Model-Based Predictive Control (MPC) Design Using Discrete Model Directly

by
Stanislaw H. Żak

1 A Simple Discrete-Time MPC

We consider a discretized model of a dynamical system of the form,

$$\boldsymbol{x}[k+1] = \boldsymbol{\Phi}\boldsymbol{x}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k] \tag{1}$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\boldsymbol{x}[k], \tag{2}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$, $\boldsymbol{\Gamma} \in \mathbb{R}^{n\times m}$, and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$.

Suppose now that the state vector $\boldsymbol{x}$ at each sampling time, k, is available to us. Our control objective is to construct a control sequence,

$$\boldsymbol{u}[k], \boldsymbol{u}[k+1], \ldots, \boldsymbol{u}[k+N_p-1], \tag{3}$$

where N_p is the prediction horizon, such that a given cost function is minimized subject to constraints devised by the controller designer. The above control sequence will result in a predicted sequence of the state vectors,

$$\boldsymbol{x}(k+1|k), \boldsymbol{x}(k+2|k), \ldots, \boldsymbol{x}(k+N_p|k),$$

which can then be used to compute predicted sequence of the plant's outputs,

$$\boldsymbol{y}(k+1|k), \boldsymbol{y}(k+2|k), \ldots, \boldsymbol{y}(k+N_p|k). \tag{4}$$

Using the above information, we compute the control sequence (3) and then apply $\boldsymbol{u}[k]$ to the plant to generate $\boldsymbol{x}[k+1]$. We repeat the process again, using $\boldsymbol{x}[k+1]$ as an initial condition to compute $\boldsymbol{u}[k+1]$, and so on.

We now present an approach to construct $\boldsymbol{u}[k]$ given $\boldsymbol{x}[k]$. Using the plant model parameters and the measurement of $\boldsymbol{x}[k]$ we evaluate the states over the prediction horizon

successively applying the recursion formula (1) to obtain,

$$\begin{aligned}
\boldsymbol{x}(k+1|k) &= \boldsymbol{\Phi}\boldsymbol{x}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k] \\
\boldsymbol{x}(k+2|k) &= \boldsymbol{\Phi}\boldsymbol{x}(k+1|k) + \boldsymbol{\Gamma}\boldsymbol{u}[k+1] \\
&= \boldsymbol{\Phi}^2\boldsymbol{x}[k] + \boldsymbol{\Phi}\boldsymbol{\Gamma}\boldsymbol{u}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k+1] \\
&\vdots \\
\boldsymbol{x}(k+N_p|k) &= \boldsymbol{\Phi}^{N_p}\boldsymbol{x}[k] + \boldsymbol{\Phi}^{N_p-1}\boldsymbol{\Gamma}\boldsymbol{u}[k] + \cdots + \boldsymbol{\Gamma}\boldsymbol{u}[k+N_p-1]
\end{aligned}$$

We represent the above set of equations in the form,

$$\begin{bmatrix} \boldsymbol{x}(k+1|k) \\ \boldsymbol{x}(k+2|k) \\ \vdots \\ \boldsymbol{x}(k+N_p|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^2 \\ \vdots \\ \boldsymbol{\Phi}^{N_p} \end{bmatrix} \boldsymbol{x}[k] + \begin{bmatrix} \boldsymbol{\Gamma} & & & \\ \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Gamma} & & \\ \vdots & & \ddots & \\ \boldsymbol{\Phi}^{N_p-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Gamma} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_p-1] \end{bmatrix}. \quad (5)$$

We wish to design a controller that would force the plant output, $\boldsymbol{y}$, to track a given reference signal, $\boldsymbol{r}$. Using (2) and (5), we compute the sequence of predicted outputs (4),

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{y}(k+1|k) \\ \boldsymbol{y}(k+2|k) \\ \vdots \\ \boldsymbol{y}(k+N_p|k) \end{bmatrix} &= \begin{bmatrix} \boldsymbol{C}\boldsymbol{x}(k+1|k) \\ \boldsymbol{C}\boldsymbol{x}(k+2|k) \\ \vdots \\ \boldsymbol{C}\boldsymbol{x}(k+N_p|k) \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{C}\boldsymbol{\Phi} \\ \boldsymbol{C}\boldsymbol{\Phi}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_p} \end{bmatrix} \boldsymbol{x}[k] \quad (6) \\
&\quad + \begin{bmatrix} \boldsymbol{C}\boldsymbol{\Gamma} & & & \\ \boldsymbol{C}\boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{C}\boldsymbol{\Gamma} & & \\ \vdots & & \ddots & \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_p-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{C}\boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{C}\boldsymbol{\Gamma} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_p-1] \end{bmatrix}. \quad (7)
\end{aligned}$$

We write the above compactly as

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{x}[k] + \boldsymbol{Z}\boldsymbol{U}, \quad (8)$$

where

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}(k+1|k) \\ \boldsymbol{y}(k+2|k) \\ \vdots \\ \boldsymbol{y}(k+N_p|k) \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_p-1] \end{bmatrix},$$

and

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{C\Phi} \\ \boldsymbol{C\Phi}^2 \\ \vdots \\ \boldsymbol{C\Phi}^{N_p} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{C\Gamma} & & & \\ \boldsymbol{C\Phi\Gamma} & \boldsymbol{C\Gamma} & & \\ \vdots & & \ddots & \\ \boldsymbol{C\Phi}^{N_p-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{C\Phi\Gamma} & \boldsymbol{C\Gamma} \end{bmatrix}. \tag{9}$$

Suppose now that we wish to construct a control sequence, $\boldsymbol{u}[k], \ldots, \boldsymbol{u}[k+N_p-1]$, that would minimize the cost function

$$J(\boldsymbol{U}) = \frac{1}{2}\left(\boldsymbol{r}_p - \boldsymbol{Y}\right)^\top \boldsymbol{Q}\left(\boldsymbol{r}_p - \boldsymbol{Y}\right) + \frac{1}{2}\boldsymbol{U}^\top \boldsymbol{R}\boldsymbol{U}, \tag{10}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^\top \succeq 0$ and $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0$ are real symmetric positive semi-definite and positive-definite weight matrices, respectively. The multiplying scalar, $1/2$, is just to make subsequent manipulations cleaner. Finally, the vector $\boldsymbol{r}_p$ consists of the values of the command signal at sampling times, $k+1, k+2, \ldots, k+N_p$, that is,

$$\boldsymbol{r}_p = \left.\begin{bmatrix} \boldsymbol{r}[k+1] \\ \boldsymbol{r}[k+2] \\ \vdots \\ \boldsymbol{r}[k+N_p] \end{bmatrix}\right\} N_p \text{ times}$$

The selection of the weight matrices, $\boldsymbol{Q}$ and $\boldsymbol{R}$ reflects our control objective to keep the tracking error $\|\boldsymbol{r}_p - \boldsymbol{Y}\|$ "small" using the control actions that are "not too large."

We first apply the first-order necessary condition (FONC) test to $J(\boldsymbol{U})$,

$$\frac{\partial J}{\partial \boldsymbol{U}} = \boldsymbol{0}^\top.$$

Then, we solve the above equation for $\boldsymbol{U} = \boldsymbol{U}^*$, where

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{U}} &= -\left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}[k] - \boldsymbol{Z}\boldsymbol{U}\right)^\top \boldsymbol{Q}\boldsymbol{Z} + \boldsymbol{U}^\top \boldsymbol{R} \\ &= \boldsymbol{0}^\top. \end{aligned}$$

Performing simple manipulations yields

$$-\boldsymbol{r}_p^\top \boldsymbol{Q}\boldsymbol{Z} + \boldsymbol{x}[k]^\top \boldsymbol{W}^\top \boldsymbol{Q}\boldsymbol{Z} + \boldsymbol{U}^\top \boldsymbol{Z}^\top \boldsymbol{Q}\boldsymbol{Z} + \boldsymbol{U}^\top \boldsymbol{R} = \boldsymbol{0}^\top.$$

Applying the transposition operation to both sides of the above equation and rearranging terms, we obtain

$$\left(\boldsymbol{R} + \boldsymbol{Z}^\top \boldsymbol{Q}\boldsymbol{Z}\right)\boldsymbol{U} = \boldsymbol{Z}^\top \boldsymbol{Q}\left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}[k]\right).$$

Note that the matrix $\left(\boldsymbol{R}+\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{Z}\right)$ is invertible, and in fact, positive definite because $\boldsymbol{R}=\boldsymbol{R}^\top\succ 0$ and $\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{Z}$ is also symmetric and at least positive semi-definite. Hence, $\boldsymbol{U}$ that satisfies the FONC is

$$\boldsymbol{U}^* = \left(\boldsymbol{R}+\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top\boldsymbol{Q}\left(\boldsymbol{r}_p-\boldsymbol{W}\boldsymbol{x}[k]\right). \tag{11}$$

Now, applying the second derivative test to $J(\boldsymbol{U})$, which we refer to as the second-order sufficiency condition (SONC), we obtain

$$\begin{aligned}\frac{\partial^2 J}{\partial \boldsymbol{U}^2} &= \boldsymbol{R}+\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{Z}\\ &\succ 0,\end{aligned}$$

which implies that $\boldsymbol{U}^*$ is a strict minimizer of J.

Using (11), we compute $\boldsymbol{u}[k]$,

$$\begin{aligned}\boldsymbol{u}[k] &= \overbrace{\begin{bmatrix}\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O}\end{bmatrix}}^{N_p \text{ block matrices}}\left(\boldsymbol{R}+\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top\boldsymbol{Q}\left(\boldsymbol{r}_p-\boldsymbol{W}\boldsymbol{x}[k]\right)\\ &= \boldsymbol{K}_{\boldsymbol{r}}\boldsymbol{r}_p-\boldsymbol{K}_{\boldsymbol{x}}\boldsymbol{x}[k],\end{aligned} \tag{12}$$

where

$$\boldsymbol{K}_{\boldsymbol{r}} = \begin{bmatrix}\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O}\end{bmatrix}\left(\boldsymbol{R}+\boldsymbol{Z}^\top\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^\top\boldsymbol{Q}, \quad \text{and} \quad \boldsymbol{K}_{\boldsymbol{x}} = \boldsymbol{K}_{\boldsymbol{r}}\boldsymbol{W}.$$

2 MPC With Constraints

An attractive feature of the model-based predictive control approach is that a control engineer can incorporate different types of constraints on the control action. We consider three types of such constraints.

