Proceedings of the
2005 IEEE Conference on Control Applications
Toronto, Canada, August 28-31, 2005

Output-Feedback Model Predictive Control with Constraints – Improve Robustness Using Past Information

Withit Chatlatanagulchai and Peter H. Meckl, *Member, IEEE*

Abstract— **We present nonlinear model predictive control design for a type of discrete-time nonlinear system. Range constraints are imposed on input, output and input rate. When states are not available, an observer based on Newton's algorithm is used to estimate all states. We also present an idea to improve robustness of the control system when plant is subjected to modeling uncertainties. By using past information, the correction to the plant model can be computed from the difference between past actual states and past model states using a simple linear least-square algorithm. The correction is applied to the plant model in the future time step. Two illustrative examples are included.**

I. Introduction

MODEL predictive control relies on a plant internal model. The optimizer is used to solve for the control trajectory over a future time horizon based on an internal model of the plant. Therefore, the accuracy of the plant model relative to the actual plant is vital to controller performance. Typically, the plant model used is linear due to its simplicity and mature theoretical support [1]. However, if the plant exhibits severe nonlinearities or the process operates at multiple set points, the usefulness of predictive control based on a linearized model is limited. Recently, researchers have considered nonlinear model predictive control (NMPC) where the internal plant model is nonlinear. A good collection of works in NMPC can be found in [2].

One of the most common NMPC is to re-linearize the nonlinear model at each operating setpoint, which reduces the problem to a linear MPC as in [3]. A more advanced setup using gain scheduling and neural network can be found in [4]. Re-linearization is a compromising solution, which requires long computation time and suffers from performance degradation when states are between setpoints. There are some efforts in modeling the nonlinear plant by a neural network (NN) using its property as a universal approximator. In [5], NN is used to learn a mapping from input to output of a nonlinear plant. The learning is performed offline. Then, feedback linearization is performed which results in a linearized plant. The control task is reduced to control of a linear plant, which is done by linear MPC. System performance depends on accuracy of the NN model and approximation of nonlinear constraints. In [6], a growing NN, whose number of neurons depends on basis function coverage and prediction error, is used to learn nonlinear plant online. However, the design is based on a rather simple plant and without constraints. Other works based on a nonlinear plant model usually lead to the use of nonlinear optimization algorithms, which may not guarantee global optimum or even feasibility of a solution.

In this paper, a mapping from input rate to output is formed over the receding horizon. The optimization problem is reduced to simple quadratic programming. The range constraints, consisting of input constraint, output constraint, and input rate constraint, are formed as part of the optimization problem. The plant model differs from the actual plant. However, the difference is assumed to be bounded where bounds may be unknown. The uncertainties can be in the form of multiplicative or additive uncertainties. Robustness is improved by using the difference between actual states and model states in the past time step. This difference is used to compute the correction terms, which are applied to the plant model in the future time step using a linear least-square algorithm. The paper also presents the case when states are not measurable. An observer based on Newton's algorithm as in [7] is applied to the system.

This paper is organized as follows. In Section II, we discuss some types of applicable systems and present constrained NMPC. In Section III, an observer based on Newton's algorithm is presented. In Section IV, robustness obtained from using past information is discussed. Two simulation examples to demonstrate the effectiveness of the scheme are presented in Section V. Section VI is the conclusion.

II. Constrained NMPC For Specific Types of Nonlinear Systems

Let the future prediction horizon be H_p, we are interested in finding state variables over the future prediction horizon $x_i(k+H_p)$. The applicable system has the following form

$$x_i(k+j) = F_{i,j}\left(\bar{x}_m(k)\right) + \sum_{p=0}^{j-1} G_{i,j,p}\left(\bar{x}_m(k)\right) u(k+p) \quad (1)$$

$$, i = 1,2,\ldots,m, \text{ and } j = 1,2,\ldots,H_p,$$

where $\bar{x}_i = \{x_1, x_2, \ldots, x_i\}$ is a set of state variables.

Not all systems can be transformed into the above form due to the dependence of $F_{i,j}$ and $G_{i,j,p}$ on the input u. Examples of some applicable systems are as follows.

All linear systems

Manuscript received February 18, 2005.

The authors are with Motion and Vibration Control Laboratory, School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907-2088 USA (e-mail: chatlata@purdue.edu, meckl@purdue.edu; phone: 765-494-0539; fax: 765-494-5686).

$$
\begin{aligned}
x_1(k+1) &= c_{11}x_1(k)+c_{12}x_2(k)+\ldots+c_{1m}x_m(k)+c_{1u}u(k),\\
&\vdots\\
x_m(k+1) &= c_{m1}x_1(k)+c_{m2}x_2(k)+\ldots+c_{mm}x_m(k)+c_{mu}u(k),\\
y(k) &= C_y x(k),
\end{aligned}
$$

are applicable with the future prediction horizon H_p as large as infinity. c_{ij} are constants and C_y is a constant vector with appropriate dimensions.

