A NONLINEAR CONTROL STRATEGY BASED ON USING A SHAPE-TUNABLE NEURAL CONTROLLER

CHYI-TSONG CHEN AND SHIH-TIEN PENG

*Department of Chemical Engineering, Feng Chia University,
Taichung, Taiwan 407, Republic of China*

WEI-DER CHANG

*Department of Automatic Control, Feng Chia University, Taichung,
Taiwan 407, Republic of China*

Key Words: Nonlinear Control Strategy, Shape-Tunable Neural Controller, Learning Algorithm, Steepest Descent Method, Nonlinear CSTR

In this paper, a nonlinear control strategy based on using a shape-tunable neural network is developed for adaptive control of nonlinear processes. Based on the steepest descent method, a learning algorithm that enables the neural controller to possess the ability of automatic controller output range adjustment is derived. The novel feature of automatic output range adjustment provides the neural controller more flexibility and capability, and therefore the scaling procedure, which is usually unavoidable for the conventional fixed-shape neural controllers, becomes unnecessary. The advantages and effectiveness of the proposed nonlinear control strategy are demonstrated through the challenge problem of controlling an open-loop unstable nonlinear continuous stirred tank reactor (CSTR).

Introduction

Most of the industrial processes are highly nonlinear in nature due to their intrinsic nonlinear dynamics and external actuator saturations, etc. The existence of nonlinearities may result in variety and complex behavior patterns that are difficult to control and make many of established linear control strategies inadequately. Therefore, controlling highly nonlinear processes presents a great challenge and has received considerable interest from control engineers (Bequette, 1991).

In recent years, the advances in the area of artificial neural networks have provided a promising approach to nonlinear process control (Bhat and McAvoy, 1990; Hunt *et al.*, 1992; Willis *et al.*, 1992). In designing a conventional back-propagation neural controller with a fixed-shape activated function, one often encounters the difficulty of selecting an optimal scaling factor for the controller output range adjustment (Cui and Shin, 1993). This is because that a fixed-shape activated function used in the output layer forces the neural network outputs to fall into a certain range. If an inadequate scaling factor is used, the fixed-shape neural controller may produce an inappropriate control input to a process or may lead to instability. However, the optimal scaling factor is system dependent and difficult to be determined exactly, especially for those process systems having highly nonlinear and/or uncertain

dynamics. A recent result of Yamada and Yabuta (1992) reconciles the above problem and revealed that the stability of a controlled system is closely related to the optimal activated function shape.

So far, limited studies have been devoted to the problem of applying both the flexibility and the nonlinear capability of neural networks provided by tuning the shape of nonlinear activated function to dynamic system controllers (Kawato *et al.*, 1987; Arai *et al.*, 1991; Yamada and Yabuta, 1992; Hu and Shao, 1992). Kawato *et al.* (1987) determined several parameters such as the optimum shape of the sigmoid functions (nonlinear activated function) by a trial and error procedure. These parameters thus obtained will prohibit the advanced application of neural network for control. With regard to the problem of self-adjusting the optimal nonlinear activated function shape, Arai *et al.* (1991) proposed an autotuning method for the slope of the sigmoid function. Their study focused on the issue of image compression and cannot be directly applied to the control system since the whole system may become unstable during the tuning process (Yamada and Yabuta, 1992). To confirm the stability conditions for linear neural network and object plants, Yamada and Yabuta (1992) proposed a special sigmoid function $h(x)=2(1-e^{-bx})/[b(1+e^{-bx})]$ and further presented an autotuning method for adjusting the parameter b . Since the slope and the saturated level of the nonlinear activated function are related and governed by a single parameter b , their approach may in general restrict the structure of the optimum shape of the nonlinear activated function. Recently, Hu and Shao (1992)

* Received on December 2, 1996. Correspondence concerning this article should be addressed to C.-T. Chen.

constructed a learning algorithm based on introducing the generalized S-shape activated function $h(x)=a/(1+e^{-bx})$. Their algorithm allows the saturation level a to be adjusted together with the slope b . However, the output of this function lies in the range of $[0, a]$, which is either positive or negative. It is therefore in some cases inconvenient for direct control applications. In order to overcome the above-mentioned disadvantages and to provide a more flexible nonlinear ability for neural network, Chen and Chang (1996) developed a more general feedforward neural network of which a two-parameter hyperbolic tangent function is used to describe the nonlinearity of each neuron. Based on the steepest descent method, they derived an autotuning algorithm by which the optimal shape of each activated function is tracked automatically such that the difficulty in selecting scaling factors is avoided. They showed by numerical examples that the shape-tuning ability enhances the nonlinear capability for neural networks significantly and provides more excellent convergent properties than conventional fixed-shape multilayered neural networks even though fewer nodes are used.

