ELSEVIER

Journal of Process Control 14 (2004) 211–230

JOURNAL OF PROCESS CONTROL

www.elsevier.com/locate/jprocont

Applying neural networks to on-line updated PID controllers for nonlinear process control

Junghui Chen*, Tien-Chih Huang

R&D Center for Membrane Technology, Department of Chemical Engineering, Chung-Yuan Christian University, Chung-Li, Taiwan 320, Republic of China

Received 19 September 2002; received in revised form 16 April 2003; accepted 2 June 2003

Abstract

The inherent time-varying nonlinearity and complexity usually exist in chemical processes. The design of control structure should be properly adjusted based on the current state. In this paper, an improved conventional PID control scheme using linearization through a specified neural network is developed to control nonlinear processes. The linearization of the neural network model is used to extract the linear model for updating the controller parameters. In the scheme of the optimal tuning PID controller, the concept of general minimum variance and constrained criterias are also considered. In order to meet most of the practical application problems, several variations of the proposed method, including the momentum filter, the updating criterion and the adjustment of the step size of the control action, are presented to make the proposed algorithm more practical. To demonstrate the potential applications of the proposed strategies, two simulation problems, including a pH neutralization and a batch reactor, are applied.

Keywords: Neural networks; Nonlinear modeling; PID controller

1. Introduction

Despite the advent of many complicated control theories and techniques, more than 95% of the control loops based on proportional-Integral-Derivative (PID) controllers are still being used in the majority of industrial processes. It can be thus said to be the ''bread and butter'' of control engineering [2] because of its simplicity in structure, robustness in operation and easy comprehension in principle. Nevertheless, the PID algorithm might be difficult to deal with in highly nonlinear and time varying chemical processes. To improve the control performance, several schemes of self-tuning PID controllers were proposed in the past. Wittenmark [25] proposed the control structure with the PID algorithm calculated via pole placement design. The method was limited in the order of the controlled processes. The self-tuning PI or PID algorithms were automatically derived from the dynamic of the controlled processes [11]. An alterative self-tuning PID controller was based on the generalized minimum variance control [6,8]. The control structure was orientated to have a PID structure. The controller parameters were obtained using a parameter estimation scheme. Other forms of self-tuning PID can be found in literature [18,17,20]. However, the above self-tuning adaptive control approaches are limited to linear system theory, i.e. these techniques assume that the control model with the linear model is operated in a linear region. If some changes in the process or environment occur, it must be manually checked whether the model is adequate to represent the real process or not since the control design is totally based on a reliable model.

Currently neural networks constitute a very large research interest. They have great capability in solving complex mathematical problems since they have been proven to approximate any continuous function as accurately as possible [12]. Hence, it has received considerable attention in the field of chemical process control and has been applied to system identification and controller design [3,24]. All of these works show that the neural network can capture the characteristics of system patterns and performance function approximation for

* Corresponding author. Fax: +886-3-265-4199.
E-mail address: jason@wavenet.cycu.edu.tw (J. Chen).

0959-1524/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0959-1524(03)00039-8

nonlinear systems [15]. Thus, it is effectively used in the control region for modeling nonlinear processes especially in the model-based control, such as the direct and indirect neural network model based control [19], nonlinear internal model control [14], and recurrent neural network model control [16]. Although the control performances of the above methods are satisfactory, the nonlinear iterative algorithm of the control design is computationally demanding because of the system based on the nonlinear neural network model. This may make the implementation strategy realistic only for control of slow dynamic systems.

In the linear control design theory, linearization of nonlinear models is often used in the control field to alleviate the design of controllers for nonlinear systems. As we know, a model estimated through linearization based on the operating point can be considered valid only in a certain region around this point. Due to the characteristics of the nonlinearities and the size of the operating region, it is necessary to consider whether to use a single linear model or to obtain more linearization models around the different operating regions. The latter called gain scheduling is often chosen to design and control the nonlinear process. It selects a set of predefined linear controllers. Each controller is tuned for a specific operating region. Gain scheduling can be implemented in various ways according to the nature of applications under consideration.

Based on the previous discussion of the linearization of the nonlinear model, several combinational methods based on neural networks and the traditional linear controller design have been developed. Fuli et al. [10] proposed a compromised method with the neural network and pole placement design. This method assumed that the plant could be linearized at each operating point. It used the linear neural network to capture the linear dynamic behavior of processes, and the multilayered feedforward neural network to identify the nonlinear part as measured disturbance. In the controller design, it used pole placement as feedback control, and the multilayered feedforward neural network as feedforward control to eliminate the nonlinear disturbance. Ahmed and Tasadduq [1] mentioned a three-stage procedure for designing controllers by linearization through a built neural network. The procedure implemented any linear controller on processes like pole-placement and optimal-control strategies. Finally, a neural network controller from the above control loop was built to replace the linear controller. Although the above two methods are based on the linearization through neural networks and pole placement design, they are not concerned with the process with the PID controller.

To improve the control performance, an on-line updated PID algorithm is proposed. It combines the general minimum variance (GMV) control law with the extracting the characteristic of the instantaneous linearized neural network model. With linearization of the neural network model, the PID algorithm can be implemented directly without any modification. This methodology is good for controlling nonlinear processes without highly demanding computation, because the controller design is based on the linear model instead of the nonlinear model.

2. Design problem statement

The block diagram of the control system to be considered is shown in Fig. 1. The controlled process is any nonlinear continuous or batch process. The PID controller from the process variable $y(t)$ to the control variable $u(t)$ is

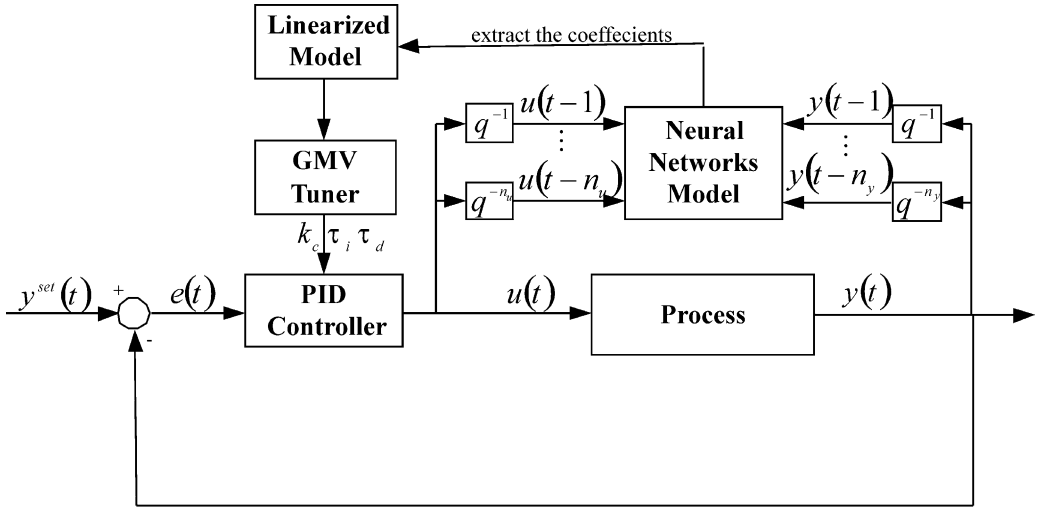

Fig. 1. The scheme of the adaptive PID control based on the instantaneous linearization of the neural network model.

$$u(t) = u_s + k_c\left(e(t) + \frac{1}{\tau_i}\int e(t)\mathrm{d}t + \tau_d\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right) \quad (1)$$

where u_s is bias value. $e(t) = y^{\mathrm{set}}(t) - y(t)$ is the output error deviated from the setpoint. k_c, τ_i and τ_d are known as the proportional gain, the integral time constant and derivative time constant respectively. A velocity form of the discrete PID control can be written as

$$\begin{aligned}\Delta u(t) &= u(t) - u(t-1)\\ &= k_c\left[(e(t) - e(t-1)) + \frac{\Delta t}{2\tau_i}(e(t) + e(t-1))\right.\\ &\quad \left. + \frac{\tau_d}{\Delta t}(e(t) - 2e(t-1) + e(t-2))\right] \end{aligned} \quad (2)$$

where the integral action of Eq. (2) is computed using the trapezoidal approximation. Rearrange the discrete form of the PID control to be in the following form

$$\begin{aligned}\Delta u(t) &= k_0 e(t) + k_1 e(t-1) + k_2 e(t-2)\\ &= \mathbf{e}^T(t)\mathbf{k}(t)\end{aligned} \quad (3)$$

where $\mathbf{k}(t) = \begin{bmatrix}k_0 & k_1 & k_2\end{bmatrix}^T$, $\mathbf{e}(t) = \begin{bmatrix}e(t) & e(t-1) & e(t-2)\end{bmatrix}^T$ and

$$\begin{aligned}k_0 &= k_c\left(1 + \frac{\Delta t}{2\tau_i} + \frac{\tau_d}{\Delta t}\right), \quad k_1\\ &= -k_c\left(1 - \frac{\Delta t}{2\tau_i} + \frac{2\tau_d}{\Delta t}\right), \quad k_2 = \frac{k_c\tau_d}{\Delta t}.\end{aligned} \quad (4)$$

In Fig. 1, the control structure is similar to an adaptive control structure. The parameters of the PID controller are adjusted by an outer loop composed of an instantaneously linearized neural network model estimator and a GMV control design calculation. In the traditional adaptive control design, the time-varying parameters of the linear model are estimated on-line by a recursive identification algorithm, including a forgetting factor to place lighter emphasis on older data. In this study, an off-line neural network model is trained to model the nonlinear process and then an instantaneous linearization of neural networks at each sampling point is conducted to get a linearized model.

In fact, the functional behavior of the proposed control structure looks similar to an adaptive controller or a gain schedule control whose model is chosen from a set of predefined linearized models, but in the instantaneous linearization of the neural network model the process dynamic parameters can be changed quickly in response to process changes. Besides, the neural network model is trained off-line; extra computation load is not needed to construct the current model in the on-line identification for the current control design. The more important aspect is that the traditional neural network control design requires training the neuron-controller on-line as the performance error back-propagates through the network at every sample. Sometimes the emulator neural network is used due to the requirement of the Jacobian of the process as shown in Fig. 2, since the process is unknown [22,26]. This nonlinear optimization problem may cause improper solution for the control design.