Nonlinear systems in the form

$$
\begin{aligned}
x_i(k+1) &= f_i\left(\bar{x}_{m-1}(k)\right)+g_i\left(\bar{x}_{m-1}(k)\right)x_{i+1}(k)\\
&\quad +c_{im}x_m(k),\ i=1,2,\ldots,m-2,\\
&\vdots\\
x_{m-1}(k+1) &= f_{m-1}\left(\bar{x}_{m-1}(k)\right)+c_{(m-1)m}x_m(k),\\
x_m(k+1) &= f_m\left(\bar{x}_{m-1}(k)\right)+g_m\left(\bar{x}_{m-1}(k)\right)u(k)+c_{mm}x_m(k),\\
y(k) &= C_y x(k),
\end{aligned}
\tag{2}
$$

where c_{ij} are constants and C_y is a constant vector, are applicable when $H_p \le 3$. $\bar{x}_i = \{x_1, x_2, \ldots, x_i\}$ is a set of state variables, $f_i, g_i \in \mathbb{R}, i = 1,2,\ldots,m$ are known nonlinear functions, $u, y \in \mathbb{R}$ are input and output, respectively.

Nonlinear systems in the form

$$
\begin{aligned}
x_i(k+1) &= f_i\left(\bar{x}_m(k)\right)+g_i\left(\bar{x}_m(k)\right)x_{i+1}(k), i=1,\ldots,m-1,\\
x_m(k+1) &= f_m\left(\bar{x}_m(k)\right)+g_m\left(\bar{x}_m(k)\right)u(k),\\
y(k) &= C_y x(k),
\end{aligned}
$$

are applicable when $H_p = 1$.

For nonlinear systems whose $F_{i,j}$ or $G_{i,j,p}$ in (1) are functions of $u(k),\ldots,u(k+j-2)$, we can replace $u(k),\ldots,u(k+j-2)$ with the control input of the past step, $u(k-1)$, which has already been computed. However, the effect of doing so, on the closed-loop system performance, is difficult to determine analytically.

Definition 1: For simplicity, we will write $h(\bar{x}_i(k))$ as $h(k)$, where h is a function of $x_1, x_2, \ldots, x_i$, at time step k.

In this section, we assume that all state variables can be directly measured. Suppose we use a control horizon, $H_u < H_p$, we have

$$
\begin{aligned}
u(k) &= \Delta u(k)+u(k-1),\\
u(k+1) &= \Delta u(k+1)+\Delta u(k)+u(k-1),\\
&\vdots\\
u(k+H_u-1) &= \Delta u(k+H_u-1)+\ldots+\Delta u(k)+u(k-1),
\end{aligned}
\tag{3}
$$

where

$$\Delta u(k+j) = u(k+j)-u(k+j-1),\ \forall j = 0,\ldots,H_u-1.$$

For $H_u \le j \le H_p - 1$, we have

$$u(k+j) = u(k+H_u-1). \tag{4}$$

From (1)-(4), we can write the mapping from input rate, Δu, to future state variables over the prediction horizon as follows:

$$X = \Psi + \Omega\,\Delta U, \tag{5}$$

where

$$
\Omega = \begin{bmatrix}
G_{1,1,0}(k) & \cdots & 0\\
\vdots & \ddots & \vdots\\
G_{m,1,0}(k) & \cdots & 0\\
\vdots & \vdots & \vdots\\
\sum_{p=0}^{H_u-1} G_{1,H_u,p}(k) & \cdots & \sum_{p=H_u-1}^{H_u-1} G_{1,H_u,p}(k)\\
\vdots & \ddots & \vdots\\
\sum_{p=0}^{H_u-1} G_{m,H_u,p}(k) & \cdots & \sum_{p=H_u-1}^{H_u-1} G_{m,H_u,p}(k)\\
\sum_{p=0}^{H_u} G_{1,H_u+1,p}(k) & \cdots & \sum_{p=H_u-1}^{H_u} G_{1,H_u+1,p}(k)\\
\vdots & \ddots & \vdots\\
\sum_{p=0}^{H_u} G_{m,H_u+1,p}(k) & \cdots & \sum_{p=H_u-1}^{H_u} G_{m,H_u+1,p}(k)\\
\vdots & \vdots & \vdots\\
\sum_{p=0}^{H_p-1} G_{1,H_p,p}(k) & \cdots & \sum_{p=H_u-1}^{H_p-1} G_{1,H_p,p}(k)\\
\vdots & \ddots & \vdots\\
\sum_{p=0}^{H_p-1} G_{m,H_p,p}(k) & \cdots & \sum_{p=H_u-1}^{H_p-1} G_{m,H_p,p}(k)
\end{bmatrix},
$$

