We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



5

Design of Self-Constructing Recurrent-Neural-Network-Based Adaptive Control

Chun-Fei Hsu [1] and Chih-Min Lin [2]
Chung Hua University [1], Yuan Ze University [2]
Taiwan, Republic of China

1. Introduction

Recently, neural-network-based adaptive control technique has attracted increasing attentions, because it has provided an efficient and effective way in the control of complex nonlinear or ill-defined systems (Duarte-Mermoud et al., 2005; Hsu et al., 2006; Lin and Hsu, 2003; Lin et al., 1999; Peng et al. 2004). The key elements of this success are the approximation capabilities of the neural networks. The parameterized neural networks can approximate the unknown system dynamics or the ideal tracking controller after learning. One must distinguish between two classes of control applications – open-loop identification and closed-loop feedback control. Identification applications are similar to signal processing/classification, so that the same open-loop algorithms may often be used. Therefore, a tremendous amount of training data must be used and considerable training time undertaken is required. On the other hand, in closed-loop feedback applications the neural network is inside the control loop, so that special steps must be taken to ensure that the tracking error and the neural network weights remain bounded in the closed-loop system. The basic issues in neural network closed-loop feedback control are to provide on-line learning algorithms that do not require preliminary off-line tuning. Some of these learning algorithms are based on the backpropagation algorithm. However, these approaches have difficulties to guarantee the stability and robustness of closed-loop system (Duarte-Mermoud et al., 2005; Lin et al., 1999). Another learning algorithms are based on the Lyapunov stability theorem. The tuning laws have been designed to guarantee the system stability in the Lyapunov sense (Hsu et al., 2006; Lin & Hsu, 2003; Peng et al., 2004).

However, these neural networks are feedforward neural networks; they belong to static mapping networks. Without aid of tapped delay, a feedforward neural network is unable to represent a dynamic mapping. The recurrent neural network (RNN) has superior capabilities as compared to feedforward neural networks, such as their dynamic response and their information storing ability (Lee & Teng, 2000; Lin & Hsu, 2004). Since an RNN has an internal feedback loop, it captures the dynamic response of a system with external feedback through delays. Thus, an RNN is a dynamic mapping network. Due to its dynamic characteristic and relatively simple architecture, the recurrent neural network is a useful tool for most real-time applications (Lin & Chen, 2006; Lin & Hsu, 2004; Tian et al., 2004; Wai et al. 2004).

Although the neural-network-based adaptive control performances are acceptable in above literatures; however, the learning algorithm only includes the parameter learning, and they

Open Access Database www.i-techonline.com

Source: Recurrent Neural Networks, Book edited by: Xiaolin Hu and P. Balasubramaniam, ISBN 978-953-7619-08-4, pp. 400, September 2008, I-Tech, Vienna, Austria

have not considered the structure learning of the neural network. If the number of hidden neurons is chosen too large, the computation load is heavy so that they are not suitable for practical applications. If the number of hidden neurons is chosen too small, the learning performance may be not good enough to achieve desired control performance. To tackle this problem, several self-structuring neural networks, consisting of structure and parameter learning phases, have been proposed (Huang et al., 2004; Leung & Tsoi, 2005; Lin et al, 2005). These learning phases not only decide the structure of neural network but also adjust the parameters of neural network. Recently, some self-structuring neural networks have been applied to solve several control problems (Lin et al., 2001; Gao & Er, 2003; Park et al., 2005). Lin et al. (2001) used a similarity measure method to avoid the newly generated membership function being too similar to the existing ones; however, the structure would grow large as the input data has large variations. Gao & Er (2003) proposed an error reduction ratio with QR decomposition to prune the hidden neurons; however, the design procedure is overly complex. Park et al. (2005) proposed a self-structuring neural network which can create new hidden neurons to increase the learning ability; unfortunately, the proposed approach can not avoid the structure of neural network growing unboundedly.

This paper proposes a recurrent-neural-network-based adaptive control (RNNAC) method, which combines neural-network-based adaptive control, robust control and self-structuring approach, for a class of unknown nonlinear systems. The proposed RNNAC system is composed of a neural controller and a robust controller. The neural controller uses a self-structuring recurrent neural network (SRNN) to approximate an ideal tracking controller. The learning process of SRNN includes the structure learning and parameter learning. In the structure learning, the SRNN can online create new hidden neurons as the incoming data is far away the existing hidden neurons, and cancel hidden neurons as the hidden neurons is inappropriate. Thus the learning capability and flexibility can be upgraded. In the parameter learning, the controller parameters can be online tuned based on the Lyapunov function, so that the stability of the closed-loop system can be guaranteed. The robust controller is designed to recover the residual of the approximation error to achieve L^2 tracking performance with desired attenuation level. Finally, the proposed RNNAC system is applied to control a nonlinear dynamic system. Simulation results are performed to demonstrate the effectiveness of the proposed design method.

2. Problem statement and ideal tracking control

The model of many practical nonlinear systems can be expressed in the *n*th-order form as

$$x^{(n)} = f(\mathbf{x}) + u \tag{1}$$

where $\mathbf{x} = [x, \dot{x}, \ldots, x^{(n-1)}]^T$ is the state vector of the system, which is assumed to be available for measurement, $f(\mathbf{x})$ is the nonlinear system dynamics which can be unknown, and u is the input of the system. The tracking control problem of the system is to find a control law so that the state trajectory x can track a reference command x_c closely. The tracking error is defined as

$$e = x_c - x\,. \tag{2}$$

If the exact model of the controlled system is well known, there exists an ideal tracking controller to achieve favorable control performance by possible canceling all the system

uncertainties (Slotine and Li, 1991). Assume that the parameters of the controlled system in (1) are well known, there exits an ideal tracking controller

$$u^* = -f(\mathbf{x}) + x_c^{(n)} + \mathbf{K}^T\mathbf{E} \tag{3}$$

where $\mathbf{E} = [e, \dot{e}, \ldots, e^{(n-1)}]^T$ and $\mathbf{K} = [k_n, \cdots, k_2, k_1]^T$. Applying the ideal tracking controller (3) to system (1) results in the following error dynamics

$$e^{(n)} + k_1 e^{(n-1)} + \cdots + k_n e = 0\,. \tag{4}$$

If k_i, $i = 1, 2, \cdots, n$ are chosen such that all roots of the polynomial $h(s) \triangleq s^n + k_1 s^{n-1} + \cdots + k_n$ lie strictly in the open left half of the complex plane, then it implies that $\lim_{t\to\infty} e = 0$ for any starting initial conditions. The error dynamics (4) can be rewritten in a vector form as

$$\dot{\mathbf{E}} = \mathbf{A}\mathbf{E} \tag{5}$$

where $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots 0 \\ \vdots & \ddots & \ddots & \ddots 0 \\ 0 & \cdots & 0 & 1 \\ -k_n & -k_{n-1} & \cdots & -k_1 \end{bmatrix}$. However, since the system dynamics $f(\mathbf{x})$ may be unknown or perturbed in practical applications, the ideal tracking controller (3) can not be precisely obtained.

3. Design of RNNAC

For achieving a favorable tracking performance and a specified attenuation level simultaneously, the developed recurrent-neural-network-based adaptive control (RNNAC) system with structure adaptation algorithm shown in Fig. 1 is assumed to take the following form

$$u_{anc} = u_{nc} + u_{rc} \tag{6}$$

where u_{nc} is the neural controller and u_{rc} is the robust controller. The neural controller using a self-structuring recurrent neural network (SRNN) to approximate the ideal tracking controller is the principal controller; and the robust controller is designed to achieve a specified L^2 robust tracking performance. The detail will be described as follows:

3.1 Description of SRNN

Radial basis function (RBF) networks have gained much popularity due to their ability to approximate complex nonlinear mappings directly from the input-output data with a simple topological structure. RBF is different from neural network with sigmoidal activation functions utilizing basis functions, which are locally responsive to input stimulus. Each output of RBF has a radially symmetrical response around the center vector. Although the RBF neural-network-based adaptive control performances are acceptable, the structure of the RBF network is determined by trial-and-error, and RBF network is unable to represent a dynamic mapping. To tackle this problem, a three-layer SRNN is shown in Fig. 2, which comprises of an input layer, a hidden layer with a feedback unit, and an output layer.