In this paper, we adopt the shape-tunable neural network that recently developed by Chen and Chang (1996) to develop a nonlinear control strategy for direct nonlinear process control. A specialized learning algorithm is proposed such that the shape-tunable neural network, which is used as the process feedback controller, is able to control nonlinear processes adaptively by simply observing the process output errors. The only *a priori* knowledge of the controlled process is the gradient information of its output response. To demonstrate the applicability and advantages of the proposed scheme, the case study of controlling an open-loop unstable nonlinear CSTR was worked out. It stresses on the following important issues: the comparison with a conventional fixed-shape neural controller, the capability of handling hard constraint on control input, the rejection of unmeasured disturbance and measurement noise, the robustness of the scheme, the effect of plant/model mismatch, and the ability to the unmodeled direct process control.

The remainder of this paper is organized as follows: In section 1, we formulate a nonlinear process control problem and introduce a neural network based control system architecture. In section 2, we present the shape-tunable neural controller and then derive a learning algorithm for on-line controller parameter adjustment. To demonstrate the proposed control scheme, the challenge problem of bring a nonlinear CSTR from a stable operating point to an unstable one was worked out in section 3. Some related and important issues are investigated subsequently. Finally, some concluding remarks are made.

1. The Control System and Problem Statement

Consider the nonlinear process system described by the following dynamic equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (1a)$$

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \quad (1b)$$

where $\mathbf{x} \in R^n$ is the vector of states, $\mathbf{y} \in R^q$ the vector of system outputs, and $\mathbf{u} \in R^p$ the vector of control inputs. Assume that the nonlinear vector functions $\mathbf{f}$ and $\mathbf{g}$ are continuous and are differentiable with respect to all of their arguments. As is usually the case, only the system output is assumed to be measured. Let the desired system output $y_d(t)$ be piecewisely continuous, then the control problem to be addressed is to determine the control input $\mathbf{u}(t)$ for the nonlinear process system Eq. (1) such that the system output $\mathbf{y}(t)$ can track the desired output $y_d(t)$ asymptotically.

The main purpose of this paper is to design a shape-tunable neural controller as the replacement of conventional ones to produce suitable control inputs $\mathbf{u}(t)$ adaptively for system Eq. (1) to achieve the goal mentioned above. Since only the process output error is assumed to be available, the neural controller is implemented in a conventional feedback control configuration and functions as a nonlinear mapping from the process output error to the control input. For the purpose of generating suitable control inputs, it would be interesting if the neural controller could learn to control the nonlinear process system in an interactive and autonomous way by simply observing the system outputs and by continuously adapting itself to the task.

2. A Nonlinear Control Strategy

2.1 A shape-tunable neural controller

To possess the ability of automatic controller output range adjustment, a feedforward shape-tunable neural network (Chen and Chang, 1996) is suggested for nonlinear control application. The novel feature of the shape-tuning is achieved through introducing a two-parameter hyperbolic tangent function

$$h(x) = \frac{a(1 - e^{-bx})}{1 + e^{-bx}} \quad (2)$$

as the nonlinear activated function to describe the output characteristics of each neuron. In Eq. (2), the parameter a represents the saturated level and b the gain factor (slope). Clearly, the nonlinear characteristics of the activated function in each neuron is

Fig. 1 The characteristics of the shape-tunable activated function --- $a=1$; — $a=5$

governed by the parameters a and b . The alterations of parameters a and b will result in a range change of the neuron output value. The characteristic behavior of such an activated function is depicted in Fig. 1. It should be noted that the shape-tuning of the activated function distinguishes the neural network a major difference with the conventional fixed-shape ones.

An m -layer feedforward neural network with shape-tunable activated function is suggested to be the process controller. To clearly explain the signal transmission in a shape-tunable neural controller, let's take the i -th neuron in the l -th layer into consideration. The neuron carries out two actions. The first action is to calculate the weighted sum $S_{i,j}$ of the inputs as

$$S_{i,l} = \sum_{j=1}^{N_{l-1}} w_{i,j,l} o_{j,l-1} - \theta_{i,l} \quad (3)$$

Next, the output of each neuron is calculated based on the shape-tunable activated function

$$o_{i,l} = \frac{a_{i,l}(1 - e^{-b_{i,l}S_{i,l}})}{1 + e^{-b_{i,l}S_{i,l}}} \quad (4)$$

where $a_{i,l}$ and $b_{i,l}$ are the saturated level and gain factor of the activated function used in i -th neuron of layer l , respectively. Here, it should be noted that a conventional fixed-shape neural controller can only produce a certain range of controller outputs according to the fix-shaped activated function used. To adjust the output range for providing suitable control inputs, a scaling factor should be used. In stead of using a scaling factor, the shape-tunable neural controller possesses the ability of controller output range adjustment by tuning the parameters a

and b . As a result, the neural controller can be adopted for various problems.

