Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 267609, 11 pages
http://dx.doi.org/10.1155/2014/267609

Research Article

Adaptive Control of Nonlinear Discrete-Time Systems by Using OS-ELM Neural Networks

Xiao-Li Li, Chao Jia, De-xin Liu, and Da-wei Ding

School of Automation and Electrical Engineering and the Key Laboratory of Advanced Control of Iron and Steel Process (Ministry of Education), University of Science and Technology Beijing, Beijing 100083, China

Correspondence should be addressed to Xiao-Li Li; lixiaoli@hotmail.com

Received 11 April 2014; Accepted 23 April 2014; Published 22 May 2014

Academic Editor: Bo Shen

As a kind of novel feedforward neural network with single hidden layer, ELM (extreme learning machine) neural networks are studied for the identification and control of nonlinear dynamic systems. The property of simple structure and fast convergence of ELM can be shown clearly. In this paper, we are interested in adaptive control of nonlinear dynamic plants by using OS-ELM (online sequential extreme learning machine) neural networks. Based on data scope division, the problem that training process of ELM neural network is sensitive to the initial training data is also solved. According to the output range of the controlled plant, the data corresponding to this range will be used to initialize ELM. Furthermore, due to the drawback of conventional adaptive control, when the OS-ELM neural network is used for adaptive control of the system with jumping parameters, the topological structure of the neural network can be adjusted dynamically by using multiple model switching strategy, and an MMAC (multiple model adaptive control) will be used to improve the control performance. Simulation results are included to complement the theoretical results.

1. Introduction

Modeling and controlling of nonlinear systems are always the main research field in control theory and engineering [1–3]; great progresses in this field have been made in the past 30 years, but many problems still exist. In the actual industrial process, exact mathematical models are always very difficult to get even with the existence of huge numbers of input-output or state data. Therefore, data based control of nonlinear systems has been concerned especially in recent 10 years. Artificial neural network has been recognized as a powerful tool for the identification and control of nonlinear systems since it was applied to nonlinear system [4]. Numerous learning algorithms have been proposed for the training of the network, among which BP algorithm [5] and RBF algorithm are used very frequently. But it is generally known that the BP faces some bottlenecks such as trivial manual parameters tuning (leaning rate, learning epochs, etc.), local minima, and poor computational scalability. Recently, a new learning algorithm for single-hidden-layer feedforward neural networks (SLFNs) named extreme learning machine (ELM) has been proposed by Huang et al. [6, 7]. The essence of ELM is that input weights and bias of hidden neurons are generated randomly; the hidden layer of SLFNs need not be tuned [8]. Furthermore, based on least-square scheme, the smallest norm solution can be calculated analytically. Given that ELM algorithm is free of trivial iterative tuning and learning process is implemented through simple generalized inverse operation, it can achieve a much faster learning speed than other traditional algorithms.

The training of batch processing ELM algorithms can be realized only after all the training samples are ready. But in actual application, the training data may come one by one or chunk by Huang et al. [8]. Therefore, Liang et al. proposed the OS-ELM [9] algorithm which can adjust its output weight adaptively when new observations are received. Unlike the batch learning algorithms, as soon as the learning process for particular training data is complemented, the data will be discarded; thus it avoids retraining of the previous observations [8].

To the best of our knowledge, a large number of researches about ELM are focused on the regression and

classification [10–12]. There are only few of published results regarding identification and control of nonlinear dynamic systems. In this paper, our interest will be kept in the identification and control of nonlinear dynamic plants by using OS-ELM neural networks. The adaptive controller based on OS-ELM neural network model can be constructed. In simulation process, on one hand, since the training process of OS-ELM neural network is sensitive to the initial training data, different regions of the initial data can cause large diversity of control result. Thus, large numbers of initial data can be classified previously into multiple regions. Then, according to the range of expected output of the controlled plant, we can choose the corresponding initial data to initialize OS-ELM neural networks. On the other hand, multiple model switching strategy will be used for the adaptive control by using OS-ELM neural network when a system with jumping parameters is controlled. Based on the two aspects which are maintained in the above, OS-ELM algorithm is extended to the field of control.

The paper is organized as follows. The structure of ELM neural network and OS-ELM algorithm are introduced in Section 2. Then, adaptive control of nonlinear dynamic system by using OS-ELM is stated in Section 3. Considering that OS-ELM is sensitive to the initial training data, the relationship between control performance and the selection of the initial training data is discussed for OS-ELM neural network; furthermore, simulation results about OS-ELM control and multiple OS-ELM models control are also shown in Section 4. Finally some conclusions are drawn in Section 5.

2. ELM Neural Networks

Let us consider standard single-hidden-layer feedforward neural networks (SLFNs) with L hidden nodes and activation function $G(\mathbf{a}_i, b_i, \mathbf{x})$; the structure is shown in Figure 1. Given N arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \Re^n \times \Re^m$, then the network can be mathematically modeled as

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i \mathbf{h}_i(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\mathbf{a}_i, b_i, \mathbf{x}), \tag{1}$$

where $\mathbf{a}_i$ is the weight vector connecting the ith hidden neuron and the input neurons, $\boldsymbol{\beta}_i$ is the weight vector connecting the ith hidden neuron and the output neurons, b_i is the bias of the ith hidden neuron, and $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output function of the ith hidden node which is bounded.

To approximate these N samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \Re^n \times \Re^m$ with zero error is to search for a solution of $(\mathbf{a}_i, b_i)$ and $\boldsymbol{\beta}_i$ satisfying

$$f_L(\mathbf{x}_j) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j \quad j = 1, 2, \ldots, N. \tag{2}$$

The above N equation can be written compactly as

$$\mathbf{H}\beta = \mathbf{T}, \tag{3}$$

where $\mathbf{H}$ is the hidden-layer output matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \qquad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}. \tag{4}$$

The Essence of ELM. The hidden layer of ELM need not be iteratively tuned. According to feedforward neural network theory both the training error $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2$ and the norm of weights $\|\boldsymbol{\beta}\|$ need to be minimized [8].

2.1. Basic ELM. The hidden node parameters $(\mathbf{a}_i, b_i)$ remain fixed after being randomly generated. The training of a SLFNs is equivalent to find a least-square solution $\widehat{\beta}$ of the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$; that is,

$$\left\|\mathbf{H}\widehat{\boldsymbol{\beta}} - \mathbf{T}\right\| = \min_{\boldsymbol{\beta}} \left\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\right\|. \tag{5}$$

The smallest norm least-squares solution of the above linear system is

$$\widehat{\boldsymbol{\beta}} = \mathbf{H}^{+}\mathbf{T}, \tag{6}$$

where $\mathbf{H}^{+}$ can be obtained by $\mathbf{H}^{+} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$, if $\mathbf{H}^T\mathbf{H}$ is nonsingular, and $\mathbf{H}^{+} = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$, if $\mathbf{H}\mathbf{H}^T$ is nonsingular [8]. To summarize, the learning process can be demonstrated as the following.

Basic ELM Algorithm. One is given a training set

$\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \Re^n \times \Re^m$, output function $G(\mathbf{a}_i, b_i, \mathbf{x})$ and hidden node number L.

