International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:2, No:6, 2008

Modeling of Crude Oil Blending via Discrete-Time Neural Networks

Xiaoou Li, and Wen Yu

***Abstract*—** Crude oil blending is an important unit operation in petroleum refining industry. A good model for the blending system is beneficial for supervision operation, prediction of the export petroleum quality and realizing model-based optimal control. Since the blending cannot follow the ideal mixing rule in practice, we propose a static neural network to approximate the blending properties. By the dead-zone approach, we propose a new robust learning algorithm and give theoretical analysis. Real data of crude oil blending is applied to illustrate the neuro modeling approach.

***Keywords*—**Neural networks, modeling, stability, crude oil.

I. Introduction

CRUDE oil blending is an attractive solution for those refiners who have the ability to blend different crude types to provide a consistent and optimal feedstock to refinery operations. Optimal crude purchasing is an effective method to improve refinery profits. In general the blending rule is nonlinear, it can be regarded as a linear mixing rule adding a nonlinear term. Crude oil blending is an optimization operations based upon real-time analyzers and process knowledge [6]. A mathematical model for crude oil blending is needed to address uncertainties in blending operation; real-time optimization (RTO) has been proposed [20]. The main drawback of RTO is that it cannot provide optimal set-points from large amounts of history data.

The exact mathematical model for crude oil blending is too complex to be handled analytically. Many attempts were made to introduce simplified models to construct model-based controller [9]. A common method to approximate the blending operation is to use linear (ideal) model [20] or to regard blending operation has a sufficient small nonlinear uncertainty [1].

Neuro modeling approach uses the nice features of neural networks, but the lack of mathematical model for the plant makes it hard to obtain theoretical results on stable learning. It is very important to assure the stability of neuro modeling in theory before we use them in some real applications. Lyapunov approach can be used directly to obtain robust training algorithms for continuous-time [23] [24] and discrete-time [13] [18] neural networks. It is well known that normal modeling algorithms are stable for ideal plants [11]. In the presence of disturbances or unmodeled dynamics, these adaptive procedures can go to instability easily. Generally, some modifications to the normal gradient algorithm or backpropagation should be applied, such that the learning process is stable. For example, in [13] some hard restrictions were added in the learning law, in [22] the dynamic backpropagation was modified with NLq stability constraints. Another generalized method is to use robust modification techniques of robust adaptive control [11]. [15] applied $\sigma-$ modification, [12] used modified $\delta-$ rule.

In this paper, we propose a novel learning algorithm for discrete-time feedforward neural network. By combining Lyapunov and dead-zone techniques, we analyze the stability of modeling error and the parameters. This learning law can guarantee both modelling error and weights bounded. The neuro modeling approach is successfully used to model crude oil blending via real data.

II. Crude Oil Blinding

Crude oils are often blended to increase the sale price or process-ability of a lower grade crude oil by blending it with a higher grade, higher price crude. The objective is to produce blended crude oil to a target specification at the lowest cost using the minimum higher cost crude oil. The crude oil feed-stocks used for blending often vary in quality and for this reason crude oil blenders normally use viscosity or density trim control systems. API (American Petroleum Institute) Gravity is the most used indication of density of crude oil. The lower the API Gravity, the heavier the compound. When the blender is started the required flow rate and component ratio is set by the control system based on the ratio in the recipe. A density or viscosity analyzer, installed at a homogeneous point in the blender header, generates a control signal, which is used to continually optimize the blended product by adjusting the component ratio. This ensures that the blended product remains as specified at all times during the batch. So normal modeling for crude oil blending is on-line. In this paper we will discuss an off-line modeling method.

We discuss a typical crude oil blending process in PEMEX (Mexican Petroleum Company), it is called Terminal Marítima de Dos Bocas Tabasco (TMDB). The flow-sheet is shown in Fig.1-(a). It has three blenders (M_1, M_2 and M_3), one dehydration equipment and one tank. We use Fig.2-(b) to

Manuscript received March 8, 2005. This work was supported in part by CONACyT under Grant 38505A

Xiaoou Li is with Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, Av.IPN 2508, México D.F., 07360, México (lixo@cs.cinvestav.mx).

Wen Yu is with the Departamento de Control Automatico, CINVESTAV-IPN, Av.IPN 2508, México D.F., 07360, México (yuw@ctrl.cinvestav.mx).

describe the static process of the crude oil blending, q_i is flow rate, p_i is the property of i th feed stock, it can be API Gravity. There are four feed-stocks, $L_3(q_1, p_1)$, Puerto Ceiba (q_2, p_2), Maya (q_3, p_3) and El Golpe (q_4, p_4). The blended product for national use (q_n, p_n) needs 2 blenders, the product for export (q_f, p_f) needs 3 blenders and is stored in a tank.

