\mathbf{w}}(k))=\frac{1}{2}\widehat{e}^2(\widehat{\mathbf{w}}(k)), \quad (18)$$

where $\widehat{e}(\widehat{\mathbf{w}}(k))=\widehat{e}(k+1)$, $\widehat{e}(k+1)=y(k+1)-\widehat{y}(k+1)$ represents the identification error at time $k+1$, and $k=0,1,\ldots$.

To obtain a better identification effect for the unknown nonlinear time-varying system, the weight coefficients of MTNI should be adjusted adaptively throughout the entire training process. A novel adaptive BP algorithm for adjusting both the learning rate and momentum factor adaptively [53–55] is proposed, and the weight parameters are updated, i.e.,

$$\widehat{\mathbf{w}}(k+1) = \widehat{\mathbf{w}}(k) + \Delta\widehat{\mathbf{w}}(k), \tag{19}$$

where

$$\Delta\widehat{\mathbf{w}}(k) = -\widehat{\eta}(k)\nabla\widehat{E}(\widehat{\mathbf{w}}(k)) + \widehat{\alpha}(k)\Delta\widehat{\mathbf{w}}(k-1), \tag{20}$$

$$\widehat{\eta}(k) = \min\left\{\widehat{\eta}(k-1)\left(1+\widehat{\eta}_0\cos\widehat{\theta}(k)\right), \widehat{\eta}_{\max}\right\}, \tag{21}$$

$$\widehat{\alpha}(k) = \widehat{\beta}(k)\widehat{\eta}^2(k), \tag{22}$$

$$\widehat{\beta}(k) = \begin{cases} \widehat{\beta}_0 \dfrac{-\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k-1)}{\|\Delta\widehat{\mathbf{w}}(k-1)\|^2}, & \text{if } \nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k-1) < 0, \\ 0, & \text{else,} \end{cases} \tag{23}$$

where $\nabla\widehat{E}(\widehat{\mathbf{w}}(k))$ is the partial derivative of $\widehat{E}(\widehat{\mathbf{w}}(k))$ with respect to $\widehat{\mathbf{w}}(k)$; $\widehat{\eta}_0, \widehat{\eta}_{\max}, \widehat{\beta}_0$ are constants, and $0 \le \widehat{\eta}_0 \le 1$, $0 < \widehat{\eta}_{\max} < 1$, $0 \le \widehat{\beta}_0 < 1$; $\widehat{\theta}(k)$ is the angle between the current gradient $-\nabla\widehat{E}(\widehat{\mathbf{w}}(k))$ and previous update $\Delta\widehat{\mathbf{w}}(k-1)$, given by $\cos\widehat{\theta}(k) = -\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k-1)/\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\|\|\Delta\widehat{\mathbf{w}}(k-1)\|$.

Theorem 1. *For any given set of weight coefficient vector $\widehat{\mathbf{w}}(0)$, if $\widehat{\mathbf{w}}(k)$ is generated by the learning rules from (19) to (23), there exists $\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k) \le 0$.*

Proof. For the first case, i.e., $\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k-1) < 0$, we have

$$\begin{aligned}\Delta\widehat{\mathbf{w}}(k) &= -\widehat{\eta}(k)\nabla\widehat{E}(\widehat{\mathbf{w}}(k)) + \widehat{\alpha}(k)\Delta\widehat{\mathbf{w}}(k-1) \\ &= \widehat{\eta}(k)\Bigg(-\nabla\widehat{E}(\widehat{\mathbf{w}}(k)) + \widehat{\beta}_0\widehat{\eta}(k) \\ &\quad\cdot \frac{-\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k-1)}{\|\Delta\widehat{\mathbf{w}}(k-1)\|^2}\Delta\widehat{\mathbf{w}}(k-1)\Bigg) = \widehat{\eta}(k) \\ &\quad\cdot\Bigg(-\nabla\widehat{E}(\widehat{\mathbf{w}}(k)) + \widehat{\beta}_0\widehat{\eta}(k) \\ &\quad\cdot\frac{\left\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\right\|\|\Delta\widehat{\mathbf{w}}(k-1)\|\cos\widehat{\theta}(k)}{\|\Delta\widehat{\mathbf{w}}(k-1)\|^2}\Delta\widehat{\mathbf{w}}(k-1)\Bigg) \\ &= \widehat{\eta}(k)\Bigg(-\nabla\widehat{E}(\widehat{\mathbf{w}}(k)) + \widehat{\beta}_0\widehat{\eta}(k)\cos\widehat{\theta}(k) \\ &\quad\cdot\frac{\left\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\right\|}{\|\Delta\widehat{\mathbf{w}}(k-1)\|}\Delta\widehat{\mathbf{w}}(k-1)\Bigg),\end{aligned} \tag{24}$$

$$\begin{aligned}\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k) &= \nabla\widehat{E}^{\mathrm{T}}(\widehat{\mathbf{w}}(k))\Delta\widehat{\mathbf{w}}(k) = \widehat{\eta}(k) \\ &\quad\cdot\Bigg(-\left\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\right\|^2 + \widehat{\beta}_0\widehat{\eta}(k)\cos\widehat{\theta}(k) \\ &\quad\cdot\frac{\left\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\right\|}{\|\Delta\widehat{\mathbf{w}}(k-1)\|}\nabla\widehat{E}^{\mathrm{T}}(\widehat{\mathbf{w}}(k))\Delta\widehat{\mathbf{w}}(k-1)\Bigg) = -\widehat{\eta}(k) \\ &\quad\cdot\left\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\right\|^2\left(1+\widehat{\beta}_0\widehat{\eta}(k)\cos^2\widehat{\theta}(k)\right)\end{aligned} \tag{25}$$

As a matter of fact, there exist $\widehat{\eta}(k) \ge 0$ and $1 + \widehat{\beta}_0\widehat{\eta}(k)\cos^2\widehat{\theta}_k > 0$ when $0 \le \widehat{\beta}_0 < 1$ and $0 \le \widehat{\eta}_0 \le 1$, as a result of which $\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k) \le 0$ holds.

For the second case, i.e., $\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k-1) \ge 0$, we get $\widehat{\beta}(k) = 0$, $\widehat{\alpha}(k) = 0$ and $\Delta\widehat{\mathbf{w}}(k) = -\widehat{\eta}(k)\nabla\widehat{E}(\widehat{\mathbf{w}}(k))$, thus,

$$\begin{aligned}\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k) &= \nabla\widehat{E}^{\mathrm{T}}(\widehat{\mathbf{w}}(k))\Delta\widehat{\mathbf{w}}(k) \\ &= -\widehat{\eta}(k)\left\|\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\right\|^2 \le 0.\end{aligned} \tag{26}$$

As revealed by the above two cases, $\nabla\widehat{E}(\widehat{\mathbf{w}}(k))\cdot\Delta\widehat{\mathbf{w}}(k) \le 0$ holds.

That completes the proof of Theorem 1. □

4. Controller

The control objective is to find a control input $u(k)$ that enables the system output to track in real time the given reference signal $r(k)$ as closely as possible in real time. In this section, we consider the controller MTNC generated automatically as follows:

$$\begin{aligned}u(k) = h\big(&k_1u(k-1),\ldots,k_{\overline{n}_u}u(k-\overline{n}_u),l_1e(k-1),\ldots, \\ &l_{n_e}e(k-n_e),k\big),\end{aligned} \tag{27}$$

where $u(k)$ is the output of MTNC, i.e., the input of the system (1), $e(k) = r(k) - y(k)$ is the tracking error at time k, $\overline{n}_u$ and n_e are the maximum delays of the output and input of MTNC, and l_i and k_j are positive constants, $i = 1,2,\ldots,n_e$, $j = 1,2,\ldots,\overline{n}_u$.

For convenience, without loss of generality, denote $t = \overline{n}_u + n_e$, and we have

$$\begin{aligned}\mathbf{z}(k) &= \left[z_1(k),\ldots,z_{\overline{n}_u}(k),z_{\overline{n}_u+1}(k),\ldots,z_t(k)\right]^{\mathrm{T}} \\ &= \left[k_1u(k-1),\ldots,k_{\overline{n}_u}u(k-\overline{n}_u),l_1e(k-1),\ldots,\right. \\ &\quad\left. l_{n_e}e(k-n_e)\right]^{\mathrm{T}}.\end{aligned} \tag{28}$$

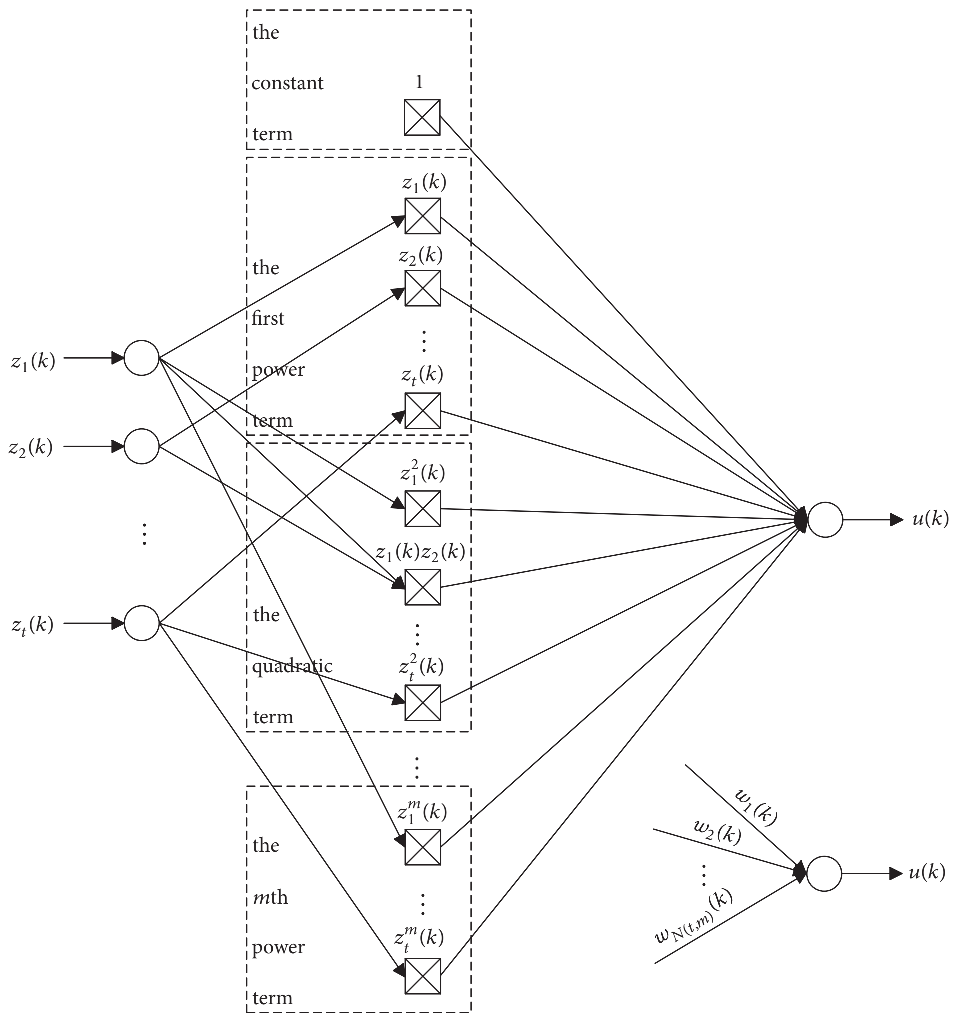

FIGURE 6: The diagram of MTNC.