2.1. Instantaneous linearization of the neural network

Assume a deterministic process where the general form represents a discrete-time nonlinear system

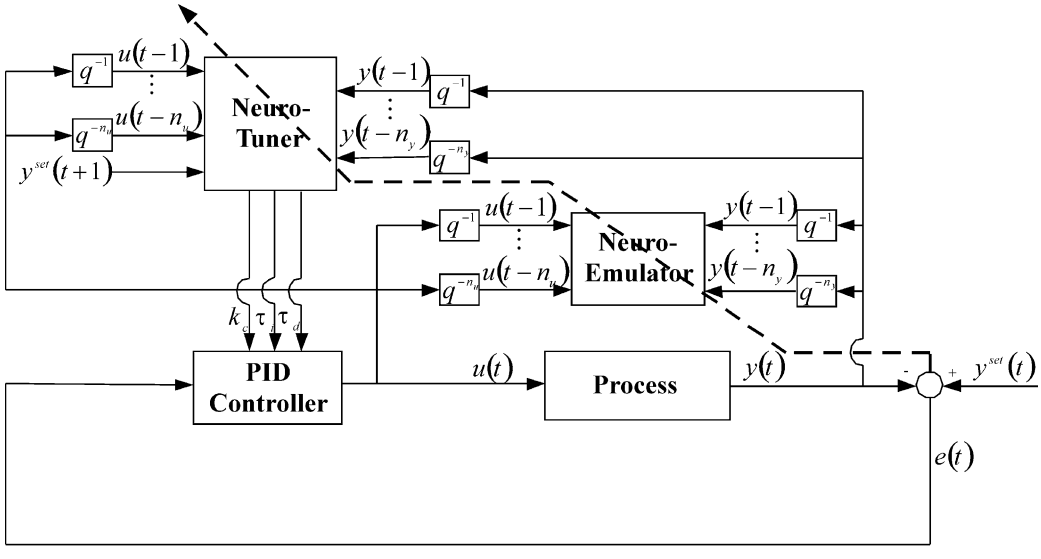

Fig. 2. The configuration of neural network controller design based on an emulator neural network model.

$$y(t) = f\big(y(t-1), \ldots, y(t-n_y), u(t-1), \ldots, u(t-n_u)\big) + e(t). \quad (5)$$

Here $y(t)$ is the process output, $u(t)$ the input, $e(t)$ a zero-mean disturbance term, and n_y and n_u indicate the number of output and input delay respectively. The process model can be written as a deterministic neural network model

$$\hat{y}^{\text{NNARX}}(t) = \text{NNARX}(\phi(t)) \quad (6)$$

where NNARX is a neural network ARX function. This type of model has been studied widely in non-linear system identification [7,27]. The regression vector is defined as

$$\phi(t) = [\mathbf{y}(t), \mathbf{u}(t)]^T = \big[y(t-1), \ldots, y(t-n_y), u(t-1), \ldots, u(t-n_u)\big]^T. \quad (7)$$

The goal of the neural network modeling is to find a parameterized structure that emulates the nonlinear process. The NNARX model used here is a three-layer feedforward neural network with a linear activation function in the output neuron and with a hyperbolic tangent activation function in the hidden neurons. It can be written as

$$\hat{y}^{\text{NNARX}}(t) = \sum_{c=1}^{N_{\text{hidden}}} w_c^o z_c[m_c(t)] + w_b^o$$

$$m_c(t) = \sum_{\substack{i=1 \\ i_y=i}}^{n_y} w_{c,i_y}^h y(t-i) + \sum_{\substack{i=1 \\ i_u=i+n_y}}^{n_u} w_{c,i_u}^h u(t-i) + w_{b,c}^h \quad (8)$$

where z_c is the transfer function for the hidden neuron c, w_c^o and w_b^o are the weights and the bias of the hidden-to-output layer, $w_{c,i}^h$ and $w_{b,c}^h$ are the weights and the bias of the input-to-hidden layer, and m_c is the summation of all products between inputs and input-to-hidden weights in the input layer.

The idea behind instantaneous linearization is to extract a linear model from the nonlinear neural network model at each sample point. The approximate linear model at time $t = \tau$ can be obtained by linearizing NNARX around the current state $\phi(t = \tau)$. Linearization means the first partial derivative term for each element of $\phi(t = \tau)$. The approximated model $\hat{y}^{\text{INST-NN}}(t)$ can be written as:

$$\hat{y}^{\text{INST-NN}}(t) = bias - \sum_{i=1}^{n_y} a_i y(t-i) + \sum_{i=1}^{n_u} b_i u(t-i) \quad (9)$$

where a_i and b_i are the linear model coefficients of the output and the input,

$$a_i = -\frac{\partial \text{NNARX}}{\partial y(t-i)}\Big|_{\Phi(t)=\Phi(\tau)} i = 1, 2, \cdots, n_y. \quad (10)$$

$$b_i = \frac{\partial \text{NNARX}}{\partial u(t-i)}\Big|_{\Phi(t)=\Phi(\tau)} i = 1, 2, \cdots, n_u. \quad (11)$$

The approximated linear model is affected by a constant bias, *bias*, depending on the current operating point.

$$bias = y(\tau) + \sum_{i=1}^{n_y} a_i y(\tau - i) + \sum_{i=1}^{n_u} b_i u(\tau - i). \quad (12)$$

Based on the descriptions of Eqs. (10) and (11), the derivatives of the output with respect to the current state $\phi(t = \tau)$ can be calculated as:

$$a_i = \frac{\partial \hat{y}^{\text{NNARX}}(t)}{\partial y(t-i)} = \sum_{c=1}^{N_{\text{hidden}}} w_c^o \frac{\partial z_c(t)}{\partial m_c(t)} \frac{\partial m_c(t)}{\partial y(t-i)} = \sum_{c=1}^{N_{\text{hidden}}} w_c^o \frac{\partial z_c(t)}{\partial m_c(t)} w_{c,i}^h \quad , i = 1, 2, \cdots, n_y \quad (13)$$

$$b_i = \frac{\partial \hat{y}^{\text{NNARX}}(t)}{\partial u(t-i)} = \sum_{c=1}^{N_{\text{hidden}}} w_c^o \frac{\partial z_c(t)}{\partial m_c(t)} \frac{\partial m_c(t)}{\partial u(t-i)} = \sum_{c=1}^{N_{\text{hidden}}} w_c^o \frac{\partial z_c(t)}{\partial m_c(t)} w_{c,(i+n_y)}^h \quad , i = 1, 2, \cdots, n_u. \quad (14)$$

The controller design based on the instantaneous linearization of the neural network model has two advantages:

2.1.1. Process modeling

Linearization of the nonlinear model is a well-known method often used among control designs. As we know, the model obtained through linearization around an operating point can be considered valid only in a certain region around this point. At a first glance, the linearized model seems to be a crude description of the actual process, but the new model obtained by linearization will be immediately updated with the change of the size of the operating range and the character of nonlinearities. In Fig. 3, the curve with a little fluctuation represents the actual process output with measured noise. Due to the characteristic of the noise cancellation of the neural network, the trained neural network model can smooth the measured data and properly represent the dynamic process. Fig. 3 also indicates that the prediction

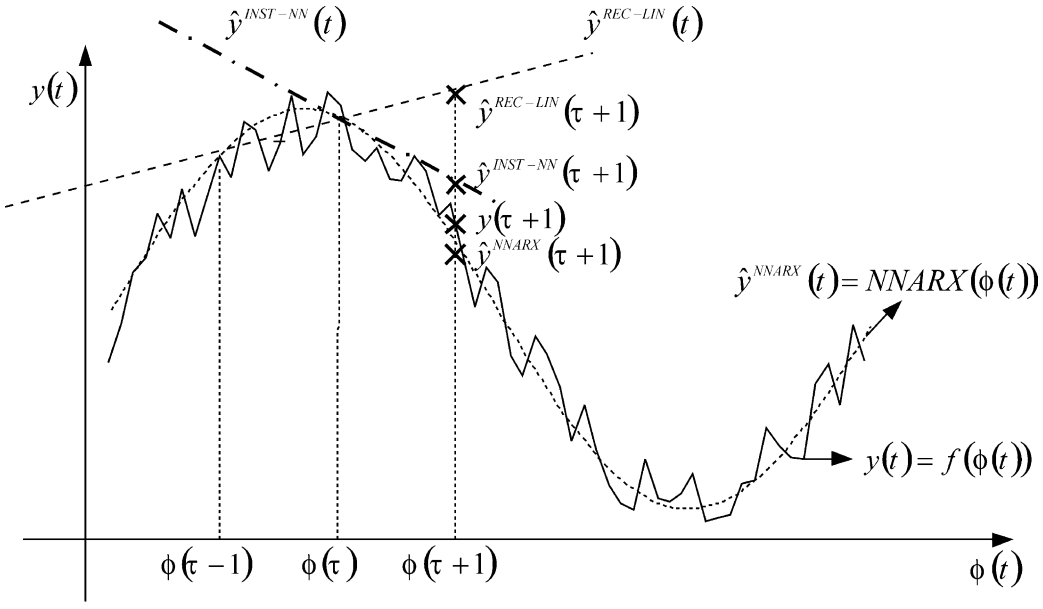

Fig. 3. Comparison of the predicted output from NNARX, INST-NN and recursive linear model at the next time step.

of the instantaneous linearization NNARX $\hat{y}^{\text{INST-NN}}(t=\tau+1)$ at one-step ahead can closely follow the process behavior. In the traditional linear control theory, the recursive identification is often used for nonlinear systems. The linear model is properly updated with the new data coming. However, the linear model $\hat{y}^{\text{REC-LIN}}(t)$ may not be quickly updated because the model only accounts for the past information until now with little consideration of the current characteristic of the nonlinear process. The model error may be significantly large when the linear model can be considered valid only in a very narrow region around this point. To get more impression of the difference of the predictability between $\hat{y}^{\text{INST-NN}}(t)$ and $\hat{y}^{\text{REC-LIN}}(t)$, a simple nonlinear dynamic system is employed [15].

$$z(t)=\frac{z(t-1)}{1+z(t-1)^2}+u(t-1)^3$$
$$y(t)=z(t)+v(k) \tag{15}$$

where $y(t)$ and $u(t)$ are the measurement of the process output and input variables at time t. The random measurement noise $v(k)$ with $N(0,0.1)$ is added to the measured output.

Our goal here is to predict one-step ahead $\hat{y}(\tau+1)$ based on the known values of the time series up to the current time point. The sum of the square of the prediction errors (SSE) of these models are listed in Table 1. Here the recursive identifications with two different forgetting factors are also used. The SSE of the recursive identification would be reduced with a larger forgetting factor that places heavier emphasis on more recent data. It is observed that SSE of the instantaneous linearization of the trained neural network model is less than that of the other models.

2.1.2. Controller design

If the controller design is based on the nonlinear model, there would be several problems, including the computation load of the iterative minimization, trapping in the local minimum of the criterion and repeating different initial points several times using the minimized criterion. It would significantly reduce interests to match the need of the realistic industrial problems. Once a linearized neural network is obtained, it can be easily applied to the rich collection of well-understood linear design techniques in the final closed-loop system. The difficulties of nonlinear processes in applying the linear control theory are eliminated, because there is an appropriate time-invariant point for each local linearization.

2.2. General minimum variance based tuning

The goal of the controller design is to seek a control signal $u(t)$ that will minimize the difference of the process output and the desired output at the next time step; i.e. the process output can reach the desired output at the next time. Besides, from the operation point of view, the variance controller output should be minimized in order to exert excessive control effort. The objective function is expressed as

$$\min_{k_c,\tau_i,\tau_d} J=\frac{1}{2}\min_{k_c,\tau_i,\tau_d}\left[e^2(t+1)+\mu\Delta u^2(t)\right] \tag{16}$$

where μ is the weighting penalty parameter. However, since $e(t+1)=y^{\text{set}}(t+1)-y(t+1)$, the objective

Table 1
Comparison among the different identification algorithms for one-step-ahead prediction

Model	NNARX	INST-NN	REC-LIN ($f = 0.8$)	REC-LIN ($f = 0.5$)
SSE	0.0424	0.5989	2.1199	9.8620

function involves a term in the future of the next time step; namely $y(t+1)$, which is not available at time t. To overcome this problem, a model based on the trained NNARX can be used; that is, $y(t+1) \cong \hat{y}^{\text{NNARX}}(t+1)$. However, the objective function based on the NNARX predicted model would involve the complicated computation for the nonlinear model. Therefore, the model can be further simplified from the linearization of the neural network model to provide estimates of $y(t+1) \cong \hat{y}^{\text{INST-NN}}(t+1)$. The modification objective is

$$\min_{k_c,\tau_i,\tau_d} J \approx \min_{k_c,\tau_i,\tau_d} L = \frac{1}{2}\min_{k_c,\tau_i,\tau_d} E\left[\left(\hat{e}^{\text{INST-NN}}(t+1)\right)^2 + \mu\Delta u^2(t)\right] \tag{17}$$

The prediction error $\hat{e}^{\text{INST-NN}}(t+1)$ based on the linearized model is $\hat{e}^{\text{INST-NN}}(t+1) = y^{set}(t+1) - \hat{y}^{\text{INST-NN}}(t+1)$. After Eq. (9) is substituted into Eq. (17), it can be rearranged in the following form

$$\begin{aligned}\hat{e}^{\text{INST-NN}}(t+1) &= y^{\text{set}}(t+1) - \sum_{i=1}^{n_y} a_k y(t-i+1) \\ &\quad - \sum_{i=1}^{n_u} b_k u(t-i+1) - bias \\ &= \Big[y^{\text{set}}(t+1) - \sum_{i=1}^{n_y} a_k y(t-k+1) \\ &\quad - \sum_{i=2}^{n_u} b_k u(t-i+1) - bias\Big] \\ &\quad - b_1 u(t) \\ &= CON - b_1 \mathbf{e}^T(t)\mathbf{k}(t)\end{aligned} \tag{18}$$

where the last term of the above equation is found using the control action $u(t)$ [Eq. (3)], and $CONT = y^{set}(t+1) - \sum_{k=1}^{n_y} a_k y(t-k+1) - \sum_{k=2}^{n_u} b_k u(t-k+1) - b_1 u(t-1) - bias$. Let the updated control parameter vector be

$$\mathbf{k}(t) = \mathbf{k}(t-1) + \Delta\mathbf{k}(t) \tag{19}$$

where $\Delta\mathbf{k}(t)$ is the space of change of the tuning parameters at the sampling instant t and $\mathbf{k}(t-1)$ is the old control parameter vector computed at the sampling $t-1$.

