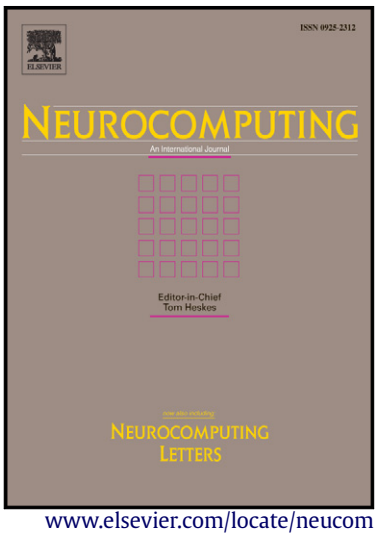


www.elsevier.com/locate/neucom

Adaptive Neural Control for a Class of MIMO Nonlinear Systems with Extreme Learning Machine

Hai-Jun Rong, Jin-Tao Wei, Jian-Ming Bai, Guang-She Zhao, Yong-Qi Liang

PII: S0925-2312(14)01136-9
DOI: http://dx.doi.org/10.1016/j.neucom.2014.01.066
Reference: NEUCOM14604

To appear in: *Neurocomputing*

Received date: 4 August 2013
Revised date: 29 November 2013
Accepted date: 19 January 2014

Cite this article as: Hai-Jun Rong, Jin-Tao Wei, Jian-Ming Bai, Guang-She Zhao, Yong-Qi Liang, Adaptive Neural Control for a Class of MIMO Nonlinear Systems with Extreme Learning Machine, *Neurocomputing*, http://dx.doi.org/10.1016/j.neucom.2014.01.066

Adaptive Neural Control for a Class of MIMO Nonlinear Systems with Extreme Learning Machine

Hai-Jun Rong [a], Jin-Tao Wei [a], Jian-Ming Bai [a,b], Guang-She Zhao [a], Yong-Qi Liang [a,*]

[a] *State Key Laboratory for Strength and Vibration of Mechanical Structures, School of Aerospace, Xi'an Jiaotong University, Xi'an, Shaanxi, 710049, China*

[b] *Optical Direction and Pointing Technique Research Department, Xi'an Institute of Optics and Precision Mechanics of CAS, Xi'an, Shaanxi, 710119, China*

Abstract

This paper presents two adaptive neural control schemes for a class of uncertain continuous-time multi-input multi-output (MIMO) nonlinear dynamic systems. Within these schemes, the single-hidden layer feedforward networks (SLFNs) are applied to approximate the unknown nonlinear functions of the systems and then the neural controller is built based on the approximated neural models. The parameters of the SLFNs are modified using the recently proposed neural algorithm named extreme learning machine (ELM), where the parameters of the hidden nodes are assigned randomly. Different from the original ELM algorithm, the output weights are updated using the adaptive laws derived based on Lyapunov stability theorem and Barbalat's lemma so that the asymptotical stability of the system can be guaranteed. The robustifying control term is also constructed to compensate for approximation errors of the SLFNs. In order to avoid the requirement of the approximation error bounds, the estimation laws derived based on the Lyapunov stability theorem and Barbalat's lemma are employed to estimate the error bounds in the second adaptive control scheme. Finally the proposed control schemes are applied to control a two-link robot manipulator. The simulation results demonstrate the effectiveness of the proposed control schemes for the MIMO nonlinear system.

Key words: Single-Hidden Layer Feedforward Network; Extreme Learning Machine; MIMO nonlinear systems

* Corresponding author
Email address: yqliang@mail.xjtu.edu.cn (Yong-Qi Liang).

1 Introduction

Adaptive control of nonlinear systems is an intense area of research and various approaches have been proposed for the analysis and design of nonlinear control systems. The most popular control approaches in the adaptive control of nonlinear systems are facilitated through advances in geometric nonlinear control theory and, in particular, feedback linearization methods [1] and backstepping methods [2]. However, when these methods are applied, the system nonlinearities have to been known a priori. This is generally not applicable in the real world because it is difficult to describe a nonlinear system using known nonlinear functions precisely.

Recently, neural networks (NNs) have received a great deal of research interest in the identification and control of dynamic systems with unknown nonlinearities. The approximating capabilities and inherent adaptive features of neural networks are appealing for modeling and control of nonlinear dynamic systems. Also from a practical perspective, the massive parallelism and fast adaptability of neural network implementations provide more incentive for further investigation in problems involving dynamic systems with unknown nonlinearities. Thus a large amount of research work [3–10] has been carried out in the discipline of neural adaptive control, especially when the controlled system is hard to be modelled mathematically under large uncertainties and strong nonlinearities. For adaptive control purposes neural networks are mostly used as approximation models of unknown nonlinearities. The control design is then based on the neural network model rather than the actual system. This removes the need for a priori knowledge of system nonlinearities. But, most of these studies are developed for the single-input single-output (SISO) nonlinear systems.

In practice, a large number of nonlinear systems possess multivariable characteristic and thus it is also important to develop adaptive control techniques for multiple-input multiple-output (MIMO) nonlinear systems with unknown nonlinear functions. Neural networks together with the adaptive control techniques have been extended to uncertain nonlinear MIMO systems in some investigations. In [11], a neural controller is proposed for a class of affine MIMO system where the radial basis function (RBF) neural networks are use to estimate certain nonlinear functions. This avoids the linearity-in-the-parameter assumption and the selection of some regression matrices. But, the control input matrix must be known in the control scheme. This is stringent and may limit its use in some practical applications. For a general class of uncertain affine MIMO nonlinear systems with an unknown control coefficient matrix some adaptive neural controls are investigated. In [12] and [13] the RBF neural networks are utilized to approximate the nonlinear functions including the control input matrix. Within these control schemes, the semi-global stability

of the closed-loop systems are guaranteed and the tracking errors are converged to a residual set whose size depends on the bounds of the unknown neural approximation errors which need to satisfy some specific inequalities and then are chosen by trial and error. In this case the exact upper bound for the tracking error cannot be easily achieved.

In addition, in all the control schemes described above, the control laws are constructed using the linearly parameterized neural network models where the centers and widths of RBFs are determined according to some prior information regarding the system to be approximated. However, in practice 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 also hard to determine. The use of inaccurate centers and widths usually results in a deterioration of the approximation performance and further increases the tracking error bound. In [14], a hybrid control system integrating principal and compensation controllers is developed for a class of affine MIMO uncertain nonlinear systems. The principal controller is based on the sliding-mode technique and uses a recurrent cerebellar model articulation controller (RCMAC) as an uncertainty observer. The compensation controller is a compensator for the approximation error of the RCMAC. The proposed control scheme guarantees the asymptotical stability of the system and convergence of the tracking error to zero. Also, an estimation law is derived to estimate the approximation error bound so that the requirement of its prior information is relaxed. But in the control scheme the control law is constructed using the nonlinearly parameterized neural network model by adjusting the mean and variance of each receptive-field basis function. In this case, the Taylor linearization technique need to be employed to transform the nonlinear model into a partially linear form by ignoring the high-order arguments. Also a high computational complexity will be caused because of a large number of tunable parameters.

In the paper, two adaptive neural control schemes for a class of affine MIMO nonlinear systems are presented. In these two schemes, the single-hidden layer feedforward netowrks (SLFNs) with additive nodes (i.e., Sigmoid units) or RBF nodes are used as the function approximators to estimate the unknown nonlinear systems. The control laws are then established based on the estimated models. Different from the existing methods, the parameters of the SLFNs are adjusted based on the recently proposed neural algorithm referred to as extreme learning machine (ELM) [15–17] which possesses the outstanding computational efficiency in terms of learning speed and generalization ability. This has been verified exclusively in many applications, e.g., some classification problems [18–22], function approximation [23,24] and flight control [25]. 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 distri-

bution 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, which makes ELM learning scheme more efficiently and much simpler. Based on this a SLFN is considered as the linearly parameterized neural network model. When such a SLFN is applied in designing the nonlinear controller, the Taylor linearization technique for the nonlinearly parameterized model can be avoided and also a less computation process can be achieved by only adjusting the output weights.

In this study, the ELM algorithm is utilized to determine the parameters of hidden nodes in SLFNs by assigning random values to them. But the output weights are adjusted using the adaptive laws derived based on Lyapunov stability theorem and Barbalat's lemma, so that the asymptotical stability of the system can be guaranteed. Also to compensate for the approximation errors of the SLFNs, the robustifying control term depending on their bounds is incorporated and activated to work together with the neural controller. To remove the requirement of a priori knowledge about the error bounds, the robustifying control term in the second control scheme is constructed using the estimated error bounds which are updated based on the adaptive laws derived from the Lyapunov stability theorem and Barbalat's lemma. Finally, the proposed adaptive neural controller is applied to control a two-link robot manipulator. The simulation results clarify the proposed control scheme for the uncertain MIMO nonlinear systems.

This paper is organized as follows. Section 2 introduces the dynamic model of a class of MIMO nonlinear systems under consideration. In Section 3, the design procedure of the adaptive neural controller is introduced together with the SLFN approximators, the ELM algorithm and the stable adaptive laws for adjusting the output weights based on the Lyapunov synthesis approach. Section 4 shows the simulation results by controlling a two-link robot manipulator. Section 5 presents the conclusions from this study.

2 Problem Formulation

Consider a class of MIMO nonlinear systems described by the following set of differential equations:

$$\begin{cases} x_1^{(n_1)} = f_1(\mathbf{x}) + g_{11}(\mathbf{x})u_1 + \cdots + g_{1p}(\mathbf{x})u_p \\ \vdots \\ x_p^{(n_p)} = f_p(\mathbf{x}) + g_{p1}(\mathbf{x})u_1 + \cdots + g_{pp}(\mathbf{x})u_p \\ y_1 = x_1, \cdots, y_p = x_p \end{cases} \tag{1}$$

where $x_i^{(n_i)} = d^{n_i}/dt^{n_i} (i = 1, 2, \cdots, p)$; $\mathbf{x}(= [x_1, \cdots, x_1^{(n_1-1)}, \cdots, x_p, \cdots, x_p^{(n_p-1)}]^T \in \mathbf{R}^n)$ with $n = n_1 + \cdots + n_p$ is the overall state vector of the system; $\mathbf{u}(= [u_1, \cdots, u_p] \in \mathbf{R}^p)$ is the control input vector.

