Neural Comput & Applic (2013) 22:577–586
DOI 10.1007/s00521-011-0805-1

Direct adaptive neural control of nonlinear systems with extreme learning machine

Hai-Jun Rong · Guang-She Zhao

Received: 7 August 2011 / Accepted: 29 December 2011 / Published online: 19 January 2012

Abstract A direct adaptive neural control scheme for a class of nonlinear systems is presented in the paper. The proposed control scheme incorporates a neural controller and a sliding mode controller. The neural controller is constructed based on the approximation capability of the single-hidden layer feedforward network (SLFN). The sliding mode controller is built to compensate for the modeling error of SLFN and system uncertainties. In the designed neural controller, its hidden node parameters are modified using the recently proposed neural algorithm named extreme learning machine (ELM), where they are assigned random values. However, different from the original ELM algorithm, the output weight is updated based on the Lyapunov synthesis approach to guarantee the stability of the overall control system. The proposed adaptive neural controller is finally applied to control the inverted pendulum system with two different reference trajectories. The simulation results demonstrate good tracking performance of the proposed control scheme.

Keywords Nonlinear systems · Single-hidden layer feedforward network · Extreme learning machine · Sliding mode controller

1 Introduction

The application of neural networks to control problems of nonlinear systems has been an active research area and attracted the interest of many control researchers. The motivation is that neural networks possess an inherent structure suitable for learning and reconstructing complex nonlinear mappings. In the neural control design, neural networks are used to approximate the unknown nonlinearities where the nonlinear relationship between the input and output data is represented by the parameters of the networks, known as weights. The principal types of neural networks in most of the neural network–based control schemes are the feedforward neural network (FNN) with sigmoidal activation function and the radial basis function neural network (RBFNN) with radial basis function [1]. The adaptive neural control problem is generally viewed as deriving the adaptation laws for the parameters existing in the neural networks. For FNN, the adjustment parameters are the weights connecting different nodes, while in RBFNN the used parameters are the output weights, the center and width of the RBFs. The mostly commonly used adjustment law for these parameters is the back-propagation and its modifications [14]. Although the gradient descent–based learning methods have achieved many practical successes in the neural control of nonlinear systems, these methods easily converge to local minima if the cost function is not convex with respect to its parameters, thus failing in complex adaptive situations. In addition, using these methods, neural control lacks the formal synthesis techniques that can guarantee global stability required by the control systems.

To solve the problem, the Lyapunov stability theory is generally synthesized in the neural control design, so that the stable adaptive neural controllers are constructed. In [1, 13], the stable adaptive controllers are proposed based on the FNN, where the stable adaptive laws are derived using the Lyapunov function. When the FNN is used to build the controller, its weights appear nonlinearly in the error

H.-J. Rong (✉) · G.-S. Zhao
MOE Key Laboratory of Strength and Vibration, School of Aerospace, Xi'an Jiaotong University, Shaanxi 710049, China
e-mail: hjrong@mail.xjtu.edu.cn

model. Thus, the Taylor's series expansion about the optimal parameter values is adopted for achieving the function approximation error equation with a linearly parameterizable form by ignoring the high-order arguments. In this case, a large approximation error may be induced and further result in the degraded control performance. Some stable adaptive controllers using RBFNN have been developed in [3, 15, 12]. In these schemes, the centers and widths of RBFs are determined according to some prior information regarding the system to be approximated. Then, the output weights are updated using the stable adaptive laws. This method is simple to implement since the weights appear linearly and stable adaptive laws can be generated in a linear fashion. However, the problem with these methods is that some systems to be approximated are time-varying and their prior information is difficult to obtain, in which the exact values for the centers and widths of the RBF neurons are hard to determine. The use of inaccurate centers and widths usually results in a deterioration of the performance. In [18, 19], the RBFNN is modeled with a nonlinearly parameterized system by updating the centers and widths together with the output weights and then used to develop the stable adaptive neural control schemes. Although these schemes with nonlinearly parameterized RBFNN approximators demonstrate superior performance than those with linearly parameterized RBFNN approximators, a high computational complexity will be caused due to a large number of tunable parameters in these schemes. This further limits the applications of the RBFNN in the context of control design for complex nonlinear dynamical systems.

In the paper, a novel direct adaptive neural control scheme for a class of nonlinear systems is introduced. The neural controller is built based on a single-hidden layer feedforward network (SLFN) with additive hidden nodes (i.e., Sigmoid units) or RBF hidden nodes. Different from the existing work that utilizes the gradient descent–based learning methods, the proposed neural controller is adjusted using the recently proposed neural algorithm referred to as extreme learning machine (ELM) [6–8, 10]. ELM algorithm possesses better computational efficiency in terms of the learning speed and generalization capability compared with the back-propagation algorithm. This has been verified in these latest work [9, 11, 20, 23, 24]. In ELM, parameters of the hidden nodes(e.g., the additive nodes and the RBF nodes) need not be adjusted during training. All the parameters of ELM could randomly be generated according to any given continuous probability distribution without any prior knowledge about the target function. All the hidden node parameters in ELM are not only independent from each other but also independent from the training data. This can be attributed to the SLFN being a linear system. When such a SLFN is applied in the complex nonlinear control problems, the Taylor's series expansion used by FNN and RBFNN for the nonlinearly parameterized system can be avoided and also a less computation process can be achieved by only adjusting the output weights.

Hence, the ELM algorithm is utilized to train the proposed neural controller by randomly assigning the parameters of hidden nodes. However, different from the original ELM algorithm, the output weights are updated using the stable adaptive laws derived from a Lyapunov function for guaranteeing the stability of the closed-loop nonlinear system. And also a sliding controller is incorporated into the neural controller and activated to work with the neural controller for offsetting the modeling errors brought by the SLFN and system disturbances. The proposed adaptive neural controller is finally evaluated on the inverted pendulum system for tracking two different reference trajectories. The simulation results clarify the proposed approach and illustrate that the output tracking error between the plant output and the desired reference output can asymptotically converge to zero.

This paper is organized as follows. Section 2 gives a brief review of the ELM. In Sect. 3, the design procedure of the adaptive neural controller is introduced together with the dynamic model of a class of nonlinear systems under consideration and the stable adaptive laws for adjusting the output weights based on a Lyapunov synthesis approach. Section 4 shows the simulation results by controlling the inverted pendulum system. Section 5 presents the conclusions from this study.

2 Description of ELM

This section briefly reviews the ELM to provide the necessary background for the development of neural controller. A brief mathematical description of SLFN incorporating both additive and RBF hidden nodes in a unified way is given first.