Substituting Eqs. (3), (18) and (19) into the objective function gives

$$L = \frac{1}{2}\left\{\text{CONT} - b_1 \mathbf{e}^T(t)[\mathbf{k}(t-1) + \Delta\mathbf{k}(t)]\right\}^2 + \frac{\mu}{2} \times \left\{\mathbf{e}^T(t)[\mathbf{k}(t-1) + \Delta\mathbf{k}(t)]\right\}^2. \tag{20}$$

When minimizing L with respect to $\Delta\mathbf{k}(t)$, we are seeking a set of PID controller parameter in the quadratic function of this objective function. The gradient of J can be computed as

$$\nabla L(\Delta\mathbf{k}(t)) = \frac{\partial L(\Delta\mathbf{k}(t))}{\partial\Delta\mathbf{k}(t)} = \mathbf{A}(t)\Delta\mathbf{k}(t) + \mathbf{d}(t) \tag{21}$$

where

$$\mathbf{A} = \begin{bmatrix} [(-b_1)^2+\mu]e^2(t) & [(-b_1)^2+\mu]e(t)e(t-1) & [(-b_1)^2+\mu]e(t)e(t-2) \\ [(-b_1)^2+\mu]e(t-1)e(t) & [(-b_1)^2+\mu]e^2(t-1) & [(-b_1)^2+\mu]e(t-1)e(t-2) \\ [(-b_1)^2+\mu]e(t-2)e(t) & [(-b_1)^2+\mu]e(t-1)e(t-2) & [(-b_1)^2+\mu]e^2(t-2) \end{bmatrix} \tag{22}$$

$$\mathbf{d}(t) = -\begin{bmatrix} [-b_1 CONTe(t)] + (b_1^2+\mu)e(t)\{\mathbf{e}^T(t)[\mathbf{k}(t-1)]\} \\ [-b_1 CONTe(t-1)] + (b_1^2+\mu)e(t-1) \\ \{\mathbf{e}^T(t)[\mathbf{k}(t-1)]\} \\ [-b_1 CONTe(t-2)] + (b_1^2+\mu)e(t-2) \\ \{\mathbf{e}^T(t)[\mathbf{k}(t-1)]\} \end{bmatrix}. \tag{23}$$

The optimal point will occur when the gradient is equal to zero. Thus, the required changes of the control parameters are

$$\Delta\mathbf{k}(t) = -\mathbf{A}^{-1}(t)\mathbf{d}(t). \tag{24}$$

Using Eqs. (4) and (19), the corresponding PID control parameters are

$$\begin{aligned} k_c(t) &= -[k_1(t) + 2k_2(t)] \\ \tau_i(t) &= \frac{-[k_1(t) + 2k_2(t)]\Delta t}{k_0(t) + k_1(t) + k_2(t)} \\ \tau_d(t) &= \frac{-k_2(t)\Delta t}{k_1(t) + 2k_2(t)}. \end{aligned} \tag{25}$$

The optimum of the above objective function is not a problem, but physically it is not suitable. The optimal PID parameters from Eq. (25) have negative values. If one wishes to obtain reasonable PID parameters, the

allowable range of this controller parameter should be defined. The constrained region is as follows:

(i) $k_c > 0$:

$$(\Delta k_1(t) + 2\Delta k_2(t)) < -(k_1(t-1) + 2k_2(t-1))$$

(ii) $\tau_i > 0$:

$$(\Delta k_0(t) + \Delta k_1(t) + \Delta k_2(t)) > -(k_0(t-1) + k_1(t-1) + k_2(t-1))$$

(iii) $\tau_d > 0$:

$$\Delta k_2(t) \geqslant -k_2(t-1).$$

Since the objective is a quadratic function[Eq. (20)] subject to the linear inequality constraints [Eq. (26)], the quadratic programming can applied here [9].

3. Heuristic rules to improve tuning

When the above-developed method is applied to a practical problem, non-smooth responses may happen due to the measurement noise and the model–process mismatch. These problems are a fact of life in complex and nonlinear chemical processes. In this section, several variations of the proposed methods will be developed to overcome these practical problems.

3.1. Momentum filter

Application of the controller action will result in amplification of the noise when the controlled variable is obtained around the desired setpoint. In order to avoid oscillations of the control parameters in the control action, the control parameters should be smoothed out. The momentum filter, like the first-order filter that reduces the variations, has been added to the control parameter changes. The updated control parameters for the momentum modification are obtained

$$\Delta\mathbf{k}^m(t) = \gamma\Delta\mathbf{k}^m(t-1) + (1-\gamma)\Delta\mathbf{k}(t)$$
$$\mathbf{k}(t) = \mathbf{k}(t-1) + \Delta\mathbf{k}^m(t) \quad (27)$$

where γ is the momentum coefficient between 0 and 1.

3.2. Updating criterion

Although the control design in the adaptive control structure should keep on-line calculating adaptive parameters of the controller, the computation of the new control action is redundant when the controlled output is close to the desired setpoint. Besides, the new computed control action in the processes which are not usually free of noise is susceptible to process noise. To overcome this problem, the concept of the statistical process control algorithm can be applied. The updated criterion, by a cumulative sum of the past error with the fixed window size, is designed to detect the deterministic shift in the desired setpoint. The combination of the most current error data sets is defined as

$$S(t) = \frac{1}{m}\sum_{i=t-m+1}^{t}(y^{\text{set}}(t) - y(t))^2 \quad (28)$$

where $S(t)$ is the current control performance. m is the size of moving window which contains $m-1$ past outputs until now. Fig. 4 shows the updated criterion based on the performance of the current moving window. An old error data would be removed and a new error data would come into the picture with the moving of the rectangular window at each sampling time. Consequently, whenever the current performance $S(t)$ is below its control limit $S(t) \leqslant \sigma^2$, the current controller parameters are assumed to be fixed until $S(t) > \sigma^2$. σ^2 is the threshold. It can be estimated from the steady-state process data when there is no change in the control action or it is based on the prior knowledge of the operating process.

3.3. Adjustment of the step size of control action

The control parameters obtained from the quadratic objective function may not be particularly optimal, because the parameters are determined through an approximation of the objective based on the instantaneous linearization of the neural network model, $L(\mathbf{k})$. It is expected that the control parameters are valid only in a neighborhood around the current state. Here the penalty term μ in Eq. (16) is used to adjust the change of the control action in order to let $L(\mathbf{k})$ be close to the

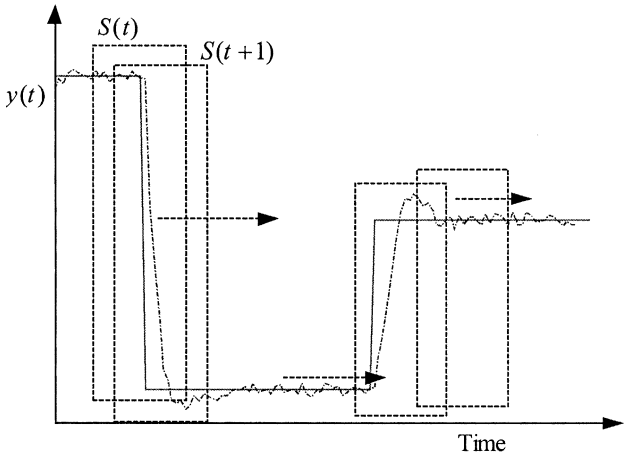

Fig. 4. Moving window used to determine when the control parameters would be updated.

true criterion $J(\mathbf{k})$. The accurate measure of the approximation can be defined

$$approx = \left|\hat{y}^{\mathrm{NNARX}}(t+1) - \hat{y}^{\mathrm{INST\text{-}NN}}(t+1)\right|. \tag{29}$$

If the difference is close to a small value, $L(\mathbf{k}(t)) = L(\mathbf{k}(t-1) + \Delta\mathbf{k}(t))$ is likely to be a reasonable approximation to $J(\mathbf{k}(t)) = J(\mathbf{k}(t-1) + \Delta\mathbf{k}(t))$. This indicates that the approximated model is good enough in the current design region; otherwise, the penalty should be adjusted by some factors to reduce or expand the region. Thus, the algorithm provides a nice compromise between the approximated linear model and the neural network model to compute the feasible control actions.

4. Procedures for the PID controller design

The proposed strategy is consisted of two phases. The first phase involves identifying the relationship of the dynamic process between the input process variable and the output process variable. NNARX is trained to derive these relationships in order to accurately predict output behavior of the possible operating condition. Thus, adding the current regression vector $\phi(t)$ to the trained network allows us to estimate the corresponding output. In Phase Two, a controller tuner is directly computed based on the instantaneous linearization of the NNARX model so as to solve the PID control parameter design problem. Some heuristic rules are incorporated to polish the calculated PID control parameters for matching the practical