$$
\begin{aligned}
\Psi = \Big[& F_{1,1}(k)+G_{1,1,0}(k)u(k-1),\ldots,F_{m,1}(k)+G_{m,1,0}(k)\\
& u(k-1),\ldots,F_{1,H_u}(k)+\sum_{p=0}^{H_u-1} G_{1,H_u,p}(k)u(k-1),\ldots,\\
& F_{m,H_u}(k)+\sum_{p=0}^{H_u-1} G_{m,H_u,p}(k)u(k-1),F_{1,H_u+1}(k)\\
& +\sum_{p=0}^{H_u} G_{1,H_u+1,p}(k)u(k-1),\ldots,F_{m,H_u+1}(k)+\sum_{p=0}^{H_u} G_{m,H_u+1,p}(k)\\
& u(k-1),\ldots,F_{1,H_p}(k)+\sum_{p=0}^{H_{p-1}} G_{1,H_p,p}(k)u(k-1),\ldots,\\
& F_{m,H_p}(k)+\sum_{p=0}^{H_{p-1}} G_{m,H_p,p}(k)u(k-1)\Big]^T,
\end{aligned}
$$

$$
\Delta U = \begin{bmatrix} \Delta u(k)\\ \vdots\\ \Delta u(k+H_u-1) \end{bmatrix},
$$

$$
\begin{aligned}
X = \Big[& x_1(k+1),\ldots,x_m(k+1),\ldots,x_1(k+H_u),\ldots,\\
& x_m(k+H_u),x_1(k+H_u+1),\ldots,x_m(k+H_u+1),\\
& \ldots,x_1(k+H_p),\ldots,x_m(k+H_p)\Big]^T.
\end{aligned}
$$

Using (1), we can write the output over the future prediction horizon as follows:

$$Y = CX = C\Psi + C\Omega\,\Delta U, \tag{6}$$

where

$$Y = \left[y(k+1),\ldots,y(k+H_u),y(k+H_u+1),\ldots,y(k+H_p) \right]^T,$$

and $C = \left[block - diag\left(C_y\right) \right]$.

Suppose our task is to design a controller to make the output track a desired trajectory as closely as possible. The desired trajectory over the prediction horizon can be put in a vector as

$$\Upsilon = \left[r(k+1), \ldots, r(k+H_p) \right]^T.$$

Definition 2: Let $\|(\bullet)\|_Q^2$ and $\|(\bullet)\|_R^2$ represent $(\bullet)^T Q(\bullet)$ and $(\bullet)^T R(\bullet)$, respectively, where $(\bullet)$ is a vector and Q, R are matrices with appropriate dimensions.

Let the cost function, which we are trying to minimize, be in the form

$$\begin{aligned} V(k) &= \|Y - \Upsilon\|_Q^2 + \|\Delta U\|_R^2 \\ &= \sum_{i=1}^{H_p} \|y(k+i) - r(k+i)\|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \|\Delta u(k+i)\|_{R(i)}^2 \\ &= \|C\Omega\Delta U + C\Psi - \Upsilon\|_Q^2 + \|\Delta U\|_R^2 \\ &= \left[\Psi^T C^T - \Upsilon^T \right] Q \left[C\Psi - \Upsilon \right] + 2\Delta U^T \Omega^T C^T \\ & \quad Q\left[C\Psi - \Upsilon \right] + \Delta U^T \left[\Omega^T C^T Q C \Omega + R \right] \Delta U. \end{aligned}$$

We see that

$$\begin{aligned} \min_{\Delta U} V(k) = \min_{\Delta U} \{ & 2\Delta U^T \Omega^T C^T Q \left[C\Psi - \Upsilon \right] \\ & + \Delta U^T \left[\Omega^T C^T Q C \Omega + R \right] \Delta U \}. \end{aligned} \quad (7)$$

Next, consider the constraints imposed on input, output and input rate as follows:

$$E\begin{bmatrix} \Delta U \\ 1 \end{bmatrix} \le 0, \; F\begin{bmatrix} U \\ 1 \end{bmatrix} \le 0, \; G\begin{bmatrix} Y \\ 1 \end{bmatrix} \le 0,$$

where

$$E = \left[\mathbf{E}, e \right] \in \mathbb{R}^{n_E \times (H_u+1)}, F = \left[F_1, F_2, \ldots, F_{H_u}, f \right] \in \mathbb{R}^{n_F \times (H_u+1)},$$

$G = \left[\Phi, g \right] \in \mathbb{R}^{n_G \times (H_p+1)}$, e, f, g are vectors.

