2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Two Neural Network Approaches to Model Predictive Control

Yunpeng Pan and Jun Wang

Abstract— **Model predictive control (MPC) is a powerful technique for optimizing the performance of control systems. However, the high computational demand in solving optimization problem associated with MPC in real-time is a major obstacle. Recurrent neural networks have various advantages in solving optimization problems. In this paper, we apply two recurrent neural network models for MPC based on linear and quadratic programming formulations. Both neural networks have good convergence performance and low computational complexity. A numerical example is provided to illustrate the effectiveness and efficiency of the proposed methods and show the different control behaviors of the two neural network approaches.**

I. Introduction

Model predictive control (MPC), which is more advanced than the well-known PID-control, has achieved great success in practical applications in recent decades. One of the key advantages of MPC is its ability to deal with input and output constraints; another is that MPC can be naturally applied for multivariable process control. Because of these advantages, MPC has been used in numerous industrial applications in the refining, petrochemical, chemical, pulp, paper, and food processing industries. Academic interests in MPC started growing in the late seventies, several publications provide a good introduction to theoretical and practical issues associated with MPC technology [1]-[3].

Most control techniques do not consider the future implication of current control actions. MPC applies on-line optimization to a system model. By taking the current state as an initial state, a cost-minimization control strategy is computed at each sample time, and at the next computation time interval, the calculation repeated with a new state. The basic structure of MPC is shown in Fig.1. As the process model of MPC is usually expressed with linear or quadratic criterion, MPC problems can be generally formulated as linear programming or quadratic programming problems. As a result, they can be solved using solution methods for linear and quadratic programming problems.

Over years, a variety of numerical methods have been developed for solving linear and quadratic programming problems. Compared with traditional numerical methods for constrained optimization, neural networks have several advantages: first, they can solve many optimization problems with time-varying parameters; second, they can handle large-scale problems with their parallelizable ability; third, they can be implemented effectively using VLSI or optical technologies [4]. Therefore, neural networks can solve optimal control problems in running times at the orders of magnitude much faster than the most popular optimization algorithms executed on general-purpose digital computers. Application areas of neural networks include, but are not limited to, system modeling, mathematical programming, associative memory, combinatorial optimization, pattern recognition and classification, robotic and process control.

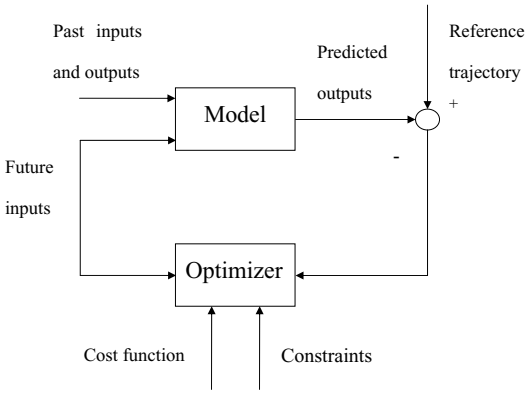

Fig. 1. Basic structure of MPC.

In the past two decades, recurrent neural networks for optimization and their engineering applications have been widely investigated. Tank and Hopfield proposed the first working recurrent neural network implemented on analog circuits [5], their work inspired many researchers to develop other neural networks for solving linear and nonlinear optimization problems. By the dual and projection methods, Hu, Liu, Xia, and Wang developed several neural networks for solving general linear and quadratic programming problems. These neural networks have shown good performance in convergence.

In this paper, we propose two neural network approaches to the design of MPC by applying two recurrent neural networks [6] and [7]. Both neural networks have desired convergence property and relatively lower computational complexity. A numerical example shows that the proposed approaches are effective and efficient in MPC controller design. Furthermore, a comparison was made between the two neural network approaches.

The rest of this paper is organized as follows. In Section II, we derive both linear and quadratic formulations for MPC controller design. In Section III, we present two recurrent neural network approaches to MPC based on linear and quadratic programming. In Section IV, we provide a numer-

The authors are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong (email: {yppan, jwang}@mae.cuhk.edu.hk)

ical example to illustrate the performance of the proposed approaches. Finally, Section V concludes this paper.