Known from [26], there exists a group of weight coefficient vectors $\mathbf{w}(k) = [w_1(k), \ldots, w_{N(t,m)}(k)]^{\mathrm{T}}$, and thus, the input $u(k)$ can be rewritten as

$$u(k) = \sum_{p=1}^{N(t,m)} w_p(k) \prod_{q=1}^{t} z_q^{\lambda(p,q)}(k), \tag{29}$$

where $N(t,m)$ represents the total number of product items for the t-ary function $h(\cdot)$ expanded into the approximate polynomial with m powers, $w_p(k)$ denotes the weight coefficient of the p-th product item, $\lambda(p,q)$ is the power of the variable $z_q(k)$ in the p-th product item, and $\sum_{q=1}^{t} \lambda(p,q) \le m$, where $p = 1, 2, \ldots, N(t,m)$.

The diagram of MTNC is shown in Figure 6.

To calculate $N(t,m)$ and $\lambda(p,q)$, the product items in (29) are rearranged as illustrated in Figure 7, i.e., storing the product items of the expansion according to their powers, respectively. We use the symbol (i,j) to denote the i-th rectangle in which the product items with j-th power are stored and store the product items with j-th power which are got by adding 1 on the power of the i-th element $z_i(k)$ from the i-th rectangle to the t-th rectangle with $j-1$-th power into (i,j), and so on, until storing the product items with m-th power which are obtained by adding 1 on the power of the t-th element $z_t(k)$ in $(t, m-1)$ into (t,m), where $i = 1, 2, \ldots, t$ and $j = 2, 3, \ldots, m$.

The calculation of $N(t,m)$ and $\lambda(p,q)$ goes as follows.

Let $P(i,j)$ represent the number of product items in (i,j), and we get

$$N(t,m) = \sum_{j=1}^{m} \sum_{i=1}^{t} P(i,j) + 1, \tag{30}$$

where

$$P(i,j) = \begin{cases} \sum_{s=i}^{t} P(s, j-1), & j = 2, 3, \ldots, m, \\ 1, & j = 1. \end{cases} \tag{31}$$

Suppose that, in (29), from the 2-th item, the p-th product item is according to the r-th product item in (i,j) of Figure 7. For clarity, the power of the element $z_q(k)$ is termed as $\lambda_{i,j}(r,q)$, and $Q_j(a,b)$ represents the number of product items

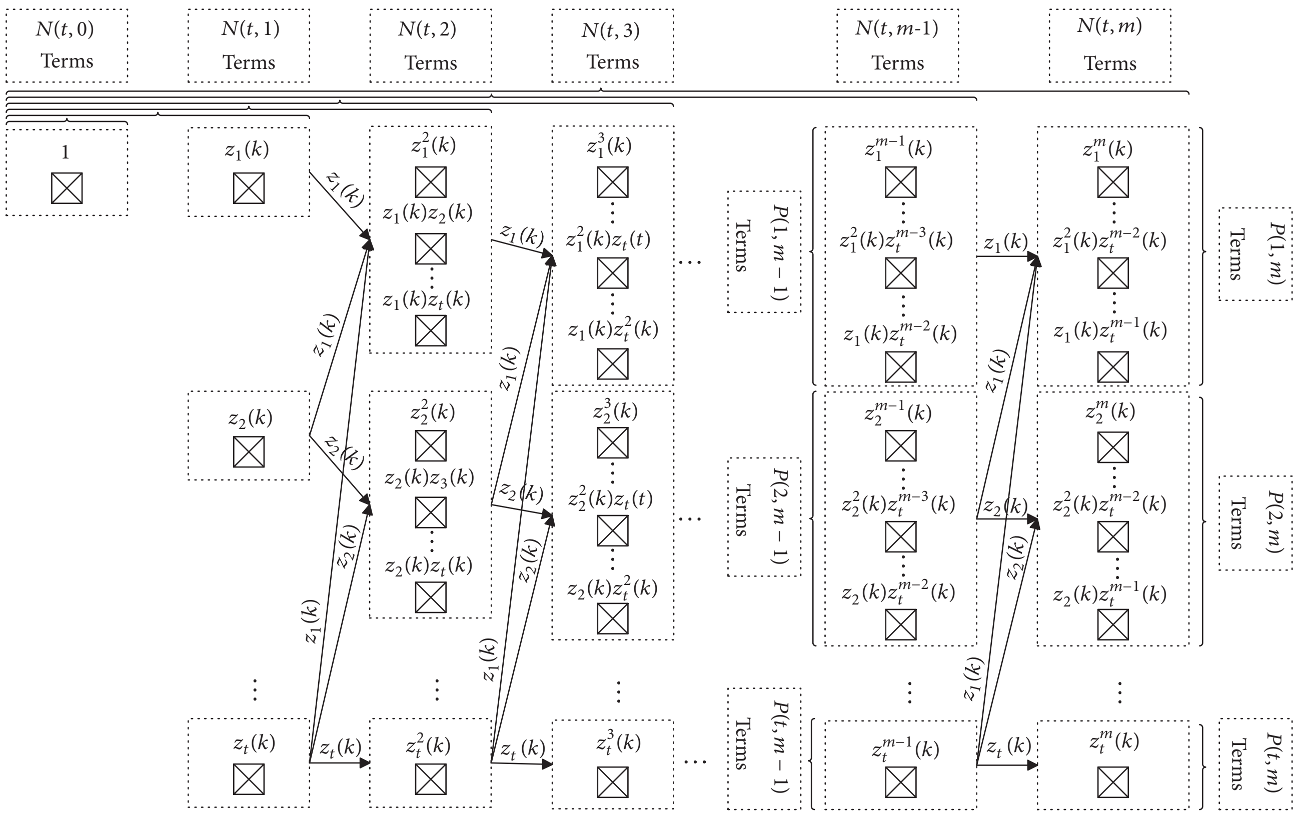

Figure 7: The new arrangement of the product items for MTNC.

with the j-th power from the a-th to the b-th rectangle. From Figure 7, it is known that

$$\lambda_{i,j}(r,q) = \begin{cases} \lambda_{i+d,j-1}\left(r - Q_{j-1}(i,i+d-1), q\right) + 1, & q = i, \\ \lambda_{i+d,j-1}\left(r - Q_{j-1}(i,i+d-1), q\right), & q \neq i, \end{cases} \quad (32)$$

$$Q_j(a,b) = \begin{cases} \sum_{s=a}^{b} P(s,j), & a \leq b, \\ 0, & a > b. \end{cases}$$

The initial values are set as follows:

$$\lambda_{i,j}(r,q) = \begin{cases} 1, & q = i, \\ 0, & q \neq i, \end{cases} \quad (j = 1),$$

$$Q_j(a,b) = \begin{cases} b - a + 1, & a \leq b, \\ 0, & a > b, \end{cases} \quad (j = 1), \quad (33)$$

where $Q_{j-1}(i, i+d-1) < r \leq Q_{j-1}(i, i+d)$, $d = 0, 1, \ldots, t-i$ and $i = 1, 2, \ldots, t$.

5. Initial Weight Values of MTNC

The convergence speed is influenced by the selection of the initial weight values of the controller. However, random choice of network parameters is the most common practice in network training. To enhance the convergence speed and avoid falling into local minimum, two steps are introduced here for selection of the initial weight values. The first is to transform the offline model (4) of the system into an extended state space description form through variable substitution, select a group of ideal output signal $y_{id}(k)$ relative to the given reference signal $r(k)$, and employ Pontryagin's minimum principle to obtain the numerical solution of the optimal control law $u^*(k)$ of the system relative to the ideal output signal $y_{id}(k)$, with the corresponding optimal output called desired output signal $y_{op}(k)$. In the second step, a set of parameter values is given randomly in the interval (-1, 1), and the CG method is applied for MTNC offline training to approximate the optimal control law $u^*(k)$. A set of weight values are then obtained as the initial values for online training MTNC, where $k = 0, 1, \ldots, N-1$. The specific steps go as follows.

5.1. Optimal Control Law. Based on the identification model (4) obtained offline for the system (1), $y(k)$ can be substituted by $\hat{y}(k)$. For convenience, set

$$\mathbf{x}(k) = \left[x_1(k), x_2(k), \ldots, x_{n_y+1}(k), x_{n_y+2}(k), \ldots, x_{\hat{t}-1}(k)\right]^T = \left[\hat{k}_1\hat{y}(k), \hat{k}_2\hat{y}(k-1), \ldots, \hat{k}_{n_y+1}\hat{y}(k-n_y), \hat{l}_2 u(k-1), \ldots, \hat{l}_{n_u+1} u(k-n_u)\right]^T, \quad (34)$$

then, we obtain the extended state space description form with the following variable substitution:

$$\begin{aligned} x_1(k+1) &= \hat{k}_1 \sum_{\hat{p}=1}^{\hat{N}(\hat{t},\hat{m})} \hat{\varpi}_{\hat{p}} \prod_{\hat{q}=1}^{\hat{t}-1} x_{\hat{q}}^{\hat{\lambda}(\hat{p},\hat{q})}(k)\, \hat{l}_1^{\hat{\lambda}(\hat{p},\hat{t})} u^{\hat{\lambda}(\hat{p},\hat{t})}(k) \\ x_j(k+1) &= \frac{\hat{k}_j}{\hat{k}_{j-1}} x_{j-1}(k), \quad \left(2 \le j \le n_y + 1\right) \\ x_{n_y+2}(k+1) &= \hat{l}_2 u(k) \\ x_j(k+1) &= \frac{\hat{l}_{j-n_y}}{\hat{l}_{j-1-n_y}} x_{j-1}(k), \quad \left(n_y + 3 \le j \le \hat{t} - 1\right) \end{aligned} \tag{35}$$

$$\hat{y}(k) = \frac{1}{\hat{k}_1} x_1(k). \tag{36}$$

Consider the following optimal control problem [56]:

$$\min_{u(k)} \tilde{E} = \frac{1}{2} \sum_{k=0}^{\mathrm{N}-1} \left(y_{id}(k) - \hat{y}(k)\right)^2, \tag{37}$$

where $\hat{y}(k)$ satisfies such the constraint conditions as (35) and (36).