2.1 Mathematical description of unified SLFN

The output of a SLFN with $\tilde{N}$ hidden nodes (additive or RBF nodes) can be represented by

$$f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i G(\mathbf{x}; \mathbf{c}_i, a_i), \quad \mathbf{x} \in \mathbf{R}^n, \ \mathbf{c}_i \in \mathbf{R}^n \tag{1}$$

where $\mathbf{c}_i$ and a_i are the learning parameters of hidden nodes, $\boldsymbol{\beta}_i$ is the output weight connecting the ith hidden node to the output node, and $G(\mathbf{x}; \mathbf{c}_i, a_i)$ is the output of the ith hidden node with respect to the input $\mathbf{x}$. For additive hidden nodes with the Sigmoid or threshold activation function $g(x)$: $R \rightarrow R$, $G(\mathbf{x}; \mathbf{c}_i, a_i)$ is given by

$$G(\mathbf{x}; \mathbf{c}_i, a_i) = g(\mathbf{c}_i \cdot \mathbf{x} + a_i), \quad a_i \in R \tag{2}$$

where $\mathbf{c}_i$ is the weight vector connecting the input layer to the ith hidden node, a_i is the bias of the ith hidden node, and $\mathbf{c}_i \cdot \mathbf{x}$ denotes the inner product of vectors $\mathbf{c}_i$ and $\mathbf{x}$ in $\mathbf{R}^n$.

For RBF hidden nodes with the Gaussian or triangular activation function $g(x)$: $R \rightarrow R$, $G(\mathbf{x};\, \mathbf{c}_i,\, a_i)$ is given by

$$G(\mathbf{x}; \mathbf{c}_i, a_i) = g(a_i \|\mathbf{x} - \mathbf{c}_i\|), \quad a_i \in R^+ \tag{3}$$

where $\mathbf{c}_i$ and a_i are the center and impact factor of the ith RBF node. R^+ indicates the set of all positive real values. The RBF network is a special case of SLFN with RBF nodes in its hidden layer. Each RBF node has its own centroid and impact factor, and its output is given by a radially symmetric function of the distance between the input and the center.

2.2 ELM

For N arbitrary distinct samples $(\mathbf{x}_k, \mathbf{t}_k) \in \mathbf{R}^n \times \mathbf{R}^m$, if a SLFN with $\tilde{N}$ hidden nodes can approximate these N samples with zero error, it then implies that there exist $\boldsymbol{\beta}_i, \mathbf{c}_i$ and a_i such that

$$\sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i G(\mathbf{x}_k; \mathbf{c}_i, a_i) = \mathbf{t}_k, \quad k = 1, \ldots, N. \tag{4}$$

Equation (4) can be written compactly as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{5}$$

where

$$\mathbf{H}(\mathbf{c}_1, \ldots, \mathbf{c}_{\tilde{N}}, a_1, \ldots, a_{\tilde{N}}, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{x}_1; \mathbf{c}_1, a_1) & \ldots & G(\mathbf{x}_1; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \ldots & \vdots \\ G(\mathbf{x}_N; \mathbf{c}_1, a_1) & \ldots & G(\mathbf{x}_N; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{6}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}^T_{N \times m} \tag{7}$$

$\mathbf{H}$ is called the hidden layer output matrix of the network [4, 8]; the ith column of $\mathbf{H}$ is the ith hidden node's output vector with respect to inputs $\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_N$ and the kth row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\mathbf{x}_k$.

The ELM algorithm has the following property [16, p. 491]: For a SLFN with additive or RBF hidden nodes and an activation function $g(x)$: $R \rightarrow R$ which is infinitely differentiable in any interval, there exists $\tilde{N} \leq N$ such that for N arbitrary distinct input vectors $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{R}^n, \quad i = 1, \ldots, \tilde{N}\}$, for any $\{(\mathbf{c}_i, a_i)\}_{i=1}^{\tilde{N}}$ randomly generated according to any continuous probability distribution $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\| < \epsilon$ with probability one, where $\epsilon > 0$ is a small positive value.

3 Design procedure of the adaptive neural controller

The paper focuses on the design of adaptive controller for a class of nonlinear systems that is expressed in the following form:

$$\begin{cases} x^{(n)} = f(\mathbf{x}) + g(\mathbf{x})u + d \\ y = x \end{cases} \tag{8}$$

where u and y are the control input and system output, respectively. $\mathbf{x}\,(= [x, \dot{x}, \ldots, x^{(n-1)}]^T \in \mathbf{R}^n)$ is the state vector of the system and assumed to be available for measurement. $f(\mathbf{x})$ is assumed to be unknown and bounded real continuous nonlinear functions of dynamic system. The input gain $g(\mathbf{x})$ is assumed to be known and bounded. Also for the accomplishment of control task, the function $g(\mathbf{x})$ is required to be nonsingular for all $\mathbf{x} \in \mathbf{R}^n$ and satisfy $g(\mathbf{x}) > 0$. d is an unknown function representing system uncertainties and external disturbances. It is also assumed that d is bounded and for some d_0 satisfies $|d| \leq d_0$.

Letting $x_1 = x, \quad x_2 = \dot{x} = \dot{x}_1, \quad x_3 = \ddot{x} = \dot{x}_2, \ldots,$ and $x_n = x^{(n-1)} = \dot{x}_{(n-1)}$, (8) becomes as

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \vdots \\ \dot{x}_{(n-1)} = x_n \\ \dot{x}_n = f(\mathbf{x}) + g(\mathbf{x})u + d \\ y = x_1 \end{cases} \tag{9}$$

where $\mathbf{x} = [x_1, x_2, ..., x_n]^T \in \mathbf{R}^n$. Equation (9) can be further written in the following state-space form:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}f(\mathbf{x}) + \mathbf{b}g(\mathbf{x})u + \mathbf{b}d \\ y = \mathbf{C}^T\mathbf{x} \end{cases} \tag{10}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & \mathbf{I}_{n-1} \\ 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{11}$$

Define the output tracking error $e = y_d - x$, the reference output vector $\mathbf{y}_d = [y_d, \dot{y}_d, \ldots, y_d^{(n-1)}]^T$, the tracking error vector $\mathbf{e} = [e, \dot{e}, \cdots e^{(n-1)}]^T = [e_1, e_2, \ldots, e_n]^T$. The control objective of designing the neural controller is to force the state $\mathbf{x} = [x, \dot{x}, \ldots, x^{n-1}]^T$ to follow the specified desired trajectory $\mathbf{y}_d = [y_d, \dot{y}_d, \ldots, y_d^{n-1}]^T$ as closely as possible.