Step 1. Randomly assign hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \ldots, L$

Step 2. Calculate the hidden-layer output matrix $\mathbf{H}$.

Step 3. Calculate the output weight $\boldsymbol{\beta}$ using the equation $\boldsymbol{\beta} = \mathbf{H}^{+}\mathbf{T}$, where $\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \cdots & \mathbf{t}_N \end{bmatrix}^T$.

2.2. OS-ELM. As mentioned above in Section 2.1, one of the solutions of the output weight vector $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{7}$$

This algorithm just needs one step learning. Indeed, basic ELM algorithm is a batch learning algorithm. However, in actual application environment the training data may come one by one or chunk by Huang et al. [8]. Therefore, Liang et al. proposed the OS-ELM [9] algorithm which can adjust

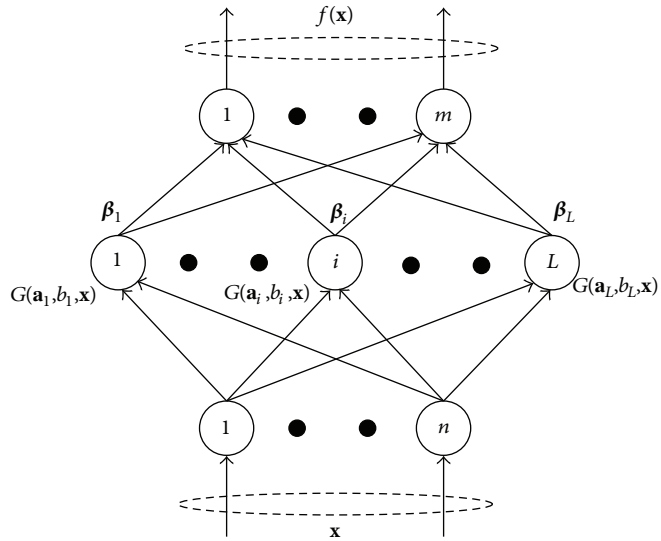


Figure 1: Single-hidden-layer feedforward networks.

its output weight $\boldsymbol{\beta}$ adaptively when new observations are received. The algorithm can be summarized as follows.

OS-ELM Algorithm. One is given a training set

$S = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \Re^n, \mathbf{t}_i \in \Re^m, i = 1, \ldots\}$, output function $G(\mathbf{a}_i, b_i, \mathbf{x})$, and hidden node number L.

Step 1 Initialization Phase. From the given training set S, a small chunk of training data $S_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$ is used to initialize the learning, where, $N_0 \geq L$.

(a) Assign random parameters of hidden nodes $(\mathbf{a}_i, b_i)$, where $i = 1, \ldots, L$.

(b) Calculate the initial hidden-layer output matrix $\mathbf{H}_0$.

(c) Calculate the initial output weight $\beta_0 = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0$, where $\mathbf{P}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$ and $\mathbf{T}_0 = [\mathbf{t}_1 \ \cdots \ \mathbf{t}_{N_0}]^T$.

(d) Set $k = 0$, where k is the number of chunks which is trained currently.

Step 2 Sequential Learning Phase. (a) Present the $(k+1)$th chunk of new observations $S_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^{k} N_j)+1}^{\sum_{j=0}^{k+1} N_j}$; here N_{k+1} denotes the number of observations in the $(k+1)$th chunk.

(b) Calculate the partial hidden-layer output matrix $\mathbf{H}_{k+1}$ for the $(k+1)$th chunk of data S_{k+1}:

$$\mathbf{H}_{k+1} = \begin{bmatrix} G\left(\mathbf{a}_1, b_1, \mathbf{x}_{(\sum_{j=0}^{k} N_j)+1}\right) & \cdots & G\left(\mathbf{a}_L, b_L, \mathbf{x}_{(\sum_{j=0}^{k} N_j)+1}\right) \\ \vdots & \cdots & \vdots \\ G\left(\mathbf{a}_1, b_1, \mathbf{x}_{\sum_{j=0}^{k+1} N_j}\right) & \cdots & G\left(\mathbf{a}_L, b_L, \mathbf{x}_{\sum_{j=0}^{k+1} N_j}\right) \end{bmatrix}. \tag{8}$$

Set $\mathbf{T}_{k+1} = [\mathbf{t}_{(\sum_{j=0}^{k} N_j)+1}, \ldots, \mathbf{t}_{\sum_{j=0}^{k+1} N_j}]^T$.

(c) Calculate the output weight

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T\right)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \beta_{k+1} &= \beta_k + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T \left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \beta_k\right). \end{aligned} \tag{9}$$

(d) Set $k = k + 1$. Go to Step 2(a).

It can be seen from the above OS-ELM algorithm that OS-ELM becomes the batch ELM when $N_0 = N$. For the detailed description, readers can refer to Liang et al. [9]. From (9), it can be seen that the OS-ELM algorithm is similar to recursive least-squares (RLS) in [13]. Hence, all the convergence results of RLS can be applied here.

3. Adaptive Control by Using OS-ELM Neural Networks

3.1. Identification. Consider the following SISO nonlinear discrete-time system:

$$y_{k+1} = f_0(\cdot) + g_0(\cdot)\, u_{k-d+1}, \tag{10}$$

where f_0 and g_0 are infinitely differentiable functions defined on a compact set of $F \subset \Re^{n+m}$, $G \subset \Re^{n+m}$. Consider

$$y_{k-n+1}, \ldots, y_k, u_{k-d-m+1}, \ldots, u_{k-d}. \tag{11}$$

y is the output, u is the input, $m \leq n$, d is the relative degree of the system, and g_0 is bounded away from zero. The arguments of f_0 and g_0 are real variables. In this paper, we consider a simplified form with $d = 1$. Then the SISO nonlinear discrete-time system can be described as

$$y_{k+1} = f_0[\mathbf{x}_k] + g_0[\mathbf{x}_k]\, u_k, \tag{12}$$

where $\mathbf{x}_k = [y_{k-n+1}, \ldots, y_k, u_{k-m}, \ldots, u_{k-1}]$. Let us first assume that there exist exact weights $\mathbf{w}^*$ and $\mathbf{v}^*$, the functions

$\widehat{f}[\mathbf{x}_k, \mathbf{w}^*]$ and $\widehat{g}[\mathbf{x}_k, \mathbf{v}^*]$ which can approximate the functions $f_0[\mathbf{x}_k]$ and $g_0[\mathbf{x}_k]$ without unmodelled dynamic. The nonlinear system (12) can be described by the neural networks as follows:

$$y_{k+1} = \widehat{f}\left[\mathbf{x}_k, \mathbf{w}^*\right] + \widehat{g}\left[\mathbf{x}_k, \mathbf{v}^*\right] u_k, \tag{13}$$

where the functions $\widehat{f}[\cdot,\cdot]$ and $\widehat{g}[\cdot,\cdot]$ can be selected as OS-ELM neural networks; that is,

$$\begin{aligned} \widehat{f}\left[\mathbf{x}_k, \mathbf{w}^*\right] &= \sum_{i=1}^{L} \mathbf{w}_i^* G\left(\mathbf{a}_i, b_i, \mathbf{x}_k\right) \\ \widehat{g}\left[\mathbf{x}_k, \mathbf{v}^*\right] &= \sum_{i=L}^{2L} \mathbf{v}_i^* G\left(\mathbf{a}_i, b_i, \mathbf{x}_k\right). \end{aligned} \tag{14}$$

$\mathbf{a}_i$, b_i can be generated randomly and kept in constant. Then (13) can be rewritten as

$$y_{k+1} = \mathbf{\Phi}_k \theta_0^*, \tag{15}$$

where $\mathbf{\Phi}_k = [G(\mathbf{a}_1, b_1, \mathbf{x}_k) \cdots G(\mathbf{a}_L, b_L, \mathbf{x}_k) G(\mathbf{a}_{L+1}, b_{L+1}, \mathbf{x}_k) u_k \cdots G(\mathbf{a}_{2L}, b_{2L}, \mathbf{x}_k) u_k]$, $\theta_0^* = \left[\mathbf{w}^* \quad \mathbf{v}^*\right]^T$.