Introduce the Hamiltonian equation:

$$\begin{aligned} H(k) &= H(\mathbf{x}(k), u(k), \boldsymbol{\lambda}(k+1), k) \\ &= \frac{1}{2}\left(y_{id}(k) - \frac{1}{\hat{k}_1} x_1(k)\right)^2 + \lambda_1(k+1) \\ &\quad \cdot \hat{k}_1 \sum_{\hat{p}=1}^{\hat{N}(\hat{t},\hat{m})} \hat{\varpi}_{\hat{p}} \prod_{\hat{q}=1}^{\hat{t}-1} x_{\hat{q}}^{\hat{\lambda}(\hat{p},\hat{q})}(k)\, \hat{l}_1^{\hat{\lambda}(\hat{p},\hat{t})} u^{\hat{\lambda}(\hat{p},\hat{t})}(k) \\ &\quad + \sum_{j=2}^{n_y+1} \lambda_j(k+1) \frac{\hat{k}_j}{\hat{k}_{j-1}} x_{j-1}(k) + \lambda_{n_y+2}(k+1)\, \hat{l}_2 u(k) \\ &\quad + \sum_{j=n_y+3}^{\hat{t}-1} \lambda_j(k+1) \frac{\hat{l}_{j-n_y}}{\hat{l}_{j-1-n_y}} x_{j-1}(k) \end{aligned} \tag{38}$$

where $\mathbf{x}(k)$ and $\boldsymbol{\lambda}(k)$ satisfy the following conditions:

$$\mathbf{x}(k+1) = \frac{\partial H(k)}{\partial \boldsymbol{\lambda}(k+1)}, \tag{39}$$

$$\boldsymbol{\lambda}(k) = \frac{\partial H(k)}{\partial \mathbf{x}(k)}. \tag{40}$$

If the control vector is constrained, Hamiltonian function takes the extreme value on the optimal control sequence by the minimum principle; i.e., take extreme value on the extreme values $\mathbf{x}^*(k)$ of the optimal trajectory and the optimal control law $u^*(k)$, that is,

$$\begin{aligned} &H\left(\mathbf{x}^*(k), u^*(k), \boldsymbol{\lambda}(k+1), k\right) \\ &\quad = \min_{u(k)\in\Omega} H\left(\mathbf{x}^*(k), u(k), \boldsymbol{\lambda}(k+1), k\right), \end{aligned} \tag{41}$$

where Ω is a bounded closed set.

If the control vector is not constrained, Hamiltonian function takes extreme value from the whole control space R, the extreme condition being

$$\frac{\partial H(k)}{\partial u(k)} = 0, \tag{42}$$

and $u(k) \in R$.

The given series of control sequence $u(k)$ ($k = 0, 1, \ldots, \mathrm{N} - 1$) can be improved by repeated iteration in the direction that makes the gradient of Hamiltonian function $H(k)$ decrease, until the necessary condition (42) is satisfied. Then we obtain the numerical solution of the optimal control law $u^*(k)$, where $k = 0, 1, \ldots, \mathrm{N} - 1$. For convenience, let $\mathbf{U} = (u(0), u(1), \ldots, u(\mathrm{N} - 1))^{\mathrm{T}}$, and the calculation steps are as follows.

Algorithm 2.

Step 1. Set any given series of control sequence $u_M(k) = u_0(k)$, where M is the number of iterations, and the initial value is set as $M = 0$, and $k = 0, 1, \ldots, \mathrm{N} - 1$.

Step 2. Solve the state variable $\mathbf{x}_M(k)$ sequentially by formula (35) based on $\mathbf{U}_M$ and the initial condition $\mathbf{x}(0)$, where $k = 1, 2, \ldots, \mathrm{N}$.

Step 3. Calculate $g_M(k) = (\partial H(k)/\partial u(k))|_{u(k)=u_M(k)}$, which is the gradient of $H(k)$ with respect to $u(k)$ in the control sequence $u_M(k)$, and set $\mathbf{G}_M = (g_M(0), g_M(1), \ldots, g_M(\mathrm{N} - 1))^{\mathrm{T}}$, where $k = 0, 1, \ldots, \mathrm{N} - 1$.

Step 4. Calculate $\|\mathbf{G}_M\|$. If $\|\mathbf{G}_M\| < \varepsilon$, stop, or else, revise the control vector: $\mathbf{U}_{M+1} = \mathbf{U}_M - \sigma \mathbf{G}_M$, i.e., $u_{M+1}(k) = u_M(k) - \sigma g_M(k)$, where ε is a given value, σ is a fixed step size, and $k = 0, 1, \ldots, \mathrm{N} - 1$.

Step 5. Let $M = M + 1$, and return to Step 2.

5.2. Initial Weight Values of MTNC. MTNC is generated automatically to approximate the numerical solution $u^*(k)$ of the optimal control law for the offline model (4) of the system (1) relative to the ideal output signal $y_{id}(k)$. The weight coefficients of MTNC are obtainable by offline learning, and the block diagram of offline learning for MTNC (29) is shown in Figure 8.

The initial weight values of MTNC are secured offline by the CG method as follows.

Define the appropriate error as

$$\bar{e}(k) = u^*(k) - u(k), \tag{43}$$

where $u(k)$ is the output of the controller MTNC at time k.

The corresponding mean square error is

$$\bar{E} = \frac{1}{2} \sum_{k=0}^{\mathrm{N}-1} \bar{e}^2(k). \tag{44}$$

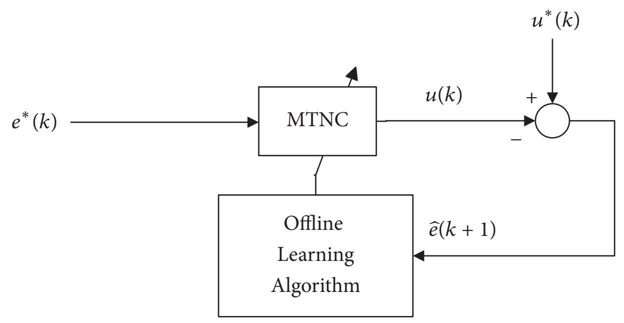

FIGURE 8: Block diagram of offline learning of MTNC.

Substituting (43) and (29) into (44) yields

$$\overline{E}=\frac{1}{2}\sum_{k=0}^{N-1}\left(u^*(k)-\sum_{p=1}^{N(t,m)}w_p\prod_{q=1}^{t}z_q^{\lambda(p,q)}(k)\right)^2. \tag{45}$$

Set

$$\begin{aligned}\boldsymbol{\alpha}(k)=&\left(\prod_{q=1}^{t}z_q^{\lambda(1,q)}(k),\prod_{q=1}^{t}z_q^{\lambda(2,q)}(k),\ldots,\right.\\&\left.\prod_{q=1}^{t}z_q^{\lambda(N(t,m),q)}(k)\right)^{\mathrm{T}},\\\mathbf{A}=&\left(\boldsymbol{\alpha}(0),\boldsymbol{\alpha}(1),\ldots,\boldsymbol{\alpha}(\mathrm{N}-1)\right),\end{aligned} \tag{46}$$

and formula (45) can be rewritten as

$$\begin{aligned}\overline{E}&=\frac{1}{2}\sum_{k=0}^{N-1}\left(u^*(k)-\mathbf{w}^{\mathrm{T}}\boldsymbol{\alpha}(k)\right)^2\\&=\frac{1}{2}\mathbf{w}^{\mathrm{T}}\sum_{k=0}^{N-1}\left(\boldsymbol{\alpha}(k)\boldsymbol{\alpha}^{\mathrm{T}}(k)\right)\mathbf{w}\\&\quad-\left(\sum_{k=0}^{N-1}u^*(k)\boldsymbol{\alpha}^{\mathrm{T}}(k)\right)\mathbf{w}+\frac{1}{2}\sum_{k=0}^{N-1}u^{*2}(k).\end{aligned} \tag{47}$$

Calculate the partial derivative of $\overline{E}$ with respect to the weight coefficient vector $\mathbf{w}$:

$$\frac{\partial\overline{E}}{\partial\mathbf{w}}=\sum_{k=0}^{N-1}\left(\boldsymbol{\alpha}(k)\boldsymbol{\alpha}^{\mathrm{T}}(k)\right)\mathbf{w}-\sum_{k=0}^{N-1}u^*(k)\boldsymbol{\alpha}(k). \tag{48}$$

Let $\mathbf{g}=\partial\overline{E}/\partial\mathbf{w}$, $\mathbf{Q}=\sum_{k=0}^{N-1}(\boldsymbol{\alpha}(k)\boldsymbol{\alpha}^{\mathrm{T}}(k))$, and $\mathbf{b}=-\sum_{k=0}^{N-1}u^*(k)\boldsymbol{\alpha}(k)$, and we have

$$\mathbf{g}=\mathbf{Q}\mathbf{w}+\mathbf{b}. \tag{49}$$

Setting $\mathbf{u}^*=(u^*(0),u^*(1),\ldots,u^*(\mathrm{N}-1))^{\mathrm{T}}$ enables us to get the vector form of $\mathbf{Q}$, $\mathbf{b}$ and $\mathbf{g}$ as follows:

$$\begin{aligned}\mathbf{Q}&=\mathbf{A}\mathbf{A}^{\mathrm{T}},\\\mathbf{b}&=-\mathbf{A}\mathbf{u}^*,\\\mathbf{g}&=\mathbf{A}\left(\mathbf{A}^{\mathrm{T}}\mathbf{w}-\mathbf{u}^*\right).\end{aligned} \tag{50}$$