2.1 Constraints on the Control Action Magnitude

Hard constraints on the control action magnitude at the time sampling k have the form

$$u_i^{\min} \le u_i[k] \le u_i^{\max}, \quad i=1,2\ldots,m \tag{13}$$

We now express constraints on the control action magnitude over the whole prediction horizon in term of $\boldsymbol{U}$. Let

$$\boldsymbol{u}^{\min} = \begin{bmatrix}u_1^{\min} & \cdots & u_m^{\min}\end{bmatrix}^\top \quad \text{and} \quad \boldsymbol{u}^{\max} = \begin{bmatrix}u_1^{\max} & \cdots & u_m^{\max}\end{bmatrix}^\top.$$

Then, we can express (13) as

$$\boldsymbol{u}^{\min} \leq \boldsymbol{u}[k] \leq \boldsymbol{u}^{\max}. \tag{14}$$

The above can be equivalently represented as

$$\begin{bmatrix} -\boldsymbol{I}_m \\ \boldsymbol{I}_m \end{bmatrix} \boldsymbol{u}[k] \leq \begin{bmatrix} -\boldsymbol{u}^{\min} \\ \boldsymbol{u}^{\max} \end{bmatrix}. \tag{15}$$

Using the above approach we can represent constraints on the rate of change of the control action over the whole prediction horizon, N_p, in terms of $\boldsymbol{U}$, by augmenting the above inequality to incorporate constraints for the remaining sampling times. That is, if the above constraints are imposed on the control action magnitude for all sampling times within the prediction horizon, then this can be expressed in terms of $\boldsymbol{U}$ as

$$\begin{bmatrix} -\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & -\boldsymbol{I}_m & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_m & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \vdots & & & & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & -\boldsymbol{I}_m & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{I}_m & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} & -\boldsymbol{I}_m \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{I}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_p-2] \\ \boldsymbol{u}[k+N_p-1] \end{bmatrix} \leq \begin{bmatrix} -\boldsymbol{u}^{\min} \\ \boldsymbol{u}^{\max} \\ -\boldsymbol{u}^{\min} \\ \boldsymbol{u}^{\max} \\ \vdots \\ -\boldsymbol{u}^{\min} \\ \boldsymbol{u}^{\max} \\ -\boldsymbol{u}^{\min} \\ \boldsymbol{u}^{\max} \end{bmatrix}. \tag{16}$$

On the other hand, if the constraints on the control action magnitudel are imposed only on the first component of $\boldsymbol{U}$, then we express this in terms of $\boldsymbol{U}$ as

$$\begin{bmatrix} -\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} -\boldsymbol{u}^{\min} \\ \boldsymbol{u}^{\max} \end{bmatrix}.$$

2.2 Constraints on the Rate of Change of the Control Action

Hard constraints on the rate of change of the control signal can be expressed as

$$\Delta u_i^{\min} \leq \Delta u_i[k] \leq \Delta u_i^{\max}, \quad i = 1, 2, \ldots, m. \tag{17}$$

We now express constraints on the rate of change of the control over the whole prediction horizon in terms of $\boldsymbol{U}$. To accomplish our goal, we first note that

$$\begin{aligned} \Delta\boldsymbol{u}[k] &= \boldsymbol{u}[k] - \boldsymbol{u}[k-1] \\ \Delta\boldsymbol{u}[k+1] &= \boldsymbol{u}[k+1] - \boldsymbol{u}[k] \\ &\vdots \\ \Delta\boldsymbol{u}[k+N_p-1] &= \boldsymbol{u}[k+N_p-1] - \boldsymbol{u}[k+N_p-2]. \end{aligned}$$

We represent the above in a matrix format,

$$\begin{bmatrix} \Delta \boldsymbol{u}[k] \\ \Delta \boldsymbol{u}[k+1] \\ \vdots \\ \Delta \boldsymbol{u}[k+N_p-2] \\ \Delta \boldsymbol{u}[k+N_p-1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ -\boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \vdots & & & & \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{I}_m & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & -\boldsymbol{I}_m & \boldsymbol{I}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_p-2] \\ \boldsymbol{u}[k+N_p-1] \end{bmatrix} - \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{O} \\ \vdots \\ \boldsymbol{O} \\ \boldsymbol{O} \end{bmatrix} \boldsymbol{u}[k-1]. \tag{18}$$

Let

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_p-1] \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ -\boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \vdots & & & & \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{I}_m & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & -\boldsymbol{I}_m & \boldsymbol{I}_m \end{bmatrix}, \quad \text{and} \quad \boldsymbol{K} = \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{O} \\ \vdots \\ \boldsymbol{O} \\ \boldsymbol{O} \end{bmatrix}.$$

Then, we can represent (18) as

$$\Delta \boldsymbol{U} = \boldsymbol{G}\boldsymbol{U} - \boldsymbol{K}\boldsymbol{u}[k-1]. \tag{19}$$

Suppose now that we are faced with constructing a control action subject to the constraints,

$$\Delta \boldsymbol{U}^{\min} \leq \Delta \boldsymbol{U} \leq \Delta \boldsymbol{U}^{\max}.$$

The above constraints can be equivalently represented as

$$\begin{bmatrix} -\Delta \boldsymbol{U} \\ \Delta \boldsymbol{U} \end{bmatrix} \leq \begin{bmatrix} -\Delta \boldsymbol{U}^{\min} \\ \Delta \boldsymbol{U}^{\max} \end{bmatrix},$$

Taking into account (19), we write the above as

$$\begin{bmatrix} -\boldsymbol{G} \\ \boldsymbol{G} \end{bmatrix} \boldsymbol{U} + \begin{bmatrix} \boldsymbol{K} \\ -\boldsymbol{K} \end{bmatrix} \boldsymbol{u}[k-1] \leq \begin{bmatrix} -\Delta \boldsymbol{U}^{\min} \\ \Delta \boldsymbol{U}^{\max} \end{bmatrix}. \tag{20}$$

The above, in turn, can be represented as

$$\begin{bmatrix} -\boldsymbol{G} \\ \boldsymbol{G} \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} -\Delta \boldsymbol{U}^{\min} - \boldsymbol{K}\boldsymbol{u}[k-1] \\ \Delta \boldsymbol{U}^{\max} + \boldsymbol{K}\boldsymbol{u}[k-1] \end{bmatrix}. \tag{21}$$

In a special case, when a control designer elects to impose constraints only on the first component of $\Delta \boldsymbol{U}$, that is, on $\Delta \boldsymbol{u}[k]$ only, then this scenario is expressed in terms of $\boldsymbol{U}$ as

$$\begin{bmatrix} -\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} -\Delta \boldsymbol{u}^{\min} - \boldsymbol{u}[k-1] \\ \Delta \boldsymbol{u}^{\max} + \boldsymbol{u}[k-1] \end{bmatrix}, \tag{22}$$

where $\Delta \boldsymbol{u}^{\min}$ and $\Delta \boldsymbol{u}^{\max}$ are constraints on the change of the control $\Delta \boldsymbol{u}[k]$, that is

$$\boldsymbol{u}^{\min} \leq \boldsymbol{u}[k] \leq \boldsymbol{u}^{\max}.$$

2.3 Constraints on the Plant Output

Recall from (8) that the predicted plant output is, $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{x}[k] + \boldsymbol{Z}\boldsymbol{U}$. Suppose now that the following constraints are imposed on the predicted plant's output,

$$\boldsymbol{Y}^{\min} \leq \boldsymbol{Y} \leq \boldsymbol{Y}^{\max}.$$

We represent the above as

$$\begin{bmatrix} -\boldsymbol{Y} \\ \boldsymbol{Y} \end{bmatrix} \leq \begin{bmatrix} -\boldsymbol{Y}^{\min} \\ \boldsymbol{Y}^{\max} \end{bmatrix}.$$

Taking into the account the expression for $\boldsymbol{Y}$ given by (8), we represent the above as

$$\begin{bmatrix} -\boldsymbol{Z} \\ \boldsymbol{Z} \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} -\boldsymbol{Y}^{\min} + \boldsymbol{W}\boldsymbol{x}[k] \\ \boldsymbol{Y}^{\max} - \boldsymbol{W}\boldsymbol{x}[k] \end{bmatrix}. \tag{23}$$

Example 1 We formulate the optimization problem that the model predictive controller is to solve at each sampling period for a discrete dynamic system model,

$$\begin{aligned} \boldsymbol{x}[k+1] &= \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \boldsymbol{x}[k] + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u[k] \\ y[k] &= \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x}[k], \end{aligned}$$

The system output is to track the unit step. The prediction horizon is $N_p = 2$. The constraints on the control signal are to be imposed only on the first component of $\boldsymbol{U}$ and $\Delta\boldsymbol{U}$, respectively, and they have the form:

$$-3 \leq u[k] \leq 5 \quad \text{and} \quad -2 \leq \Delta u[k] \leq 1.$$

Because $N_p = 2$,

$$\boldsymbol{U} = \begin{bmatrix} u[k] \\ u[k+1] \end{bmatrix} \quad \text{and} \quad \Delta\boldsymbol{U} = \begin{bmatrix} \Delta u[k] \\ \Delta u[k+1] \end{bmatrix}.$$

The constraints on the control magnitude on the first component of $\boldsymbol{U}$ can be expressed as

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

The constraints on the first component of $\Delta\boldsymbol{U}$ have the form

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} 2 - u[k-1] \\ 1 + u[k-1] \end{bmatrix}.$$

Let

$$\boldsymbol{Y} = \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \end{bmatrix} = \boldsymbol{W}\boldsymbol{x}[k] + \boldsymbol{Z}\boldsymbol{U} \quad \text{and} \quad \boldsymbol{r}_p = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

where $\boldsymbol{Y}$ is the vector containing predicted plant outputs over the prediction horizon and $\boldsymbol{r}_p$ is the vector of the reference signals over the prediction horizon. Then, the optimization problem has the form

$$\min \tfrac{1}{2} \left(\boldsymbol{r}_p - \boldsymbol{Y}\right)^\top \boldsymbol{Q} \left(\boldsymbol{r}_p - \boldsymbol{Y}\right) + \tfrac{1}{2}\boldsymbol{U}^\top \boldsymbol{R}\boldsymbol{U}$$

subject to

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 1 & 0 \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} 3 \\ 5 \\ 2 - u[k-1] \\ 1 + u[k-1] \end{bmatrix},$$

where $\boldsymbol{Q} = \boldsymbol{Q}^\top \succeq 0$ and $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0$. For example, $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{I}_2$.

Example 2 We illustrate the design of the discrete MPC subject to the constraints on the control input on a plant modeled by the following continuous state-space model,

$$\left.\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} -0.1 & -3.0 \\ 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \\ y &= \begin{bmatrix} 0 & 10 \end{bmatrix} \boldsymbol{x}. \end{aligned}\right\} \tag{24}$$

The system with the transfer function as the above system was used by Wang [7, p. 70] to test her discrete MPC designs. We use different optimizer than Wang. The reader can compare the results of two different MPC synthesis methods. The MPC controller's objective is to force the plant's output to track the unit step.