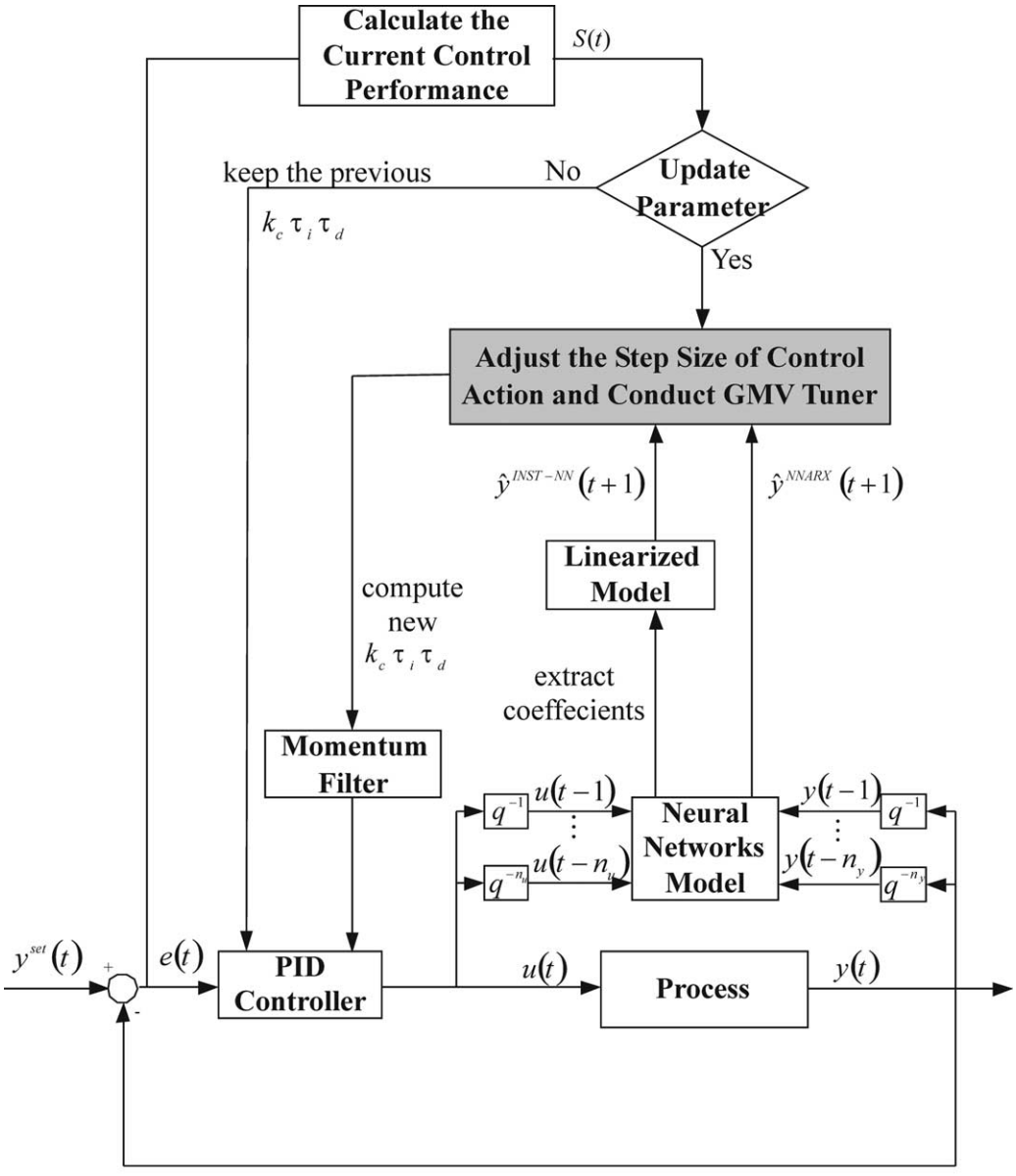

Fig. 5. The scheme of the optimal PID controller parameter design based on the instantaneous linearization of NNARX.

applications. Fig. 5 schematically depicts the on-line design procedure. The detailed procedure is summarized as follows:

Phase One: Identify the relationship between the input and the output to predict the output at the next step.

Step 1: Train a NNARX model based on the experimental data. Once trained, the NNARX model represents a nonlinear or complex function for the output that it learned. Set $k=1$.

Phase Two: Determine the current optimal PID controller parameters.

Step 2: At the new sampling time k, compute the performance of the current data window [Eq. (28)] and determine whether the control parameters should be updated. If the current performance $S(t)$ is below its control limit, the PID controller parameters are kept on the same values; and go to Step 7. Otherwise, the update control parameters are carried out; and go to Step 3.

Step 3: Extract the linearized model through the linearization of the NNARX model around the current input–output pair.

Step 4: With the current linearized model and the difference between the predicted output and the process output, determine the update control parameters using Eqs. (24) and (25).

Step 5: Measure the accuracy of the approximated linear model by Eq. (29). If the weighted penalty needs to be changed in order to resize the control action, go to Step 4 to recompute the new updated parameters based on the adjusted penalty; otherwise, go to Step 6. In Fig. 5, the gray region for the adjustment procedures of the penalty is zoomed in to describe the adjsuted penalty procedure in Fig. 6.

Step 6: Add the momentum filter to the change of the controller parameters using Eq. (27).

Step 7: Implement the new control action on the process based on the calculated PID control parameters. Collect the current process input–output. Then set

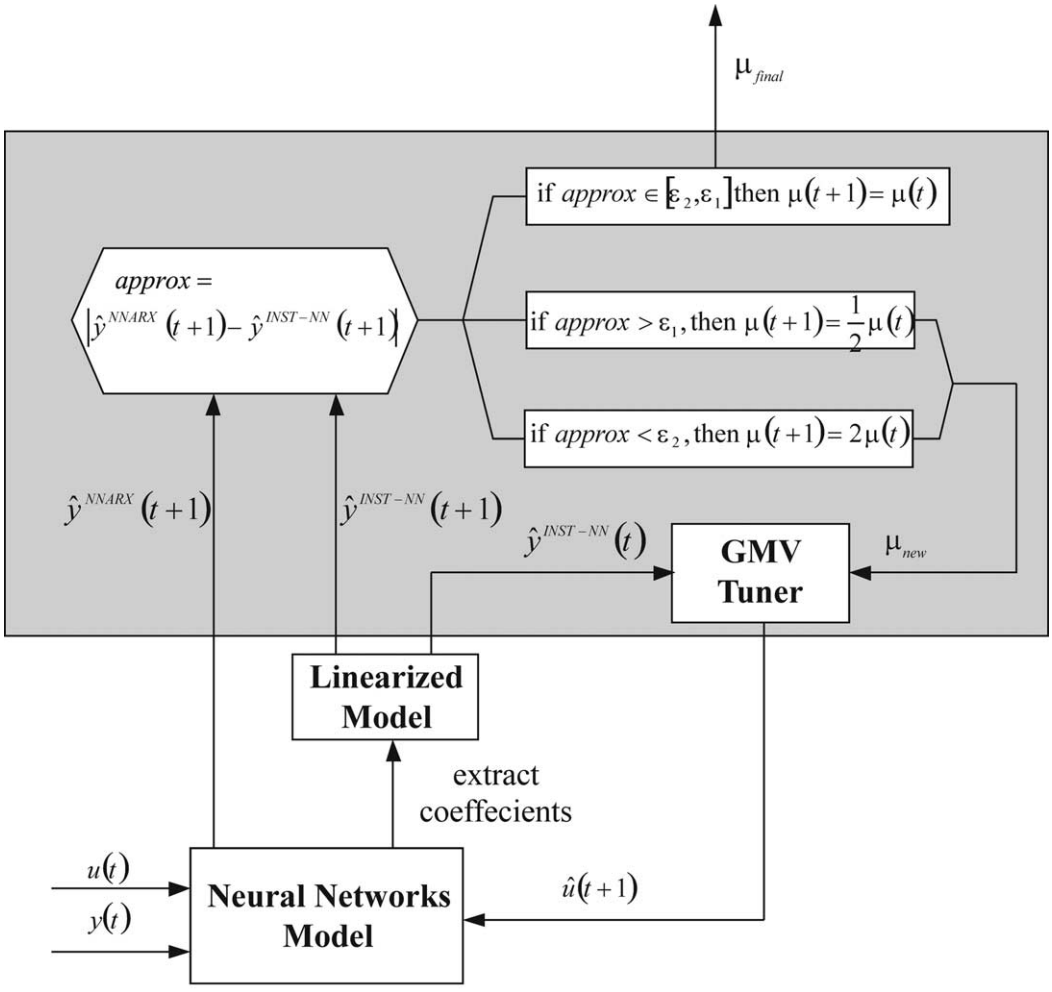

Fig. 6. The scheme of the adjustment of the size of the control step by the change of the penalty parameter. If the control action step does not yield a value less than ε_1 for *approx*, the control action is recomputed by$\mu(t+1)=2\mu(t)$ in order to get a small control action. If the control action does produce a smaller value ε_2 for *approx*, μ is divided by $\mu(t+1)=0.5\mu(t)$ in order to increase the size of the control action.

$k = k + 1$ and repeat the procedure from Step 2 to Step 7 for the next time step.

5. Illustration examples

In this section, two simulation examples involving process of different dynamics are discussed to demonstrate the wide applicability of the proposed on-line updated tuning algorithm. The proposed algorithm shown as follows is called INST-NNPID, which stands for the on-line updated PID controller with the instantaneous linearized NNARX model. INST-NNPID provides improved performance over the traditional self-tuning PID algorithms when applied to nonlinear processes. Each of these examples will be discussed separately in the sub-sections as follows.

5.1. Example 1: nonlinear pH neutralization system

pH neutralization is quite common in the chemical process. This example of dynamic modeling is based on the case of Nahas et al. [14]. The pH CSTR system shown in Fig. 7 has three input streams, including acid (HNO_3), buffer ($NaHCO_3$) and base stream (NaOH) respectively. The extremely nonlinear relationships are consisted of two reaction invariants, three nonlinear ordinary equations and one nonlinear algebraic equation.

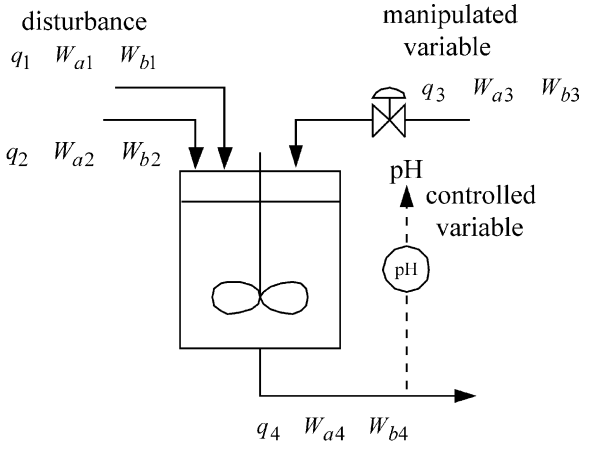

Fig. 7. The scheme diagram of pH CSTR.

Charge balance

$$W_a \equiv [H^+] - [OH^-] - [HCO_3^-] - 2[CO_3^=] \tag{30}$$

Carbonate ionbalance

$$W_b \equiv [H_2CO_3] + [HCO_3^-] + [CO_3^=] \tag{31}$$

$$\frac{dh}{dt} = \frac{1}{A}(q_1 + q_2 + q_3 - C_v h^{0.5}) \tag{32}$$

$$\frac{dW_{a4}}{dt} = \frac{1}{Ah}[(W_{a1} - W_{a4})q_1 + (W_{a2} - W_{a4})q_2 + (W_{a3} - W_{a4})q_3] \tag{33}$$

$$\frac{dW_{b4}}{dt} = \frac{1}{Ah}[(W_{b1} - W_{b4})q_1 + (W_{b2} - W_{b4})q_2 + (W_{b3} - W_{b4})q_3] \tag{34}$$

$$W_{a4} + 10^{14-pH} + W_{b4}\frac{1 + 2 \times 10^{pH-pK_2}}{1 + 10^{pK_1-pH} + 10^{pH-pK_2}} - 10^{-pH} = 0. \tag{35}$$

In Eqs. (30)–(35), h is the liquid level, W_{a4} and W_{b4} are the reaction invariants of the effluent stream, and q_1, q_2 and q_3 are the acid, buffer and base flow rates, respectively. The controlled variable is the pH value (pH). The nominal conditions and the operating parameters are listed in Table 2.

In the initial phase, a set of training data representing the process is obtained. Then a NNARX model is developed for the process under study. The dynamic mathematical model is for the response of pH to changes in q_3 flow rate. When a pseudo-random variation q_3 between 850 and 1050 ml/min is added to the process to generate the variation of pH between 6.5 and 9. Two past inputs and two past outputs are used to construct the NNARX model. A test data is also undertaken to verify the prediction capability of the trained NNARX in order to see if the NNARX model is good for the use of the proposed method [4,13].

