Iterative Identification of Neuro-Fuzzy-Based Hammerstein Model with Global Convergence

Li Jia,[†] Min-Sen Chiu,*[,†] and Shuzhi Sam Ge[‡]

Departments of Chemical & Biomolecular Engineering and Electrical & Computer Engineering, National University of Singapore, Singapore 119260

In this paper, a neuro-fuzzy-based model is used to describe the nonlinearity of the Hammerstein process without any prior process knowledge, thus avoiding the inevitable restrictions on static nonlinear function encountered by using the polynominal approach. In doing so, a clustering algorithm is presented in order to identify the centers and widths of the neuro-fuzzy-based Hammerstein model, and an updating algorithm guaranteeing the global convergence of the weights of the model is developed based on the Lyapunov approach. As a result, the proposed method can avoid the problems of initialization and convergence of the model parameters, which are usually resorted to a trial and error procedure in the existing iterative algorithms used for the identification of Hammerstein model. Examples are used to illustrate the performance and applicability of the proposed neuro-fuzzy-based Hammerstein model.

1. Introduction

The Hammerstein model is a block-oriented nonlinear model as depicted in Figure 1, which consists of the cascade structure of a static nonlinear function, $N(\cdot)$, followed by a linear dynamic block $H(z)$. It has been shown that such a model structure can effectively represent and approximate many industrial processes. For example, the nonlinear dynamics of chemical processes, such as pH neutralization processes,[1,2] distillation columns,[3,4] heat exchangers,[1,3] polymerization reactor,[5,6] and dryer process,[7] have been modeled with the Hammerstein model. The Hammerstein model was first investigated by Narendra and Gallman[8] who proposed an iterative method to identify Hammerstein processes. Several other identification methods for Hammerstein models were presented.[9,10] Recently, Sung[11] proposed an identification method by using a special test signal that enables the decoupling of the identification of the linear dynamic part from that of a static nonlinear function. However, the aforementioned results assume that the unknown static nonlinear characteristic is of a polynomial form. As a result, if the nonlinearity is not continuous or not in the polynomial form, the algorithms do not converge.[12,13] Moreover, high-degree polynomials have oscillatory behavior and parameter estimation is often numerically ill-conditioned.[14]

To overcome the aforementioned problem, neural networks and fuzzy systems have been suggested to model the static nonlinearity of Hammerstein processes owing to their ability to model a nonlinear function to any arbitrary accuracy.[15] Consequently, this approach does not assume the structure of the static nonlinearity. Su and McAvoy[5] proposed a neural network Hammerstein modeling approach in which a two-step procedure is used to identify the nonlinear function separately from the linear part utilizing steady-state data and transient data, respectively. Duwaish and Karim[13] used a hybrid model which consists of a neural network in series with an ARMA model, and a recursive algorithm was derived to estimate the weights of neural network and the parameters of ARMA model simultaneously from input and output data. A similar idea was explored by Abonyi et al.[16] who proposed two identification methods for a fuzzy Hammerstein model, which consists of a static fuzzy model connected in series with a linear dynamic model. However, a rigorous analysis of the convergence of the iterative algorithms developed[5,13,16] is currently not available.

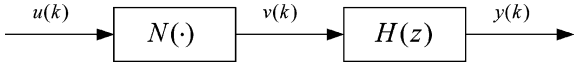

Figure 1. The structure of the Hammerstein model.

To circumvent the existing problems in Hammerstein model identification, a neuro-fuzzy-based identification method for Hammerstein processes is presented in this paper. More specifically, a neuro-fuzzy-based model is used to describe the nonlinearity of the Hammerstein process without any prior process knowledge, thus avoiding the inevitable restrictions on static nonlinear function encountered by using the polynominal approach. In addition, the identification of the linear dynamic part and the static nonlinear function are carried out independently by using the special input signal adopted from Sung.[11] To identify the proposed model, a clustering algorithm is presented in order to identify the antecedent parameters, i.e., the centers and widths of the neuro-fuzzy-based Hammerstein model. Furthermore, an updating algorithm for the consequent parameters (or weights) of the model is developed based on the Lyapunov approach. As a result, the proposed method can avoid the problems of initialization and convergence of the model parameters, which are usually resorted to the trial and error procedure in the previous work.

The rest of this paper is organized as follows. The Hammerstein process identification problem and neuro-fuzzy-based Hammerstein model are given in section 2. A neuro-fuzzy-based identification method is presented

* To whom correspondence should be addressed. Tel: (+65) 6874 2223. Fax: (+65) 6779 1936. E-mail: checms@nus.edu.sg.
[†] Department of Chemical & Biomolecular Engineering.
[‡] Department of Electrical & Computer Engineering.

10.1021/ie0495574 CCC: $30.25
Published on Web 02/11/2005

in section 3. Two simulation examples are given in section 4, followed by section 5 which concludes the work.

2. Neuro-Fuzzy-Based Hammerstein Model

2.1. Hammerstein Process. The Hammerstein process as depicted in Figure 1 consists of a static nonlinearity, $N(\cdot)$, and a linear dynamics, $H(z)$, as given by

$$v(k) = N(u(k)) \tag{1}$$

$$y(k+1) = a^{\mathrm{T}}\mathcal{Y}(k) + b^{\mathrm{T}}\mathcal{V}(k) \tag{2}$$

where

$$\begin{gathered}\mathcal{Y}(k) = [y(k), y(k-1), ..., y(k+1-n_a)]^{\mathrm{T}} \\ \mathcal{V}(k) = [v(k), v(k-1), ..., v(k+1-n_b)]^{\mathrm{T}} \\ a = [a_1, a_2, ..., a_{n_a}]^{\mathrm{T}} \\ b = [b_1, b_2, ..., b_{n_b}]^{\mathrm{T}}\end{gathered} \tag{3}$$

where $u(k)$ and $y(k)$ denote the process input and output at the kth sampling instant, respectively, $v(k)$ is an unmeasurable internal variable, a_i $(i = 1, 2, ..., n_a)$ and b_j $(j = 1, 2, ..., n_b)$ are the model parameters, n_a and n_b are integers related to the model order, and the function, $N(\cdot)$, represents the static nonlinearity of the process.