According to feedback linearization method [18] and dynamics system given by (10), the desired control input u^* can be obtained as follows:

$$u^* = \frac{1}{g(\mathbf{x})}[y_d^{(n)} - f(\mathbf{x}) - d + \mathbf{k}^T\mathbf{e}] \qquad (12)$$

where $\mathbf{k}$ $(=[k_n, k_{n-1},\ldots, k_1]^T \in \mathbf{R}^n)$ is the real number vector. By substituting (12) into (10), the following equation is obtained

$$e^n + k_1 e^{(n-1)} + \cdots + k_n e = 0 \qquad (13)$$

By properly choosing k_i ($i = 1, 2, \ldots, n$), all roots of the polynomial $s^n + k_1 s^{(n-1)} + \cdots + k_n = 0$ are in the open left half plane, which implies that the tracking error will converge to zero. Equation (13) also shows that the tracking of reference trajectory is asymptotically achieved from any initial conditions, i.e., $\lim_{t\to\infty}|\mathbf{e}| = 0$.

Since the function $f(\mathbf{x})$ and d are unknown, the equivalence control law of (12) cannot be implemented. To circumvent this problem, the SLFN is utilized to approximate the equivalence control law, which will be described in the following section.

3.1 Adaptive neural controller

The proposed direct adaptive neural control scheme is illustrated in Fig. 1, where two controllers are incorporated, viz, a neural controller u_n and a sliding controller u_s. The neural controller u_n is connected in parallel with the sliding mode controller u_s to generate an overall control signal u.

Thus, the overall control law u is the summation of the two control signals,

$$u = u_n + u_s \qquad (14)$$

The neural controller is built based on the SLFN to approximate the perfect control law represented by the unknown function $f(\mathbf{x})$, which is given by

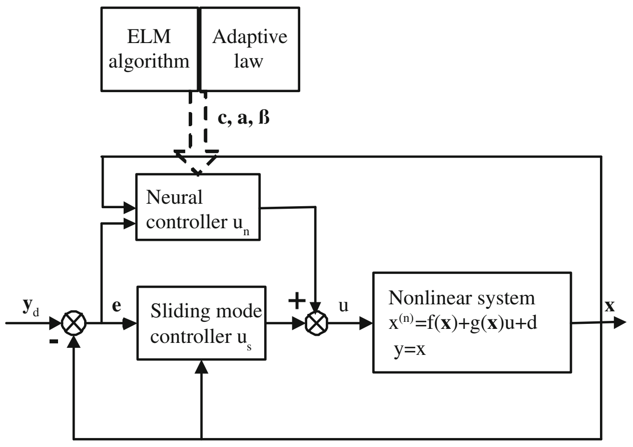

Fig. 1 The proposed adaptive neural control scheme

$$u_n^* = \frac{1}{g(\mathbf{x})}[y_d^{(n)} - f(\mathbf{x}) + \mathbf{k}^T\mathbf{e}] \qquad (15)$$

Besides, external disturbances and unmodeled dynamics represented by d are unknown and also the approximation error between the ideal model and the approximated model will arise when the SLFN is utilized to approximate the equivalence control law of (15). The sliding mode controller u_s is used to augment the equivalence controller in order to compensate for the approximation error and the system disturbances.

When the SLFN is utilized to approximate the designed nonlinear controller, the hidden node parameters (input weights and biases or centers and impact factors) of the SLFN are determined based on the ELM. Thus according to the property of ELM, these parameters need not be tuned during training and may simply be assigned with random values. In this case, (5) becomes a linear system. Then the neural control law u_n of (15) is given by

$$u_n = \mathbf{H}(\mathbf{z}; \mathbf{c}, \mathbf{a})\boldsymbol{\beta} \qquad (16)$$

where $\mathbf{z} = [y_d^n; \mathbf{x}; \mathbf{e}]^T \in \Re^{2n+1}$ is the input vector.

For fixed hidden node parameters ($\{\mathbf{c}, \mathbf{a}\}$), training a SLFN is simply equivalent to finding a least-square solution of the output weights $\boldsymbol{\beta}$ in (5). According to the universal approximation theorem of a SLFN [5], there exists the optimal output weight parameters $\boldsymbol{\beta}^*$ to approximate the control law, which is defined as

$$\boldsymbol{\beta}^* \overset{\Delta}{=} \arg\min\left[\sup_{\mathbf{z}\in M_z}\|u_n^* - u_n\|\right] \qquad (17)$$

such that

$$u_n^* = \mathbf{H}(\mathbf{z})\boldsymbol{\beta}^* + \epsilon_n(\mathbf{z}) \qquad (18)$$

where $\|\cdot\|$ represents the two-norm of a vector, and M_z is the predefined compact set of input vector $\mathbf{z}$. Huang et al [8] also have proved that when the number of hidden nodes $\tilde{N}$ is equal to the number of distinct training samples N, the SLFN can approximate these training samples with zero error. However, in most cases, the number of hidden nodes is much less than the number of distinct training samples, thus the approximation error ϵ arises. But with the number of hidden nodes increasing, the inherent approximation error ϵ can be reduced arbitrarily. Therefore, it is reasonable to assume that ϵ_n caused by approximating the control input is bounded with the constant ϵ_N, which is given by

$$|\epsilon_n(\mathbf{z})| \le \epsilon_N \qquad (19)$$

Based on (10), (14) and (18), the system tracking error can be shown to be

$$\dot{\mathbf{e}} = \boldsymbol{\Lambda}\mathbf{e} + \mathbf{B}\left[u_n^* - u_n - u_s - \frac{d}{g(\mathbf{x})}\right] \qquad (20)$$

where

$$\mathbf{\Lambda} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -k_n & -k_{n-1} & -k_{n-2} & \cdots & -k_1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ g(\mathbf{x}) \end{bmatrix} \tag{21}$$

According to (16) and (18), the system error of (20) can further be expressed as

$$\dot{\mathbf{e}} = \mathbf{\Lambda e} + \mathbf{B}\left[\mathbf{H}(\mathbf{z})\tilde{\boldsymbol{\beta}} + \epsilon_n(\mathbf{z}) - u_s - \frac{d}{g(\mathbf{x})}\right] \tag{22}$$

where $\tilde{\boldsymbol{\beta}}(= \boldsymbol{\beta}^* - \boldsymbol{\beta})$ is the parameter error. The problem of asymptotic tracking now becomes the problem of adapting the parameter $\boldsymbol{\beta}$ in order to make the approximation error as small as possible. Different from the original ELM algorithm, the output weight $\boldsymbol{\beta}$ is adjusted based on the adaptive law derived using Lyapunov second method, in which the stability of the entire control system is satisfied. The adaptive law of updating the parameter $\boldsymbol{\beta}$ by satisfying the stability of the entire system is illustrated in the following section.

3.2 Stable adaptive law

To derive the stable tuning law, consider the following Lyapunov function candidate:

$$V = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{1}{2\eta}\tilde{\boldsymbol{\beta}}^T\tilde{\boldsymbol{\beta}} \tag{23}$$

where $\mathbf{P}$ is the symmetric and positive definite matrix, and η is a positive constant that will appear in the adaptation law and is referred to as the learning rate.