We proceed to generate the discrete state-space model employing the MATLAB's command `c2dm`. We use the sampling time $h = 0.1$. The command input is the unit step. The weight matrices in the cost function (10) are $\boldsymbol{R} = 0.01\boldsymbol{I}_3$ and $\boldsymbol{Q} = \boldsymbol{I}_3$. The prediction horizon is $N_p = 3$. Our objective is to design a discrete-MPC subject to the constraints on the control of the form,

$$-0.3 \leq u[k] \leq 0.5$$

First, we consider the case when the constraints are imposed only on the first component of $\boldsymbol{U}(k)$. The constraints take the form of (22). Thus, in our example, the constraints can be written as,

$$\begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u[k] \\ u[k+1] \\ u[k+2] \end{bmatrix} \leq \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}.$$

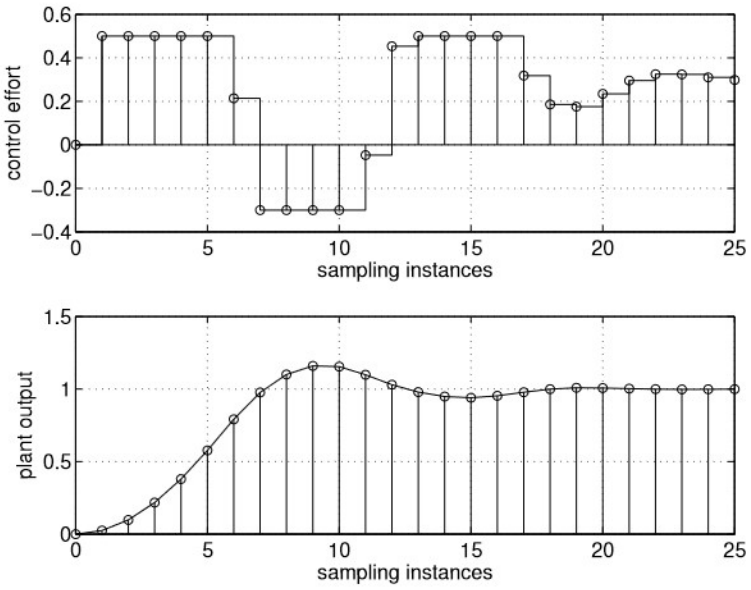

Figure 1: Plots of control effort and the plant's output versus time of the closed-loop system of Example 2 for the case when the constraint was imposed only on the first component of $\boldsymbol{U}$.

We employed the first-order Lagrangian algorithm for inequality constraints, where $\alpha_i = \beta_i = 0.005$. We set zero initial conditions. Plots of the control effort and the plant's output are shown in Figure 1. We wish to emphasize that the simulations were performed for the discretized plant. Thus, the control and output were evaluated at the sampling instances and marked on the plots using circles. For the purpose of visualization the control action was presented in the form of a staircase function while the plant output values were connected by straight lines.

Next we tested the discrete MPC where the constraints, $-0.3 \leq u[k] \leq 0.5$, were imposed on all the elements of $\boldsymbol{U}$. We use (16) to express the constraints, where

$$\begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u[k] \\ u[k+1] \\ u[k+2] \end{bmatrix} \leq \begin{bmatrix} 0.3 \\ 0.5 \\ 0.3 \\ 0.5 \\ 0.3 \\ 0.5 \end{bmatrix}.$$

Example 3 In this example, we apply the discrete MPC controller from Example 2 to the continuous plant modeled by (24). The MPC controller's objective is to force the plant's

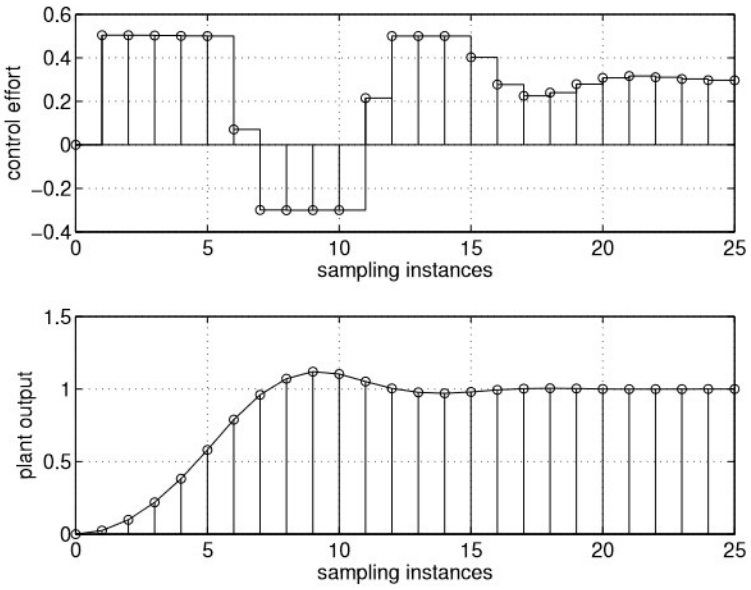

Figure 2: Plots of control effort and the plant's output versus time of the closed-loop system of Example 2 for the case when the constraints were imposed on all the components of $\boldsymbol{U}$.

output to track the unit step. The zero-order hold is applied to the discrete MPC controller's output sequence resulting in a piece-wise constant input to the continuous plant model. In Figure 3, we show plots of the plant output as well as the control effort versus time. As in the previous example, the prediction horizon is $N_p = 3$ and the constraints on the control are

$$-0.3 \leq u[k] \leq 0.5\,.$$

3 Stability of Basic MPC

In general, a system controlled by an MPC is not guaranteed to be stable. In this section, we prove stability of a basic MPC algorithm that we present now.

Algorithm 1 Basic MPC Algorithm

1. Given $\boldsymbol{x}[k] = \boldsymbol{x}[k|k]$. Let $\mathcal{U}$ denote the set of admissible controllers

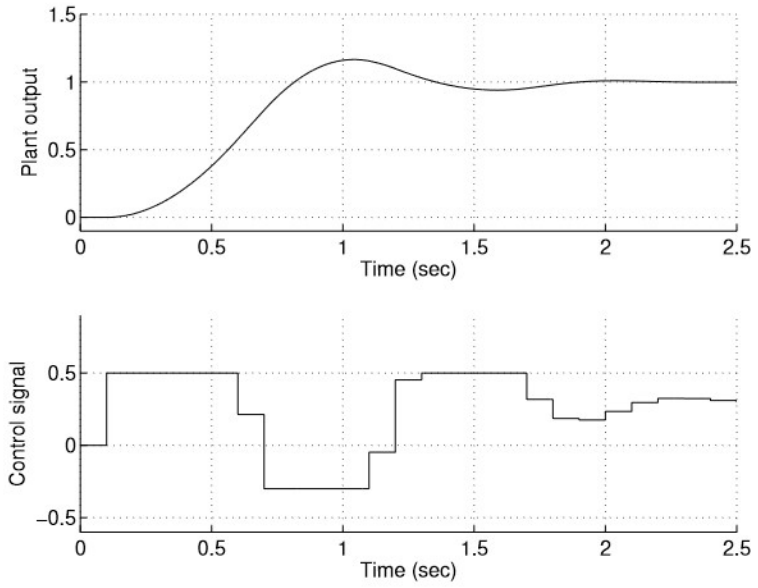

Figure 3: Plots of control effort and the plant's output versus time of the closed-loop system of Example 3 for the case when the constraints were imposed only on the first component of $\boldsymbol{U}$.

2. Solve

$$\min_{\boldsymbol{u}[k],\boldsymbol{u}[k+1],\ldots,\boldsymbol{u}[k+N_p-1]} \sum_{j=k}^{k+N_p-1} \boldsymbol{x}[j+1]^\top \boldsymbol{Q}\boldsymbol{x}[j+1] + \boldsymbol{u}[j]^\top \boldsymbol{R}\boldsymbol{u}[j]$$

subject to

$$\boldsymbol{x}[j+1] = \boldsymbol{\Phi}\boldsymbol{x}[j] + \boldsymbol{\Gamma}\boldsymbol{u}[j], \quad \boldsymbol{u}[j] \in \mathcal{U}$$

to obtain the admissible control sequence

$$\boldsymbol{u}[k], \boldsymbol{u}[k+1], \ldots, \boldsymbol{u}[k+N_p-1]$$

3. Apply $\boldsymbol{u}[k] = \boldsymbol{u}[k|k]$

4. Set $k := k+1$

5. Go to Step 1

Our objective is to prove the stability of the above basic MPC algorithm for the simple case when the plant has the form

$$\boldsymbol{x}[k+1] = \boldsymbol{\Phi}\boldsymbol{x}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k]. \tag{25}$$

We now state and prove a theorem concerned with the stability of the plant modeled by (25) driven by the control sequence computed at each time step using the Basic MPC algorithm.