For brevity, let us define the following vectors

$$\begin{gathered}\psi(k) = [\mathcal{Y}^{\mathrm{T}}(k), \mathcal{V}^{\mathrm{T}}(k)]^{\mathrm{T}} \\ \theta = [a^{\mathrm{T}}, b^{\mathrm{T}}]^{\mathrm{T}}\end{gathered} \tag{4}$$

Equation 2 can then be represented as

$$y(k+1) = \theta^{\mathrm{T}}\psi(k) \tag{5}$$

The objective of the proposed identification scheme is to obtain a Hammerstein model such that the following cost function E is made acceptably small, i.e.,

$$E(\hat{N}(\cdot), \hat{\theta}) = \frac{1}{2N_P}\sum_{j=1}^{N_P}(\hat{y}(j) - y(j))^2 \leq \epsilon \tag{6}$$

for a given tolerance ϵ and subject to

$$\hat{v}(k) = \hat{N}(u(k)) \tag{7}$$

$$\hat{y}(k+1) = \hat{a}^{\mathrm{T}}\mathcal{Y}(k) + \hat{b}^{\mathrm{T}}\,\hat{\mathcal{V}}(k) = \hat{\theta}^{\mathrm{T}}\,\hat{\psi}(k) \tag{8}$$

where

$$\begin{gathered}\hat{\psi}(k) = [\mathcal{Y}^{\mathrm{T}}(k), \hat{\mathcal{V}}^{\mathrm{T}}(k)]^{\mathrm{T}} \\ \hat{\theta} = [\hat{a}^{\mathrm{T}}, b^{\mathrm{T}}]^{\mathrm{T}} \\ \hat{\mathcal{V}}(k) = [\hat{v}(k), \hat{v}(k-1), ..., \hat{v}(k+1-n_b)]^{\mathrm{T}} \\ \hat{a} = [\hat{a}_1, \hat{a}_2, ..., \hat{a}_{n_a}]^{\mathrm{T}} \\ \hat{b} = [\hat{b}_1, \hat{b}_2, ..., \hat{b}_{n_b}]^{\mathrm{T}}\end{gathered} \tag{9}$$

where $\hat{y}$ is the output of the Hammerstein model, eqs 7 and 8, $\hat{v}(k)$ is the estimated internal variable, $\hat{a}_i$ $(i = 1, 2, ..., n_a)$ and $\hat{b}_j$ $(j = 1, 2, ..., n_b)$ are the estimated model parameters, the function, $\hat{N}(\cdot)$, represents the estimated nonlinear static function, and N_P is the number of input and output data.

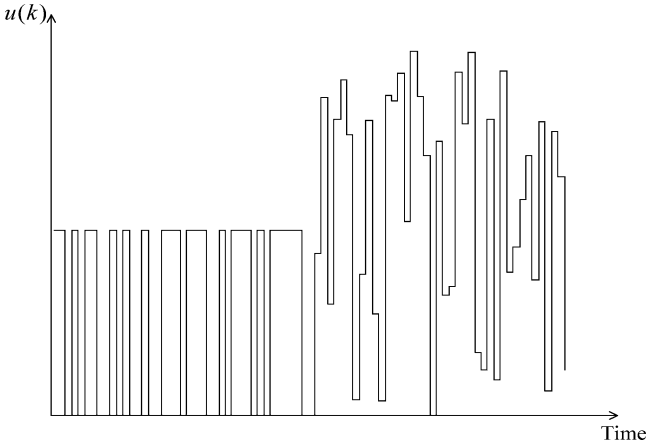

Figure 2. Illustration of the special test signal $u(k)$.

2.2. Neuro-Fuzzy-Based Hammerstein Model. In this paper, the Hammerstein process is identified by a neuro-fuzzy-based Hammerstein model in which the nonlinearity of the process $N(\cdot)$ is represented by a neuro-fuzzy model and the linear dynamics $H(z)$ by an ARX model. A special test signal in Figure 2 as proposed by Sung[11] is adopted to identify the Hammerstein process. For the binary input $u(k)$ as shown in Figure 2, the corresponding internal variable $v(k)$ would also display a binary signal of different magnitude from that of $u(k)$. As such, the linear dynamics can be identified within a constant gain factor by treating the input $u(k)$ as the internal variable $v(k)$. As a result, linear identification methods can be used to identify the linear dynamic model directly from the binary input signal and the corresponding output of the Hammerstein process. As noted by Sung,[11] the advantage of using this special signal results in the identification problem of the linear dynamic part separated from that of the nonlinear static part. This is because the binary signal is designed to excite the linear dynamic part without activating the nonlinear static part of the Hammerstein process, whereas the multistep signal is used to activate the nonlinearity of the Hammerstein process.

The static nonlinear function $N(\cdot)$ is represented by a neuro-fuzzy system, a multilayer feedforward network which integrates the Takagi-Sugeno fuzzy logic system and radial basis function (RBF) neural network into a connection structure. Considering both computational efficiency and ease of adaptation, the zero-order Takagi-Sugeno fuzzy rule[17] is chosen,

$$R^l\text{: IF } u(k) \text{ is } F^l\text{, THEN } \hat{v}(k) = W_l,\ l = 1, 2, ..., N \tag{10}$$

where R^l $(l = 1, 2, ..., N)$ denotes the lth fuzzy rule, N is the total number of fuzzy IF−THEN rules, $u(k)$ is the input variable of the system, $\hat{v}(k)$ is the output variable of the lth fuzzy rule, F^l is the fuzzy set defined on the corresponding universe, and W_l is the lth consequence of the fuzzy rule.

The neuro-fuzzy system consists of four layers as shown in Figure 3:

Layer 1: This layer is the input layer, whose nodes just transmit the input variable $u(k)$ to the next layer directly.

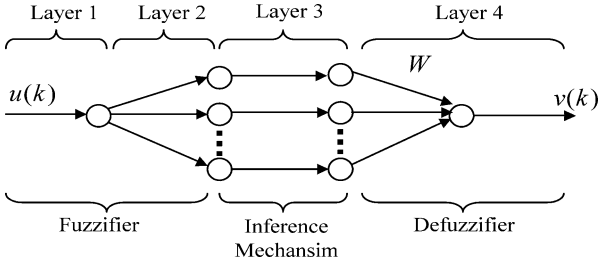

Figure 3. The structure of the neuro-fuzzy system.

Layer 2: This layer is the membership function layer that receives the signals from the input layer and calculates the membership of the input variable. The membership function chosen in this paper is the Gaussian membership function as described by

$$\mu_l = \exp\left(-\frac{(u(k) - c_l)^2}{\sigma_l^2}\right), \quad l = 1, 2, ..., N \tag{11}$$

where c_l and σ_l are center and width, respectively.

Layer 3: This layer is the rule layer. The number of the nodes in this layer represents the number of fuzzy rules. Since there is one input variable $u(k)$, it computes the fired strength of a rule as μ_l.

Layer 4: This layer is the output layer. All consequence weights are fully connected to the output node in which defuzzification is performed. The output of the whole neuro-fuzzy system is then given by

$$\hat{v}(k) = \sum_{l=1}^{N} \phi_l(u(k)) W_l \tag{12}$$

where

$$\phi_l(u(k)) = \frac{\mu_l}{\sum_{l=1}^{N} \mu_l}, \quad l = 1, 2, ..., N \tag{13}$$

To simplify the notation, $\phi_l(u(k))$ will be denoted as $\phi_l(k)$ in the following development. Because the neuro-fuzzy system has universal approximation capabilities,[15] the class of the Hammerstein process considered in this paper is more general than that based on the polynomial approach.