Using (22), the derivative of the Lyapunov function is given as

$$\begin{aligned} \dot{V} &= \frac{1}{2}[\dot{\mathbf{e}}^T\mathbf{P}\mathbf{e} + \mathbf{e}^T\mathbf{P}\dot{\mathbf{e}}] + \frac{1}{\eta}\dot{\tilde{\boldsymbol{\beta}}}^T\tilde{\boldsymbol{\beta}} \\ &= \frac{1}{2}\mathbf{e}^T(\mathbf{\Lambda}^T\mathbf{P} + \mathbf{P\Lambda})\mathbf{e} + \left(\mathbf{e}^T\mathbf{PBH} + \frac{1}{\eta}\tilde{\boldsymbol{\beta}}^T\right)\boldsymbol{\beta} \\ &\quad - \mathbf{e}^T\mathbf{PB}\left(\frac{d}{g(\mathbf{x})} - \epsilon_n(\mathbf{z})\right) - \mathbf{e}^T\mathbf{PB}u_s \\ &= -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + \left(\mathbf{e}^T\mathbf{PBH} + \frac{1}{\eta}\dot{\tilde{\boldsymbol{\beta}}}^T\right)\boldsymbol{\beta} \\ &\quad - \mathbf{e}^T\mathbf{PB}\left(\frac{d}{g(\mathbf{x})} - \epsilon_n(\mathbf{z})\right) - \mathbf{e}^T\mathbf{PB}u_s \end{aligned} \tag{24}$$

where $\mathbf{J} = -(\mathbf{\Lambda}^T\mathbf{P} + \mathbf{P\Lambda})$. $\mathbf{J}$ is a symmetric definite matrix and is selected by the user.

To eliminate $\tilde{\boldsymbol{\beta}}$ from $\dot{V}$, the adaptive law can be selected as

$$\dot{\tilde{\boldsymbol{\beta}}}^T = -\eta\mathbf{e}^T\mathbf{PBH} \tag{25}$$

Equivalently, since $\dot{\tilde{\boldsymbol{\beta}}} = -\dot{\boldsymbol{\beta}}$, the tuning rule of the output weight $\boldsymbol{\beta}$ is given by

$$\dot{\boldsymbol{\beta}}^T = \eta\mathbf{e}^T\mathbf{PBH} \tag{26}$$

It is noteworthy that the tuning rule of the output weight $\boldsymbol{\beta}$ in the original ELM is achieved from the least-square error method. This is offline batch learning process where the training data are processed simultaneously. But in the proposed control scheme, the adaptive law for the output weight $\boldsymbol{\beta}$ is derived using Lyapunov second method to guarantee the stability of the entire control system. The adjustment of the output weight $\boldsymbol{\beta}$ is online sequential learning process where the training data arrive one-by-one. Although the modification of the output weights is different, the hidden node parameters (input weights and biases or centers and impact factors) of the SLFN are randomly determined like the original ELM algorithm.

Equation (24) becomes

$$\dot{V} = -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} - \mathbf{e}^T\mathbf{PB}\left(\frac{d}{g(\mathbf{x})} - \epsilon_n(\mathbf{z})\right) - \mathbf{e}^T\mathbf{PB}u_s \tag{27}$$

To make (27) less than or equal to zero, the sliding mode control term u_s is chosen as

$$u_s = \left(\frac{d_0}{g(\mathbf{x})} + \epsilon_N\right)\operatorname{sgn}(\mathbf{e}^T\mathbf{PB}) \tag{28}$$

where

$$\operatorname{sgn}(\mathbf{e}^T\mathbf{PB}) = \begin{cases} 1 & \mathbf{e}^T\mathbf{PB} > 0 \\ -1 & \mathbf{e}^T\mathbf{PB} < 0 \end{cases} \tag{29}$$

Hence, (27) can be written as follows:

$$\begin{aligned} \dot{V} \le & -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + |\mathbf{e}^T\mathbf{PB}|\left(\frac{|d|}{g(\mathbf{x})} + |\epsilon_n(\mathbf{z})|\right) \\ & - \mathbf{e}^T\mathbf{PB}\left(\left(\frac{d_0}{g(\mathbf{x})} + \epsilon_N\right)\operatorname{sgn}(\mathbf{e}^T\mathbf{PB})\right) \end{aligned} \tag{30}$$

Since $|\mathbf{e}^T\mathbf{PB}| = (\mathbf{e}^T\mathbf{PB})\operatorname{sgn}(\mathbf{e}^T\mathbf{PB})$, substituting (19) into (30) yields

$$\begin{aligned} \dot{V} \le & -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + |\mathbf{e}^T\mathbf{PB}|\left(\frac{|d|}{g(\mathbf{x})} - \frac{d_0}{g(\mathbf{x})}\right) \\ & + |\mathbf{e}^T\mathbf{PB}|(|\epsilon_n(\mathbf{z})| - \epsilon_N) \end{aligned} \tag{31}$$

Notice that

$$\begin{aligned} & |\mathbf{e}^T\mathbf{PB}|\left(\frac{|d|}{g(\mathbf{x})} - \frac{d_0}{g(\mathbf{x})}\right) \le 0 \\ & |\mathbf{e}^T\mathbf{PB}|(|\epsilon_n(\mathbf{z})| - \epsilon_N) \le 0 \end{aligned} \tag{32}$$

$$\dot{V} \leq -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} \leq 0 \tag{33}$$

The above equation illustrates that $\dot{V}$ is negative semidefinite and the overall control scheme is guaranteed to be stable. Using Barbalat's lemma [18, 21], it can be seen that as $t \to \infty, \mathbf{e} \to 0$. This further shows that the tracking error of the system will converge to zero.

Generally, the system reference signals are bounded and thus the control signals (u_n, u_s) must be required to be bounded. Equation (28)shows that the control signal u_s is bounded, but the bound of the control signal u_n need to be guaranteed by the bound of the parameter $\boldsymbol{\beta}$. In order to avoid the problem that the parameter $\boldsymbol{\beta}$ may drift to infinity over time, the projection algorithm [18] is applied by limiting its upper bound. The constraint set for $\boldsymbol{\beta}$ is defined as follows:

$$\Upsilon \triangleq \{\|\boldsymbol{\beta}\| \leq \boldsymbol{\beta}_0\} \tag{34}$$

where β_0 is the positive constant. Using the projection algorithm, the parameter adjustment law becomes as

$$\dot{\boldsymbol{\beta}}^T = \begin{cases} & \text{if} \quad \|\boldsymbol{\beta}\| < \boldsymbol{\beta}_0 \quad \text{or} \\ \eta\mathbf{e}^T\mathbf{PBH} & \text{if} \quad \|\boldsymbol{\beta}\| = \boldsymbol{\beta}_0 \\ & \text{and} \quad \mathbf{e}^T\mathbf{PBH}\boldsymbol{\beta} \leq 0 \\ \Pr\{\eta\mathbf{e}^T\mathbf{PBH}\} & \text{if} \quad \|\boldsymbol{\beta}\| = \boldsymbol{\beta}_0 \\ & \text{and} \quad \mathbf{e}^T\mathbf{PBH}\boldsymbol{\beta} > 0 \end{cases} \tag{35}$$

where the projection operator $\Pr\{*\}$ is given as

$$\Pr\{\eta\mathbf{e}^T\mathbf{PBH}\} = \eta\mathbf{e}^T\mathbf{PBH} - \eta\mathbf{e}^T\mathbf{PB}\frac{\boldsymbol{\beta}^T\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|^2}\mathbf{H} \tag{36}$$