5.1.1. Setpoint change

The setpoints of pH values have been changed from 7 to 9 and then from 9 to 8. First the response to the

Table 2
Simulation conditions for Example 1

$A = 207$ cm^2	$C_v = 8.75$ ml cm^{-1} s^{-1}	$pK_1 = 6.35$
$pK_2 = 10.25$	$W_{a1} = 3 \times 10^{-3}$ M	$W_{a2} = -3 \times 10^{-2}$ M
$W_{a3} = -3.05 \times 10^{-3}$ M	$W_{b3} = 5 \times 10^{-5}$ M	$W_{b1} = 0$
$W_{b2} = 3 \times 10^{-2}$ M	$q_1 = 16.6$ ml s^{-1}	$q_2 = 0.55$ ml s^{-1}
$q_3 = 15.6$ ml s^{-1}	$C_1 = 0.003$ M HNO_3	$C_2 = 0.03$ M $NaHCO_3$
$C_3 = 0.003$ M NaOH0.00005 M $NaHCO_3$		
Steady state:	$h = 14.0$ cm	$pH_4 = 7.0$

controlled system with the fixed controller parameters without updating is tested. When the system is operated at a pH around 8, the controller parameters are computed based on the minimum ITAE (integral of the time-weighted absolute error) tuning formula [23]. The plot in Fig. 8 shows that the fixed tuning parameters result in significant overshoot and oscillatory behavior. As would be expected, the adaptive tuning parameters are needed for the nonlinear process. Fig. 9 demonstrates the performance using INST-NNPID. The penalty parameter is fixed at a larger value ($\mu = 20$) due to this high nonlinearity of the pH CSTR system. More time is required to adjust the process behavior after the setpoint is changed. However, the controlled results will aggressively react to the change of the setpoint beforehand if a smaller value of μ is selected. Using the proposed rule to automatically adjust μ, when the adaptive penalty is taken into consideration, the control response and each updated control parameter are separately shown in Figs. 10 and 11. Unlike the previous study with fixed penalty, the PID controller with the adaptive penalty of the process output follows the desired setpoint much more quickly. Fig. 12 also provides the reason why the variable penalty should be changed. When the setpoint is changed at time 25, there is a significant difference between the predictions from $\hat{y}^{\text{INST-NN}}(t+1)$ and $\hat{y}^{\text{NNARX}}(t+1)$. The penalty tends to increase for the reduced aggressive control action. When the difference between $\hat{y}^{\text{INST-NN}}(t+1)$ and $\hat{y}^{\text{NNARX}}(t+1)$ is smaller enough between time 45 and 75, the process output is closed to the setpoint. The penalty tends to decrease for the increased aggressive control action.

In the past, many self-tuning controllers (STC) were developed. STC emphasizes the combination of a recursive identification procedure and a selected controller synthesis. Despite intensive research activities, different branches of STC focus only on the control design method while the process model still follows a regression standard model form by applying the recursive least square method. The STC methods include self-tuning PID controllers (STPID) [6], modified Ziegler–Nichols PID controllers (MZNPID) [5], etc. MZNPID leads to more aggressive control action because stability is considered rather than the control performance. Thus, MZNPID may not be good for our purposes. On the other hand, the STPID scheme requires careful selection of the prefilter coefficients and the integral action constant to obtain more stable responses. Thus, a trial-and-error procedure is required to obtain these tuning parameters, sometimes causing bigger oscillatory responses. The behavior of the closed loop response in the process for these design methods is shown in Fig. 13. Besides, the proposed design method with a recursive linear model (called REC-LINPID) is also tested and shown in Fig. 13. This algorithm, unlike STPID, also performs well without any prior design parameter. For fair comparison of methods, the numerical measures of the performance based on SSE are listed in Table 3. The lack of mechanism cannot quickly update the model to

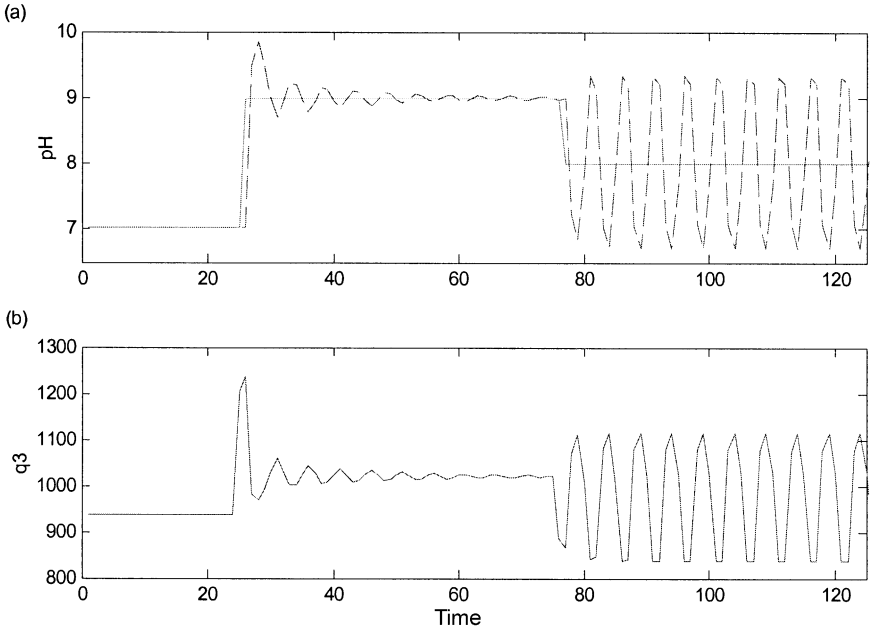

Fig. 8. PID controller with the fixed controller parameters in setpoint change results for Example 1: (a) pH (dashed line) and setpoint (solid line); (b) q_3.

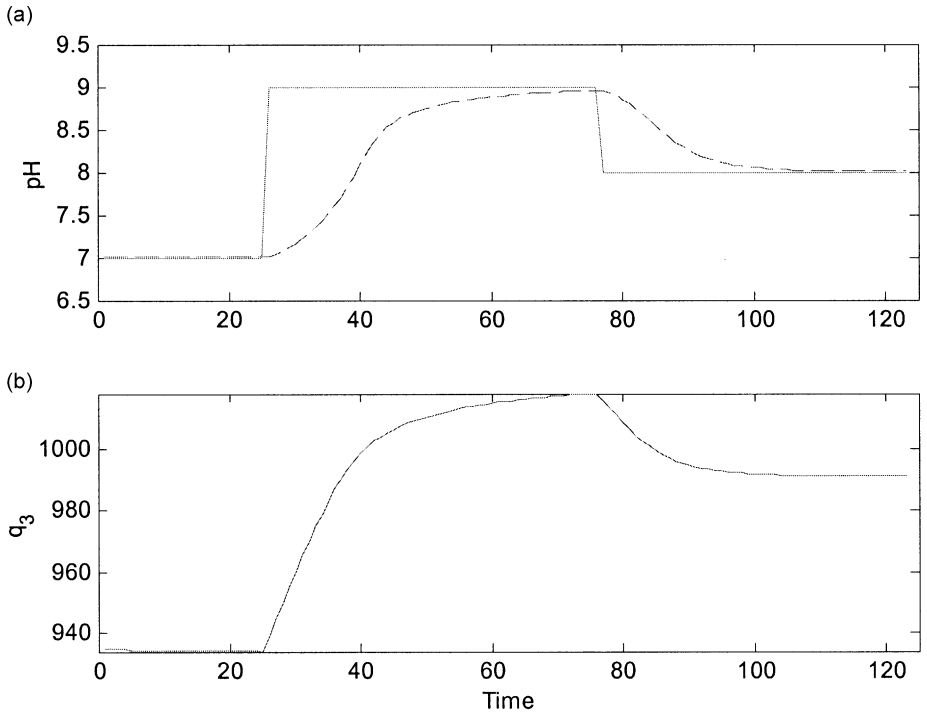

Fig. 9. INST-NNPID with the fixed penalty ($\mu = 20$) in setpoint change results for Example 1: (a) pH (dashed line) and setpoint (solid line); (b) q_3.

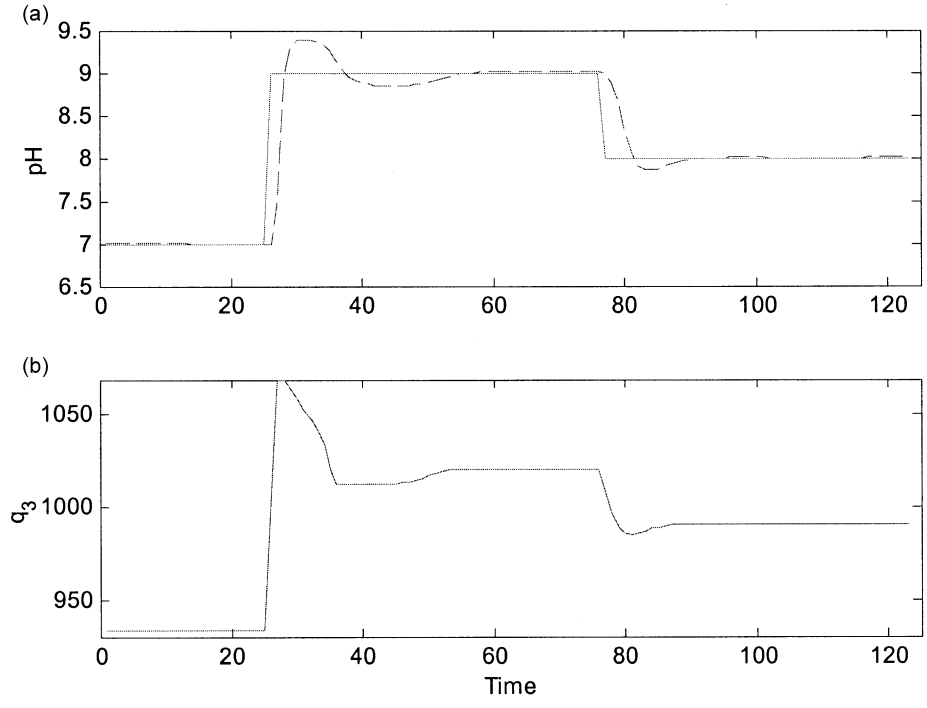

Fig. 10. INST-NNPID with the adaptive penalty in setpoint change results for Example 1: (a) pH (dashed line) and setpoint (solid line); (b) q_3.