Last, the output of the neuro-fuzzy-based Hammerstein model, eq 8, can be represented as

$$\hat{y}(k+1) = \hat{a}^{\mathrm{T}} \mathscr{Y}(k) + \hat{b}^{\mathrm{T}} \Phi^{\mathrm{T}}(k) W \tag{14}$$

where

$$W = [W_1, W_2, ..., W_N]^{\mathrm{T}} \tag{15}$$

$$\Phi(k) = \begin{bmatrix} \phi_1(k) & \phi_1(k-1) & ... & \phi_1(k+1-n_b) \\ \phi_2(k) & \phi_2(k-1) & ... & \phi_2(k+1-n_b) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_N(k) & \phi_N(k-1) & ... & \phi_N(k+1-n_b) \end{bmatrix}_{N \times n_b} \tag{16}$$

3. Identification of Neuro-Fuzzy-Based Hammerstein Model

This section presents the two-step identification procedure for the proposed neuro-fuzzy-based Hammerstein model in detail.

Step I: Identification of the Linear Dynamics $H(z)$. As discussed previously, the identification of the linear dynamic part can be completely separated from that of the nonlinear static part by using the binary signal.[11] Thus, the identification of the linear dynamic part is simplified to an identification problem based on the binary input signal and the corresponding output of the Hammerstein process. For example, the least squares method can be adopted to estimate the parameters of the linear dynamics:

$$\hat{\theta} = [\Psi \Psi^{\mathrm{T}}]^{-1} \Psi Y \tag{17}$$

where

$$\Psi = [\psi(1), \psi(2), ..., \psi(N_{P_L})]_{(n_a+n_b)\times N_{pL}}$$

$$Y = [y(1), y(2), ..., y(N_{P_L})]^{\mathrm{T}}_{N_{pL}\times 1}$$

where N_{P_L} is the number of binary signals.

Step II: Identification of the Nonlinear Static $N(\cdot)$. Based on the entire training data, the procedure of identifying the neuro-fuzzy-based model is to estimate the parameters of c_l, σ_l, and W_l. This is a nonlinear optimization problem. In this paper, a clustering algorithm is presented in order to estimate the antecedent parameters, c_l and σ_l, of the neuro-fuzzy-based model. In addition, an updating algorithm for the consequent parameter, W_l, of the model is developed based on the Lyapunov stability theory and thus it guarantees the global convergence of W_l in an iterative fashion. The proposed method can be summarized as follows:

(a) Collect the input data $u(k)$ ($k = 1, 2, ..., N_P$). The input data point $u(1)$ is set as the first cluster (fuzzy rule) and its cluster center is $c_1 = u(1)$. The number of input data points belonging to the first cluster, $\bar{N}_1$, and the number of fuzzy clusters, $\bar{N}$, at this time are thus respectively given by $\bar{N}_1 = 1$ and $\bar{N} = 1$. (b) For the kth training data point $u(k)$, compare the similarity of the kth training data point to every cluster center c_l ($l = 1, 2, ..., \bar{N}$) and find a cluster denoted by L (fuzzy rule) whose center is closest to $u(k)$ as determined by the following criteria:[18]

$$S_L = \max_{1 \le l \le \bar{N}} (\sqrt{e^{-||u(k)-c_l||^2}}) \tag{18}$$

(c) Determine whether a new cluster (fuzzy rule) should be added or not according to the following criteria:

- If $S_L < S_0$, where S_0 is a prespecified threshold, then the kth training data point does not belong to all the existing clusters and a new cluster will be added with its center located at $c_{\bar{N}+1} = u(k)$. Set $\bar{N} = \bar{N} + 1$ and $\bar{N}_{\bar{N}} = 1$, and keep other clusters unmodified.
- If $S_L \ge S_0$, the kth training data point belongs to the Lth cluster. Set $\bar{N}_L = \bar{N}_L + 1$ and the Lth cluster is updated as follows:

$$c_L = c_L + \frac{\lambda}{\bar{N}_L + 1}(u(k) - c_L), \quad \lambda \in [0,1] \tag{19}$$

(d) Set $k = k + 1$ and repeat steps (b) and (c) until all training data points are clustered into the corresponding cluster. At the completion of the first three steps, the number of clusters (fuzzy rule) is fixed as $\bar{N}$, and the width of the fuzzy set can be calculated as

$$\sigma_l = \min_{j=1,2,\ldots,\bar{N};j=l} \frac{|c_l - c_j|}{\rho} \tag{20}$$

where ρ is the overlap parameter ($1 \le \rho \le 2$).[19]

(e) After the parameters $\hat{a}$, $\hat{b}$, c_l, and σ_l have been identified in the aforementioned steps, we now consider the problem of determining the consequence vector W. Before providing the proof of the convergence of the weight updating algorithm, we assume that there exists an ideal constant vector W^* such that

$$y(k+1) = \hat{a}^{\mathrm{T}}\mathcal{Y}(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)W^* + d(k), \quad \forall\, k \in \Omega_k \tag{21}$$

where $d(k)$ is the approximation error that represents the minimum possible deviation of the ideal approximator $\hat{a}^{\mathrm{T}}\mathcal{Y}(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)W^*$ from the output of process $y(k+1)$. The approximation error depends on the choice of the number of the fuzzy rules determined by steps (a)−(d). Universal approximation theorem[15] indicates that if the number of fuzzy rules $\bar{N}$ is sufficiently large, then the magnitude of $d(k)$ can be made as small as possible. In practice, although more fuzzy rules could give smaller prediction error for training data, it is generally achieved at the expense of poor generalization capability against the validation data. Thus, the prediction errors of both training data and validation data should be considered in choosing the number of fuzzy rules.

Denote $\hat{W}(k)$ and $e(k) = \hat{y}(k) - y(k)$ to be the estimated weight vector and the prediction error at the sampling instant k, the following auxiliary error $e_1(k)$[20] is introduced:

$$e_1(k) = \frac{\alpha e(k) + \hat{a}^{\mathrm{T}}\mathcal{Y}(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\hat{W}(k) - y(k+1)}{1 + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}} \tag{22}$$

where α is a constant subject to a constraint given in Theorem 1.

From eqs 21 and 22, one obtains

$$e_1(k) = \frac{\alpha e(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\tilde{W}(k) - d\,(k)}{1 + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}} \tag{23}$$

where

$$\tilde{W}(k) = \hat{W}(k) - W^* \tag{24}$$

Using $e_1(k)$ and $e(k)$, the estimated output $\hat{y}(k+1)$ can be predicted by the following equation:[20]

$$\hat{y}(k+1) = \beta e(k) + \hat{a}^{\mathrm{T}}\mathcal{Y}(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\hat{W}(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1(k) \tag{25}$$

where β is a constant subject to a constraint given in Theorem 1. In eq 25, the first and last terms are relative to the error $e(k)$ and the auxiliary error $e_1(k)$, respectively, which are used to compensate the modeling error. The remaining term $\hat{a}^{\mathrm{T}}\mathcal{Y}(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\hat{W}(k)$ is the output of the neuro-fuzzy-based model.

