Optimization of Crude Oil Blending with Neural Networks

Wen Yu, José de Jesús Rubio and América Morales

Abstract—Crude oil blending is an important unit in petroleum refining industry. Most of blend automation system is a real-time optimizer (RTO). RTO is a model-based optimization approach that uses current process information to update the model and predict the optimal operating policy. But in many oil fields, they hope to make decision and do supervision control based on the history data, i.e., they want to know the optimal inlet flow rates without on-line analyzers. To overcome the drawback of the conventional RTO, in this paper we use neural networks to model the blending process by the history data. Then the optimization is carried out via the neural model. The contributions of this paper are: (1) We propose a new approach to solve the problem of blending optimization based on history data. (2) Sensitivity analysis of the neural optimization is given. (3) Real data of a oil field is used to show effectiveness of the proposed method.

I. INTRODUCTION

Crude oil blending is a optimization operations based upon extensive process knowledge and experience. Many advance control techniques are proposed recently to improve crude oil blending qualities [13][20]. A common method to approximate the blending operation is to use linear (ideal) model [13] or regard blending operation has a sufficient small nonlinear uncertainty [1]. Some attempts were made to introduce simplified models in order to construct “model-based” controller [10]. Nowadays, many crude oil blending still use linear relation to calculate the control input. The quality of crude oil exportation will be improved if optimization technique is used. Some commercial crude oil optimizing controllers are based on the technique of linear programming

\[ \min c^T q(k) \]

subject: \[ \begin{align*}
A_{eq}q(k) &= b_{eq} \\
q_{\text{min}} &\leq q(k) \leq q_{\text{max}} \\
h[q(k)] &\leq 0
\end{align*} \] (1)

where \( c \) is a constant vector containing process economics, \( q(k) \in \mathbb{R}^n \) is the feedstock at time \( k \), \( A_{eq}q(k) = b_{eq} \) is mass balance in the blending process, \( q_{\text{min}} \leq q(k) \leq q_{\text{max}} \) is feedstock availability constraint, \( h[q(k)] \) is product demand. If the blending process is regarded as a linear model

\[ h[q(k)] = Bq(k) \]

where \( B \) is known constant matrix. So (1) can be solved by standard linear programming (LP) technique (for example Simplex). In general the mixing rule is nonlinear, an alternative method is to add a correction term (uncertainties) to the linear mixing rule, by real data analysis we found that for crude oil blending there are about 10%-30% nonlinearity. To address the blending operation uncertainties, model-based real-time optimization (RTO) have been proposed [12][13], it uses the following model for \( h[q(k)] \)

\[ p_f(k) = h[q(k)] \]

where \( p_f(k) \) is product output which is measurable from an on-line analyzer. Some set of adjustable model parameters are updated by \( p_f(k) \), then the optimum control \( q^*(k+1) \) are calculated using predictions of future product quality values by the updated model. In many cases, the optimal inlet flow rates based on the history data are required. These set-points can be used for decision and supervision control. In order to optimize crude oil blending without on-line analyzer, a realistic model for crude oil blending based on operation data is needed. Recent results show that neural network technique seems to be very effective to model a broad category of complex nonlinear systems when we do not have complete model information [17][16]. Form input/output pair \( [q(k), p_f(k)] \),\( k = 1,2, \ldots \), we can model the uncertainty restriction \( h[q(k)] \) in (1). By using some RTO techniques, the problem of optimization of crude oil blending based on the history data can be solved.

Since (1) is a linear programming with uncertainty restrictions, these uncertainties can seriously degrade the performance of the optimization. [20] gave a probability constrained approach to analyze the optimization under uncertainty restrictions, but it is limited to situations where uncertain parameters enter the constraints nonlinearly and uncertain economics enter the objective function. In this paper, we use the term "robustness" which is refer to the ability of the system to calculate an “optimal” operating policy that will be feasible respect to some bounded uncertainties. The sensitivity analysis of the neural optimization is given by algebra method, it is more simple and the uncertainties are only required to be bounded. Finally, the neural optimization approach is successfully used to solve crude oil blending via real data.