Besides, it should be noted that the presence of sgn(.) function in the sliding control term u_s will introduce undesired chattering which may excite high-frequency dynamics. To solve the chattering problem, many mechanism such as the boundary layer [16], the saturation function [17] and differentiable approximation [2] have been developed for smoothing the sgn(.). Here, the saturation function [17] is utilized to replace the sgn(.) function for avoiding the undesired chattering. Thus, (28) is changed as

$$u_s = \left(\frac{d_0}{g(\mathbf{x})} + \epsilon_N\right) \text{sat}\left(\frac{\mathbf{e}^T\mathbf{PB}}{v}\right) \tag{37}$$

where v is an arbitrary small positive constant denoting the width of the boundary layer. The sat(.) function is given by

$$\text{sat}\left(\frac{\mathbf{e}^T\mathbf{PB}}{v}\right) = \begin{cases} \frac{\mathbf{e}^T\mathbf{PB}}{v} & \left|\frac{\mathbf{e}^T\mathbf{PB}}{v}\right| \leq 1 \\ \text{sgn}\left(\frac{\mathbf{e}^T\mathbf{PB}}{v}\right) & \left|\frac{\mathbf{e}^T\mathbf{PB}}{v}\right| > 1 \end{cases} \tag{38}$$

With this replacement, the sliding surface function $\mathbf{e}^T\mathbf{PB}$ with an arbitrary initial value will reach and stay within the boundary layer $\mathbf{e}^T\mathbf{PB} \leq v$.

3.3 Design procedure of direct adaptive neural controller

The design steps about the proposed direct adaptive neural controller are summarized as follows:

Step (1) Specify the design parameters

- Specify $k_1, \ldots, k_n$ such that all roots of the polynomial $s^n + k_1 s^{(n-1)} + \cdots + k_n = 0$ are in the open left half plane.
- Specify a positive definite matrix $\mathbf{J}$ and obtain a symmetric matrix $\mathbf{P}$ by solving the equation $\mathbf{J} = -(\boldsymbol{\Lambda}^T\mathbf{P} + \mathbf{P}\boldsymbol{\Lambda})$.
- Specify the parameter d_0, the modeling error magnitude ϵ_N, the positive constant η, the width of boundary layer v and the number of hidden nodes $\tilde{N}$ for the practical problem.

Step(2) Construct the neural controller using ELM algorithm

- Assign random hidden node parameters $(\mathbf{c}_i, a_i)$, $i = 1, \ldots, \tilde{N}$.
- Calculate the hidden node output vector $\mathbf{H}$ for input $\mathbf{z}$:

$$\mathbf{H} = \mathbf{H}(\mathbf{c}_1, \ldots, \mathbf{c}_{\tilde{N}}, a_1, \ldots, a_{\tilde{N}}; \mathbf{z}) \tag{39}$$

  where the details of $\mathbf{H}$ are referred in [4, 8].
- Construct the neural control input u_n using (16).

Step (3) Adapt the output weight parameter

- Apply the feedback control input $u = u_n + u_s$ to the dynamic system expressed by (8), where u_n is the neural control input given by (16) and u_s is the sliding mode control input given by (28).
- Use (35) to adjust the output weight parameter $\boldsymbol{\beta}$.

In the following section, two simulation examples will be performed to evaluate the proposed neural controller.

4 Simulations

In this section, the inverted pendulum system [22] is utilized to evaluate the performance of the proposed direct adaptive neural controller. Moreover, both the Gaussian RBF activation function $G(\mathbf{x}, \mathbf{c}, a) = \exp(-\|\mathbf{x} - \mathbf{c}\|^2/a)$ and the Sigmoidal additive activation function $G(\mathbf{x}, \mathbf{c}, a) = 1/(1 + \exp(-(\mathbf{c} \cdot \mathbf{x} + a)))$ have been used in the simulations to evaluate the proposed controller.

Example 1 Consider the following inverted pendulum system:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{g\sin x_1 - \frac{mlx_2^2\cos x_1 \sin x_1}{m_c+m}}{l\left(\frac{4}{3} - \frac{m\cos^2 x_1}{m_c+m}\right)} + \frac{\frac{\cos x_1}{m_c+m}}{l\left(\frac{4}{3} - \frac{m\cos^2 x_1}{m_c+m}\right)}u + d \tag{40}$$

where $g = 9.8$ m/s^2, $m_c = 1$ kg, $m = 0.1$ kg and $l = 0.5$ m. In the example, the reference trajectory is $y_d(t) = (0, 0)^T$ and the initial value of the state vector (x_1, x_2) is $(-\pi/9, 0)^T$. d represents the system uncertainty and is given by $0.2\sin(4\pi t)$. In the simulation, the controller design parameters are chosen as follows: $k_1 = 8$, $k_2 = 16$, $d_0 = 0.2$, $J = \mathrm{diag}(10, 10)$, $\eta = 5$, the modeling error magnitude $\epsilon_N = 0.01$, the width of the boundary layer $v = 0.01$, the number of hidden nodes $\tilde{N} = 10$. The hidden node parameters $(\mathbf{c}, a)$ are chosen randomly in the intervals $[-1, 1]$ and $[0, 1]$, respectively.