Subtracting $y(k+1)$ from both sides of eq 25, we have

$$e(k+1) = \beta e(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\,\tilde{W}(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1(k) - d(k) \tag{26}$$

Next, denote $\Delta(k)$ as follows:

$$\Delta(k) = \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\tilde{W}(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1(k) \tag{27}$$

From eqs 23 and 26, $e(k+1)$ and $e_1(k)$ can be, respectively, expressed by

$$e(k+1) = \beta e(k) + \Delta(k) - d(k) = \beta e(k) + \Delta_1(k) \tag{28}$$

$$e_1(k) = \alpha e(k) + \Delta(k) - d(k) = \alpha e(k) + \Delta_1(k) \tag{29}$$

where $\Delta_1(k) = \Delta(k) - d(k)$. Note that eqs 28 and 29 can be viewed as a linear dynamic system, denoted as the error model used in the stability analysis of adaptive systems.[21] Since $\hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)$ is bounded, the objective is to adjust $\tilde{W}(k)$ to let e_1 asymptotically converge to a compact set, and thus e and $\tilde{W}$ are bounded. The updating algorithm is proposed as follows:

$$\Delta W(k) = -\Phi(k)\,\hat{b}e_1(k) \tag{30}$$

$$\hat{W}(k+1) = \hat{W}(k) + \Delta W(k) \tag{31}$$

Theorem 1. *Considering the linear dynamic system in eqs 28 and 29 with the adaptive law eq 30 and the approximation error $d(k)$ is constrained by $|d(k)| \le d_0$, if the Lyapunov function candidate is chosen as*

$$V(k+1) = \gamma e^2(k+1) + \tilde{W}^{\mathrm{T}}(k+1)\tilde{W}(k+1) \tag{32}$$

where $\gamma < 1$ is a positive constant, and the parameters α and β in eqs 22 and 25 satisfy $(\alpha^2/\gamma) + (\beta^2/(1-\gamma)) \le 1$, then $\Delta V(k+1) < 0$ holds as long as (e, e_1) is outside the compact set Θ_v. Consequently, the error e and the auxiliary error e_1 asymptotically converge to the compact sets Θ_e and Θ_{e_1}, respectively, and $\tilde{W}$ is bounded.

$$\Theta_v = \{(e, e_1)|(\alpha^2 + 2\gamma_3)e^2(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}{e_1}^2(k) \le {d_0}^2\} \tag{33}$$

$$\Theta_e = \left\{e(k)\middle|\left|e(k)\right| \le \frac{d_0}{\sqrt{\alpha^2 + 2\gamma_3}}\right\} \tag{34}$$

$$\Theta_{e_1} = \left\{e_1(k)\middle|\left|e_1(k)\right| \le \frac{d_0}{\sqrt{\hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}}}\right\} \tag{35}$$

$$\gamma_1 = \sqrt{\frac{1-\gamma}{2}}$$

$$\gamma_2 = \frac{\beta\gamma}{2\gamma_1}$$

$$\gamma_3 = \frac{(1-\beta^2)\gamma}{2} - \alpha^2 - {\gamma_2}^2 \tag{36}$$

Proof: Considering eqs 28 and 32, we have

$$\Delta V(k+1) = V(k+1) - V(k) = \gamma(\beta^2 - 1)e^2(k) + 2\beta\gamma e(k)\Delta_1(k) + \gamma{\Delta_1}^2(k) + \tilde{W}^{\mathrm{T}}(k+1)\tilde{W}(K+1) - \tilde{W}^{\mathrm{T}}(k)\tilde{W}(k) \tag{37}$$

By using the relation $\tilde{W}(k+1) = \hat{W}(k+1) - W^* = \tilde{W}(k) + \Delta W(k)$, eq 37 is rewritten as

$$\Delta V(k+1) = \gamma(\beta^2 - 1)e^2(k) + 2\beta\gamma e(k)\Delta_1(k) + \gamma\Delta_1{}^2(k) + 2\tilde{W}^{\mathrm{T}}(k)\Delta W(k) + \Delta W^{\mathrm{T}}(k)\Delta W(k)$$
$$= -2(\alpha^2 + \gamma_2{}^2 + \gamma_3)e^2(k) + 4\gamma_1\gamma_2 e(k)\Delta_1(k) + (1 - 2\gamma_1{}^2)\Delta_1{}^2(k) + 2\tilde{W}^{\mathrm{T}}(k)\Delta W(k) + \Delta W^{\mathrm{T}}(k)\Delta W(k) \quad (38)$$

Based on eq 29, eq 38 is rearranged as

$$\Delta V(k+1) = -2(\gamma_2 e(k) - \gamma_1\Delta_1(k))^2 - 2(\alpha^2 + \gamma_3)e^2(k) + \Delta_1{}^2(k) + 2\tilde{W}^{\mathrm{T}}(k)\Delta W(k) + \Delta W^{\mathrm{T}}(k)\Delta W(k)$$
$$= -2(\gamma_2 e(k) - \gamma_1\Delta_1(k))^2 - 2(\alpha^2 + \gamma_3)e^2(k) + 2\Delta_1(k)e_1(k) - 2\alpha e(k)\Delta_1(k) - \Delta_1{}^2(k) + 2\tilde{W}^{\mathrm{T}}(k)\Delta W(k) + \Delta W^{\mathrm{T}}(k)\Delta W(k) \quad (39)$$

According to eqs 27, 29, and 30, we obtain

$$\Delta V(k+1) = -2(\gamma_2 e(k) - \gamma_1\Delta_1(k))^2 - 2(\alpha^2 + \gamma_3)e^2(k) - 2\alpha e(k)\Delta_1(k) - \Delta_1{}^2(k) + 2e_1(k)(\hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\tilde{W}(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1(k) - d(k)) - 2\tilde{W}^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1(k) + \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1{}^2(k)$$
$$= -2(\gamma_2 e(k) - \gamma_1\Delta_1(k))^2 - (\alpha^2 + 2\gamma_3)e^2(k) - e_1{}^2(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1{}^2(k) - 2e_1(k)d(k) \quad (40)$$

Noting that $-2e_1(k)d(k) \le e_1{}^2(k) + d_0{}^2$, eq 40 further reduces to

$$\Delta V(k+1) \le -2(\gamma_2 e(k) - \gamma_1\Delta_1(k))^2 - (\alpha^2 + 2\gamma_3)e^2(k) - e_1{}^2(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1{}^2(k) + e_1{}^2(k) + d_0{}^2$$
$$\le -(\alpha^2 + 2\gamma_3)e^2(k) - \hat{b}^{\mathrm{T}}\Phi^{\mathrm{T}}(k)\Phi(k)\hat{b}e_1{}^2(k) + d_0{}^2 \quad (41)$$

Note that Θ_v and Θ_e are compact sets because of $\alpha^2 + 2\gamma_3 > 0$, resulting from the constraint $(\alpha^2/\gamma) + (\beta^2/(1 - \gamma)) \le 1$. It is also clear that Θ_{e_1} is a compact set. Equation 41 shows that $\Delta V(k+1) < 0$ as long as (e, e_1) is outside the compact set Θ_v. According to the Lyapunov theorem,[22] the close-loop dynamic system represented by eqs 28 and 29 is stable, indicating that e and e_1 will asymptotically converge to the compact sets, Θ_e and Θ_{e_1}, respectively, which implies the boundness of $\tilde{W}$. This completes the proof. Q.E.D.