As shown in [1], let $\mathbf{F}_i = \sum_{j=i}^{H_u} F_j$, $\mathbf{F} = \left[\mathbf{F}_1, \ldots, \mathbf{F}_{H_u} \right]$, we can transform the input constraint into a constraint on input rate as follows:

$$\mathbf{F}\,\Delta U \le -\mathbf{F}_1 u(k-1) - f. \quad (8)$$

Output constraint can also be transformed into a constraint on input rate. By using (6), we have

$$\begin{aligned} G\begin{bmatrix} C\Psi + C\Omega\,\Delta U \\ 1 \end{bmatrix} &= \Phi\left[C\Psi + C\Omega\,\Delta U \right] + g, \\ \Phi C\Omega\,\Delta U &\le -\Phi C\Psi - g. \end{aligned} \quad (9)$$

The constraint on input rate can be put in the form

$$\mathbf{E}\,\Delta U \le -e. \quad (10)$$

Combining (8)-(10), we can write all constraints in one matrix inequality as follows:

$$\begin{bmatrix} \mathbf{F} \\ \Phi C\Omega \\ \mathbf{E} \end{bmatrix} \Delta U \le \begin{bmatrix} -\mathbf{F}_1 u(k-1) - f \\ -\Phi C\Psi - g \\ -e \end{bmatrix}. \quad (11)$$

Usually, all constraints are given in ranges such as

$$\Delta u_{\min} \le \Delta u(k+i) \le \Delta u_{\max}, \; \forall i = 0, \ldots, H_u - 1,$$

$$u_{\min} \le u(k+i) \le u_{\max}, \; \forall i = 0, \ldots, H_u - 1,$$

$$y_{\min} \le y(k+i) \le y_{\max}, \; \forall i = 1, \ldots, H_p.$$

It can be shown that, for range constraints as above, we have

$$E = \begin{bmatrix} \mp 1/\Delta u_{\min} & 0 & \cdots & 0 & \pm 1 \\ \pm 1/\Delta u_{\max} & 0 & \cdots & 0 & \mp 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mp 1/\Delta u_{\min} & \pm 1 \\ 0 & 0 & \cdots & \pm 1/\Delta u_{\max} & \mp 1 \end{bmatrix} \in \mathbb{R}^{2H_u \times (H_u+1)},$$

$$F = \begin{bmatrix} \mp 1/u_{\min} & 0 & \cdots & 0 & \pm 1 \\ \pm 1/u_{\max} & 0 & \cdots & 0 & \mp 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mp 1/u_{\min} & \pm 1 \\ 0 & 0 & \cdots & \pm 1/u_{\max} & \mp 1 \end{bmatrix} \in \mathbb{R}^{2H_u \times (H_u+1)},$$

$$G = \begin{bmatrix} \mp 1/y_{\min} & 0 & \cdots & 0 & \pm 1 \\ \pm 1/y_{\max} & 0 & \cdots & 0 & \mp 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mp 1/y_{\min} & \pm 1 \\ 0 & 0 & \cdots & \pm 1/y_{\max} & \mp 1 \end{bmatrix} \in \mathbb{R}^{2H_p \times (H_p+1)}.$$

We can see that the tracking problem reduces to the constrained optimization problem where we minimize (7), subject to the constraint (11). This is a standard quadratic programming problem, and standard algorithms are available for its solution. After the optimized ΔU vector is obtained, the user can select whether to apply only the first element of ΔU to the plant, i.e. $u(k) = u(k-1) + \Delta u(k)$, and discard the rest or to apply a few elements. The first choice requires the whole cycle to be repeated at every time step.

III. An Observer Based on Newton's Algorithm

In this section, we will design an observer for a system more extensive than the system (1). Consider a discrete-time system in the form:

$$\begin{aligned} & x_i(k+1) = f_i(\overline{x}_m(k)) + g_i(\overline{x}_m(k)) x_{i+1}(k), \; i = 1, 2, \ldots, m-1, \\ & x_m(k+1) = f_m(\overline{x}_m(k)) + g_m(\overline{x}_m(k)) u(k), \\ & y(k) = h(\overline{x}_m(k)), \end{aligned}$$

where all symbols are defined as in Section II. Define U_{N-1} as a vector of inputs at all $N-1$ previous time steps as follows:

$$U_{N-1} \triangleq \left[u\left(k-N+1\right), \ldots, u\left(k-1\right) \right]^T.$$

Define Y_N as a vector of outputs at current time step k and all $N-1$ previous time steps. We obtain

$$\begin{aligned} Y_N &\triangleq \left[y\left(k-N+1\right), y\left(k-N+2\right), \ldots, y\left(k-1\right), y\left(k\right) \right]^T \\ &= \left[h^x\left(k-N+1\right), F^{x,u}\left(k-N+1\right) \circ h^x\left(k-N+2\right), \right. \\ & \quad \ldots, F^{x,u}\left(k-N+1\right) \circ \ldots \circ F^{x,u}\left(k-2\right) \circ h^x\left(k-1\right), \\ & \quad \left. F^{x,u}\left(k-N+1\right) \circ \ldots \circ F^{x,u}\left(k-1\right) \circ h^x\left(k\right) \right]^T \\ &\triangleq H\left(\overline{x}_m(k-N+1), U_{N-1}\right), \end{aligned} \quad (12)$$

where $h^x(j)$ denotes $h\left(\overline{x}_m(j)\right)$, $F^{x,u}(j)$ represents mapping from $\overline{x}_m(j+1), u(j+1)$ to $\overline{x}_m(j)$ and the symbol