For the given numerical solution $u^*(k)$ of the optimal control, the weight coefficient vector $\mathbf{w}$ can be updated once in the negative gradient direction after each learning. Let $\mathbf{w}_\tau$ represent the value of $\mathbf{w}$ after the τth iterative training; then we have $\mathbf{w}_{\tau+1}=\mathbf{w}_\tau-\mu_\tau\mathbf{g}_\tau$, where $\mu_\tau=\mathbf{g}_\tau^{\mathrm{T}}\mathbf{g}_\tau/\mathbf{g}_\tau^{\mathrm{T}}\mathbf{Q}\mathbf{g}_\tau$. However, as the gradient path to the minimum point is zigzag, the search direction remains vertical to the last. Fortunately, the problem can be solved effectively by employing the CG method, whereby the weight can be updated as follows:

$$\mathbf{w}_{\tau+1}=\mathbf{w}_\tau+\mu_\tau\mathbf{p}_\tau, \tag{51}$$

where $\mathbf{p}_\tau=-\mathbf{g}_\tau+a_{\tau-1}\mathbf{p}_{\tau-1}$, $\mu_\tau=\mathbf{g}_\tau^{\mathrm{T}}\mathbf{g}_\tau/\mathbf{p}_\tau^{\mathrm{T}}\mathbf{Q}\mathbf{p}_\tau$, $a_{\tau-1}=\mathbf{g}_\tau^{\mathrm{T}}\mathbf{Q}\mathbf{p}_{\tau-1}/\mathbf{p}_{\tau-1}^{\mathrm{T}}\mathbf{Q}\mathbf{p}_{\tau-1}$, and the initial value is defined as $\mathbf{p}_1=-\mathbf{g}_1$.

6. Real-Time Learning for Weights of MTNC

Due to the real-time modeling error for the unknown nonlinear time-varying system and the uncertainties existing in practical applications, the controller with fixed weight coefficients cannot ensure the lasting robust performance of the system. Therefore, it is required that the controller be capable of adjusting automatically for real-time control. Similar to the real-time learning algorithm for MTNI, a novel adaptive BP algorithm is proposed here for MTNC real-time training [53–55].

The performance function is defined as

$$E(\mathbf{w}(k))=\frac{1}{2}e^2(\mathbf{w}(k)), \tag{52}$$

where $e(\mathbf{w}(k))=e(k+1)$, $e(k+1)=r(k+1)-y(k+1)$ represents the practical tracking error at time $k+1$.

The weight coefficients of MTNC are updated according to (53)-(57):

$$\mathbf{w}(k+1)=\mathbf{w}(k)+\Delta\mathbf{w}(k), \tag{53}$$

where

$$\Delta\mathbf{w}(k)=-\eta(k)\nabla E(\mathbf{w}(k))+\alpha(k)\Delta\mathbf{w}(k-1), \tag{54}$$

$$\eta(k)=\min\left\{\eta(k-1)\left(1+\eta_0\cos\theta(k)\right),\eta_{\max}\right\}, \tag{55}$$

$$\alpha(k) = \beta(k)\eta^2(k), \tag{56}$$

$$\beta(k) = \begin{cases} \beta_0 \dfrac{-\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k-1)}{\|\Delta\mathbf{w}(k-1)\|^2}, & \text{if } \nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k-1) < 0, \\ 0, & \text{else}, \end{cases} \tag{57}$$

$\nabla E(\mathbf{w}(k))$ is the partial derivative of $E(\mathbf{w}(k))$ with respect to $\mathbf{w}(k)$; η_0, $\eta_{\max}$, β_0 are constants, and $0 \le \eta_0 \le 1$, $0 < \eta_{\max} < 1$, $0 \le \beta_0 < 1$; $\theta(k)$ is the angle between the current gradient $-\nabla E(\mathbf{w}(k))$ and the previous update $\Delta\mathbf{w}(k-1)$, given by $\cos\theta(k) = -\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k-1)/\|\nabla E(\mathbf{w}(k))\|\|\Delta\mathbf{w}(k-1)\|$.

From (54) and (57), we get

$$\begin{aligned} \nabla E(\mathbf{w}(k)) &= e(\mathbf{w}(k)) \frac{de(\mathbf{w}(k))}{d\mathbf{w}(k)} \\ &= -e(\mathbf{w}(k)) \frac{dy(k+1)}{d\mathbf{w}(k)}. \end{aligned} \tag{58}$$

As the system considered is unknown, the output $y(k)$ of the actual system can be replaced with the output $\hat{y}(k)$ of the identification model (4), that is,

$$y(k+1) \approx \hat{y}(k+1) = \sum_{\hat{p}=1}^{\widehat{N}(\hat{t},\widehat{m})} \widehat{w}_{\hat{p}}(k) \prod_{\hat{q}=1}^{\hat{t}} \hat{z}_{\hat{q}}^{\hat{\lambda}(\hat{p},\hat{q})}(k), \tag{59}$$

where $\hat{t}$, $\widehat{m}$, $\widehat{N}(\hat{t},\widehat{m})$, $\hat{p}$, $\hat{q}$, $\hat{\lambda}(\hat{p},\hat{q})$, $\widehat{w}_{\hat{p}}(k)$ and $\hat{z}_{\hat{q}}(k)$ have the same meanings as in (4), and $k = 0, 1, 2, \ldots$.

Thus,

$$\frac{dy(k+1)}{d\mathbf{w}(k)} \approx \frac{d\hat{y}(k+1)}{d\mathbf{w}(k)}, \tag{60}$$

$$\frac{d\hat{y}(k+1)}{d\mathbf{w}(k)} = \hat{l}_1 \frac{\partial\hat{y}(k+1)}{\partial\hat{z}_{\hat{t}}(k)} \frac{du(k)}{d\mathbf{w}(k)}. \tag{61}$$

In (61), the first part on the right side of the equation can be calculated using the real-time identification model (4), and the second term can be done as follows:

$$\frac{du(k)}{d\mathbf{w}(k)} = \boldsymbol{\alpha}(k) \tag{62}$$

where $\boldsymbol{\alpha}(k)$ is mentioned before.

Theorem 3. *For any given set of weight vector $\mathbf{w}_0$ of MTNC used to approximate the optimal control law $u^*(k)$ offline, by taking the weight vector $\mathbf{w}_0^*$ as the initial values for online learning, we have $\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k) \le 0$ if $\mathbf{w}(k)$ is updated with the learning rules from (53) to (57).*

Proof. For the first case, i.e., if $\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k-1) < 0$ holds, then

$$\begin{aligned} \Delta\mathbf{w}(k) &= -\eta(k)\nabla E(\mathbf{w}(k)) + \alpha(k)\Delta\mathbf{w}(k-1) \\ &= \eta(k)\Bigg(-\nabla E(\mathbf{w}(k)) + \beta_0\eta(k) \\ &\quad \cdot \frac{-\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k-1)}{\|\Delta\mathbf{w}(k-1)\|^2}\Delta\mathbf{w}(k-1)\Bigg) = \eta(k) \\ &\quad \cdot \Bigg(-\nabla E(\mathbf{w}(k)) + \beta_0\eta(k) \\ &\quad \cdot \frac{\|\nabla E(\mathbf{w}(k))\|\,\|\Delta\mathbf{w}(k-1)\|\cos\theta(k)}{\|\Delta\mathbf{w}(k-1)\|^2}\Delta\mathbf{w}(k-1)\Bigg) \\ &= \eta(k)\Bigg(-\nabla E(\mathbf{w}(k)) + \beta_0\eta(k) \\ &\quad \cdot \cos\theta(k)\frac{\|\nabla E(\mathbf{w}(k))\|}{\|\Delta\mathbf{w}(k-1)\|}\Delta\mathbf{w}(k-1)\Bigg), \end{aligned} \tag{63}$$

$$\begin{aligned} &\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k) = \nabla E^{\mathrm{T}}(\mathbf{w}(k))\Delta\mathbf{w}(k) = \eta(k) \\ &\quad \cdot \Bigg(-\|\nabla E(\mathbf{w}(k))\|^2 + \beta_0\eta(k)\cos\theta(k) \\ &\quad \cdot \frac{\|\nabla E(\mathbf{w}(k))\|}{\|\Delta\mathbf{w}(k-1)\|}\nabla E^{\mathrm{T}}(\mathbf{w}(k))\Delta\mathbf{w}(k-1)\Bigg) = -\eta(k) \\ &\quad \cdot \|\nabla E(\mathbf{w}(k))\|^2\left(1 + \beta_0\eta(k)\cos^2\theta(k)\right). \end{aligned} \tag{64}$$

As a matter of fact, $\eta(k) \ge 0$ and $1 + \beta_0\eta(k)\cos^2\theta(k) > 0$ when $0 \le \beta_0 < 1$ and $0 \le \eta_0 \le 1$, thus $\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k) \le 0$ holds.

For the second case, i.e., if $\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k-1) \ge 0$, then $\beta(k) = 0$, $\alpha(k) = 0$, and $\Delta\mathbf{w}(k) = -\eta(k)\nabla E(\mathbf{w}(k))$, then

$$\begin{aligned} \nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k) &= \nabla E^{\mathrm{T}}(\mathbf{w}(k))\Delta\mathbf{w}(k) \\ &= -\eta(k)\|\nabla E(\mathbf{w}(k))\|^2 \le 0. \end{aligned} \tag{65}$$

The above two cases verify that $\nabla E(\mathbf{w}(k)) \cdot \Delta\mathbf{w}(k) \le 0$ holds.

That completes the proof of Theorem 3. □

7. Algorithm for MTN Optimal Control Scheme

Steps of the algorithm for MTN optimal control scheme are summarized as follows.

Algorithm 4.

Step 1. Build the model MTNI for the system (1), train it offline, and take the trained weight parameters as the initial values of MTNI for online identification, where $k = 0, 1, \ldots, \mathrm{N}-1$. In the process of offline training, M sequence is selected as the persistently exciting input signal for system (1) and model (4).