2.2 A learning algorithm

For control purpose, the neural controller must be told what the values of the controller parameters, $a_{i,l}$, $b_{i,l}$, $w_{i,j,l}$ and $\theta_{i,l}$, should be in order to adapt the changing process to have desired outputs. However, in practice, only the process output errors are available so that to provide a suitable set of controller parameters adaptively a learning scheme should be applied.

Before attempting to derive a specialized learning algorithm, we define an error (the system performance) function as follows:

$$\begin{aligned} E(t) &= \frac{1}{2} \sum_{i=1}^q (y_{id}(t) - y_i(t))^2 \\ &= \frac{1}{2} \sum_{i=1}^q e_i^2(t) \end{aligned} \quad (5)$$

where $y_{id}(t)$ and $y_i(t)$ are the desired and actual values of the i -th process output, respectively. The objective of learning is to determine how to increase or decrease the controller parameters $w_{i,j,l}$, $\theta_{i,l}$, $a_{i,l}$ and $b_{i,l}$ in order to minimize the error function $E(t)$. This can be done by using the steepest descent gradient rule (Rumelhart and McClelland, 1986) formulated as follows:

$$\Delta w_{i,j,l} = -\eta \frac{\partial E}{\partial w_{i,j,l}} \quad (6a)$$

$$\Delta \theta_{i,l} = -\eta \frac{\partial E}{\partial \theta_{i,l}} \quad (6b)$$

$$\Delta a_{i,l} = -\eta \frac{\partial E}{\partial a_{i,l}} \quad (6c)$$

$$\Delta b_{i,l} = -\eta \frac{\partial E}{\partial b_{i,l}} \quad (6d)$$

where η is called the learning rate. The value of η used, in general, has a profound effect on the local properties of the surface of the error function and determines the speed of convergence. It should be noted that for convenience the learning rate used in Eqs. (6a) – (6d) is set identical. Now, by using the chain rule, the derivatives of the error function with respect to the controller parameters can be explicitly expressed as:

$$\frac{\partial E}{\partial w_{i,j,l}} = \sum_{v=1}^q \left\{ \frac{\partial E}{\partial y_v} \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial o_{\mu,m}} \frac{\partial o_{\mu,m}}{\partial w_{i,j,l}} \right) \right\} \quad (7a)$$

$$\frac{\partial E}{\partial \theta_{i,l}} = \sum_{v=1}^q \left\{ \frac{\partial E}{\partial y_v} \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial o_{\mu,m}} \frac{\partial o_{\mu,m}}{\partial \theta_{i,l}} \right) \right\} \quad (7b)$$

$$\frac{\partial E}{\partial a_{i,l}} = \sum_{v=1}^q \left\{ \frac{\partial E}{\partial y_v} \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial o_{\mu,m}} \frac{\partial o_{\mu,m}}{\partial a_{i,l}} \right) \right\} \quad (7c)$$

$$\frac{\partial E}{\partial b_{i,l}} = \sum_{v=1}^q \left\{ \frac{\partial E}{\partial y_v} \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial o_{\mu,m}} \frac{\partial o_{\mu,m}}{\partial b_{i,l}} \right) \right\} \quad (7d)$$

where $o_{\mu,m}$ is the μ -th output of the last layer. Using the chain rule once again, the partial derivatives of $o_{\mu,m}$ with respect to controller parameters can be found as:

$$\frac{\partial o_{\mu,m}}{\partial w_{i,j,l}} = \frac{\partial o_{\mu,m}}{\partial S_{i,l}} \frac{\partial S_{i,l}}{\partial w_{i,j,l}} \quad (8a)$$

$$\frac{\partial o_{\mu,m}}{\partial \theta_{i,l}} = \frac{\partial o_{\mu,m}}{\partial S_{i,l}} \frac{\partial S_{i,l}}{\partial \theta_{i,l}} \quad (8b)$$

$$\frac{\partial o_{\mu,m}}{\partial a_{i,l}} = \frac{\partial o_{\mu,m}}{\partial o_{i,l}} \frac{\partial o_{i,l}}{\partial a_{i,l}} \quad (8c)$$

$$\frac{\partial o_{\mu,m}}{\partial b_{i,l}} = \frac{\partial o_{\mu,m}}{\partial o_{i,l}} \frac{\partial o_{i,l}}{\partial b_{i,l}} \quad (8d)$$

Consequently, from Eqs. (3) and (4) and performing appropriate derivatives, we have the partial derivatives in the above equations as:

$$\frac{\partial S_{i,l}}{\partial w_{i,j,l}} = o_{j,l-1} \quad (9)$$

$$\frac{\partial S_{i,l}}{\partial \theta_{i,l}} = -1 \quad (10)$$

$$\frac{\partial o_{i,l}}{\partial a_{i,l}} = \frac{o_{i,l}}{a_{i,l}} \quad (11)$$

$$\frac{\partial o_{i,l}}{\partial b_{i,l}} = \frac{a_{i,l} S_{i,l}}{2} \left(1 - \frac{o_{i,l}}{a_{i,l}} \right) \left(1 + \frac{o_{i,l}}{a_{i,l}} \right) \quad (12)$$

and

$$\frac{\partial o_{i,l}}{\partial S_{i,l}} = \frac{1}{2} a_{i,l} b_{i,l} \left(1 - \frac{o_{i,l}}{a_{i,l}} \right) \left(1 + \frac{o_{i,l}}{a_{i,l}} \right) \quad (13)$$