II. Problem Formulation

Consider the following linear discrete-time system:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \tag{1}$$

with the constraints

$$\begin{aligned} u_{\min} \leq & u(k) \leq u_{\max}, \\ \Delta u_{\min} \leq & \Delta u(k) \leq \Delta u_{\max}, \\ y_{\min} \leq & y(k) \leq y_{\max}, \end{aligned} \tag{2}$$

which represents the dynamics of the plant under consideration. In (1)-(2), $k \geq 0$, $x(k) \in \Re^n$ is the state vector, $u(k) \in \Re^m$ is the input vector, and $y(k) \in \Re^p$ is the output vector. $u_{\min} \leq u_{\max}$ and $y_{\min} \leq y_{\max}$ are vectors of upper and lower bounds.

Model predictive control is a step-by-step optimization technique: at each sampling time, measure of estimate the current state, obtain the optimal input vector by solving the following optimization problem is solved at each time k.

A. Quadratic Programming Formulation

With a quadratic criterion, MPC can be formulated as the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{j=1}^{N} [r(k+j|k) - y(k+j|k)]^T Q [r(k+j|k) - \\ & y(k+j|k)] + \sum_{j=0}^{N_u-1} \Delta u(k+j|k)^T R \Delta u(k+j|k) \\ \text{s.t.} \quad & u_{\min} \leq u(k+j|k) \leq u_{\max}, \quad j = 0, ..., N_u - 1; \\ & \Delta\, u_{\min} \leq \Delta u(k+j|k) \leq \Delta u_{\max}, \quad j = 0, ..., N_u - 1; \\ & y_{\min} \leq y(k+j|k) \leq y_{\max}, \quad j = 1, ..., N; \end{aligned} \tag{3}$$

where k is the current time step, $y(k+j|k)$ denotes the predicted output, $r(k+j|k)$ denotes the reference trajectory of output signal (desired output) at sampling instant k, and $\Delta u(k+j|k)$ denotes the input increment, where $\Delta u(k+j|k) = u(k+j|k) - u(k-1+j|k)$. $Q \in \Re^{p \times p}$ and $R \in \Re^{m \times m}$ are appropriate weighting matrices. N is the predictive horizon ($1 \leq N$). N_u denote the control horizon ($0 < N_u \leq N$). After N_u control moves, $\Delta u(k+j|k)$ becomes zero.

According to the process model (1):

$$y(k+j|k) = CA^j x(k) + C \sum_{i=0}^{j-1} A^i B u(k+j-i-1|k), \quad j = 1, ..., N. \tag{4}$$

Define following vectors:

$$\begin{aligned} \bar{y}(k) &= [y(k+1|k) \quad \cdots \quad y(k+N|k)]^T \in \Re^{Np}, \\ \bar{u}(k) &= [u(k|k) \quad \cdots \quad u(k+N_u-1|k)]^T \in \Re^{N_u m}, \\ \Delta\bar{u}(k) &= [\Delta u(k|k) \quad \cdots \quad \Delta u(k+N_u-1|k)]^T \in \Re^{N_u m}, \\ \bar{r}(k) &= [r(k+1|k) \quad \cdots \quad r(k+N|k)]^T \in \Re^{Np}, \end{aligned}$$

where the reference trajectory $\bar{r}(k)$ is known in advance. The predicted output $\bar{y}(k)$ are expressed in the following form:

$$\begin{aligned} \bar{y}(k) &= Sx(k) + M\bar{u}(k) \\ &= Sx(k) + M\Delta\bar{u}(k) + Vu(k-1), \end{aligned} \tag{5}$$

where

$$S = [CA \quad CA^2 \quad \cdots \quad CA^N]^T \in \Re^{Np \times n},$$

$$V = \begin{bmatrix} CB \\ C(A+I)B \\ \vdots \\ C(A^{N_u-1} + \cdots + A + I)B \\ C(A^{N_u} + \cdots + A + I)B \\ \vdots \\ C(A^{N-1} + \cdots + A + I)B \end{bmatrix} \in \Re^{Np \times m},$$

$$M = \begin{bmatrix} CB & \cdots & 0 \\ C(A+I)B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ C(A^{N_u-1} + ...I)B & \cdots & CB \\ C(A^{N_u} + ...I)B & \cdots & C(A+I)B \\ \vdots & \ddots & \vdots \\ C(A^{N-1} + ...I)B & \cdots & C(A^{N-N_u} + ...I)B \end{bmatrix},$$