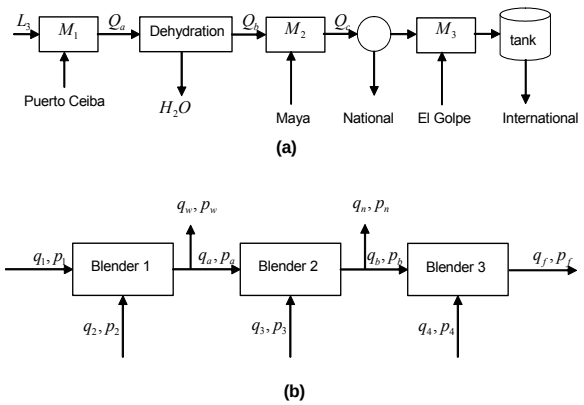

Fig.1 TMDB crude oil blending process

For each blender the static properties can be analyzed by thermodynamic. If the partial molar volume of a component in a solution is nearly the same as the molar volume of the pure component, the molar volume is simply the average of the molar volumes of the pure components at the same temperature and pressure. The thermodynamic property is ideal

$$p_a = \sum_{i=1}^{2} p_i x_i, \quad q_a = \sum_{i=1}^{2} q_i, \quad x_i = \frac{q_i}{q_a}$$

where x_i, q_i and p_i are the volume fraction, flow rate and API Gravity of i th feed-stock, p_a and q_a are the flow rate and the API Gravity of the blended product of Blender 1 (M_1) . Unfortunately, this equation is correct only in the ideal condition, in order to make it universally valid a correction term Δ is added

$$p_a = \sum_{i=1}^{2} p_i x_i + \Delta \tag{1}$$

where Δ is called the property change of mixing. Several approaches can approximate Δ, for examples

- Interaction model [20][1]

$$\Delta = \alpha x_1 x_2 \tag{2}$$

where α is the interaction coefficient between the two components

- Zahed model [27]

$$\Delta = \sum_{i=1}^{2} M_i (x_i p_i)^k \tag{3}$$

where M_i and k are constants.

All of above models are only suitable in some special conditions and the parameters of these models should be determined by experience data.

Since all of p_i and q_i in Fig.1-(b) are available, we can model each blander with input/output data, then connect them together, we call this method as *distribute model*. If the mixing rule is given by a interaction model as (2), the total blending is

$$\begin{aligned}
p_f &= \tfrac{1}{q_f}(p_4 q_4 + p_b q_b) + \alpha_3 x_4 x_b \\
&= \tfrac{1}{q_f} p_4 q_4 + \tfrac{1}{q_f}[p_3 x_3 + \left(\tfrac{q_a}{q_a}(p_2 q_2 + p_1 q_1) + \alpha_1 x_1 x_2\right) \\
&(1 - x_3) + \alpha_2 x_3 (1 - x_3)](1 - x_4) + \alpha_3 x_4 (1 - x_4) \\
&= \tfrac{1}{q_f}(p_1 q_1 + p_2 q_2 + p_3 q_3 + p_4 q_4) \\
&+ \tfrac{1}{q_f}[x_3 \alpha_2 - p_1 x_1 x_3 - p_1 x_1 x_4 \\
&- p_2 x_2 x_3 - p_2 x_2 x_4 - p_3 x_3 x_4 \\
&+ x_1 x_2 \alpha_1 - x_3 x_4 \alpha_2 + p_1 x_1 x_3 x_4 + p_2 x_2 x_3 x_4 \\
&- x_1 x_2 x_3 \alpha_1 - x_1 x_2 x_4 \alpha_1 + x_1 x_2 x_3 x_4 \alpha_1 - x_3^2 \alpha_2 \\
&+ x_3^2 x_4 \alpha_2 + x_4 \alpha_3 - x_4^2 \alpha_3]
\end{aligned} \tag{4}$$

where $q_f = q_1 + q_2 + q_3 + q_4 - q_w - q_n$, α_i is mixing rule coefficient for i -th blender.

We can also regard the it as multiple components blending process as in Fig., we call it as *integrated model.* The model can be expressed as

$$p_f = \sum_{i=1}^{4} p_i x_i + \Delta \tag{5}$$

If the mixing rule is given by a interaction model as (2)

$$\Delta = \sum_{i=1}^{4} \sum_{k=i+1}^{4} \alpha_{i,k} x_i x_k \tag{6}$$

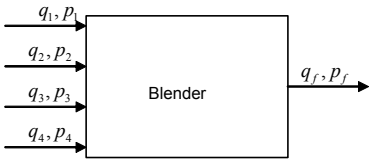

Fig.2 Integrated model

III. Modeling of Crude Oil Blending via Discrete-Time Neural Networks

The mathematical models discussed in Section 2 works only in some special conditions. In real application we have only input/output data, neural network can be applied to identify crude oil blending. Static neural networks can be used to identify the nonlinear parts Δ of the distribute model (1) or the integrated model (5) , it can also identify the whole blender (linear and nonlinear). This section will present a new stable learning algorithm for static neuro modeling.