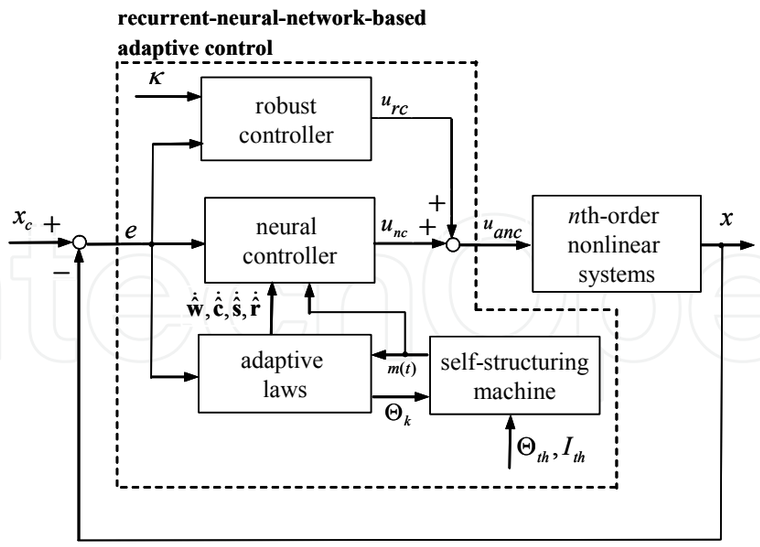


Fig. 1 Block diagram of self-constructing RNNAC system.

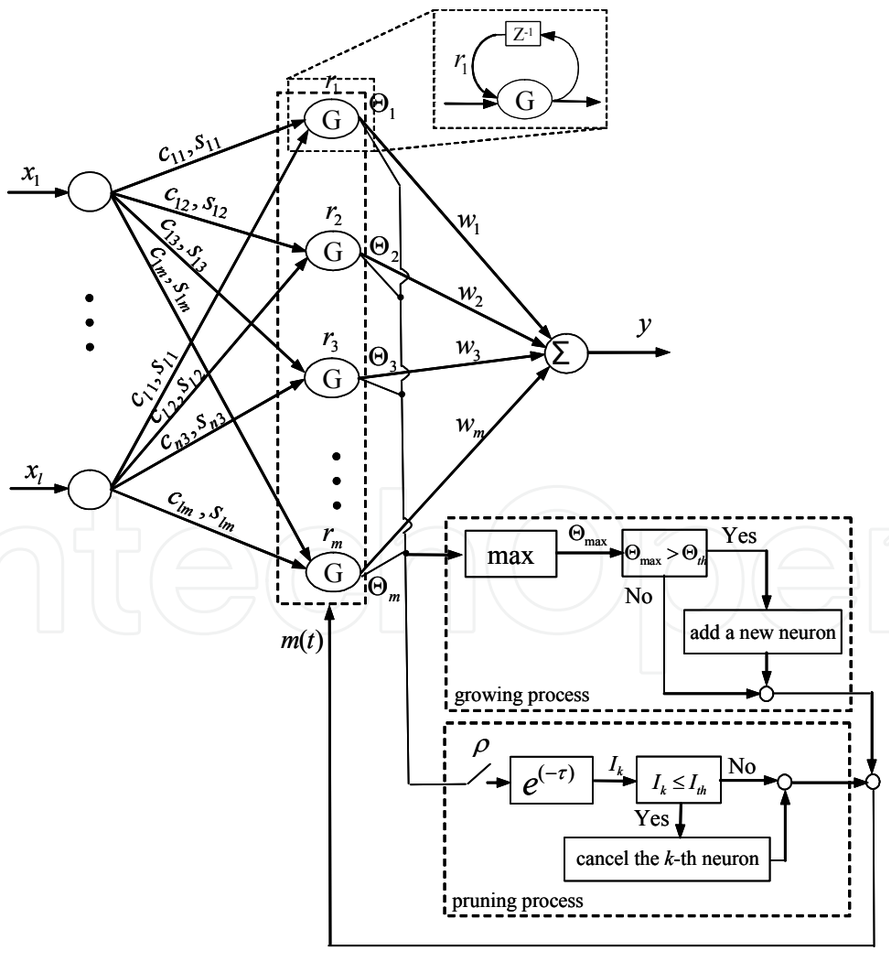


Fig. 2 The structure of self-structuring recurrent neural network.

The recurrent feedback is embedded in the network by adding feedback connections in the hidden layer. Then, the developed SRNN captures the dynamic response with external feedback through delays. The output of SRNN with m neurons for an input vector $\mathbf{x} = [x_1, x_2, ..., x_l]^T$ is given by

$$y = \sum_{k=1}^{m} w_k \Theta_k(\|\mathbf{x} - \mathbf{c}_k\|, \mathbf{s}_k, r_k) \tag{7}$$

where $\mathbf{c}_k = [c_k^1 \; c_k^2 \; .... \; c_k^l]^T$ and $\mathbf{s}_k = [s_k^1 \; s_k^2 \; .... \; s_k^l]^T$ are the center and width vectors of RBF, respectively; r_k is the internal feedback gain of RBF; w_k represents the connection weights between the hidden layer; and $\Theta_k(\|\mathbf{x} - \mathbf{c}_k\|, \mathbf{s}_k, r_k)$ represents the firing weight of the k-th hidden neuron which is given as

$$\Theta_k(\|\mathbf{x} - \mathbf{c}_k\|, \mathbf{s}_k, r_k) = \prod_{i=1}^{l} \exp[-(x_i + \Theta_{kp} r_k - c_k^i)^2 / s_k^{i\,2}] \tag{8}$$

where c_k^i and s_k^i are the center and width of RBF in the k-th term of the i-th input variable x_i, respectively; and Θ_{kp} is the output signal of the k-th hidden neuron in the previous time. Define the vectors $\mathbf{c}$, $\mathbf{s}$ and $\mathbf{r}$ collecting all parameters of the hidden layer as

$$\mathbf{c} = [\mathbf{c}_1^T \; \mathbf{c}_2^T \; .... \; \mathbf{c}_m^T]^T \tag{9}$$

$$\mathbf{s} = [\mathbf{s}_1^T \; \mathbf{s}_2^T \; .... \; \mathbf{s}_m^T]^T \tag{10}$$

$$\mathbf{r} = [r_1 \; \cdots \; r_m]^T . \tag{11}$$

Then, the output of the SRNN can be represented in a vector form

$$y(\mathbf{x}, \mathbf{c}, \mathbf{s}, \mathbf{r}, \mathbf{w}) = \mathbf{w}^T \mathbf{\Theta}(\mathbf{x}, \mathbf{c}, \mathbf{s}, \mathbf{r}) \tag{12}$$

where $\mathbf{w} = [w_1 \; w_2 \; ... w_m]^T$ and $\mathbf{\Theta} = [\Theta_1 \; \Theta_2 \; ... \; \Theta_m]^T$.

If the number of the hidden neurons m is chosen too large, the computation load is heavy so that they are not suitable for online practical applications. If the number of the hidden neurons m is chosen too small, the learning performance may be not good enough to achieve desired performance.

To solve this problem, this paper proposes an online structuring learning algorithm. The first step of the structure learning is to determine whether or not to add a new hidden neuron (Lin et al., 2001). In the growing process, the firing weight of a hidden neuron for each incoming data x_i can be represented as the degree to which the incoming data belong to the existing hidden neurons. According to the degree measure, the criterion of generating a new hidden neuron for new incoming data is described as follows. Find the maximum degree $\Theta_{\max}$ defined as

$$\Theta_{\max} = \max_{1 \le k \le m(t)} \Theta_k \tag{13}$$

where $m(t)$ is the number of the existing hidden neurons at the time *t*. It can be observed that if the maximum degree $\Theta_{\max}$ is small as the incoming data is far away the existing hidden neurons. If $\Theta_{\max} \le \Theta_{th}$ is satisfied, where $\Theta_{th} \in (0,1)$ is a pre-given threshold, then a new hidden neuron is generated. The Θ_{th} denotes the adding threshold value. If Θ_{th} is chosen to be large, the hidden neurons of SRNN can be easily created; on the other hand, if Θ_{th} is chosen to be small, the hidden neurons of SRNN can be difficulty created. For the practical implement, as the unknown control system dynamics are too complex, the Θ_{th} should be chosen as a large value so that more hidden neurons can be created to increase the learning ability. The number $m(t)$ is incremented

$$m(t+1) = m(t) + 1 . \tag{14}$$

The parameters associated with the new hidden neuron are given by

$$c_{i(m+1)}^{new} = x_i \tag{15}$$

$$s_{i(m+1)}^{new} = \sigma \tag{16}$$

$$w_{(m+1)}^{new} = r_{(m+1)}^{new} = 0 \tag{17}$$

where x_i is the new incoming data and σ is the width of a radial basis function.