Step 2. Select an ideal output signal $y_{id}(k)$ for system (1) relative to the given reference signal $r(k)$.

Step 3. Call Algorithm 2 to calculate the optimal control law $u^*(k)$ of the system (1) relative to the ideal output signal $y_{id}(k)$ offline, and the corresponding optimal output signal $y^*(k)$ is taken as the desired output signal.

Step 4. Construct MTNC to fit the optimal control law $u^*(k)$, select a group of initial weight parameter vector $\mathbf{w}_0$ in the interval (-1, 1) in a random way for offline training, and train MTNC by CG method to obtain the initial weight parameter vector $\mathbf{w}_0^*$ for online training.

Step 5. Obtain $r(k)$, $u(k)$, and $y(k)$ by sampling, and take the real-time tracking error $e(k)$ as $e(k) = r(k) - y(k)$, where $k = 0, 1, \ldots$.

Step 6. Obtain the input signal $u(k)$ of the system (1) by substituting $\mathbf{z}(k)$ into MTNC controller (29), and $u(k)$ into (1) and (4) respectively.

Step 7. Utilize the formulas (19)-(23) to adjust the weight parameter vector for MTNI online and the formulas (53)-(57) to tune that for MTNC online.

Step 8. Return to Step 5 to continue the process.

8. Simulation Example

To demonstrate the effectiveness of the proposed MTN optimal control scheme, simulation results are presented from the following example, that is, a modification of example 2 derived from [57]. Consider the following SISO nonlinear time-varying discrete system described by the input-output difference equation:

$$y(k+1) = \frac{a(k)\, y(k) + b(k)}{1 + y^2(k)} + c(k)\, y(k-1) + u(k) + c(k)\, u^2(k-1), \tag{66}$$

where $a(k) = 0.96(1 - 0.5e^{-0.2k})$, $b(k) = 0.1\lg(k+1)$ and $c(k) = e^{-0.2k}$. The given reference signal is defined as $r(k)$=1. MTNI takes the 4-15-1 structure with 4 input nodes, 2 powers, and 1 output node, and its input vector is $[y(k), y(k-1), u(k-1), u(k)]^{\mathrm{T}}$. Let $\widehat{\mathbf{z}}(k) = [\widehat{k}_1 y(k), \widehat{k}_2 y(k-1), \widehat{l}_2 u(k-1), \widehat{l}_1 u(k)]^{\mathrm{T}}$, and set $\widehat{k}_1$, $\widehat{k}_2$, $\widehat{l}_1$, $\widehat{l}_2$ as 0.001. As comparison, the system model is also built by the NN identifier (NNI) and PIDNN identifier (PIDNNI). NNI is built from a three-layer NN of the 4-50-1 structure with 4 input nodes, 50 hidden neurons, and 1 output neuron. As confirmed by our simulation, a better identification result can be obtained when 50 hidden neurons are chosen for NNI. The activation functions for the hidden and output layer are chosen as $\widehat{a}_h(x) = 1/(1+e^{-x})$ and $\widehat{a}_o(x) = (1-e^{-x})/(1+e^{-x})$, respectively. PIDNNI is built of a three-layer network of the 4-3-1 structure with 4 input nodes, 3 hidden neurons, and 1 output neuron. The PID neuron structure is obtainable from [58, 59]. At the same time, input vectors for NNI and PIDNNI are the same as for MTNI. For offline identification, the initial weight parameters for MTNI and NNI are chosen randomly in the interval $(-1, 1)$. The initial weight parameters between the input and hidden layer for PIDNNI are chosen as $\widehat{\mathbf{w}}_{s1} = [1, 1, 1, 1]^{\mathrm{T}}$, $\widehat{\mathbf{w}}_{s2} = [0.1, 0.1, -0.1, -0.1]^{\mathrm{T}}$, and $\widehat{\mathbf{w}}_{s3} = [1, 1, -1, -1]^{\mathrm{T}}$. The initial weight parameters between the hidden and output layer for PIDNNI are taken as $\widehat{\mathbf{w}}_d = [0.1, 0.1, 0.1]^{\mathrm{T}}$, and the BP algorithm is chosen for the learning process. Substituting the output $\widehat{y}(k)$ of MTNI for the output $y(k)$ of the system (66) gives

$$\begin{aligned} \mathbf{x}(k) &= [x_1(k), x_2(k), x_3(k)]^{\mathrm{T}} \\ &= \left[\widehat{k}_1 \widehat{y}(k), \widehat{k}_2 \widehat{y}(k-1), \widehat{l}_2 u(k-1)\right]^{\mathrm{T}}. \end{aligned} \tag{67}$$

And its corresponding extended state space is

$$\begin{aligned} x_1(k+1) &= \widehat{k}_1 \widehat{w}_1 + \widehat{k}_1 \widehat{w}_2 x_1(k) + \widehat{k}_1 \widehat{w}_3 x_2(k) \\ &\quad + \widehat{k}_1 \widehat{w}_4 x_3(k) + \widehat{k}_1 \widehat{l}_1 \widehat{w}_5 u(k) \\ &\quad + \widehat{k}_1 \widehat{w}_6 x_1^2(k) + \widehat{k}_1 \widehat{w}_7 x_1(k) x_2(k) \\ &\quad + \widehat{k}_1 \widehat{w}_8 x_1(k) x_3(k) \\ &\quad + \widehat{k}_1 \widehat{l}_1 \widehat{w}_9 x_1(k) u(k) + \widehat{k}_1 \widehat{w}_{10} x_2^2(k) \\ &\quad + \widehat{k}_1 \widehat{w}_{11} x_2(k) x_3(k) \\ &\quad + \widehat{k}_1 \widehat{l}_1 \widehat{w}_{12} x_2(k) u(k) + \widehat{k}_1 \widehat{w}_{13} x_3^2(k) \\ &\quad + \widehat{k}_1 \widehat{l}_1 \widehat{w}_{14} x_3(k) u(k) \\ &\quad + \widehat{k}_1 \widehat{l}_1^2 \widehat{w}_{15} u^2(k), \\ x_2(k+1) &= \frac{\widehat{k}_2}{\widehat{k}_1} x_1(k), \\ x_3(k+1) &= \widehat{l}_2 u(k), \end{aligned} \tag{68}$$

$$\widehat{y}(k) = \frac{1}{\widehat{k}_1} x_1(k) \tag{69}$$

where $\widehat{w}_j$ $(j = 1, 2, \ldots, 15)$ can be obtained offline.

To generate the control input, MTNC takes the 3-10-1 structure with 3 input nodes, 2 powers, and 1 output node, its input vector being $\mathbf{z}(k) = [k_1 u(k-1), l_1 e(k-1), l_2 e(k-2)]^{\mathrm{T}}$, where k_1, l_1, l_2 are set as 0.001. As comparison, the NN adaptive controller (NNAC) is built from a three-layer NN of the 3-50-1 structure with 3 input neurons, 50 hidden neurons, and 1 output neuron. As demonstrated by our experiments, a better control result can be obtained when 50 hidden neurons are chosen for NNAC. The activation functions for the hidden and output layer are chosen as $a_h(x) = 1/(1+e^{-x})$ and $a_o(x) = (1-e^{-x})/(1+e^{-x})$, respectively. The PIDNN adaptive controller (PIDNNAC) takes the 2-3-1 structure with 2 input nodes, 3 hidden neurons, and 1 output node, respectively. Before the real-time control process, the MTNC controller is generated automatically by fitting the optimal control law $u^*(k)$ using the CG method, with the initial weight values of MTNC chosen randomly in the interval (-1, 1) and the iterations of fitting set as ite=100. For online identification and control, the initial weight values of MTNI and MTNC are taken as those from the offline learning. The novel adaptive BP algorithm with the learning rules from (19) to (23) is employed to update the weight values of MTNI to build the system model in real time, and the formulas from (53) to (57) are utilized to update the those of MTNC to implement the output tracking control of the system (66) relative to the given reference signal $r(k)$. The initial weight values for NNAC are chosen randomly in the interval (-1, 1), those of PIDNNAC between the input and hidden layer are set as $\mathbf{w}_{s1} = [1, 1]^{\mathrm{T}}$, $\mathbf{w}_{s2} = [0.1, -0.1]^{\mathrm{T}}$, and $\mathbf{w}_{s3} = [1, -0.1]^{\mathrm{T}}$, and those of PIDNNI are set as $\mathbf{w}_d = [0.1, 0.1, 0.1]^{\mathrm{T}}$. BP algorithm is adopted to train NNAC and PIDNNAC online. For online identification and control, the initial values of $\hat{\eta}(0)$, $\hat{\eta}_0$, $\hat{\beta}_0$, and $\hat{\eta}_{\max}$ for MTNI are set as 0.2, 0.001, 0.01, and 0.5, respectively, and those of $\eta(0)$, η_0, β_0, and $\eta_{\max}$ for MTNC as 0.2, 0.001, 0.01, and 0.5, respectively. The learning factor is set as $\hat{\alpha}_{\mathrm{N}} = 0.2$ for NNI, $\alpha_{\mathrm{N}} = 0.2$ for NAC, $\hat{\alpha}_{\mathrm{P}} = 0.3$ for PIDNNI, and $\alpha_{\mathrm{P}} = 0.3$ for PIDNNAC. The tracking results and errors for the system with MTNC, NNAC, and PIDNNAC are shown as in Figures 9 and 10, respectively, and the corresponding control inputs are presented by Figure 11.

Remark 5. In Figures 9, 10, and 11, r represents the given reference signal; yMC, yNAC, and yPNAC are the actual output responses for MTNC, NN, and PIDNN control schemes, respectively; eMC, eNAC, and ePNAC represent the corresponding tracking errors; uMC, uNAC, and uPNAC are the corresponding control inputs.

From Figures 9 and 10, it can be seen that the overshoots are 31.19%, 88.43%, and 67.29%; the performance index is $E<10^{-3}$ after the iterations of 14, 20, and 24 with MTN, NN, and PIDNN control schemes; the steady-state error worked out by taking the average of the absolute errors from the iterations of 14 to 300 is 0.0024 with the MTN optimal control scheme, 0.0011 taken from 20 to 300 with the traditional NN adaptive control scheme, and 8.0383×10^{-4} taken from 24 to 300 with the PIDNN control scheme. Simulation results demonstrate that the proposed control scheme outperforms the others.