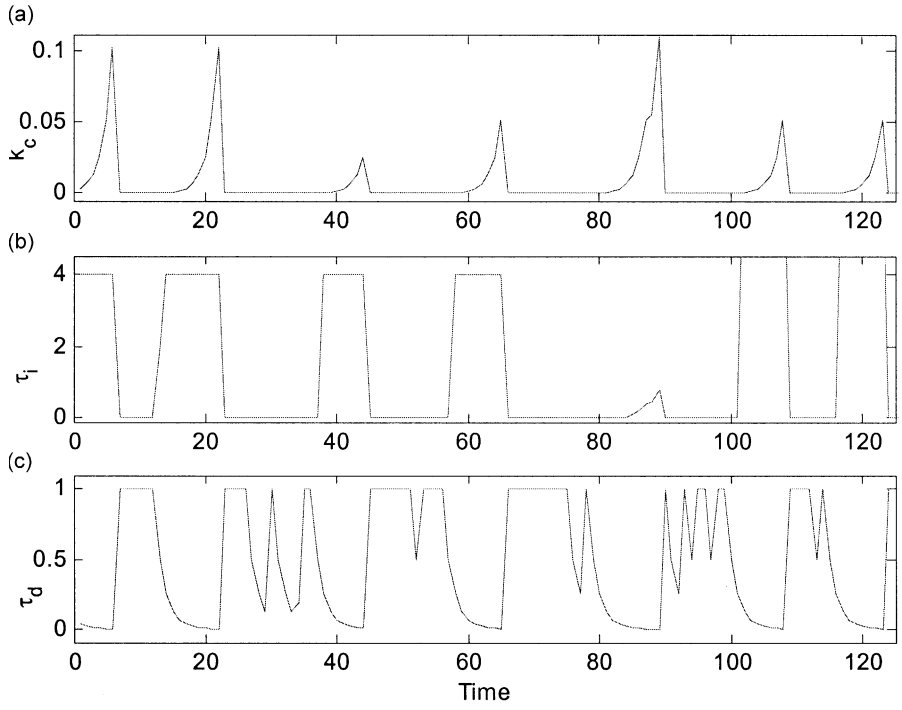

Fig. 11. The self-tuning PID controller parameters in setpoint change results for Example 1: (a) k_c; (b) τ_i; (c) τ_d.

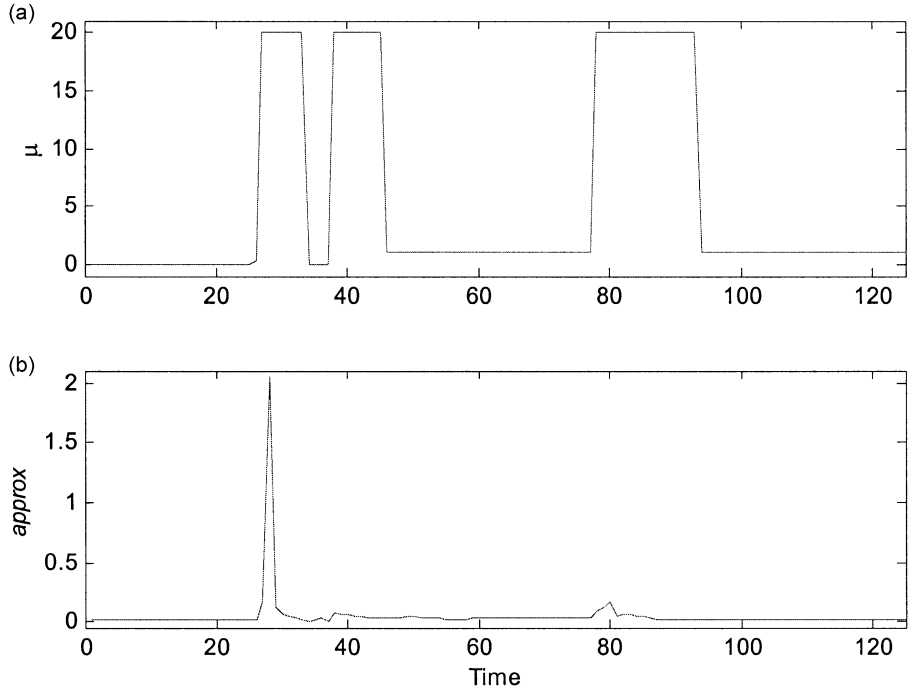

Fig. 12. The adjustment μ with the approximated error $\left|\hat{y}^{\text{NNARX}}(t+1) - \hat{y}^{\text{INST-NN}}(t+1)\right|$ in the setpoint change results for Example 1: (a) μ; (b) the approximated error.

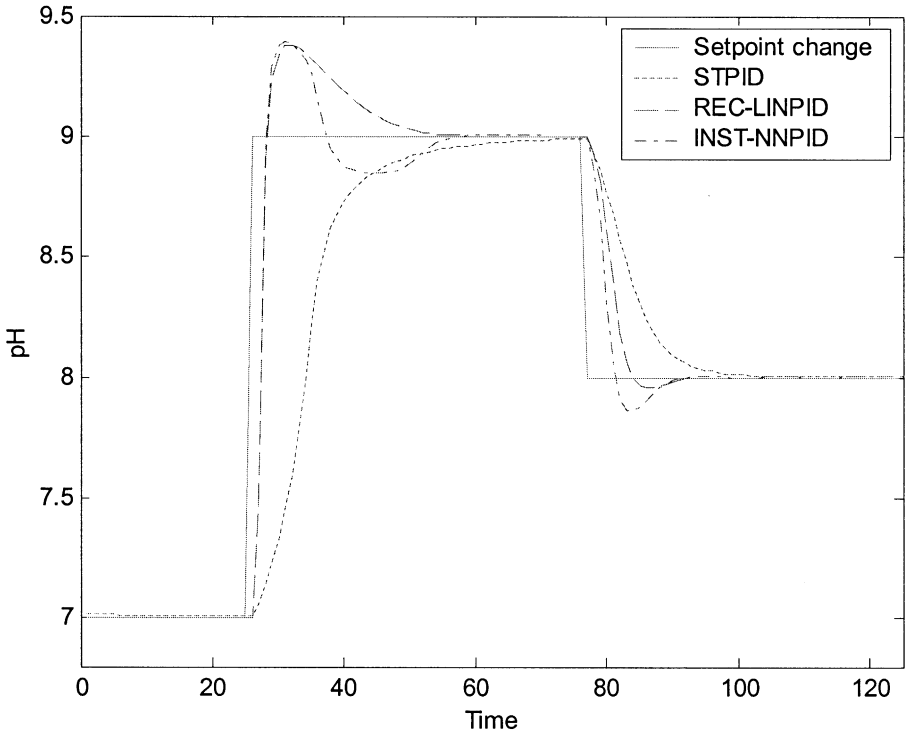

Fig. 13. Comparison between INST-NNPID, REC-LINPID and STPID in setpoint change results for Example 1.

Table 3
Comparison SSE among the different adaptive PID controller algorithms

Example	Model	INST-NNPID	REC-LINPID	STPID
1	SSE (in setpoint change)	17.5477	19.4597	30.4284
	SSE (in disturbance change)	0.1659	–	0.2782
2	SSE (in tracking trajectory)	13 764	24 373	–

cope with an actual nonlinear chemical process, so the performances of STPID and REC-LINPID are worse than that of the proposed method.

5.1.2. Disturbance change

The disturbance variable, q_2 changes from 990 to 1010 ml/min at time 50 and from 1010 to 1020 ml/s at time 100. Fig. 14 depicts the rejected disturbance results. REC-LINPID causes unstable responses because the penalty cannot be properly adjusted. It is not included in Fig. 14. In Table 3, the corresponding SSE values also indicate that, even without measured disturbances, the proposed strategy for the nonlinear control design is appropriate.

Based on the above results, the proposed method not only has less computation load when INST-NNPID is used but also has good performance at the setpoint change and the disturbance change even though the pH neutralization is highly nonlinear.

5.2. Example 2: nonlinear batch reactor

The batch reactor is sketched in Fig. 15. The reactant is fed into the reactor. Assume the mixture is blended well in the reactor and the liquid density is kept constant during operation. The consecutive reaction takes place in the reactor as:

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C} \tag{36}$$

where A $\rightarrow$ B has second-order kinetics and B $\rightarrow$ C has first-order kinetics. It is noted that the energy dissipation induced by the shaft work and heat loss to the surroundings is ignored. According to the above assumptions, the mass balances for components A and B and the energy balance are formulated as follows:

$$\frac{\mathrm{d}C_\mathrm{A}}{\mathrm{d}t} = -k_1 C_\mathrm{A}^2, \quad C_\mathrm{A}(0) = C_{\mathrm{A}0} \tag{37}$$

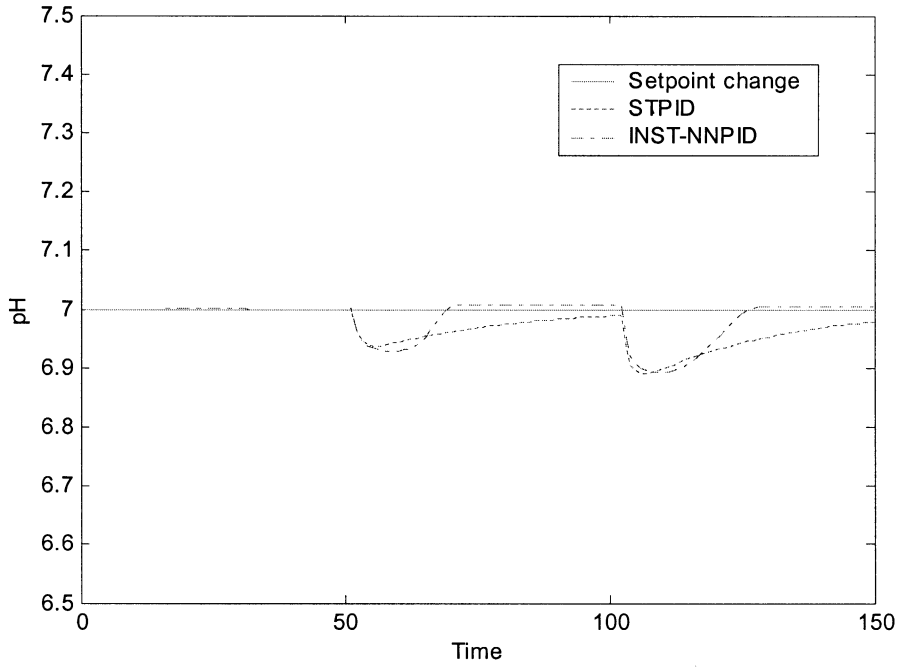

Fig. 14. Comparison of INST-NNPID, REC-LINPID and STPID in disturbance change results for Example 1.

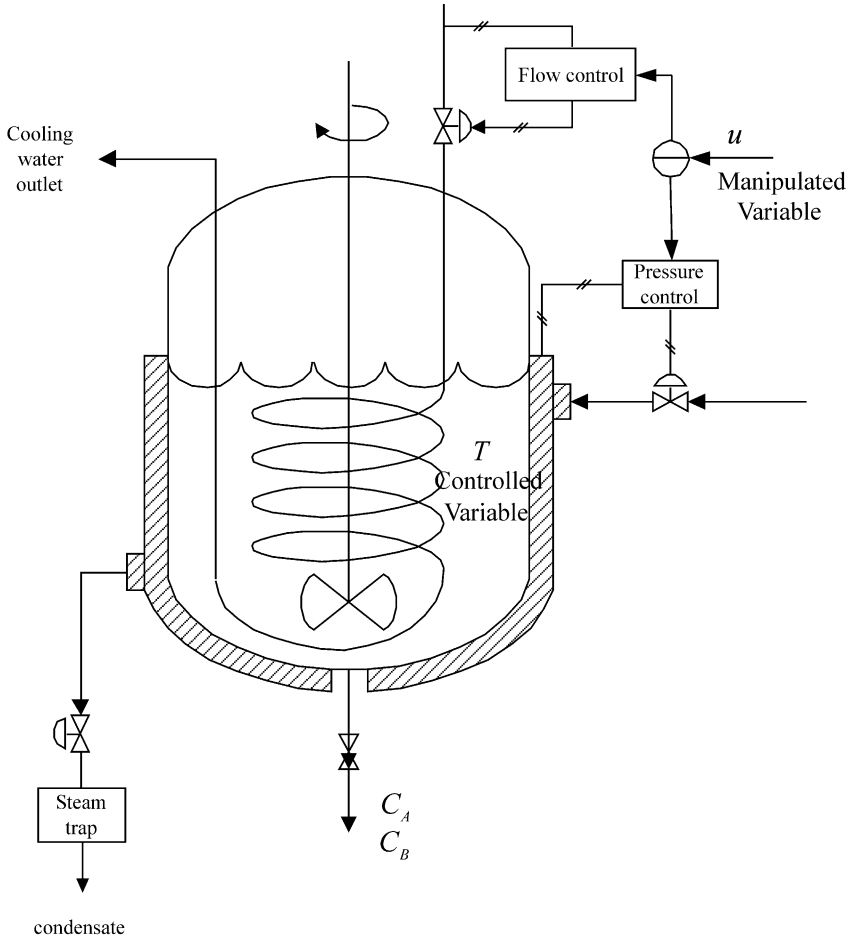

Fig. 15. The scheme diagram of the batch reactor.