The mixing property can be written in following form $p_f(k) = \Phi[u_1(k), \cdots, u_8(k)]$, or

$$y(k) = \Phi[X(k)] \tag{7}$$

where $X(k) = [u_1(k), \cdots, u_8(k)]^T$, $y(k)$ is the blended API Gravity value at time k $y(k) = p_f(k)$, $\Phi(\cdot)$ is an unknown nonlinear function representing the blending operation, $u_i(k)$ are measurable scalar inputs, they are API Gravity and flow rates, for example $u_1(k) = \frac{q_1}{q_o}$, $u_2(k) = p_1$, $u_7(k) = \frac{q_4}{q_o}$, $u_8(k) = p_4$. We consider multilayer neural network(or multilayer perceptrons) to model the blending properties as in (7)

$$\hat{y}(k) = V_k \phi[W_k X(k)] \tag{8}$$

where the scalar output $\hat{y}(k)$ and vector input $X(k) \in R^{n\times 1}$, the weights in output layer are $V_k \in R^{1\times m}$, the weights in hidden layer are $W_k \in R^{m\times n}$, ϕ is m dimension vector function. The typical presentation of the element $\phi_i(.)$ is sigmoid function. The identified blending system (7) can be represented as

$$y(k) = V^* \phi[W^* X(k)] - \mu(k)$$

where V^* and W^* are set of unknown weights which may minimize the modeling error $\mu(k)$. The nonlinear plant (7) can be also expressed as

$$y(k) = V^0 \phi[W^* X(k)] - \delta(k) \tag{9}$$

where V^0 is an known matrix chosen by users, in general, $\|\delta(k)\| \geq \|\mu(k)\|$. Using Taylor series around the point of $W_k X(k)$, the modeling error can be represented as

$$\begin{aligned} e(k) &= \hat{y}(k) - y(k) \\ &= V_k \phi[W_k X(k)] - V^0 \phi[W^* X(k)] + \delta(k) \\ &= V_k \phi[W_k X(k)] - V^0 \phi[W_k X(k)] \\ &+ V^0 \phi[W_k X(k)] - V^0 \phi[W^* X(k)] + \delta(k) \\ &= \widetilde{V}_k \phi[W_k X(k)] + V^0 \phi' \widetilde{W}_k X(k) + \zeta(k) \end{aligned} \tag{10}$$

where ϕ' is the derivative of nonlinear activation function $\phi(\cdot)$ at the point of $W_k X(k)$, $\widetilde{W}_k = W_k - W^*$, $\widetilde{V}_k = V_k - V^0$, $\zeta(k) = V^0 \varepsilon(k) + \delta(k)$, here $\varepsilon(k)$ is second order approximation error of the Taylor series.

In this paper we are only interested in open-loop modeling, we can assume that the plant (7) is bounded-input and bounded-output stable, *i.e.*, $y(k)$ and $u(k)$ in (7) are bounded. Since $X(k) = [u(k), u(k-1), u(k-2), \cdots]^T$, $X(k)$ is bounded. By the boundedness of the sigmoid function ϕ, we assume that $\delta(k)$ in (9) is bounded, also $\varepsilon(k)$ is bounded. So $\zeta(k)$ in (10) is bounded. The following theorem gives a new robust learning algorithm and stable analysis for the neural modeling.

Theorem 1: If we use the multilayer neural network (8) to model the crude oil blending (7), the following dead-zone backpropagation-like algorithm

$$\begin{aligned} W_{k+1} &= W_k - \eta_k e(k) \phi' V^{0T} X^T(k) \\ V_{k+1} &= V_k - \eta_k e(k) \phi^T \end{aligned} \tag{11}$$

where $\eta_k = \dfrac{s_k}{1 + \left\|\phi' V^{0T} X^T(k)\right\|^2 + \|\phi\|^2}$,

$$s_k = \begin{cases} \eta & e(k)^2 \geq \frac{\eta}{\pi}\overline{\zeta} \\ 0 & e(k)^2 < \frac{\eta}{\pi}\overline{\zeta} \end{cases}, \quad 1 \geq \eta > 0,$$

$$\pi = \frac{\eta}{(1+\kappa)^2} > 0, \quad \overline{\zeta} = \max_k [\zeta^2(k)],$$

$\kappa = \max_k \left(\left\|\phi' V^{0T} X^T(k)\right\|^2 + \|\phi\|^2 \right)$. This updating law can make the modelling error $e(k)$ and the weights of neural networks bounded

$$\|e(k)\| \in L_\infty, \quad W_k \in L_\infty, \quad V_k \in L_\infty \tag{12}$$

Also the average of the modelling error satisfies

$$J = \limsup_{T\to\infty} \frac{1}{T}\sum_{k=1}^{T} e^2(k) \le \frac{\eta}{\pi}\bar{\zeta} \tag{13}$$