Let $\mathbf{w}_k$ and $\mathbf{v}_k$ denote the estimates of $\mathbf{w}^*$ and $\mathbf{v}^*$ at time k. The neural network identification model can be shown as follows:

$$\widehat{y}_{k+1} = \widehat{f}\left[\mathbf{x}_k, \mathbf{w}_k\right] + \widehat{g}\left[\mathbf{x}_k, \mathbf{v}_k\right] u_k \tag{16}$$

that is,

$$\widehat{y}_{k+1} = \mathbf{\Phi}_k \widehat{\theta}_k, \tag{17}$$

where $\widehat{\theta}_k = \left[\begin{smallmatrix} \mathbf{w}_k \\ \mathbf{v}_k \end{smallmatrix}\right]$. Define e_{k+1}^* as

$$e_{k+1}^* = y_{k+1} - \widehat{y}_{k+1} = y_{k+1} - \mathbf{\Phi}_k \widehat{\theta}_k. \tag{18}$$

Referring to the OS-ELM algorithm, taking into account the existence of time-series in control system, the on-line observation data comes one by one; if we choose $N_j \equiv 1$, the updating ruler can be rewritten as

$$\begin{aligned} \mathbf{P}_k &= \mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1} \mathbf{\Phi}_k^T \mathbf{\Phi}_k \mathbf{P}_{k-1}}{1 + \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T} \\ \widehat{\theta}_{k+1} &= \widehat{\theta}_k + \mathbf{P}_k \mathbf{\Phi}_k^T e_{k+1}^*, \end{aligned} \tag{19}$$

where $\widehat{\theta}_0 = \theta_0$ and $\mathbf{P}_0 = (\mathbf{\Phi}_0^T \mathbf{\Phi}_0)^{-1}$ can be obtained in OS-ELM Step 1 Initialization Phase.

Due to the existence of unmodelling dynamics, the nonlinear system (15) can be represented as

$$y_{k+1} = \mathbf{\Phi}_k \theta_0^* + \Delta_f, \tag{20}$$

where Δ_f is the model error and satisfies $\sup |\Delta_f| \le \varepsilon$. Then a deed-zone algorithm will be used:

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \frac{\sigma_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T \mathbf{\Phi}_k \mathbf{P}_{k-1}}{1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T} \tag{21}$$

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \sigma_k \mathbf{P}_k \mathbf{\Phi}_k^T e_{k+1}^*, \tag{22}$$

where

$$\sigma_k = \begin{cases} 1 & \dfrac{\left|e_{k+1}^*\right|^2}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} > \varepsilon^2 \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Theorem 1. *Given the system described by* (20)*, if the updating ruler described by* (21)–(23)*, we have the following results:*

(i)

$$\left\|\theta_0^* - \widehat{\theta}_k\right\|^2 \le \left\|\theta_0^* - \widehat{\theta}_{k-1}\right\|^2 \le \left\|\theta_0^* - \widehat{\theta}_1\right\|^2, \tag{24}$$

(ii)

$$\lim_{N \to \infty} \sum_{k=1}^{N} \sigma_k \left[\frac{e_{k+1}^{*2}}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} - \varepsilon^2\right] < \infty \tag{25}$$

and this implies

(a)

$$\lim_{k \to \infty} \sigma_k \left[\frac{e_{k+1}^{*2}}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} - \varepsilon^2\right] = 0, \tag{26}$$

(b)

$$\lim_{k \to \infty} \sup \frac{e_{k+1}^{*2}}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} \le \varepsilon^2. \tag{27}$$

For the detailed proof of Theorem 1, readers can refer to [14]; it can also be found in the appendix.

3.2. Adaptive Control Using OS-ELM. In this section we discuss the adaptive control problem using OS-ELM neural networks. For the system described as

$$y_{k+1} = f_0\left[\mathbf{x}_k\right] + g_0\left[\mathbf{x}_k\right] u_k, \tag{28}$$

OS-ELM neural network identification model as

$$\widehat{y}_{k+1} = \widehat{f}\left[\mathbf{x}_k, \mathbf{w}_k\right] + \widehat{g}\left[\mathbf{x}_k, \mathbf{v}_k\right] u_k, \tag{29}$$

control u_k can be given as below:

$$u_k = \frac{-\widehat{f}\left[\mathbf{x}_k, \mathbf{w}_k\right] + r_{k+1}}{\widehat{g}\left[\mathbf{x}_k, \mathbf{v}_k\right]}, \tag{30}$$

where r_{k+1} appears as desired output. The control u_k is applied to both the plant and the neural network model.

3.2.1. OS-ELM Adaptive Control Algorithm

Step 1 Initialization Phase. Given a random input sequence u_k, where $k = 1, \ldots, N_0$, from the controlled plant (12), we can get a small chunk of off-line training data $S_0 = \{(\mathbf{x}_i, y_{i+1})\}_{i=1}^{N_0} \subset \mathfrak{R}^n \times \mathfrak{R}^1$. Given output function $G(\mathbf{a}_i, b_i, \mathbf{x})$ and hidden node number $2L$ which are used to initialize the learning, where $N_0 \ge 2L$.

(a) Assign random parameters of hidden nodes $(\mathbf{a}_i, b_i)$, where $i = 1, \ldots, 2L$.

(b) Calculate the initial hidden-layer output matrix $\mathbf{\Phi}_0$:

$$\mathbf{\Phi}_0 = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_{N_0} \end{bmatrix}_{N_0 \times 2L}, \tag{31}$$

where $\phi_i = [G(\mathbf{a}_1, b_1, \mathbf{x}_i) \cdots G(\mathbf{a}_1, b_1, \mathbf{x}_i) G(\mathbf{a}_{L+1}, b_{L+1}, \mathbf{x}_i) u_i \cdots G(\mathbf{a}_{2L}, b_{2L}, \mathbf{x}_i) u_i], \ i = 1, \ldots, N_0$.