$M \in \Re^{Np \times N_u m}$, I denotes the identity matrix. Define vectors:

$$\begin{aligned} \Delta\bar{u}_{\max} &= [\Delta u_{\max} \cdots \Delta u_{\max}]^T \in \Re^{N_u m}, \\ \bar{u}_{\min} &= [u_{\min} \cdots u_{\min}]^T \in \Re^{N_u m}, \\ \bar{u}_{\max} &= [u_{\min} \cdots u_{\max}]^T \in \Re^{N_u m}, \\ H &= \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & I & I \end{bmatrix} \in \Re^{N_u m \times N_u m}. \end{aligned}$$

Thus, the optimization problem (3) can be expressed in the following form:

$$\begin{aligned} \min \quad & [\bar{r}(k) - Sx(k) - M\Delta\bar{u}(k) - Vu(k-1)]^T Q [\bar{r}(k) - \\ & Sx(k) - M\Delta\bar{u}(k) - Vu(k-1)] + \Delta\bar{u}^T(k) R \Delta\bar{u}(k) \\ \text{s.t.} \quad & \bar{u}_{\min} \leq \bar{u}(k) + H\Delta\bar{u}(k) \leq \bar{u}_{\max} \\ & \Delta\bar{u}_{\min} \leq \Delta\bar{u}(k) \leq \Delta\bar{u}_{\max} \\ & \bar{y}_{\min} \leq \bar{y}(k) + M(k)\Delta\bar{u}(k) \leq \bar{y}_{\max} \end{aligned} \tag{6}$$

By defining the variable vector $v = \Delta\bar{u}(k) \in \Re^{N_u m}$, the problem (3) can be rewritten as a standard quadratic programming problem form:

$$\begin{aligned} \min \quad & \frac{1}{2} v^T W v + c^T v \\ \text{s.t.} \quad & l_{\min} \leq Gv \leq l_{\max} \end{aligned} \tag{7}$$

where the coefficient matrices and vectors are

$$l_{\min} = (-\infty \quad \Delta\bar{u}_{\min}]^T \in \Re^{3N_u m+2Np},$$
$$l_{\max} = [b \quad \Delta\bar{u}_{\max}]^T \in \Re^{3N_u m+2Np},$$
$$W = 2(M^T Q M + R) \in \Re^{N_u m \times N_u m},$$
$$c = -2M^T Q(\bar{r}(k) - Sx(k) - Vu(k-1)) \in \Re^{N_u m},$$
$$E = [-H \quad H \quad -M \quad M]^T \in \Re^{(2N_u m+2Np)\times N_u m},$$
$$G = [E \quad I]^T \in \Re^{(3N_u m+2Np)\times N_u m}$$
$$b = \begin{bmatrix} -\bar{u}_{\min} + \bar{u}(k) \\ \bar{u}_{\max} - \bar{u}(k) \\ -\bar{y}_{\min} + \bar{y}(k) \\ \bar{y}_{\max} - \bar{y}(k) \end{bmatrix} \in \Re^{2N_u m+2Np}.$$

The solution to the quadratic programming problem (7) gives the vector of control action $\Delta\bar{u}(k)$, which is used to calculate the optimal input signal. Since the objective function in is strictly convex (due to W being positive definite), and the feasible region of linear constraints is a closed convex set, the solution to the quadratic programming problem is unique and satisfies the Karush-Kuhn-Tucker (KKT) optimality conditions [8].