Proof: If $e(k)^2 \ge \frac{\eta}{\pi}\bar{\zeta}$, the updating law is (11) with $\eta_k = \dfrac{\eta}{1+\left\|\phi' V^{0T} X^T(k)\right\|^2 + \|\phi\|^2}$. We selected a positive defined matrix L_k as

$$L_k = \left\|\tilde{W}_k\right\|^2 + \left\|\tilde{V}_k\right\|^2 \tag{14}$$

where $\tilde{W}_k = W_k - W^*$, $\tilde{V}_k = V_k - V^0$, $\left\|\tilde{W}_k\right\|^2 = \sum_{i=1}^{n} \tilde{W}_k{}^2 = tr\{\tilde{W}_k \tilde{W}_k\}$. From the updating law (11), we have

$$\tilde{W}_{k+1} = \tilde{W}_k - \eta_k e(k)\phi' V^{0T} X^T(k)$$
$$\tilde{V}_{k+1} = \tilde{V}_k - \eta_k e(k)\phi^T$$

Since ϕ' is diagonal matrix, and by using (10) we have

$$\begin{aligned}\Delta L_k &= \left\|\tilde{W}_k - \eta_k e(k)\phi' V^{0T} X^T(k)\right\|^2 \\ &+ \left\|\tilde{V}_k - \eta_k e(k)\phi^T\right\|^2 - \left\|\tilde{W}_k\right\|^2 - \left\|\tilde{V}_k\right\|^2 \\ &= \eta_k^2 e^2(k)\left(\left\|\phi' V^{0T} X^T(k)\right\|^2 + \|\phi\|^2\right) \\ &- 2\eta_k \|e(k)\| \left\|V^0 \phi' \tilde{W}_k X(k) + \tilde{V}_k \phi\right\|\end{aligned} \tag{15}$$

By (10) we know

$$e(k) = \tilde{V}_k \phi[W_k X(k)] + V^0 \phi' \tilde{W}_k X(k) + \zeta(k)$$

Since $\eta > \eta_k > 0$, the last term in (15) is

$$\begin{aligned}&2\eta_k \|e(k)\| \left\|V^0 \phi' \tilde{W}_k X(k) + \tilde{V}_k \phi\right\| \\ &= 2\eta_k \|e(k)[e(k) - \zeta(k)]\| \\ &\ge 2\eta_k e^2(k) - 2\eta_k \|e(k)\zeta(k)\| \\ &\ge 2\eta_k e^2(k) - \eta_k e^2(k) - \eta_k \zeta^2(k) \\ &\ge \eta_k e^2(k) - \eta\zeta^2(k)\end{aligned}$$

So

$$\begin{aligned}&\Delta L_k \le -\eta_k e^2(k) \\ &\left[1 - \eta_k\left(\left\|\phi' V^{0T} X^T(k)\right\|^2 + \|\phi\|^2\right)\right] + \eta\zeta^2(k) \\ &\le -\pi e^2(k) + \eta\zeta^2(k)\end{aligned} \tag{16}$$

where π is defined in (13). Because $e(k)^2 \ge \frac{\eta}{\pi}\bar{\zeta}$, $\Delta L_k \le 0$, L_k is bounded, so W_k and V_k are bounded. By (10) $e(k)$ is bounded. If $e(k)^2 < \frac{\eta}{\pi}\bar{\zeta}$, $W_{k+1} = W_k$ and $V_{k+1} = V_k$, so W_k and V_k are bounded, $\|e(k)\|^2 < \frac{\eta}{\pi}\bar{\zeta} < \infty$ is also bounded. For all $e(k)$, (12) is correct.

If $e(k)^2 \ge \frac{\eta}{\pi}\bar{\zeta}$, (16) can be rewritten as

$$\Delta L_k \le -\pi e^2(k) + \eta\zeta^2(k) \le \pi e^2(k) + \eta\bar{\zeta} \tag{17}$$

Summarizing (17) from 1 up to T, and by using $L_T > 0$ and L_1 is a constant, we obtain

$$L_T - L_1 \le -\pi \textstyle\sum_{K=1}^{T} e^2(k) + T\eta\bar{\zeta}$$
$$\pi \textstyle\sum_{K=1}^{T} e^2(k) \le L_1 - L_T + T\eta\bar{\zeta} \le L_1 + T\eta\bar{\zeta}$$

Combing with $e(k)^2 < \frac{\eta}{\pi}\bar{\zeta}$, for all $e(k)$ (13) is established.

Remark 1: V^0 does not effect the stability property of the neuro modeling, but it influences the modeling accuracy, see (13). We design an off-line method to find a better value for V^0. If we let $V^0 = V_0$, the algorithm (11) can make the modeling error convergent, *i.e.*., V_k will make the modeling error smaller than that of V_0. V^0 may be selected by following steps:

1) Start from any initial value for $V^0 = V_0$, $k = 0$.
2) Update V_t by the learning law (11), until $k = T_0$.
3) If the $\|e(T_0)\| < \|e(0)\|$, let V_T as a new V^0, $V^0 = V_{T_0}$, go to 2 to repeat the modeling process.
4) If the $\|e(T_0)\| \ge \|e(0)\|$, stop this off-line modeling, now V_{T_0} is the final value for V^0.