$\circ$ represents composite function. We see that Y_N is a $N\times 1$ vector of available measured outputs. $H\left(\bar{x}_m(k-N+1), U_{N-1}\right)$ is a vector of known previous control inputs and m unknown states, which are $x_1(k-N+1),\ldots,x_m(k-N+1)$. Therefore, we can use Newton's algorithm to solve for the m unknowns of the equation, $Y_N - H\left(\bar{x}_m(k-N+1), U_{N-1}\right)=0$. For simplicity, we assume $N=m$. The Newton's algorithm as well as the convergence theorem of this algorithm can be found in [8]. Newton's algorithm to solve for roots of

$$Y_N - H\left(\bar{x}_m(k-N+1), U_{N-1}\right)=0,$$

is given by

$$\begin{aligned}\hat{x}^{i+1}(k-N+1) &= \hat{x}^{i}(k-N+1)\\ &+\left[\frac{\partial H}{\partial x}\left(\hat{x}^{i}(k-N+1), U_{N-1}\right)\right]^{-1}\\ &\bullet\left(Y_N - H\left(\hat{x}^{i}(k-N+1), U_{N-1}\right)\right).\end{aligned}\tag{13}$$

$\hat{x}^{i}(k-N+1)$ denotes $\left[\hat{x}_1(k-N+1),\ldots,\hat{x}_m(k-N+1)\right]^T$ at i^{th} iteration of the algorithm. In the case where $N>m$, the inverse of the Jacobian is replaced by pseudo-inverse as in [8].

After $\hat{\bar{x}}_m(k-N+1)$ are obtained, we can find the states at current time step *k* as follows:

$$\hat{\bar{x}}_m(k) = F^{\hat{x},u}(k-1)^{-1}\circ\ldots\circ F^{\hat{x},u}(k-N+1)^{-1}.$$

IV. Improving Robustness Using Past Information

Obtaining an accurate plant model, especially for chemical process plant, usually requires lots of investment in time and money. Due to this reason, most plant models are simplified versions of the actual plant. The difference between the plant model and the actual plant is called the model uncertainty. Since MPC relies on the accuracy of this plant model, unless the uncertainty is taken into account explicitly in the control design algorithm, controller performance will be compromised. The idea behind this section is that we can always improve the future by looking into the past. We assume that the difference between actual and model plants is bounded. The correction to the future plant model is computed from this difference using the linear least-square algorithm. Consider the model plant as follows:

$$\begin{aligned}x_i(k+1) &= \hat{f}_i(\bar{x}_m(k)) + \hat{g}_i(\bar{x}_m(k))x_{i+1}(k),\ \ i=1,2,\ldots,m-1,\\ x_m(k+1) &= \hat{f}_m(\bar{x}_m(k)) + \hat{g}_m(\bar{x}_m(k))u(k),\\ y(k) &= C_y x(k),\end{aligned}$$

where $\hat{f}_i, \hat{g}_i, \forall i=1,\ldots,m,$ are known functions. Suppose the actual plant has the following form:

$$\begin{aligned}x_i(k+1) &= f_i(\bar{x}_m(k)) + g_i(\bar{x}_m(k))x_{i+1}(k),\ i=1,2,\ldots,m-1,\\ x_m(k+1) &= f_m(\bar{x}_m(k)) + g_m(\bar{x}_m(k))u(k),\\ y(k) &= C_y x(k),\end{aligned}$$

where $f_i, g_i, \forall i=1,\ldots,m,$ are unknown functions.

Assumption 1: The differences between actual plant functions and plant model functions are bounded, that is, $f_i(k)=\hat{f}_i(k)+\Delta f_i(k)$, $g_i(k)=\hat{g}_i(k)+\Delta g_i(k)$, $\forall i=1,\ldots,m,$ where $\|\Delta f_i\|\le\varepsilon_{fi}$, $\|\Delta g_i\|\le\varepsilon_{gi}$, ε_{fi} and ε_{gi} are some finite numbers.