B. Linear Programming Formulation

Although the quadratic criterion is popular and has been widely used in various MPC applications. Several MPC algorithms using linear programming have been presented. For example, Zadeh and Whalen [9] and Propoi [10] introduce the approaches to solve MPC problem based on linear programming in the early sixties. And some other authors published their investigation concerning the linear programming based MPC [11]-[14].

In this section, we formulate MPC as a standard linear programming problem. With l_1 criterion, MPC can be formulated other than (3) as follow:

$$\begin{aligned} \min \sum_{j=1}^{N} Q[r(k+j|k) - y(k+j|k)] + \sum_{j=0}^{N_u-1} R\Delta u(k+j|k) \\ \text{s.t.} \quad u_{\min} \le u(k+j|k) \le u_{\max}, \quad j = 0, ..., N_u - 1; \\ \Delta\ u_{\min} \le \Delta u(k+j|k) \le \Delta u_{\max}, \quad j = 0, ..., N_u - 1; \\ y_{\min} \le y(k+j|k) \le y_{\max}, \quad j = 1, ..., N. \end{aligned} \tag{8}$$

The optimization problem (8) can be further formulated as a standard linear programming problem using the standard method [15].

Define the following vectors:

$$\phi(k) = [\phi, \cdots, \phi]^T \in \Re^{Np},$$
$$\varphi(k) = [\varphi, \cdots, \varphi]^T \in \Re^{N_u m},$$
$$s = [\Delta\bar{u}^T(k) \quad \phi^T(k) \quad \varphi^T(k)]^T \in \Re^{2N_u m+Np},$$

that satisfies

$$\begin{aligned} -\phi(k) \le \pm Q[\bar{r}(k) - \bar{y}(k)], \\ -\varphi(k) \le \pm R\Delta\bar{u}(k), \end{aligned} \tag{9}$$

where $\pm$ means that the constraints is duplicated for each sign. The problem (8) can be rewritten in as a standard linear programming problem:

$$\begin{aligned} \min \quad & f^T s \\ \text{s.t.} \quad & h_{\min} \le Fs \le h_{\max} \end{aligned} \tag{10}$$

where the coefficient matrices and vectors are:

$$h_{\min} = [l_{\min} \quad Q(\bar{r}(k) - Sx(k) - Vu(k-1)) \quad O \quad -\infty]^T,$$
$$h_{\max} = [l_{\max} \quad \infty \quad Q(\bar{r}(k) - Sx(k) - Vu(k-1)) \quad O]^T,$$
$$h_{\min}, h_{\min} \in \Re^{(3N_u+2N)m+2Np},$$
$$f = [0, \cdots, 0, 1, \cdots, 1]^T \in \Re^{2N_u m+Np},$$
$$F = \begin{bmatrix} G & O & O \\ QM & I & O \\ R & O & I \\ QM & -I & O \\ R & O & -I \end{bmatrix},$$
$$F \in \Re^{[(3N_u+2N)m+2Np]\times(2N_u m+Np)},$$

where O denotes the zero matrix.

As the linear programming problem (10) depends on the current state $x(k)$ and past input $u(k-1)$, we will introduce a neural network to solve the problem at each time interval in the next section.

III. Neural Network Approaches

In this section, based on the linear and quadratic programming formulations (7) and (10) in the previous section, we propose two neural network approaches to MPC.

A. Neural Network Model 1

Over years, various neural network models have been developed for solving quadratic programming problems. Including the penalty-based neural network [16], the Lagrange neural network [17],the deterministic annealing network [18], the primal-dual neural network [19], the dual neural networks [20]-[22], and the recurrent neural network with a discontinuous hard-limiting activation function [23].

In particular, Liu and Wang [6] developed a one-layer recurrent neural network called the simplified dual neural network for solving quadratic programming problems, which has showed good performance and lower computational complexity. In this part of the section, we apply the neural network for MPC.