(c) Calculate the initial output weight as follows:

$$\theta_0 = \mathbf{P}_0 \mathbf{\Phi}_0^T \mathbf{Y}_0, \tag{32}$$

where $\mathbf{P}_0 = \left(\mathbf{\Phi}_0^T \ \mathbf{\Phi}_0\right)^{-1}$, $\theta_0 = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{v}_0 \end{bmatrix}$, and $\mathbf{Y}_0 = \left[y_2 \ \cdots \ y_{N_0+1}\right]^T$.

(d) Set $k = N_0 + 1$.

Step 2 Adaptive Control Phase. (a) Calculate the control u_k and the output of the plant y_{k+1}:

$$u_k = \frac{-\widehat{f}\left[\mathbf{x}_k, \mathbf{w}_k\right] + r_{k+1}}{\widehat{g}\left[\mathbf{x}_k, \mathbf{v}_k\right]}$$
$$y_{k+1} = f_0\left[\mathbf{x}_k\right] + g_0\left[\mathbf{x}_k\right] u_k, \tag{33}$$

where $\widehat{\theta}_{N_0+1} = \begin{bmatrix} \mathbf{w}_{N_0+1} \\ \mathbf{v}_{N_0+1} \end{bmatrix} \triangleq \theta_0$, $\mathbf{x}_k = [y_{k-n+1}, \ldots, y_k, u_{k-m}, \ldots, u_{k-1}]$.

(b) Calculate the partial hidden-layer output matrix $\mathbf{\Phi}_k$ for the kth observation data:

$$\mathbf{\Phi}_k = \left[G\left(\mathbf{a}_1, b_1, \mathbf{x}_k\right) \ \cdots \ G\left(\mathbf{a}_L, b_L, \mathbf{x}_k\right) \ G\left(\mathbf{a}_{L+1}, b_{L+1}, \mathbf{x}_k\right) u_k \ \cdots \ G\left(\mathbf{a}_{2L}, b_{2L}, \mathbf{x}_k\right) u_k\right]. \tag{34}$$

(c) Calculate the output weight

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \frac{\sigma_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T \mathbf{\Phi}_k \mathbf{P}_{k-1}}{1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T}$$
$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \sigma_k \mathbf{P}_k \Phi_k^T e_{k+1}^*, \tag{35}$$

where $\mathbf{P}_{N_0} \triangleq \mathbf{P}_0$

$$\sigma_k = \begin{cases} 1 & \dfrac{\left|e_{k+1}^*\right|^2}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} > \varepsilon^2 \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

(d) Set $k = k + 1$. Go to Step 2(a).

3.3. Stability Analysis. Define the error between the neural network identification state and the reference model state as

$$\widehat{e}_{k+1} = \widehat{y}_{k+1} - r_{k+1}; \tag{37}$$

that is,

$$\widehat{e}_{k+1} = \widehat{f}\left[\mathbf{x}_k, \mathbf{w}_k\right] + \widehat{g}\left[\mathbf{x}_k, \mathbf{v}_k\right] u_k - r_{k+1}. \tag{38}$$

By (30), we have

$$\widehat{e}_{k+1} = r_{k+1} - r_{k+1} = 0. \tag{39}$$

The control error is defined as

$$e_{k+1} = y_{k+1} - r_{k+1}; \tag{40}$$

that is,

$$e_{k+1} = y_{k+1} - \widehat{y}_{k+1} + \widehat{y}_{k+1} - r_{k+1}, \tag{41}$$

so,

$$e_{k+1} = e_{k+1}^* + \widehat{e}_{k+1} = e_{k+1}^*. \tag{42}$$

Using the properties of the parameter estimation algorithm stated in Theorem 1, we immediately have

$$\lim_{k \to \infty} \sigma_k \left[\frac{e_{k+1}^2}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} - \varepsilon^2\right] = 0. \tag{43}$$

Because the activation function $G(\mathbf{a}_i, b_i, \mathbf{x})$ is bounded, then $\widehat{f}$ and $\widehat{g}$ are bounded. If u_k is chosen as (30), then u_k is bounded; hence $\mathbf{\Phi}_k$ is bounded. From (43) we have

$$\lim_{k \to \infty} \sup \frac{e_{k+1}^2}{\left(1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T\right)^2} \leq \varepsilon^2. \tag{44}$$

Considering that $\mathbf{\Phi}_k$ is bounded, then e_{k+1}^2 is bounded. From (40), $y_{k+1} = e_{k+1} + r_{k+1}$, r_k is bounded; then y_{k+1} is bounded too.

To sum up, we find that OS-ELM neural networks have the capability of identification of and controling nonlinear systems. However, as we know, the conventional adaptive control systems are usually based on a fixed or slowly adaptive model. It cannot react quickly to abrupt changes and will result in large transient errors before convergence. In this case, MMAC algorithm is presented as a useful tool. We can construct multiple OS-ELM neural networks to cover the uncertainty of the parameters of the plant by initializing OS-ELM neural network, respectively, in different position. Meanwhile, an effective index function can be used to select the optimal identification model and the optimal controller. MMAC based on OS-ELM algorithm can improve the control property of the system greatly and avoid the influence of jumping parameter on the plant.

3.4. Multiple Model Adaptive Control. Multiple model adaptive control can be regarded as an extension of conventional indirect adaptive control. The control system contains M identification models, denoted by I_l, $l \in \{1, 2, \ldots, M\}$. According to (16) and (30), multiple models set can be established as the following:

$$I_l : \widehat{y}_{k+1}^l = \widehat{f}^l\left[\mathbf{x}_k, \mathbf{w}_k^l\right] + \widehat{g}^l\left[\mathbf{x}_k, \mathbf{v}_k^l\right] u_k^l, \tag{45}$$

where

$$\begin{aligned} \widehat{f}^l\left[\mathbf{x}_k, \mathbf{w}_k^l\right] &= \sum_{i=1}^{L} w_i^l G\left(\mathbf{a}_i, b_i, \mathbf{x}_k\right) \\ \widehat{g}^l\left[\mathbf{x}_k, \mathbf{v}_k^l\right] &= \sum_{i=L}^{2L} v_i^l G\left(\mathbf{a}_i, b_i, \mathbf{x}_k\right). \end{aligned} \tag{46}$$

The control u_k^l can be rewritten as

$$u_k^l = \frac{-\widehat{f}^l\left[\mathbf{x}_k, \mathbf{w}_k^l\right] + r_{k+1}}{\widehat{g}^l\left[\mathbf{x}_k, \mathbf{v}_k^l\right]}. \tag{47}$$

Define $e_{k+1}^{*l} \triangleq y_{k+1} - \widehat{y}_{k+1}^l$. In any instant, one of the models I_l is selected by a switching rule, and the corresponding control input is used to control the plant. Referring to Theorem 1 the index function has the form

$$J_k^l = \sum_{\tau=1}^{k} \sigma_\tau \left[\frac{e_{\tau+1}^{*2}}{\left(1 + \sigma_\tau \mathbf{\Phi}_\tau \mathbf{P}_{\tau-1} \mathbf{\Phi}_\tau^T\right)^2} - \varepsilon^2 \right]. \tag{48}$$

At every sample time, the model that corresponds to the minimum J_k^l is chosen; that is, I_i is chosen if

$$J_k^i = \min_l J_k^l \tag{49}$$

and u_k^i is used as the control input at that instant. The structure of multiple models adaptive control can be shown in Figure 2.