Remark 2: Since we assume neural networks cannot match nonlinear systems exactly, we can not make the parameters (weights) convergence, we would like only to force the output of neural networks to follow the output of the plant, i.e. the modeling error is stable. Although the weights cannot converge to their optimal values, (13) shows that the modeling error will convergence to the ball radius $\frac{\eta}{\pi}\bar{\zeta}$. Even if the input is persistent exciting, the modeling error $\zeta(k)$ will not make the weights convergent to their optimal values. It is possible that the output error is convergent, but the weight errors are very high when the networks structure is not fine

defined. The relations of the output error and the weight errors are shown in (10). Simpler case is that we use line in the weights and the neural networks can match the nonlinear plant exactly

$$\text{plant}: y = W^*\phi[X(k)]$$
$$\text{neural networks}: \hat{y} = W_t\phi[X(k)]$$
$$\text{output error}: (y-\hat{y}) = (W^* - W_t)\phi[X(k)]$$

If $\phi[X(k)]$ is large, small output error $(y-\hat{y})$ does not mean good convergence of the weight error $(W^* - W_t)$.

Remark 3: Noise (or disturbance) is an important issue in the system modeling. There are two types disturbances: external and internal. Internal disturbance can be regarded as unmodeled dynamic $\delta(k)$ in (9). A bounded internal disturbance does not effect the theory results in this paper, but can enlarge the modeling error if the internal disturbance becomes bigger. External disturbance can be regarded as measurement noise, input noise, etc. In the point of structure, input noises are increased feedforward through each layer [2]. For example, a noise $\varsigma(k)$ is multiple by $V_k\phi[W_k\varsigma(k)]$ and arrives the output. Measurement noise is enlarged due to backpropagation of modeling error (11), therefore the weights of neural networks are influenced by output noise. On the other hand small external disturbance can accelerate convergent rate according to the persistent exciting theory [18], small disturbances in the input $u(t)$ or in output $y(t)$ can enrich frequency of the signal $X(t)$, this is good for parameters convergence. In the following simulation we can see this point. With this prior knowledge V^0, we may start the modeling (11).

IV. Application Study

In this section, we will use real data of PEMEX and the neural networks proposed in Section 3 to model crude oil blending. The TMDB crude oil blending process in PEMEX is shown in Fig.1, where the analyzers of API and flow rates are installed the input/output points of each block. The data is recorded in the form of Microsoft Excel daily. Each day, we have input data $[q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4]^T$ and output data $[q_f, p_f]^T$. We use "a=xlsread(data)" command to transform the data sheet into Matlab. The training data are two years' records, 730 input/output pairs. The testing data, 28 input/output pairs, are one month's records which are in the other year. In this way, we can assure the testing phase is independent of the training phase. The outputs of each blender (M_1, M_2, M_3) in Fig.1 are changed daily and different, Fig. gives one month's API Gravity of two blenders.

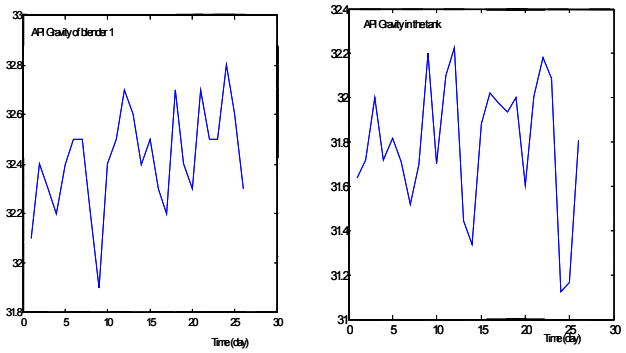

Fig.3 API gravity in one month

We see that the nonlinearity of the crude oil blending is strong; it is not easy to identify it by a simple model. We use three methods to compare the algorithm proposed in this paper, see Fig.4

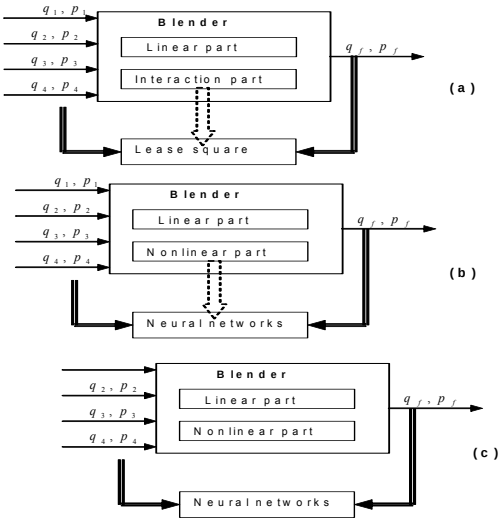

Fig.4 Modeling crude oil blending

In Fig.4-(a) we regard the real data satisfy the interaction model (6), *i.e.,*

$$p_f = \sum_{i=1}^{4} p_i x_i + \sum_{i=1}^{4}\sum_{j=i+1}^{4} \alpha_{i,j} x_i x_j \qquad (18)$$

where $x_i = q_i / q_f$, the parameters $\alpha_{i,j}$ are linear with the data, we can use standard least square technique to calculate the parameter · (18) can be written as