If the following notation is defined as,

$$\begin{aligned} \mathbf{x}^{(n)} &= [\mathbf{x}_1^{(n_1)} \cdots \mathbf{x}_p^{(n_p)}]^T \in \mathbf{R}^p \\ \mathbf{f}(\mathbf{x}) &= \begin{bmatrix} f_1(\mathbf{x}) \cdots f_p(\mathbf{x}) \end{bmatrix}^T \in \mathbf{R}^p \\ \mathbf{G}(\mathbf{x}) &= \begin{bmatrix} g_{11}(\mathbf{x}) \cdots g_{1p}(\mathbf{x}) \\ \vdots \quad \ddots \quad \vdots \\ g_{p1}(\mathbf{x}) \cdots g_{pp}(\mathbf{x}) \end{bmatrix} \in \mathbf{R}^{p \times p} \end{aligned} \tag{2}$$

system (1) becomes as

$$\mathbf{x}^{(n)} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} \tag{3}$$

For the accomplishment of control task for system (3), the following assumptions are made:
Assumption 1: The matrix $\mathbf{G}(\mathbf{x})$ is positive definite for all $\mathbf{x}$ in a compact region $\Omega_x \in \mathbf{R}^n$, then it satisfies $\mathbf{G}(\mathbf{x}) \geq \gamma_0 \mathbf{I}_p$ with $\gamma_0 > 0$, $\gamma_0 \in R$.

Assumption 2: The state $\mathbf{x}$ of the system is available for measurement.

Assumption 3: The reference signals $y_{d_i}(t), i = 1, 2, \cdots, p$ are known bounded functions of time with bounded known derivatives.

By denoting

$$\begin{aligned} \mathbf{y}_d^{(n)} &= [y_{d1}^{(n_1)}, \cdots, y_{dp}^{(n_p)}]^T \in \mathbf{R}^p \\ \mathbf{y}_d &= [y_{d1}, \cdots, y_{d1}^{(n_1-1)}, \cdots, y_{dp}, \cdots, y_{dp}^{(n_p-1)}]^T \in \mathbf{R}^n \end{aligned} \tag{4}$$

the output tracking error equals as,

$$\mathbf{e} = \mathbf{y}_d - \mathbf{x} = [e_1, \cdots, e_1^{(n_1-1)}, \cdots, e_p, \cdots, e_p^{(n_p-1)}]^T \in \mathbf{R}^n \tag{5}$$

Based on the assumptions made above, the control law $\mathbf{u}$ of system (3) is designed using feedback linearization method as follows:

$$\mathbf{u} = \mathbf{G}^{-1}(\mathbf{x})[\mathbf{y}_d^{(n)} - \mathbf{f}(\mathbf{x}) + \mathbf{k}^{\mathbf{T}}\mathbf{e}] \tag{6}$$

where $\mathbf{k}(= [\mathbf{k}_n, \mathbf{k}_{n-1}, \cdots, \mathbf{k}_1] \in \mathbf{R}^{n \times p})$ is the real number matrix. By substi-

tuting Eqn (6) into Eqn (3), the following equation is obtained:

$$\begin{cases} e_1^{(n_1)} + k_{1,1}e_1^{(n_1-1)} + \cdots + k_{1,(n_1-1)}\dot{e}_1 + k_{1,n_1}e_1 = 0 \\ e_2^{(n_2)} + k_{2,1}e_2^{(n_2-1)} + \cdots + k_{2,(n_2-1)}\dot{e}_2 + k_{2,n_2}e_2 = 0 \\ \vdots \\ e_p^{(n_p)} + k_{p,1}e_p^{(n_p-1)} + \cdots + k_{p,(n_p-1)}\dot{e}_p + k_{p,n_p}e_p = 0 \end{cases} \tag{7}$$

By properly choosing $k_{i,n_i}(i = 1, 2, \cdots, p)$, all roots of the polynomial $s^{(n_i)} + k_{i,1}s^{(n_i-1)} + \cdots + k_{i,n_i} = 0$ are in the open left half plane, which implies that the tracking error will converge to zero. Eqn (7) also shows that the tracking of reference trajectory is asymptotically achieved from any initial conditions, i.e., $\lim_{t\to\infty}\|\mathbf{e}\| = 0$.

However, when $\mathbf{f}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$ are unavailable directly for the control design, the ideal control law $\mathbf{u}$ is hard to achieve. In this study, a neural control law is designed using the SLFNs to learn these uncertain dynamics and given by

$$\mathbf{u}_n = \hat{\mathbf{G}}^{-1}(\mathbf{x})[\mathbf{y}_d^{(n)} - \hat{\mathbf{f}}(\mathbf{x}) + \mathbf{k}^{\mathbf{T}}\mathbf{e}] \tag{8}$$

where $\hat{\mathbf{f}}(\mathbf{x})$ and $\hat{\mathbf{G}}(\mathbf{x})$ are the approximation models from the SLFNs.

The approximation errors between the ideal models and the approximated models will arise in Eqn (8) when the SLFNs are utilized to approximate $\mathbf{f}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$. To compensate for the approximation errors, a robustifying control term $\mathbf{u}_s$ is used to augment the neural controller above and thus the overall control law is considered as

$$\mathbf{u} = \mathbf{u}_n + \mathbf{u}_s \tag{9}$$

By adding and substructing $\hat{\mathbf{G}}(\mathbf{x})\mathbf{u}_n$ to Eqn (3), the error equation of system (3) becomes as,

$$\begin{pmatrix} e_1^{(n_1)} + k_{11}e_1^{(n_1-1)} + \cdots + k_{1,(n_1)}e_1 \\ \vdots \\ e_p^{(n_p)} + k_{p1}e_p^{(n_p-1)} + \cdots + k_{p,(n_p)}e_p \end{pmatrix} = \begin{pmatrix} \hat{f}_1(\mathbf{x}) - f_1(\mathbf{x}) \\ \vdots \\ \hat{f}_p(\mathbf{x}) - f_p(\mathbf{x}) \end{pmatrix} + (\hat{\mathbf{G}}(\mathbf{x}) - \mathbf{G}(\mathbf{x})) \begin{pmatrix} u_{n1} \\ \vdots \\ u_{np} \end{pmatrix} - \mathbf{G}(\mathbf{x}) \begin{pmatrix} u_{s1} \\ \vdots \\ u_{sp} \end{pmatrix} \tag{10}$$

or, equivalently,

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} + \mathbf{B}\left[(\hat{\mathbf{f}}(\mathbf{x}) - \mathbf{f}(\mathbf{x})) + (\hat{\mathbf{G}}(\mathbf{x}) - \mathbf{G}(\mathbf{x}))\mathbf{u}_n\right] - \mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s \tag{11}$$

in which

$$\begin{aligned} \mathbf{A} &= \text{diag}\left[\mathbf{A}_1, \cdots, \mathbf{A}_p\right] \in \mathbf{R}^{n\times n} \\ \mathbf{B} &= \text{diag}\left[\mathbf{B}_1, \cdots, \mathbf{B}_p\right] \in \mathbf{R}^{n\times p} \end{aligned} \tag{12}$$

and

$$\mathbf{A}_i = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -k_{i,n_i} & -k_{i,n_i-1} & -k_{i,n_i-2} & \cdots & -k_{i,1} \end{bmatrix}_{n_i\times n_i}, \mathbf{B}_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}_{n_i\times 1} \tag{13}$$

The control objective of designing the neural controller is to force the system output $\mathbf{x}$ to follow the specified desired reference trajectory $\mathbf{y}_d$ as closely as possible. In the following section, we will describe the design procedure of the proposed adaptive neural controller together with the SLFN considered in the paper for approximating the function $\mathbf{f}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$.

3 Design Procedure of Adaptive Neural Controller

3.1 Mathematical Description of Unified SLFN

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

$$f_{\tilde{N}}(\mathbf{x}) = \sum_{l=1}^{\tilde{N}} \boldsymbol{\beta}_l \Psi(\mathbf{x}; \mathbf{c}_l, a_l), \quad \mathbf{x} \in \mathbf{R}^n, \mathbf{c}_l \in \mathbf{R}^n \tag{14}$$

where $\mathbf{c}_l$ and a_l are the learning parameters of hidden nodes, $\boldsymbol{\beta}_l$ is the output weight connecting the l-th hidden node to the output node, and $\Psi(\mathbf{x}; \mathbf{c}_l, a_l)$ is the output of the l-th hidden node with respect to the input $\mathbf{x}$. For additive hidden nodes with the Sigmoid or threshold activation function $\psi(x) : R \to R$, $\Psi(\mathbf{x}; \mathbf{c}_l, a_l)$ is given by

$$\Psi(\mathbf{x}; \mathbf{c}_l, a_l) = \psi(\mathbf{c}_l \cdot \mathbf{x} + a_l), \quad a_l \in R \tag{15}$$

where $\mathbf{c}_l$ is the weight vector connecting the input layer to the l-th hidden node, a_l is the bias of the l-th hidden node, and $\mathbf{c}_l \cdot \mathbf{x}$ denotes the inner

product of vectors $\mathbf{c}_l$ and $\mathbf{x}$ in $\mathbf{R}^n$.

For RBF hidden nodes with the Gaussian or triangular activation function $\psi(x) : R \to R$, $\Psi(\mathbf{x}; \mathbf{c}_l, a_l)$ is given by

$$\Psi(\mathbf{x}; \mathbf{c}_l, a_l) = \psi(a_l \|\mathbf{x} - \mathbf{c}_l\|), \quad a_l \in R^+ \tag{16}$$

where $\mathbf{c}_l$ and a_l are the center and impact factor of l-th 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.