Consider (7) as a primal problem, then its dual problem is:

$$\begin{aligned} \min \quad & -\frac{1}{2} v^T W v + l_{\min}^T \alpha - l_{\max}^T \beta \\ \text{s.t.} \quad & Wv + c - G^T\alpha + G^T\beta = 0 \end{aligned} \tag{11}$$

where $\alpha \in \Re^{2N_u m+2Np}$, $\beta \in \Re^{2N_u m+2Np}$ are dual decision variables.

By defining $z = \alpha - \beta$, the Karush-Kauhn-Tucker condition of (7) are:

$$\begin{aligned} &Wv + c - G^T z = 0, \\ &Gv = \lambda(Gv - z), \end{aligned} \tag{12}$$

where $\lambda(\cdot)$ is a piecewise linear function, defined as:

$$\lambda(\varepsilon_i) = \begin{cases} l_{\min}, & \varepsilon_i < l_{\min}; \\ \varepsilon_i, & l_{\min} \le \varepsilon_i \le l_{\max}; \\ l_{\max}, & \varepsilon_i > l_{\max}. \end{cases} \tag{13}$$

Based on (12) (13), the dynamic equation of neural network for solving quadratic programming problem (7) can be described as:

- state equation

$$\epsilon \frac{dz}{dt} = -Gv + \lambda(Gv - z), \tag{14}$$

- output equation

$$v = W^{-1}(G^T z - c), \tag{15}$$

where $z \in \Re^{2N_u m + 2Np}$ is the state vector, ϵ is a scaling parameter that control the convergence rate of the neural network.

According to the convergence analysis in [6], we can ensure that the proposed neural network is Lyapunov stable and globally convergent to the optimal solution.

B. Neural Network Model 2

It has been shown that linear programming problems can also be solved using neural networks [24]-[26]. Recently, a one-layer recurrent neural network with a discontinuous hard-limiting activation function for linear programming was developed [7]. In this paper, we present this neural network for solving (10).

To apply the neural network model, let us further formulate (10) as:

$$\begin{aligned} \min \quad & f^T s \\ \text{s.t.} \quad & b_{\min} \le s \le b_{\max} \end{aligned} \tag{16}$$

where

$$\begin{aligned} b_{\min} &= F^T h_{\min} \in \Re^{N_u(m+1)+N}, \\ b_{\max} &= F^T h_{\max} \in \Re^{N_u(m+1)+N}. \end{aligned}$$

According to the KKT conditions, s^* is an optimal solution of (16), if and only if there exist a $w^* \in \Re^{2N_u m + Np}$ such that (s^*, w^*) satisfies the following optimality conditions:

$$\begin{aligned} &f + w = 0, \\ &\begin{cases} w_i \ge 0, & s_i = b_{\max(i)}; \\ w_i = 0, & b_{\min(i)} \le s_i \le b_{\max(i)}; \\ w_i \le 0, & s_i = b_{\max(i)}. \end{cases} \end{aligned} \tag{17}$$

Based on the above conditions, the dynamic equation of the proposed recurrent neural network model is described as follows:

$$\epsilon \frac{ds}{dt} = -\sigma g(s) - f, \tag{18}$$

where ϵ is a positive scaling constant, σ is a nonnegative gain parameter. $g(\cdot)$ is a discontinuous activation function, defined as:

$$g_i(s_i) = \begin{cases} 1, & s_i > b_{\max(i)}; \\ [0,1], & s_i = b_{\max(i)}; \\ 0, & b_{\min(i)} < s_i < b_{\max(i)}; \\ [-1,0], & s_i = b_{\min(i)}; \\ -1, & s_i < b_{\min(i)}, \end{cases} \tag{19}$$

where $i = 1, 2, ..., 2N_u m + Np$.

It is proven in [7] that the neural network is globally convergent to the optimal solution.