Denoting the relationships between the hidden layers and the output layer as

$$\frac{\partial o_{\mu,m}}{\partial S_{i,l}} = \delta_{\mu,i,l} \quad (14)$$

and

$$\frac{\partial o_{\mu,m}}{\partial o_{i,l}} = \xi_{\mu,i,l} \quad (15)$$

then the values of $\delta_{\mu,i,l}$ and $\xi_{\mu,i,l}$ can be determined,

respectively, by

$$\delta_{\mu,i,l} = \frac{1}{2} \xi_{\mu,i,l} a_{i,l} b_{i,l} \left(1 - \frac{o_{i,l}}{a_{i,l}} \right) \left(1 + \frac{o_{i,l}}{a_{i,l}} \right) \quad (16)$$

and

$$\xi_{\mu,i,l} = \begin{cases} 1, & \text{if } l=m, \text{ and } \mu=i \\ 0, & \text{if } l=m, \text{ and } \mu \neq i \\ \sum_{j=1}^{N_{l+1}} \delta_{\mu,j,l+1} w_{j,i,l+1}, & \text{if } l < m \end{cases} \quad (17)$$

Finally, by substituting Eqs. (9) – (17) into Eqs. (8a) – (8d) and then incorporating the resulting equations into Eqs. (6a) – (6d), we have the following complete learning algorithm:

$$w_{i,j,l}(t+T) = w_{i,j,l}(t) + \eta \sum_{v=1}^q \left\{ e_v(t) \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial u_{\mu}} \delta_{\mu,i,l} o_{j,l-1} \right) \right\} \quad (18a)$$

$$\theta_{i,l}(t+T) = \theta_{i,l}(t) - \eta \sum_{v=1}^q \left\{ e_v(t) \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial u_{\mu}} \delta_{\mu,i,l} \right) \right\} \quad (18b)$$

$$a_{i,l}(t+T) = a_{i,l}(t) + \eta \sum_{v=1}^q \left\{ e_v(t) \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial u_{\mu}} \xi_{\mu,i,l} \frac{o_{i,l}}{a_{i,l}} \right) \right\} \quad (18c)$$

$$b_{i,l}(t+T) = b_{i,l}(t) + \eta \sum_{v=1}^q \left\{ e_v(t) \sum_{\mu=1}^p \left(\frac{\partial y_v}{\partial u_{\mu}} \delta_{\mu,i,l} \frac{S_{i,l}}{b_{i,l}} \right) \right\} \quad (18d)$$

where T is the sampling period. The only unknowns in Eqs. (18a) – (18d) are the gradients of system output with respect to the inputs, $\frac{\partial y_v}{\partial u_{\mu}}$, which is the (v, μ) -th component of the Jacobian matrix of the controlled system. Usually, the partial derivative $\frac{\partial y_v}{\partial u_{\mu}}$ depends on the operating point and can be estimated by using process physical model. However, difficulty may arise in estimating the true value of $\frac{\partial y}{\partial u}$ when an accurate physical model is unavailable or the process is subject to uncertainties and/or unmeasured disturbances. Fortunately, $\eta \frac{\partial y_v}{\partial u_{\mu}}$ in the proposed learning algorithm can be rewritten as

$$\begin{aligned} \eta \frac{\partial y_v}{\partial u_{\mu}} &= \eta \left| \frac{\partial y_v}{\partial u_{\mu}} \right| \text{sign} \left(\frac{\partial y_v}{\partial u_{\mu}} \right) \\ &= \tilde{\eta} \text{sign} \left(\frac{\partial y_v}{\partial u_{\mu}} \right) \end{aligned} \quad (19)$$