(a) Noise Interference Experiments. At time 100, a Gaussian white noise with the mean of 0 and the standard deviation of 0.2 is added to system (66). The simulation results are presented in Figures 12 and 13, and the corresponding control inputs are shown in Figure 14.

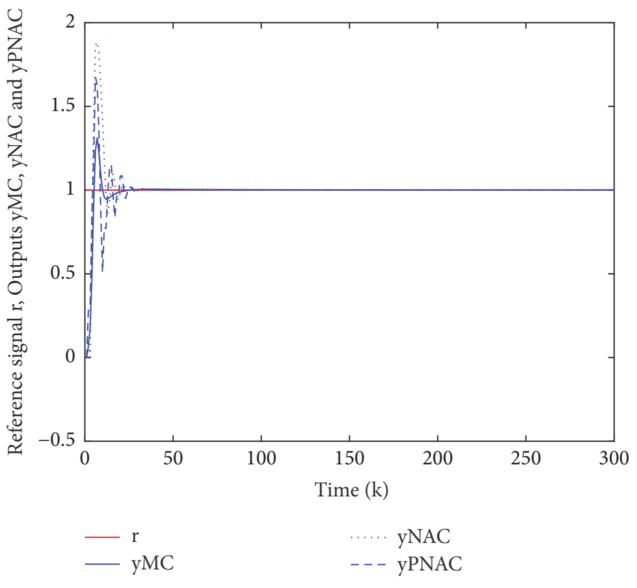

FIGURE 9: The tracking trajectories.

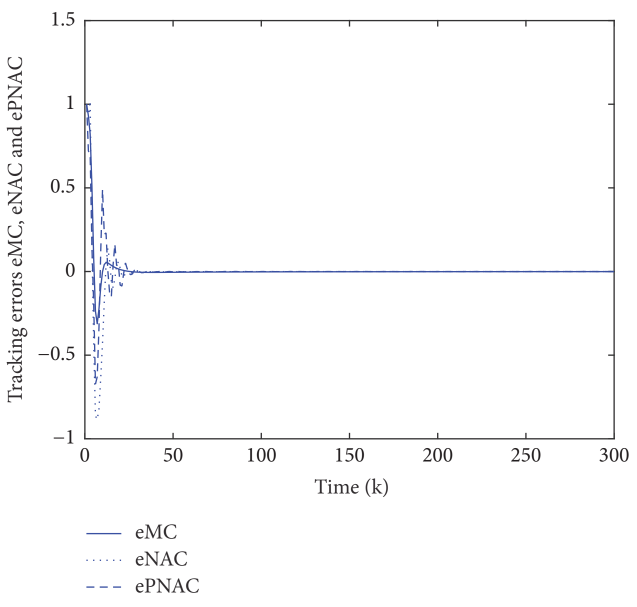

FIGURE 10: The tracking errors.

As demonstrated by Figures 12 and 13, the MTN optimal control scheme promises a better robustness for the noise interference than the other two.

For clearer illustration of the robustness of the proposed control scheme, the noise interference is expanded 30 times based on the above discussion, and the simulation results are presented in Figures 15 and 16 and the corresponding control inputs are shown in Figure 17.

(b) Input Superposition Experiments. When the external superposition $d(k) = 0.05\sin(0.005\pi k)$ is added to the given reference signal $r(k)$, the simulation results are shown in Figures 18 and 19 and the corresponding control inputs are shown as in Figure 20.

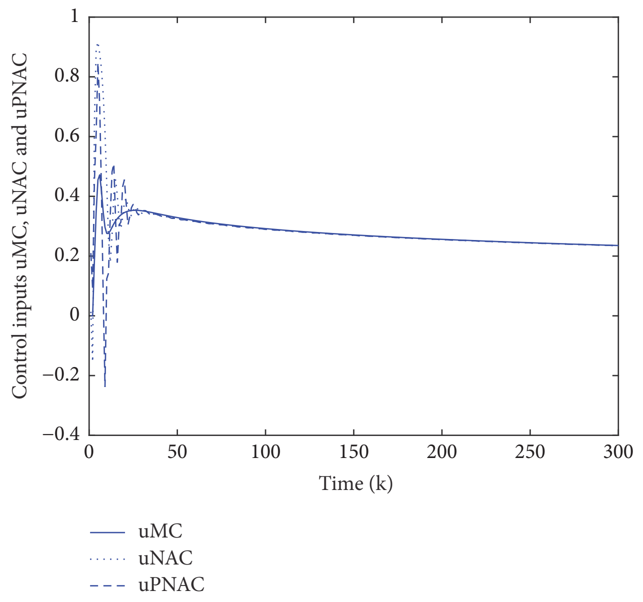

FIGURE 11: The control inputs.

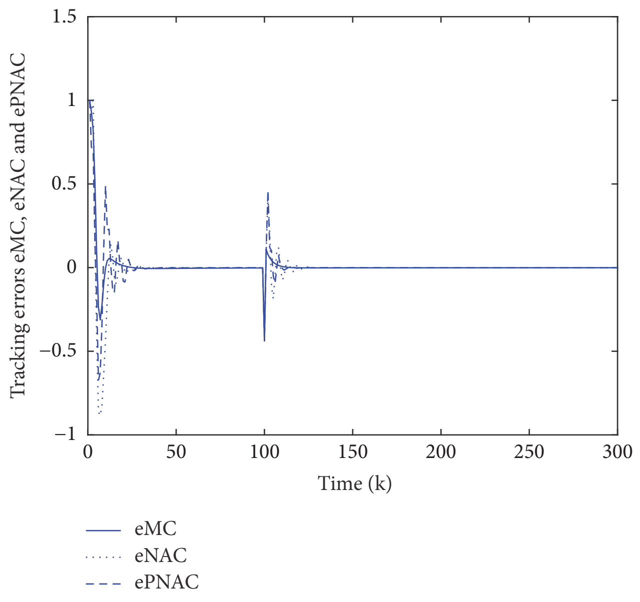

FIGURE 13: The tracking errors with noise interference.

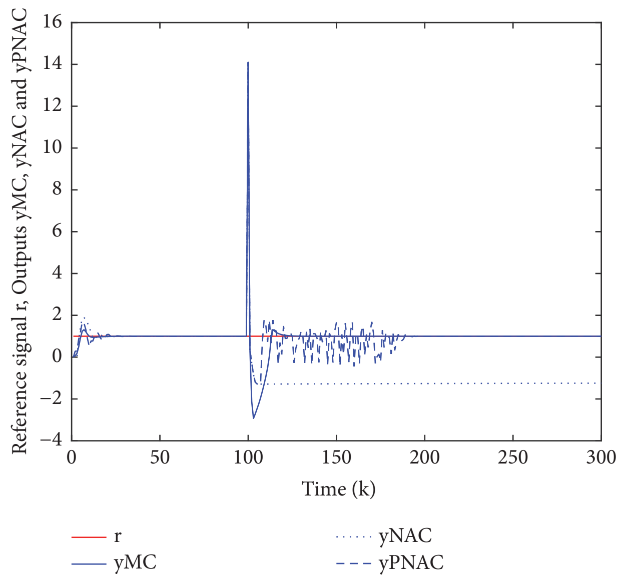

FIGURE 12: The tracking trajectories with noise interference.

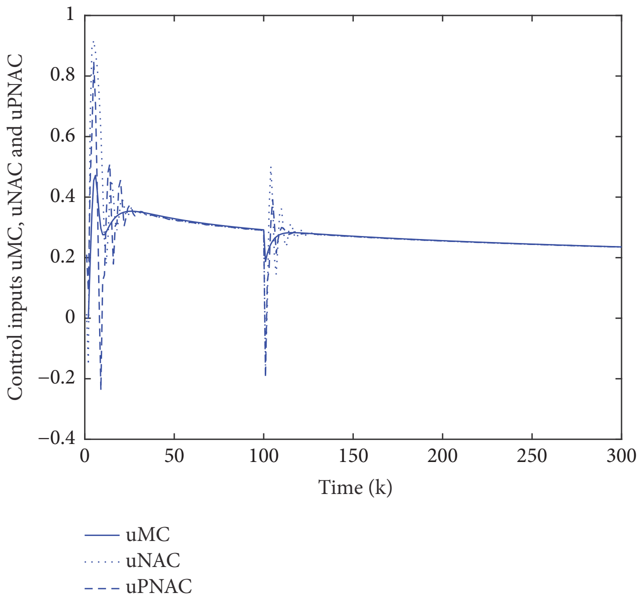

FIGURE 14: The control inputs with noise interference.

As demonstrated by Figures 9–20, the proposed MTN scheme is valid and of desirable for the alteration of the system parameters.

Remark 6. For each control period, 29 times of multiplication operations and 14 times of addition operations are needed for MTNI, and 19 times of multiplication and 9 times of addition for MTNC. That is, 48 times of multiplication and 23 times of addition are required by the MTN scheme. However, with the exponential function expanded into finite terms with 2 powers, 8-time multiplication and 9-time addition operations are needed for each hidden node, and 57-time multiplication and 61-time addition operations are required for the output layer node of NNI. Meanwhile, data normalization and inverse normalization are considered for the online identification process, and the two functions are defined as $f_o(x) = (2/\pi)\arctan x$ and $f_i(x) = \tan((\pi/2)x)$, respectively. For each normalization, 5-time multiplication and 1-time addition operations are needed with $f_o(x)$ expanded into finite terms with 3 powers; 5-time multiplication and 1-time addition operations are required with $f_i(x)$ expanded into finite terms with 3 powers. For NNAC, 1-time multiplication and 3-time addition operations are demanded for the input layer, 7-time multiplication and 8-time addition operations for each hidden node, and 57-time multiplication and 61-time addition operations for each output layer node. That is, for each control period,

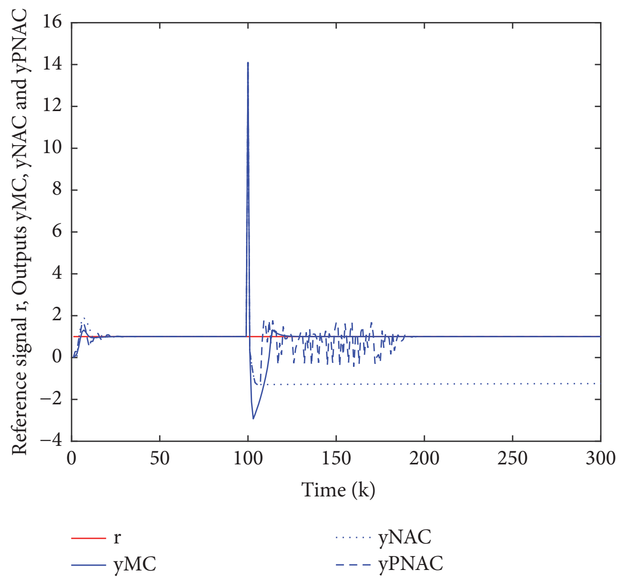

Figure 15: The tracking trajectories with expanded noise interference.

