ECE 680 Fall 2011

Model-based Predictive Control (MPC)

by
Stanislaw H. Żak

1 Introduction

The model-based predictive control (MPC) methodology is also referred to as the moving horizon control or the receding horizon control. The idea behind this approach can be explained using an example of driving a car. The driver looks at the road ahead of him and taking into account the present state and the previous action predicts his action up to some distance ahead, which we refer to as the prediction horizon. Based on the prediction, the driver adjusts the driving direction. The MPC main idea is illustrated in Figure 1.

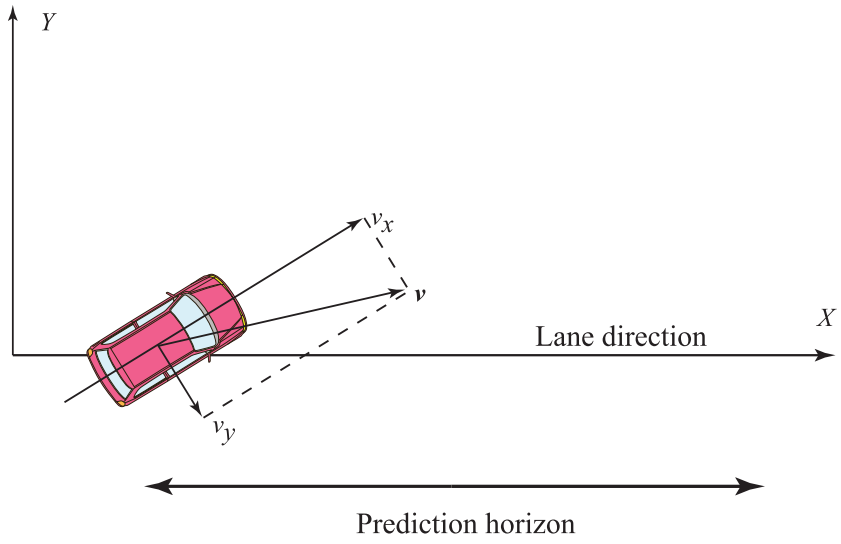

Figure 1: The driver predicts future travel direction based on the current state of the car and the current position of the steering wheel.

The MPC is constructed using control and optimization tools. The objective of this write-up is to introduce the reader to the linear MPC which refers to the family of MPC schemes in which linear models of the controlled objects are used in the control law synthesis.

In the MPC approach, the current control action is computed on-line rather than using a pre-computed, off-line, control law.

A model predictive controller uses, at each sampling instant, the plant's current input and output measurements, the plant's current state, and the plant's model to

- calculate, over a finite horizon, a future control sequence that optimizes a given performance index and satisfies constraints on the control action;
- use the first control in the sequence as the plant's input.

The MPC strategy is illustrated in Figure 2, where N_p is the prediction horizon, $u(t+k|t)$ is the predicted control action at $t+k$ given $u(t)$. Similarly, $y(t+k|t)$ is the predicted output at $t+k$ given $y(t)$.

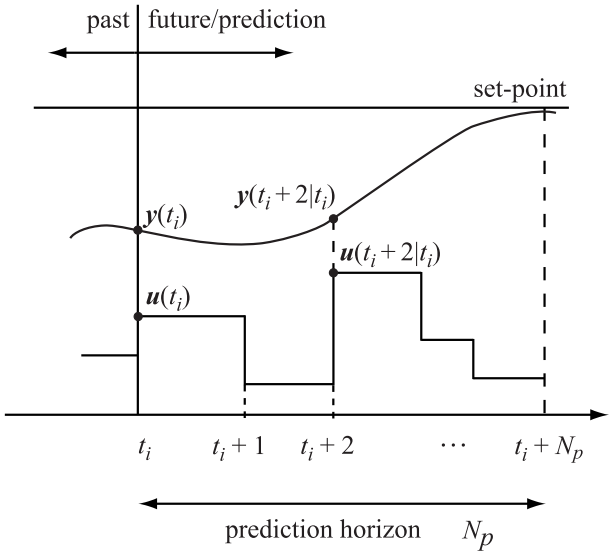

Figure 2: Controller action construction using model-based predictive control (MPC) approach.