Equations 34 and 35 reveal that the worst approximation error d_0 is proportional to the radii of compact sets Θ_e and Θ_{e_1}. As discussed previously, d_0 can be made as small as possible if fuzzy rules $\bar{N}$ is sufficiently large, leading to very small e and e_1. This point will be illustrated in the simulation results in the next section. Furthermore, compared with the existing iterative algorithms, such as gradient descent algorithm,[18] the novel introduction of Lyapunov synthesis in determining the consequent parameter of the neuro-fuzzy-based Hammerstein model not only guarantees the convergence of the weight parameters but also avoids the problems of initialization, which is usually resorted to trial and error procedure in the existing iterative algorithms. Starting with any initial values of $\hat{W}(0)$ and $e(0)$, the updating law, eq 31, will always guarantee that $e(k)$ asymptotically converges to the compact set, Θ_e.

In summary, the proposed neuro-fuzzy-based Hammerstein model is constructed by the following steps:

Step 1. The parameters of the linear model, $\hat{a}$ and $\hat{b}$, are estimated by using the process data excited by the binary input signal.

Step 2. Set S_0 and ρ, and determine the parameters c_l and σ_l by using the clustering algorithm discussed in (a)−(d).

Step 3. Set α and β and initialize $\hat{W}(0) = 0$ and $e(0) = 0$, and use the entire training data to update $\hat{W}$.

Step 4. For the kth training data, compute $\hat{y}(k)$ by eq 25 and $e(k) = \hat{y}(k) - y(k)$, and update $\hat{W}(k)$ by eqs 30 and 31.

Step 5. Set $k = k + 1$ and repeat step 4 and step 5 until $|e(k)|$ is smaller than a specified error tolerance.

Step 6. Stop.

4. Examples

Example 1: Considering the following Hammerstein process whose static nonlinearity is a discontinuous function:

$$v(k) = \begin{cases} \tanh(2u(k)), & u(k) \le 1.5 \\ -\dfrac{\exp(u(k)) - 1}{\exp(u(k)) + 1} & u(k) > 1.5 \end{cases} \quad (42)$$

$$y(k) = 0.8y(k-1) + 0.4v(k-1) \quad (43)$$

To identify the proposed neuro-fuzzy-based Hammerstein model, the special test signal consisting of binary signal and independent random multistep signal with uniform distribution between [0, 4] and corresponding process output are simulated as shown in Figure 4. The linear dynamic model is obtained from the first 200 binary input signal and corresponding output data:

$$\hat{y}(k) = 0.8y(k-1) + 0.3856\,\hat{v}(k-1) \quad (44)$$

Next, the nonlinear static function is estimated by using the following design parameters: $S_0 = 0.995$, $\rho = 1.5$, $\lambda = 0.01$, $\alpha = -0.1$, and $\beta = 0.1$. This results in a neuro-fuzzy-based model with 32 rules and the MSE of the predictive error is 2.7×10^{-4}. To illustrate the prediction capability of the proposed model, another 1000 input−output data are generated for validation purpose. The validation results are shown in Figure 5 and the corresponding MSE $= 1.5 \times 10^{-4}$.

To check the accuracy of the neuro-fuzzy-based model, the nonlinear process of eqs 42 and 43 is rewritten as follows:

$$v(k) = \begin{cases} 1.037\tanh(2u(k)), & u(k) \le 1.5 \\ \dfrac{-1.037(\exp(u(k)) - 1)}{\exp(u(k)) + 1} & u(k) > 1.5 \end{cases} \quad (45)$$

$$y(k) = 0.8y(k-1) + \frac{0.4}{1.037}v(k-1) \quad (46)$$

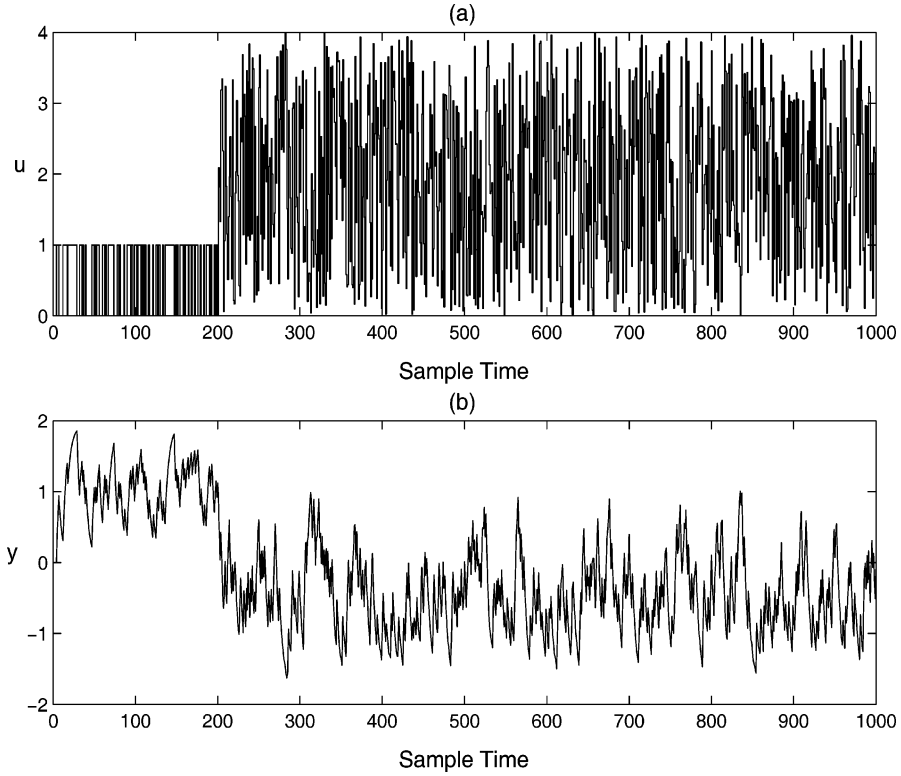

Figure 4. The training data used in example 1.

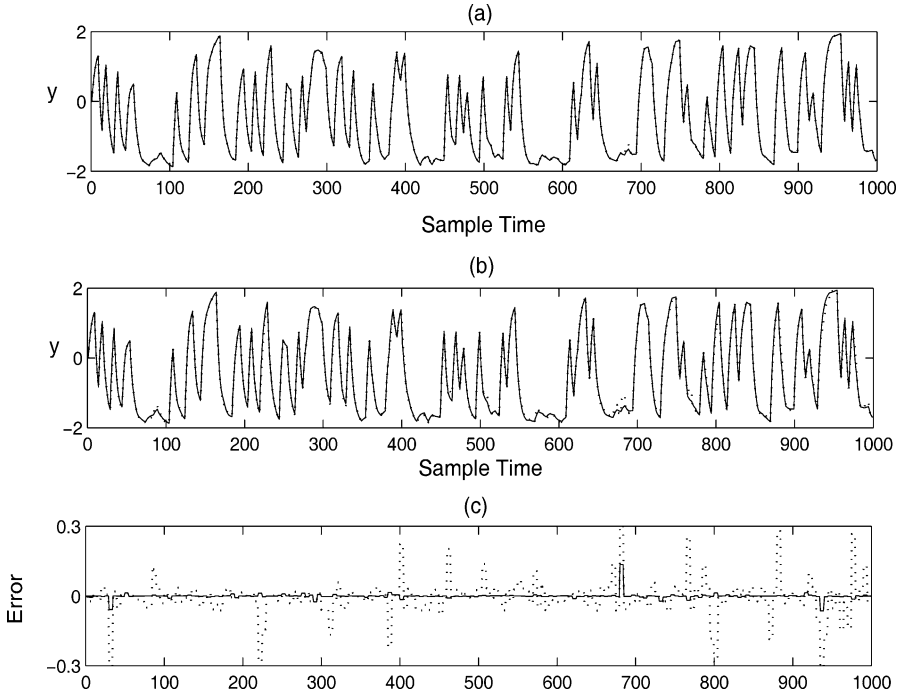

Figure 5. Validation result. (a) Process output (solid line) and predicted output of the proposed method (dotted line); (b) process output (solid line) and predicted output of the polynomial-based Hammerstein model (dotted line); (c) proposed method (solid line) and polynomial-based Hammerstein model (dotted line).