3.2 Extreme Learning Machine

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

$$\sum_{l=1}^{\tilde{N}} \boldsymbol{\beta}_l \Psi(\mathbf{x}_k; \mathbf{c}_l, a_l) = \mathbf{t}_k, \quad k = 1, \cdots, N. \tag{17}$$

Eqn (17) can be written compactly as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{18}$$

where

$$\begin{aligned} &\mathbf{H}(\mathbf{c}_1, \cdots, \mathbf{c}_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}, \mathbf{x}_1, \cdots, \mathbf{x}_N) \\ &= \begin{bmatrix} \Psi(\mathbf{x}_1; \mathbf{c}_1, a_1) & \cdots & \Psi(\mathbf{x}_N; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ \Psi(\mathbf{x}_1; \mathbf{c}_1, a_1) & \cdots & \Psi(\mathbf{x}_N; \mathbf{c}_{\tilde{N}}, a_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \end{aligned} \tag{19}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{\tilde{N}}{}^T \end{bmatrix}_{\tilde{N} \times p} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times p}^T \tag{20}$$

$\mathbf{H}$ is called the hidden layer output matrix of the network [17,26]; the lth row of $\mathbf{H}$ is the l-th hidden node's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ and the k-th column of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\mathbf{x}_k$.

The adaptive neural control scheme will be designed with the help of the following lemma.

Lemma 1 [17]: For a SLFN with additive or RBF hidden nodes and an activation function $\psi(x) : R \to R$ which is infinitely differentiable in any interval, there exists $\tilde{N} \leq N$ such that for N arbitrary distinct input vectors $\{\mathbf{x}_l | \mathbf{x}_l \in \mathbf{R}^n, l = 1, \cdots, \tilde{N}\}$, for any $\{(\mathbf{c}_l, a_l)\}_{l=1}^{\tilde{N}}$ 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.

In the following two adaptive neural control schemes equipped with the robustifying control term are developed sequentially.

3.3 Adaptive Neural Control Scheme with Known Error Bounds

When the SLFNs are utilized to approximate the unknown nonlinear dynamics $\mathbf{f}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$ in the designed control scheme, the hidden node parameters ($\mathbf{c_l}$, a_l, $l = 1, 2, \cdots, \tilde{N}$) of the SLFNs need not be tuned during learning and may simply be assigned with random values according to the property of ELM. In this case, Eqn (14) becomes a linearly parameterized model. Then the function $\hat{\mathbf{f}}(\mathbf{x})$ and $\hat{\mathbf{G}}(\mathbf{x})$ in the neural control law $\mathbf{u}_n$ are given by

$$\begin{aligned} \hat{f}_i(\mathbf{x}) &= \mathbf{H}_{fi}(\mathbf{x}; \mathbf{c}_{fi}, \mathbf{a}_{fi})\boldsymbol{\beta}_{fi}; \qquad i = 1, 2, \cdots, p \\ \hat{g}_{ij}(\mathbf{x}) &= \mathbf{H}_{gij}(\mathbf{x}; \mathbf{c}_{gij}, \mathbf{a}_{gij})\boldsymbol{\beta}_{gij}; \qquad j = 1, 2, \cdots, p \end{aligned} \tag{21}$$

where $\hat{f}_i(\mathbf{x})$ is the i-th element of $\hat{\mathbf{f}}(\mathbf{x})$; $\hat{g}_{ij}(\mathbf{x})$ is the i-th row and j-th column element of $\hat{\mathbf{G}}(\mathbf{x})$.

In our study, the hidden node parameters $\{\mathbf{c}_{fi}, \mathbf{a}_{fi}\}$ and $\{\mathbf{c}_{gij}, \mathbf{a}_{gij}\}$ existing in $\hat{f}_i(\mathbf{x})$ and $\hat{g}_{ij}(\mathbf{x})$ are updated using ELM algorithm where all these parameters could randomly be generated according to any given continuous probability distribution without any prior knowledge about the target function. The parameters in all hidden nodes are not only independent from each other but also independent from the learning data. Based on this, different hidden node output vector $\mathbf{H}_{fi}(\mathbf{x})$ and $\mathbf{H}_{gij}(\mathbf{x})$ in the function vector $\hat{\mathbf{f}}(\mathbf{x})$ and the matrix $\hat{\mathbf{G}}(\mathbf{x})$ can be replaced by a common hidden node output vector $\mathbf{H}(\mathbf{x})$ when all the parameters $\{\mathbf{c}_{fi}, \mathbf{a}_{fi}\}$ and $\{\mathbf{c}_{gij}, \mathbf{a}_{gij}\}$ are assigned randomly by a common set $\{\mathbf{c}, \mathbf{a}\}$. Thus Eqn.(21) can be further written as follows:

$$\hat{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} \mathbf{H}(\mathbf{x})\boldsymbol{\beta}_{f1} \\ \vdots \\ \mathbf{H}(\mathbf{x})\boldsymbol{\beta}_{fp} \end{bmatrix} = \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_f \tag{22}$$

$$\begin{aligned}\hat{\mathbf{G}}(\mathbf{x}) &= \begin{bmatrix} \mathbf{H}(\mathbf{x})\boldsymbol{\beta}_{g11} & \cdots & \mathbf{H}(\mathbf{x})\boldsymbol{\beta}_{g1p} \\ \vdots & \vdots & \vdots \\ \mathbf{H}(\mathbf{x})\boldsymbol{\beta}_{gp1} & \cdots & \mathbf{H}(\mathbf{x})\boldsymbol{\beta}_{gpp} \end{bmatrix} \\ &= \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_G \end{aligned} \tag{23}$$

where $\boldsymbol{\beta}_f \in \mathbf{R}^{\tilde{N}p}$ and $\boldsymbol{\beta}_G \in \mathbf{R}^{\tilde{N}p\times p}$ are the parameter matrices, $\boldsymbol{\phi}(\mathbf{x}) \in \mathbf{R}^{p\times \tilde{N}p}$ is the block-diagonal matrix which consists of the hidden node output vector $\mathbf{H}(\mathbf{x})$.

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 Eqn (18). According to the universal approximation theorem of the ELM, there exists the optimal output weight parameters $\boldsymbol{\beta}_f^*$ and $\boldsymbol{\beta}_G^*$ to approximate the nonlinear dynamic functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$, which are defined as

$$\begin{aligned} \boldsymbol{\beta}_f^* &\triangleq \text{argmin}_{\boldsymbol{\beta}_f\in\Omega_f}\left[\sup_{\mathbf{x}\in\Omega_x}\|\mathbf{f}(\mathbf{x}) - \hat{\mathbf{f}}(\mathbf{x})\|\right] \\ \boldsymbol{\beta}_G^* &\triangleq \text{argmin}_{\boldsymbol{\beta}_G\in\Omega_G}\left[\sup_{\mathbf{x}\in\Omega_x}\|\mathbf{G}(\mathbf{x}) - \hat{\mathbf{G}}(\mathbf{x})\|\right] \end{aligned} \tag{24}$$

such that

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_f^* + \boldsymbol{\epsilon}_f(\mathbf{x}) \\ \mathbf{G}(\mathbf{x}) &= \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_G^* + \boldsymbol{\epsilon}_G(\mathbf{x}) \end{aligned} \tag{25}$$

where $\boldsymbol{\epsilon}_f(\mathbf{x})$ and $\boldsymbol{\epsilon}_G(\mathbf{x})$ are the approximation errors. Ω_x, Ω_f and Ω_G are the predefined compact set for input $\mathbf{x}$, parameters $\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_G$, which are defined as,

$$\begin{aligned} \Omega_x &= \{\mathbf{x} \in \mathbf{R}^n : \|\mathbf{x}\| \le M_x\} \\ \Omega_f &= \left\{\boldsymbol{\beta}_f \in \mathbf{R}^{\tilde{N}p} : \|\boldsymbol{\beta}_f\| \le M_{\beta_f}\right\} \\ \Omega_G &= \left\{\boldsymbol{\beta}_G \in \mathbf{R}^{\tilde{N}p\times p} : \text{tr}(\boldsymbol{\beta}_G^T\boldsymbol{\beta}_G) \le M_{\beta_G}\right\} \end{aligned} \tag{26}$$

where M_x, M_{β_f} and M_{β_G} are positive constants that are specified by the designer.

Using Eqs. (22), (23) and (25), the error dynamics (11) can be formulated as,

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} + \mathbf{B}\left[\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_f + \boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_G\mathbf{u}_n - \boldsymbol{\epsilon}_f(\mathbf{x}) - \boldsymbol{\epsilon}_G(\mathbf{x})\mathbf{u}_n\right] - \mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s \tag{27}$$

where $\tilde{\boldsymbol{\beta}}_f(= \boldsymbol{\beta}_f - \boldsymbol{\beta}_f^*)$ and $\tilde{\boldsymbol{\beta}}_G(= \boldsymbol{\beta}_G - \boldsymbol{\beta}_G^*)$ are the parameter errors.

Assumption 4: The matrix $\hat{\mathbf{G}}(\mathbf{x})$ is invertible for all $\mathbf{x} \in \Omega_x$ and $\boldsymbol{\beta}_G \in \Omega_G$.

Assumption 5: The approximation errors $\boldsymbol{\epsilon}_f(\mathbf{x})$ and $\boldsymbol{\epsilon}_G(\mathbf{x})$ are upper bounded

by some constants $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_G$ over the compact set Ω_x, i.e., $\|\boldsymbol{\epsilon}_f(\mathbf{x})\| \leq \bar{\varepsilon}_f$ and $\|\boldsymbol{\epsilon}_G(\mathbf{x})\| \leq \bar{\varepsilon}_G$.

Huang *et al.* [17] have proved that when the number of hidden nodes $\tilde{N}$ is equal to the number of distinct training samples N, the SLFN with randomly assigned hidden nodes 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 $\boldsymbol{\epsilon}_f(\mathbf{x})$ and $\boldsymbol{\epsilon}_G(\mathbf{x})$ caused by approximating $\mathbf{f}(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})$ are bounded with the constants $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_G$ respectively.