Theorem 1 *If in the basic MPC algorithm, the control sequence can be selected so that for all $k \geq 0$,*

$$\boldsymbol{x}(k+N_p|k) = \mathbf{0},$$

then the trajectory of the plant (25) *driven by the basic MPC algorithm is converging to the origin.*

Proof For a given initial $\boldsymbol{x}[k]$, we use the basic MPC algorithm to obtain $\boldsymbol{u}[k]$. We then compute $\boldsymbol{x}[k+1]$ and repeat the process. The performance index $J[k+1]$ for the initial state $\boldsymbol{x}[k+1]$ is

$$J[k+1] = \sum_{j=k+1}^{k+N_p} \boldsymbol{x}[j+1]^\top \boldsymbol{Q}\boldsymbol{x}[j+1] + \boldsymbol{u}[j]^\top \boldsymbol{R}\boldsymbol{u}[j].$$

We denote the optimal value of the cost $J[k]$ as $J^*[k]$. Minimizing $J[k]$ gives the optimal control sequence

$$\boldsymbol{u}^*[k], \boldsymbol{u}^*[k+1], \ldots, \boldsymbol{u}^*[k+N_p-1].$$

A feasible, not necessarily optimal, control sequence for $k+1$ can be formed using the elements of the previous control sequence and adding the element $\boldsymbol{u}[k+N_p] = \mathbf{0}$ to it to obtain

$$\boldsymbol{u}^*[k+1], \boldsymbol{u}^*[k+2], \ldots, \boldsymbol{u}^*[k+N_p-1], \mathbf{0}.$$

By assumption, we have $\boldsymbol{x}(k+N_p|k) = \mathbf{0}$. Hence we will also have $\boldsymbol{x}(k+N_p+1|k+1) = \mathbf{0}$. We next manipulate $J[k+1]$ so that we express it as a function of $J^*[k]$. To this end, we add and subtract to the right hand side of $J[k+1]$ two terms $\boldsymbol{x}^*[k+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[k+1] + \boldsymbol{u}^*[k]^\top \boldsymbol{R}\boldsymbol{u}^*[k]$ to obtain

$$\begin{aligned}
J[k+1] &= \sum_{j=k+1}^{k+N_p-1} \boldsymbol{x}^*[j+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[j+1] + \boldsymbol{u}^*[j]^\top \boldsymbol{R}\boldsymbol{u}^*[j] \\
&\quad + \boldsymbol{x}[k+N_p+1]^\top \boldsymbol{Q}\boldsymbol{x}[k+N_p+1] + \boldsymbol{u}[k+N_p]^\top \boldsymbol{R}\boldsymbol{u}[k+N_p] \\
&\quad + \boldsymbol{x}^*[k+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[k+1] + \boldsymbol{u}^*[k]^\top \boldsymbol{R}\boldsymbol{u}^*[k] - \boldsymbol{x}^*[k+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[k+1] - \boldsymbol{u}^*[k]^\top \boldsymbol{R}\boldsymbol{u}^*[k] \\
&= \sum_{j=k}^{k+N_p-1} \boldsymbol{x}^*[j+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[j+1] + \boldsymbol{u}^*[j]^\top \boldsymbol{R}\boldsymbol{u}^*[j] + \boldsymbol{x}[k+N_p+1]^\top \boldsymbol{Q}\boldsymbol{x}[k+N_p+1] \\
&\quad + \boldsymbol{u}[k+N_p]^\top \boldsymbol{R}\boldsymbol{u}[k+N_p] - \boldsymbol{x}^*[k+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[k+1] - \boldsymbol{u}^*[k]^\top \boldsymbol{R}\boldsymbol{u}^*[k] \\
&= J^*[k] + \boldsymbol{x}[k+N_p+1]^\top \boldsymbol{Q}\boldsymbol{x}[k+N_p+1] \\
&\quad + \boldsymbol{u}[k+N_p]^\top \boldsymbol{R}\boldsymbol{u}[k+N_p] - \boldsymbol{x}^*[k+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[k+1] - \boldsymbol{u}^*[k]^\top \boldsymbol{R}\boldsymbol{u}^*[k]. \qquad (26)
\end{aligned}$$

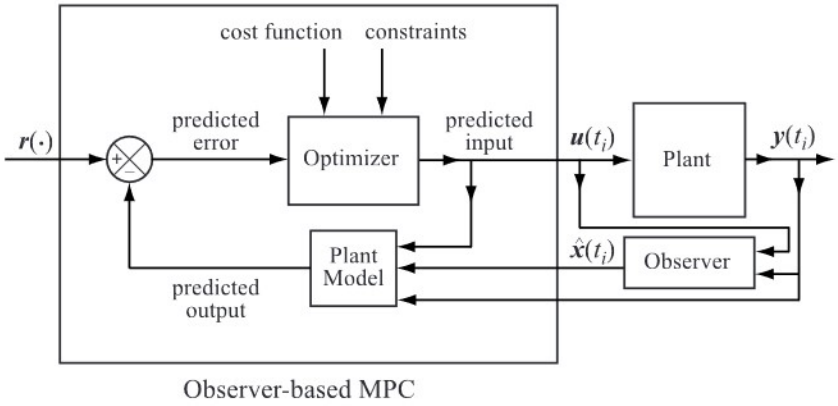

Figure 4: MPC implementation when the plant's state is not available.

By assumption, $\boldsymbol{x}(k+N_p|k) = \boldsymbol{0}$. Once we reached the origin, we set $\boldsymbol{u}[k+N_p] = \boldsymbol{0}$. Taking the above into account in (26) gives

$$\begin{aligned} J^*[k+1] &\leq J[k+1] \\ &= J^*[k] - \boldsymbol{x}^*[k+1]^\top \boldsymbol{Q}\boldsymbol{x}^*[k+1] - \boldsymbol{u}^*[k]^\top \boldsymbol{R}\boldsymbol{u}^*[k] \\ &< J^*[k] \end{aligned}$$

since $\boldsymbol{Q} = \boldsymbol{Q}^\top \succeq 0$ and $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0$. Thus $\Delta J^*[k] < 0$, that is, $J^*[k]$ is a Lyapunov function for the system driven by the basic MPC algorithm. Hence the system trajectory driven by the basic MPC algorithm is converging to the origin. The proof is complete.

□

Remark 1 The constraint $\boldsymbol{x}(k+N_p|k) = \boldsymbol{0}$ is artificial. The system state does not reach the origin at $k+N_p$. This constraint is continuously shifted forward in time, that is, we have a moving time horizon. This is one of the reasons the MPC is also referred to as the receding horizon control (RHC) algorithm.

If the plant state is not accessible, we need to use an observer in the controller implementation as shown in Figure 4.

Example 4 In this Example, we design an MPC controller for the nonlinear θ-r robot manipulator. We first perform Taylor linearization of the nonlinear model about a selected equilibrium pair $(\boldsymbol{z}_e, \boldsymbol{u}_e)$. We proceed as follows:

- we calculate $\boldsymbol{u}_e$ that yields the corresponding equilibrium state

$$\boldsymbol{z}_e = \begin{bmatrix} \pi/4 & 0 & 2 & 0 \end{bmatrix}^\top \tag{27}$$

and then Taylor linearize the model about the equilibrium pair $(\boldsymbol{z}_e, \boldsymbol{u}_e)$. We have

$$\boldsymbol{u}_e = \begin{bmatrix} 113.1371 \\ 21.2132 \end{bmatrix}.$$

We use MATLAB's `jacobian` and `subs` functions to perform linearization. The result is a linearized model of the form,

$$\begin{aligned} \frac{d}{dt}\Delta\boldsymbol{x} &= \begin{bmatrix} 0 & 1.0000 & 0 & 0 \\ 5.0449 & 0 & -0.9992 & 0 \\ 0 & 0 & 0 & 1.0000 \\ -6.9367 & 0 & 0 & 0 \end{bmatrix} \Delta\boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 0.0455 & 0 \\ 0 & 0 \\ 0 & 0.3333 \end{bmatrix} \Delta\boldsymbol{u} \\ \Delta\boldsymbol{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Delta\boldsymbol{x}. \end{aligned}$$

- We discretize the linearized model and use it as the basis for your MPC design. We select the sampling interval $h = 0.05$ and use the command

  ```
  [phi,gamma]=c2d(A,B,h)
  ```

  to obtain the matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Gamma}$.

- In this part, we design an MPC without any constraints and implement the controller on the nonlinear continuous model. Our MPC controller is to move the robot from some initial condition to the equilibrium given by (27). We present plots for the initial state

  $$\boldsymbol{x}(0) = \begin{bmatrix} -45^\circ & 0 & 0.2 & 0 \end{bmatrix}^\top.$$

  The initial position of the robot is depicted in Figure 5.

  Our target and the reference vector $\boldsymbol{r}_p$ are coded in MATLAB as

```
target=[pi/4 2]';
rp=kron(ones(Np,1),target)
```

  where N_p is the prediction horizon. We used $N_p = 25$. The weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are:

```
R = .05*eye(2*Np);
Q = 5*eye(2*Np);
```

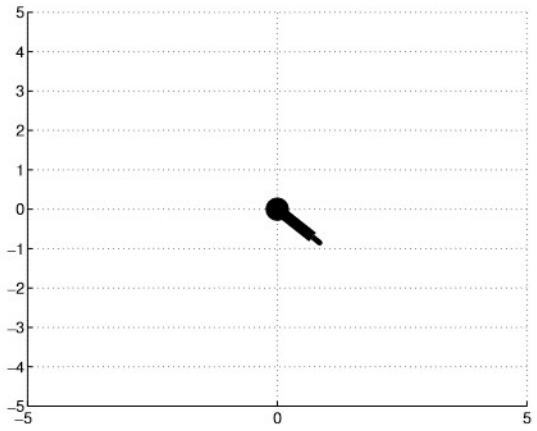

Figure 5: Initial position of the θ-r robot of Example 4.

The $\boldsymbol{W}$ and $\boldsymbol{Z}$ matrices given by (9) can be obtained as follows:

```
% Calculating W and Z

phi_a=[phi zeros(4,2); C*phi eye(2,2)];
gamma_a=[gamma;C*gamma];
C_a=[zeros(2,4) eye(2,2)];

W=[];
for i=1:Np
    W=[W;C_a*phi_a^i];
end

Z=zeros(Np*p,Np*m);
Z(1:p,1:m)=C_a*gamma_a;
temp=C_a*gamma_a;
for i=1:Np-1
    temp=[C_a*phi_a^i*gamma_a temp];
    Z(i*p+1:(i+1)*p,1:size(temp,2))=temp;
end
```

We next initialized the animation of the θ-r robot motion using the handle graphics commands and the forward Euler method.

```
% Animation initialization

x=[-45*pi/180 0 .2 0]';
```

```
u=[0; 0];
y = C*x;

axis([-5 5 -5 5])
set(gca,'Fontsize',font_size)
m_1_pos = [r1*cos(x(1));r1*sin(x(1))];
m_2_pos = [x(3)*cos(x(1));x(3)*sin(x(1))];
bar_1 = line('xdata',[0 m_1_pos(1)],'ydata',[0 ...
   m_1_pos(2)],'linewidth',10);
bar_2 = line('xdata',[m_1_pos(1) m_2_pos(1)],'ydata',...
    [m_1_pos(2) m_2_pos(2)],'linewidth',5,'erase','xor','color','r');
mass_1 = line('xdata',m_1_pos(1),'ydata',m_1_pos(2),...
    'marker','o','markersize',1,'erasemode','none');
mass_2 = line('xdata',m_2_pos(1),'ydata',m_2_pos(2),...
    'marker','o','markersize',4,'markeredgecolor','r','markerfacecolor','r');
hinge = line('xdata',0,'ydata',0,'marker','o','markersize',20,...
    'markeredgecolor','k','markerfacecolor','k');
grid
```