$$p_f(k) - \sum_{i=1}^{4} p_i(k) x_i(k) = \sum_{i=1}^{4}\sum_{j=i+1}^{4} \alpha_{i,j} x_i(k) x_j(k) \quad (19)$$

where $k = 1 \cdots 730$. We define

$$y(k) = p_f(k) - \sum_{i=1}^{4} p_i(k)x_i(k),$$
$$z_l(k) = x_i(k)x_j(k), \quad \theta_l = \alpha_{i,j}$$

where $i = 1\cdots 4, \quad j = (i+1)\cdots 4, \quad l = 1\cdots 10.$ (19) can be written in matrix form

$$Y = Z\Theta$$

where $Y = [y(1),\ldots,y(730)]^T$,

$$Z = \begin{bmatrix} z_1(1) & \cdots & z_{10}(1) \\ \vdots & \ddots & \vdots \\ z_1(730) & \cdots & z_{10}(730) \end{bmatrix}^T, \qquad \Theta = [\theta_1,\ldots,\theta_{10}]^T .$$

The least square solution is

$$\Theta = (Z^T Z)^{-1} Z^T Y$$

After we obtain Θ (or $\alpha_{i,j}$), the other month's data are used to check the model (18). We apply the input data $[q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4]^T$ to (18), the output $\hat{p}_f$ of the interaction model and the real data are shown in Fig..

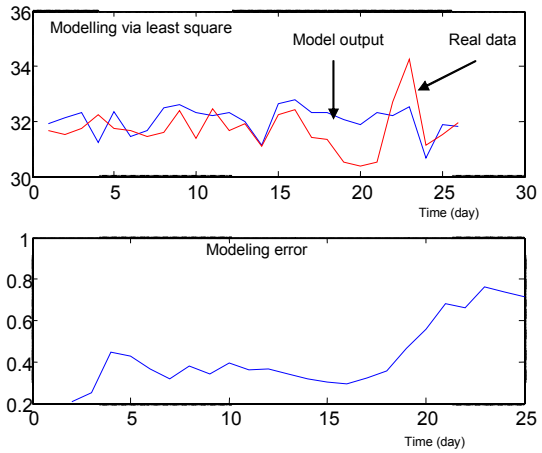

Fig.5 Identification via interaction model and least square

In Fig.4-(b) we assume the crude oil blending can be expressed as linear and nonlinear parts,

$$p_f = \sum_{i=1}^{4} p_i x_i + \Delta \qquad (20)$$

It can be expressed as

$$p_f - \sum_{i=1}^{4} p_i x_i = \Delta(q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4)$$

We use following neural network model to identify Δ,

$$\hat{y}(k) = V_k \phi[W_k X(k)] \qquad (21)$$

where k -th sample timer interval is one day . The input to neural network is $[q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4]$, the output of neural network corresponds to $\left(p_f - \sum_{i=1}^{4} p_i x_i\right)$, so $X(k) = [q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4]^T$. We choose the 5 nodes in hidden layer, so $W_k \in R^{5\times 8}$, $V_k \in R^{1\times 5}$, the initial conditions for the elements of W_k and V_k are random number in $[0,1]$. The modelling error is

$$e(k) = \hat{y}(k) - \left[p_f(k) - \sum_{i=1}^{4} p_i(k)x_i(k)\right]$$

We use the learning algorithm (11) proposed in this paper, *i.e.,*

$$W_{k+1} = W_k - \eta_k e(k)\phi' V^{0T} X^T(k)$$
$$V_{k+1} = V_k - \eta_k e(k)\phi^T \qquad (22)$$

where $\eta_k = \dfrac{s_k}{1 + \left\|\phi' V^{0T} X^T(k)\right\|^2 + \|\phi\|^2}$, $\eta = 1$, $\frac{\eta}{\pi}\overline{\zeta} = 0.2$, $\phi(\cdot) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\phi'(\cdot) = \sec h(x) = \frac{2}{e^x + e^{-x}}$. 730 pairs $[X(k), y(k)]$ are applied to train the neural networks (21), and other 28 pairs data are used to test the training result, the modeling results are shown in Fig.6

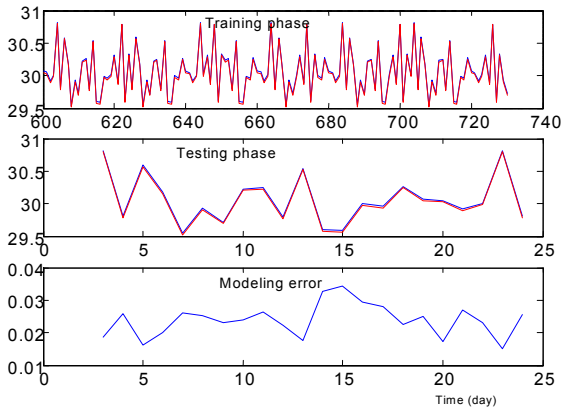

Fig.6 Identification for nonlinear part

In Fig.4-(c) we consider the crude oil blending is a black-box nonlinear process, we use neural network to model the whole system. The plant is

$$p_f = f(q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4) \qquad (23)$$

The input to neural network is $[q_1, p_1, q_2, p_2, q_3, p_3, q_4, p_4]$, the output of neural network corresponds to p_f. The modelling error is

$$e(k) = \hat{y}(k) - p_f(k)$$

We use the same neural networks and the same algorithm as in (22). We use 1500 data (four and half year´s data) to training it, after $k > 1300$, the weights are converged. Then we use another one year's flow rates of the feed stocks to test our neural model, the modeling results are shown in Fig.7.