Using Assumption 1, we have

$$\begin{aligned}x_i(k+1) &= \hat{f}_i(k)+\Delta f_i(k)+(\hat{g}_i(k)+\Delta g_i(k))x_{i+1}(k),\\ &i=1,2,\ldots,m-1,\\ x_m(k+1) &= \hat{f}_m(k)+\Delta f_m(k)+(\hat{g}_m(k)+\Delta g_m(k))u(k),\\ y(k) &= C_y x(k).\end{aligned}$$

Let the past information horizon be H_h, we have $A_i = B_i\mathbf{X}_i$, where

$$A_i=\begin{bmatrix} x_i(k)-\hat{f}_i(k-1)\\ -\hat{g}_i(k-1)x_{i+1}(k-1)\\ \hline \vdots\\ \hline x_i(k-H_h+1)-\hat{f}_i(k-H_h+1)\\ -\hat{g}_i(k-H_h+1)x_{i+1}(k-H_h+1)\end{bmatrix},$$

$$B_i=\begin{bmatrix}1 & x_{i+1}(k-1)\\ \vdots & \vdots\\ 1 & x_{i+1}(k-H_h+1)\end{bmatrix},$$

$\mathbf{X}_i=\left[\Delta f_i, \Delta g_i\right]^T$, $\forall i=1,\ldots,m-1.$

When $i=m,$ the above still holds by replacing x_{i+1} with u. Matrices A_i, B_i are known past information. Finding $\mathbf{X}_i$ reduces to an optimization problem

$$\min_{\mathbf{X}_i}\frac{1}{2}\left\|A_i - B_i\mathbf{X}_i\right\|^2,$$

which is a standard linear programming problem. After the correction vector $\mathbf{X}_i$ is obtained, the correction is applied to the plant model in the next step and the whole cycle is repeated.

V. Simulation Results

To demonstrate the effectiveness of the control design, we present two simulation examples as follows.

Example 1: One-link flexible-joint robot manipulator in vertical plane

The manipulator in Fig. 1, can be put in the form

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= \frac{1}{\left(m_l a^2 + J_l + m_p b^2 + J_p\right)}[-kx_1 - \left(m_l a + m_p b\right)g\sin x_1\\ &\quad -\left(c_l + d\right)x_2 + \frac{k}{r}x_3 + \frac{d}{r}x_4],\\ \dot{x}_3 &= x_4,\\ \dot{x}_4 &= \frac{1}{\left(J_m + \frac{J_s}{r^2}\right)}\left(\frac{k}{r}x_1 + \frac{d}{r}x_2 - \frac{k}{r^2}x_3 - \left(c_m + \frac{d}{r^2}\right)x_4 + u\right),\\ y &= x_1,\end{aligned}$$

where $x_1=\theta_l$, $x_2=\dot{\theta}_l$, $x_3=\theta_m$, $x_4=\dot{\theta}_m$, $u=T$. Table I shows description of parameters and their values.

For convenience, we replace groups of parameters by c_i and rewrite the plant in discrete time with sampling period

t_s as

$$
\begin{aligned}
x_{1,k+1} &= x_{1,k} + t_s\ x_{2,k},\\
x_{2,k+1} &= x_{2,k} + t_s\left(c_1 x_{1,k} + c_2 \sin x_{1,k} + c_3 x_{2,k} + c_4 x_{3,k} + c_5 x_{4,k}\right),\\
x_{3,k+1} &= x_{3,k} + t_s\ x_{4,k},\\
x_{4,k+1} &= x_{4,k} + t_s\left(c_6 x_{1,k} + c_7 x_{2,k} + c_8 x_{3,k} + c_9 x_{4,k} + c_{10} u_k\right),
\end{aligned}
\tag{14}
$$

where $x_{i,j}$ denotes x_i at time step j. Let $N = m = 4$, the equation (12) can be obtained. We then can find the Jacobian $\partial H / \partial x$ and the Newton's algorithm in (13) can proceed. In this simulation, we let all initial guesses, $\hat{x}^1(k-3)$, equal 0.1 with 2 iterations. After the output y is measured at time step k, all states are estimated and used in the controller design algorithm in Section II. We use $H_p = 3, H_u = 2$. Equation (1) can then be obtained.

Note that the discrete-time system (14) is in the form of (2) and it can be transformed to (1) when $H_p \le 3$. Our objective is to make the output track a signal, obtained from passing a square wave with amplitude 10 and frequency 0.05 Hz to a filter with transfer function $1/\left(s^3 + 6s^2 + 12s + 8\right)$, as closely as possible by using only output measurement. The sampling interval used is 0.01 sec. Cost function matrices are $Q = 1000\, I_{3\times 3}, R = 100\, I_{2\times 2}$. ΔU is obtained from solving optimization problem (7) using Matlab command *quadprog*. Then $u(k)$ is obtained by adding the first element of ΔU to $u(k-1)$.

The simulation results are given in Fig. 2 to Fig. 3. Fig. 2 shows tracking result, input and input rate. Fig. 3 shows state estimation errors. The observer estimates the actual states closely. We obtain an acceptable tracking performance using estimated states from the observer.