Eqn (27) shows that the parameters ($\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_G$) need to be further modified in order to make the approximation errors as small as possible. The adaptive laws for them are designed as follows:

$$\begin{aligned} \dot{\boldsymbol{\beta}}_f &= -\eta_1 \boldsymbol{\phi}^T(\mathbf{x})\mathbf{B}^T\mathbf{P}\mathbf{e} \\ \dot{\boldsymbol{\beta}}_G &= -\eta_2 \boldsymbol{\phi}^T(\mathbf{x})\mathbf{B}^T\mathbf{P}\mathbf{e}\mathbf{u}_n^T \end{aligned} \tag{28}$$

where η_1 and η_2 are positive constants. $\mathbf{P}$ is the symmetric and positive definite matrix that satisfies the following relationship:

$$\mathbf{J} = -(\mathbf{A}^\mathbf{T}\mathbf{P} + \mathbf{P}\mathbf{A}) \tag{29}$$

where $\mathbf{J}$ is a symmetric definite matrix and selected by the designer.

It should be noted that in the original ELM algorithm the output weights $\boldsymbol{\beta}$ are adjusted based on the least-square error method. In this study, the adaptive laws for parameters ($\boldsymbol{\beta}_f$, $\boldsymbol{\beta}_G$) are derived using Lyapunov second method in order to guarantee the stability of the entire closed-loop control system.

Meanwhile, to compensate for the approximation errors of the SLFNs, the robustifying control term is taken as,

$$\mathbf{u}_s = \mathrm{sgn}(\mathbf{B}^T\mathbf{P}\mathbf{e})\frac{(\bar{\varepsilon}_f + \bar{\varepsilon}_G\|\mathbf{u}_n\|)}{\gamma_0} \tag{30}$$

The property of the proposed adaptive neural control scheme is described by the following theorem.

Theorem 1: Consider the MIMO nonlinear system (1). Suppose that the Assumptions 1-5 are satisfied. If the adaptive neural control law is chosen as (8) with the robustifying control term (30) and parameter adaptation laws (28), all signals in the closed-loop system are bounded and the tracking errors converge to zero as $t \to \infty$.

Proof. Consider the following Lyapunov function candidate:

$$V = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{1}{2\eta_1}\tilde{\boldsymbol{\beta}}_f^T\tilde{\boldsymbol{\beta}}_f + \frac{1}{2\eta_2}\mathrm{tr}(\tilde{\boldsymbol{\beta}}_G^T\tilde{\boldsymbol{\beta}}_G) \tag{31}$$

Using Eqn (27), the derivative of the Lyapunov function is given as

$$\begin{aligned}\dot{V} &= \frac{1}{2}(\dot{\mathbf{e}}^T\mathbf{P}\mathbf{e} + \mathbf{e}^T\mathbf{P}\dot{\mathbf{e}}) + \frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\beta}}}_f^T\tilde{\boldsymbol{\beta}}_f + \frac{1}{\eta_2}\mathrm{tr}(\dot{\tilde{\boldsymbol{\beta}}}_G^T\tilde{\boldsymbol{\beta}}_G) \\ &= \frac{1}{2}\mathbf{e}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{e} + \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_f + \frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\beta}}}_f^T\tilde{\boldsymbol{\beta}}_f + \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_G\mathbf{u}_n \\ &+ \frac{1}{\eta_2}\mathrm{tr}(\dot{\tilde{\boldsymbol{\beta}}}_G^T\tilde{\boldsymbol{\beta}}_G) - \mathbf{e}^T\mathbf{P}\mathbf{B}(\boldsymbol{\epsilon}_f(\mathbf{x}) + \boldsymbol{\epsilon}_G(\mathbf{x})\mathbf{u}_n) - \mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s\end{aligned} \tag{32}$$

Applying Eqn (29), Eqn (32) becomes as,

$$\begin{aligned}\dot{V} = &-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_f + \frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\beta}}}_f^T\tilde{\boldsymbol{\beta}}_f + \mathrm{tr}\left(\mathbf{u}_n\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_G\right) \\ &+ \frac{1}{\eta_2}\mathrm{tr}(\dot{\tilde{\boldsymbol{\beta}}}_G^T\tilde{\boldsymbol{\beta}}_G) - \mathbf{e}^T\mathbf{P}\mathbf{B}\left(\boldsymbol{\epsilon}_f(\mathbf{x}) + \boldsymbol{\epsilon}_G(\mathbf{x})\mathbf{u}_n\right) - \mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s\end{aligned} \tag{33}$$

where $\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_G\mathbf{u}_n = \mathrm{tr}\left(\mathbf{u}_n\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_G\right)$ is utilized.

Based on the adaptation laws (28), Eqn (33) is further formulated as,

$$\dot{V} = -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} - \mathbf{e}^T\mathbf{P}\mathbf{B}(\boldsymbol{\epsilon}_f(\mathbf{x}) + \boldsymbol{\epsilon}_G(\mathbf{x})\mathbf{u}_n) - \mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s \tag{34}$$

Substituting Eqn (30) into Eqn (34) yields

$$\begin{aligned}\dot{V} \leq &-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|\left(\|\boldsymbol{\epsilon}_f(\mathbf{x})\| + \|\boldsymbol{\epsilon}_G(\mathbf{x})\|\|\mathbf{u}_n\|\right) \\ &- \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|\left(\bar{\varepsilon}_f + \bar{\varepsilon}_G\|\mathbf{u}_n\|\right)\end{aligned} \tag{35}$$

where the inequality $\mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{G}(\mathbf{x})\mathrm{sgn}(\mathbf{B}^T\mathbf{P}\mathbf{e}) \geq \gamma_0\|\mathbf{e}^T\mathbf{P}\mathbf{B}\|$ is used.

Eqn (35) becomes

$$\begin{aligned}\dot{V} \leq &-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|(\|\boldsymbol{\epsilon}_f(\mathbf{x})\| - \bar{\varepsilon}_f) \\ &+ \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|(\|\boldsymbol{\epsilon}_G(\mathbf{x})\|\|\mathbf{u}_n\| - \bar{\varepsilon}_G\|\mathbf{u}_n\|)\end{aligned} \tag{36}$$

Notice that

$$\begin{aligned}&\|\mathbf{e}^T\mathbf{P}\mathbf{B}\|(\|\boldsymbol{\epsilon}_f(\mathbf{x})\| - \bar{\varepsilon}_f) \leq 0 \\ &\|\mathbf{e}^T\mathbf{P}\mathbf{B}\|(\|\boldsymbol{\epsilon}_G(\mathbf{x})\|\|\mathbf{u}_n\| - \bar{\varepsilon}_G\|\mathbf{u}_n\|) \leq 0\end{aligned} \tag{37}$$

Then the following inequality is obtained as,

$$\dot{V} \leq -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} \leq 0 \tag{38}$$

Hence, $\dot{V}$ is negative semidefinite and $V \in L_\infty$, which implies that the signals $\mathbf{e}$, $\tilde{\boldsymbol{\beta}}_f$ and $\tilde{\boldsymbol{\beta}}_G$ are all bounded. Since V is nonincreasing and bounded, the following result by integrating (38) with respect to time can be shown,

$$\lim_{t\to\infty}\int_0^t(-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e})dt < \infty \tag{39}$$

which implies that $\mathbf{e} \in L_2$. Because of $\mathbf{e} \in L_2 \cap L_\infty$ and $\dot{\mathbf{e}} \in L_\infty$, it can be concluded that as $t \to \infty$, $\mathbf{e} \to 0$ according to Barbalat's lemma [27,28]. This further shows that the tracking errors of the system will converge to zero. □

Remark 1: The adaptive neural control law (8) uses the robustifying term $\mathbf{u}_s$ to attenuate the effects of the modelling errors $\boldsymbol{\epsilon}_f(\mathbf{x})$ and $\boldsymbol{\epsilon}_G(\mathbf{x})$. But the knowledge of the error bounds $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_G$ is required for calculating the $\mathbf{u}_s$. In fact, these parameters are unknown and their exact values are hard to be determined. Next suitable adaptive laws will be derived for estimating the error bounds $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_G$.

3.4 Adaptive Neural Control Scheme with Unknown Error Bounds

In order to obtain a control law that guarantees control objectives without knowing the error bounds *a priori*, another adaptive control scheme is proposed here. Within this adaptive control scheme, the neural control law and the parameter adaptation laws are the same as (8) and (28). The only difference is that the error bounds $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_G$ in the robustifying term $\mathbf{u}_s$ are not constant and estimated online, which is given as,

$$\mathbf{u}_s = \mathrm{sgn}(\mathbf{B}^T\mathbf{P}\mathbf{e})\frac{(\hat{\varepsilon}_f + \hat{\varepsilon}_G\|\mathbf{u}_n\|)}{\gamma_0} \tag{40}$$

where $\hat{\varepsilon}_f$ and $\hat{\varepsilon}_G$ are the estimated values of the error bounds $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_G$. Their adaptation laws are given by,

$$\begin{aligned}\dot{\hat{\varepsilon}}_f &= \eta_3\|\mathbf{e}^T\mathbf{P}\mathbf{B}\| \\ \dot{\hat{\varepsilon}}_G &= \eta_4\|\mathbf{e}^T\mathbf{P}\mathbf{B}\|\|\mathbf{u}_n\|\end{aligned} \tag{41}$$

where η_3 and η_4 are positive constants. The property of the control scheme is described by the following theorem.

Theorem 2: Consider the MIMO nonlinear system (1). Suppose that the Assumptions 1-5 are satisfied. If the adaptive neural control law with parameter

adaptation laws (28) is chosen as (8) and the robustifying control term with the estimation laws of error bounds (41) is designed as (40), all signals in the closed-loop system are bounded and the tracking errors converge to zero as $t \to \infty$.