where $\text{sign}(x)$ is the sign of x and $\tilde{\eta} \equiv \eta \left| \frac{\partial y_v}{\partial u_\mu} \right|$. This implies that if we treat $\tilde{\eta}$ as an apparent (constant) learning rate, the accurate value of $\frac{\partial y_v}{\partial u_\mu}$ becomes unnecessary. Certainly, this requires $\left| \frac{\partial y_v}{\partial u_\mu} \right| < \infty$ for all $t \geq 0$. Such an simplification avoids the difficulty of estimation of an accurate value of $\frac{\partial y_v}{\partial u_\mu}$. Therefore, if the sign of $\frac{\partial y_v}{\partial u_\mu}$ at each instant is known, one can get a simplified learning algorithm for on-line parameter adjustment. The sign of $\frac{\partial y_v}{\partial u_\mu}$, the system response direction, can be evaluated explicitly if the plant model is available. In general, an accurate but complex physical model is not necessary any more since the model is used for estimating the system response direction rather than its accurate value. If the real plant were used directly in the scheme, the system response directions can be obtained by using first-order differences (Psaltis *et al.*, 1987) with changing each input to the plant slightly at the operating point and measuring the change at the output. It is interesting to note that for a broad class of monotone-responded industrial processes, *i.e.*, the system output increases (decreases) as the control input increases, their system response direction is constant and can be obtained from either a step response experiment or the physical properties of the control plant. For such a process, the sign of $\frac{\partial y}{\partial u}$ can be replaced by the sign of steady state gain. Consequently, the proposed scheme can be applied without process model. It is therefore of great appealing for some highly uncertain and/or unmodeled processes having fixed response direction.

Here, it should be noted that the presented learning algorithm is not far from that of the traditional back-propagation algorithm. However, it provides more flexibility and nonlinear capability to neural controller and in essence it includes the conventional fixed-shape one as a special case. Especially, the proposed algorithm provides the ability of controller output adjustment. To close this section, we summarize the procedure of performing the proposed learning algorithm for controller parameter tuning at each time step as follows:

Step 1: Observing process output errors $e_v(t)$, for $v=1, 2, \dots, q$ and estimating the gradients $\frac{\partial y_v}{\partial u_\mu}$ (or the sign of $\frac{\partial y_v}{\partial u_\mu}$), for $v=1, 2, \dots, q$ and $\mu=1, 2, \dots, p$.

Step 2: Set $l=m$. Determining $\xi_{\mu,i,l}$ from Eq. (17) and then $\delta_{\mu,i,l}$ from Eq. (16) for $i=1, 2, \dots, N_l$ and $\mu=1, 2, \dots, p$.

Step 3: Updating controller parameters according to Eq. (18) for layer l .

Step 4: Set $l=l-1$ and go back to step 2 and 3 to update controller parameters until $l=1$.

3. A Case Study: Controlling an Open-Loop Unstable Nonlinear CSTR

The aim of this section is to demonstrate the advantages and effectiveness of the proposed shape-tunable neural controller through controlling a nonlinear CSTR. The dynamic equations of the nonlinear CSTR is given by (Ray, 1981):

$$\dot{x}_1 = -x_1 + D_a(1-x_1)e^{\frac{x_2}{1+x_2/\varphi}} \quad (20a)$$

$$\dot{x}_2 = -(1+\beta)x_2 + BD_a(1-x_1)e^{\frac{x_2}{1+x_2/\varphi}} + \beta u \quad (20b)$$

where x_1 and x_2 represent the dimensionless reactant concentration and reactor temperature, respectively. The control input u is the dimensionless cooling jacket temperature. The physical parameters in the CSTR model equations are D_a , φ , B and β which correspond to the Damköhler number, the activated energy, heat of reaction and heat transfer coefficient, respectively. Based on the nominal system parameters: $D_a=0.072$, $\varphi=20$, $B=8$, and $\beta=0.3$, the open-loop CSTR exhibits three steady states $(x_1, x_2)_A=(0.144, 0.886)$, $(x_1, x_2)_B=(0.445, 2.750)$ and $(x_1, x_2)_C=(0.765, 4.705)$. **Figure 2**, the phase portrait of the CSTR, shows that the upper and lower steady states $(x_1, x_2)_A$ and $(x_1, x_2)_C$ are stable, while the middle one, $(x_1, x_2)_B$, is unstable. The control objective is to bring the nonlinear CSTR from the lower stable equilibrium point $(x_1, x_2)_A$ to the unstable one $(x_1, x_2)_B$. All of the results presented are based on the temperature control of the reactor.

In order to provide gradient information at each control step, we convert the system equation into its discrete form by the Euler method (Åström and Wittenmark, 1984) as follows:

$$x_1(k+1) = x_1(k) + T \left[-x_1(k) + D_a(1-x_1(k))e^{\frac{x_2(k)}{1+x_2(k)/\varphi}} \right] \quad (21a)$$

$$x_2(k+1) = x_2(k) + T \left[-(1+\beta)x_2(k) + BD_a(1-x_1(k))e^{\frac{x_2(k)}{1+x_2(k)/\varphi}} + \beta u(k) \right] \quad (21b)$$

$$y(k) = x_2(k) \quad (21c)$$

where T denotes the sampling time and k the k -th

Fig. 2 The phase portrait of the open-loop nonlinear CSTR

control step. The relationship between the system output y and control input u can be obtained by substituting Eq. (21b) into Eq. (21c) with $k=k+1$:

$$y(k+1) = x_2(k) + T \left[-(1+\beta)x_2(k) + BD_a(1-x_1(k))e^{\frac{x_2(k)}{1+x_2(k)/\varphi}} + \beta u(k) \right] \quad (22)$$