Example 2: A nonlinear discrete-time SISO plant model, as given in [9], is used. The plant model is

$$
\begin{aligned}
x_1(k+1) &= 1.4x_1^2(k)/\left(1+x_1^2(k)\right) + 0.3x_2(k),\\
x_2(k+1) &= x_1(k)/\left(1+x_1^2(k)+x_2^2(k)\right) + u(k),\\
y(k) &= x_2(k).
\end{aligned}
\tag{15}
$$

To demonstrate the effectiveness of the algorithm in Section IV, let the model be different from the actual plant

$$
\begin{aligned}
x_1(k+1) &= 1.4x_1^2(k)/\left(1+x_1^2(k)\right)\times 4\sin(x_1(k)) + 0.3x_2(k)\\
&\quad - 0.1\cos(0.05k)\cos(x_1(k)),\\
x_2(k+1) &= x_1(k)/\left(1+x_1^2(k)+x_2^2(k)\right)\times x_1(k)x_2(k) + u(k)\\
&\quad + 0.1\cos(0.05k)\cos(x_1(k)),\\
y(k) &= x_2(k).
\end{aligned}
$$

Measured states are obtained from the actual plant, whereas the plant model (15) is used in controller design. The correction terms to the plant model are obtained from the algorithm in Section IV with the past information horizon $H_h = 1$.

The control objective is to make the output y follow a trajectory, $y_d = 0.5\sin(k/120) + 0.5\sin(k/60)$, as closely as possible.

We use the following system parameters, matrices and initial values:

$H_p = 3, H_u = 2, x_1(0) = x_2(0) = 0.1, Q = I^{3\times 3}, R = 0.1\times I^{2\times 2}$

$C = block - diag([0\ \ 1]) \in \mathbb{R}^{3\times 6}$.

Only the first element of ΔU is applied as the input signal to the plant at step k.

Fig. 4 demonstrates the effectiveness of the model-correction algorithm when it is applied at time step, $k = 500$, onward. We can see that tracking error is reduced significantly.

For constrained case, we impose an output constraint, $-0.1 \le y \le 0.5$, to the system with result in Fig. 5a. Fig. 5b is the result from imposing an input constraint, $-0.5 \le u \le 0.5$. From Fig. 5a, we see that the model-correction algorithm reduces the violation of the output constraint, which may result in the ability to operate closer to the constraint. Fig. 5b shows that model uncertainty does not affect the control input signal.

VI. Conclusion

This paper extends the linear MPC to specific types of nonlinear systems. When states are not measurable, we show the possibility of incorporating an observer based on Newton's algorithm into the system to estimate the states. We also suggest the idea of improving robustness of the design by using information from the past. The simulation examples show applicability to real-life systems as well as the potential to save resources by being able to operate closer to the constraints.

References

[1] J. M. Maciejowski, *Predictive Control with Constraints.* England: Prentice Hall, 2002.

[2] F. Allgower and A. Zheng, *Nonlinear Model Predictive Control.* Switzerland: Birkhauser, 2000.

[3] M. Huzmezan and J. M. Maciejowski, "Reconfiguration and scheduling in flight using quasi-LPV high-fidelity models and MBPC control," in *Proc. American Control Conference,* Philadelphia, 1998, pp. 3649-3653.

[4] S. Piche, B. S. Rodsari, D. Johnson and M. Gerules, "Nonlinear model predictive control using neural networks," *IEEE Control Systems Magazine,* pp. 53-62, Jun. 2000.

[5] M. A. Botto, T. J. J. Vandenboom, A. Krijgsman and J. Costa, "Predictive control based on neural network models with I/O feedback linearization," *Int. Journal of Control,* vol. 72, no. 17, pp. 1538-1554, 1999.

[6] G. P. Liu, V. Kadirkamanathan and S. A. Billings, "Predictive control for non-linear systems using neural networks," *Int. Journal of Control,* vol. 71, no. 6, pp. 1119-1132, 1998.

[7] P. E. Moraal and J. W. Grizzle, "Observer design for nonlinear systems with discrete-time measurements," *IEEE Trans. on Automatic Control,* vol. 40, no. 3, pp. 395-404, 1995.

[8] D. G. Luenberger, *Optimization by Vector Space Methods.* New York: Wiley, 1969.

[9] S. S. Ge, G. Y. Li, and T. H. Lee, "Adaptive NN control for a class of strict-feedback discrete-time nonlinear systems," *Automatica,* vol. 39, no. 5, pp. 807-819, 2003.

TABLE I
DESCRIPTION OF PARAMETERS AND THEIR VALUES USED IN EXAMPLE 1

		Value in SI
a	Distance from sprocket center to link C.G.	0.09443
b	Distance from sprocket center to payload C.G.	0.254
J_m	Motor inertia about motor C.G.	0.000525
J_s	Sprocket inertia about sprocket C.G.	0.00005
J_l	Link inertia about link C.G.	0.07
J_p	Payload inertia about payload C.G.	0.0005
m_l	Link mass	3.545
m_p	Payload mass	1.0
k	Spring stiffness coefficient	6.77
d	Spring internal damping	0.001507
c_m	Coefficient of viscous friction at motor bearing	0.02
c_l	Coefficient of viscous friction at joint bearing	0.01
r	Gear ratio	5.3

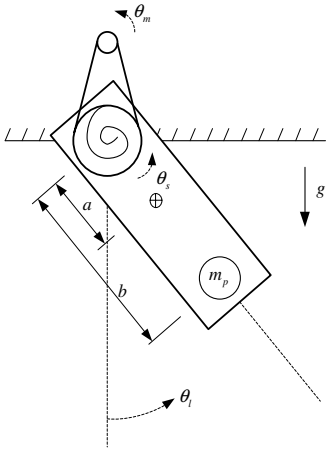

Fig. 1. Schematic diagram of one-link flexible joint robot manipulator.