The simulation results of state x_1 and x_2 from the Sigmoidal additive and Gaussian RBF activation functions are illustrated in Figs. 2 and 3, where the reference signals are included. From the two figures, it can be found that the difference between the system output and the reference output is large at the beginning, but the system still keeps stable and the system output asymptotically converges to the desired trajectory. Figure 4 illustrates that the parameter $\boldsymbol{\beta}$ from the two activation functions is bounded throughout the control process. This guarantees the bound of the control signal, which is depicted in Fig. 5. The figure further shows that the control signal is smooth without high-frequency chattering during the whole control process. Based on the Sigmoidal additive and Gaussian RBF activation functions, the performance evaluation of the proposed stable control method under different number of

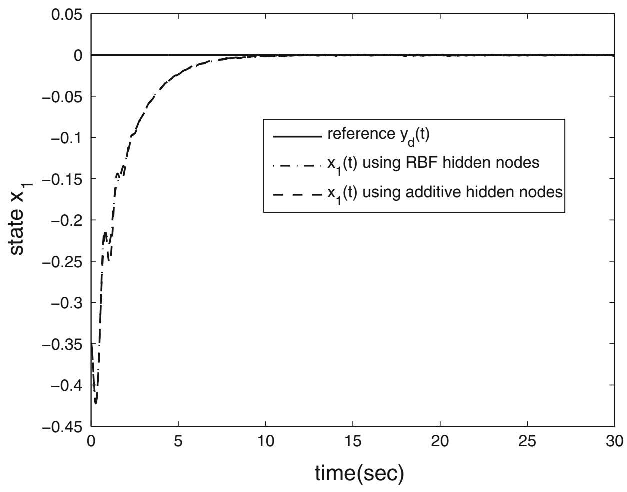

Fig. 2 Closed-loop system state $x_1(t)$ and its reference trajectory $y_d(t)$ in example 1

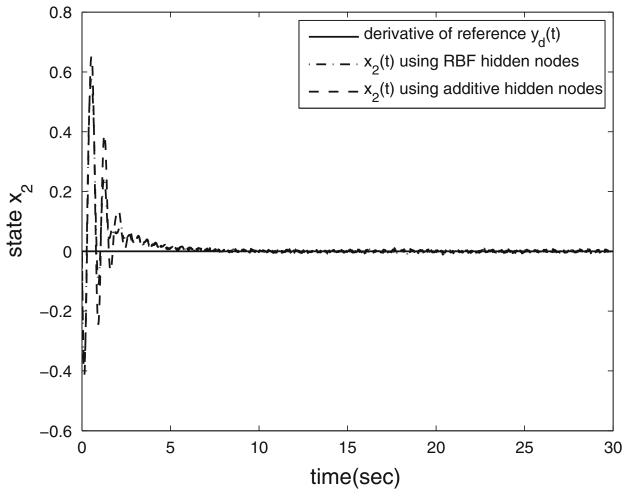

Fig. 3 Closed-loop system state $x_2(t)$ and its reference trajectory $\dot{y}_d(t)$ in example 1

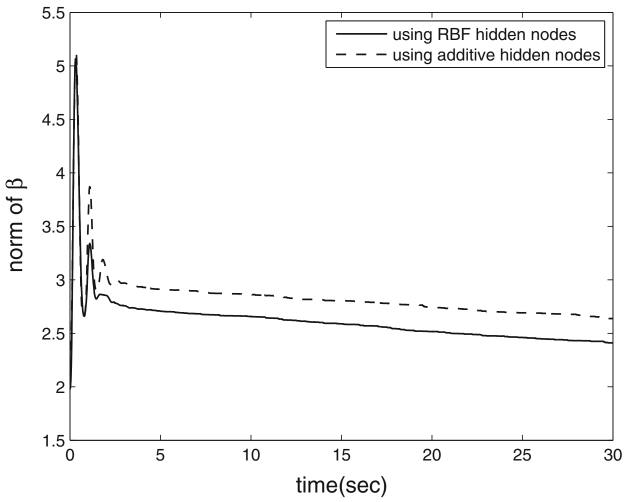

Fig. 4 Norm of output weight $\boldsymbol{\beta}$ in example 1

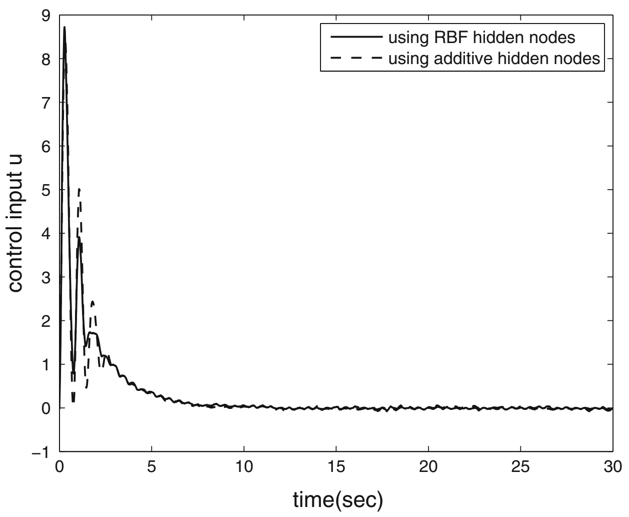

Fig. 5 Control input signal u in example 1

Table 1 Performance evaluation under different number of hidden nodes

# of nodes	Sigmoidal additive			Gaussian RBF		
	E_{x_1}	E_{x_2}	E_{ave}	E_{x_1}	E_{x_2}	E_{ave}
5	0.0535	0.0684	0.0610	0.0534	0.0511	0.0523
10	0.0517	0.0516	0.0517	0.0528	0.0449	0.0489
15	0.0562	0.0518	0.0540	0.0560	0.0542	0.0551
20	0.0616	0.0586	0.0601	0.0605	0.0609	0.0607
25	0.0611	0.0615	0.0613	0.0626	0.0642	0.0634

hidden nodes is demonstrated in Table 1. One can observe from the table that the tracking RMS errors including state x_1, state x_2 and their average are insensitive to the chosen value of hidden nodes, although the number of hidden nodes is varied from 5 to 25. This states that the designed controller has good robustness to the number of hidden nodes. In addition, one can observe from the simulation results that the controller constructed using the Gaussian RBF hidden nodes and Sigmoidal additive hidden nodes has the similar performance.

To further evaluate the performance of the proposed control method, the original ELM algorithm is used here to train the SLFN for approximating the control input described in (12), where the output weights are updated using the least-square error method. The tracking performance of the state x_1 and x_2 achieved from the original ELM and the proposed method of the paper is demonstrated in Figs. 6 and 7. It is noteworthy that only the simulation results from the Sigmoidal additive activation function are shown here because the similar results between the Gaussian RBF and Sigmoidal additive activation functions are obtained. Thus the results from the Gaussian RBF activation function are omitted here. From the two figures, it can be found that the state x_2 from the two methods can follow the reference trajectory very well after the initial tracking error. However, the state x_1 obtained from the original ELM cannot follow its reference trajectory well and converge to the reference trajectory during the simulation process. This depicts that the original ELM lacks the stability proof of the whole control system, and thus the zero convergence of the tracking error cannot be satisfied.