C. Control Scheme

The control scheme based on neural networks can be summarized as follows:

1. Let $k = 1$. Set terminal time T, sample time t, predictive horizon N, control horizon N_u, weighting matrices Q and R.
2. Calculate process model matrices S, V, M, neural network parameters W, G, f, $l_{\max}$, $l_{\min}$, $b_{\max}$, $b_{\min}$.
3. Solve the quadratic and linear programming problems (7) and (10) using the proposed two neural networks, obtaining the optimal control action $\Delta\bar{u}(k)$.
4. Calculate the optimal input vector $u(k) = \Delta u(k|k) + u(k-1)$.
5. If $k < T$, set $k = k + 1$, return to step 2; otherwise, end.

IV. Numerical Example

Consider a quadruple-tank process described in [27], the objective is to control the level of the two lower tanks y_1 and y_2, using the two pumps u_1 and u_2. The process model of the quadruple-tank system is

$$\begin{aligned} \dot{x}_1 =& -\frac{a_1}{A_1}\sqrt{2gx_1} + \frac{a_3}{A_1}\sqrt{2gx_3} + \frac{\gamma_1\rho_1}{A_1}u_1, \\ \dot{x}_2 =& -\frac{a_2}{A_2}\sqrt{2gx_2} + \frac{a_4}{A_2}\sqrt{2gx_4} + \frac{\gamma_2\rho_2}{A_2}u_2, \\ \dot{x}_3 =& -\frac{a_3}{A_3}\sqrt{2gx_3} + \frac{(1-\gamma_2)\rho_2}{A_3}u_2, \\ \dot{x}_4 =& -\frac{a_4}{A_4}\sqrt{2gx_4} + \frac{(1-\gamma_1)\rho_1}{A_1}u_1, \\ y_1 =& \rho_c x_1, \quad y_2 = \rho_c x_2, \end{aligned} \tag{20}$$

where x_i is the water level in tank i. Choose the controller parameters as follows: cross-section of tank $A_1 = A_3 =$

$28, A_2 = A_4 = 32$; cross-section of the outlet hole $a_1 = a_3 = 0.071, a_2 = a_4 = 0.057$; acceleration due to gravity $g = 981$; gains $\rho_c = 0.5$; the fraction of water flowing to tank from pump $\gamma_1 = 0.7, \gamma_2 = 0.6$; prediction horizon $N = 10$; control horizon $N_u = 2$; sampling time $t = 1[s]$; weighting matrices $Q = I, \quad R = 10I$; scaling constant $\epsilon = 0.1$.

The process model is linearized around the operational points $x_{1o} = 12.4$, $x_{2o} = 12.7$, $x_{3o} = 1.8$, $x_{4o} = 1.4$. Consider input constraints $u_{\min} = 0$, $u_{\max} = 6$; output constraints $y_{\min} = 0$, $y_{\max} = 7.5$.

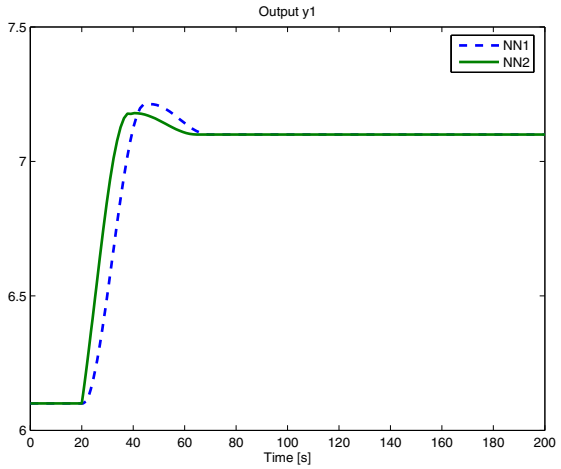

Fig. 2. Output responses in tank 1 of NN1 and NN2 approaches.