This work was supported by CONACyT under Grant 38505A.
Wen Yu and José de Jesús Rubio are with the Departamento de Control Automático, CINVESTAV-IPN, Av. IPN 2508, México D.F., 07360, México ywu@ctrl.cinvestav.mx
América Morales is with the Instituto Mexicano del Petróleo, Eje Central Lazano Cardenas 152, Mexico D.F., 07730, Mexico bmorales@imp.mx

0-7803-8682-5/04/$20.00 ©2004 IEEE
II. MODELLING AND OPTIMIZATION OF CRUDE OIL BLENDING WITH NEURAL NETWORKS

The objective is to produce blended crude oil to a target specification at the lowest cost using the minimum higher cost crude oil. 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. In this paper we discuss a typical crude oil blending process in PEMEX (Mexican Petroleum Company), it is called Terminal Maritima de dos Bocas Tabasco (TMDB). The flow-sheet is shown in Fig. 1-(a). It has three blenders, one dehydration equipment and one tank. We use Fig.1-(b) to describe the static properties of the crude oil blending.

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

$$\rho_o = \sum_{i=1}^{2} p_i u_i, \quad q_o = \sum_{i=1}^{2} q_i, \quad u_i = \frac{q_i}{q_o}$$

where $u_i, q_i$ and $p_i$ are the volume fraction, flow rate and API Gravity of $i$th feed-stock, $p_o$ and $q_o$ are the flow rate and the API Gravity of the blended product. Unfortunately, this equation is correct only in the ideal condition, in order to make it universally valid a correction term $\Delta_1$ is added

$$p_o = \sum_{i=1}^{2} p_i u_i + \Delta_1$$

where $\Delta_1$ is called the property change of mixing. Several approaches can approximate $\Delta_1$, for examples

- Interaction model [13][1]

$$\Delta_1 = \sum_{i=1}^{2} M_i (u_i p_i)$$

- Zahed model [19]

$$\Delta_1 = \alpha u_1 u_2$$

where $\alpha$ is the interaction coefficient between the two components.

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

$$p_f = \sum_{i=1}^{4} p_i u_i + \Delta$$

where $p_f$ is the final product, $\Delta$ is the whole nonlinearity. All of above models are only suitable in some special conditions and the parameters of these models should be determined by experience data. Neural networks modeling is in sense of "black-box", it is suitable for modelling of crude oil blending. Static neural networks can be used to identify the nonlinear parts $\Delta$ or the whole integrated model (6). The static property can be written in following form

$$y(k) = \Phi [u_1 (k), u_2 (k), u_3 (k), u_4 (k)]$$

In general

$$y(k) = \Phi (X (k))$$

where

$$X (k) = [u_1 (k), u_2 (k), u_3 (k), u_4 (k)]^T$$

$\Phi (\cdot)$ is an unknown nonlinear function representing the blending operation, $u (k)$ is measurable scalar input, it is API Gravity and flow rate, $y(k)$ is the blended API Gravity value at time $k$. We consider multilayer neural network to model the static operation of the blending as in (7)

$$\hat{y} (k) = V_k \phi (W_k X (k))$$

where the scalar output $\hat{y}(k)$ and vector input $X (k) \in R^n \times 1$ is defined in (8), the weights in output layer are $V_k \in R^1 \times m$, the weights in hidden layer are $W_k \in R^m \times n$, $\phi$ is $m$-dimension vector function. The typical presentation of the element $\phi_i (\cdot)$ is sigmoid function. The identified blending system (7) can be represented as

$$y (k) = V^o \phi (W^o X (k)) - \mu (k)$$
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

\[
y(k) = V^0 \phi[W^* X(k)] - \delta(k)
\]

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 identification error can be represented as

\[
e(k) = \hat{\gamma}(k) - y(k) = V_k \phi[W_k X(k)] - \delta(k)
\]

\[
- V^0 \phi[W^* X(k)] + \delta(k)
\]

\[
= V_k \phi[W_k X(k)] - V^0 \phi[W^* X(k)] + \delta(k)
\]

\[
+ V^0 \phi[W_k X(k)] - V^0 \phi[W^* X(k)] + \delta(k)
\]

\[
= \hat{V}_k \phi[W_k X(k)] + V^0 \phi[W^* X(k)] + \zeta(k)
\]

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

In this paper we are only interested in open-loop identification, 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 \( X(k) \) is bounded. By the boundedness of the sigmoid function \( \phi(\cdot) \), we assume that \( \delta(k) \) in (10) is bounded, also \( \varepsilon(k) \) is bounded. So \( \zeta(k) \) in (11) is bounded. The following theorem gives a stable backpropagation-like algorithm for discrete-time multilayer neural network.