Then, to prevent the structure growing unboundedly, the structure learning considers whether or not to prune the existing hidden neurons which are inappropriate. A significance of the *k*-th hidden neuron is defined as (Hsu, 2007)

$$I_k(t+1) = \begin{cases} I_k(t)\exp(-\tau), \ if\ \Theta_k < \delta \\ I_k(t) \qquad\quad , \ if\ \Theta_k \ge \delta \end{cases}, \ k = 1, ,2,..., m(t) \tag{18}$$

where the initial value of I_k is 1; δ is the threshold value; and τ is the elimination speed constant. The pruning algorithm is derived from the observation that if the significance gets fading when the firing weight Θ_k is smaller than the threshold value δ. If $I_k \le I_{th}$ is satisfied, where I_{th} a pre-given threshold, then the *k*-th hidden neuron is cancelled. I_{th} denotes the significance threshold value. If I_{th} is chosen to be large, the neurons of SRNN can be easily canceled. For practical implement, as the computation load is the important issue, I_{th} should be chosen as a large value so that more hidden neurons can be pruned. Hence, the computation load can be decreased. In summary, the flow chart of the structure learning algorithm is shown in Fig. 3. The major contribution of SRNN is that it can operate directly without spending much time on pre-determining the structuring of neural network.

3.2 SRNN approximation

Let the number of optimal hidden neurons be m^* and can divide into two parts. The first part contains m hidden neurons which are the activated part, and the secondary part contains $m^* - m$ hidden neurons which do not exist yet. Thus, by the universal approximation theorem, an optimal SRNN approximator can be designed to approximate y, such that (Park et al., 2005)

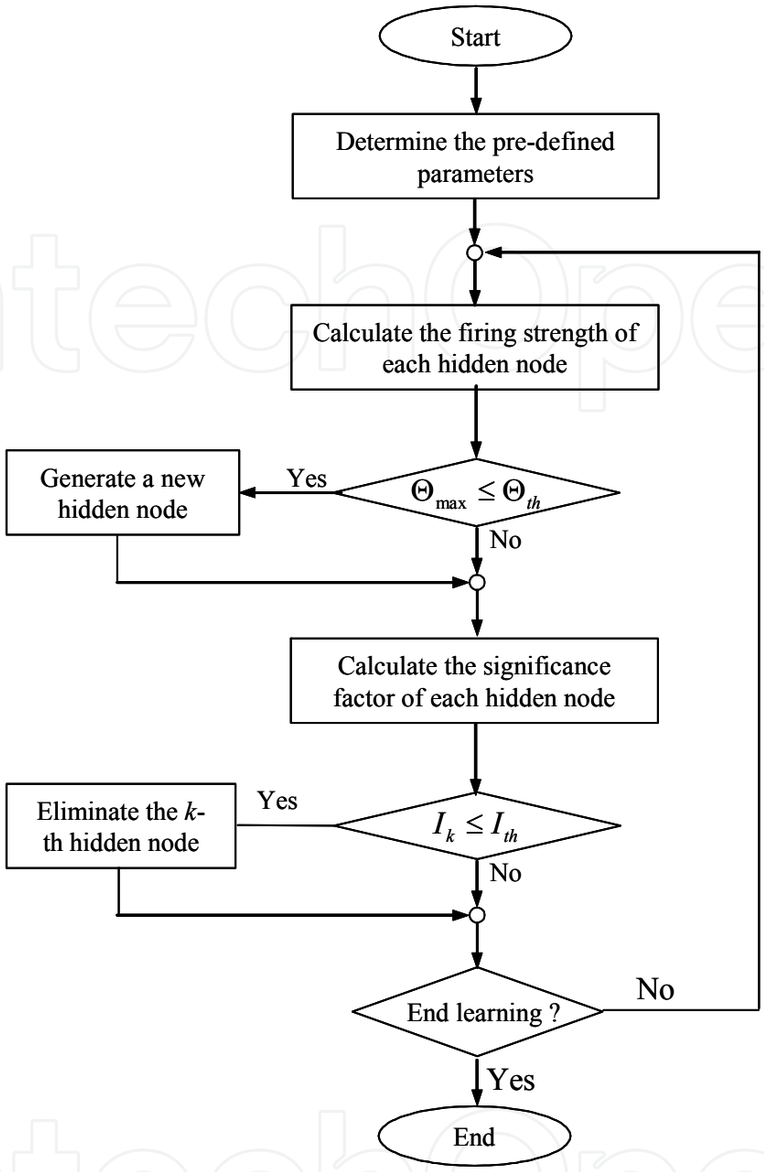


Fig. 3 The flow chart of the structure learning algorithm for SRNN.

$$y = \mathbf{w}^{*T}\mathbf{\Theta}^{*}(\mathbf{x},\mathbf{c}^{*},\mathbf{s}^{*},\mathbf{r}^{*}) + \mathbf{w}_{u}^{*T}\mathbf{\Theta}_{u}^{*}(\mathbf{x},\mathbf{c}_{u}^{*},\mathbf{s}_{u}^{*},\mathbf{r}_{u}^{*}) + \Delta \tag{19}$$

where $\mathbf{w}^{*}$, $\mathbf{\Theta}^{*}$, $\mathbf{c}^{*}$, $\mathbf{s}^{*}$ and $\mathbf{r}^{*}$ are activated parts of optimal weights; $\mathbf{w}_{u}^{*}$, $\mathbf{\Theta}_{u}^{*}$, $\mathbf{c}_{u}^{*}$, $\mathbf{s}_{u}^{*}$ and $\mathbf{r}_{u}^{*}$ are inactivated parts of optimal weights; and Δ is the approximation error. Since these optimal parameters are unobtainable, a SRNN estimator $\hat{y}$ is defined as

$$\hat{y} = \hat{\mathbf{w}}^{T}\hat{\mathbf{\Theta}}(\mathbf{x},\hat{\mathbf{c}},\hat{\mathbf{s}},\hat{\mathbf{r}}) \tag{20}$$

where $\hat{\mathbf{w}}$, $\hat{\mathbf{\Theta}}$, $\hat{\mathbf{c}}$, $\hat{\mathbf{s}}$ and $\hat{\mathbf{r}}$ are the estimated values of $\mathbf{w}^{*}$, $\mathbf{\Theta}^{*}$, $\mathbf{c}^{*}$, $\mathbf{s}^{*}$ and $\mathbf{r}^{*}$, respectively. Define the estimated error $\tilde{y}$ as

$$\tilde{y} = y - \hat{y} = \mathbf{w}^{*T}\boldsymbol{\Theta}^{*} + \mathbf{w}_{u}^{*T}\boldsymbol{\Theta}_{u}^{*} + \Delta - \hat{\mathbf{w}}^{T}\hat{\boldsymbol{\Theta}} = \tilde{\mathbf{w}}^{T}\hat{\boldsymbol{\Theta}} + \hat{\mathbf{w}}^{T}\tilde{\boldsymbol{\Theta}} + \tilde{\mathbf{w}}^{T}\tilde{\boldsymbol{\Theta}} + \mathbf{w}_{u}^{*T}\boldsymbol{\Theta}_{u}^{*} + \Delta \tag{21}$$

where $\tilde{\mathbf{w}} = \mathbf{w}^{*} - \hat{\mathbf{w}}$ and $\tilde{\boldsymbol{\Theta}} = \boldsymbol{\Theta}^{*} - \hat{\boldsymbol{\Theta}}$. In this study, a method is proposed to guarantee the closed-loop stability and perfect tracking performance, and to tune the center and the width of the radial basis function and the recurrent weight on line. For achieving this goal, linearization technique is employed to transform the nonlinear functions into partially linear form so that the expansion of $\tilde{\boldsymbol{\Theta}}$ in a Taylor series to obtain (Lin and Chen, 2006)

$$\tilde{\boldsymbol{\Theta}} = \mathbf{T}_{c}^{T}\tilde{\mathbf{c}} + \mathbf{T}_{s}^{T}\tilde{\mathbf{s}} + \mathbf{T}_{r}^{T}\tilde{\mathbf{r}} + \mathbf{h} \tag{22}$$

where $\tilde{\mathbf{c}} = \mathbf{c}^{*} - \hat{\mathbf{c}}$, $\tilde{\mathbf{s}} = \mathbf{s}^{*} - \hat{\mathbf{s}}$; $\tilde{\mathbf{r}} = \mathbf{r}^{*} - \hat{\mathbf{r}}$; $\mathbf{T}_{c} = \left[\frac{\partial\Theta_{1}}{\partial\mathbf{c}} \cdots \frac{\partial\Theta_{m}}{\partial\mathbf{c}}\right]|_{\mathbf{c}=\hat{\mathbf{c}}}$; $\mathbf{T}_{s} = \left[\frac{\partial\Theta_{1}}{\partial\mathbf{s}} \cdots \frac{\partial\Theta_{m}}{\partial\mathbf{s}}\right]|_{\mathbf{s}=\hat{\mathbf{s}}}$; $\mathbf{T}_{r} = \left[\frac{\partial\Theta_{1}}{\partial\mathbf{r}} \cdots \frac{\partial\Theta_{m}}{\partial\mathbf{r}}\right]|_{\mathbf{r}=\hat{\mathbf{r}}}$; and $\mathbf{h}$ is a vector of higher-order terms. Substituting (22) into (21), it is obtained that