Proof. Consider the following Lyapunov function candidate:

$$V = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{1}{2\eta_1}\tilde{\boldsymbol{\beta}}_f^T\tilde{\boldsymbol{\beta}}_f + \frac{1}{2\eta_2}\mathrm{tr}(\tilde{\boldsymbol{\beta}}_G^T\tilde{\boldsymbol{\beta}}_G) + \frac{1}{2\eta_3}\tilde{\varepsilon}_f^2 + \frac{1}{2\eta_4}\tilde{\varepsilon}_G^2 \tag{42}$$

where $\tilde{\varepsilon}_f = \bar{\varepsilon}_f - \hat{\varepsilon}_f$ and $\tilde{\varepsilon}_G = \bar{\varepsilon}_G - \hat{\varepsilon}_G$.

Using Eqn (27), the derivative of the Lyapunov function is given as

$$\begin{aligned}
\dot{V} &= \frac{1}{2}(\dot{\mathbf{e}}^T\mathbf{P}\mathbf{e} + \mathbf{e}^T\mathbf{P}\dot{\mathbf{e}}) + \frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\beta}}}_f^T\tilde{\boldsymbol{\beta}}_f + \frac{1}{\eta_2}\mathrm{tr}(\dot{\tilde{\boldsymbol{\beta}}}_G^T\tilde{\boldsymbol{\beta}}_G) \\
&- \frac{1}{\eta_3}(\bar{\varepsilon}_f - \hat{\varepsilon}_f)\dot{\hat{\varepsilon}}_f - \frac{1}{\eta_4}(\bar{\varepsilon}_G - \hat{\varepsilon}_G)\dot{\hat{\varepsilon}}_G \\
&= \frac{1}{2}\mathbf{e}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{e} + \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_f + \frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\beta}}}_f^T\tilde{\boldsymbol{\beta}}_f + \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\tilde{\boldsymbol{\beta}}_G\mathbf{u}_n \\
&+ \frac{1}{\eta_2}\mathrm{tr}(\dot{\tilde{\boldsymbol{\beta}}}_G^T\tilde{\boldsymbol{\beta}}_G) - \mathbf{e}^T\mathbf{P}\mathbf{B}(\boldsymbol{\epsilon}_f(\mathbf{x}) + \boldsymbol{\epsilon}_G(\mathbf{x})\mathbf{u}_n) - \mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s \\
&- \frac{1}{\eta_3}(\bar{\varepsilon}_f - \hat{\varepsilon}_f)\dot{\hat{\varepsilon}}_f - \frac{1}{\eta_4}(\bar{\varepsilon}_G - \hat{\varepsilon}_G)\dot{\hat{\varepsilon}}_G
\end{aligned} \tag{43}$$

Applying Eqn (29) and the adaptation laws (28), Eqn (43) becomes as,

$$\begin{aligned}
\dot{V} = &-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} - \mathbf{e}^T\mathbf{P}\mathbf{B}(\boldsymbol{\epsilon}_f(\mathbf{x}) + \boldsymbol{\epsilon}_G(\mathbf{x})\mathbf{u}_n) - \mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{G}(\mathbf{x})\mathbf{u}_s \\
&- \frac{1}{\eta_3}(\bar{\varepsilon}_f - \hat{\varepsilon}_f)\dot{\hat{\varepsilon}}_f - \frac{1}{\eta_4}(\bar{\varepsilon}_G - \hat{\varepsilon}_G)\dot{\hat{\varepsilon}}_G
\end{aligned} \tag{44}$$

Substituting Eqn (40) into Eqn (44) yields

$$\begin{aligned}
\dot{V} \leq &-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|\left(\|\boldsymbol{\epsilon}_f(\mathbf{x})\| + \|\boldsymbol{\epsilon}_G(\mathbf{x})\|\|\mathbf{u}_n\|\right) \\
&- \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|\left(\hat{\varepsilon}_f + \hat{\varepsilon}_G\|\mathbf{u}_n\|\right) - \frac{1}{2\eta_3}(\bar{\varepsilon}_f - \hat{\varepsilon}_f)\dot{\hat{\varepsilon}}_f - \frac{1}{2\eta_4}(\bar{\varepsilon}_G - \hat{\varepsilon}_G)\dot{\hat{\varepsilon}}_G
\end{aligned} \tag{45}$$

Based on Assumption 5, Eqn (45) becomes

$$\begin{aligned}
\dot{V} \leq &-\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} + \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|(\bar{\varepsilon}_f - \hat{\varepsilon}_f) + \|\mathbf{e}^T\mathbf{P}\mathbf{B}\|(\bar{\varepsilon}_G\|\mathbf{u}_n\| - \hat{\varepsilon}_G\|\mathbf{u}_n\|) \\
&- \frac{1}{\eta_3}(\bar{\varepsilon}_f - \hat{\varepsilon}_f)\dot{\hat{\varepsilon}}_f - \frac{1}{\eta_4}(\bar{\varepsilon}_G - \hat{\varepsilon}_G)\dot{\hat{\varepsilon}}_G
\end{aligned} \tag{46}$$

Substituting Eqn (41) into Eqn (46) results in

$$\dot{V} \leq -\frac{1}{2}\mathbf{e}^T\mathbf{J}\mathbf{e} \leq 0 \tag{47}$$

Similar to the proof of Theorem 1, all the signals $\mathbf{e}, \tilde{\boldsymbol{\beta}}_f$, $\tilde{\boldsymbol{\beta}}_G$, $\tilde{\varepsilon}_f$ and $\tilde{\varepsilon}_G$ are all bounded since $\dot{V}$ is negative semidefinite. Using Barbalat's lemma [27,28], it can be seen that as $t \to \infty$, $\mathbf{e} \to 0$, which shows that the tracking errors of the system will converge to zero. □

3.5 Implementation of Adaptation Laws

To assure that the network parameters lie within the desired region of constraint as in (26), the update laws of the parameters $\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_G$ in (28) are modified according to the projection algorithm, which are given as

$$\dot{\boldsymbol{\beta}}_f = \begin{cases} -\eta_1\boldsymbol{\phi}^T(\mathbf{x})\mathbf{B}^T\mathbf{P}\mathbf{e} & \text{if} \quad \|\boldsymbol{\beta}_f\| < M_{\beta_f} \quad \text{or} \\ & \text{if} \quad \|\boldsymbol{\beta}_f\| = M_{\beta_f} \\ & \quad \text{and } \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_f \geq 0 \\ -\eta_1\boldsymbol{\Gamma}_1 & \text{if} \quad \|\boldsymbol{\beta}_f\| = M_{\beta_f} \\ & \quad \text{and } \mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_f < 0 \end{cases}$$

$$\dot{\boldsymbol{\beta}}_g = \begin{cases} -\eta_2\boldsymbol{\phi}^T(\mathbf{x})\mathbf{B}^T\mathbf{P}\mathbf{e}\mathbf{u}_n^T & \text{if} \quad \text{tr}(\boldsymbol{\beta}_G^T\boldsymbol{\beta}_G) < M_{\beta_G} \quad \text{or} \\ & \text{if} \quad \text{tr}(\boldsymbol{\beta}_G^T\boldsymbol{\beta}_G) = M_{\beta_G} \\ & \quad \text{and tr}\left(\mathbf{u}_n\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_G\right) \geq 0 \\ -\eta_2\boldsymbol{\Gamma}_2 & \text{if} \quad \text{tr}(\boldsymbol{\beta}_G^T\boldsymbol{\beta}_G) = M_{\beta_G} \\ & \quad \text{and tr}\left(\mathbf{u}_n\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_G\right) < 0 \end{cases} \tag{48}$$

where

$$\begin{aligned} \boldsymbol{\Gamma}_1 &= \boldsymbol{\phi}^T(\mathbf{x})\mathbf{B}^T\mathbf{P}\mathbf{e} - \frac{\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_f}{\|\boldsymbol{\beta}_f\|^2}\boldsymbol{\beta}_f \\ \boldsymbol{\Gamma}_2 &= \boldsymbol{\phi}^T(\mathbf{x})\mathbf{B}^T\mathbf{P}\mathbf{e}\mathbf{u}_n^T - \frac{\text{tr}(\mathbf{u}_n\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\beta}_G)}{\text{tr}(\boldsymbol{\beta}_G^T\boldsymbol{\beta}_G)}\boldsymbol{\beta}_G \end{aligned} \tag{49}$$

The utilization of projection algorithm can ensure the convergence of the parameter estimates and the estimated parameters remain bounded rather than converge to their optimal values. In this case the requirement of the persistently exciting (PE) condition is avoided when the parameters are adjusted as described in Eqn (48) during the proposed adaptive control scheme.

Remark 2: The proposed adaptive neural control scheme is illustrated in Fig. 1. This controller structure consists of two components. The first component $\mathbf{u}_n$ represents the adaptive neural control law and provides asymptotic convergence of the tracking error. The second component $\mathbf{u}_s$ represents the robustifying control law, which is used to compensate for approximation errors between the nonlinear functions of the system and the SLFNs with optimal parameters. It is noteworthy that although the form of $\mathbf{u}_s$ is consistent with the sliding-mode control term, it is different from the traditional one. The reason is that it does not guarantee that the state trajectory will slide along the manifold $\mathbf{B}^T\mathbf{P}\mathbf{e} = 0$ as traditionally guaranteed with nonadaptive sliding-mode control [29,30].

Remark 3: In the proposed control scheme, the neural control term $\mathbf{u}_n$ is constructed using the SLFNs for learning the unknown system dynamics. Different from the existing methods, the parameters of hidden nodes in SLFNs are assigned randomly based on the ELM algorithm, which simplifies the controller design procedure. Also the output weights are updated based on the stable adaptive laws (48), which guarantees the asymptotical stability of the system. Besides, to avert the priori knowledge about the approximation error bounds in calculating $\mathbf{u}_s$, they are estimated online using the estimation laws (41).