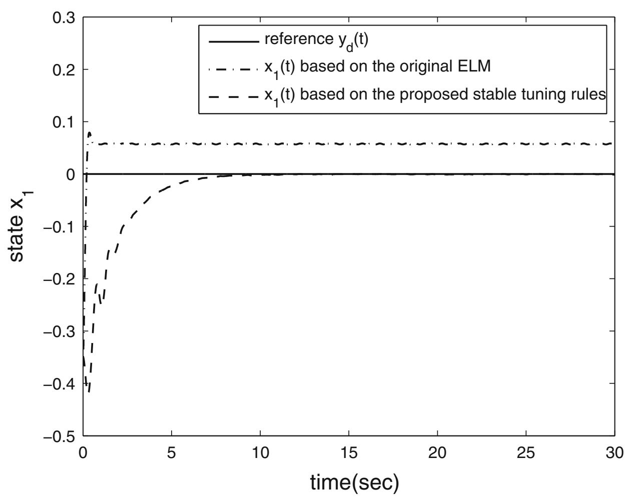

Fig. 6 State $x_1(t)$ output comparison between the original ELM and the proposed method in example 1

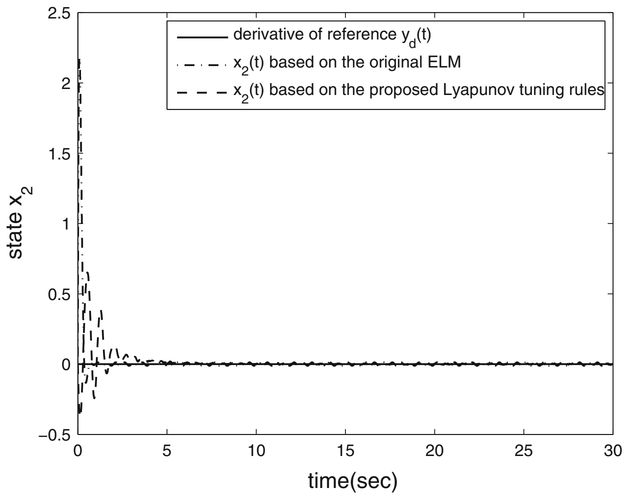

Fig. 7 State $x_2(t)$ output comparison between the original ELM and the proposed method in example 1

Tables 2 and 3 depict the performance evaluation of the controller when the random intervals of input weights are varied. From the two tables, it is clear to observe that similar results are obtained for different random input intervals, and thus the proposed controller will not be affected by the arbitrary input weights.

Example 2 We again consider the inverted pendulum system as shown in Example 1. But in the example the different reference trajectories $y_d(t) = (\frac{\pi}{30}\sin(t), \frac{\pi}{30}\cos(t))^T$ are required to be tracked by the proposed direct adaptive neural controller under the initial state $x(0) = (-\pi/18, -0.05)^T$. The controller design parameters used in the simulations are the same as those in Example 1 except that the number of hidden nodes is set as 25.

Table 2 Performance evaluation under different random intervals for parameter **c**

Input intervals	[−0.5, 0.5]	[−1, 1]	[−2, 2]	[−3, 3]	[−4, 4]
E_{x_1}	0.0569	0.0517	0.0577	0.0595	0.0612
E_{x_2}	0.0580	0.0516	0.0465	0.0450	0.0419
E_{ave}	0.0575	0.0517	0.0521	0.0523	0.0516

Table 3 Performance evaluation under different random intervals for parameter **a**

Input intervals	[0, 0.5]	[0, 1]	[0, 2]	[0, 3]	[0, 4]
E_{x_1}	0.0551	0.0517	0.0614	0.0612	0.0632
E_{x_2}	0.0513	0.0516	0.0555	0.0434	0.0555
E_{ave}	0.0532	0.0517	0.0585	0.0523	0.0594

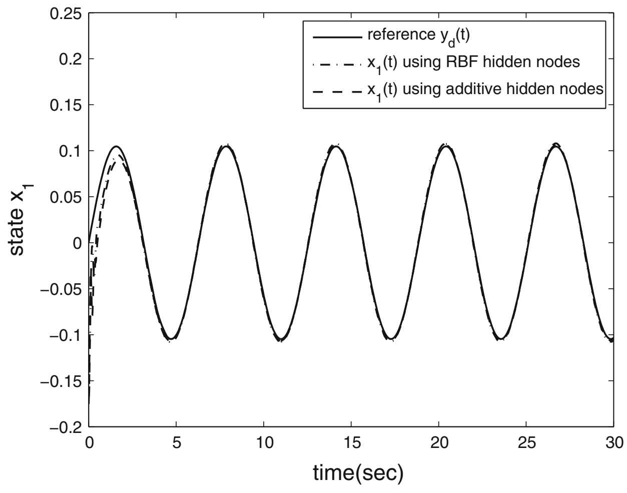

Fig. 8 Closed-loop system state $x_1(t)$ and its reference trajectory $y_d(t)$ in Example 2

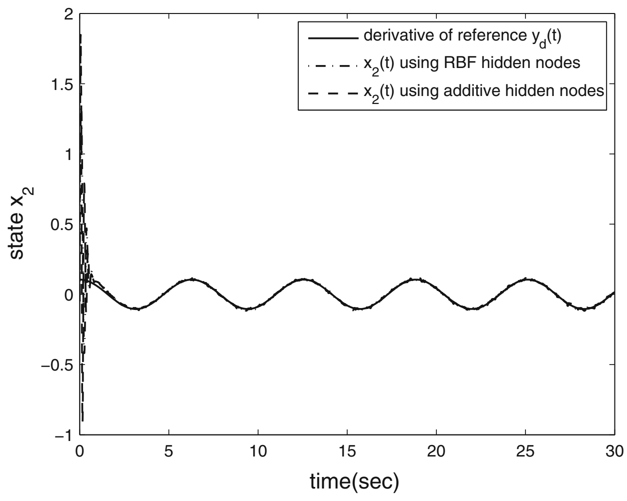

Fig. 9 Closed-loop system state $x_2(t)$ and its reference trajectory $\dot{y}_d(t)$ in example 2