The animation of the closed-loop system driven by the unconstrained MPC controller has the form,

```
% Animation

t=0;
time=[t];
r_telescop=[x(3)];
angle=[x(1)];
torque=[u(1)];
force=[u(2)];
while t<tf
    t = t+h;
    pause(0.03)
    numf=(-2*m2*x(2)*x(3)*x(4)-g*cos(x(1))*(m1*r1+m2*x(3))+u(1));
    denf=(m1*r1^2+m2*x(3)^2);
    f2=numf/denf;
    xdot=[x(2);f2;x(4);x(2)^2*x(3)-g*sin(x(1))+u(2)/m2];
    xnew = x + xdot*h;
    y=C*xnew;
    xa = [xnew-x;y];
    x = xnew;
    delta_u = [eye(2) zeros(2,2*Np-2)]*inv(R+Z'*Q*Z)*Z'*Q*(rp-W*xa);
```

```
    u = u + Δ_u;

    m_1_pos = [r1*cos(x(1));r1*sin(x(1))];
    m_2_pos = [(r1+x(3))*cos(x(1));(r1+x(3))*sin(x(1))];

    set(bar_1,'xdata',[0 m_1_pos(1)],'ydata',[0 m_1_pos(2)]);
    set(bar_2,'xdata',[m_1_pos(1) m_2_pos(1)],'ydata',[m_1_pos(2) ...
        m_2_pos(2)]);
    set(mass_1,'xdata',m_1_pos(1),'ydata',m_1_pos(2));
    set(mass_2,'xdata',m_2_pos(1),'ydata',m_2_pos(2));
    set(hinge,'xdata',0,'ydata',0);
    drawnow;
%    t = t+h;
    time=[time t];
    r_telescop=[r_telescop x(3)];
    angle=[angle x(1)];
    torque=[torque u(1)];
    force=[force u(2)];
end
```

In the above we collect data to obtain plots. Plots can be obtained using the following commands,

```
% Plots

figure
plot(time,r_telescop,time,2*ones(length(time)),'r--')
grid
xlabel('Time (sec)','Fontsize',font_size)
ylabel('r (m)','Fontsize',font_size)
set(gca,'Fontsize',font_size)
figure
plot(time,angle.*180/pi,time,45*ones(length(time)),'r--')
grid
xlabel('Time (sec)','Fontsize',font_size)
ylabel('angle (rad)','Fontsize',font_size)
set(gca,'Fontsize',font_size)
figure
plot(time,torque)
grid
xlabel('Time (sec)','Fontsize',font_size)
ylabel('Torque (Nm)','Fontsize',font_size)
```

```
set(gca,'Fontsize',font_size)
figure
plot(time,force)
grid
xlabel('Time (sec)','Fontsize',font_size)
ylabel('Force (N)','Fontsize',font_size)
set(gca,'Fontsize',font_size)
```

- In this part, we impose constraints on r of the form,

$$1 \leq r \leq 2\,\mathrm{m}.$$

This means that we need to re-design our MPC. To take into the account the above constraints, we modified our animation part as follows:

```
% Animation of the constrained MPC

t=0;
time=[t];
r_telescop=[x(3)];
angle=[x(1)];
torque=[u(1)];
force=[u(2)];
alpha=0.01;
beta=alpha;
Δ_U=zeros(2*Np,1);
mu=zeros(4*Np,1);
% r_low=1;
% r_high=2;
y_low=[-10*pi 1]';
y_high=[10*pi 2]';
Y_min=kron(ones(Np,1),y_low);
Y_max=kron(ones(Np,1),y_high);
Dg=[-Z; Z];
while t<tf
    t = t+h;
    numf=(-2*m2*x(2)*x(3)*x(4)-g*cos(x(1))*(m1*r1+m2*x(3))+u(1));
    denf=(m1*r1^2+m2*x(3)^2);
    f2=numf/denf;
    xdot=[x(2);f2;x(4);x(2)^2*x(3)-g*sin(x(1))+u(2)/m2];
    xnew = x + xdot*h;
    y=C*xnew;
```

```
    xa = [xnew-x;y];
    x = xnew;

    for i = 1:200
%   J=0.5*(rp-W*xa-Z*Δ_U)'*Q*(rp-W*xa-Z*Δ_U)+Δ_U'*R*Δ_U;
%   J is the cost function

      gf=-Z'*Q'*rp + Z'*Q'*W*xa + Z'*Q'*Z*Δ_U + R'*Δ_U;
%   gf is the gradient of J

      g_con=[-Z; Z]*Δ_U - [-Y_min + W*xa; Y_max - W*xa];
%   g_con is the function of the constraints

%   Dg=[-Z; Z];
      Δ_U = Δ_U - alpha*(gf+Dg'*mu);
      mu=max(mu + beta*g_con,0);
    end

    Δ_u = [eye(2) zeros(2,2*Np-2)]*Δ_U;
    u = u + Δ_u;

    m_1_pos = [r1*cos(x(1));r1*sin(x(1))];
    m_2_pos = [x(3)*cos(x(1));x(3)*sin(x(1))];

    set(bar_1,'xdata',[0 m_1_pos(1)],'ydata',[0 m_1_pos(2)]);
    set(bar_2,'xdata',[m_1_pos(1) m_2_pos(1)],'ydata',[m_1_pos(2) ...
       m_2_pos(2)]);
    set(mass_1,'xdata',m_1_pos(1),'ydata',m_1_pos(2));
    set(mass_2,'xdata',m_2_pos(1),'ydata',m_2_pos(2));
    set(hinge,'xdata',0,'ydata',0);

    drawnow;
%    t = t+h;
    time=[time t];
    r_telescop=[r_telescop x(3)];
    angle=[angle x(1)];
    torque=[torque u(1)];
    force=[force u(2)];
end
```

- A Luenberger asymptotic observer was designed for the linearized system. We selected the observer's gain, $\boldsymbol{L}$, using the following commands,

```
poles_ob=[-7+i -7-i -7.5+i -7.5-i];
L=place(A',C',poles_ob)';
```

The combined MPC controller-observer compensator was synthesized to drive the nonlinear continuous model of the θ-r robot. The animation of the robot motion driven by the compensator was implemented as follows:

```
% Animation of the  MPC controller-observer compensator

ye=C*xe;
while t<tf
    t = t+h;
    numf=(-2*m2*x(2)*x(3)*x(4)-g*cos(x(1))*(m1*r1+m2*x(3))+u(1));
    denf=(m1*r1^2+m2*x(3)^2);
    f2=numf/denf;
    xdot=[x(2);f2;x(4);x(2)^2*x(3)-g*sin(x(1))+u(2)/m2];
    xnew = x + xdot*h;
    y=C*xnew;
    x = xnew;
    zdot=(A-L*C)*z+B*(u-ue)+L*(y-ye);
    z_new=z+zdot*h;
    za=[z_new-z;y];
    z=z_new;
    for i = 1:1500
        gf=-Z'*Q*rp + Z'*Q*W*za + Z'*Q*Z*Δ_U + R*Δ_U;
        Dg=[-Z; Z];
        g_con=[-Z; Z]*Δ_U - [-Y_min + W*za; Y_max - W*za];
        Δ_U = Δ_U - alpha*(gf+Dg'*mu);
        mu=max(mu + beta*g_con,0);
    end

    Δ_u = [eye(2) zeros(2,2*Np-2)]*Δ_U;
%   Δ_u = Δ_U(1:2);
    u = u + Δ_u;
```

In all three simulations, the controllers were able to bring the robot to its final desired position shown in Figure 6. However, the transients were quite different. The reader is encouraged to compare the performance of the above three controllers.

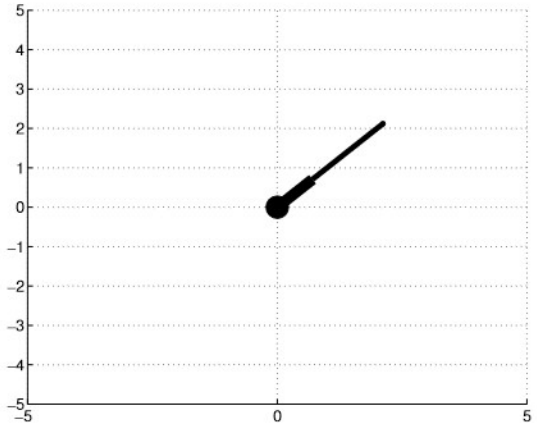

Figure 6: Final position of the θ-r robot of Example 4.

4 Notes

The key reason for huge popularity of the MPC approach is its ability to systematically take into account constraints thus allowing processes to operate at the limits of achievable performance [2]. For some impressive industrial applications of MPCs see [2, 1]. Nonlinear model predictive controllers are presented in [4, 6]. For a comprehensive treatment of the MPC, we recommend Maciejowski [3].

An insightful description of model predictive control is offered by Mayne et al. [5, pp. 789–790], where they write, "Model predictive control (MPC) or receding horizon control (RHC) is a form of control in which the current control action is obtained by solving *on-line*, at each sampling instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant. This is its main difference from conventional control which uses a pre-computed control law. With the clarity gained by hindsight, it can be recognized that the *raison d'être* for model predictive control is its ability to handle control problems where off-line computation of a control law is difficult or impossible although other features, such as its capability for controlling multivariable plants, were initially deemed more important. Nearly every application imposes constraints; actuators are naturally limited in the force (or equivalent) they can apply, safety limits states such as temperature, pressure and velocity and efficiency often dictates steady-state operation close to the boundary of the set of permissible states. The prevalence of hard constraints is accompanied by a dearth of control methods for handling them, despite a continuous demand from industry that has had, in their absence, to resort often to ad hoc methods. Model predictive control is one of few suitable methods, and this fact makes it an important tool for the control engineer, particularly in the process industries where plants

being controlled are sufficiently 'slow' to permit its implementation."

5 Practice Problems Prepared by Dr. Guisheng Zhai

PP 1 Consider the MPC optimization problem for the scalar discrete dynamical model

$$x[k+1] = x[k] + u[k]\,, \quad x[0] = 10$$

with the prediction horizon $N_p = 2$ and the performance index

$$J = \frac{1}{2}\sum_{i=1}^{N_p}\left(x(k+i|k)^2 + u[k+i-1]^2\right).$$

Solve for $u[0]$ and $u[1]$ at $k = 0$.

PP 2 Consider the MPC optimization problem for the discrete dynamical model

$$\boldsymbol{x}[k+1] = \boldsymbol{\Phi}\boldsymbol{x}[k] + \boldsymbol{\Gamma}u[k]\,, \quad \boldsymbol{x}[0] = \begin{bmatrix} 10 \\ 10 \end{bmatrix},$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad \boldsymbol{\Gamma} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

For the prediction horizon $N_p = 3$ and the performance index

$$J = \frac{1}{2}\sum_{i=1}^{N_p}\left(\boldsymbol{x}(k+i|k)^\top \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \boldsymbol{x}(k+i|k) + 0.1u[k+i-1]^2\right),$$

use MATLAB to perform the calculations and plot the evolution of state trajectories in time.