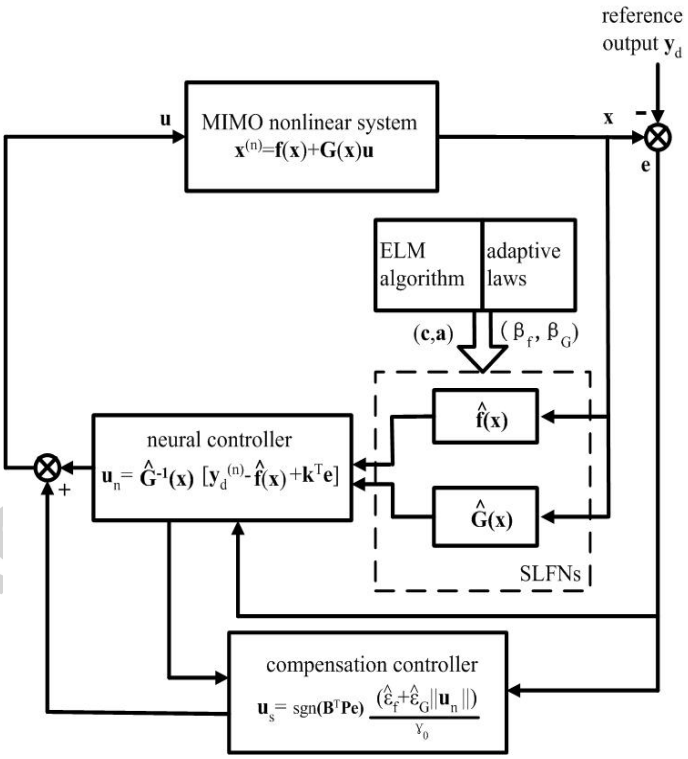


Figure 1. The overall neural control scheme.

The design steps about the proposed adaptive neural controller are summarized below.

Design of Adaptive Neural Controller

Step 1: Specify the design parameters

(1) Specify the parameter $k_{i,n_i}(i = 1, \cdots, p)$ such that all roots of the polynomial $s^n + k_{i,1}s^{(n_i-1)} + \cdots + k_{i,n_i} = 0$ are in the open left half plane.

(2) Specify a positive definite matrix $\mathbf{J}$ and obtain a symmetric matrix $\mathbf{P}$ by solving the equation $\mathbf{J} = -(\mathbf{A^T P} + \mathbf{PA})$.

(3) Specify the parameter γ_0, the learning rate $\eta_i(i = 1, \cdots, 4)$ and the number of hidden nodes $\tilde{N}$ for the practical problem.

Step 2: Construct the neural controller using ELM algorithm

(1) Assign random hidden node parameters $(\mathbf{c}_i, a_i)$, $i = 1, \cdots, \tilde{N}$.

(2) Calculate the hidden node output vector $\mathbf{H}$ for input $\mathbf{x}$:

$$\mathbf{H}(\mathbf{x}) = \mathbf{H}(\mathbf{c}_1, \cdots, \mathbf{c}_{\tilde{N}}, a_1, \cdots, a_{\tilde{N}}; \mathbf{x}) \tag{50}$$

where the details of $\mathbf{H}(\mathbf{x})$ are referred in [26,31].

(3) Construct the matrix $\boldsymbol{\phi}(\mathbf{x})$ using the hidden node output vector $\mathbf{H}(\mathbf{x})$.

(4) Calculate the approximated dynamic functions $\hat{\mathbf{f}}(\mathbf{x})$ and $\hat{\mathbf{G}}(\mathbf{x})$ based on Eqn (22) and Eqn (23).

(5) Construct the neural control input $\mathbf{u}_n$ using Eqn (8).

Step 3: Adapt the output weight parameters $\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_G$ using Eqn (48).

Step 4: Construct the robustifying control law $\mathbf{u}_s$ using Eqn (40), where the bounds of approximation errors are updated using Eqn (41).

Step 5: Obtain the feedback control input $\mathbf{u} = \mathbf{u}_n + \mathbf{u}_s$ and apply it in Eqn (3).

Step 6: Return to *Step 2* and continue from the second procedure.

In the following section, a simulation example will be performed to evaluate the proposed neural control scheme.

4 Simulations

In this section, the performance of the proposed adaptive neural control scheme is evaluated on a two-link robot manipulator used widely in the literature [32,33]. The dynamic model of the two-link robot manipulator is described by the following highly nonlinear coupled differential equation:

$$\ddot{\mathbf{q}} = -\mathbf{M}^{-1}(\mathbf{q})\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{M}^{-1}(\mathbf{q})\boldsymbol{\tau} \tag{51}$$

where $\mathbf{q} = [q_1, q_2]^T$, $\dot{\mathbf{q}} \in \mathbf{R}^2$, and $\ddot{\mathbf{q}} \in \mathbf{R}^2$ are the vectors of joint position, velocity, and acceleration respectively. $\boldsymbol{\tau} \in \mathbf{R}^2$ represents control torque supplied by the actuators.

$\mathbf{M}(\mathbf{q}) \in \mathbf{R}^{2\times 2}$ represents the inertia matrix, which is a symmetric positive-

definite matrix and given as,

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} a_1 + 2a_3\cos q_2 & a_2 + a_3\cos q_2 + a_4\sin q_2 \\ a_2 + a_3\cos q_2 + a_4\sin q_2 & a_2 \end{bmatrix}$$

$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbf{R}^{2\times 1}$ denotes the vector of centripetal, Coriolis friction forces and gravity, which is given by,

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -(a_3\sin q_2 - a_4\cos q_2)\dot{q}_1\dot{q}_2 - (a_3\sin q_2 - a_4\cos q_2)(\dot{q}_1 + \dot{q}_2)\dot{q}_2 \\ (a_3\sin q_2 - a_4\cos q_2)\dot{q}_1^2 \end{bmatrix}$$

with

$$\begin{aligned} a_1 &= I_1 + m_1 l_{c1}^2 + I_e + m_e l_{ce}^2 + m_e l_1^2 \\ a_2 &= I_e + m_e l_{ce}^2 \\ a_3 &= m_e l_1 l_{ce}\cos\delta_e \\ a_4 &= m_e l_1 l_{ce}\sin\delta_e \end{aligned}$$

where m_1 and m_e denote the masses of the two robot links, l_1 expresses the length of link 1, l_{c1} and l_{ce} are the center-of-gravity lengths of the two robot links, I_1 and I_e represent the centroidal inertial moment of the two links, δ_e is the angle of payload with respect to link 2.

Let $\ddot{\mathbf{x}} = [\ddot{q}_1, \ddot{q}_2]^T$, $\mathbf{u} = [\tau_1, \tau_2]^T$ and $\mathbf{x} = [q_1, \dot{q}_1, q_2, \dot{q}_2]^T$, Eqn (51) is expressed as,

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} \tag{52}$$

The equation is in the input-output form of (3) and also $\mathbf{G}(\mathbf{x}) = \mathbf{M}^{-1}$ is positive-definite since the matrix $\mathbf{M}$ is positive-definite. In the simulation, the parameters of the two-link robot manipulator are chosen as: $m_1 = 1$kg, $m_e = 2$kg, $l_1 = 1$m, $I_1 = 0.12$kgm^2, $I_e = 0.25$kgm^2, $\delta_e = 30^0$, $l_{c1} = 0.5$m, $l_{ce} = 0.6$m. The desired tracking trajectories are $q_{1d} = \sin(\pi t)$ and $q_{2d} = \sin(\pi t)$. The initial states are $\mathbf{x}(0) = [0.5, 0, 0.25, 0]^T$. 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 $G(\mathbf{x}, \mathbf{c}, a) = 1/(1 + \exp(-(\mathbf{c} \cdot \mathbf{x} + a)))$ are used in the simulations to design the proposed neural controller. The parameters associated with the controller design are as follows: $k_{i,n_i} = 40(i = 1, 2; n_i = 1, 2)$, $\mathbf{J} = \text{diag}[100]_4$, $\eta_1 = \eta_2 = 5$, $\eta_3 = \eta_4 = 0.001$, the initial error magnitude $\hat{\varepsilon}_f = 0$ and $\hat{\varepsilon}_G = 0$, the number of hidden nodes $\tilde{N} = 10$. The parameters $(\mathbf{c}, a)$ are chosen randomly in the intervals $[-1, 1]$ and $[0, 1]$ respectively.

The simulation results of the position and velocity tracking for link 1 and link 2 are illustrated in Fig. 2 and Fig. 3, where solid line represents the

reference signals, dash dot line represents the results using RBF hidden nodes and dash line represents the results using Sigmoidal additive hidden nodes. From Fig. 2, it can be found that the difference between the actual position and velocity of link 1 and the reference values is large at the beginning, but the system still keeps stable and the system outputs quickly converges to the desired trajectory. Similarly the actual position and velocity of link 2 both follow the reference signals very well. The control torques of link 1 and link 2 are illustrated in Fig. 4 where one can observe that the control input signals are smooth. Fig. 5 and Fig. 6 depict that the output weights ($\boldsymbol{\beta}_f$, $\boldsymbol{\beta}_G$) and the approximation error bounds ($\hat{\varepsilon}_f$, $\hat{\varepsilon}_G$) are bounded throughout the control process with stable parameter update laws. In addition, one can observe from all these figures that the simulation results achieved using the Gaussian RBF hidden nodes and Sigmoidal additive hidden nodes are similar.

The performance evaluation of the proposed adaptive neural control method under different number of hidden nodes is demonstrated in Table 1 where E_{x_1} and $E_{\dot{x}_1}$ represent the RMS errors of position and velocity for link 1 while E_{x_2} and $E_{\dot{x}_2}$ are the RMS errors of the position and velocity for link 2. E_{ave} is their average. One can observe from the table that the tracking RMS errors are insensitive to the chosen value of hidden nodes although the number of hidden nodes is varied from 5 to 30. This further shows that the designed neural controller has good robustness to the number of hidden nodes. Note that only the simulation results from the Sigmoidal additive hidden nodes are shown here due to the similar results between the RBF and Sigmoid hidden nodes. Thus the results from the RBF hidden nodes are omitted here.