The nonlinear static function of eq 45 and that predicted by the neuro-fuzzy-based model are compared in Figure 6. It is clear that the neuro-fuzzy-based model has very accurate approximation of the actual nonlinearity. For comparison purposes, the polynomial-based Hammerstein model is also constructed by using the identical training data and the linear model given in eq 44. The MSEs of different polynomial models are summarized in Table 1 and the model of the order of 10 is chosen to be the best model. The validation result of this chosen model and its predicted static nonlinearity are given in Figures 5 and 6, respectively. It is obvious that the predictive performance of the neuro-fuzzy-based Hammerstein model is superior to that of the polynomial-based Hammerstein model.

To further evaluate the effectiveness of the proposed Lyapunov-based updating algorithm, the gradient descent algorithm is employed to update the weights of the neuro-fuzzy-based model. Table 2 summarizes MSEs for both training and validation results obtained by two

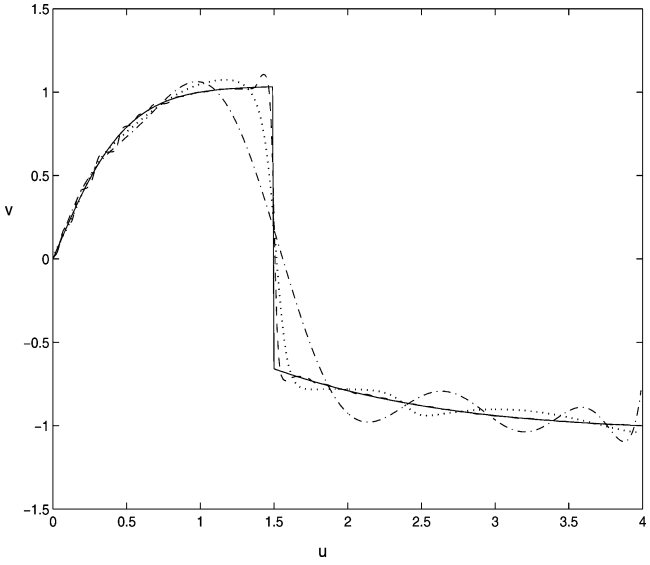

Figure 6. Nonlinear static function. Solid line: actual process. Dashed line: proposed method. Dash−dot line: polynomial approach. Dotted line: gradient descent algorithm.

Table 1. MSEs of Various Polynomial-Based Hammerstein Models

order	MSE(training)	MSE(validation)
5	9.0×10^{-3}	1.2×10^{-2}
6	6.3×10^{-3}	8.9×10^{-3}
7	7.3×10^{-3}	6.3×10^{-3}
8	5.1×10^{-3}	7.3×10^{-3}
9	4.9×10^{-3}	6.9×10^{-3}
10	4.4×10^{-3}	6.3×10^{-3}
11	5.1×10^{-3}	6.4×10^{-3}

Table 2. MSEs of the Proposed Method and Gradient Descent Algorithm

model	training data	validation data
proposed method	2.7×10^{-4}	1.5×10^{-4}
gradient descent algorithm	1.6×10^{-3}	1.7×10^{-3}

methods, respectively. The predicted nonlinear static function by the gradient descent algorithm is shown in Figure 6. Figure 7 compares the convergence of the two algorithms. It can be seen that the proposed method not only gives better prediction accuracy but also has faster

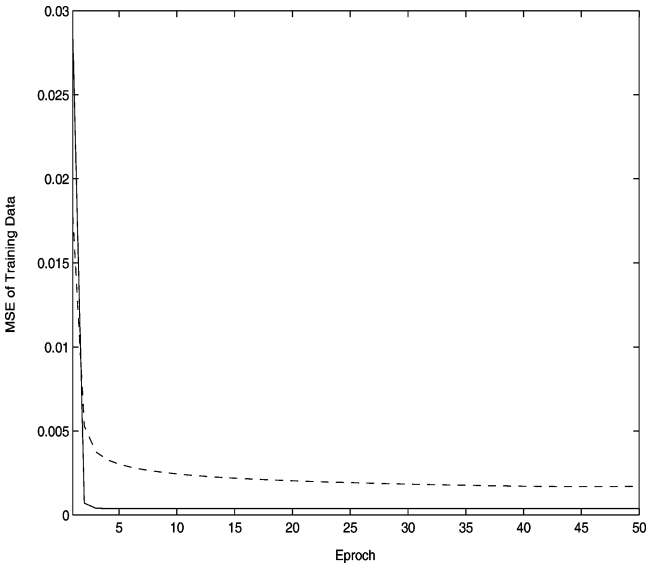

Figure 7. Convergence of the proposed method (solid line) and gradient descent algorithm (dashed line).

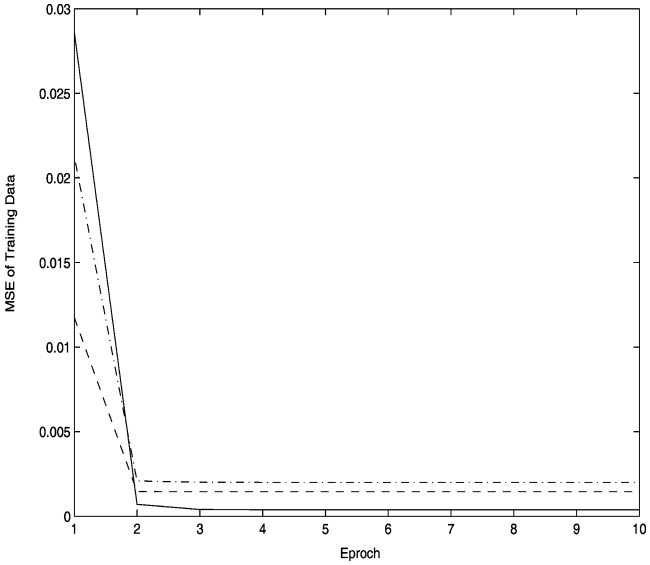

Figure 8. Convergence of different neuro-fuzzy-based models. Solid line: $S_0 = 0.995$, $\bar{N} = 32$. Dashed line: $S_0 = 0.99$, $\bar{N} = 21$. Dash−dot line: $S_0 = 0.95$, $\bar{N} = 10$.