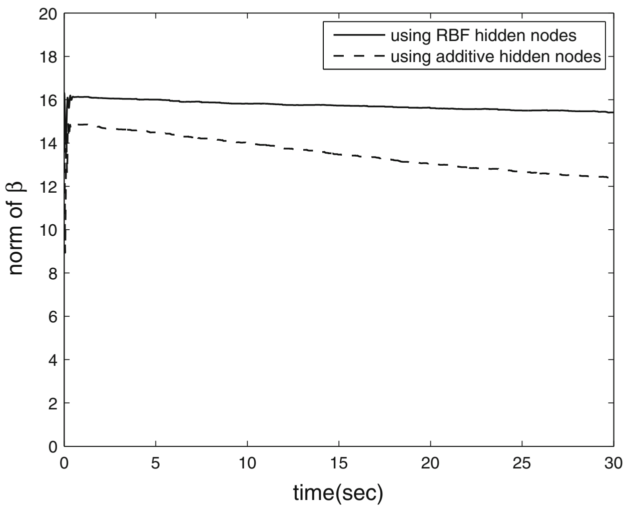

Fig. 10 Norm of output weight $\boldsymbol{\beta}$ in example 2

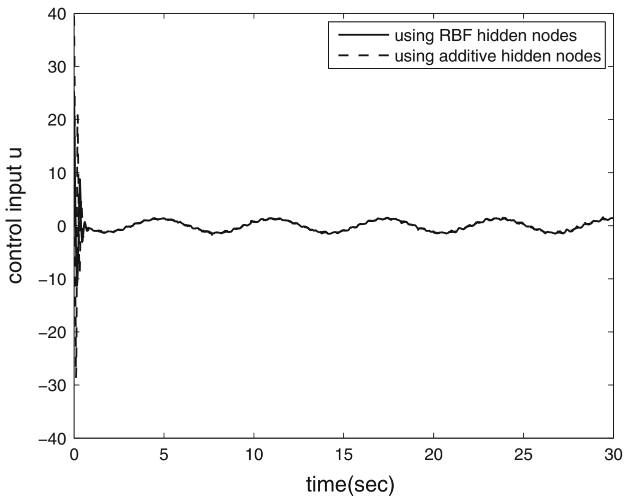

Fig. 11 Control input signal u in example 2

Figures 8 and 9 demonstrate the desired outputs and simulation results of states x_1 and x_2. From the two figures, one can find that though the reference trajectories are changed, the system outputs achieved by the proposed neural controller still asymptotically track the desired trajectories well. Again, the output weight $\boldsymbol{\beta}$ is bounded throughout the control process as shown in Fig. 10. The control signal as depicted in Fig. 11 is also smooth without high-frequency chattering. Besides, one can note that the simulation results from the Gaussian RBF hidden nodes and Sigmoidal additive hidden nodes are similar and both demonstrate good tracking performance.

5 Conclusions

In this paper, a direct adaptive neural control scheme for a class of high-order nonlinear continuous systems is proposed. The scheme incorporates the neural controller and the sliding mode controller. The neural controller is built using the SLFN and its hidden node parameters are determined based on the ELM algorithm by assigning randomly the values to them. The output weights of the SLFN are updated based on the stable adaption laws derived from a Lyapunov function in order to achieve the asymptotic stability of the proposed control system. The sliding mode controller is designed to handle the modeling error of the SLFN and system uncertainty. Computer simulation studies of an inverted pendulum system verify the adaptation and tracking performance of the proposed control approach. Also the simulation results demonstrate that the designed controller has good robustness with respect to different number of hidden nodes and the arbitrary input weights. Moreover, Gaussian RBF hidden nodes and Sigmoidal additive hidden nodes are utilized to evaluate the proposed control scheme and obtain similar performance.

Acknowledgments This work is funded in part by National Natural Science Foundation of China (Grant No. 61004055); Shaanxi Provincial Natural Science Foundation of China (Grant No. 2009JQ8003); the Scientific Research Plan for New Teachers, Xi'an Jiaotong University; and the Principal's Fund of Xi'an Jiaotong University (Grant No. xjj2009017). The authors would like to thank the reviewers for their comments and suggestions that have improved the quality of presentation.

References

1. Bošković JD (1997) A stable neural network-based adaptive control scheme for a class of nonlinear plants. In: Proceedings of the 36th conference on decision and control. San Diego, USA, pp 472–477
2. Burton JA, Zinober ASI (1986) Continuous approximation of variable structure control. Int J Syst Sci 17(6):875–885
3. Gomi H, Kawato M (1992) Neural network control for a closed-loop system using feedback-error-learning. Neural Netw 6(7): 933–946
4. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9(1): 224–229
5. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
6. Huang GB, Siew CK (2004) Extreme learning machine: RBF network case. In: Proceedings of the eighth international conference on control, automation, robotics and vision (ICARCV 2004). Kunming, China, pp 6–9
7. Huang GB, Zhu QY, Mao KZ, Siew CK, Saratchandran P, Sundararajan N (2006) Can threshold networks be trained directly. IEEE Trans Circ Syst II 53(3):187–191
8. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
9. Huang GB, Zhu QY, Siew CK (2006) Real-time learning capability of neural networks. IEEE Trans Neural Netw 17(4): 863–878
10. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: A new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol 2. Budapest, Hungary (25–29 July, 2004), pp 985–990
11. Liang NY, Saratchandran P, Huang GB, Sundararajan N (2006) Classification of mental tasks from EEG signals using extreme learning machine. Int J Neural Syst 16(1):29–38
12. Man Z, Wu HR, Palaniswami M (1998) An adaptive tracking controller using neural networks for a class of nonlinear systems. IEEE Trans Neural Netw 9(5):947–955
13. Niu Y, Wang X, Lam J, Ho D (2003) Sliding-mode control for nonlinear state-delayed systems using neural-network approximation. IEE Proc Control Theory Appl 150(3):233–239
14. Passino KM (2005) Biomimicry for optimization control and automation. Springer-Verlag, London, UK
15. Sanner RM, Slotine JJE (1992) Gaussian networks for direct adaptive control. IEEE Trans Neural Netw 3(6):837–863
16. Slotine JJE (1984) Sliding controller design for non-linear systems. Int J Control 40(2):421–433
17. Slotine JJE, Li WP (1991) Applied Nonlinear Control. Prentice-Hall, Englewood Cliffs, NJ
18. Sundararajan N, Saratchandran P, Yan L (2001) Fully tuned radial basis function neural networks for flight control. Kluwer academic publishers, Boston
19. Suresh S, Narasimhan S, Nagarajaiah S, Sundararajan N (2010) Fault-tolerant adaptive control of nonlinear base-isolated buildings using emran. Eng Str 32(8):2477–2487
20. Suresh S, Saraswathi S, Sundararajan N (2010) Performance enhancement of extreme learning machine for multi-category sparse data classification problems. Eng Appl Artif Int 23(7): 1149–1157
21. Wang LX (1993) Stable adaptive fuzzy control of nonlinear systems. IEEE Trans Fuzzy Syst 1(2):146–155
22. Wang LX (1994) Adaptive fuzzy systems and control: design and stability analysis. Prentice-Hall, Upper Saddle River, NJ
23. Xu JX, Wang W, Goh JCH, Lee G (2005) Internal model approach for gait modeling and classification. In: The 27th annual international conference of the IEEE engineering in medicine and biology society (EMBS). Shanghai, China (September 1–4, 2005)
24. Zhang R, Huang GB, Sundararajan N, Saratchandran P (2007) Multi-category classification using an extreme learning machine for microarray gene expression cancer diagnosis. IEEE/ACM Trans Comput Biol Bioinf 4(3):485–495