2 Basic Structure of MPC

In Figure 3, we show a basic structure of an MPC-controlled plant, where we assume that the plant's state is available to us.

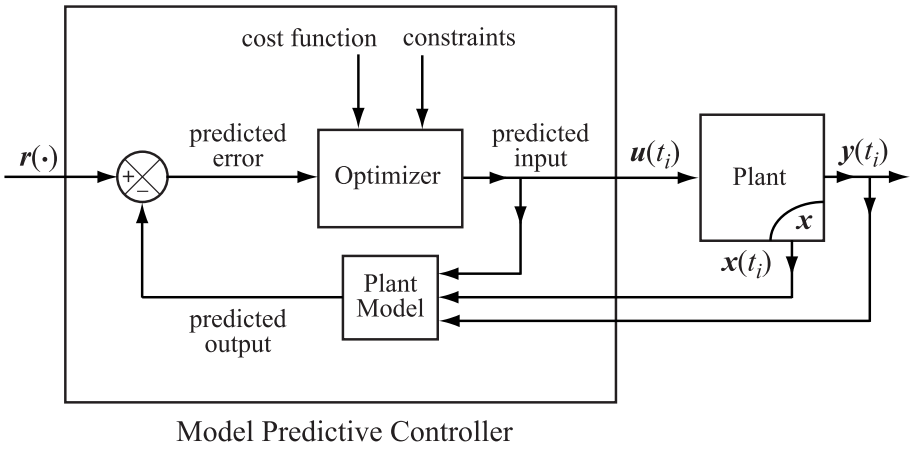

Figure 3: State feedback model predictive controller.

3 From Continuous to Discrete Models

Our objective here is to present a method for constructing linear discrete-time models from given linear continuous-time models. The obtained discrete models will be used to perform computations to generate control commands.

We use a sample-and-hold device that transforms a continuous signal, $f(t)$, into the staircase signal,

$$f(kh), \quad kh \leq t < (k+1)h,$$

where h is the sampling period. The sample and zero-order hold (ZOH) operation is illustrated in Figure 4

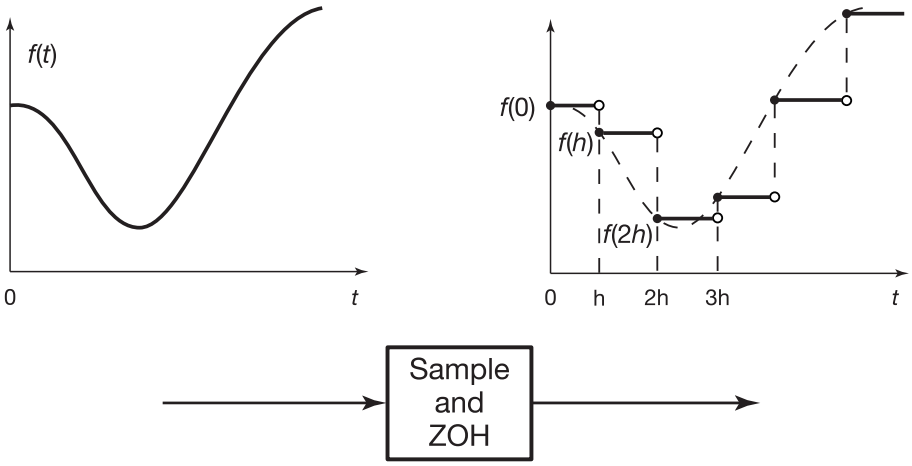

Figure 4: Sample and zero-order hold (ZOH) element operating on a continuous function.