$$\begin{aligned}\tilde{y} &= \tilde{\mathbf{w}}^{T}\hat{\boldsymbol{\Theta}} + \hat{\mathbf{w}}^{T}(\mathbf{T}_{c}^{T}\tilde{\mathbf{c}} + \mathbf{T}_{s}^{T}\tilde{\mathbf{s}} + \mathbf{T}_{r}^{T}\tilde{\mathbf{r}} + \mathbf{h}) + \tilde{\mathbf{w}}^{T}\tilde{\boldsymbol{\Theta}} + \mathbf{w}_{u}^{*T}\boldsymbol{\Theta}_{u}^{*} + \Delta \\ &= \tilde{\mathbf{w}}^{T}\hat{\boldsymbol{\Theta}} + \tilde{\mathbf{c}}^{T}\mathbf{T}_{c}\hat{\mathbf{w}} + \tilde{\mathbf{s}}^{T}\mathbf{T}_{s}\hat{\mathbf{w}} + \tilde{\mathbf{r}}^{T}\mathbf{T}_{r}\hat{\mathbf{w}} + \varepsilon\end{aligned} \tag{23}$$

where $\hat{\mathbf{w}}^{T}\mathbf{T}_{c}^{T}\tilde{\mathbf{c}} = \tilde{\mathbf{c}}^{T}\mathbf{T}_{c}\hat{\mathbf{w}}$, $\hat{\mathbf{w}}^{T}\mathbf{T}_{s}^{T}\tilde{\mathbf{s}} = \tilde{\mathbf{s}}^{T}\mathbf{T}_{s}\hat{\mathbf{w}}$ and $\hat{\mathbf{w}}^{T}\mathbf{T}_{r}^{T}\tilde{\mathbf{r}} = \tilde{\mathbf{r}}^{T}\mathbf{T}_{r}\hat{\mathbf{w}}$ are used since they are scales; and the uncertain term $\varepsilon \equiv \hat{\mathbf{w}}^{T}\mathbf{h} + \tilde{\mathbf{w}}^{T}\tilde{\boldsymbol{\Theta}} + \mathbf{w}_{u}^{*T}\boldsymbol{\Theta}_{u}^{*} + \Delta$.

3.3 RNNAC design

By substituting (6) into (1) and using (3) and (23), the tracking error dynamic equation can be obtained as follows

$$\begin{aligned}\dot{\mathbf{E}} &= \mathbf{A}\mathbf{E} + \mathbf{b}(u^{*} - u_{nc} - u_{rc}) \\ &= \mathbf{A}\mathbf{E} + \mathbf{b}(\tilde{\mathbf{w}}^{T}\hat{\boldsymbol{\Theta}} + \tilde{\mathbf{c}}^{T}\mathbf{T}_{c}\hat{\mathbf{w}} + \tilde{\mathbf{s}}^{T}\mathbf{T}_{s}\hat{\mathbf{w}} + \tilde{\mathbf{r}}^{T}\mathbf{T}_{r}\hat{\mathbf{w}} + \varepsilon - u_{rc})\end{aligned} \tag{24}$$

where $\mathbf{b} = [0 \ldots 0\ 1]^{T}$. In case of the existence of ε, consider a specified L^{2} tracking performance (Lee et al., 2005; Lin and Lin 2002; Wang et al., 2002)

$$\int_{0}^{T}\mathbf{E}^{T}\mathbf{Q}\mathbf{E}\,dt \le \mathbf{E}^{T}(0)\mathbf{P}\mathbf{E}(0) + \frac{\tilde{\mathbf{w}}^{T}(0)\tilde{\mathbf{w}}(0)}{\eta_{1}} + \frac{\tilde{\mathbf{c}}^{T}(0)\tilde{\mathbf{c}}(0)}{\eta_{2}} + \frac{\tilde{\mathbf{s}}^{T}(0)\tilde{\mathbf{s}}(0)}{\eta_{3}} + \frac{\tilde{\mathbf{r}}^{T}(0)\tilde{\mathbf{r}}(0)}{\eta_{4}} + \rho^{2}\int_{0}^{T}\varepsilon^{2}\,dt \tag{25}$$

where η_{1}, η_{2}, η_{3} and η_{4} are the positive constants, $T \in [0,\infty]$ and $\varepsilon \in L^{2}$. The κ is a design gain, ρ is a prescribed attenuation level, and the positive definite matrices $\mathbf{P}$ and $\mathbf{Q}$ satisfy the following Riccati-like equation

$$\mathbf{P}\mathbf{A} + \mathbf{A}^{T}\mathbf{P} + \mathbf{Q} + \mathbf{P}\mathbf{b}\left(\frac{1}{\rho^{2}} - \frac{2}{\kappa}\right)\mathbf{b}^{T}\mathbf{P} = \mathbf{0} \tag{26}$$

with $2\rho^2 \geq \kappa$. The design objective is to tune the parameters of SRNN to specify an adequate control law so that the worst effect of approximation error ε on tracking error vector $\mathbf{E}$ is guaranteed to be less than or equal to prescribed attenuation level ρ. If the system starts with initial conditions $\mathbf{E}(0)=0$, $\tilde{\mathbf{w}}(0)=0$, $\tilde{\mathbf{c}}(0)=0$, $\tilde{\mathbf{s}}(0)=0$ and $\tilde{\mathbf{r}}(0)=0$, then the L^2 tracking performance in (25) can be rewritten as

$$\sup_{\varepsilon \in L^2[0,T]} \frac{\int_0^T \mathbf{E}^T \mathbf{Q} \mathbf{E}\, dt}{\int_0^T \varepsilon^2\, dt} \leq \rho . \tag{27}$$

where the L^2-gain from ε to the tracking error $\mathbf{E}$ must be equal to or less than ρ. The following theorem can be stated and proved.

Theorem 1: Consider an *n*th-order nonlinear system expressed by (1). The control system is designed as (6), in which the adaptation laws of the neural controller are designed as

$$\dot{\hat{\mathbf{w}}} = -\dot{\tilde{\mathbf{w}}} = \eta_1 \mathbf{E}^T \mathbf{P} \mathbf{b} \hat{\boldsymbol{\Theta}} \tag{28}$$

$$\dot{\hat{\mathbf{c}}} = -\dot{\tilde{\mathbf{c}}} = \eta_2 \mathbf{E}^T \mathbf{P} \mathbf{b} \mathbf{T}_c \hat{\mathbf{w}} \tag{29}$$

$$\dot{\hat{\mathbf{s}}} = -\dot{\tilde{\mathbf{s}}} = \eta_3 \mathbf{E}^T \mathbf{P} \mathbf{b} \mathbf{T}_s \hat{\mathbf{w}} \tag{30}$$

$$\dot{\hat{\mathbf{r}}} = -\dot{\tilde{\mathbf{r}}} = \eta_4 \mathbf{E}^T \mathbf{P} \mathbf{b} \mathbf{T}_r \hat{\mathbf{w}} \tag{31}$$

and the robust controller is designed as

$$u_{rc} = \frac{1}{\kappa} \mathbf{b}^T \mathbf{P} \mathbf{E} \tag{32}$$

then the stability of the system can be guaranteed.