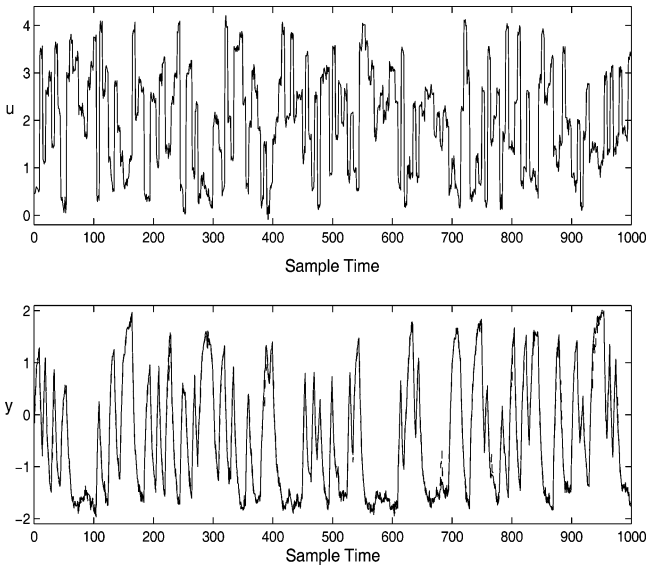

Figure 9. Validation result (with noisy process data). Solid line: process. Dashed line: model.

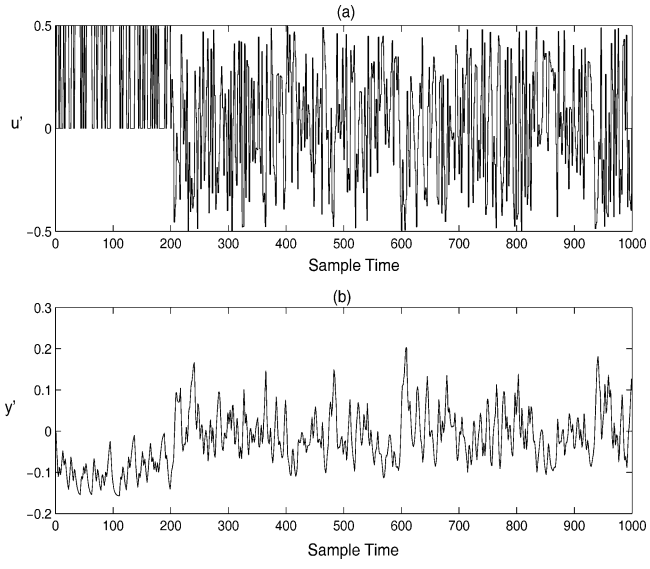

Figure 10. The training data used in example 2.

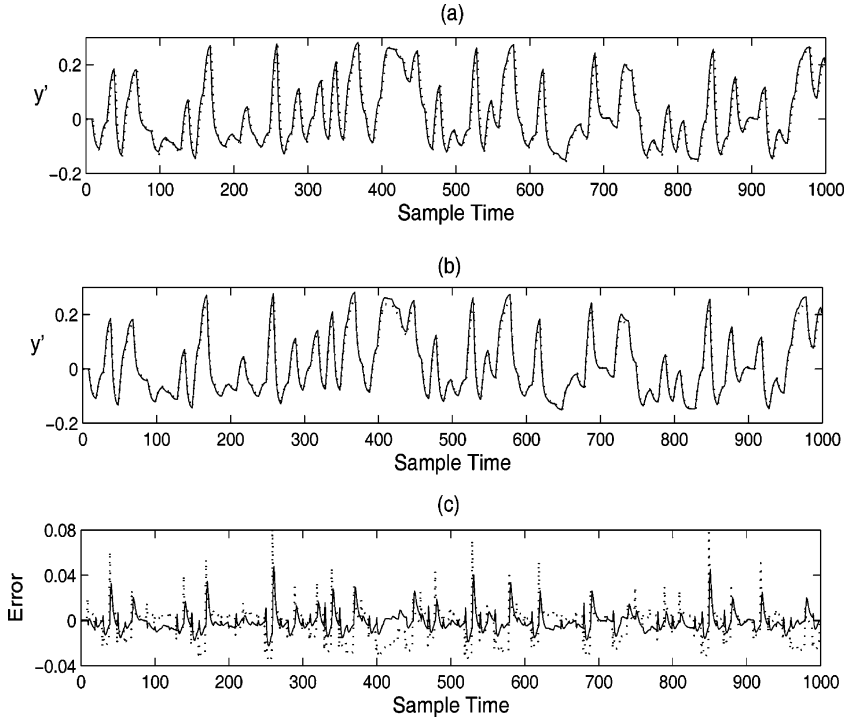

Figure 11. Validation result. (a) Process output (solid line) and predicted output of the proposed method (dotted line); (b) process output (solid line) and predicted output of Narendra's algorithm (dotted line); (c) proposed method (solid line); Narendra's algorithm (dotted line).

Table 3. MSEs of the Proposed Method and Narendra's Algorithm

model	training data	validation data
proposed method	7.1633×10^{-5}	7.2109×10^{-5}
Narendra's algorithm	2.2023×10^{-4}	2.3090×10^{-4}

convergence rate for training data. Last, Figure 8 illustrates that the predictive performance of the proposed neuro-fuzzy-based Hammerstein model is indeed improved by increasing the number of fuzzy rule $\bar{N}$.

To test the robustness of the proposed method, both process output and input variables are corrupted by 2% Guassian white noise. As seen from Figure 9, the proposed method maintains good prediction accuracy despite that both training and validation data contain the noisy signals.

Example 2: Consider a polymerization reactor where free-radical polymerization of methyl methacrylate occurs with azo-bis-isobutyronitrile as initiator and toluene as solvent. This process was modeled by a Hammerstein model[6] and its dynamics can be described as follows:[6,23]

$$\dot{x}_1 = 10(6 - x_1) - 2.4568x_1\sqrt{x_2}$$

$$\dot{x}_2 = 80u - 10.1022x_2$$

$$\dot{x}_3 = 0.002412x_1\sqrt{x_2} + 0.112191x_2 - 10x_3 \quad (47)$$

$$\dot{x}_4 = 245.978x_1\sqrt{x_2} - 10x_4$$

$$y = x_4/x_3$$

where u is the flow rate of the initiator and y is the product number average molecular weight. The steady-state parameters of the polymerization reactor are as follows: $u_0 = 0.016783$ m^3 h^{-1}, $y_0 = 25000.5$, $x_{10} = 5.50677$, $x_{20} = 0.132906$, $x_{30} = 0.0019752$, $x_{40} = 49.3818$, and the sampling time is 0.05 h.