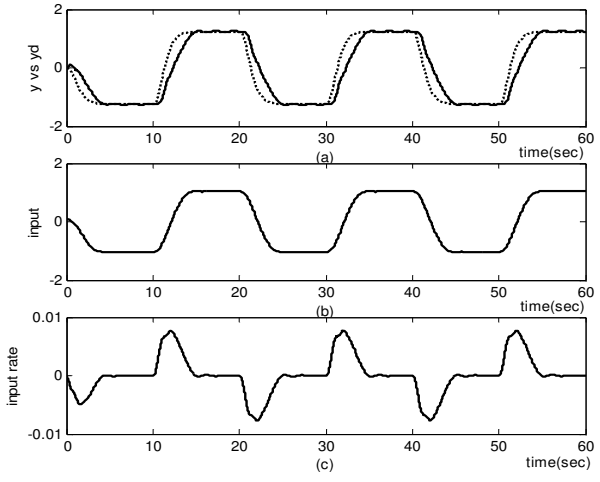

Fig. 2. Output, input, and input rate during 60 sec: (a) output vs. reference, (b) input, (c) input rate.

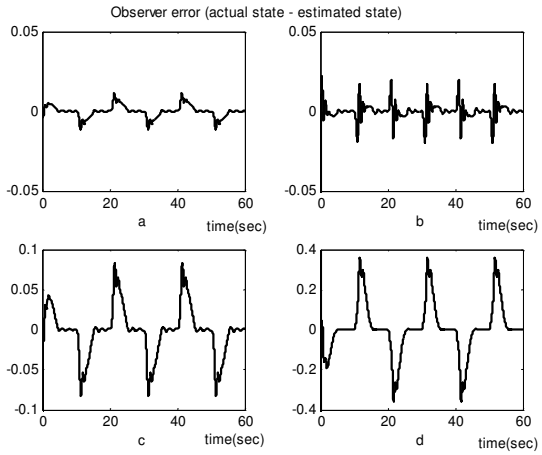

Fig. 3. Estimated errors during 60 sec: (a) $\hat{x}_1 - x_1$, (b) $\hat{x}_2 - x_2$, (c) $\hat{x}_3 - x_3$, (d) $\hat{x}_4 - x_4$.

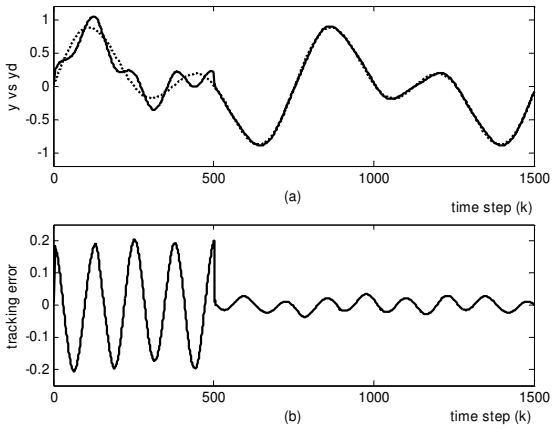

Fig. 4. Robust tracking performance during 1500 steps when actual plant differs from plant model: (a) tracking result. The solid line is the actual output y and the dotted line is the reference output y_d. (b) tracking error. The model correction algorithm using past information is turned on at time step $k = 500$.

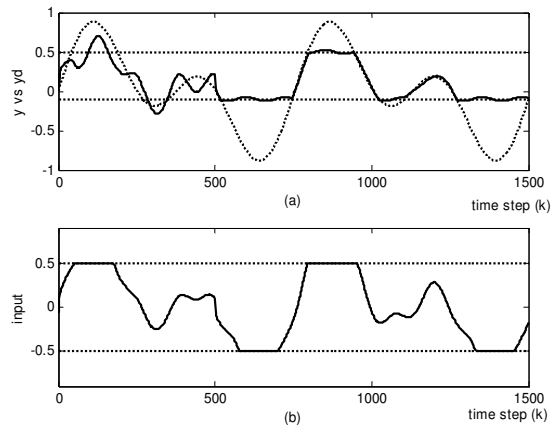

Fig. 5. Robust constrained tracking performance during 1500 steps when actual plant differs from plant model: (a) tracking result. The solid line is the actual output y and the dotted line is the reference output y_d. (b) control input. The model correction algorithm using past information is turned on at time step $k = 500$.