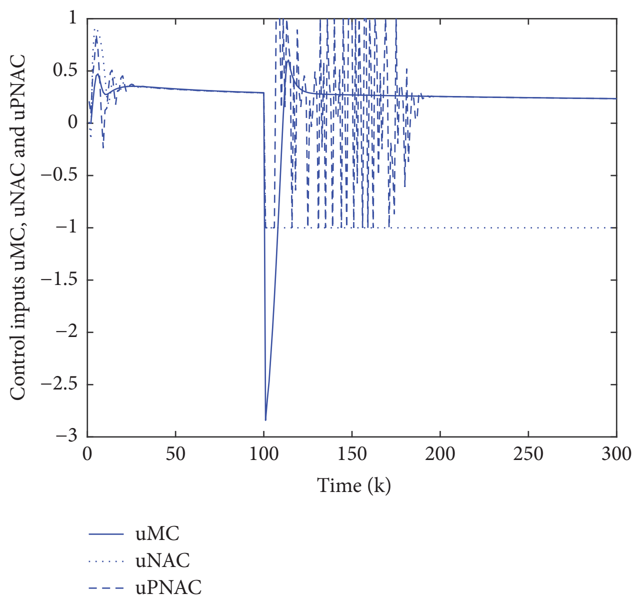

Figure 17: The control inputs with expanded noise interference.

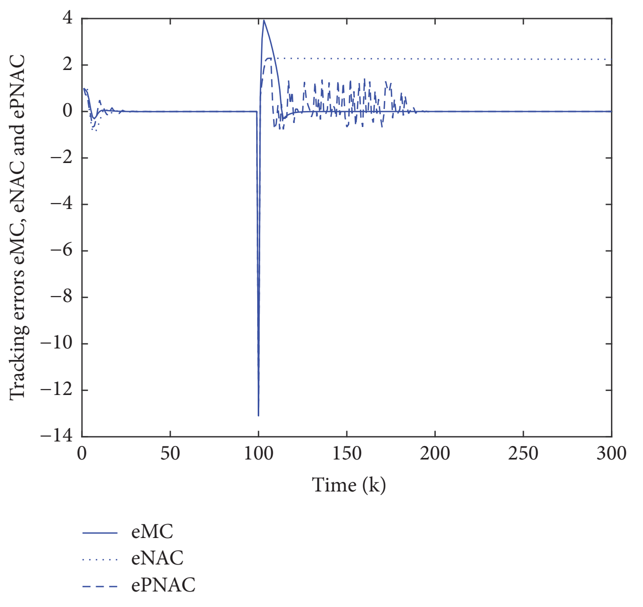

Figure 16: The tracking errors with expanded noise interference.

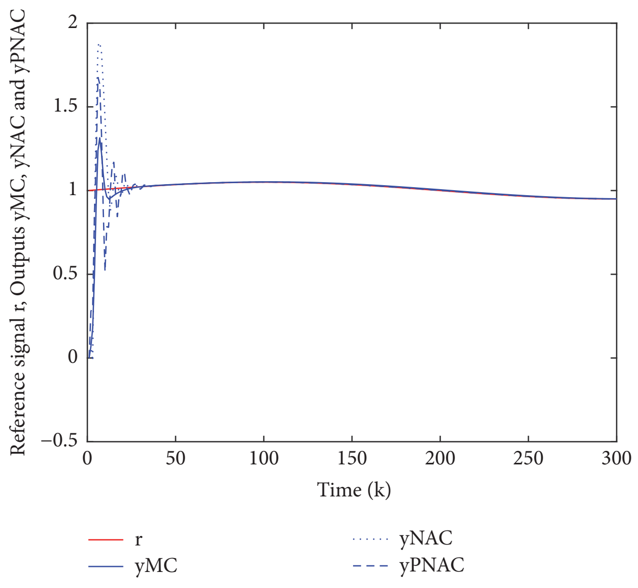

Figure 18: The tracking trajectories with signal superposition.

875-time multiplication and 977-time addition operations are needed implementing the traditional neural network adaptive control scheme. With the PIDNN control scheme, for PIDNNI, 12-time multiplication operations and 11-time addition operations are required for the hidden layer nodes, and 3-time multiplication and 2-time addition operations for the output layer node. Meanwhile, data normalization and inverse normalization are also considered for the online identification process, and the two functions are defined as $f_o(x) = (2/\pi)\arctan x$ and $f_i(x) = \tan((\pi/2)x)$, respectively. Then, for each normalization, 5-time multiplication and 1-time addition operations are needed with $f_o(x)$ expanded into finite terms with 3 powers; 5-time multiplication and 1-time addition operations are required with $f_i(x)$ expanded into finite terms with 3 powers. For PIDNNAC, 6-time multiplication and 5-time addition operations are demanded for the hidden layer nodes, and 3-time multiplication and 2-time addition operations for the output layer node. That is, for each control period, 34-time multiplication and 22-time addition operations are needed implementing the PIDNN control scheme, while fewer operations are required by the MTN and PIDNN schemes. And compared to the type TMS320F28335 of DSP with the dominant frequency of 150MHz, the computation time for each control period is 473.3ns, 12346.7ns, and 373.3ns with the MTN, NN, and

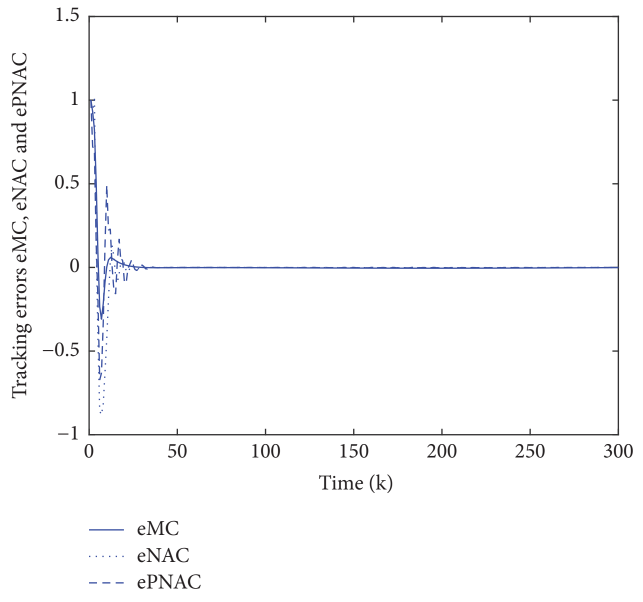

Figure 19: The tracking errors with signal superposition.

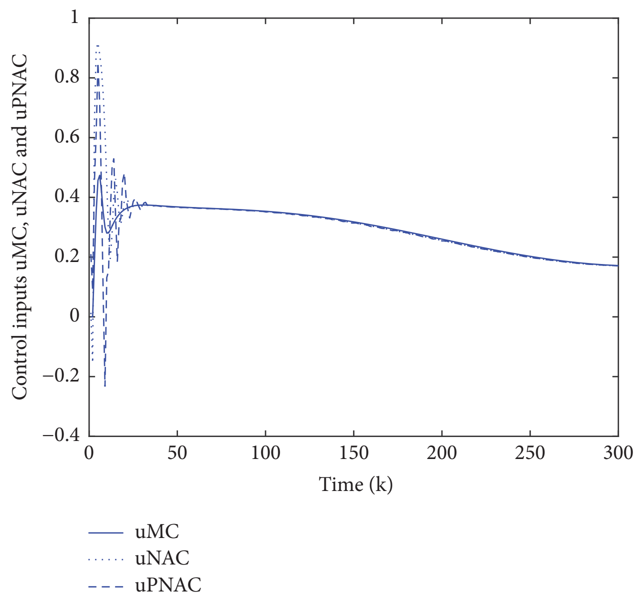

Figure 20: The control inputs with signal superposition.

PIDNN control scheme, respectively. As verified by the simulation and calculation results, the MTN and PIDNN control schemes promise more desirable real-time control than the NN scheme.

9. Conclusions

For the SISO nonlinear time-varying discrete system without mechanism model, a MTN optimal control scheme has been proposed to secure its real-time output tracking control based on the given reference signal.

Main contributions of the paper can be summarized as follows: (1) MTN optimal control scheme has been proposed for general nonlinear time-varying discrete system control design; (2) MTNI identifier and MTNC controller have been built to simplify the network structure and raise the convergence speed; (3) initial value selection scheme for the weight parameters of the controller has been developed; (4) novel adaptive BP learning algorithm has been proposed to adjust the weight parameters for a faster convergence speed.

Simulation results show that the proposed control scheme is effective and capable of enabling the system's actual output response to well track the given reference signal in real time.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Jiao-Jun Zhang and Hong-Sen Yan contributed equally to this work.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 61673112 and 60934008, the Fundamental Research Funds for the Central Universities of China under Grants 2242017K10003 and 2242014K10031, and the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD). The authors thank Professor Li Lu for his valuable comments and suggestions.

References

[1] A. Astolfi, G. Kaliora, and Z. P. Jiang, "Output feedback stabilization and approximate and restricted tracking for a class of cascaded systems," *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, vol. 50, no. 9, pp. 1390–1396, 2005.

[2] D. Karagiannis, A. Astolfi, and R. Ortega, "Two results for adaptive output feedback stabilization of nonlinear systems," *Automatica*, vol. 39, no. 5, pp. 857–866, 2003.

[3] A. Isidori, "A tool for semiglobal stabilization of uncertain non-minimum-phase nonlinear systems via output feedback," *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, vol. 45, no. 10, pp. 1817–1827, 2000.