Referring to paper [15], Narendra and Xiang established 4 kinds of structure of MMAC. (1) All models are fixed; (2) all models are adaptive; (3) $(M-1)$ fixed models and one adaptive model; (4) $(M-2)$ fixed models, one free running adaptive model, and one reinitialized adaptive model. We can get a similar result of the stability. For the detailed proof of stability of MMAC, the reader can refer to [15–17].

4. Simulation Results

4.1. Adaptive Control. In this section, we present results of simulations of adaptive control nonlinear discrete-time systems by using OS-ELM neural networks. The nonlinear systems will be considered as below:

$$y_{k+1} = \frac{y_k * (1 - 0.5 * F) + 0.5 * F}{1 + y_k^2} - 0.5 * \left(1 + y_k\right) u_k, \tag{50}$$

where we define $f[y_k] = (y_k * (1 - 0.5 * F) + 0.5 * F)/(1 + y_k^2)$ and $g[y_k] = -0.5 * (1 + y_k)$; $F = 0.1$ is a scalar. The control goal is to force the system states to track a reference model trajectory. We select the reference model as

$$r_k = 0.2 * \sin\left(\frac{2\pi k}{100}\right). \tag{51}$$

The single-input/single-output nonlinear discrete-time system can be modeled by

$$\widehat{y}_{k+1} = \widehat{f}\left[y_k, \mathbf{w}_k\right] + \widehat{g}\left[y_k, \mathbf{v}_k\right] u_k, \tag{52}$$

where $\widehat{f}$ and $\widehat{g}$ are OS-ELM neural networks. The neural networks $\widehat{f}$ and $\widehat{g}$ have the same structure $\aleph_{1,50,1}$. Based on the OS-ELM adaptive control algorithm, the parameters of the neural networks $\mathbf{w}_k$ and $\mathbf{v}_k$ are updated to $\mathbf{w}_{k+1}$ and $\mathbf{v}_{k+1}$ using OS-ELM Algorithm.

For OS-ELM control algorithm, in Step 1, we choose $N_0 = 300$; the control u_k is selected randomly from $[0, 0.2]$. A small chunk of off-line training data is used to calculate the initial output weights $\mathbf{w}_0$ and $\mathbf{v}_0$. Since ELM algorithm just adjusts the output weight, it shows fast and accurate identification results.

Following this, in Step 2, both identification and control are implemented. The response of the controlled system with a reference input $r_k = \sin(2\pi k/100)$ by using OS-ELM algorithm is shown in Figure 3. We can see that the controlled system can track the reference model rapidly. The control error is shown in Figure 4.

4.2. Choice of the Data Set. In simulation process, we find that OS-ELM is sensitive to the initial training data. Initial training data determine the initial value of adaptive control directly. When we select different initial training samples, the performance of the control results varies greatly. Figure 5 shows the control results using OS-ELM neural networks with different initial training samples.

In Figure 6 we select 6 different ranges of random input-output pair. We can find that different ranges of input values and the corresponding output values concentrate in a single area. According to the output ranges of the controlled plant, we can choose a specific data area to train the neural network. In Figure 5, the output range of the controlled system is in the $[-0.2, 0.2]$. If we adopt $[0, 0.2]$ random input (red area in Figure 6(b), initializing OS-ELM by using this set data), the control result is shown in Figure 5(b). On the contrary, if the random input in the other area (Green, Blue, yellow area, and so on in Figure 6(b)) is used to initialize OS-ELM, the control result is poor (Figure 5(a)).

On one hand, in the actual production process, we have a large number of field data which is distributed over a large range. On the other hand, we generally know the objective output range of the controlled plant. Then, according to the output range of the controlled plant, we can choose a specific data area to train the neural network. In this way, the accuracy of control can be improved and the computation time can be saved.

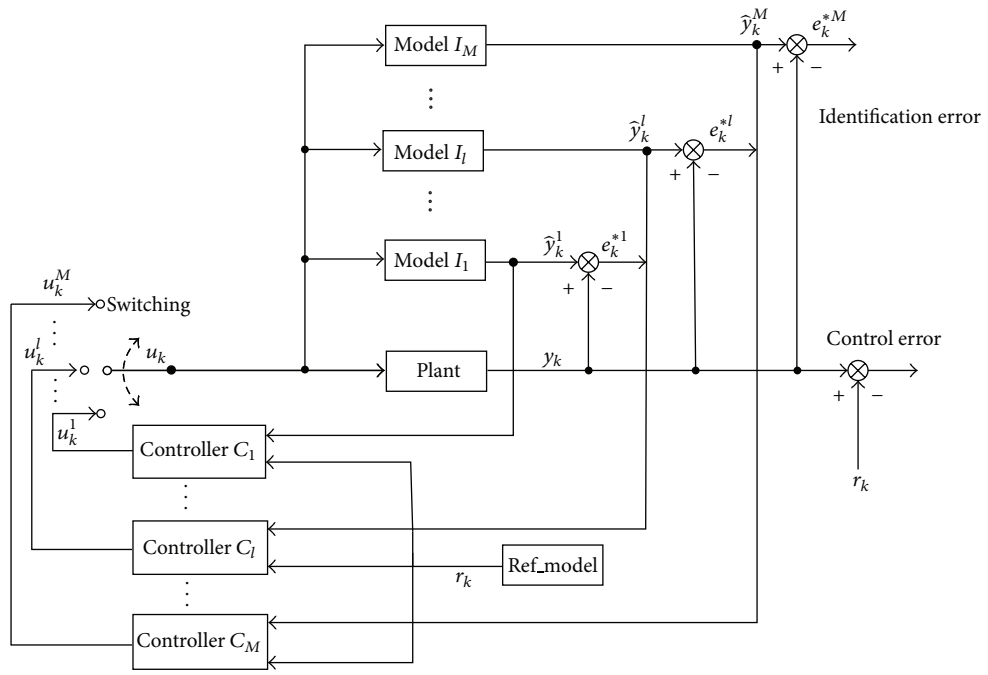


Figure 2: Structure of multiple models adaptive control.

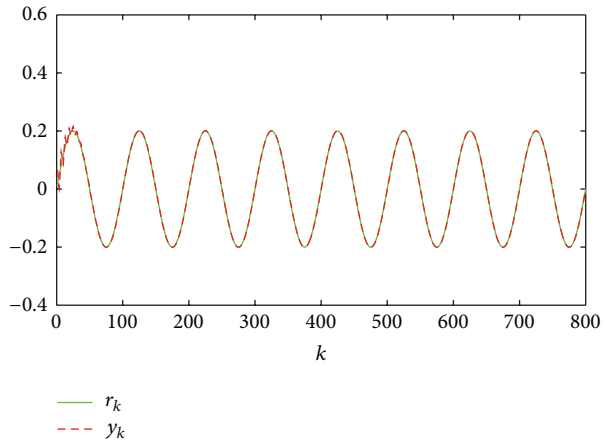


Figure 3: Nonlinear systems adaptive control by using OS-ELM.