Proof:

Consider a Lyapunov function in the following form

$$V(\mathbf{E}, \tilde{\mathbf{w}}, \tilde{\mathbf{c}}, \tilde{\mathbf{s}}, \tilde{\mathbf{r}}) = \frac{1}{2} \mathbf{E}^T \mathbf{P} \mathbf{E} + \frac{\tilde{\mathbf{w}}^T \tilde{\mathbf{w}}}{2\eta_1} + \frac{\tilde{\mathbf{c}}^T \tilde{\mathbf{c}}}{2\eta_2} + \frac{\tilde{\mathbf{s}}^T \tilde{\mathbf{s}}}{2\eta_3} + \frac{\tilde{\mathbf{r}}^T \tilde{\mathbf{r}}}{2\eta_4} . \tag{33}$$

Differentiating (33) with respect to time and using (24) and (28) ~ (31), it can be obtained that

$$\dot{V}(\mathbf{E}, \tilde{\mathbf{w}}, \tilde{\mathbf{c}}, \tilde{\mathbf{s}}, \tilde{\mathbf{r}}) = \frac{1}{2} \dot{\mathbf{E}}^T \mathbf{P} \mathbf{E} + \frac{1}{2} \mathbf{E}^T \mathbf{P} \dot{\mathbf{E}} + \frac{\tilde{\mathbf{w}}^T \dot{\tilde{\mathbf{w}}}}{\eta_1} + \frac{\tilde{\mathbf{c}}^T \dot{\tilde{\mathbf{c}}}}{\eta_2} + \frac{\tilde{\mathbf{s}}^T \dot{\tilde{\mathbf{s}}}}{\eta_3} + \frac{\tilde{\mathbf{r}}^T \dot{\tilde{\mathbf{r}}}}{\eta_4}$$

$$= \frac{1}{2} \mathbf{E}^T (\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A}) \mathbf{E} + \mathbf{E}^T \mathbf{P} \mathbf{b} (\tilde{\mathbf{w}}^T \hat{\boldsymbol{\Theta}} + \tilde{\mathbf{c}}^T \mathbf{T}_c \hat{\mathbf{w}} + \tilde{\mathbf{s}}^T \mathbf{T}_s \hat{\mathbf{w}} + \tilde{\mathbf{r}}^T \mathbf{T}_r \hat{\mathbf{w}} + \varepsilon - u_{rc})$$

$$+ \frac{\tilde{\mathbf{w}}^T \dot{\tilde{\mathbf{w}}}}{\eta_1} + \frac{\tilde{\mathbf{c}}^T \dot{\tilde{\mathbf{c}}}}{\eta_2} + \frac{\tilde{\mathbf{s}}^T \dot{\tilde{\mathbf{s}}}}{\eta_3} + \frac{\tilde{\mathbf{r}}^T \dot{\tilde{\mathbf{r}}}}{\eta_4}$$

$$= \frac{1}{2}\mathbf{E}^T(\mathbf{A}^T\mathbf{P} + \mathbf{PA})\mathbf{E} + \tilde{\mathbf{w}}^T(\mathbf{E}^T\mathbf{Pb}\hat{\boldsymbol{\Theta}} + \frac{\dot{\tilde{\mathbf{w}}}}{\eta_1}) + \tilde{\mathbf{c}}^T(\mathbf{E}^T\mathbf{PbT}_c\hat{\mathbf{w}} + \frac{\dot{\tilde{\mathbf{c}}}}{\eta_2})$$

$$+ \tilde{\mathbf{s}}^T(\mathbf{E}^T\mathbf{PbT}_s\hat{\mathbf{w}} + \frac{\dot{\tilde{\mathbf{s}}}}{\eta_3}) + \tilde{\mathbf{r}}^T(\mathbf{E}^T\mathbf{PbT}_r\hat{\mathbf{w}} + \frac{\dot{\tilde{\mathbf{r}}}}{\eta_4}) + \mathbf{E}^T\mathbf{Pb}(\varepsilon - u_{rc})$$

$$= \frac{1}{2}\mathbf{E}^T(\mathbf{A}^T\mathbf{P} + \mathbf{PA})\mathbf{E} + \mathbf{E}^T\mathbf{Pb}(\varepsilon - u_{rc}) . \tag{34}$$

Using (26) and (32), equation (34) can be rewritten as

$$\dot{V}(\mathbf{E}, \tilde{\mathbf{w}}, \tilde{\mathbf{m}}, \tilde{\mathbf{s}}, \tilde{\mathbf{r}}) = \frac{1}{2}\mathbf{E}^T(\mathbf{A}^T\mathbf{P} + \mathbf{PA} - \frac{2}{\kappa}\mathbf{Pbb}^T\mathbf{P})\mathbf{E} + \frac{1}{2}\varepsilon^T\mathbf{b}^T\mathbf{PE} + \frac{1}{2}\mathbf{E}^T\mathbf{Pb}\varepsilon$$

$$= \frac{1}{2}\mathbf{E}^T(-\mathbf{Q} - \frac{1}{\rho^2}\mathbf{Pbb}^T\mathbf{P})\mathbf{E} + \frac{1}{2}\varepsilon^T\mathbf{b}^T\mathbf{PE} + \frac{1}{2}\mathbf{E}^T\mathbf{Pb}\varepsilon$$

$$= -\frac{1}{2}\mathbf{E}^T\mathbf{QE} - \frac{1}{2}(\frac{1}{\rho}\mathbf{b}^T\mathbf{PE} - \rho\varepsilon)^T(\frac{1}{\rho}\mathbf{b}^T\mathbf{PE} - \rho\varepsilon) + \frac{1}{2}\rho^2\varepsilon^2$$

$$\le -\frac{1}{2}\mathbf{E}^T\mathbf{QE} + \frac{1}{2}\rho^2\varepsilon^2 \tag{35}$$

where $(\frac{1}{\rho}\mathbf{b}^T\mathbf{PE} - \rho\varepsilon)^T(\frac{1}{\rho}\mathbf{b}^T\mathbf{PE} - \rho\varepsilon) \ge 0$ and $\varepsilon^T\mathbf{b}^T\mathbf{PE} = \mathbf{E}^T\mathbf{Pb}\varepsilon$ are used. Integrating the above equation from $t = 0$ to $t = T$, yields

$$V(T) - V(0) \le -\frac{1}{2}\int_0^T \mathbf{E}^T\mathbf{QE}\, dt + \frac{1}{2}\rho^2\int_0^T \varepsilon^2\, dt \tag{36}$$

Since $V(T) \ge 0$, the above inequality implies the following inequality

$$\frac{1}{2}\int_0^T \mathbf{E}^T\mathbf{QE}\, dt \le V(0) + \frac{1}{2}\rho^2\int_0^T \varepsilon^T\varepsilon\, dt \tag{37}$$

Using (34), this inequality is equivalent to inequality (25). Since $V(0)$ is finite if the approximation error $\varepsilon \in L^2$, that is $\int_0^T \varepsilon^2 d\tau < \infty$, it implies that $\lim_{t\to\infty}|\mathbf{E}| = 0$.

In the following, the design algorithm of RNNAC with structure adaptation algorithm is summarized as follows:

Step 1: Initialize the pre-defined parameters of RNNAC.

Step 2: The tracking error is given in (2).

Step 3: The neural controller is given as (20), where the parameter are estimated by (28)-(31), respectively.

Step 4: The robust controller is given as (32).

Step 5: The control law is given as (6).

Step 6: Determine whether or not to add a new hidden neuron by $\Theta_{\max} \le \Theta_{th}$ condition, and determine whether or not to cancel a existing node by a significance index I_k.

Step 7: Return to Step 2.

4. Simulation results

Consider a second-order chaotic system such as the Duffing's equation describing a special nonlinear circuit or a pendulum moving in a viscous medium (Chen and Dong, 1993; Jiang, 2002)

$$\ddot{x} = f(\mathbf{x}) + u \tag{38}$$

where $f(\mathbf{x}) = -p\dot{x} - p_1 x - p_2 x^3 + q\cos(\omega t)$ is the system dynamics, t is the time variable, ω is the frequency, u is the control effort and p, p_1, p_2 and q are real constants. The chaotic dynamic system can be observed in many nonlinear circuits and mechanical systems.

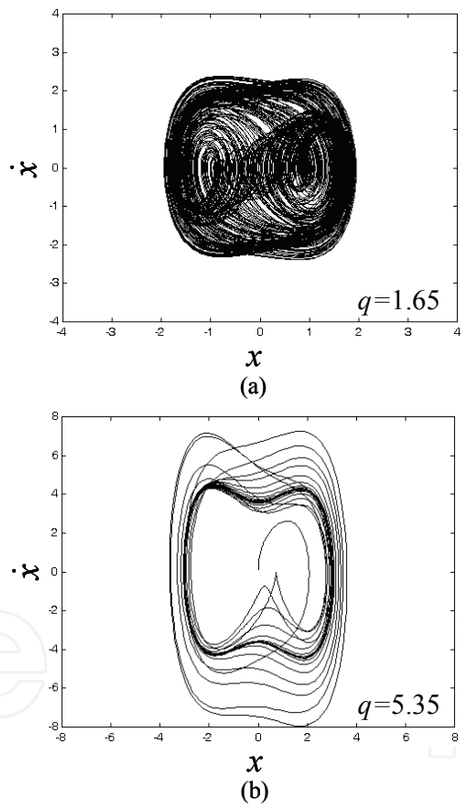


Fig. 4 Phase plane of uncontrolled chaotic system.