[4] A. Isidori, A. R. Teel, and L. Praly, "A note on the problem of semiglobal practical stabilization of uncertain nonlinear systems via dynamic output feedback," *Systems & Control Letters*, vol. 39, no. 3, pp. 165–171, 2000.

[5] Z. P. Jiang, "Decentralized disturbance attenuating output-feedback trackers for large-scale nonlinear systems," *Automatica*, vol. 38, no. 8, pp. 1407–1415, 2002.

[6] G. Damm, R. Marino, and F. Lamnabhi-Lagarrigue, "Adaptive nonlinear output feedback for transient stabilization and voltage regulation of power generators with unknown parameters," *International Journal of Robust and Nonlinear Control*, vol. 14, no. 9-10, pp. 833–855, 2004.

[7] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 1, no. 1, pp. 4–27, 1990.

[8] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *International Journal of Control*, vol. 56, no. 2, pp. 319–346, 1992.

[9] S. Chen and S. A. Billings, "Representations of nonlinear systems: the NARMAX model," *International Journal of Control*, vol. 49, no. 3, pp. 1013–1032, 1989.

[10] A. Patrikar and J. Provence, "Nonlinear system identification and adaptive control using polynomial networks," *Mathematical and Computer Modelling*, vol. 23, no. 1-2, pp. 159–173, 1996.

[11] I. Galván and J. Zaldivar, "Application of recurrent neural networks in batch reactors, part I. NARMA modelling of the dynamic behavior of the heat transfer fluid temperature," *Chemical Engineering and Processing*, vol. 36, no. 6, pp. 505–518, 1997.

[12] I. Galván and J. Zaldivar, "Application of recurrent neural networks in batch reactors, part II. Nonlinear inverse and predictive control of the heat transfer fluid temperature," *Chemical Engineering and Processing*, vol. 37, no. 2, pp. 149–161, 1998.

[13] S. L. Kukreja, H. L. Galiana, and R. E. Kearney, "NARMAX representation and identification of ankle dynamics," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 1, pp. 70–81, 2003.

[14] S. W. Pang, K. P. Yu, and J. X. Zou, "Nonlinear time-varying system identification based on time-varying NARMA model," *Engineering Mechanics*, vol. 23, no. 12, pp. 25–29, 2006.

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[16] H. S. Yu, J. Z. Peng, and Y. D. Tang, "Identification of nonlinear dynamic systems using Hammerstein-type neural network," *Mathematical Problems in Engineering*, vol. 2014, Article ID 959507, 9 pages, 2014.

[17] Q. Zhu and J. Cao, "Stability analysis for stochastic neural networks of neutral type with both Markovian jump parameters and mixed time delays," *Neurocomputing*, vol. 73, no. 13-15, pp. 2671–2680, 2010.

[18] J. G. Kuschewski, S. H. Żak, and S. Hui, "Application of Feedforward Neural Networks to Dynamical System Identification and Control," *IEEE Transactions on Control Systems Technology*, vol. 1, no. 1, pp. 37–49, 1993.

[19] P. S. Sastry, G. Santharam, and K. P. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 5, no. 2, pp. 306–319, 1994.

[20] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 4, pp. 349–366, 2000.

[21] L. Q. Xu and D. C. Hu, "Comparison of two typical fault-tolerance algorithms of neural networks," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 28, no. 5, pp. 700–707, 2002.

[22] H. Cui and X. Peng, "Short-term city electric load forecasting with considering temperature effects: an improved ARIMAX model," *Mathematical Problems in Engineering*, vol. 2015, Article ID 589374, 10 pages, 2015.

[23] Q. Zhu and J. Cao, "Stability analysis of markovian jump stochastic BAM neural networks with impulse control and mixed time delays," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 3, pp. 467–479, 2012.

[24] D. Niu, Y. Lu, X. Xu, and B. Li, "Short-term power load point prediction based on the sharp degree and chaotic RBF neural network," *Mathematical Problems in Engineering*, vol. 2015, Article ID 231765, 8 pages, 2015.

[25] H. S. Yan, "Comparison between neural network, multi-variable polynomial regression and multi-dimensional Taylor network," Tech. Rep., Research Institute of Control and Optimization of Manufacturing Systems, School of Automation, Southeast University, China, 2014.

[26] B. Zhou and H. S. Yan, "Financial time series forecasting based on wavelet and multi-dimensional Taylor network dynamics model," *Systems Engineering—Theory and Practice*, vol. 33, no. 10, pp. 2654–2662, 2013, (in Chinese).

[27] B. Zhou and H. S. Yan, "Time series forecasting based on the empirical mode decomposition multi-dimensional Taylor network model," in *Proceedings of the 9th International Conference on Natural Computation (ICNC '13)*, pp. 1194–1198, China, 2013.

[28] B. Zhou and H. S. Yan, "Non-linear system identification and prediction based on dynamics cluster multi-dimensional Taylor network model," *Control and Decision*, vol. 29, no. 1, pp. 33–38, 2014 (Chinese).

[29] B. Zhou and H. S. Yan, "A dynamics model based on intermittent feedback multi-dimensional Taylor network," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 40, no. 7, pp. 1517–1521, 2014.

[30] Y. Lin, H. S. Yan, and B. Zhou, "Non-linear time series prediction method based on multi-dimensional Taylor network and its applications," *Control and Decision*, vol. 29, no. 5, pp. 795–801, 2014.

[31] Y. Lin, H. S. Yan, and B. Zhou, "A novel modeling method based on multi-dimensional Taylor network and its application in time series prediction," *Advanced Materials Research*, vol. 940, pp. 480–484, 2014.

[32] Y. Lin and H. S. Yan, "The model of multi-scale alternate positive negative feedbackics and its applications," *Control Theory and Applications*, vol. 33, no. 7, pp. 879–888, 2016.

[33] H. S. Yan, *Multi-Dimensional Taylor Network Optimal Control*, School of Automation, Southeast University, Nanjing, China, 2017, http://automation.seu.edu.cn/Articles.aspx?id=3487.

[34] Q. M. Sun and H. S. Yan, "Optimal adjustment control of SISO nonlinear systems based on multi-dimensional taylor network only by output feedback," *Advanced Materials Research*, vol. 1049-1050, pp. 1389–1391, 2014.

[35] H. S. Yan and A. M. Kang, "Asymptotic tracking and dynamic regulation of SISO non-linear system based on discrete multi-dimensional Taylor network," *IET Control Theory & Applications*, vol. 11, no. 10, pp. 1619–1626, 2017.

[36] D. S. Laila and A. Astolfi, "Input-to-state stability for discrete-time time-varying systems with applications to robust stabilization of systems in power form," *Automatica*, vol. 41, no. 11, pp. 1891–1903, 2005.

[37] F. Mazenc, "Strict Lyapunov functions for time-varying systems," *Automatica*, vol. 39, no. 2, pp. 349–353, 2003.

[38] S. K. Sood and R. Sandhu, "Matrix based proactive resource provisioning in mobile cloud environment," *Simulation Modelling Practice and Theory*, vol. 50, pp. 83–95, 2015.

[39] B. Pearlmutter, "Gradient Descent: Second order momentum and saturating error," in *Proceedings of the Neural Information Processing Systems*, pp. 887–894, 1992.

[40] Y. H. Zweiri, J. F. Whidborne, and L. D. Seneviratne, "A three-term backpropagation algorithm," *Neurocomputing*, vol. 50, pp. 305–318, 2003.

[41] X. Yu, M. O. Efe, and O. Kaynak, "A general backpropagation algorithm for feedforward neural networks learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 13, no. 1, pp. 251–254, 2002.

[42] D. Z. Feng, W. X. Zheng, and Y. Jia, "Neural network learning algorithms for tracking minor subspace in high-dimensional data stream," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 16, no. 3, pp. 513–521, 2005.

[43] Y. Ichikawa and T. Sawa, "Neural network application for direct feedback controllers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 3, no. 2, pp. 224–231.

[44] Y. Li, Y. Li, F. Li, B. Zhao, and Q. Li, "The research of temperature compensation for thermopile sensor based on improved PSO-BP algorithm," *Mathematical Problems in Engineering*, vol. 2015, Article ID 854945, 6 pages, 2015.

[45] C. F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 2, pp. 997–1006, 2004.

[46] Y. Da and G. Xiurun, "An improved PSO-based ANN with simulated annealing technique," *Neurocomputing*, vol. 63, pp. 527–533, 2005.

[47] I. Roth and M. Margaliot, "Analysis of artificial neural network learning near temporary minima: a fuzzy logic approach," *Fuzzy Sets and Systems*, vol. 161, no. 19, pp. 2569–2584, 2010.

[48] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.

[49] A. Minai and R. Williams, "Back-propagation heuristics: a study of the extended delta-bar-delta algorithm," in *Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks*, vol. 1, pp. 595–600, San Diego, Calif, USA, 1990.

[50] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*, Prentice Hall, Upper Saddle River, NJ, USA, 1989.

[51] J. Moreno-Valenzuela and C. Aguilar-Avelar, *Motion Control of Underactuated Mechanical Systems*, Springer International Publishing AG, Cham, Switzerland, 2018.

[52] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1985.

[53] H. Shao and G. Zheng, "Convergence analysis of a back-propagation algorithm with adaptive momentum," *Neurocomputing*, vol. 74, no. 5, pp. 749–752, 2011.

[54] H. Shao, D. Xu, G. Zheng, and L. Liu, "Convergence of an online gradient method with inner-product penalty and adaptive momentum," *Neurocomputing*, vol. 77, no. 1, pp. 243–252, 2012.

[55] L. W. Chan and F. Fallside, "An adaptive training algorithm for back propagation networks," *Computer Speech and Language*, vol. 2, no. 3-4, pp. 205–218, 1987.

[56] W. C. Cohen, *Optimal Control Theory: An Introduction*, Dover, NY, USA, 2004.

[57] S. Li, J. Li, J. Qiu, H. Ji, and K. Zhu, "Control design for arbitrary complex nonlinear discrete-time systems based on direct NNMRAC strategy," *Journal of Process Control*, vol. 21, no. 1, pp. 103–110, 2011.

[58] H. L. Shu, "Analysis of PID neural network multivariable control systems," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 25, no. 1, pp. 105–111, 1999.

[59] H. Shu and Y. Pi, "PID neural networks for time-delay systems," *Computers & Chemical Engineering*, vol. 24, no. 2-7, pp. 859–862, 2000.