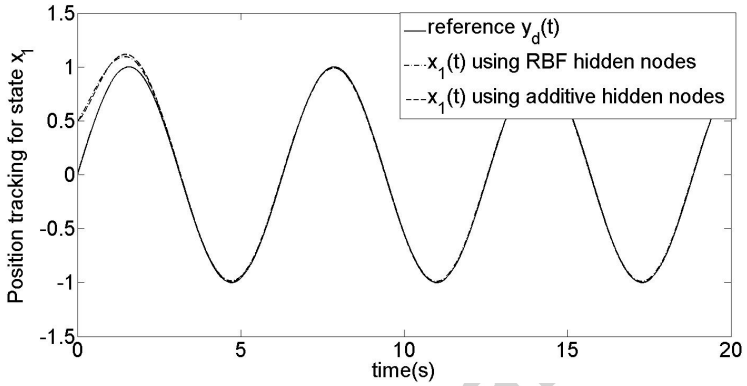
Besides, the simulation results from an adaptive fuzzy controller using fuzzy system [32] are listed here for evaluating the performance of the proposed neural control scheme. It is noteworthy that the parameters of the fuzzy system used here are consistent with those given in [32]. Figure 7 and 8 depicts the comparison performance between the two methods. It is clear that the proposed neural controller has a slower convergence speed than the fuzzy controller. But smaller velocity errors are obtained during the initial learning process compared with the fuzzy controller. Table 2 gives an overview of the comparison results from the two methods based on RMS errors. It can be seen that all the RMS errors of the neural controller are smaller than those of fuzzy controller with much lesser number of neurons.

5 Conclusions

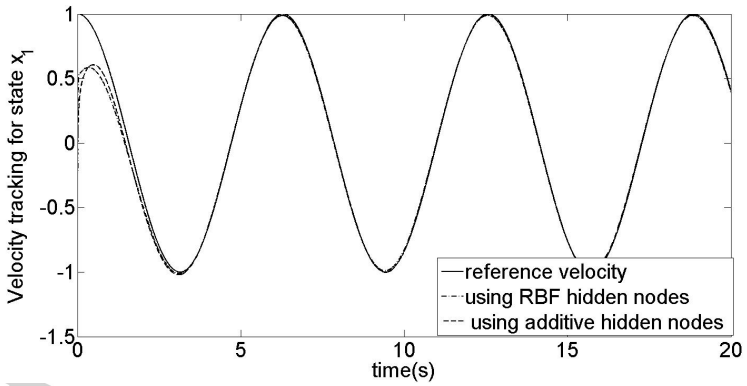
In this paper, two stable adaptive neural control schemes for a class of MIMO nonlinear systems are proposed. In these schemes, the SLFNs are used to approximate the unknown nonlinear dynamics and then the neural controller is

Table 1
Performance Evaluation under Different Number of Hidden Nodes

No. of Hidden Nodes	E_{x_1}	$E_{\dot{x}_1}$	E_{x_2}	$E_{\dot{x}_2}$	E_{ave}
5	0.1140	0.1212	0.0560	0.0628	0.0885
10	0.0954	0.1040	0.0464	0.0548	0.0752
15	0.1245	0.1378	0.0600	0.0726	0.0987
20	0.0921	0.1090	0.0446	0.0570	0.0757
25	0.0929	0.0444	0.1114	0.0601	0.0772
30	0.1105	0.1337	0.0521	0.0738	0.0925



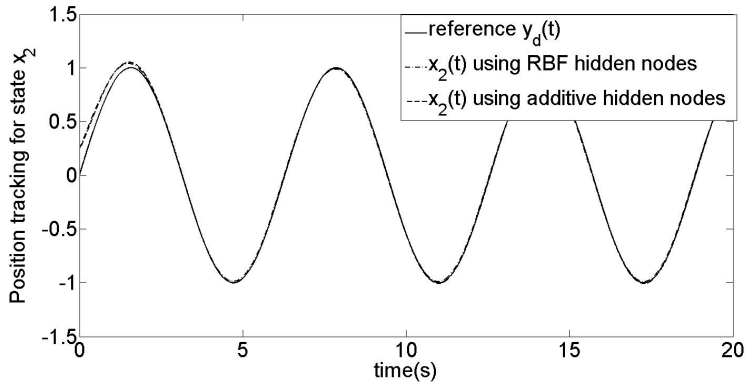
(a) Position tracking



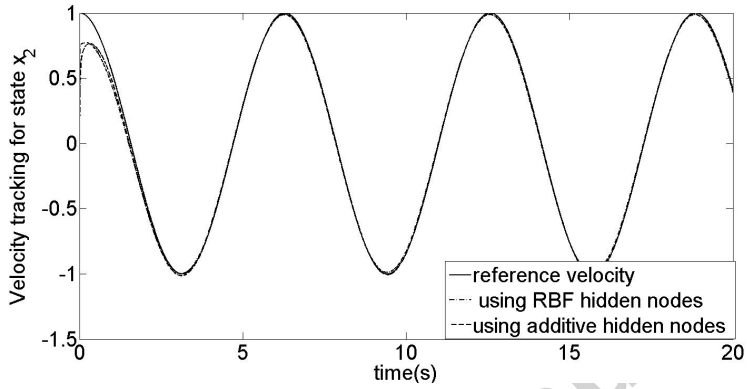
(b) Velocity tracking

Figure 2. Position and Velocity Tracking of Link 1.

built based on the approximated models. The hidden node parameters of the SLFNs are determined based on the ELM algorithm by assigning randomly the values to them. The output weights of the SLFNs are updated based on the stable adaption laws derived from the Lyapunov stability theorem and Barbalat's lemma in order to achieve the asymptotic stability of the proposed

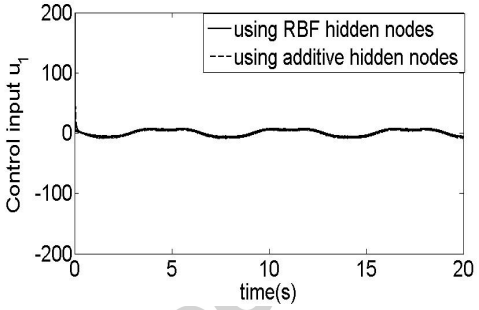


(a) Position tracking

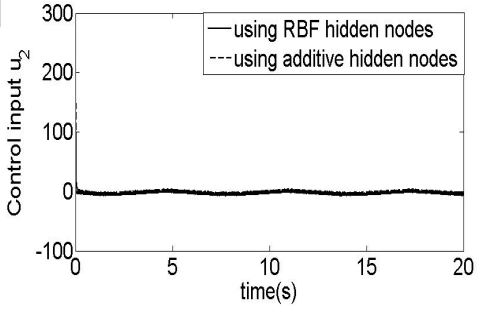


(b) Velocity tracking

Figure 3. Position and Velocity Tracking of Link 2.



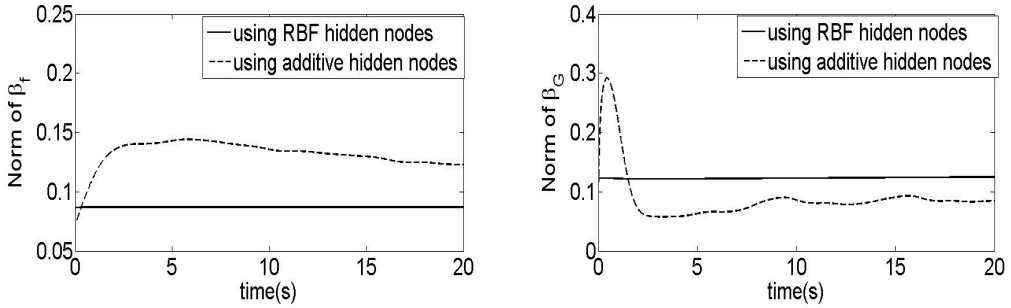
(a) Control torque of link 1



(b) Control torque of link 2

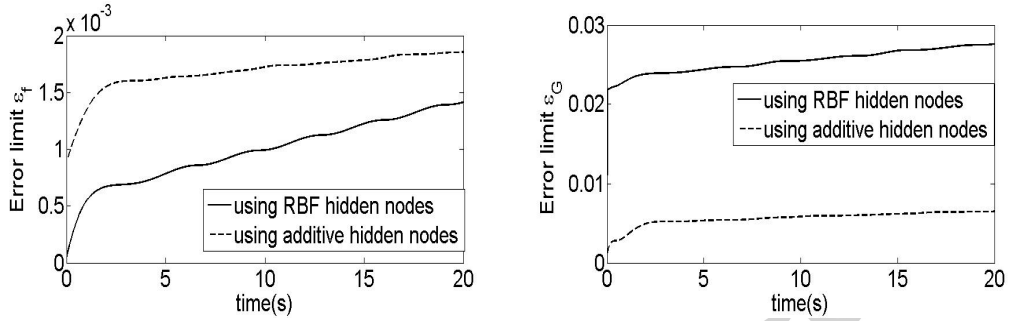
Figure 4. Control Torque of the Two Links.

control system. The robustifying control law is also incorporated with the neural controller to handle the modelling errors of the SLFNs. In order to remove the requirement about the apriori knowledge of the approximation error bounds, the estimated values are utilized in the second control scheme based on the estimation laws derived based on the Lyapunov stability theorem and Barbalat's lemma. Computer simulation studies of a two-link robot manipulator verify the adaptation and tracking performance of the proposed



(a) Norm of $\boldsymbol{\beta}_f$ (b) Norm of $\boldsymbol{\beta}_G$

Figure 5. Norm of output weights $\boldsymbol{\beta}_f$ and $\boldsymbol{\beta}_G$.



(a) Error bound of $\hat{\varepsilon}_f$ (b) Error bound of $\hat{\varepsilon}_G$

Figure 6. Neural Approximation Error Bounds $\hat{\varepsilon}_f$ and $\hat{\varepsilon}_G$.