Recently, control of the chaotic dynamic system has become a significant research topic in the physics, mathematics and engineering communities. Chaotic dynamic system is a nonlinear deterministic system that displays complex, noisy-like and unpredictable behavior. Depending on the choice of these constants, it is known that the solutions of (38) may exhibit periodic, almost periodic and chaotic behavior. For observing the chaotic unpredictable behavior, the open-loop system behavior with $u = 0$ was simulated with

$p = 0.4$, $p_1 = -1.1$, $p_2 = 1.0$ and $\omega = 1.8$. The phase plane plots from an initial condition point (0, 0) are shown in Figs. 4(a) and 4(b) for $q = 1.65$ (chaotic) and $q = 5.35$ (period 1), respectively (Chen and Dong, 1993). It is shown that the uncontrolled chaotic dynamic system has different chaotic trajectories with different q values. The interest in the chaotic equation is the problem of how to design a controller to drive a chaotic trajectory to track a reference command closely.

The proposed RNNAC with structure adaptation algorithm is applied to control a nonlinear dynamic system. It should be emphasized that the development of the proposed control method does not need to know the system dynamics of the control system. A SRNN approximator is used to online estimate an ideal tracking controller with the online structuring and parameter learning algorithms. The structure learning possesses the ability of both adding and pruning hidden neurons, and the parameter learning adjusts the interconnection weights of neural network to achieve favorable approximation performance. The parameters of RNNAC are selected as $k_1 = 1$, $k_2 = 2$, $\eta_1 = 50$, $\eta_2 = \eta_3 = \eta_4 = 10$, $\sigma = 2.0$, $\Theta_{th} = 0.5$, $\tau = 0.01$, $\delta = 0.2$, and $I_{th} = 0.1$. The choices of these values are through some trials to achieve satisfactory control performance considering the requirement of stability and possible operating conditions. Properly choosing the values of k_1 and k_2, the desired system dynamics such as rise time, overshoot, and settling time can be easily designed by the second-order system shown in (4). The parameters η_1, η_2, η_3 and η_4 are the leaning rates of the interconnection weights. If the leaning rates are chosen to be small, then the parameters convergence of RNNAC will be easily achieved; however, this will result in slow learning speed. On the other hand, if the leaning rates are chosen to be large, then the learning speed will be fast; however, the RNNAC system may become more unstable for the parameter convergence. For a choice of $\mathbf{Q} = \mathbf{I}$, solve the Riccati-like equation (26) with $2\rho^2 = \kappa$, then

$$\mathbf{P} = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}. \tag{39}$$

The simulation results of the RNNAC system with $\kappa = 1.0$ for $q = 1.65$ and $q = 5.35$ are shown in Figs. 5 and 6, respectively. The tracking responses of state x are shown in Figs. 5(a) and 6(a); the tracking responses of state $\dot{x}$ are shown in Figs. 5(b) and 6(b); the associated control efforts are shown in Figs. 5(c) and 6(c); and the numbers of hidden neurons are shown in Figs. 5(d) and 6(d), respectively. Simulation results show that the robust tracking performance of the proposed RNNAC system has been achieved. To attenuate an arbitrarily desired level via L^2 tracking design technique as small as possible. The simulation results of the proposed RNNAC system with $\kappa = 0.1$ for $q = 1.65$ and $q = 5.35$ are shown in Figs. 7 and 8, respectively. The tracking responses of state x are shown in Figs. 7(a) and 8(a); the tracking responses of state $\dot{x}$ are shown in Figs. 7(b) and 8(b); the associated control efforts are shown in Figs. 7(c) and 8(c); and the numbers of hidden neurons are shown in Figs. 7(d) and 8(d), respectively. From these simulation results, it can be seen that robust tracking performance can be also achieved without any knowledge of system dynamic functions; moreover, better system performance can be achieved as soon as the robust gain κ is decreased.

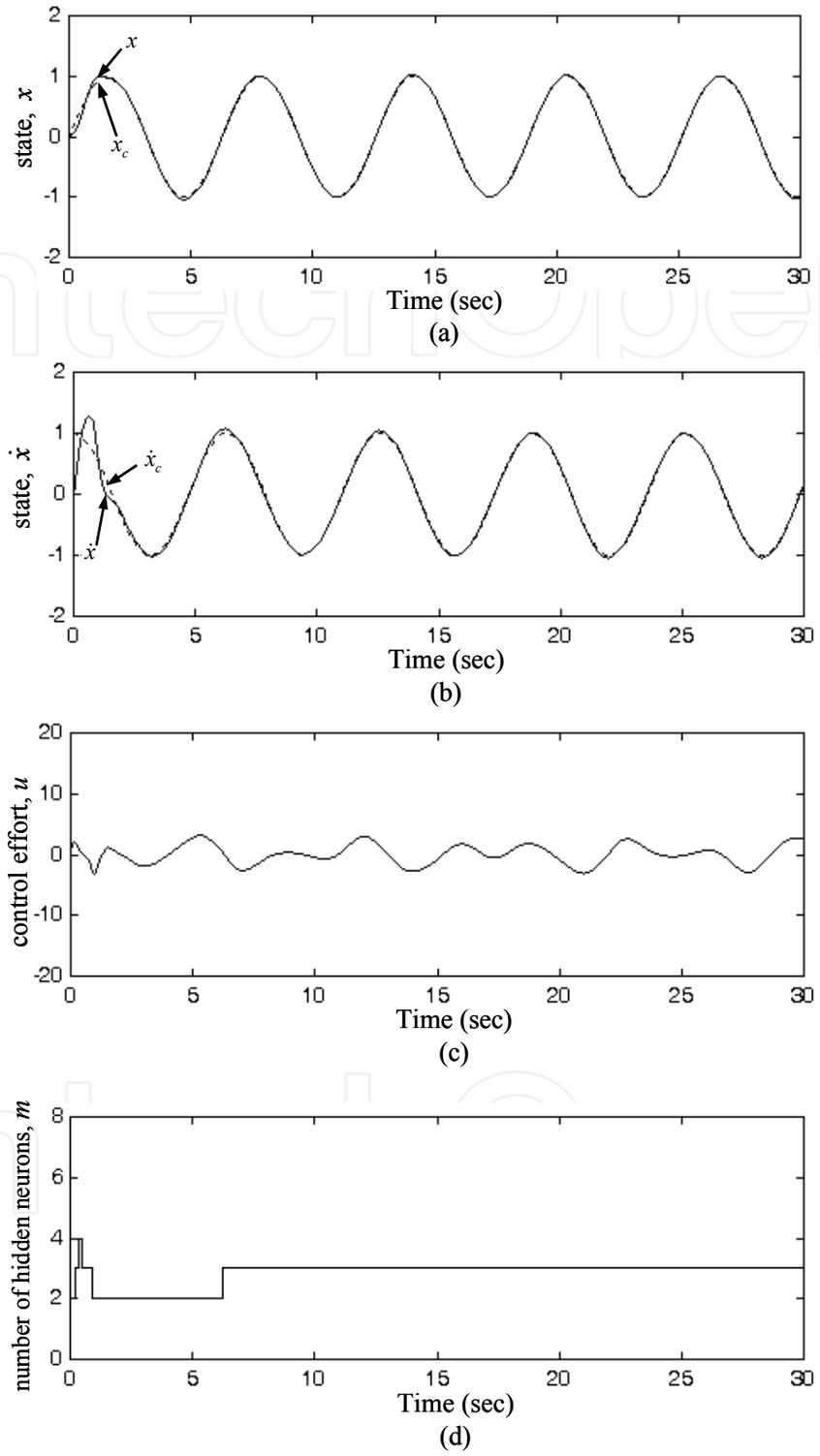


Fig. 5 Simulation results for $q = 1.65$ with $\kappa = 1.0$.