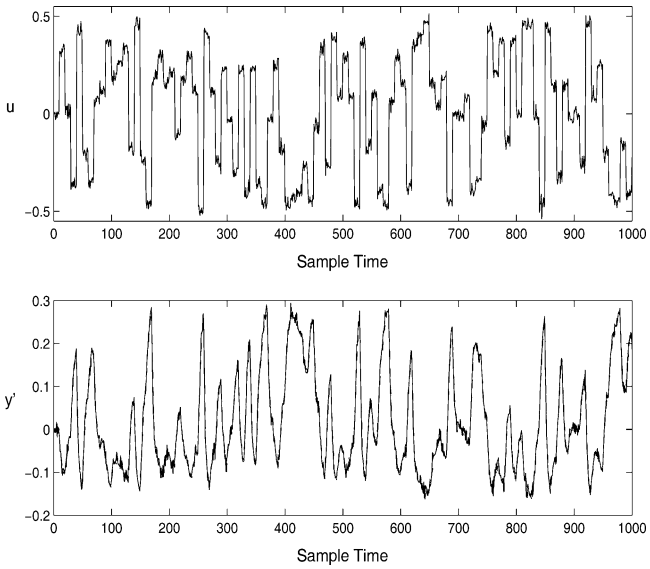

Figure 12. Validation result (with noisy process data). Solid line: process. Dashed line: model.

Both input and output variables are first normalized using $u' = (u - u_0)/u_0$ and $y' = (y - y_0)/y_0$. The binary signal and independent random signal with uniform distribution between [−0.5, 0.5] are used to simulate 1000 input−output data for training purposes, as shown in Figure 10. To proceed with the proposed identification algorithm, set the design parameters as follows: $S_0 = 0.99$, $\rho = 2$, $\lambda = 0.01$, $\alpha = -0.1$, and $\beta = 0.1$, which leads to a neuro-fuzzy-based Hammerstein model with eight fuzzy rules and the linear dynamic model is $y'(k+1) = 0.8594\,y'(k) - 0.0447v(k)$. For comparison purposes, the following polynominal Hammerstein model is identified by using the Narendra's iterative algorithm:[8]

$$v(k) = u'(k) - 0.1775u'^2(k) + 0.1817u'^3(k) \quad (48)$$

$$y'(k+1) = 0.7882y'(k) - 0.0619v(k) \quad (49)$$

The comparison between the proposed algorithm and Narendra's algorithm is shown in Table 3 and Figure 11. It is evident that the proposed algorithm yields a more accurate Hammerstein model than that obtained by Narendra's algorithm.

The robustness of the proposed method is evaluated by introducing 2% Gaussian white noise to the measured process variables. As illustrated in Figure 12, the proposed method has reasonable robustness to process noise.

5. Conclusion

In this paper, a neuro-fuzzy-based model has been used to describe the nonlinearity of the Hammerstein process without any prior process knowledge, thus avoiding the inevitable restrictions on static nonlinear function encountered by using the polynominal approach. A clustering algorithm is presented to identify the antecedent parameters of the neuro-fuzzy-based Hammerstein model. Furthermore, an iterative identification procedure is developed to identify and guarantee the global convergence of the consequent parameters of the model. Examples illustrate the proposed method yields better predictive performance than its counterparts.

Literature Cited

(1) Lakshminarayanan, S.; Shah, S. L.; Nandakumar, K. Identification of hammerstein models using multivariate statistical tools. *Chem. Eng. Sci.* **1995**, *50*, 3599−3613.

(2) Fruzzetti, K. P.; Palazoglu, A.; McDonald, K. A. Nonlinear model predictive control using hammerstein models. *J. Process Control* **1997**, *7*, 31−41.

(3) Eskinat, E.; Johnson, S. H.; Luyben, W. L. Use of hammerstein models in identification of nonlinear systems. *AIChE J.* **1991**, *37*, 255−268.

(4) Pearson, R. K.; Pottmann, M. Gray-box identification of block-oriented nonlinear models. *J. Process Control* **2000**, *10*, 301−315.

(5) Su, H. T.; McAvoy, T. J. Integration of multilayer percepton networks and linear dynamic models: a hammerstein modelling approach. *Ind. Eng. Chem. Res.* **1993**, *26*, 1927−1936.

(6) Ling, W. M.; Rivera, D. E. Control relevant model reduction of volterra series models. *J. Process Control* **1998**, *8*, 79−88.

(7) Rollins, D. K.; Bhandari, N.; Bassily, A. M.; Colver, G. M.; Chin, S. T. A continuous-time nonlinear dynamic predictive modeling method for hammerstein processes. *Ind. Eng. Chem. Res.* **2003**, *42*, 860−872.

(8) Narendra, K. S.; Gallman, P. G. An iterative method for the identification of nonlinear systems using the hammerstein model. *IEEE Trans. Automat. Control* **1966**, *11*, 546−550.

(9) Chang, F. H. I.; Luus, R. A noniterative method for identification using hammerstein model. *IEEE Trans. Automat. Control* **1971**, *16*, 464−468.

(10) Gallman, P. G. A comparison of two hammerstein model identification algorithms. *IEEE Trans. Automat. Control* **1976**, *21*, 124−126.

(11) Sung, S. W. System identification method for hammerstein processes. *Ind. Eng. Chem. Res.* **2002**, *41*, 4295−4302.

(12) Lang, Z. Q. On identification of the controlled plants described by the hammerstein systems. *IEEE Trans. Automat. Control* **1994**, *39*, 569−573.

(13) Duwaish, H. A.; Karim, M. N. A new method for the identification of hammerstein model. *Automatica* **1997**, *33*, 1871−1975.

(14) Zhu, Y. C. Identification of hammerstein models for control using ASYM. *Int. J. Control* **2000**, *73*, 1692−1702.

(15) Wang, L. X. Fuzzy systems are universal approximators. *Proc. IEEE Int. Conf. Fuzzy Syst.* **1992**, 1163−1170.

(16) Abonyi, J.; Babuska, R.; Botto, M. A.; Szeifert, F.; Nagy, L. Identification and control of nonlinear systems using fuzzy hammerstein models. *Ind. Eng. Chem. Res.* **2000**, *39*, 4302−4314.

(17) Takagi, T.; Sugeno, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man, Cybern* **1985**, *SMC-15*, 116−132.

(18) Wang, L. X. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*; Prentice Hall: Englewood Cliffs, NJ, 1994.

(19) Lin, C. T.; Lee, C. S. G. Neural networks based fuzzy logic control and decision system. *IEEE Trans. Comput.* **1991**, *40*, 1320−1336.

(20) Tan, S. H.; Hao, J. B.; Vandewalle, J. Efficient identification of RBF neural net models for nonlinear discrete-time multivariable dynamical systems. *Neurocomputing* **1995**, *9*, 11−26.

(21) Lin, Y. H.; Narendra, K. S. A new error model for adaptive systems. *IEEE Trans. Automat. Control* **1980**, *AC-25*, 585−587.

(22) Narendra, K. S.; Yin, Y. H.; Valavani, L. S. Stable adaptive controller design-part II: proof of stability. *IEEE Trans. Automat. Control* **1980**, *AC-25*, 440−448.

(23) Doyle, F. J.; Ogunnaike, B. A.; Pearson, R. K. Nonlinear model predictive control using second-order volterra models. *Automatica* **1995**, *31*, 697−703.

Received for review May 24, 2004
Revised manuscript received December 1, 2004
Accepted January 3, 2005

IE0495574