**Theorem 1**: If we use the multilayer neural network (9) to model the crude oil blending (7), the following dead-zone backpropagation-like algorithm

\[
W_{k+1} = W_k - \eta_k e(k) \phi' V^0 T X^T(k)
\]

\[
V_{k+1} = V_k - \eta_k e(k) \phi T
\]

(12)

where \( \eta_k = \frac{s_k}{1 + \| \phi' V^0 T X^T(k) \|^2 + \| \phi \|^2} \)

\[
s_k = \begin{cases} 1 & e(k)^2 \geq \frac{\pi}{\kappa} \\ 0 & e(k)^2 < \frac{\pi}{\kappa} \end{cases}
\]

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

\[\kappa = \max_k \left( \| \phi' V^0 T X^T(k) \|^2 + \| \phi \|^2 \right)\]

this updating law can make modelling error \( e(k) \) and the weights of neural networks bounded

\[
\| e(k) \| \leq L_{\infty}, \quad W_k \in L_{\infty}, \quad V_k \in L_{\infty}
\]

(13)

The average of the modelling error satisfies

\[
J = \lim_{T \to \infty} \sup_{T} \frac{1}{T} \sum_{k=1}^{T} e^2(k) \leq \frac{\pi \bar{\pi}}{\pi}
\]

(14)

**Proof**: We selected a positive defined matrix \( L_k \) as

\[
L_k = \| \hat{W}_k \|^2 + \| \hat{V}_k \|^2
\]

(15)

From the updating law (12), we have

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

\[
\hat{V}_{k+1} = \hat{V}_k - \eta_k e(k) \phi T
\]

Since \( \phi' \) is diagonal matrix, and by using (11) we have

\[
\Delta L_k = \| \hat{W}_k - \eta_k e(k) \phi' V^0 T X^T(k) \|^2
\]

\[
+ \| \hat{V}_k - \eta_k e(k) \phi T \|^2 - \| \hat{W}_k \|^2 - \| \hat{V}_k \|^2
\]

\[
= \eta_k^2 e^2(k) \left( \| \phi' V^0 T X^T(k) \|^2 + \| \phi \|^2 \right)
\]

\[
- 2\eta_k \| \phi(k) \| \| \hat{V}_k \| \phi \hat{W}_k X(k) + \hat{V}_k \phi \| \| \phi \|^2
\]

\[
= \eta_k^2 e^2(k) \left( \| \phi' V^0 T X^T(k) \|^2 + \| \phi \|^2 \right)
\]

\[
- 2\eta_k \| \phi(k) \| \| \phi(k) - \zeta(k) \| \| \phi \|^2
\]

\[
\leq - \eta_k e^2(k) \left[ 1 - \eta_k \left( \| \phi' V^0 T X^T(k) \|^2 + \| \phi \|^2 \right) \right]
\]

\[
+ \eta_k^2 e^2(k)
\]

\[
\leq - \pi e^2(k) + \eta_k^2 e^2(k)
\]

(16)

where \( \pi \) is defined in (14). Because

\[
n \left[ \min(\bar{\pi}^2) + \min(\bar{\pi}^2) \right] \leq L_k \leq n \left[ \max(\bar{\pi}^2) + \max(\bar{\pi}^2) \right]
\]

where \( n \left[ \min(\bar{\pi}^2) + \min(\bar{\pi}^2) \right] \) and \( n \left[ \max(\bar{\pi}^2) + \max(\bar{\pi}^2) \right] \) are \( \mathcal{K}_\infty \)-functions, and \( \pi e^2(k) \) is an \( \mathcal{K}_\infty \)-function, \( \eta^2(k) \) is a \( \mathcal{K}_\infty \)-function. From (11) and (15) we know \( V_k \) is the function of \( e(k) \) and \( \zeta(k) \), so \( L_k \) admits a smooth ISS-Lyapunov function, the dynamic of the identification error is input-to-state stable [7]. Because the "INPUT" \( \zeta(k) \) is bounded and the dynamic is ISS, the "STATE" \( e(k) \) is bounded.

(16) can be rewritten as

\[
\Delta L_k \leq - \pi e^2(k) + \eta_k^2 e^2(k) \leq \pi e^2(k) + \eta_k^2 e^2(k)
\]

(17)

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

\[
L_T - L_1 \leq - \pi \sum_{K=1}^{T} e^2(k) + T \eta_k^2 \sum_{K=1}^{T} e^2(k)
\]

\[
\pi \sum_{K=1}^{T} e^2(k) \leq L_1 + L_T + T \eta_k^2 \sum_{K=1}^{T} e^2(k)
\]

(14) is established. \( \blacksquare \)

\( V^0 \) does not effect the stability property of the neuro identification, but it influences the identification accuracy, see (14). We design an off-line method to find a better value for \( V^0 \). If we let \( V^0 = V_0 \), the algorithm (12) can make the identification error convergent, i.e., \( V_k \) will make the identification 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 \( V_k \) by the learning law (12), 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 identification process.
4. Stop this off-line identification, now \( V_{T_0} \) is the final value for \( V^0 \).
The optimization of crude oil blending is to minimize the following cost

$$\min c^T q(k)$$

where $c$ is a constant vector containing process economics, $q(k) \in \mathbb{R}^n$ is the feedstock at time $k$. The constraints are as follows

1. Feedstock availability constraint,

$$q_{i,\min} \leq q_i \leq q_{i,\max}$$

2. Mass balance

$$\sum_{i=1}^{n} q_i = q_f, \text{ or } \sum_{i=1}^{n} u_i = 1$$

where $u_i$ is defined in (2).

3. Product quality requirements

$$p_{f,\min} \leq p_f \leq p_{f,\max}$$

The optimization is

$$\min c^T u(k)$$

subject:

$$\begin{align*}
A_{eq} u(k) &= b_{eq} \\
u_{\min} \leq u(k) &\leq u_{\max} \\
p_{f,\min} \leq p_f &\leq p_{f,\max}
\end{align*}$$

where $u(k) = [u_1 \cdots u_4], n = 4, A_{eq} = [1 \cdots 1], b_{eq} = 1$. We can see that the last restriction cannot be applied in LP directly. From (6) we know

$$p_f(k) = u(k)^T p + \Delta(k)$$

where $p_f(k)$ is measurable, $u(k)$ is calculated by optimization method. From control theory viewpoint, $\Delta(k)$ can be estimated by the prior $q(k-1)$. From a constant initial condition for $\Delta$, for example $\Delta(0) = 0$, $\Delta$ can be estimated after $q(k-1)$ is calculated by optimization

$$\Delta(k) = p_f(k-1) - u(k-1)^T p$$

So the last restriction of (22) becomes

$$p_{f,\min} \leq u(k)^T p + \Delta(k) \leq p_{f,\max}$$

where $\Delta(k)$ is calculated by (24), it is also called bias updating [5]. Now (22) is

$$\min c^T u(k)$$

subject:

$$\begin{align*}
A_{eq} u(k) &= b_{eq} \\
u_{\min} \leq u(k) &\leq u_{\max} \\
A u(k) &\leq b
\end{align*}$$

where $A = \begin{bmatrix} p & -p \end{bmatrix}, b = \begin{bmatrix} p_{f,\max} - \Delta(k) \\ -p_{f,\min} + \Delta(k) \end{bmatrix}$. (25) can be solved by standard linear programming by (24).

The optimization problem of (25) needs real-time analyzer for $p_f(k)$. In this paper $p_f(k)$ is calculated from the neural model (9) which is obtained from the history data

$$\hat{p}_f = V^\star \phi [W^\star u(k)]$$

where $V^\star$ and $W^\star$ are the final values of the weights after the learning by the updating law (12). For the neural optimization,

$$\Delta(k) = V^\star \phi [W^\star u(k-1)] - u(k-1)^T p$$

III. SENSITIVITY ANALYSIS OF THE NEURAL OPTIMIZATION

We rewrite (25) in the following form

$$\min c^T u$$

subject:

$$\begin{align*}
D_1^T u + b_1 &= 0 \\
D_2 u + b_2 + \mu(u) &= 0
\end{align*}$$

where

$$\begin{align*}
D_1 &= [1, \ldots, 1]^T, & b_1 &= -1, \\
D_2 &= \begin{bmatrix} I_{n \times n}, -I_{n \times n}, p^T, -p^T \end{bmatrix}^T \\
b_2 &= [-u_{\max}, u_{\min}, -p_{\max}, p_{\min}]^T \\
\mu(u) &= [0_n, 0_n, \Delta(u), -\Delta(u)]
\end{align*}$$

$\Delta(u)$ is nonlinearity which is defined in (24). The equation (28) can be expressed in Lagrange form

$$L(u, \lambda, \mu) = c^T u + \lambda [D_1^T u + b_1] + \mu^T [D_2 u + b_2 + \mu(u)]$$

where $\mu = [\mu_1, \ldots, \mu_{2n+2}]^T \in \mathbb{R}^{2n+2}$. Using Kuhn-Tucker condition, the solution satisfies

$$\frac{\partial L}{\partial u} = 0, \quad \frac{\partial L}{\partial \lambda} = 0, \quad \frac{\partial L}{\partial \mu} = 0$$

$$\mu^T [D_2 u + b_2 + \mu(u)] = 0, \quad \mu^T \geq 0$$

So

$$Ax + M(x) = y_{sp}$$

$$A = \begin{bmatrix} D_1^T & 0_{n \times (2n+3)} \\
D_2 & I_{(2n+3) \times (2n+3)} \end{bmatrix}$$

$$M(x) = \begin{bmatrix} 0_{n \times (2n+3)} \\
\mu(u) \nabla \mu(u)^T \mu \end{bmatrix}$$

$$y_{sp} = \begin{bmatrix} -b_1 \\
-b_2 \end{bmatrix} + c - \lambda D_1 - D_2^T \mu$$

where $x = [0_{2n+1}, u^T]^T, p = 3n + 3, \nabla \mu(u) = [\frac{\partial \mu(u)}{u_1}, \ldots, \frac{\partial \mu(u)}{u_n}]$

$$A = \begin{bmatrix} D_1^T & 0_{n \times (2n+3)} \\
D_2 & I_{(2n+3) \times (2n+3)} \end{bmatrix}$$

$$M(x) = \begin{bmatrix} 0_{n \times (2n+3)} \\
\mu(u) \nabla \mu(u)^T \mu \end{bmatrix}$$

$$y_{sp} = \begin{bmatrix} -b_1 \\
-b_2 \end{bmatrix} + c - \lambda D_1 - D_2^T \mu$$

It is a algebraic system. Since $D_1 = [1, \ldots, 1]^T, A$ is invertible. To analyze the sensitivity, let us consider the following assumptions
A.1 \( x_{op} \) is the unique solution of the algebraic system of equation (31) for all \( t \geq 0 \), i.e., there is only one value of \( x_{op} \) such that \( Ax_{op} + M(x_{op}) = y_{sp} \). This means that \( u_{op} \) is the unique extremum of the program of equation (28). If \( u_{op} \) is not unique solution, the convergence results described bellow will hold only locally (in a neighborhood of \( x_{op} \)).

A.2 If \( \bar{A} \) is an estimated of \( A \), both \( A \) and \( \bar{A} \) are invertible matrixes, this means the regularity of \( u_{op} \) relative to the problem of equation (28).

By the assumptions A.1 and A.2, the linear programming problem (28) is equivalent to the algebraic system (31). Let define the modeling error as

\[
B(x) = A - \bar{A} \ x + M(x)
\]  

(32)

Substituting it to (31) we have

\[
B(x) + \bar{A} x = y_{sp}
\]  

(33)

According to the basic idea of bias update, the modeling error \( B_k \) can be estimated by the real output \( i.e. \)

\[
B_k = y_{k-1} - \bar{A} x_{k-1}
\]  

(34)

where \( y_{k-1} \) is measurable output. \( y_{sp} \) can be regarded as the desired output of the algebraic system of equation (33). (33) and (34) can be written as

\[
x_k = \left( I_n - \bar{A} \right)^{-1} \left( y_{sp} - y_{k-1} + \bar{A} x_{k-1} \right)
\]  

(35)

Because the real output satisfies (31), \( y_{k-1} = \bar{A} x_{k-1} + M(x_{k-1}) \). (35) is

\[
x_k = x_{k-1} + \left( I_n - \bar{A} \right)^{-1} \left( y_{sp} - Ax_{k-1} - M(x_{k-1}) \right)
\]  

(36)

By assumption A.1, we know that \( x_k \rightarrow x_{op} \) asymptotically (hence \( u_k \rightarrow u_{op} \) asymptotically). From (33), we have \( y_{sp} = Ax_{op} + M(x_{op}) \). Let us define \( e_k = x_k - x_{op} \), so (36) becomes

\[
e_k = e_{k-1} - \left( I_n - \bar{A} \right)^{-1} \left[ M(x_{k-1}) - M(x_{op}) \right]
\]  

(37)

It has a unique equilibrium point \( e_{eq} = 0 \), at least in a neighborhood of the origin. Notice that, if \( M(x) = 0 \) and \( \bar{A} = \bar{A} \), then \( e_k = 0 \), that is, if there is not uncertainty, the update scheme convergent in one step; hence, in the presence of small uncertainty, the bias update scheme will converge asymptotically to the optimum value \( x_{op} \). This is stated in the following proposition.

**Proposition 1:** Consider the optimization problem (28). Then \( u_k \rightarrow u_{op} \) if:

\[
\left \| I_n - \bar{A} \left( A + J_M(x) \right) \right \| < 1
\]  

(38)

for all \( x \in \mathbb{R}^p \), where \( J_M(x) \) is the Jacobian matrix of the nonlinearity \( M(x) \).

With this result, we can make the following comments:

**Remark 1:** The main conclusion form the last proposition is that, compared to the ideal mixing rule, nonlinearities should be small enough in order to guarantee convergence of the linear programming. If the nonlinear mixing affects \( \Delta(y) \) is too much, \( I_n - \bar{A} \) is big, it is expected that \( J_M(x) \) acts as a perturbation to the matrix \( A \), in this way, equation (38) is not fulfilled. This fact limits seriously the applicability of linear programming with bias update scheme to solve blending control problems with significant nonlinear affects.

**Remark 2:** The optimization scheme can be seen as an integral acting on the regulation error \( y_{sp} - y \). In fact, from equation (35) it is easy to get the next equation:

\[
x_k = x_0 + \left( I_n - \bar{A} \right)^{-1} \sum_{j=1}^{k-1} (y_{sp} - y_j)
\]  

where \( \bar{A} \) plays the role of an integral control gain. In continuous case, such integral action looks like \( x(t) = x_0 + \int_0^t (y_{sp} - y(t)) \, dt \). In this way, the purpose of the bias update scheme is to add integral control action into the optimizer controller for compensate for model-plant errors.

**Remark 3:** When we use neural network to model the nonlinearity in equation (31) we have

\[
Ax + V_k \phi [W_k X(k)] + \delta = y_{sp}
\]  

(39)

where \( \delta \) is modeling error of the neural network. Because \( V_k \phi [W_k X(k)] \) is known , we can join it in \( y_{sp} \), so

\[
Ax + \delta = y_a
\]  

(40)

where \( y_a = y_{sp} - V_k \phi [W_k X(k)] \). Obviously,

\[
\| \delta \| \ll \| M(x) \|
\]  

(41)

The optimization of crude oil blending with neural networks modeling can improve robust property.

**IV. CASE STUDY**

In this section, PEMEX’s real data is applied on the neural optimization. The data is in the form of Microsoft Excel daily, we use "a=xlsread(data)" command to transform the data sheet into Matlab. We use two years data, 730 records, to identify the process (training phase), then we
use one month data in the other year to check the modelling result (testing phase). The neural network model is (9) with $W_k \in \mathbb{R}^{3 \times 4}$, $V_k \in \mathbb{R}^{1 \times 9}$, $W_0$ and $V_0$ are random number in $[0, 1]$. The learning algorithm is (12) with $\epsilon = 0.2$. The modelling results is shown in Fig.2.

The neural optimization is (25) and (27). The constant vector $c = [34, 32, 10, 20]^T$, the bounds of feedstocks are $u_{\text{min}} = [0, 0, 0, 0]^T$, $u_{\text{max}} = [0.6, 0.5, 0.5, 0.2]^T$, $p_{f, \text{min}} = 30$, $p_{f, \text{max}} = 35$. Fig.3 shows the optimization results. Here (a) is the process of the neural optimization, the final values are $u_1 = 0.01$, $u_2 = 0.3$, $u_3 = 0.5$, $u_4 = 0.199$. (b) is the cost value $c^T u(k)$ of the neural optimization. (c) is the real operation data of a month, (d) is the corresponding cost. We see that the neural optimization can minimize the cost under the product restrictions. This little change can bring a great of economical benefit in a oil field.

V. CONCLUSION

The main contributions of this paper are: (1) we propose a new approach to solve the problem of blending optimization based on history data. (2) Sensitivity analysis of the neural optimization is given. (3) Real data of a oil field is used to show effectiveness of the proposed method.

REFERENCES