PP 3 Formulate the optimization problem using the model predictive controller to solve at each sampling time for a discrete dynamical model

$$\begin{aligned} \boldsymbol{x}[k+1] &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & -2 & -3 \end{bmatrix}\boldsymbol{x}[k] + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} u[k] \\ \boldsymbol{y}[k] &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\boldsymbol{x}[k], \end{aligned}$$

where the system output is to track the vector $[1\ 0.1]^\top$. The prediction horizon is $N_p = 3$. The constraint on the control signal is given in terms of the sum of the control inputs during the prediction horizon as $|u[k] + u[k+1] + u[k+2]| \leq 10$, and the constraint on the system state is $x_1[k] \leq 5$ for all k.

PP 4 Use the KKT multipliers method to solve the following optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x_1, x_2) = -x_1 x_2 \\ \text{subject to} \quad & x_1^2 + x_2^2 \leq 1,\ x_1 \geq 0,\ x_2 \geq 0. \end{aligned}$$

PP 5 Write the KKT conditions for the optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x_1, x_2, x_3) = (x_1 - 2)^2 + (x_2 - 3)^2 + (x_3 - 4)^2 \\ \text{subject to} \quad & x_1^2 + x_2^2 + x_3^2 \leq 1, \quad 4x_1 + x_2 + 2x_3 = 2. \end{aligned}$$

PP 6 Write the first-order Lagrangian algorithm for Practice Problem 4 and use MATLAB to perform calculations.

PP 7 Use the standard MATLAB function `quadprog` to perform calculations for the following optimal problem.

$$\begin{aligned} \text{minimize} \quad & 2x_1^2 + x_1 x_2 + x_2^2 + x_1 + 2x_2 \\ \text{subject to} \quad & x_1 + x_2 = 1; \quad x_1 \geq 0; \quad x_2 \geq 0\,. \end{aligned}$$

PP 8 Use the standard MATLAB function `fmincon` to perform calculations for Practice Problem 4.

PP 9 For the minimization problem

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \boldsymbol{\Omega}, \end{aligned}$$

prove that if $\boldsymbol{\Omega}$ is a convex set and $f(\boldsymbol{x})$ is a strictly convex function, then the optimal soultion (if exists) is unique.

PP 10 Consider the MPC optimization problem for the discrete dynamical system (2) with the performance index

$$J(k) = \frac{1}{2} \sum_{i=1}^{N_p} \left(\boldsymbol{r} - \boldsymbol{y}(k+i|k)\right)^\top \boldsymbol{Q} \left(\boldsymbol{r} - \boldsymbol{y}(k+i|k)\right) + \frac{1}{2} \sum_{i=0}^{N_u - 1} \boldsymbol{u}(k+i)^\top \boldsymbol{R} \boldsymbol{u}(k+i)\,, \qquad (28)$$

where $\boldsymbol{Q} \succeq 0$ and $\boldsymbol{R} \succ 0$, N_p is the prediction horizon, and N_u is the control horizon. Assume that $N_p \geq N_u$, and $\boldsymbol{u}(j) = \boldsymbol{u}(k + N_u - 1)$ for all $N_u \leq j \leq N_p$. Derive an explicit algorithm calculating the optimal control $\boldsymbol{u}(k)$.

6 Solutions to Practice Problems Prepared by Dr. Guisheng Zhai

SPP 1 First, we use the dynamical system modeling equation to obtain

$$\begin{aligned} x(1|0) &= x[0] + u[0] \\ x(2|0) &= x(1|0) + u[1] = x[0] + u[0] + u[1] \end{aligned} .$$

The performance index for $k = 0$ over the prediction horizon is

$$\begin{aligned} J &= \tfrac{1}{2}\left\{x(1|0)^2 + x(2|0)^2 + u[0]^2 + u[1]^2\right\} \\ &= \tfrac{1}{2}\left\{\left(x[0] + u[0]\right)^2 + \left(x[0] + u[0] + u[1]\right)^2 + u[0]^2 + u[1]^2\right\} . \end{aligned}$$

Applying the first-order necessary condition test to J, we obtain

$$\begin{cases} \frac{\partial J}{\partial u[0]} = 3u[0] + u[1] + 2x[0] = 0 \\ \frac{\partial J}{\partial u[1]} = u[0] + 2u[1] + x[0] = 0. \end{cases}$$

Substituting $x[0] = 10$ and solving the above equations gives

$$u[0] = -6,\ u[1] = -2.$$

SPP 2 A MATLAB program used to perform calculations is given below.

```
%  System data and parameters
Phi = [1 -1; 1 1];
Gamma = [1; 0.5];
C = eye(2);
x = [10; 10];

font_size=14;

Np = 3;
target=[0 0]';                    % reference for y to track
rp=kron(ones(Np,1),target);

R = 0.1*eye(size(Gamma,2)*Np);
Q0 = [1 0; 0 2];
Q = kron(eye(Np),Q0);
%% Set matrices W and Z
W=[];
for j = 1:1:Np
    W = [W; C*Phi^j];
```

```
end

p=size(C,1);
m=size(Gamma,2);
Z=zeros(Np*p,Np*m);
Z(1:p,1:m)=C*Gamma;
temp=C*Gamma;
for i=1:Np-1
    temp=[C*Phi^i*Gamma temp];
    Z(i*p+1:(i+1)*p,1:size(temp,2))=temp;
end
%% State Calculation
X = [];
Kend = 6;

for k=0:1:Kend
    X = [X x];
    U = inv(R+Z'*Q*Z)*Z'*Q*(rp-W*x);
    Uk = [eye(m) zeros(m,m*Np-m)]*U;
    Y = W*x + Z*U
    x = Phi*x+ Gamma*Uk;
end
%% Plots
plot(0:1:Kend,X)
grid on
set(gca,'FontSize',font_size)
leg=legend('$x_1[k]$','$x_2[k]$');
set(leg,'Interpreter','latex','location','northeast');
xlabel('$k$','Interpreter','latex','FontSize',font_size);
ylabel('$x[k]$','Interpreter','latex','FontSize',font_size);
```

Plots of the system states versus time are shown in Figure 7.

SPP 3 For $N_p = 3$,

$$\boldsymbol{U} = \begin{bmatrix} u[k] \\ u[k+1] \\ u[k+2] \end{bmatrix},$$

and the constraint $|u[k] + u[k+1] + u[k+2]| \leq 10$ can be expressed as

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} 10 \\ 10 \end{bmatrix}.$$

To express the constraint on the system state $x_1[k] \leq 5$, we define an auxiliary output

$$z(k) = \boldsymbol{c}_z \boldsymbol{x}[k], \quad \boldsymbol{c}_z = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

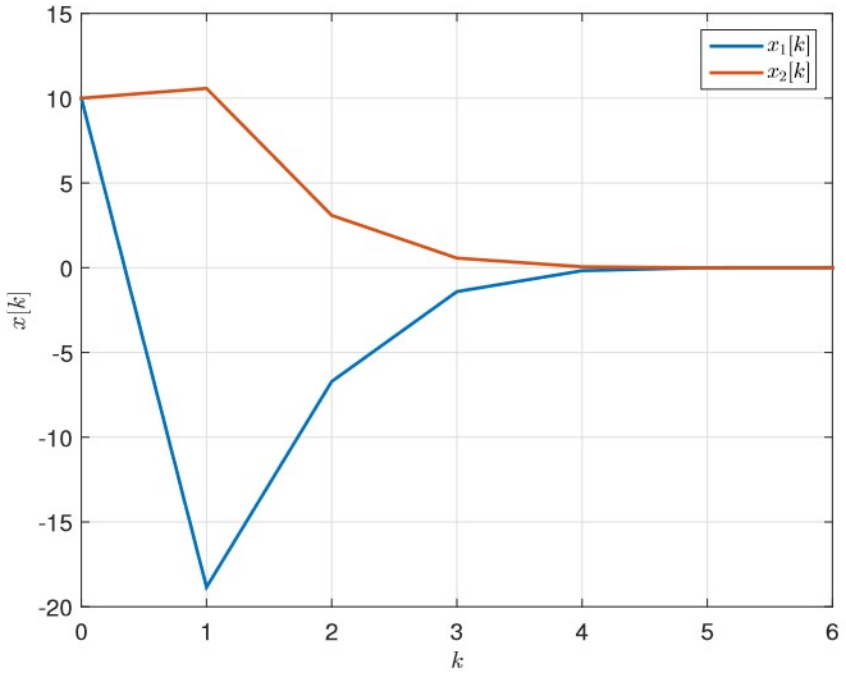

Figure 7: Plots of state trajectories versus time for Practice Problem 2

and represent the constraint as

$$\begin{bmatrix} z(k+1|k) \\ z(k+2|k) \\ z(k+3|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_z \boldsymbol{x}(k+1|k) \\ \boldsymbol{c}_z \boldsymbol{x}(k+2|k) \\ \boldsymbol{c}_z \boldsymbol{x}(k+3|k) \end{bmatrix} = \boldsymbol{W}_z \boldsymbol{x}[k] + \boldsymbol{Z}_z \boldsymbol{U} \leq \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}, \tag{29}$$

where

$$\boldsymbol{W}_z = \begin{bmatrix} \boldsymbol{c}_z \\ \boldsymbol{c}_z \boldsymbol{\Phi} \\ \boldsymbol{c}_z \boldsymbol{\Phi}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{bmatrix}$$

and

$$\boldsymbol{Z}_z = \begin{bmatrix} \boldsymbol{c}_z \boldsymbol{\Gamma} & & \\ \boldsymbol{c}_z \boldsymbol{\Phi} \boldsymbol{\Gamma} & \boldsymbol{c}_z \boldsymbol{\Gamma} & \\ \boldsymbol{c}_z \boldsymbol{\Phi}^2 \boldsymbol{\Gamma} & \boldsymbol{c}_z \boldsymbol{\Phi} \boldsymbol{\Gamma} & \boldsymbol{c}_z \boldsymbol{\Phi} \boldsymbol{\Gamma} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -4 & 1 & 1 \end{bmatrix}.$$

To define the tracking objective, we let

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}(k+1|k) \\ \boldsymbol{y}(k+2|k) \\ \boldsymbol{y}(k+3|k) \end{bmatrix} = \boldsymbol{W} \boldsymbol{x}[k] + \boldsymbol{Z} \boldsymbol{U} \quad \text{and} \quad \boldsymbol{r}_p = \begin{bmatrix} 1 \\ 0.1 \\ 1 \\ 0.1 \\ 1 \\ 0.1 \end{bmatrix},$$

where

$$\boldsymbol{c} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{c\Phi} \\ \boldsymbol{c\Phi}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & -2 & -3 \\ 0 & 1 & 0 \\ 3 & 4 & 8 \end{bmatrix},$$

and

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{c\Gamma} & & \\ \boldsymbol{c\Phi\Gamma} & \boldsymbol{c\Gamma} & \\ \boldsymbol{c\Phi}^2\boldsymbol{\Gamma} & \boldsymbol{c\Phi\Gamma} & \boldsymbol{c\Phi\Gamma} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 0 \\ 11 & -4 & 1 \end{bmatrix}.$$

Then, the optimization problem has the form

$$\min \tfrac{1}{2} \left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}[k] - \boldsymbol{Z}\boldsymbol{U}\right)^\top \boldsymbol{Q} \left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}[k] - \boldsymbol{Z}\boldsymbol{U}\right) + \tfrac{1}{2}\boldsymbol{U}^\top \boldsymbol{R}\boldsymbol{U}$$

subject to

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ -1 & -2 & -3 \end{bmatrix} \boldsymbol{U} \leq \begin{bmatrix} 10 \\ 10 \\ 5 \\ 5 \\ 5 \end{bmatrix} - \boldsymbol{W}_z \boldsymbol{x}[k],$$

where $\boldsymbol{Q} = \boldsymbol{Q}^\top \succeq 0$ and $\boldsymbol{R} = \boldsymbol{R}^\top \succ 0$. We can take, for example, $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{I}_3$.