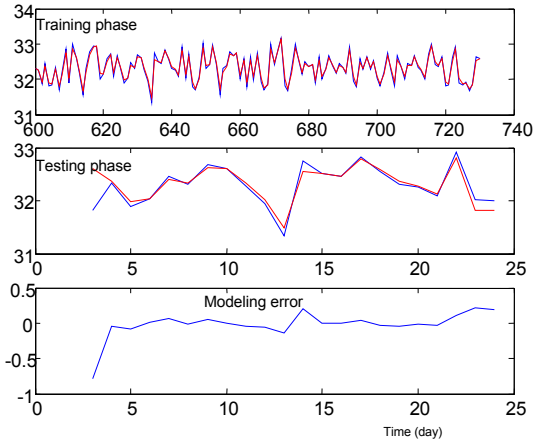

Fig.7 Black-box identification

Three different methods in Fig.4 give different modelling errors. We define the average modeling errors as

$$J_e = \frac{1}{n}\sum_{k=1}^{n}\left|p_f(k) - \hat{p}_f(k)\right|$$

where $\hat{p}_f(k)$ is the output of the models. For least square method, nonlinear part modeling and black-box modeling, J_e is 0.6, 0.0, 0.1 respectively. We have the following conclusions:

1) It is reasonable to divide the blending process into linear and nonlinear parts
2) The interaction model (18) for the nonlinear part is not suitable in crude oil blending.
3) Neural networks and the robust learning algorithm proposed in this paper are effective for modelling of crude oil blending.

Now we compare the dead-zone learning algorithm proposed in this paper (22) with normal backpropagation algorithm [16] in the training phase. We use the same multilayer neural networks as [16], it is $\Pi_{8,5,1}$ (The numbers of input layer, hidden layer and output layer are 8,5,1 , respectively.). We use a fixed learning rate $\eta = 0.05$. We found after $\eta > 0.1$ the normal backpropagation algorithm become unstable. The performance comparison can be realized by mean squared errors

$$J(N) = \frac{1}{2N}\sum_{k=1}^{N} e^2(k)$$

The comparison results are shown in Fig.8. We can see that the stable algorithm proposed in this paper has a fast convergence rate, $J(730) = 0.005$. The modeling error of normal backpropagation algorithm is bigger, $J(730) = 0.078$.

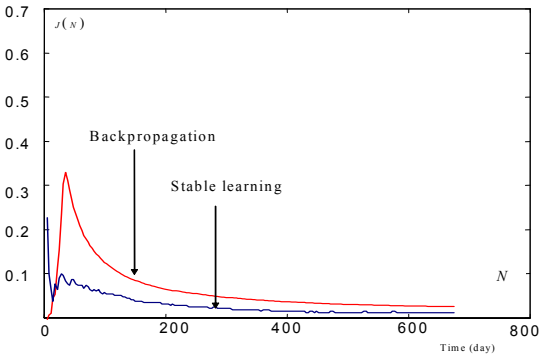

Fig.8 Comparison

V. Conclusion

In this paper a new dead-zone learning algorithm for discrete-time neural network is proposed. The theoretical analysis of stability and convergence of the neural networks are given. A real application for modelling of crude oil blending is provided. This method has great benefit for realizing model-based optimal control.

References

[1] J.Alvarez-Ramirez, A.Morales, R.Suarez, *Robustness of a class of bias update controllers for blending systems*, Industrial Engineering Chemistry Research, Vol.41, No.19, 4786-4793, 2002.
[2] M.Brown, C.J.Harris, *Neurofuzzy adaptive modelling and control*, Prentice Hall, 1994.
[3] D-M.Chang, C-C.Yu and I-L.Chien, *Coordinated control of blending systems*, IEEE Trans. Control Systems Technology,Vol.6, No.4, 495-506, 1998.
[4] B.Egardt, *Stability of Adaptive Controllers*, Lecture Notes in Control and Information Sciences, Vol.20, Springer-Verlag, Berlin, 1979.
[5] Z.Feng and A.N.Michel, *Robustness Analysis of a Class of Discrete-Time Systems with Applications to Neural Networks*, Proc. of American Control Conference, 3479-3483, San Deigo, 1999.
[6] J.H.Gary, G.E.Handwerk, *Petroleum Refining Technology and Economics*, Marcer Dekker, New York, 1994.