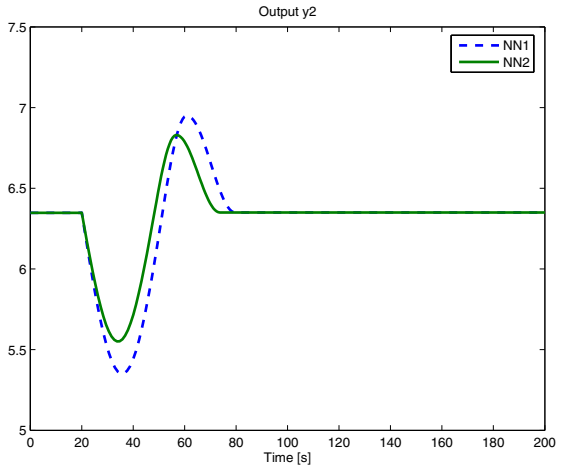

Fig. 3. Output responses in tank 2 of NN1 and NN2 approaches

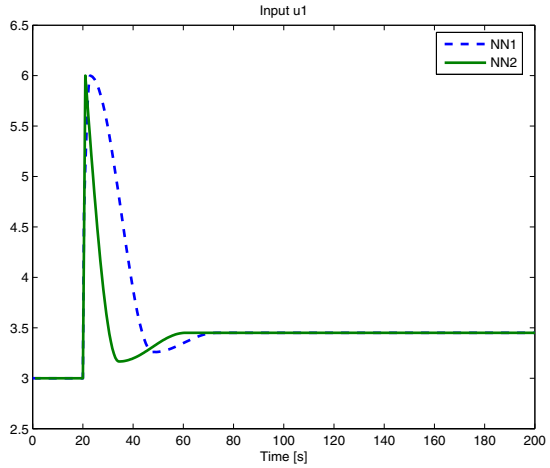

Fig. 4. Control signals in tank 1 of NN1 and NN2 approaches.

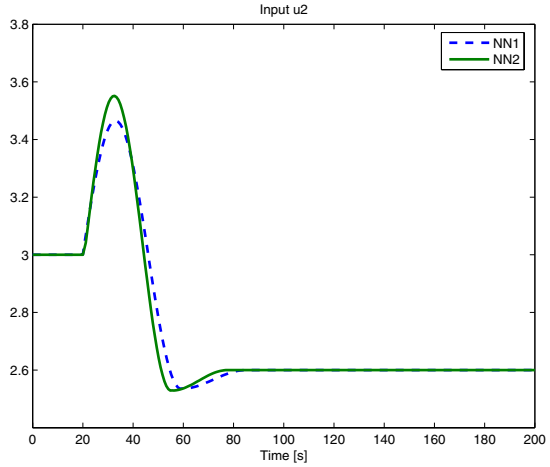

Fig. 5. Control signals in tank 2 of NN1 and NN2 approaches.

The simulation results are showed in Figs. 2 - 5. At time $k = 20$, a step reference change is commanded in y_1. At the same time a step load disturbance enters in y_2. We can see that both the proposed neural network approaches are effective, based on the advantages in parallel computation and hardware implementation of neural networks, the proposed approaches can solve the problem in real time efficiently.

The neural network controller based on linear programming (NN2) responds faster than the one based on quadratic programming (NN1). That is because NN2 approach have less computational cost than NN1 approach. We can conclude

that the NN2 approach is more suitable for solving control problems with large size and stringent real-time requirement. However, using approaches based on linear programming may result in a poor control performance, which depends on the the selection of the weighting matrices Q and R [13].

To further demonstrate the advantages of the proposed approaches, we can compare our approaches with other neural network approaches to MPC. In [28], a continuous-time neural network was applied to MPC, but the the control performance is sacrificed as the approximation strategy yields a sub-optimal solution; In [29], a discrete-time structured neural network was proposed to solve the quadratic programming problem involved in MPC exactly, however, the architecture of the neural network is more complex for hardware implementation than both two neural network approaches proposed in this paper. As a result, the proposed approaches have good performance and are more suitable for hardware implementation.

V. Conclusion

In this paper, we present two recurrent neural network approaches to design model predictive controllers based on both linear and quadratic programming formulations. Simulation results show that the proposed neural network approaches are effective and efficient in solving MPC problems. Furthermore, a comparison between the two neural network approaches has been made, which shows their different control behaviors. Finally, we compare the proposed approaches with some existing neural network approaches to MPC.