SPP 4 We first represent the optimization problem in a standard form as follows,

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) = -x_1 x_2 \\ \text{subject to} & g_1(\boldsymbol{x}) = x_1^2 + x_2^2 - 1 \leq 0 \\ & g_2(\boldsymbol{x}) = -x_1 \leq 0 \\ & g_3(\boldsymbol{x}) = -x_2 \leq 0. \end{array}$$

The Lagrange function is

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\mu}) &= f(\boldsymbol{x}) + \mu_1 g_1(\boldsymbol{x}) + \mu_2 g_2(\boldsymbol{x}) + \mu_3 g_3(\boldsymbol{x}) \\ &= -x_1 x_2 + \mu_1(x_1^2 + x_2^2 - 1) + \mu_2(-x_1) + \mu_3(-x_2). \end{aligned}$$

Then, the KKT conditions are given by

$$\left.\begin{array}{l} \mu_1 \geq 0,\ \mu_2 \geq 0,\ \mu_3 \geq 0 \\ \nabla_x L(\boldsymbol{x}, \boldsymbol{\mu}) = \left[\begin{array}{c} -x_2 + 2\mu_1 x_1 - \mu_2 \\ -x_1 + 2\mu_1 x_2 - \mu_3 \end{array}\right] = 0 \\ \mu_1 g_1(\boldsymbol{x}) = \mu_1(x_1^2 + x_2^2 - 1) = 0 \\ \mu_2 g_2(\boldsymbol{x}) = \mu_2(-x_1) = 0 \\ \mu_3 g_3(\boldsymbol{x}) = \mu_3(-x_2) = 0, \end{array}\right\} \tag{30}$$

where

$$\begin{aligned} g_1(\boldsymbol{x}) &= x_1^2 + x_2^2 - 1 \leq 0 \\ g_2(\boldsymbol{x}) &= -x_1 \leq 0 \\ g_3(\boldsymbol{x}) &= -x_2 \leq 0. \end{aligned}$$

It is seen from the equations in (30) that if μ_2 or μ_3 is nonzero, then x_1 or x_2 must be zero, respectively. This leads to $f(\boldsymbol{x}) = 0$, which is obviously not optimal. Thus, $\mu_2 = \mu_3 = 0$. Moreover, if $\mu_1 = 0$, then we obtain from $\nabla_x L(\boldsymbol{x}, \boldsymbol{\mu}) = \mathbf{0}$ that $x_1 = x_2 = 0$, which is not optimal either. This implies $\mu_1 \neq 0$, and $x_1^2 + x_2^2 - 1 = 0$. After some manipulations, we obtain

$$\mu_1 = \frac{1}{2}, \quad x_1 = x_2 = \frac{1}{\sqrt{2}}$$

and the optimal value of $f(\boldsymbol{x})$ is $-\frac{1}{2}$.

SPP 5 The KKT conditions for the general optimization problem

$$\begin{array}{lll} \text{minimize} & f(\boldsymbol{x}) & \\ \text{subject to} & g_1(\boldsymbol{x}) \leq 0, & \ldots, \quad g_P(\boldsymbol{x}) \leq 0 \\ & h_1(\boldsymbol{x}) = 0, & \ldots, \quad h_M(\boldsymbol{x}) = 0 \end{array} \tag{31}$$

require that there exist $\boldsymbol{x}^*$, μ_i^*, $i = 1, \ldots, P$, and λ_j^*, $j = 1, \ldots, M$ such that

$$\left\{\begin{array}{l} \nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{P} \mu_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^{M} \lambda_j^* \nabla h_j(\boldsymbol{x}^*) = \mathbf{0} \\ \mu_i^* g_i(\boldsymbol{x}^*) = 0, \quad \mu_i^* \geq 0, \quad i = 1, \ldots, P \\ g_i(\boldsymbol{x}^*) \leq 0, \quad i = 1, \ldots, P \\ h_j(\boldsymbol{x}^*) = 0, \quad j = 1, \ldots, M\,. \end{array}\right. \tag{32}$$

We apply the above conditions to our problem and obtain the KKT conditions (omitting

the asterisk $*$ for simplicity of notation)

$$\begin{cases} \begin{bmatrix} 2x_1 - 4 \\ 2x_2 - 6 \\ 2x_3 - 8 \end{bmatrix} + \mu \begin{bmatrix} 2x_1 \\ 2x_2 \\ 2x_3 \end{bmatrix} + \lambda \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} = \mathbf{0} \\ \mu(x_1^2 + x_2^2 + x_3^2 - 1) = 0, \quad \mu \geq 0 \\ x_1^2 + x_2^2 + x_3^2 \leq 1 \\ 4x_1 + x_2 + 2x_3 = 2\,. \end{cases} \tag{33}$$

SPP 6 We use the cost function $f(\boldsymbol{x})$ and the constraint function $\boldsymbol{g}(\boldsymbol{x})$ defined in (31) to write the first-order Lagrangian algorithm as

$$\begin{aligned} \boldsymbol{x}^{(i+1)} &= \boldsymbol{x}^{(i)} - \alpha_i \left(\nabla f\left(\boldsymbol{x}^{(i)}\right) + D\boldsymbol{g}\left(\boldsymbol{x}^{(i)}\right)^\top \boldsymbol{\mu}^{(i)} \right) \\ &= \boldsymbol{x}^{(i)} - \alpha_i \left(\begin{bmatrix} -x_2^i \\ -x_1^i \end{bmatrix} + \mu_1^i \begin{bmatrix} 2x_1^i \\ 2x_2^i \end{bmatrix} + \mu_2^i \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \mu_3^i \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right) \\ &= \boldsymbol{x}^{(i)} - \alpha_i \begin{bmatrix} -x_2^i + 2\mu_1^i x_1^i - \mu_2^i \\ -x_1^i + 2\mu_2^i x_2^i - \mu_3^i \end{bmatrix} \\ \boldsymbol{\mu}^{(i+1)} &= \left[\boldsymbol{\mu}^{(i)} + \beta_i \boldsymbol{g}\left(\boldsymbol{x}^{(i)}\right) \right]_+ \\ &= \max\left(\boldsymbol{\mu}^{(i)} + \beta_i \begin{bmatrix} (x_1^i)^2 + (x_2^i)^2 - 1 \\ -x_1^i \\ -x_2^i \end{bmatrix}, \mathbf{0} \right). \end{aligned}$$

where α_i's and β_i's are Lagrangian coefficients at the i-th iteration.

A MATLAB program that implements the above algorithm is given below.

```
% Inialization
x1 = 1.0; x2= 1.0;
x = [x1; x2];
mu1 = 0; mu2 = 0;  mu3 =0;
mu = [mu1; mu2; mu3];
alpha = 0.01;
beta = 0.01;
% Iteration for computing x and mu
for i = 1:200
    nablaf = [-x(2); -x(1)];
    nablag1 = 2*x; nablag2=[-1; 0]; nablag3=[0; -1];
    nablag = [nablag1 nablag2 nablag3];
    next_x = x - alpha*(nablaf+nablag*mu);
```

```
    g = [x'*x-1; -x(1); -x(2)];
    mu = max(mu + beta*g, 0);

    if abs(next_x-x)<1.0e-5
        break;
    end

    x=next_x;
end
```

SPP 7 The MATLAB command `quadprog` deals with the optimization problem with quadratic objective functions and linear constraints. For the optimization problem,

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{f} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \\ & \boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \\ & \boldsymbol{l}_b \le \boldsymbol{x} \le \boldsymbol{u}_b\,, \end{aligned}$$

where $\boldsymbol{A}$, $\boldsymbol{A}_{eq}$, $\boldsymbol{b}$ and $\boldsymbol{b}_{eq}$ are constant matrices/vectors defining linear constraints, and $\boldsymbol{l}_b$ and $\boldsymbol{u}_b$ are constant vectors defining the lower bound and the upper bound of $\boldsymbol{x}$, respectively. Then, the command for the above problem is

`x = quadprog(H,f,A,b,Aeq,beq,lb,ub)`

For the present problem, the MATLAB command list is

```
H = [4 1; 1 2];
f = [1; 1];
A = -eye(2); b = zeros(2,1);
Aeq = [1 1];    beq = 1;
x = quadprog(H,f,A,b,Aeq,beq)
```

or

```
H = [4 1; 1 2];
f = [1; 1];
Aeq = [1 1];    beq = 1;
lb = zeros(2,1);
x = quadprog(H,f,[],[],Aeq,beq,lb)
```

Both give the optimal solution $x_1 = 0.25, x_2 = 0.75$.