Based on the above equation, the gradient information that required in the learning algorithm can be estimated by

$$\frac{\partial y}{\partial u} = \frac{\partial y(k+1)}{\partial u(k)} = T\beta \quad (23)$$

In the following simulation studies, the neural controller is selected to be composed of three layers. The input and output layer consists of a single neuron respectively, since the CSTR is a single-input/single-output system. The node number used in the hidden layer is arbitrary set as ten. For the sake of easing the computational burden, only the shape of the activated function in the output layer is adjusted on-line while the nonlinear activated functions in the other nodes are kept constant to be $h(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$. Although this effort increases two additional parameters as compared to the conventional neural controllers, it provides more flexibility and nonlinear capability for control. Especially, the scaling procedure becomes unnecessary. For comparison purpose, in what follows, all the initial values of the controller parameters, $w_{i,j,l}$ and $\theta_{i,l}$, are set equal to 0.5 and the initial shape parameters a and b used in the output layer are set to be $a=1$ and $b=2$. The sampling time

Fig. 3 Comparison of the proposed controller (A) with the conventional fixed-shape neural controller using different scaling factors (B) 0.5 (C) 1 (without scaling) (D) 10

T and the learning rate η are set as 0.2 and 0.3, respectively. Having the previous preparations, we are ready to investigate the following issues:

3.1 Comparison with fixed-shape neural controller

Figure 3 shows the comparison of the shape-tunable neural controller with the conventional fixed-shape neural controller. As can be seen from this figure, there exists an offset when the conventional fixed-shape neural controller with scaling factor 0.5 is applied. This is because that the control input produced from the neural controller was scaled too small, which is prone to losing the ability to bring the system to the desired set point. On the other hand, if a scaling factor of 10 of the fixed-shape neural controller is introduced, the offset is eliminated. However, it produces a long-lasting oscillation in the transient. These results clearly imply that the scaling factor used in the conventional fixed-shape neural controller has a profound effect on the control performance. Unfortunately, the optimal scaling factor is difficult to be determined *a priori*, especially for those systems having high uncertainties and/or imprecisely known dynamics. In contrast to the fixed-shape neural controller, the proposed neural controller has the ability of automatic shaping. This ability enhances the flexibility and nonlinearity of neural controller and avoids the difficulty of selecting an optimal scaling factor. As shown here, the shape-tunable neural controller gives a better control performance. Furthermore, **Fig. 4** shows that with the proposed learning algorithm the optimal activated function shape is tracked automatically in a short time.

Fig. 4 The time progress of the shape parameters

To have a further comparison, we assume that there exists an actuator saturation between the controller and the process system. It is well known that the problem of input constraint caused by the actuator saturation is presented in almost every chemical processes. The existence of input saturation imposes extra nonlinearity on system. For simulation, let the actuator saturation under consideration be modeled by

$$\hat{u} = \text{sat}(u) \quad (24)$$

where

$$\text{sat}(u) = \begin{cases} u, & |u| \leq 1 \\ \text{sign}(u), & |u| > 1 \end{cases} \quad (25)$$

In the presence of actuator, the block diagram of the overall system is shown in Fig. 5. The actuator saturation limits the control input by the range of actuators $(-1,1)$. This suggests a scaling factor of 1 for the fixed-shape neural controller since the outputs of the controller naturally lie between -1 and 1 . The comparison of control performances of these two schemes was shown in Fig. 6. The comparison of control inputs was also shown. It can be seen from this figure that the control performance by the proposed scheme is apparently better than that of conventional fixed-shape one. The control inputs generated by the shape-tunable neural controller reached its maximum value soon to give a faster transient. However, the fixed-shape neural controller can not afford enough control inputs. The reason is that the outputs of a fixed-shape neural controller usually lie within their $\pm 90\%$ of its saturated level. It can not reach its extreme value without infinite large inputs. Therefore, a slower response was obtained. Based on the above results,

Fig. 5 A CSTR control system with actuator saturation

Fig. 6 Closed-loop responses with control input hard constraint:
— the proposed shape-tunable neural controller;
--- the conventional fixed-shape neural controller with scaling factor of 1

the advantages of the proposed shape-tunable nonlinear control strategy over the conventional fixed-shape one are evident. In what follows, we shall investigate some more important issues of applying the proposed nonlinear control strategy.

3.2 Unmeasured disturbance rejection

The first issue to be considered is to examine the disturbance rejection ability by using the proposed shape-tunable neural controller. The disturbance is assumed to be unmeasurable. For the later simulation studies, we assume that there exists a step feed temperature disturbance of amplitude 0.2, *i.e.*, the disturbance was added in the right hand side of the process dynamic equation (20b), after 250 control steps (50 min). Figure 7 shows the closed-loop response to the unmeasured disturbance. The control input and the time progress of shape parameters a and b were also shown. It can be seen from this figure that the control input changes immediately and reaches its new steady state value to accommodate the unmeasured disturbance. As a result, excellent disturbance rejection performance was obtained.