Table 4
Simulation conditions for Example 2

$A_{10} = 1.1\ \mathrm{m^3\ kmol^{-1}\ s^{-1}}$	$\gamma_1 = 41.8\ ^\circ\mathrm{C\ s^{-1}}$	$\beta_2 = 0.0515\ ^\circ\mathrm{C\ s^{-1}}$
$A_{20} = 172.2\ \mathrm{s^{-1}}$	$\gamma_2 = 83.6^\circ\mathrm{C\ s^{-1}}$	$C_A(0) = 1\ \mathrm{kmol\ m^{-1}}$
$E_1 = 20900\ \mathrm{kJ\ kg^{-1}\ K^{-1}}$	$\alpha_1 = 4.3145\ ^\circ\mathrm{C\ s^{-1}}$	$C_B(0) = 0\ \mathrm{kmol\ m^{-1}}$
$E_2 = 41800\ \mathrm{kJ\ kg^{-1}\ K^{-1}}$	$\alpha_2 = -0.1099\ ^\circ\mathrm{C\ s^{-1}}$	$T(0) = 25^\circ\mathrm{C}$
$R = 8.3143\ \mathrm{kJ\ kmol^{-1}\ K^{-1}}$	$\beta_1 = 1.4962\ ^\circ\mathrm{C\ s^{-1}}$	

$$\frac{\mathrm{d}C_\mathrm{B}}{\mathrm{d}t} = k_1 C_\mathrm{A}^2 - k_2 C_\mathrm{B}, \quad C_\mathrm{B}(0) = C_{\mathrm{B}0} \tag{38}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \gamma_1 k_1 C_\mathrm{A}^2 + \gamma_2 k_2 C_\mathrm{B} + (\alpha_1 + \alpha_2 T) + (\beta_1 + \beta_2 T)u \tag{39}$$

where

$$\begin{aligned}
\gamma_1 &= -\Delta H_1/\rho C_p \\
\gamma_2 &= -\Delta H_2/\rho C_p \\
\alpha_1 &= \left(U_j A_j T_{s,\min} + U_{c,\max} A_c T_c\right)/\rho C_p V \\
\alpha_2 &= -\left(U_j A_j + U_{c,\max} A_c\right)/\rho C_p V \\
\beta_1 &= \left[U_j A_j \left(T_{s,\max} - T_{s,\min}\right) - \left(U_{c,\max} - U_{c,\min}\right) A_c T_c\right]/\rho C_p V \\
\beta_2 &= \left(U_{c,\max} - U_{c,\min}\right) A_c/\rho C_p V
\end{aligned} \tag{40}$$

k_1 and k_2 are the reaction rate constants which are temperature-dependent,

$$\begin{aligned}
k_1 &= A_{10}\exp(-E_1/RT) \\
k_2 &= A_{20}\exp(-E_2/RT).
\end{aligned} \tag{41}$$

The parametric variable u lies in the region of 0 and 1. When $u = 0$, it represents the maximum cooling and minimum heating; on the other hand, when $u = 1$, it represents the minimum cooling and maximum heating. Hence, the input constraint ($0 \leqslant u \leqslant 1$) of this case needs to be considered during control. The process measured T with noise $N(0, 1.0)$ is presented. The parameter definitions are listed in Table 4. In this case, the control objective is to maximize the yield of component B by keeping the operation under the following optimal temperature trajectory [21]:

$$T_d(t) = 54 + 71\exp\left(-2.5 \times 10^{-3} t\right). \tag{42}$$

To train the NNARX model, the open-loop data of output (T) driven by pseudo random binary sequence inputs (u) are first collected. The time interval for each input change is 30 s, the sampling time is 1 s and the overall data number is 3000. The data set is imposed

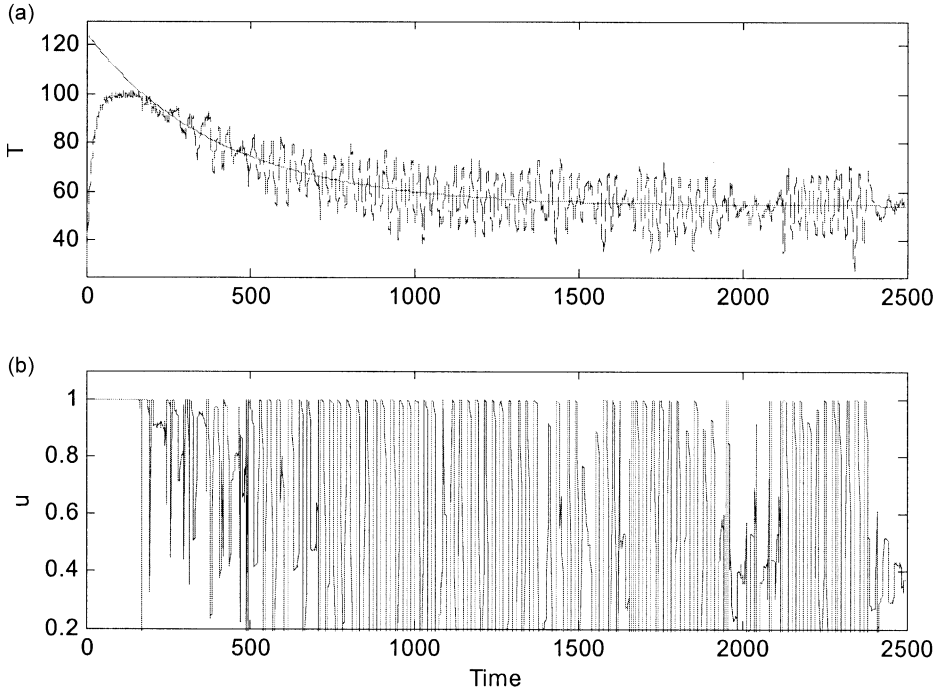

Fig. 16. INST-NNPID without the heuristic rules (i) and (ii) in tracking trajectory results for Example 2: (a) T (dashed line) and setpoint (solid line); (b) u.

on the NNARX model with different input lag terms to search for the appropriate weights in the model. The smallest overall prediction error results from $n_y = n_u = 3$. A set of validation data is also generated in the same manner for verification of the developed model before implementing the model for the rest of the control study.

After the predicted model is developed, the self-tuning control can be applied to the trajectory tracking of the batch case. Fig. 16 demonstrates the tracking ability of the algorithms. The control action of the manipulated signal u is depicted at the bottom of the figure. The control output can follow the setpoint. Fig. 17 indicates that the large, high-frequency variation in the control

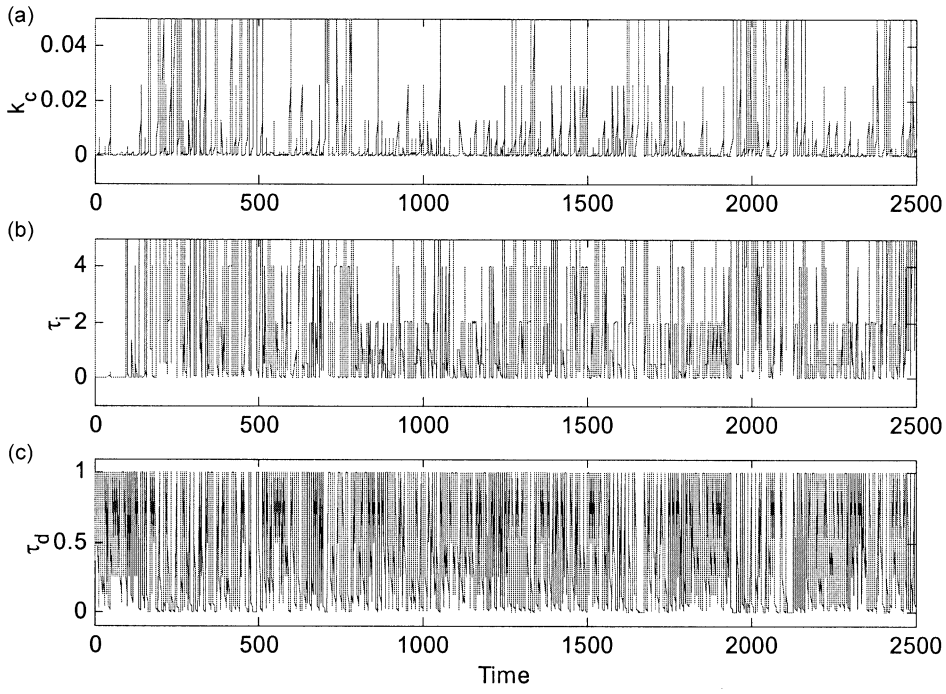

Fig. 17. INST-NNPID controller parameters without the heuristic rules (i) and (ii) in tracking trajectory results for Example 2: (a) k_c; (b) τ_i; (c) τ_d.

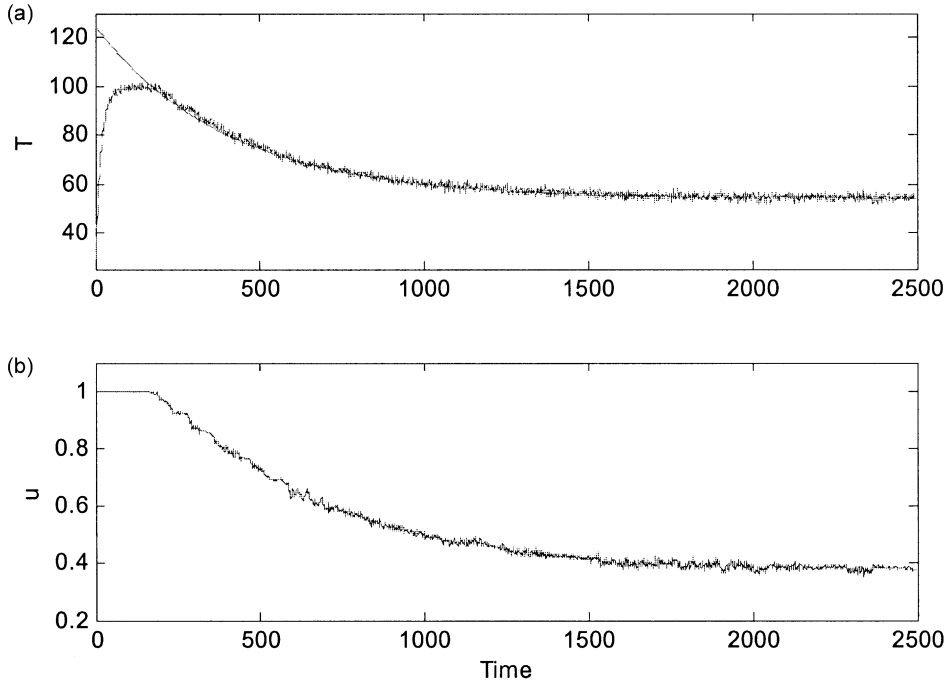

Fig. 18. INST-NNPID in tracking trajectory results for Example 2: (a) T (dashed line) and setpoint (solid line); (b) R.