SPP 8 For the optimization problem,

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{c}(x) \leq \boldsymbol{0} \\ & \boldsymbol{c}_{eq}(\boldsymbol{x}) = 0 \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \\ & \boldsymbol{l}_b \leq \boldsymbol{x} \leq \boldsymbol{u}_b \,, \end{aligned}$$

where $\boldsymbol{A}$, $\boldsymbol{A}_{eq}$, $\boldsymbol{b}$ and $\boldsymbol{b}_{eq}$ are constant matrices/vectors defining linear constraints, $\boldsymbol{c}(\boldsymbol{x})$ and $\boldsymbol{c}_{eq}(\boldsymbol{x})$ are vector functions defining nonlinear constraints, and $\boldsymbol{l}_b$ and $\boldsymbol{u}_b$ are constant vectors defining the lower bound and the upper bound of $\boldsymbol{x}$, respectively. The MATLAB command for the above problem is

```
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

where `fun` is used to define the obective function $f(x)$, `x0` defines the starting point, and `nonlcon` is used to define the nonlinear constraints $\boldsymbol{c}(\boldsymbol{x})$ and $\boldsymbol{c}_{eq}(\boldsymbol{x})$.

For the present problem, the MATLAB program is as follows, which returns the optimal result $x_1 = x_2 = 0.7071$.

```
% Use it after defining the objective function and the nonlinear constraints
% [0.1; 0.2] is the initial (starting) point
x = fmincon(@objectfun,[0.1;0.2],[],[],[],[],[],[],@nonlconstr)
%  objectfun.m: Define the objection function
function f = objecfun(x)
f = -x(1)*x(2)
% nonlconstr.m: Define the nonlinear constraints
function [c,ceq] = nonlconstr(x)
c = [x(1)^2 + x(2)^2 - 1;
     -x(1);
     -x(2)];
ceq = [];
```

SPP 9 Suppose that there are two optimal solutions $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, that minimizes the strictly convex function $f(\boldsymbol{x})$, that is, for all $\boldsymbol{x} \in \boldsymbol{\Omega} \setminus \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$, $f(\boldsymbol{x}) > f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)$.

Since $\boldsymbol{\Omega}$ is convex, for any $0 < \lambda < 1$, we have $\lambda\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \boldsymbol{\Omega}$. Moreover, since $f(\boldsymbol{x})$ is strictly convex, we obtain

$$f(\lambda\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) < \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1)$$

which contradicts the assumption that $\boldsymbol{x}_1$ is a minimizer. Therefore, the optimal solution is unique.

SPP 10 We evaluate the states over the prediction horizon successively applying the recursion formula (1) to obtain,

$$\begin{aligned}
\boldsymbol{x}(k+1|k) &= \boldsymbol{\Phi}\boldsymbol{x}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k] \\
\boldsymbol{x}(k+2|k) &= \boldsymbol{\Phi}\boldsymbol{x}(k+1|k) + \boldsymbol{\Gamma}\boldsymbol{u}[k+1] \\
&\quad = \boldsymbol{\Phi}^2\boldsymbol{x}[k] + \boldsymbol{\Phi}\boldsymbol{\Gamma}\boldsymbol{u}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k+1] \\
&\vdots \\
\boldsymbol{x}(k+N_u|k) &= \boldsymbol{\Phi}^{N_u}\boldsymbol{x}[k] + \boldsymbol{\Phi}^{N_u-1}\boldsymbol{\Gamma}\boldsymbol{u}[k] + \cdots + \boldsymbol{\Gamma}\boldsymbol{u}[k+N_u-1] \\
\boldsymbol{x}(k+N_u+1|k) &= \boldsymbol{\Phi}^{N_u+1}\boldsymbol{x}[k] + \boldsymbol{\Phi}^{N_u}\boldsymbol{\Gamma}\boldsymbol{u}[k] + \cdots + (\boldsymbol{\Phi}\boldsymbol{\Gamma}+\boldsymbol{\Gamma})\boldsymbol{u}[k+N_u-1] \\
&\vdots \\
\boldsymbol{x}(k+N_p|k) &= \boldsymbol{\Phi}^{N_p}\boldsymbol{x}[k] + \boldsymbol{\Phi}^{N_p-1}\boldsymbol{\Gamma}\boldsymbol{u}[k] + \cdots + \left(\sum_{j=0}^{N_p-N_u}\boldsymbol{\Phi}^j\boldsymbol{\Gamma}\right)\boldsymbol{u}[k+N_u-1]\,.
\end{aligned}$$

We represent the above set of equations in the form,

$$\boldsymbol{X} = \boldsymbol{W}_x\boldsymbol{x}[k] + \boldsymbol{Z}_u\boldsymbol{U},$$

where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}(k+1|k) \\ \boldsymbol{x}(k+2|k) \\ \vdots \\ \boldsymbol{x}(k+N_u|k) \\ \boldsymbol{x}(k+N_u+1|k) \\ \vdots \\ \boldsymbol{x}(k+N_p|k) \end{bmatrix}, \quad \boldsymbol{W}_x = \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^2 \\ \vdots \\ \boldsymbol{\Phi}^{N_u} \\ \boldsymbol{\Phi}^{N_u+1} \\ \vdots \\ \boldsymbol{\Phi}^{N_p} \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}[k] \\ \boldsymbol{u}[k+1] \\ \vdots \\ \boldsymbol{u}[k+N_u-1] \end{bmatrix}$$

and

$$\boldsymbol{Z}_u = \begin{bmatrix} \boldsymbol{\Gamma} & & & \\ \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Gamma} & & \\ \vdots & & \ddots & \\ \boldsymbol{\Phi}^{N_u-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Gamma} \\ \boldsymbol{\Phi}^{N_u}\boldsymbol{\Gamma} & \cdots & \boldsymbol{\Phi}^2\boldsymbol{\Gamma} & \boldsymbol{\Phi}\boldsymbol{\Gamma}+\boldsymbol{\Gamma} \\ \vdots & & \ddots & \\ \boldsymbol{\Phi}^{N_p-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{\Phi}^{N_p-N_u+1}\boldsymbol{\Gamma} & \sum_{j=0}^{N_p-N_u}\boldsymbol{\Phi}^j\boldsymbol{\Gamma} \end{bmatrix}.$$

To describe the performance index using the above notations, we compute the sequence of

predicted outputs (4),

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}(k+1|k) \\ \boldsymbol{y}(k+2|k) \\ \vdots \\ \boldsymbol{y}(k+N_p|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{x}(k+1|k) \\ \boldsymbol{C}\boldsymbol{x}(k+2|k) \\ \vdots \\ \boldsymbol{C}\boldsymbol{x}(k+N_p|k) \end{bmatrix} = \boldsymbol{W}\boldsymbol{x}[k] + \boldsymbol{Z}\boldsymbol{U}$$

where

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{\Phi} \\ \boldsymbol{C}\boldsymbol{\Phi}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_u} \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_u+1} \\ \vdots \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_p} \end{bmatrix}, \quad \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{C}\boldsymbol{\Gamma} & & & \\ \boldsymbol{C}\boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{C}\boldsymbol{\Gamma} & & \\ \vdots & & \ddots & \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_u-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{C}\boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{C}\boldsymbol{\Gamma} \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_u}\boldsymbol{\Gamma} & \cdots & \boldsymbol{C}\boldsymbol{\Phi}^2\boldsymbol{\Gamma} & \boldsymbol{C}\boldsymbol{\Phi}\boldsymbol{\Gamma} + \boldsymbol{C}\boldsymbol{\Gamma} \\ \vdots & & \ddots & \\ \boldsymbol{C}\boldsymbol{\Phi}^{N_p-1}\boldsymbol{\Gamma} & \cdots & \boldsymbol{C}\boldsymbol{\Phi}^{N_p-N_u+1}\boldsymbol{\Gamma} & \sum_{j=0}^{N_p-N_u} \boldsymbol{C}\boldsymbol{\Phi}^j\boldsymbol{\Gamma} \end{bmatrix}.$$

Define

$$\boldsymbol{r}_p = \mathbf{1}_{N_p} \otimes \boldsymbol{r}, \quad \boldsymbol{Q}_p = \boldsymbol{I}_{N_p} \otimes \boldsymbol{Q}, \quad \boldsymbol{R}_u = \boldsymbol{I}_{N_u} \otimes \boldsymbol{R}$$

and write the performance index (28) as

$$J(\boldsymbol{U}) = \frac{1}{2} \left(\boldsymbol{r}_p - \boldsymbol{Y}\right)^\top \boldsymbol{Q}_p \left(\boldsymbol{r}_p - \boldsymbol{Y}\right) + \frac{1}{2} \boldsymbol{U}^\top \boldsymbol{R}_u \boldsymbol{U}. \tag{34}$$

Then, we first apply the first-order necessary condition (FONC) test to $J(\boldsymbol{U})$,

$$\frac{\partial J}{\partial \boldsymbol{U}} = -\left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}[k] - \boldsymbol{Z}\boldsymbol{U}\right)^\top \boldsymbol{Q}_p \boldsymbol{Z} + \boldsymbol{U}^\top \boldsymbol{R}_u = \mathbf{0}^\top.$$

Performing simple manipulations yields

$$\boldsymbol{U}^* = \left(\boldsymbol{R}_u + \boldsymbol{Z}^\top \boldsymbol{Q}_p \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\top \boldsymbol{Q}_p \left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}[k]\right). \tag{35}$$

Now, applying the second derivative test to $J(\boldsymbol{U})$, which we refer to as the second-order sufficiency condition (SONC), we obtain

$$\frac{\partial^2 J}{\partial \boldsymbol{U}^2} = \boldsymbol{R}_u + \boldsymbol{Z}^\top \boldsymbol{Q}_p \boldsymbol{Z} \succ 0,$$

which implies that $\boldsymbol{U}^*$ is a strict minimizer of $J(\boldsymbol{U})$.

Using (35), we compute $\boldsymbol{u}[k]$,

$$\begin{aligned} \boldsymbol{u}[k] &= \overbrace{\begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{bmatrix}}^{N_u \text{ block matrices}} \left(\boldsymbol{R}_u + \boldsymbol{Z}^\top \boldsymbol{Q}_p \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\top \boldsymbol{Q}_p \left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}\right) \\ &= \boldsymbol{K}_{\boldsymbol{r}} \boldsymbol{r}_p - \boldsymbol{K}_{\boldsymbol{x}} \boldsymbol{x}[k], \end{aligned} \tag{36}$$

where

$$\boldsymbol{K}_{\boldsymbol{r}} = \begin{bmatrix} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{bmatrix} \left(\boldsymbol{R}_u + \boldsymbol{Z}^\top \boldsymbol{Q}_p \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\top \boldsymbol{Q}_p, \quad \boldsymbol{K}_{\boldsymbol{x}} = \boldsymbol{K}_{\boldsymbol{r}} \boldsymbol{W}.$$

References

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

[2] B. Kouvaritakis and M. Cannon, editors. *Nonlinear Predictive Control: Theory and Practice*. The Institution of Electrical Engineers, London, 2001.

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

[4] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, July 1990.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[6] H. Michalska and D. Q. Mayne. Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 40(6):995–1006, June 1995.

[7] L. Wang. *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, London, 2009.