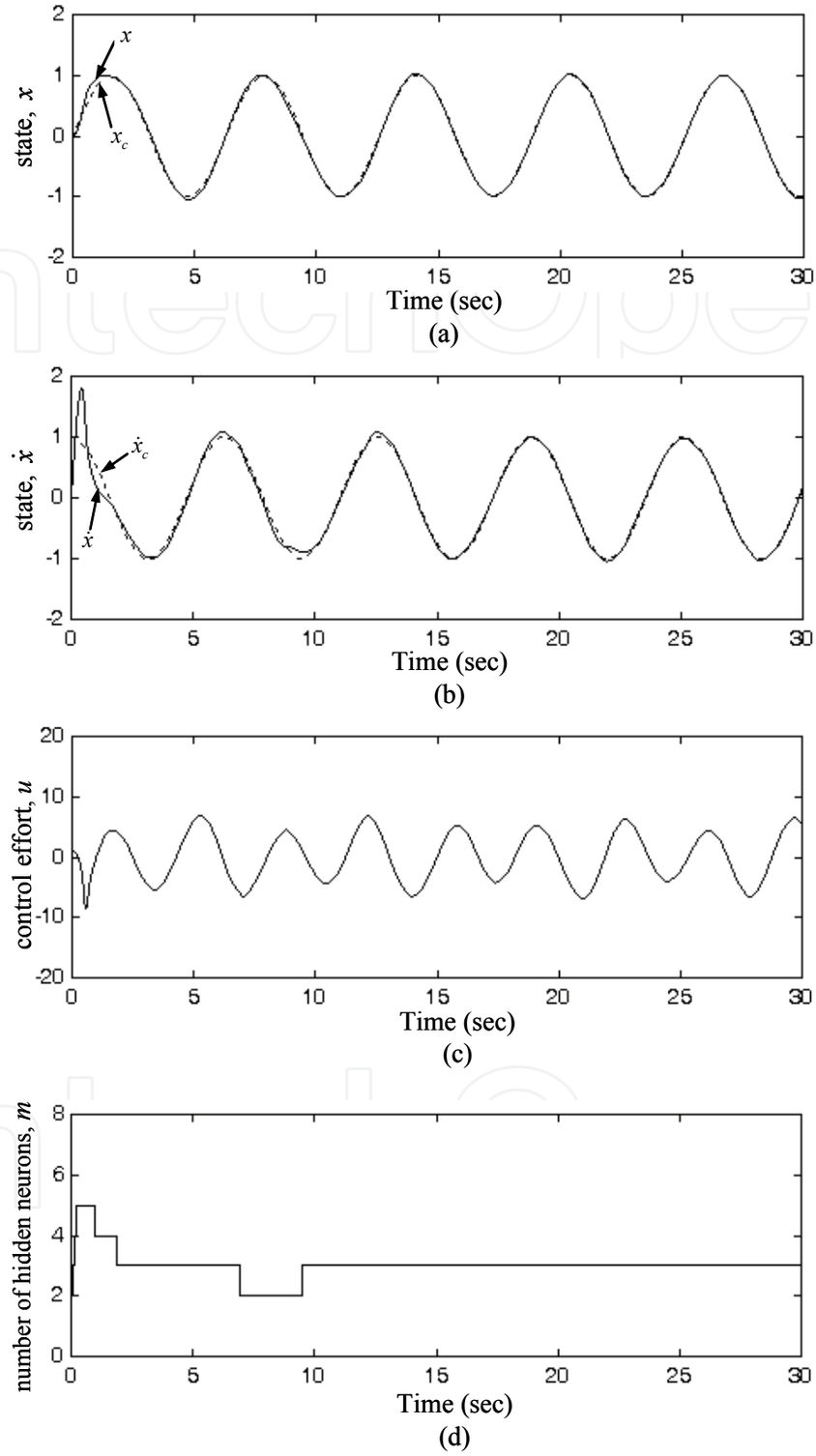


Fig. 6 Simulation results for $q = 5.35$ with $\kappa = 1.0$.

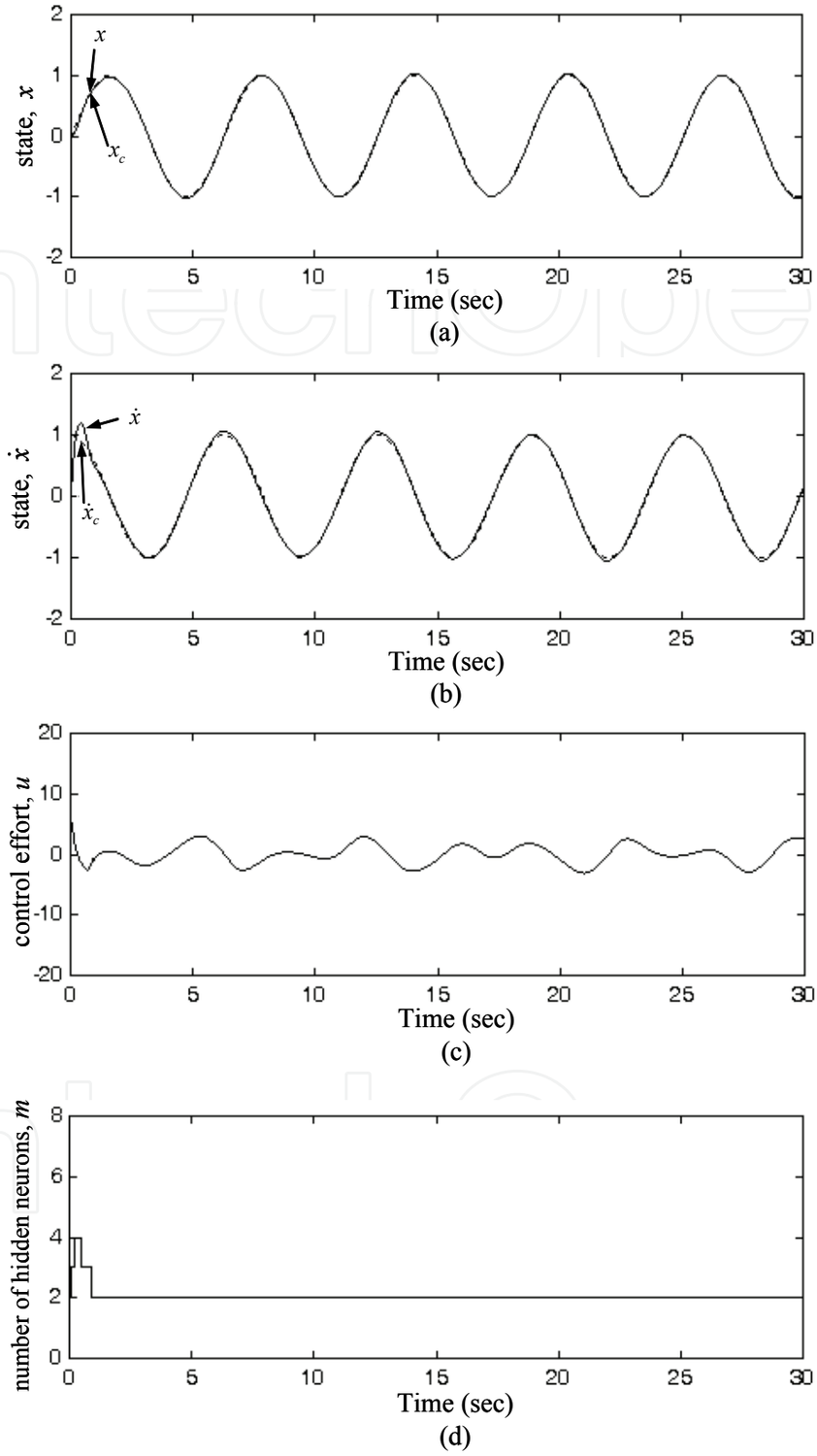


Fig. 7. Simulation results for $q = 1.65$ with $\kappa = 0.1$.