Fig. 7 Unmeasured disturbance rejection: The upper graph shows the closed-loop response and the control input. The lower one shows the time progress of the shape parameters

Fig. 9 Robustness of the proposed control scheme: The upper graph shows the closed-loop response and the control input. The lower one shows the time progress of the shape parameters

Fig. 8 The upper graph shows the closed-loop response with measurement noise. The lower one shows the control input

The effect of measurement noise on the control performance was also studied. Temperature measurement noise was simulated by adding zero-mean random numbers with standard deviation of 0.1. **Figure 8** shows that despite of the relatively large measurement noise, the proposed control scheme can still easily bring the nonlinear CSTR to its unstable equilibrium point and remain there.

3.3 Effect of number of neurons on closed-loop performance

Another important issue that needs to be investigated is the robustness of the proposed nonlinear control strategy. The problem of interest would be whether the neural controller still works

properly when some nodes in the hidden layer are suddenly broken down during control operation. **Figure 9** shows clearly that there is no significant impact upon the system output when five randomly selected nodes are broken down after 250 control steps. Only slight hump occurs and then back to the unstable equilibrium point in a short time. No oscillation or divergence behavior arises. This is because that with the learning algorithm the neural controller is able to adapt itself immediately to the sudden change. Besides, it is interesting to observe that the optimal shape of the activated function is almost unaffected by this sudden node failure in the hidden layer. Accordingly, the robust performance observed in the simulation can be explained by the effectiveness and robustness of the learning algorithm.

3.4 Plant/Model mismatch

Due to its intrinsic nonlinearity, the static and dynamic behavior of the nonlinear CSTR is a strong function of the system parameter values. When the system parameters change and/or the model is not so accurate to present the actual process, there is considerable difference between the actual process output and model output. To explore the plant uncertainty on the essential behavior of the control system, we assume that the value of the Damköhler number, D_a , in the actual system is suddenly changed to 0.12 after 250 control steps (50 min) due to parameter uncertainty while in the model the Damköhler number is still of the value of 0.072. This parameter uncertainty introduces a 66.7% variation in D_a and makes the actual process to have only one open-loop equilibrium point, which is quite different from the process model that has three equilibrium

Fig. 10 Plant/model mismatch process control: The upper graph shows the closed-loop response and the control input. The lower one shows the time progress of the shape parameters (after 250 steps, model: $D_a=0.072$; system: $D_a=0.12$)

points. **Figure 10** shows the system response to the sudden plant/model mismatch. As can be seen from this figure that the effect of model/plant mismatch can soon be accommodated by changing the level of control input. These results corroborated the advantage of shape-tuning ability.

3.5 Direct control without using the process model

In the view of the proposed control scheme, the major function of the process model is to estimate the gradient information $\frac{\partial y}{\partial u}$. Referring to Eq. (23), the

gradient information of this nonlinear CSTR is constant and will not be changed during operating. In other words, when the control input (cooling jacket temperature) increases, the reactor temperature increases; when the control input decreases, so the reactor temperature. Therefore, one can replace $\eta \frac{\partial y}{\partial u}$ by $\tilde{\eta} \text{sign}\left(\frac{\partial y}{\partial u}\right)$ or a merely a constant $\tilde{\eta}$ in the learning algorithm since $\text{sign}\left(\frac{\partial y}{\partial u}\right)=1$. More

precisely, to implement the proposed control scheme, only the system output error is required. **Figure 11** shows the system response with adding zero-mean measurement noise with standard deviation 0.1. This figure shows that an excellent response was obtained even though the process model is not required.

Conclusion

In this paper, a nonlinear control strategy based

Fig. 11 Unmodeled nonlinear process direct control: The upper graph shows the closed-loop response with measurement noise. The lower one shows the control input ($\tilde{\eta}=0.3$)

on using a shape-tunable neural network has been proposed for nonlinear process control. A learning algorithm that enables the shape-tunable neural network to control a nonlinear process adaptively has been derived. Due to the ability of automatic shaping, the proposed control scheme overcomes the difficulty of selecting an optimal scaling factor and enhances the neural controller more flexibility and nonlinear capability. In applying the proposed control scheme, the only *a priori* knowledge of the controlled process system is the information of system output response, which can be estimated directly by using system's physical model. For those process systems whose output response direction is unchanged over the operating range, the proposed control scheme makes the unmodeled nonlinear process control possible. The advantages and effectiveness of the proposed scheme have been demonstrated through the challenge problem of controlling an open-loop unstable CSTR.

Simulation results show that the controller is directly operational. It should also be mentioned that there is no real time constraints—compared with the conventional PID controllers, where a constant sampling rate must be guaranteed. In contrast, the shape-tunable neural controller can make efficient use of the available computing power. Despite these advantages, the controller still has some drawbacks. Certainly, an adaptive controller using direct control by error back propagation can be very sensitive to the dynamic properties of the plant. Also, it has been found that the control quality depends heavily on the initial values of controller parameters. Nevertheless, the proposed shape-tunable neural control strategy is a promising approach for complex plants that can not

be controlled by conventional ones in a satisfiable manner. A detailed investigation of the stability issues of the proposed adaptive scheme was beyond the scope of this paper and should be preserved for future work.

Acknowledgment

This work was partially supported by the National Science Council of the Republic of China under Grant NSC84-2214-E-035-005.

Nomenclature

a	= saturated level
$a_{i,j}$	= saturated level in i -th neuron of layer j
b	= gain factor
$b_{i,j}$	= gain factor in i -th neuron of layer j
B	= heat of reaction
Da	= Damköhler number
e_i	= i -th process output error
E	= error function defined in Eq. (5)
$\mathbf{f}$	= vector function in Eq. (1a)
$\mathbf{g}$	= vector function in Eq. (1b)
h	= hyperbolic tangent function defined in Eq. (2)
N_k	= node number in layer k
$o_{i,j}$	= i -th neuron of layer j
p	= no. of control inputs
q	= no. of system outputs
$S_{i,j}$	= weight sum in i -th neuron of layer j
$\text{sign}(u)$	= sign of u
$\text{sat}(u)$	= saturation function defined in Eq. (25)
t	= time
T	= sampling period
u	= control input
$w_{i,j,l}$	= weight connecting the j -th neuron in $(l-1)$ -th layer and i -th neuron in l -th layer
$\mathbf{x}$	= vector of states
x_1	= dimensionless reactant concentration
x_2	= dimensionless reactor temperature
$\mathbf{y}$	= system output vector
$\mathbf{y}_d$	= desired output vector
β	= heat transfer coefficient
$\delta_{\mu,i,l}$	= coefficient defined in Eq. (14)
φ	= activated energy
η	= learning rate

$\tilde{\eta}$	= learning rate
$\theta_{i,l}$	= threshold of i -th neuron in l -th layer
$\xi_{\mu,i,l}$	= coefficient defined in Eq. (15)

Literature Cited

- Arai, M., R. Kohon and H. Imai; "Adaptive Control of a Neural Network with a Variable Function of a Unit and Its Application," *Trans. Inst. Electron., Inform. and Commun. Eng.*, **74-A**, 551-559 (1991)
- Åström, K.J. and B. Wittenmark; *Computer Controlled System: Theory and Design*, Prentice-Hall, New Jersey (1984)
- Bequette, B.W.; "Nonlinear Control of Chemical Processes: A Review," *Ind. Eng. Chem. Res.*, **30**, 1391-1398 (1991)
- Bhat, N. and T.J. McAvoy; "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems," *Computers Chem. Engng.*, **14**, 573-583 (1990)
- Chen, C.T. and W.D. Chang; "A Feedforward Neural Network with Function Shape Autotuning," *Neural Networks*, **9**, 627-641 (1996)
- Cui, X. and K.G. Shin; "Direct Control and Coordination Using Neural Networks," *IEEE Trans. System, man, and Cybernetics*, **23**, 686-697 (1993)
- Hu, Z. and H. Shao; "The Study of Neural Network Adaptive Control Systems," *Control and Decision*, **7**, 361-366 (1992)
- Hunt, K.J., D. Sbarbaro, R. Zbikowski and P.J. Gawthrop; "Neural Networks for Control Systems-A Survey," *Automatica*, **28**, 1083-1111 (1992)
- Kawato, M., Y. Uno, M. Isobe and R. Suzuki; "A Hierarchical Model for Voluntary Movement and Its Application to Robotics," in Proc., IEEE Int. Conf. Network, **IV**, p. 573-582, San Diego, CA (1987)
- Psaltis, D., A. Siders and A. Yamamura; "Neural Controller," in Proc. IEEE Int. Conf. Neural Networks **IV**, p. 551-558, San Diego, CA (1987)
- Ray, W. H.; *Advanced Process Control*, McGraw-Hill, New York, USA (1981)
- Rumelhart, D.E. and J.L. McClelland; *Parallel Distributed Computing: Exploration in the Microstructure of Cognition*, MIT Press, Cambridge, Mass. (1986)
- Willis, M.J., G.A. Montague, C. Di. Massimo, M.T. Tham and A.J. Morris; "Artificial Neural Networks in Process Estimation and Control," *Automatica*, **28**, 1181-1187 (1992)
- Yamada, T. and T. Yabuta; "Neural Network Controller Using Auto-Tuning Method for Nonlinear Functions," *IEEE Transactions on Neural Networks*, **3**, 595-601 (1992)