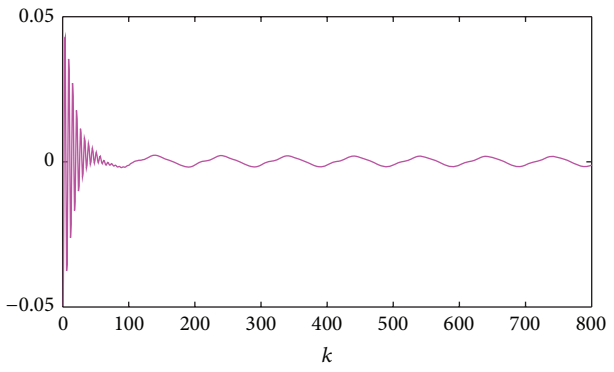


Figure 4: Control error.

4.3. Multiple Models Adaptive Control. From the above simulation result, we find that OS-ELM neural networks show perfect capability of identification. But from the OS-ELM adaptive control algorithm we can see that it contains two steps (initialization phase and adaptive control phase). This algorithm just shows the perfect capability of identification with fixed or slowly time-varying parameters. For the system with jumping parameters, once the change of parameters happens, the OS-ELM adaptive control algorithm needs to implement Step 1 again or the control performance will be very poor. To solve this kind of problem, MMAC (multiple models adaptive control) will be presented as a very useful tool.

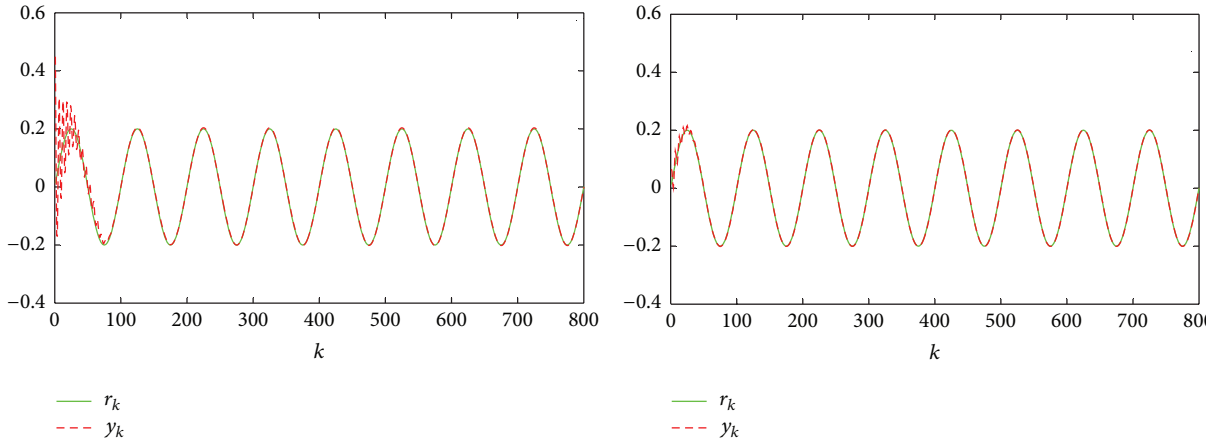
Consider the controlled plant as follows:

$$y_{k+1} = \frac{y_k * (1 - 0.5 * F) + 0.5 * F}{1 + y_k^2} - 0.5 * (1 + y_k) u_k, \tag{53}$$

where

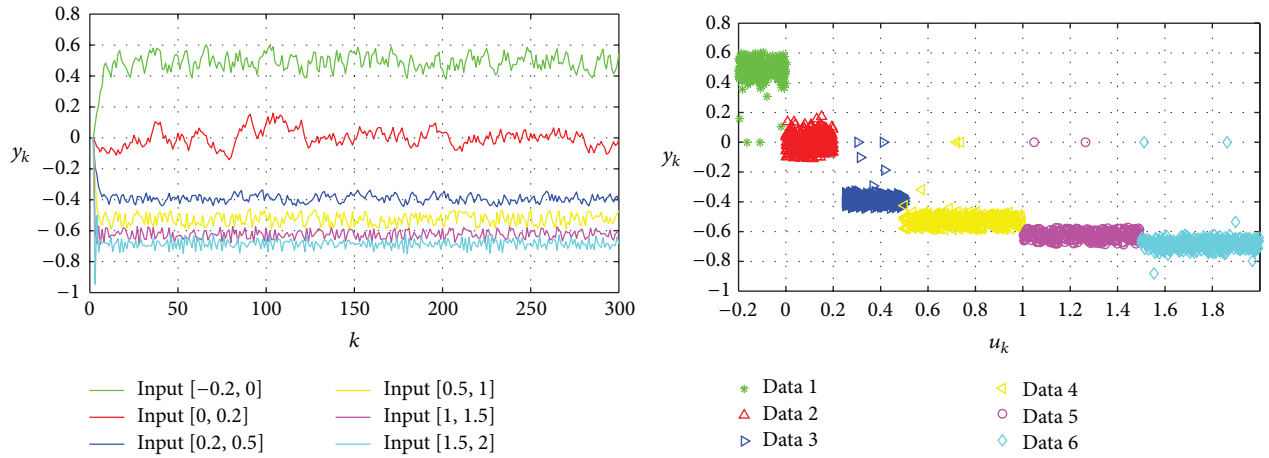
$$F = \begin{cases} 0.1, & 1 \le k < 500 \\ 0.15, & 500 \le k < 1000 \\ 0.355, & 1000 \le k \le 1500. \end{cases} \tag{54}$$

We choose the index function as (48) and construct 4 OS-ELM neural network models. Among them, the weights of $1^{\#}$ model are reinitialized ones; $2^{\#}, 3^{\#}$ models are fixed models which are formed according to the environment of the changing parameters. $4^{\#}$ is free running adaptive model.



(a) $(-0.2, 0)$ random input (b) $(0, 0.2)$ random input

FIGURE 5: OS-ELM control results with different initial training samples.



(a) (b)

FIGURE 6: Data set. (a) Output results with different input data. (b) Different ranges of input values and the corresponding output values concentrate in a single area.

In any sample time, $1^{\#}$ model chooses the optimal model from the models set which is formed by models ($2^{\#}$, $3^{\#}$, and $4^{\#}$ model). Figure 7 shows the response of multiple models and signal model.

In Figure 7(a), we can see that signal model adaptive control shows poor control quality for the system with jumping parameters. Once the parameters change abruptly, the neural network model needs to adjust its weight which is far from the true value. Figure 7(c) shows the bad control input. In MMAC, the reinitialized adaptive model can initiate the weights by choosing the best weights of a set of models based on the past performance of the plant. At every sample time, based on the index function, one of the best models in model set is selected; the parameters of reinitialized model can be updated from the paraments of this model. From Figure 7(b), we can see that MMAC can improve the control quality dramatically compared to Figure 7(a). Figure 7(d) shows the control sequence and Figure 7(e) shows the switching procedure of controlling MMAC.

5. Conclusion

The application of OS-ELM in the identification and control of nonlinear dynamic system is studied carefully. OS-ELM neural network can improve the accuracy of identification and the control quality dramatically. Since the training process of OS-ELM neural network is sensitive to the initial training data, different scope of training data can be used to initialize OS-ELM neural network based on the output range of the system. The topological structure of OS-ELM neural network can be adjusted dynamically by using multiple model switching strategy, when it is used to control systems with jumping parameters. Simulation results show that MMAC

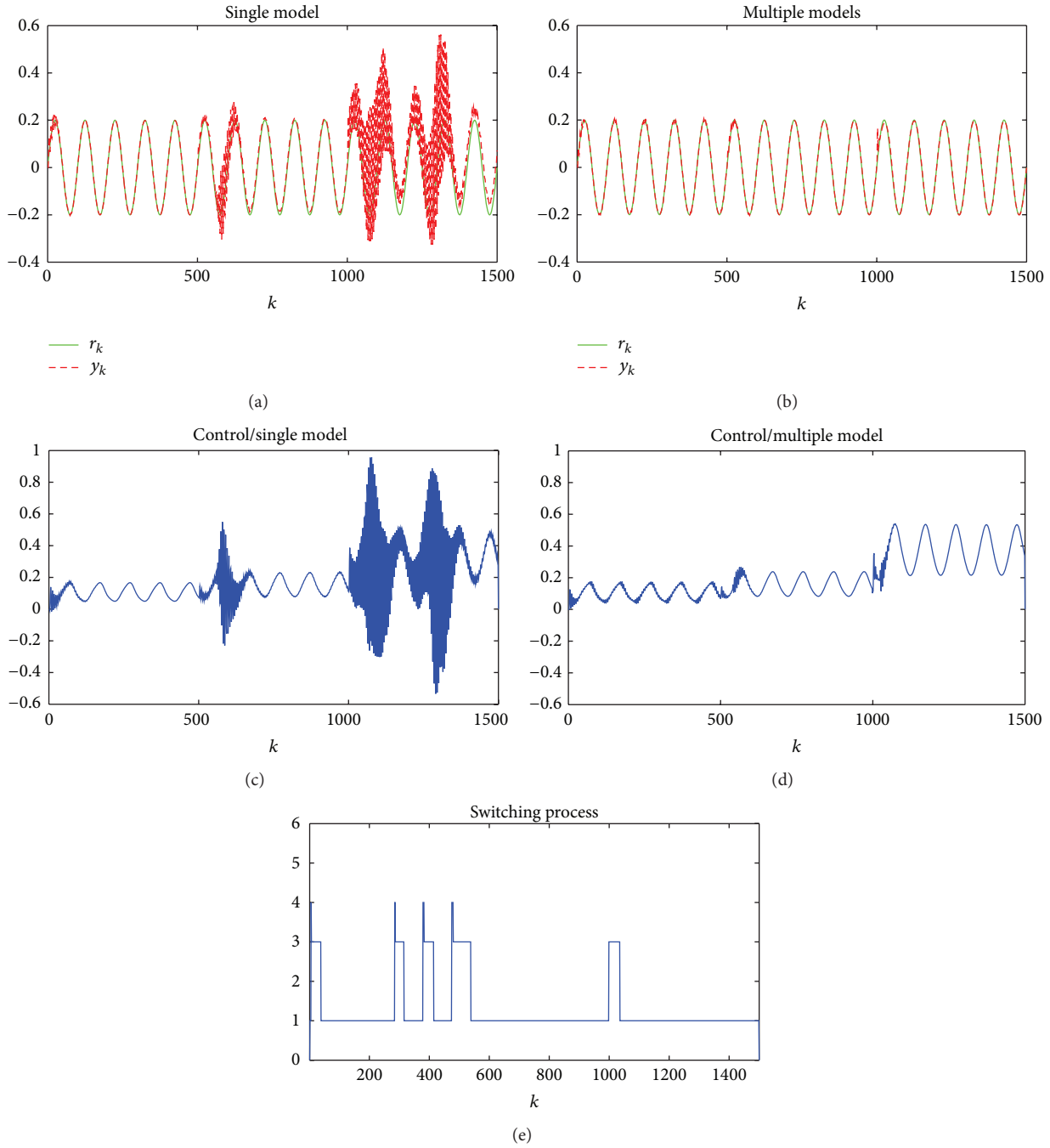


Figure 7: Multiple OS-ELM neural network models adaptive control.

based on OS-ELM which is presented in this paper can improve the control performance dramatically.

Appendix

Lemma A.1 (matrix inversion lemma, see [18]). *If*

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k-1}^{-1} + \mathbf{\Phi}_k^T \mathbf{\Phi}_k \sigma_k, \tag{A.1}$$

where the scalar $\sigma_k > 0$, *then* $\mathbf{P}_k$ *is related to* $\mathbf{P}_{k-1}$ *via*

$$\mathbf{P}_k = \mathbf{P}_{k-1} - \frac{\sigma_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T \mathbf{\Phi}_k \mathbf{P}_{k-1}}{1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T}. \tag{A.2}$$

Also,

$$\mathbf{P}_k \mathbf{\Phi}_k^T = \frac{\mathbf{P}_{k-1} \mathbf{\Phi}_k^T}{1 + \sigma_k \mathbf{\Phi}_k \mathbf{P}_{k-1} \mathbf{\Phi}_k^T}. \tag{A.3}$$

Proof of Theorem 1. Defining $\beta_k \triangleq 1/(1+\sigma_k \boldsymbol{\Phi}_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T)$, then from (20), (22), and (A.3) we have

$$\widetilde{\theta}_{k+1} = \widetilde{\theta}_k - \sigma_k \beta_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T e_{k+1}^*, \tag{A.4}$$

where $\widetilde{\theta}_{k+1} \triangleq \theta_0^* - \widehat{\theta}_{k+1}$. Then, from (A.4) we can obtain

$$\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} + \Delta_f = \beta_k e_{k+1}^*. \tag{A.5}$$

Introducing Lyapunov function as $V_{k+1} = \widetilde{\theta}_{k+1}^T \mathbf{P}_k^{-1} \widetilde{\theta}_{k+1}$, taking (A.1) into it, hence, we can obtain

$$\begin{aligned} V_{k+1} &= \widetilde{\theta}_{k+1}^T \mathbf{P}_{k-1}^{-1} \widetilde{\theta}_{k+1} + \widetilde{\theta}_{k+1}^T \boldsymbol{\Phi}_k^T \boldsymbol{\Phi}_k \sigma_k \widetilde{\theta}_{k+1} \\ &= \left[\widetilde{\theta}_k - \sigma_k \beta_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T e_{k+1}^*\right]^T \mathbf{P}_{k-1}^{-1} \left[\widetilde{\theta}_k - \sigma_k \beta_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T e_{k+1}^*\right] \\ &\quad + \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1}\right)^2 \\ &= \widetilde{\theta}_k^T \mathbf{P}_{k-1}^{-1} \widetilde{\theta}_k - 2\sigma_k \boldsymbol{\Phi}_k \widetilde{\theta}_k \beta_k e_{k+1}^* \\ &\quad + \sigma_k^2 \beta_k^2 e_{k+1}^{*2} \boldsymbol{\Phi}_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T + \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1}\right)^2. \end{aligned} \tag{A.6}$$

Submitting (A.5) in the above equation,

$$\begin{aligned} V_{k+1} &= V_k - 2\sigma_k \boldsymbol{\Phi}_k \widetilde{\theta}_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} + \Delta_f\right) \\ &\quad + \sigma_k^2 \boldsymbol{\Phi}_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} + \Delta_f\right)^2 + \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1}\right)^2. \end{aligned} \tag{A.7}$$

Referring to [14], we can obtain

$$\begin{aligned} V_{k+1} &= V_k - \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1}\right)^2 - 2\sigma_k \boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} \Delta_f \\ &\quad - \sigma_k^2 \boldsymbol{\Phi}_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} + \Delta_f\right)^2 \\ &\le V_k - \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1}\right)^2 - 2\sigma_k \boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} \Delta_f; \end{aligned} \tag{A.8}$$

that is,

$$\begin{aligned} V_{k+1} &\le V_k - \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1}\right)^2 - 2\sigma_k \boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} \Delta_f \\ &= V_k + \sigma_k \Delta_f^2 - \sigma_k \left(\boldsymbol{\Phi}_k \widetilde{\theta}_{k+1} + \Delta_f\right)^2 \\ &= V_k + \sigma_k \Delta_f^2 - \sigma_k \beta_k^2 e_{k+1}^{*2} \\ &= V_k - \sigma_k \left(\beta_k^2 e_{k+1}^{*2} - \Delta_f^2\right). \end{aligned} \tag{A.9}$$

Finally, if we choose σ_k as (36), we have

$$V_{k+1} \le V_k - \sigma_k \left(\beta_k^2 e_{k+1}^{*2} - \varepsilon^2\right); \tag{A.10}$$

that is,

$$V_{k+1} - V_k \le -\sigma_k \left(\beta_k^2 e_{k+1}^{*2} - \varepsilon^2\right) \le 0. \tag{A.11}$$

V_k is nonnegative; then

$$\widetilde{\theta}_{k+1}^T \mathbf{P}_k^{-1} \widetilde{\theta}_{k+1} \le \widetilde{\theta}_k^T \mathbf{P}_{k-1}^{-1} \widetilde{\theta}_k \le \widetilde{\theta}_1^T \mathbf{P}_0^{-1} \widetilde{\theta}_1. \tag{A.12}$$

Now from the matrix inversion lemma, it follows that

$$\lambda_{\min}\left(\mathbf{P}_k^{-1}\right) \ge \lambda_{\min}\left(\mathbf{P}_{k-1}^{-1}\right) \ge \lambda_{\min}\left(\mathbf{P}_0^{-1}\right); \tag{A.13}$$

this implies

$$\begin{aligned} \lambda_{\min}\left(\mathbf{P}_0^{-1}\right) \left\|\widetilde{\theta}_{k+1}\right\|^2 &\le \lambda_{\min}\left(\mathbf{P}_k^{-1}\right) \left\|\widetilde{\theta}_{k+1}\right\|^2 \le \widetilde{\theta}_{k+1}^T \mathbf{P}_k^{-1} \widetilde{\theta}_{k+1} \\ &\le \widetilde{\theta}_1^T \mathbf{P}_0^{-1} \widetilde{\theta}_1 \le \lambda_{\min}\left(\mathbf{P}_0^{-1}\right) \left\|\widetilde{\theta}_1\right\|^2. \end{aligned} \tag{A.14}$$

This establishes part (i).

Summing both sides of (A.11) from 1 to N with V_k being nonnegative, we have

$$V_{k+1} \le V_1 - \lim_{N\to\infty} \sum_{k=1}^{N} \sigma_k \left(\beta_k^2 e_{k+1}^{*2} - \varepsilon^2\right); \tag{A.15}$$

we immediately get

$$\lim_{N\to\infty} \sum_{k=1}^{N} \sigma_k \left[\frac{e_{k+1}^{*2}}{\left(1+\sigma_k \boldsymbol{\Phi}_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T\right)^2} - \varepsilon^2\right] < \infty. \tag{A.16}$$

(ii) holds. Then

$$\lim_{k\to\infty} \sigma_k \left[\frac{e_{k+1}^{*2}}{\left(1+\sigma_k \boldsymbol{\Phi}_k \mathbf{P}_{k-1} \boldsymbol{\Phi}_k^T\right)^2} - \varepsilon^2\right] = 0. \tag{A.17}$$

(a) and (b) hold. □

Conflict of Interests

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

Acknowledgments

This work is partially supported by the Fund of National Natural Science Foundation of China (Grant no. 61104013), Program for New Century Excellent Talents in Universities (NCET-11-0578), the Fundamental Research Funds for the Central Universities (FRF-TP-12-005B), and Specialized Research Fund for the Doctoral Program of Higher Education (20130006110008).

References

[1] C. Yang, S. S. Ge, and T. H. Lee, "Output feedback adaptive control of a class of nonlinear discrete-time systems with unknown control directions," *Automatica*, vol. 45, no. 1, pp. 270–276, 2009.

[2] X.-S. Wang, C.-Y. Su, and H. Hong, "Robust adaptive control of a class of nonlinear systems with unknown dead-zone," *Automatica*, vol. 40, no. 3, pp. 407–413, 2004.

[3] M. Chen, S. S. Ge, and B. Ren, "Adaptive tracking control of uncertain MIMO nonlinear systems with input constraints," *Automatica*, vol. 47, no. 3, pp. 452–465, 2011.

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

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[6] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 985–990, July 2004.

[7] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[8] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[9] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[10] S. Suresh, R. Venkatesh Babu, and H. J. Kim, "No-reference image quality assessment using modified extreme learning machine classifier," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 541–552, 2009.

[11] A. A. Mohammed, R. Minhas, Q. M. Jonathan Wu, and M. A. Sid-Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2588–2597, 2011.

[12] F. Benot, M. Van Heeswijk, Y. Miche et al., "Feature selection for nonlinear models with extreme learning machines," *Neurocomputing*, vol. 102, pp. 111–124, 2013.

[13] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*, Wiley, New York, NY, USA, 2013.

[14] H. Jiang, "Directly adaptive fuzzy control of discrete-time chaotic systems by least squares algorithm with dead-zone," *Nonlinear Dynamics*, vol. 62, no. 3, pp. 553–559, 2010.

[15] K. S. Narendra and C. Xiang, "Adaptive control of discrete-time systems using multiple models," *Institute of Electrical and Electronics Engineers: Transactions on Automatic Control*, vol. 45, no. 9, pp. 1669–1686, 2000.

[16] X.-L. Li, D.-X. Liu, J.-Y. Li, and D.-W. Ding, "Robust adaptive control for nonlinear discrete-time systems by using multiple models," *Mathematical Problems in Engineering*, vol. 2013, Article ID 679039, 10 pages, 2013.

[17] X. L. Li, C. Jia, D. X. Liu et al., "Nonlinear adaptive control using multiple models and dynamic neural networks," *Neurocomputing*, vol. 136, pp. 190–200, 2014.

[18] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control*, Prentice-Hall, 1984.