References

[1] E. F. Camacho and C. Bordons, *Model Predictive Control*, Springer, London, U.K., 2004.

[2] J. Richalet, A. Testud, L. J., J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, pp. 413-428, 1978.

[3] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733-764, 2003.

[4] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, London, U.K. Wiley, 1993.

[5] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst*, vol. CAS-33, pp. 533-541, May 1986.

[6] S. Liu and J. Wang, "A simplified dual neural network for quadratic 525 programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500-1510, Nov. 2006.

[7] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for linear programming," *Neural Computation*, in press, 2008.

[8] S. Boyd and L. Vandenbeghe, *Convex Optimization*, Cambridge, U.K., Cambridge Univ. Press, 2004.

[9] L. A. Zadeh and L. H. Whalen, "On optimal control and linear programming," *IEEE Trans. Automat. Contr.*, vol. AC-7, pp. 45-46, Jan. 1962.

[10] A. I. Propoi, "Use of linear programming methods for synthesizing sampled- data automatic systems," *Automat. Rem. Control.*, vol. 24, no. 7, pp. 837-844, 1963.

[11] T. S. Chang and D. E. Seborg, "A linear programming approach for multivariable feedback control with inequality constraints," *Int. J. Control*, vol. 37, no. 3, pp. 583-597, 1983.

[12] H. Genceli and M. Nikolaou, "Robust stability analysis of constrained ' -norm model predictive control," *AIChE J.*, vol. 39, no. 12, pp. 1954-1965, 1993.

[13] C. V. Rao and J. B. Rawlings, "Linear programming and model predictive control", *J. Process Control*, vol. 10, pp. 283-289, 2000.

[14] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming - the explicit solution," *IEEE Trans. Automatic Control*, vol. 47, pp. 1974-1985, 2002.

[15] P. J. Campo and M. Morari, "Model predictive optimal averaging level control," *AIChE J.*, vol. 35, no. 4, pp. 579-591, 1989.

[16] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 554-562, May 1988.

[17] S. Zhang and A. G. Constantinides, "Lagrange programming neural networks," *IEEE Trans. Circuits Syst. II.*, vol. 39, no. 7, pp. 441-452, Jul. 1992.

[18] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Netw.*, vol. 7, no. 4, pp. 629-641, 1994.

[19] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Networks.*, vol. 7, no. 6, pp. 1544-1547, Nov. 1996.

[20] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. 31, no. 1, pp. 147-154, Feb. 2001.

[21] Y. Zhang and J. Wang, "A dual neural network for convex quadratic programming subject to linear equality and inequality constraints," *Phys. Lett.*, A, pp. 271-278, Jun. 2002.

[22] Y. Xia, G. Feng, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and linear piecewise equations," *Neural Netw.*, vol. 17, no. 7, pp. 1003-1015, 2004.

[23] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming," *IEEE Trans. Neural Netw.*, in press, 2008.

[24] A. Rodriguez-Vazquez, A. Rueda, J. L. Huertas, R. Dominguez-Castro, "Swithed-Capacitor Neural Neworks for Linear Programming," *Electrical Letters.*, vol. 24, no. 8, pp. 469-498, Apr. 1988.

[25] J. Wang, "Analysis and design of a recurrent neural network for linear programming," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 40, no. 9, pp. 613-618, Sep. 1993.

[26] X. Wu, Y. Xia, and W. Chen, "A high-performance neural network for solving linear and quadratic programming problems," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 643-651, May 1996.

[27] K. H. Johansson, "The quadruple-tank process: a multivariable laboratory with an adjustable aero," *IEEE Trans. Control Systems Technology*, vol. 8, no. 3, pp. 456-465, May, 2000.

[28] J. M. Quero and E. F. Camacho, "Neural network for constrained predictive control," *IEEE Trans. Circuits Syst. I.*, vol. 40, no. 9, pp. 621-626, May, 1993.

[29] L. Wang and F. Wan, "Strucured neural networks for constrained model predictive control," *Automatica*, vol. 37, pp. 1235-1243, 2001.