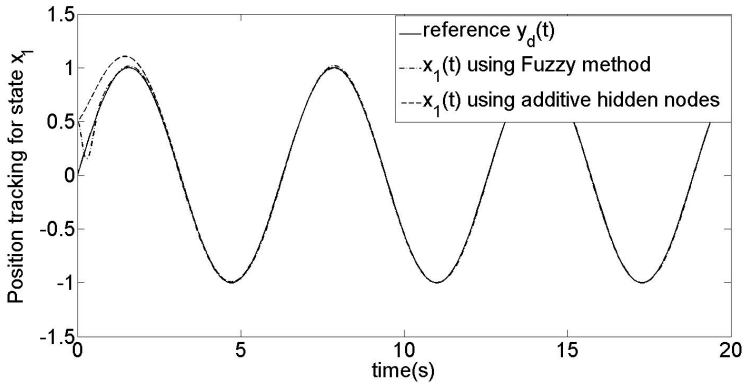
Table 2
Performance Comparison between the proposed Neural Controller and the Fuzzy Controller

Controllers	E_{x_1}	$E_{\dot{x}_2}$	E_{x_2}	$E_{\dot{x}_2}$	E_{ave}	No. of Neurons/rules
Neural	0.0954	0.1040	0.0464	0.0548	0.0752	10
Fuzzy [32]	0.1291	0.5791	0.0624	0.3544	0.2812	3^4

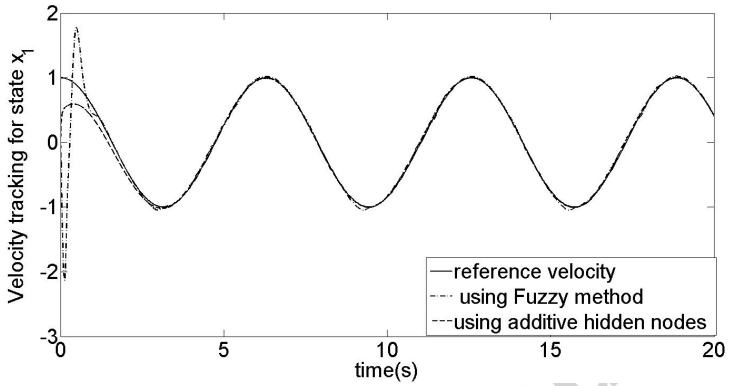
control approach. Also the simulation results demonstrate that the designed controller has good robustness with respect to the different number of hidden nodes. Moreover, Gaussian RBF hidden nodes and sigmoidal additive hidden nodes are utilized to evaluate the proposed control scheme and obtain similar results. The proposed approach presented in this paper is applicable to more general MIMO nonlinear systems satisfying certain assumptions although special application of the robot manipulator dynamics is discussed.

Acknowledgement

This work is funded in part by National Natural Science Foundation of China (Grant No. 61004055, 61105028), National Science Council of ShaanXi Province (Grant No. 2014JM8337) and the Fundamental Research Funds for the Central



(a) Position tracking



(b) Velocity tracking

Figure 7. Performance Comparison of Position and Velocity Tracking for Link 1.

Universities.

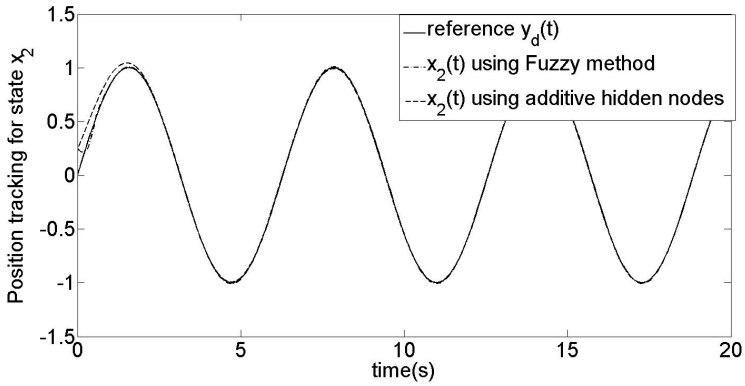
References

[1] R. Marino and P. Tomei, *Nonlinear Control Design: Geometric, Adaptive, Robust*. Prentice Hall International, UK, 1995.

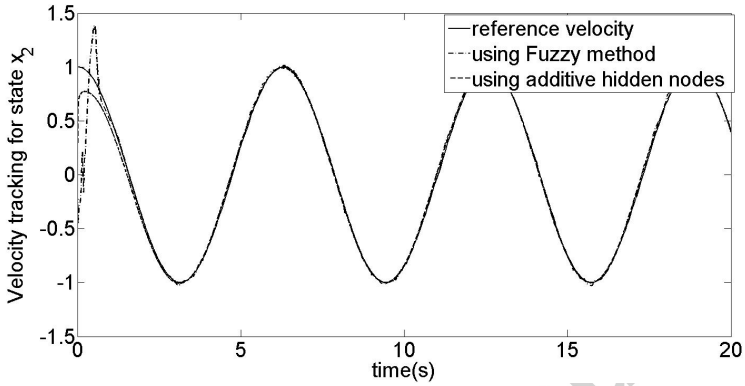
[2] B. Xu, F. Sun, H. Liu, and J. Ren, "Adaptive kriging controller design for hypersonic flight vehicle via back-stepping," *IET Control Theory and Applications*, vol. 6, no. 4, pp. 487–497, 2012.

[3] 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.

[4] S. Suresh, S. N. Omkar, V. Mani, and N. Sundararrajan, "Nonlnear adaptive neural controller for unstable aircraft," *Journal of Guidance Control and Dynamics*, vol. 28, no. 6, pp. 1103–1111, 2005.



(a) Position tracking



(b) Velocity tracking

Figure 8. Performance Comparison of Position and Velocity Tracking for Link 2.

[5] H.-J. Rong, S. Suresh, and G.-S. Zhao, "Stable indirect adaptive neural controller for a class of nonlinear system," *Neurocomputing*, vol. 74, pp. 2582–2590, 2011.

[6] H.-J. Rong and G.-S. Zhao, "Direct adaptive neural control of nonlinear systems with extreme learning machine," *Neural Computing and Applications*, vol. 22, pp. 577–586, 2013.

[7] B. Xu, F. Sun, C. Yang, D. Gao, and J. Ren, "Adaptive discrete-time controller design with neural network for hypersonic flight vehicle via back-stepping," *International Journal of Control*, vol. 84, no. 9, pp. 1543–1552, 2011.

[8] B. Xu and Z. Shi, "Universal kriging control of hypersonic aircraft model using predictor model without back-stepping," *IET Control Theory and Applications*, vol. 7, no. 4, pp. 573–583, 2013.

[9] B. Xu, D. Wang, F. Sun, and Z. Shi, "Direct neural control of hypersonic flight vehicles with prediction model in discrete time," *Neurocomputing*, vol. 115, no. 4, pp. 39–48, 2013.

[10] B. Xu, C. Yang, and Z. Shi, "Reinforcement learning output feedback NN control using deterministic learning technique," *IEEE Transactions on Neural Networks and Learning Systems, DOI: 10.1109/TNNLS.2013.2292704.*

[11] C. Kwan and F. L. Lewis, "Robust backstepping control of nonlinear systems using neural networks," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 30, no. 6, pp. 753–766, 2000.

[12] H. Xu and A. Ioannou, "Robust adaptive control for a class of MIMO nonlinear systems with guaranteed error bounds," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 728–742, 2003.

[13] M. Chen, S. S. Ge, and B. V. E. How, "Robust adaptive neural network control for a class of uncertain MIMO nonlinear systems with input nonlinearities," *IEEE Transactions on Neural Networks*, vol. 21, no. 5, pp. 796–812, 2010.

[14] C.-M. Lin, L.-Y. Chen, and C.-H. Chen, "RCMAC hybrid control for MIMO uncertain nonlinear systems using sliding-mode technology," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 708–720, 2007.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "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), pp. 985–990, 25-29 July, 2004.

[16] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004)*, pp. 6–9, Kunming, China, 2004.

[17] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[18] N.-Y. Liang, P. Saratchandran, G.-B. Huang, and N. Sundararajan, "Classification of mental tasks from EEG signals using extreme learning machine," *International Journal of Neural Systems*, vol. 16, no. 1, pp. 29–38, 2006.

[19] S. Suresh, S. Saraswathi, and N. Sundararajan, "Performance enhancement of extreme learning machine for multi-category sparse data classification problems," *Engineering Application of Artificial Intelligence*, vol. 23, no. 7, pp. 1149–1157, 2010.

[20] H.-J. Rong, O.-Y. Song, A.-H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1-3, pp. 359–366, 2008.

[21] A. A. Mohammed, R. Minhas, Q. M. J. Wu, and M. A. Sid-Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognition*, vol. 44, pp. 2588–2597, 2011.

[22] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 59, no. 8, pp. 496–500, 2012.

[23] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.

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

[25] B. Xu, Y. Pan, D. Wang, and F. Sun, "Discrete-time hypersonic flight control based on extreme learning machine," *Neurocomputing, DOI:10.1016/j.neucom.2013.02.049*.

[26] G.-B. Huang and H. A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.

[27] L.-X. Wang, "Stable adaptive fuzzy control of nonlinear systems," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 2, pp. 146–155, 1993.

[28] N. Sundararajan, P. Saratchandran, and L. Yan, *Fully tuned radial basis function neural networks for flight control*. Boston: Kluwer academic publishers, 2001.

[29] J. T. Spooner and K. M. Passino, "Stable adaptive control using fuzzy systems and neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 3, pp. 339–359, 1996.

[30] V. I. Utkin, "Variable structure systems with sliding modes," *IEEE Transactions on Automatic Control*, vol. 22, no. 2, pp. 212–222, 1977.

[31] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003.

[32] S. Labioda, M. S. Boucheritb, and T. M. Guerrac, "Adaptive fuzzy control of a class of MIMO nonlinear systems," *Fuzzy Sets and Systems*, vol. 151, pp. 59–77, 2005.

[33] H.-J. Rong, "Indirect adaptive fuzzy-neural control of robot manipulator," in *2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*, pp. 1776–1781, Liverpool, UK, 2012.