[7] W.C.Healy, C.W.Maassen and R.T.Peterson, *A new approach to blending octanes*, Proc.24th Meeting of American Petroleum Institute's Division of Refining, New york, 1959.

[8] J.J.Hopfield, *Neurons with grade response have collective computational propierties like those of a two-state neurons*, Proc. of the National Academy of Science, USA, vol. 81, 3088-3092, 1984.

[9] W.L.Luyben, Process Modeling, Simulation and Control for Chemical Enqineers, 2nd edition, McGraw-Hill, Inc.,1990.

[10] K.Murakami and D.E.Seborg, *Constrained parameter estimation with applications to blending operations*, Journal of Precess Control, Vol.10, 195-202, 2000.

[11] P.A.Ioannou and J.Sun, *Robust Adaptive Control, Prentice-Hall*, Inc, Upper Saddle River: NJ, 1996.

[12] S.Jagannathan and F.L.Lewis, *Identification of Nonlinear Dynamical Systems Using Multilayered Neural Networks*, Automatica, Vol.32, No.12, 1707-1712, 1996.

[13] L.Jin ad M.M.Gupta, *Stable Dynamic Backpropagation Learning in Recurrent Neural Networks*, IEEE Trans. Neural Networks, Vol.10, No.6, 1321-1334, 1999.

[14] Z.P.Jiang and Y.Wang, *Input-to-State Stability for Discrete-Time Nonlinear Systems*, Automatica, Vol.37, No.2, 857-869, 2001.

[15] E.B.Kosmatopoulos, M.M.Polycarpou, M.A.Christodoulou and P.A.Ioannou, *High-Order Neural Network Structures for Identification of Dynamical Systems*, IEEE Trans. on Neural Networks, Vol.6, No.2, 442-431, 1995.

[16] K.S.Narendra and K.Parthasarathy, *Identification and Control of Dynamical Systems Using Neural Networks*, IEEE Trans. Neural Networks, Vol.1, No.1, 4-27, 1990.

[17] A.Muller, *New method produces accurate octane blending values*, Oil & Gas J., Vol.23, No.3, 80-90, 1992.

[18] M.M.Polycarpou and P.A.Ioannou, *Learning and Convergence Analysis of Neural-Type Structured Networks*, IEEE Trans. Neural Networks, Vol.3, No.1, 39-50, 1992.

[19] J.M.Smith, H.C.Van Ness, *Introduction to Chemical Engineering Thermodynamics*, McGraw-Hill Book Company, New York, 1982.

[20] A.Singh, J.F.Forbes, P.J.Vermeer, S.S.Woo, *Model-based real-time optimization of automotive gasoline blending operations*, Journal of Process Control, Vol.10, 43-58, 2000.

[21] Q.Song, *Robust Training Algorithm of Multilayered Neural Networks for Identification of Nonlinear Dynamic Systems*, IEE Proceedings - Control Theory and Applications, Vol.145, No.1, 41-46,1998.

[22] J.A.K. Suykens, J. Vandewalle, B. De Moor, *NLq Theory: Checking and Imposing Stability of Recurrent Neural Networks for Nonlinear Modelling*, IEEE Transactions on Signal Processing (special issue on neural networks for signal processing), Vol.45, No.11, 2682-2691, 1997.

[23] J.A.K.Suykens, J.Vandewalle and B.De Moor, Lur'e Systems with Multilayer Perceptron and Recurrent Neural Networks; *Absolute Stability and Dissipativity*, IEEE Trans. on Automatic Control, Vol.44, 770-774, 1999.

[24] Wen Yu, Alexander S. Poznyak, Xiaoou Li, *Multilayer Dynamic Neural Networks for Nonlinear System On-line Identification*, International Journal of Control, Vol.74, No.18, 1858-1864,2001.

[25] Wen Yu, Xiaoou Li, *Discrete-Time Neuro Identification without Robust Modification, IEE Proceedings- Control Theory and Applications*, Vol.150, No.3, 311-316, 2003.

[26] Wen Yu, *Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms*, Information Sciences: An International Journal, Vol.158, No.1, 131-147, 2004.

[27] A.H.Zahed, S.A.Mullah and M.D.Bashir, *Predict octane number for gasoline blends*, Hydrocarbon Processing, No.5, 85-87, 1993.

[28] Y.Zhang, D.Monder and J.F.Forbes, *Real-time optimization under parametric uncertainty a probability constrained approach*, Journal of Process Control, Vol.12, 373-389, 2002.

Wen Yu (M'97–SM'04) He received the B.S. degree from Tsinghua University, Beijing, China in 1990 and the M.S. and Ph.D. degrees, both in Electrical Engineering, from Northeastern University, Shenyang, China, in 1992 and 1995, respectively. From 1995 to 1996, he served as a Lecture in the Department of Automatic Control at Northeastern University, Shenyang, China. In 1996, he joined CINVESTAV-IPN, México, where he is a professor in the Departamento de Control Automático. He has held a research position with the Instituto Mexicano del Petróleo, from December 2002 to November 2003. His research interests include adaptive control, neural networks, and fuzzy Control.