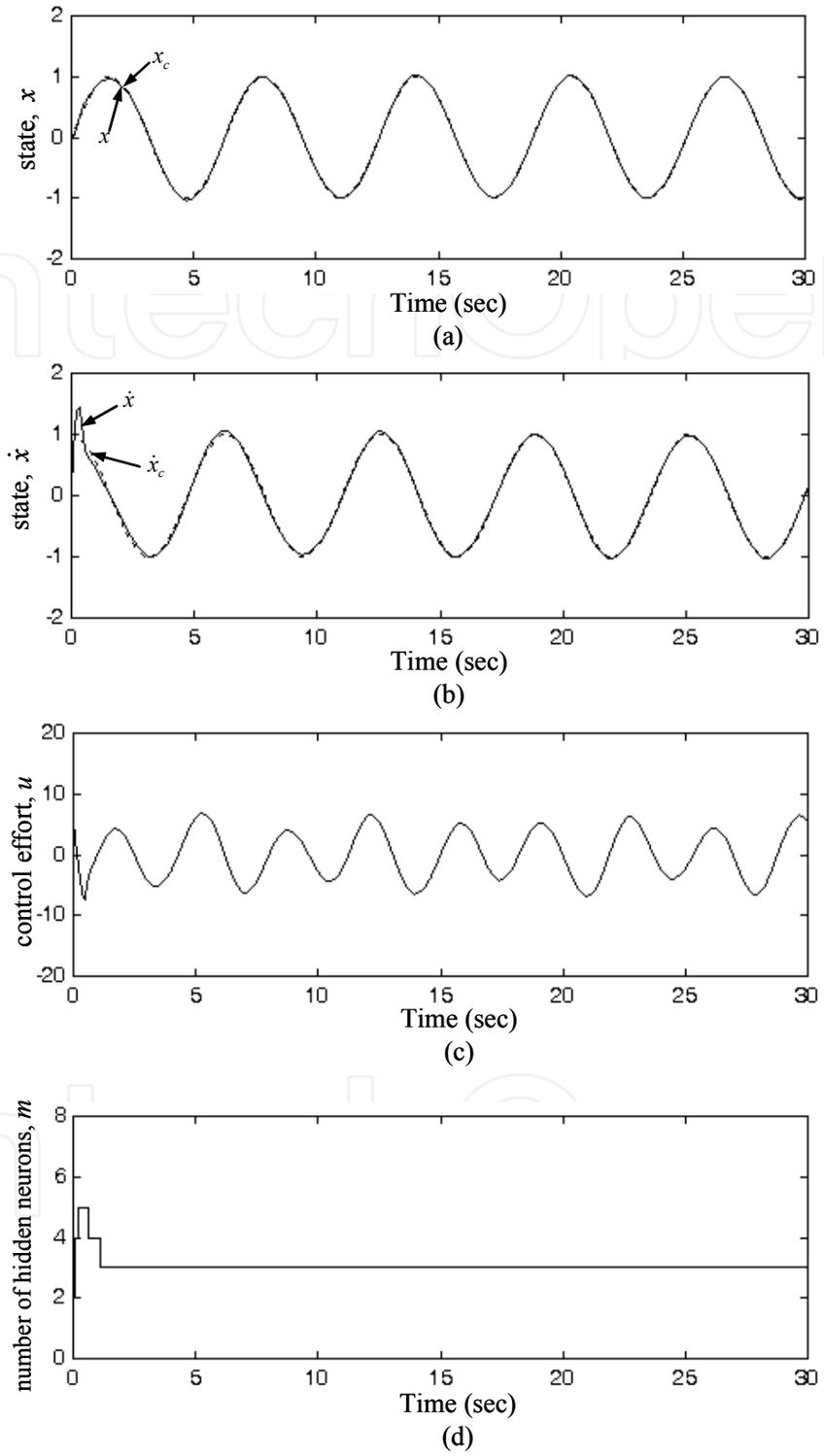


Fig. 8 Simulation results for $q = 5.35$ with $\kappa = 0.1$.

5. Conclusions

This paper develops a recurrent-neural-network-based adaptive control (RNNAC) system with structure adaptation algorithm, which is composed of a neural controller and a robust controller. In the neural controller design, a self-structuring recurrent neural network (SRNN) is utilized to mimic an ideal tracking controller. In the SRNN approximator, a dynamic generating and pruning mechanism of the neural stricture is developed to cope with the tradeoff between the approximation accuracy and computation load. The robust controller is designed to attenuate the effects of the approximation error on the tracking performance using L^2 tracking technique. Finally, the developed RNNAC system is used to control a nonlinear chaotic dynamic system to demonstrate its effectiveness. Simulation results indicate that a small attenuation level can be achieved if the magnitude of weighting factor κ is chosen small.

6. Acknowledgment

The authors appreciate the partial financial support from the National Science Council of Republic of China under grant NSC 95-2622-E-155-004-CC3.

7. References

Chen G, Dong X (1993) On feedback control of chaotic continuous-time systems. IEEE Trans Circuits Syst I 40 (9): 591-601

Duarte-Mermoud MA, Suarez AM, Bassi DF (2005) Multivariable predictive control of a pressurized tank using neural networks. Neural Comput Appl 15 (1): 18-25

Gao Y, Er MJ (2003) *Online adaptive fuzzy neural identification and control of a class of MIMO nonlinear systems.* IEEE Trans Fuzzy Syst 11 (4): 462-477

Huang GB, Saratchandran P, Sundararajan N (2004) An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern B Cybern 34 (6): 2284-2292

Hsu CF, Lin CM, and Lee TT (2006) Wavelet adaptive backstepping control for a class of nonlinear systems. IEEE Trans Neural Netw 17 (5): 1175-1183

Hsu CF (2007) Self-organizing adaptive fuzzy neural control for a class of nonlinear systems. IEEE Trans Neural Netw 18 (4): 1232-1241

Jiang ZP (2002) Advanced feedback control of the chaotic Duffing equation. IEEE Trans Circuits Syst I 49 (2): 244-249

Lee CH, Teng CC (2000) Indentification and control of dynamic systems using recurrent fuzzy neural networks. IEEE Trans Fuzzy Syst 8 (4): 349-366

Lee TS, Lin CH, Lin FJ (2005) An adaptive H^∞ controller design for permanent magnet synchronous motor drives. Control Eng Pract 13 (4): 425-439

Leung CS, Tsoi AC (2005) Combined learning and pruning for recurrent radial basis function networks based on recursive least square algorithms. Neural Comput Appl 15 (1): 62-78

Lin CL, Lin TY (2002) Approach to adaptive neural net-based H^∞ control design. IEE Proc Control Theory Appl 149 (4): 331-342

Lin CM, Hsu CF (2003) Neural network hybrid control for antilock braking systems. IEEE Trans Neural Netw 14 (2): 351-359

Lin CM, Hsu CF (2004) Supervisory recurrent fuzzy neural network control of wing rock for slender delta wings. IEEE Trans Fuzzy Syst 12 (5): 733-742

Lin CM, Chen, CH (2006) Adaptive RCMAC sliding mode control for uncertain nonlinear systems. Neural Comput Appl 15 (3): 253-267

Lin CT, Cheng WC, Liang SF (2005) *An on-line ICA-mixture-model-based self-constructing fuzzy neural network.* IEEE Trans Circuits Syst I 52 (1): 207-221

Lin FJ, Hwang WJ, Wai RJ (1999) A supervisory fuzzy neural network control system for tracking periodic inputs. IEEE Trans Fuzzy Syst 7 (1): 41-52

Lin FJ, Lin CH, Shen PH (2001) Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. IEEE Trans Fuzzy Syst 9 (5): 751-759

Park JH, Huh SH, Kim SH, Seo SJ, Park GT (2005) Direct adaptive controller for nonaffine nonlinear systems using self-structuring neural networks. IEEE Trans Neural Netw 16 (2): 414-422

Peng YF, Wai RJ, Lin CM (2004) Implementation of LLCC-resonant driving circuit and adaptive CMAC neural network control for linear piezoelectric ceramic motor. IEEE Trans Ind Electron 51 (1): 35-48

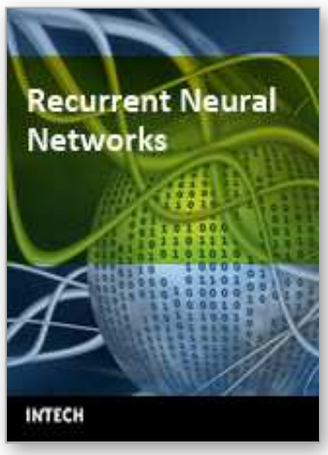
Tian L; Wang J, Mao Z (2004) *Constrained motion control of flexible robot manipulators based on recurrent neural networks.* IEEE Trans Syst Man Cybern B Cybern 34 (3): 1541-1552

Slotine J-JE, Li WP (1991) Applied Nonlinear Control. Prentice-Hall, Englewood Cliffs, NJ

Wai RJ, Lin CM, Peng YF (2004) Adaptive hybrid control for linear piezoelectric ceramic motor drive using diagonal recurrent CMAC network. IEEE Trans Neural Netw 15 (6): 1491-1506

Wang WY, Chan ML, Hsu CCJ, Lee TT (2002) H^{∞} tracking-based sliding mode control for uncertain nonlinear systems via an adaptive fuzzy-neural approach. IEEE Trans Syst Man Cybern B Cybern 32 (4): 483-492

IntechOpen



Recurrent Neural Networks

Edited by Xiaolin Hu and P. Balasubramaniam

ISBN 978-953-7619-08-4
Hard cover, 400 pages
Publisher InTech
Published online 01, September, 2008
Published in print edition September, 2008

The concept of neural network originated from neuroscience, and one of its primitive aims is to help us understand the principle of the central nerve system and related behaviors through mathematical modeling. The first part of the book is a collection of three contributions dedicated to this aim. The second part of the book consists of seven chapters, all of which are about system identification and control. The third part of the book is composed of Chapter 11 and Chapter 12, where two interesting RNNs are discussed, respectively.The fourth part of the book comprises four chapters focusing on optimization problems. Doing optimization in a way like the central nerve systems of advanced animals including humans is promising from some viewpoints.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chun-Fei Hsu and Chih-Min Lin (2008). Design of Self-Constructing Recurrent-Neural-Network-Based Adaptive Control, Recurrent Neural Networks, Xiaolin Hu and P. Balasubramaniam (Ed.), ISBN: 978-953-7619-08-4, InTech, Available from: http://www.intechopen.com/books/recurrent_neural_networks/design_of_self-constructing_recurrent-neural-network-based_adaptive_control

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen