Optimization of crude oil blending with neural networks and bias update scheme

Wen Yu and Xiaoou Li

1Departamento de Control Automatico, CINVESTAV-IPN, A.P. 14-740, Av.IPN 2508, Mexico D.F., 07360, Mexico
2Departamento de Computación, CINVESTAV-IPN A.P. 14-740, Av.IPN 2508, Mexico D.F., 07360, Mexico

Crude oil blending is an important unit operation in petroleum refining industry. There are two difficulties for commercial crude oil optimizing controllers: (1) the optimization is stable only in some special conditions, the simplex-base algorithm is not robust; (2) the optimal control should be realized on-line and it cannot be analyzed off-line based on the history data. In this paper, we propose a neural networks approach to overcome these two drawbacks. We first use a recurrent neural networks to solve the linear programming with bias update. Then we use a static neural networks to modeling crude oil blending process based on the history data. Input-to-state stability approach is applied to access robust learning algorithms of the neural networks. Numerical simulations are provided to illustrate the successful application of neural networks on optimization.

1. INTRODUCTION

Crude oil blending is an attractive solution for those refiners who have the ability to blend different types crude oil to provide a consistent and optimal feedstock to refinery operations. Optimal crude purchasing is an effective method to improve refinery profit margins. Crude oil blending is an optimization blending operations based upon extensive process knowledge and experience [4]. The optimal crude properties are provided by refinery models that define optimal utilization of downstream units for various crude types.

Many commercial crude oil optimizing controllers are based on the technique of linear programming. In general the mixing rule is nonlinear, an alternative method is to add a correction term (uncertainties) to the linear mixing rule. To address the blending operation uncertainties, real-time optimization have been proposed [10][11]. Bias updating scheme is a key toll to solve the problem of real-time optimization. The main drawback of bias updating is that it requires the blended qualities are available from on-line measurement. Sometimes we also need offline optimization, for example we hope to find a reference optimal inlet flow rate based on history data, this control is useful for decision and supervision control.

The idea behind the neural networks-based optimization techniques is that the objective function and constraints are mapped into a closed-loop networks so that when a constraint
violation occurs, the magnitude and direction of the violation are fed back to adjust the states of the neurons of the networks. In this way overall energy of the networks is always decreasing until it achieved a minimum [5].

A lot of numerical algorithms have been developed for solving the optimization problem using computer, such as Optimization Toolbox of Matlab. The main disadvantage of these algorithms is that they generally converge slowly and there exists NP-hard combinatorial optimization problem. An alternative method is to use a dedicated electrical circuit which simulates both the objective and the constraint functions [2].

In order to improve crude oil blending qualities, many advance control techniques are proposed recently [11][17]. All of these approaches need good models of blending operation. The exact mathematical model for crude oil blending is too complex to be handled analytically. Many attempts were made to introduce simplified models in order to construct model-based controller [7]. A common method to approximate the blending operation is to use linear (ideal model) [11] or regard blending operation has a sufficient small nonlinear uncertainty [1]. Practically most of real blending handle no-ideal.

A realistic model for crude oil blending based on operation data seems to be very important for engineers. 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 [15][14]. 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.

In this paper, we first we use a recurrent neural networks to solve the linear programming with bias update. Then we use a static neural networks to modeling crude oil blending process based on the history data. Input-to-state stability approach is applied to access robust learning algorithms of the neural networks. Numerical simulations are provided to illustrate the successful application of neural networks on optimization.

2. PRELIMINARIES

Definition 1 X is a $\mathbb{R}^n$ is said to be convex, iff any $a, b \in X$ implies $[a, b] \subseteq X$, where

$$[a, b] = \{x \in \mathbb{R}^n | x = \lambda a + (1 - \lambda)b, 0 \leq \lambda \leq 1\}$$

Definition 2 $X \subseteq \mathbb{R}$ is said to be convex ($X \subseteq \mathbb{R}^n$ is a non-empty convex set), iff

$$f(\lambda a + (1 - \lambda)b) = \lambda f(a) + (1 - \lambda)f(b), 0 \leq \lambda \leq 1$$

for any $a, b \in X$.

Definition 3 $f : X \rightarrow \mathbb{R}$ is said to be affine function, if $f$ and $-f$ are convex.

Definition 4 H is a hyperplane in $\mathbb{R}^n$, iff there exists $a \in \mathbb{R}^n$, $a \neq 0$, and $\alpha \in \mathbb{R}$ such that

$$H = \{x \in \mathbb{R}^n | (a, x) = \alpha\}$$

where $(a, x)$, is the Euclidean inner product on $\mathbb{R}^n$, $\alpha$ is a normal vector of H.

Definition 5 The program $P$

$$P : \begin{align*}
\min f(x) \\
\text{subject } g_i \leq 0, h_j = 0, i = 1, \ldots, r, j = 1, \ldots, m
\end{align*}$$

is said to be convex program if $f$ and $g_i$ are convex functions and $h_j$ are affine functions.

Definition 6 $x$ is called a feasible solution to $P$, iff $x$ satisfies $r + m$ constraints of $P$.

Definition 7 Binding set $B = \{i | g_i = 0\}$.

Definition 8 Regular point $x_0$, if the gradients $Vg_i(x_0)$, $1 \leq j \leq m$ are linearly independent

$$f(x) = \frac{1}{2}x^TAx + a^T x + b$$

where A is a symmetric matrix, $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, is convex (strictly convex) on $\mathbb{R}^n$ iff A is positive semi-definite (positive definite).

Theorem 1 (Kuhn-Tucker optimal theorem [9]) Let $P$ be a convex program, $x^*$ be a feasible solution to $P$. Suppose $g_i$ and $h_j$ are differentiable at $x^*$, $x^*$ is a regular point. Then $x^*$ is an optimal solution to $P$ iff there exists $\lambda \in \mathbb{R}^r$, $\lambda \geq 0$ and $\mu \in \mathbb{R}^m$ satisfy

$$\lambda_i g_i(x^*) = 0, i = 1, \ldots, r$$

$$\nabla f(x^*) + \sum_{i=1}^{r} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{m} \mu_j \nabla h_j(x^*) = 0$$

Each equality constraint (1) can be replaced by two inequality constraints, standard programing (1) can be extended into program $P_1$.

$$P_1 : \begin{align*}
\min f(x) \\
\text{subject } g_i \leq 0, i = 1, \ldots, r + 2m
\end{align*}$$

Theorem 2 (Penalty function theorem [Luenberger]) Let $P_1$ be a extended program, $f, g_i \in C^1$, $\{x_k\}_i$ is a strictly increasing sequence. Define

$$L(s_k, x) = f(x) + \frac{s_k}{2} \sum_{i=1}^{r+2m} g_i^+(x)^2$$

where $g_i^+(x) = \max(0, g_i(x)) > 0$ is the magnitude of the violation of the ith constraint. Let the minimizer of $L(s_k, x)$ be $x_k$, then any limit point of the sequence $\{x_k\}_i$ is an optimal solution to $P_1$. (The true minimizer can only be obtained when the penalty parameter $s$ is infinite.) Furthermore, if $x_k \rightarrow x^*$ and $x^*$ is a regular point, then $s_k g_i^+(x_k) \rightarrow A$, which us the Lagrange multiplier associated with $g_i$. Given $\epsilon > 0$, there exists a sufficiently large $s_k$ such that minimizer of $L(s_k, x)$ lies in $N(\epsilon) = \{x ||x - x^*|| < \epsilon\}$. 

Engineering Intelligent Systems
Let us discuss some concepts of input-to-state stability (ISS). Consider following discrete-time nonlinear system

\[ x(k + 1) = f[x(k), u(k)], \quad y(k) = h[x(k)] \quad (3) \]

where \( u(k) \in \mathbb{R}^m \) is the input vector, \( x(k) \in \mathbb{R}^n \) is a state vector, and \( y(k) \in \mathbb{R}^l \) is the output vector. \( f \) and \( h \) are general nonlinear smooth functions. Let us now recall following definitions.

**Definition 9** A system (3) is said to be globally input-to-state stability if there exists a K-function \( \gamma(\cdot) \) (continuous and strictly increasing \( \gamma(0) = 0 \)) and KL-function \( \beta(\cdot) \) (K-function and \( \lim_{n \to \infty} \beta(s_k) = \infty \)) such that, for each \( \sup(||u(k)||) < \cdot \) and each initial state \( x_0 \in \mathbb{R}^n \), it holds that

\[ \|x(k, x_0, u(k))\| \leq \beta(||x_0||, k) + \gamma(||u(k)||) \]

**Definition 10** A smooth function \( V : \mathbb{R}^n \to \mathbb{R} \geq 0 \) is called a smooth ISS-Lyapunov function for system (3) if: (a) there exists a \( K_\infty \)-function \( \alpha_1(\cdot) \) and \( \alpha_2(\cdot) \) (K-function and \( \lim_{n \to \infty} \beta(s_k) = \infty \)) such that

\[ \alpha_1(s) \leq V(s) \leq \alpha_2, \quad \forall s \in \mathbb{R}^n \]

(b) There exist a \( K_\infty \)-function \( \alpha_3(\cdot) \) and a K-function \( \alpha_4(\cdot) \) such that

\[ V_{k+1} - V_k \leq -\alpha_3(||x(k)||) + \alpha_4(||u(k)||), \quad \text{for all} \quad x(k) \in \mathbb{R}^n, u(k) \in \mathbb{R}^m. \]

**Theorem 3** For a discrete-time nonlinear system, the following are equivalent [14]

a. It is input-to-state stability (ISS).

b. It is robustly stable.

c. It admit a smooth ISS-Lyapunov function.

**Property.** If a nonlinear system is input-to-state stable, the behavior of the system remains bounded when its inputs are bounded.

From (3) we have

\[ y(k) = h[x(k)] = F_1[x(k)], \]

\[ y(k + 1) = h[f[x(k), u(k)]] = F_2[x(k), u(k)] \]

\[ y(k + n - 1) = F_n[x(k), u(k), u(k + 1) \cdots u(k + n - 2)] \]

(4)

Denoting \( Y(k) = [y(k), y(k+1), \ldots, y(k+n-1)]^T, U(k) = [u(k), u(k+1), \ldots, u(k+n-2)]^T \), so \( Y(k) = F[x(k), U(k)] \), \( F = [F_1, F_2, \ldots, F_n] \). If \( \frac{\partial Y}{\partial u} \) is non-singular at \( x = 0, U = 0 \), (4) can be expressed as

\[ x(k + 1) = g(Y(k) + 1). \]

This leads to the NARMAX model [13]

\[ y(k) = h[x(k)] = \Phi[y(k - 1), y(k - 2), \ldots, u(k - 1), u(k - 2), \ldots] \]

\[ = \Phi[X(k)] \quad (5) \]

where

\[ X(k) = [y(k - 1), y(k - 2), \ldots, u(k - d), u(k - d - 1), \ldots]^T \]

\[ \Phi(\cdot) \] is an unknown nonlinear difference equation representing the plant dynamics, \( u(k) \) and \( y(k) \) are measurable scalar input and output, \( d \) is delay time. One can see that Definition 9, 10 and Theorem 3 do not depend on the exact expression of nonlinear systems. In this paper, we will apply ISS to the NARMAX model (5).

### 3. OPTIMIZING CRUDE OIL BLENDING WITH NEURAL NETWORKS AND BIAS UPDATE SCHEME

Crude oil blending can be considered a batch process where production is fixed by refinery production schedule. Crude oil blending process is subject to a number of operation restrictions, such as limits on the availability of blender feedstocks and product storage facilities. Further, the objectives of crude oil blending are to maximize the value of all products while their quality is satisfied, or minimize the cost of the blending. Crude oil blending control is a problem of optimization.

Crude oil blending can be considered a homogeneous mixture of \( n \) inlet components, with flow rates \( q = [q_1, \ldots, q_n]^T \) and property \( p = [p_1, \ldots, p_n]^T \), the flow rate and the property of the outlet mixture is defined as \( q_f \) and \( p_f \). By assuming negligible storage dynamic effects, the mixing rule is a nonlinear function

\[ p_f = f(q, p) \quad (7) \]

The blending control problem is a cost optimization on the manipulated variables or control input vector \( q \), the object is in the form of linear programming

\[ \min c^T q \quad (8) \]

where \( c \) is the cost parameter vector of the the feedstock. The constraints are as follows

1. Minimum and maximum availability

\[ q_{i, \min} \leq q_i \leq q_{i, \max} \quad (9) \]

2. Mass balance

\[ \sum_{i=1}^{n} q_i = q_f \quad (10) \]

3. Quality requirements

\[ p_{f, \min} \leq p_f \leq p_{f, \max} \quad (11) \]

A standard form of linear programming which can be solved by simplex method is [6]

\[
\begin{cases}
\min c^T q \\
\text{subject: } Aq \leq b, \quad Ae_q q = e_q, \quad q_{\min} \leq q \leq q_{\max}
\end{cases}
\]

We can see that (9) and (10) can be included in the standard form (12), but (11) cannot be applied in (12) directly, bins
update scheme [3] is used to deal with this problem. A basic assumption of bias update is the mixing rule (7) can be expressed as a known linear and a unknown nonlinear parts

\[ p_f = \frac{q^T p}{\sum_{i=1}^n q_i} + \Delta \]  \hspace{1cm} (13)

where \( p_f \) is measurable, \( p \) is constant vector, \( q \) will be decided by optimization. From control theory viewpoint, \( A \) can be estimated by the prior \( q \). From a constant initial condition for \( A \), for example \( A(0) = 0 \), \( A \) can be estimated after \( q \) is calculated by optimization

\[ \Delta(k+1) = p_f(k) - \frac{q^T(k) p}{\sum_{i=1}^n q_i} \]  \hspace{1cm} (14)

Now (8), (9), (10), (13) and (14) are transformed into the standard linear programming form (12),

\[
\begin{align*}
\text{min} & \quad c^T q(k) \\
\text{subject} & \quad Aq(k) \leq b, \quad Aeq q(k) = b_eq, \quad q_{min} \leq q(k) \leq q_{max}
\end{align*}
\]  \hspace{1cm} (15)

where \( A = \left[ \begin{array}{cc} p & -p \\ -p & p \end{array} \right], \quad b = \left[ \begin{array}{c} p_{f, max} - \Delta(k) \\ -p_{f, min} + \Delta(k) \end{array} \right], \quad A_eq = [1 \cdots 1], \quad b_eq = q_f.
\]

In many engineering and scientific applications, a fast convergence solutions of optimization problems are widely required. However traditional algorithms for computer, such as Optimization Toolbox of Matlab, may not be efficient since the computing time required for a solution is greatly dependent on the dimension and structure of the problems. One possible and very promising approach to fast optimization is to apply neural networks [5]. Traditional technique for solving optimization problem normally involve an iterative process. Consider following linear programming

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{subject} & \quad g(x) = Dx - b \leq 0
\end{align*}
\]  \hspace{1cm} (16)

where \( a, x \in R^n, \quad b \in R^m, \quad V_f = a, \quad V_g = D^T \). A so-called energy function is chosen as

\[ E(x) = f(x) + \frac{s}{2} \sum_{j=1}^m g_j^+(x)^2 \]

where \( g_j^+(x) = \max \{0, g_j(x)\}, \quad s > 0 \) is called penalty factor. Taking the derivative of \( E(x) \) with respect to time yields

\[ \frac{dE}{dt} = \sum_{j=1}^m \frac{\partial E}{\partial x_j} \frac{dx_j}{dt} = (V_f + s \sum_{j=1}^m g_j^+(x) \nabla g_j)^T \dot{x} \]

We define a neural network as

\[ C_i \dot{x} = -\nabla f_i - s_i \sum_{j=1}^m g_j^+(x) \nabla g_j \]

or in matrix form

\[ C \dot{x} = -\nabla f - s \nabla gg^+ \]

where \( C \) is a positive diagonal matrix (self-capacitance), so

\[ \dot{E} = \left[ -C \dot{x} \right]^T \dot{x} = -\nabla f \cdot \nabla f \leq 0 \]

\( E(x) \) is a Lyapunov function. This ensures that the system will converge to a stable equilibrium point without oscillation.

**Theorem 4** [8] Let linear programming (LP) (16) be a feasible program, \( x^* \) is a minimizer of the LP. \( E \) is defined as

\[ E(x) = f(x) + \frac{s}{2} \sum_{j=1}^m g_j^+(x)^2 \]

The set of equilibrium points \( \hat{x} \) of the neural network

\[ C \dot{x} = -\nabla f - s \nabla gg^+ \]

is the set of minimizers of \( E \). Given \( \epsilon > 0 \), there exists a sufficiently large \( s \) such that the neural networks is stable, and \( \dot{x} \) satisfies

\[ \min_{x \in \mathbb{R}^n} ||x^* - \hat{x}|| < \epsilon \]

This theorem ensures that the system will converge to a stable equilibrium point without oscillation. It requires the penalty function theorem (Kuhn-Tucker optimal theorem [9]) to hold. The relationship between the equilibrium point of neural networks and the true minimizer to the original program must be clarified, since there could be more than one equilibrium point with respect to one minimizer.

For crude oil blending optimization (12), we should transform it into the standard form (16). \( A_eq q(k) = b_eq \) and \( q_{min} \leq q(k) \leq q_{max} \) can be written as

\[ b_eq \leq A_eq q(k) \leq b_eq \]

\[ q_{min} \leq q(k) \leq q_{max} \]

So

\[ \begin{align*}
\text{min} & \quad c^T q(k) \\
\text{subject} & \quad g(q) = Dq - b \leq 0
\end{align*} \]  \hspace{1cm} (17)

with

\[ D = \left[ \begin{array}{ccc} p & -p & -A_eq & -A_eq \end{array} \right]^T, \quad b = \left[ \begin{array}{c} p_{f, max} - \Delta(k) \\ -p_{f, min} + \Delta(k) \\ b_eq - b_eq \cdot q_{max} - q_{min} \end{array} \right]^T, \quad g_j(x) = D_j x - b_j, \quad j = 1 \cdots m, \quad m = 2n + 4. \]

A recurrent neural network to solve crude oil blending optimization is as \( C = I \)

\[ \hat{q}_i = -c_i - s_i \sum_{j=1}^m d_{i,j} \gamma [g_j(q)] \]  \hspace{1cm} (18)

where \( i = 1 \cdots n, \quad \gamma [x] = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}, \quad s_i > 0. \)

Since \( D \) and \( b \) are in discrete-time, following difference technique is used to get the discrete-time states of the continuous time neural networks (1),

\[ q = f'(q) \]  \hspace{1cm} (19)
with $q = [q_1 \cdots q_n]^T$, $f = [f_1 \cdots f_n]^T$, $f_i(q) = -c_i - s_i \sum_j d_{ij} g_j(q_j)$. Let us define $s_1 = f(q_k)$, $s_2 = f(q_k + s_1)$, $s_3 = f(q_k + s_1 + s_2)$. If $\sum_{i=1}^{2n} s_i \leq 1$, then $q_{k+1} = q_k + s_1 + s_2 + s_3, k = 0, 1, 2, \ldots$. The scheme diagram for crude oil blending with neural networks and bias update is shown in Fig. 1.

4. OFF-LINE OPTIMIZATION OF CRUDE OIL BLENDING WITH NEURAL NETWORKS

The optimization approach proposed in Section III can only be applied on-line, because the bias update requires the outlet property $p_f$ which is obtained from an on-line analyzer (see Fig. 1). Sometimes we need off-line optimization, for example we hope to find a reference optimal inlet flow rate based on history data, this control is useful for decision and supervision control. In this section we will give an answer on how to design a optimization control from history data.

Neural networks modeling is in sense of "black-box", it is suitable to model crude oil blending process via input/output data in the database. The static properties of crude oil blending can be written in following general form

$$y(k) = \Phi [u_1(k), u_2(k), \ldots u_n(k), p] = \Phi [X(k)]$$

where

$$X(k) = [u_1(k), u_2(k), \ldots u_n(k), p]^T$$

$\Phi(\cdot)$ is an unknown nonlinear difference equation representing the blending operation, $u_i(k)$ is flow rates $q_i$, $p$ is the property of feedstock $p = [p_1 \cdots p_n]^T$. We consider multilayer neural network (or multilayer perceptrons, see Fig. the part of MLP) to model the static operation of the blending (5)

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

where the scalar output $\hat{y}(k)$ and vector input $X(k) \in \mathbb{R}^{2n \times 1}$ is defined in (6), the weights in output layer are $V_k \in \mathbb{R}^{1 \times m}$, the weights in hidden layer are $W_k \in \mathbb{R}^{m \times 2n}$, $\phi$ is a dimension vector function.

The typical presentation of the element $\phi(\cdot)$ is sigmoid function. The identified blending system (\cdot) can be represented be written 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 (5) may be also expressed as

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

where $V^0$ is an known matrix chosen by the user. 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

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

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

$$= V_k \phi [W_k X(k)] + V^0 \phi [W_k X(k)] + \phi' \psi(k)$$

where $\phi'$ is the derivative of nonlinear activation function $\phi(\cdot)$ at the point of $W_k X(k)$. $\psi_k = W_k - W^*$, $\vec{V}_k = V_k -$
Theorem 5 If we use the multilayer neural network (20) to identify nonlinear process (5), the following backpropagation-like algorithm can make identification error \( e(k) \) bounded

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

(21)

where \( \eta_k = \frac{\sigma}{1 + \|\phi' V^0 X^T (k)\|^2 + \|\phi\|^2} \), \( 0 < \eta \leq 1 \). The average of the identification error satisfies

\[
J = \lim sup_{T \to \infty} T
\]

(22)

where \( \pi = \frac{\kappa}{1 + \xi} \left[ 1 - \frac{\kappa}{1 + \xi} \right] > 0 \), \( \kappa = \max_k \), \( \xi = \max(\xi^2(k)) \).

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

\[
L_k = \left\| \tilde{W}_k \right\|^2 + \left\| \tilde{V}_k \right\|^2
\]

(23)

From the updating law (21), we have

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

Since \( \phi' \) is diagonal matrix, and by using (23) we have

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

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

\[
\times \left\| V^0 \phi \tilde{W}_k X(k) + \tilde{V}_k \phi \right\|
\]

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

\[
\times \left\| e(k) \left( \phi(k) - \xi(k) \right) \right\|
\]

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

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

(24)

where \( \pi \) is defined in (22). Because

\[
\eta_k \left[ \min (\tilde{W}_k^2) + \min (\tilde{V}_k^2) \right] \leq L_k
\]

\[
\leq \eta_k \left[ \max (\tilde{W}_k^2) + \max (\tilde{V}_k^2) \right]
\]

where \( \eta_k \left[ \min (\tilde{W}_k^2) + \min (\tilde{V}_k^2) \right] \) and \( \eta_k \left[ \max (\tilde{W}_k^2) + \max (\tilde{V}_k^2) \right] \) are \( K \) functions, and \( e^2(k) \) is an \( K \) function. From (23) and (23) we know \( V_k \) is the function of \( e(k) \) and \( \xi(k) \), so \( L_k \) admits a smooth ISS-Lyapunov function, the dynamic of the identification error is input-to-state stable. Because the INPUT \( \xi(k) \) is bounded and the dynamic is ISS, the STATE \( e(k) \) is bounded.

Summing (25) from 1 to \( T \), and by using \( L_T > 0 \) and \( L_1 \) is a constant, we obtain

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

(25)

Remark 1. \( V^0 \) does not effect the stability property of the retrained identification, but it influences the identification accuracy, see (22). We design an off-line method to find a better value for \( V^0 \). If we let \( V^0 = V_0 \), the algorithm (21) 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 \( V^0 = V_0 \).

2) Do identification with this \( V^0 \) until \( T_0 \).

3) If the \( \| e(0) \| \leq \| e(0) \| \), let \( V_T \) as a new \( V^0 \), i.e., \( V^0 = V_T \), go to 2 to repeat the identification process.

4) If the \( \| e(T_0) \| \leq \| e(0) \| \), stop this off-line identification, now \( V_{T_0} \) is the final value for \( V^0 \).

With this prior knowledge \( V^0 \), we may start the identification (21).

The learning law (21) can assure the neural networks (20) have similar properties as the history data in the database. So we can replay the crude oil blending plant and the analyzers with the neural networks (20), and realize off-line optimization. The scheme diagram of off-line optimization of crude oil blending with neural networks is shown in Fig.3.

The basic assumption of real-time analyzer for bias update can be avoided, after learning the output of neural networks (20) is

\[
\hat{p}_f = V_k \phi' [W_k X(k)]
\]

(26)

From the quality requirements \( p_{\min} \leq p_f \leq p_{\max} \)

\[
p_{\min} \leq V_k \phi' [W_k X(k)] \leq p_{\max}
\]

(8), (9), (10), (13) and (14) can be written as standard linear programming form (12).

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

(14)

subject: \( A_{eq} q (k) = b_{eq}, \quad q_{\min} \leq q (k) \leq q_{\max} \)
Figure 3 Diagram of off-line optimization of crude oil blending with neural networks

\[ b = \begin{bmatrix} p_{f, \text{max}} - A(k) \\ -p_{f, \text{min}} + \Delta(k) \end{bmatrix}, \quad A_{eq} = [1 \ldots 1], \]
\[ b_{eq} = q_f. \]

5. CASE STUDY

The case study uses a real crude oil blending process in Tabasco of Mexico. One kind of crude oil blending is called Terminal Maritima de dos Bocas Tabasco (TMDB). It has three serial blending points, each point has two components for blending operation. The whole blending process is shown in Fig.4.

\( q_s \) and \( q_e \) are constant flow rate. A normal optimization operation is as follows. First, we design a simple model for this process.

The mass balance are as follows,
\[ q_a = q_1 + q_2 - q_b, \quad \text{or } x_1 + x_2 = 1 \]
\[ q_b = q_a + q_3 - q_c, \quad \text{or } x_0 + x_3 = 1 \]
\[ q_f = q_b + q_4, \quad \text{or } x_b + x_4 = 1 \]

where
\[ x_a = \frac{q_a}{q_3 + q_a}, \quad x_3 = \frac{q_3}{q_3 + q_a}, \]
\[ x_b = \frac{q_b}{q_4 + q_b}, \quad x_4 = \frac{q_4}{q_4 + q_b} \]

The mixing rule is given by an interaction model \([1][11][12]\) which is a linear function plus a nonlinear function
\[ p = \sum_{i=1}^{2} p_i^* x_i + a \prod_{i=1}^{2} x_i \]

where \( a \) is a mixing rule coefficient.
\[ p_f = \frac{1}{q_f} (p_4 q_4 + p_6 q_6) + \alpha_3 x_3 x_b \]
\[ = \frac{1}{q_f} p_4 q_4 + \frac{1}{q_f} \left( \frac{q_6}{q_6} (p_3 q_3 + p_6 q_6) + \alpha_2 x_3 x_a \right) \times (1 - x_4) + \alpha_3 x_4 (1 - x_4) \]
\[ = \frac{1}{q_f} p_4 q_4 + \frac{1}{q_f} \left[ p_3 x_3 + \left( \frac{q_6}{q_6} (p_2 q_2 + p_1 q_1) + \alpha_1 x_1 x_2 \right) \times (1 - x_3) + \alpha_2 x_3 (1 - x_3) \right] \times (1 - x_4) + \alpha_3 x_4 (1 - x_4) \]
\[ = \frac{1}{q_f} (p_1 q_1 + p_2 q_2 + p_3 q_3 + p_4 q_4) + \frac{1}{q_f} \left[ x_3 x_2 - p_1 x_1 x_2 - p_1 x_1 x_4 - p_3 x_3 x_3 - p_3 x_3 x_4 - p_1 x_1 x_3 + x_1 x_3 x_2 + x_1 x_3 x_4 + x_1 x_3 x_4 x_2 - x_3 x_2 x_2 x_2 - x_3 x_4 x_4 x_4 \right] \]

where \( q_f = q_1 + q_2 + q_3 + q_4 - q_b - q_e \). The optimization is to find \( q = [q_1, q_2, q_3, q_4]^T \). The parameters are selected as
\[ q_{\text{max}} = 2 \times [10^5, 10^5, 10^5, 10^5]^T, \]
\[ q_{\text{min}} = [10^3, 10^3, 10^3, 10^3]^T, \quad q_b = q_e = 10^2 \]
\[ p = [32, 33.8, 31.8, 33.15]^T, \quad \alpha = [\alpha_1, \alpha_2, \alpha_3]^T \]
\[ = [-1.29, -1.13, -1.31]^T \]
\[ q_f = 1.8 \times 10^5, \quad p_f_{\text{min}} = 32.2, \]
\[ p_{f, \text{max}} = 33, \quad c = kp, \quad k = 0.78. \]

This crude oil blending is used to compare the performance of three blend optimization

1) the conventional linear programming with bias update approach;
The conventional linear programming with bias update approach is

\[
\begin{align*}
\text{min} & \quad c^T q & \\
\text{subject} & \quad Aq \leq b, & A_{eq}q = b_{eq}, & q_{\min} \leq q \leq q_{\max}
\end{align*}
\]

(28)

where \( A_{eq} = [1, 1, 1]^T \), \( b_{eq} = q_f + q_t + q_e \), \( A = \left[ \begin{array}{c} p/q_f \\ -p/q_f \end{array} \right], b = \left[ \begin{array}{c} p_f,\max - \Delta(k) \\ -p_f,\min + \Delta(k) \end{array} \right] \), \( \Delta(k) = p_f(k) - q_f^2/q_f \), \( A(0) = 0 \). Start from \( q(0) = [2 \times 10^4, 3 \times 10^4, 4 \times 10^4, 9 \times 10^4]^T, p_f \) is calculated by (27). The response of simplex with bias update approach is shown in Fig. 5, where at \( k = 15 \) \( p_f,\max \) is changed to 32.8.

If we use another model, Zahed model [16], for this crude oil blending

\[ p_f = m_0 + \frac{1}{q_f} \left( p_f^{1.25} q_1 + p_f^{1.25} q_2 + p_f^{1.25} q_3 + p_f^{1.25} q_4 \right)^{0.8} \]

(29)

where \( m_0 = 47 \). The simplex with bias update is not stable with the same parameters, see Fig. 6. For real crude oil blending, neither (27) nor (29) can model the process. The simplex with bias update has robustness problem [1], because the approach is dependent on the part linear model. For Zahed model, we have to find a suitable \( A \) such that simplex is stable.

On the other hand, neural networks for optimization use only input/output data, the robustness of bias update can be improved. We use recurrent neural networks (18) to realize optimization calculating. First, (28) is transformed into (17) with \( n = 4 \), \( m = 2n + 4 = 12 \). We use (19) to calculate the optimization. Fig. 7 and Fig. 8 give stable optimization via neural networks for interaction model Zahed model.

The second part of this section is to study off-line optimization via neural networks. In order to training neural networks effectively, we use fraction model instead of flow rate model (15), because fraction is in the section in [0, 1] but flow rate is in the section in \([10^4, 10^5]\).

No matter interaction model or Zahed model, they can be expressed in a standard form of

\[ p_f(k) = y(k) = \Phi[X(k)] \]

(30)

The neural network is select as

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

(31)

where \( W_k \in \mathbb{R}^{9 \times 4}, V_k \in \mathbb{R}^{1 \times 9} \), the initial conditions for the elements of \( \Phi^{\text{out}} \), \( W_k \) and \( V_k \) are random number in \([0, 1]\).

The learning algorithm is

\[ W_{k+1} = W_k - \eta_k e(k) \phi^{\text{out}} X^T(k) \]
\[ V_{k+1} = V_k - \eta_k e(k) \phi^T \]

Figure 4: TMDB crude oil blending process.

Figure 5: Response of simplex with bias update approach.

Figure 6: Unstable for simplex with update bias.
Figure 7 Recurrent neural network to solve optimization (interactive model).

Figure 8 Recurrent neural network to solve optimization (Zahed model).

Figure 9 Static neural network models crude oil blending (interactive model).

Figure 10 Static neural network models crude oil blending (Zahed model).

where $\eta_k = \frac{1}{1+||\phi^\top X^T(k)||^2+||\phi||^2}$, $\phi(k) = \hat{y}(k) - y(k)$,

$\phi(\cdot) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, 

$\phi'(\cdot) = \sec h(x) = \frac{2}{e^x + e^{-x}}$

We use 1500 data to training it, after $k > 1300$, the weights are converged. Then we use different flow rates of the feed stocks to test our neural model. We select

$x_1(k) = 0.1 \left[1 + \sin \left(\frac{3\pi}{6} k\right)\right]$

$x_2(k) = 0.2 \left[1 + \cos \left(\frac{2\pi}{6} k\right)\right]$

$x_3(k) = 0.1 \left[1 + \sin \left(\frac{4\pi}{6} k\right)\right]$

$x_4(k) = 0.1 \left[1 + \sin \left(\frac{3\pi}{6} k\right)\right]$

$x_5(k) = 1 - \left[x_1(k) + x_2(k) + x_3(k) + x_4(k)\right]$

These data are put into the blending system (30) and blending model (31) at same time, the octane number of gasoline blending is shown The training results and testing results of interaction model or Zahed model are shown in Fig.9 and Fig.10. Let us define the mean squared error for finite time is

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

where $N$ is simulation time. In the training phase $J_1(1500) = 0.0058$. in the testing phase $J_1(1500) = 0.0087$. Modeling errors depend on the complexity of the particular model selected and how close it is to the actual plant. In this example the modeling is bigger in testing phase than training phase. The worse results due to the neural networks cannot match the plant exactly. From the point of identification, it is because the model is not close to the plant. We should mention that model structure influences modeling error, but does not destroy stability of identification process.

A fraction optimization model for (15) is

$\min \, c^T x(k)$

subject: $A x(k) \leq b$, $A_{eq} x(k) = b_{eq}$, $x_{min} \leq x(k) \leq x_{max}$

where $x_1(k) = q(k)$, $A_{eq} = [1 \ldots 1]$, $b_{eq} = 1$. Neural networks based off-line optimization results are shown in
The main contributions of this paper are some applications of neural networks on optimization are proposed. We use a recurrent neural networks to solve the linear programming with bias update. Then we use a static neural networks to modeling crude oil blending process based on the history data. Input-to-state stability approach is applied to access robust learning algorithms of the neural networks. So we can overcome the main drawbacks of optimizing crude oil blending, which are the simplex-base optimization is not robust and the optimal control can only be realized on-line.

6. CONCLUSION

The main contributions of this paper are some applications of neural networks on optimization are proposed. We use a recurrent neural networks to solve the linear programming with bias update. Then we use a static neural networks to modeling crude oil blending process based on the history data. Input-to-state stability approach is applied to access robust learning algorithms of the neural networks. So we can overcome the main drawbacks of optimizing crude oil blending, which are the simplex-base optimization is not robust and the optimal control can only be realized on-line.

REFERENCES

15. W. Yu, X. Li, Fuzzy Neural Identification by Online Clustering with Application on Crude Oil Blending, IEEE International Conference on Fuzzy Systems, Pages(s):269-276, 2006