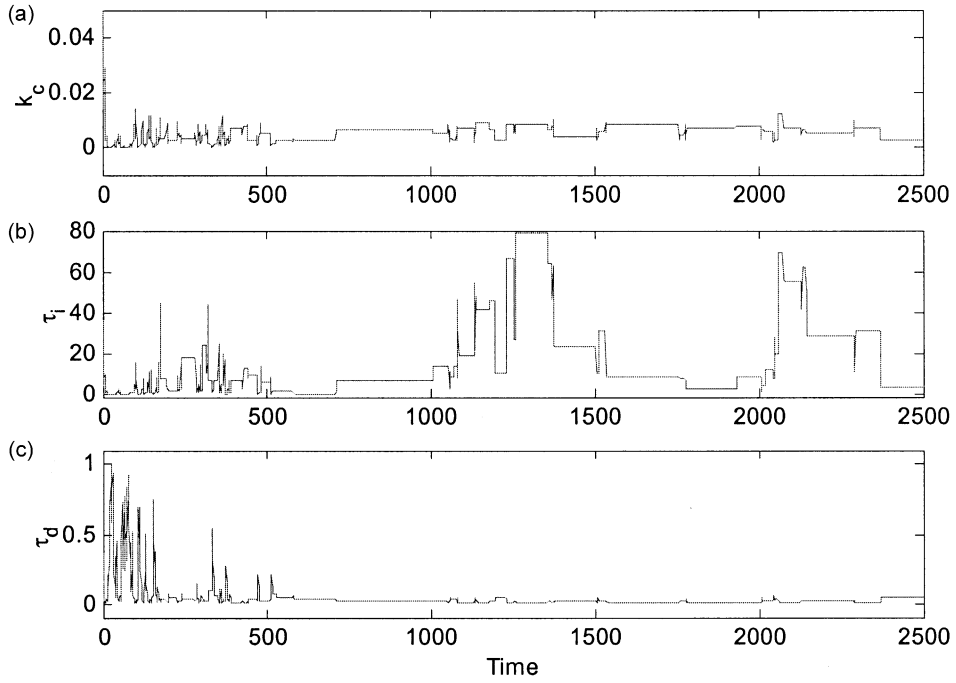

Fig. 19. INST-NNPID controller parameters in tracking trajectory results for Example 2: (a) k_c; (b) τ_i; (c) τ_d.

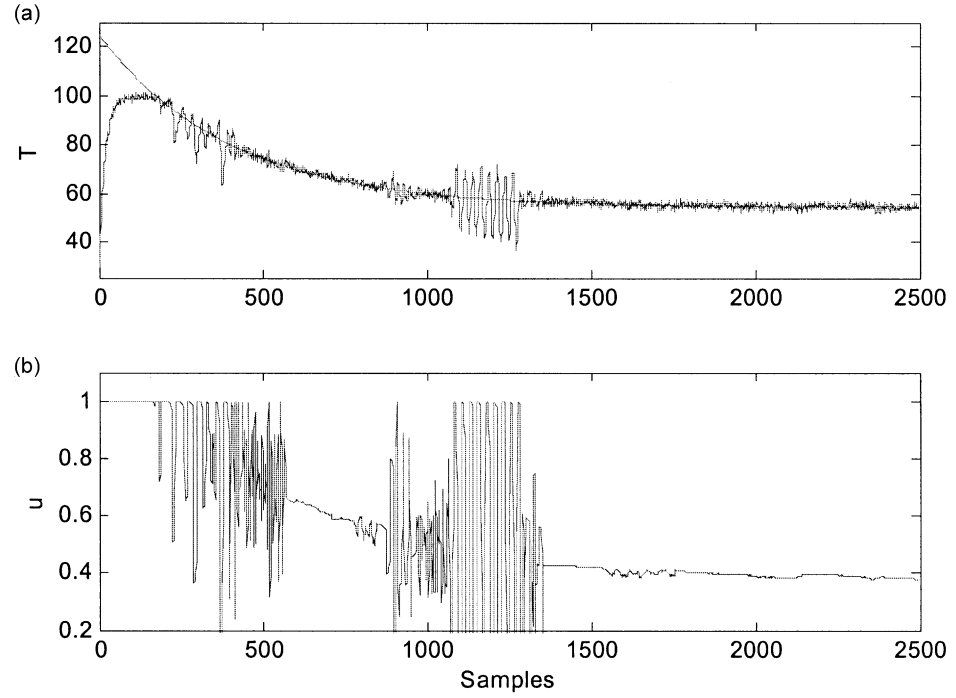

Fig. 20. REC-LINPID in tracking trajectory results for Example 2: (a) T (dashed line) and setpoint (solid line); (b) R.

parameters still exists because of the noise measurement to a large extent even if the output temperature is around the desired setpoint. By applying heuristic rules (i) and (ii), the momentum filter and the updated condition (with $m = 5$), a much more satisfactory dynamic response and much smoother performance is achieved (Figs. 18 and 19).

Because of the linear model error and difficulty in tuning STPID parameters, only the results of the self-tuning control design based on REC-LINPID are only considered here. In Fig. 20, the control outputs based on REC-LINPID can follow the setpoint; however, REC-LINPID has significant variation in trajectory tracking due to the improperly updated ability in the

current operational model. The proposed method has better trajectory tracking ability than REC-LINPID since the former is based on the instantaneous linearized model, and the latter, based on the linear model.

From a computational perspective, if the proposed method is based on the neural network model, the optimal control action is iterative in order to converge to an acceptable accuracy. This strategy may become unrealistic for controlling the nonlinear dynamic process. By applying INST-NNPID, the solution has a unique minimum that can be found directly.

6. Conclusion

A new control approach using a minimum variation controller and a linearized neural network model around its current operation region is developed. Like gain scheduling, the instantaneous linearization technique could be used to design a gain-scheduling type of the control system at each time interval. However, as opposed to gain scheduling control, the control parameters based on the neural network model have an infinite scheduling resolution for updating parameters. The proposed method has several advantages. First, the linear control scheme applied to the nonlinear control design has less computation compared with the nonlinear control counterpart based on the nonlinear neural network model. Second, the nonlinear characteristics of neural networks can be incorporated into the control design. Third, this method provides a useful physical interpretation of the dynamics process. However, there are several serious drawbacks. First, the linearized model may be valid only in a narrow region when the process around the current operating range is very nonlinear. Second, the measurement noise and the model–process mismatch may generate the defected linearized model. These drawbacks could have a serious impacts on controller design. To overcome these problems, several variations of the heuristic rules, including the momentum filter, the updated condition and the adjusted step size of the control action, are developed to make the proposed algorithm more practical. The control strategy is the balance of the nonlinear and the conventional linear control designs to improve the control performance for the nonlinear process. Without the convergence problem, a unique minimum of the control design can be found directly because the model is linear. Besides, the implementation is much simpler and computationally less demanding. This strategy is particularly suitable for the chemical process that is characterized by high nonlinearity and analytical difficulty. The proposed method is demonstrated using two examples of a continuous pH neutralization process and a batch reactor. The results show that the proposed method with the heuristic rules is effective in following the setpoint and reducing the variance of the output caused by unknown disturbance. Although the proposed method yields encouraging outcomes, all the results presented in this paper are based on the simulation models. Additional work using a real plant should be conducted in the future.

Acknowledgements

This work was partly sponsored by the National Science Council, R.O.C. and by the Ministry of Economic, R.O.C.

References

[1] M.S. Ahmed, I.A. Tasadduq, Neural-net controller for nonlinear plants: design approach through linearisation, IEE Proc-Control Theory Appl 141 (1994) 315.
[2] K.J. Aström, T. Hägglund, PID Controllers: theory design and tuning, Instrument Society of America, 1995.
[3] N. Bhat, T.J. McAvoy, Use of neural nets for dynamic modeling and control of chemical process systems, Comput. Chem. Eng. 14 (1990) 573.
[4] S.A. Billings, H.B. Jamaluddin, S. Chen, Properties of neural networks with applications to modeling non-linear dynamic systems, Int. J. Control 55 (1992) 193.
[5] V.J. Bohim Bobal, R. Prokop, Practice aspects of self-tuning controllers, Int. J. Adapt. Control Signal Process. 13 (1999) 671.
[6] F. Cameron, D.E. Seborg, A self-tuning controller with a PID structure, Int. J. Control 30 (1983) 401.
[7] J. Chen, Systematic derivations of model predictive control based on artificial neural network, Chem. Engin. Commun. 164 (1998) 35.
[8] D.W. Clark, P.J. Gawthrop, Self-tuning control, in: Proc. IEE, Pt-D 126 (1979) 633.
[9] T.F. Edgar, D.M. Himmelblau, L.S. Lasdon, Optimization of Chemical Process, second ed., McGraw-Hill, 2001.
[10] W. Fuli, L. Mingzhong, Y. Yinghua, Neural network pole placement controller for nonlinear systems through linearisation, American Control Conference (1997) 1984.
[11] P.J. Gawthrop, Self-tuning PID controllers: algorithm and implementation, IEEE Trans. Autom. Control 31 (1986) 201.
[12] K. Hornik, M. Stinchcombe, H. White, Mutliplayer feedforward neural networks are universal approximators, Neural Networks 2 (1989) 359.
[13] I.J. Leontaritis, S.A. Billings, Model selection and validation methods for non-linear systems, Int. J. Control 45 (1987) 311.
[14] E.P. Nahas, M.A. Henson, D.E. Seborg, Nonlinear internal model control strategy for neural network models, Comput. Chem. Eng. 16 (1992) 1039.
[15] K.S. Narendra, K. Parthasarathy, Identification and control of dynamic systems using neural networks, IEEE Trans. Neural Networks 1 (1990) 4.
[16] M. Nikolaou, V. Hanagandi, Control of nonlinear dynamical systems modeled by recurrent neural networks, Am. Inst. Chem. Eng. J. 39 (1993) 1890.
[17] R. Ortega, R. Kelly, PID self-tuners: Some theoretical and practice aspects, IEEE Trans. Ind. Electron. 31 (1984) 312.
[18] C.G. Proudfoot, P.J. Gawthrop, O.L.R. Jacobs, Self-tuning PI control of a pH neutralization process, in: Proc. IEE, Pt-D 130 (1983) 267.
[19] D.C. Psichogios, L.H. Ungar, Direct and indirect model based

control using artificial neural networks, Ind. Engin. Chem. Res. 30 (1991) 2564.
[20] F. Radke, R. Isermann, A parameter-adaptive PID controller with stepwise parameter optimization, Automatic 23 (1987) 449.
[21] W. H. Ray, S. Szelely, Process Optimization, Wiley, New York, 1973.
[22] A. Saiful, S. Omatu, Neuromorphic self-tuning PID controller, in: Proc. 1993 IEEE ICNN, San Francisco (1993) 552.
[23] C.A. Smith, A.B. Corripio, Principles and Practice of Automatic Process Control, 2nd ed., John Wiley & Sons, 1997.
[24] H.T. Su, T.J. McAvoy, Integration of multilayer perceptron networks and linear dynamic models: a Hammerstein Modeling Approach, Ind. Engin. Chem. Res. 32 (1993) 1927.
[25] B. Wittenmark, Self-tuning PID Controllers Based on Pole Placement, Lund Institute Technical Report TFRT-7179, 1979.
[26] Y.-K. Yeo, T.-I. Kwon, A Neural PID controller for the pH neutralization process, Ind. Engin. Chem. Res. 38 (1999) 978.
[27] D.L. Yu, J.B. Gomm, D. Williams, On-line predictive control of a chemical process using neural network models, in: IFAC 14th Triennial World Congress, Beijing, PR China, 1999, pp. 121–126.