Suppose that we are given a continuous-time model,

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}_0 = \boldsymbol{x}(t_0) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t). \end{aligned}$$

The solution to the state equation is

$$\boldsymbol{x}(t) = e^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}(t_0) + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau. \tag{1}$$

We assume that the input to the system is generated by a sample-and-hold device and has the form,

$$\boldsymbol{u}(t) = \boldsymbol{u}(k), \quad kh \le t < (k+1)h.$$

Let $t_0 = kh$ and $t = (k+1)h$ and let us use shorthand notation,

$$\boldsymbol{x}(kh) = \boldsymbol{x}(k)$$

Then taking into account that $\boldsymbol{u}(k)$ is constant on the interval $[kh, (k+1)h)$ we represent (1) as

$$\begin{aligned} \boldsymbol{x}(k+1) &= e^{\boldsymbol{A}h}\boldsymbol{x}(k) + \int_{kh}^{(k+1)h} e^{\boldsymbol{A}(kh+h-\tau)}\boldsymbol{B}\boldsymbol{u}(k)d\tau \\ &= e^{\boldsymbol{A}h}\boldsymbol{x}(k) + \int_{kh}^{(k+1)h} e^{\boldsymbol{A}(kh+h-\tau)}\boldsymbol{B}d\tau\,\boldsymbol{u}(k). \end{aligned} \tag{2}$$

Consider now the second term on the right-hand side of the above equation. Let

$$\eta = kh + h - \tau.$$

Then we can represent (2) as

$$\begin{aligned} \boldsymbol{x}(k+1) &= e^{\boldsymbol{A}h}\boldsymbol{x}(k) + \int_{kh}^{(k+1)h} e^{\boldsymbol{A}(kh+h-\tau)}\boldsymbol{B}d\tau\,\boldsymbol{u}(k) \\ &= e^{\boldsymbol{A}h}\boldsymbol{x}(k) + \int_{0}^{h} e^{\boldsymbol{A}\eta}\boldsymbol{B}d\eta\,\boldsymbol{u}(k) \\ &= \boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Gamma}\boldsymbol{u}(k), \end{aligned} \tag{3}$$

where

$$\boldsymbol{\Phi} = e^{\boldsymbol{A}h} \quad \text{and} \quad \boldsymbol{\Gamma} = \int_{0}^{h} e^{\boldsymbol{A}\eta}\boldsymbol{B}d\eta.$$

The discrete output equation has the form

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k).$$

4 Simple Discrete-Time MPC

The development a discrete MPC controller in this section follows that of [9, Chapter 1]. We consider a discretized model of a dynamic system of the form,

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{\Phi}\boldsymbol{x}(k) + \boldsymbol{\Gamma}\boldsymbol{u}(k) && (4) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k), && (5) \end{aligned}$$

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

Applying the backward difference operator, $\Delta\boldsymbol{x}(k+1) = \boldsymbol{x}(k+1) - \boldsymbol{x}(k)$, to (4) gives

$$\Delta\boldsymbol{x}(k+1) = \boldsymbol{\Phi}\Delta\boldsymbol{x}(k) + \boldsymbol{\Gamma}\Delta\boldsymbol{u}(k), \tag{6}$$

where $\Delta\boldsymbol{u}(k+1) = \boldsymbol{u}(k+1) - \boldsymbol{u}(k)$.

We now apply the backward difference operator to (5) to obtain

$$\begin{aligned} \Delta\boldsymbol{y}(k+1) &= \boldsymbol{y}(k+1) - \boldsymbol{y}(k) \\ &= \boldsymbol{C}\boldsymbol{x}(k+1) - \boldsymbol{C}\boldsymbol{x}(k) \\ &= \boldsymbol{C}\Delta\boldsymbol{x}(k+1). \end{aligned}$$

Substituting into the above (6) yields

$$\Delta\boldsymbol{y}(k+1) = \boldsymbol{C}\boldsymbol{\Phi}\Delta\boldsymbol{x}(k) + \boldsymbol{C}\boldsymbol{\Gamma}\Delta\boldsymbol{u}(k).$$

Hence,

$$\boldsymbol{y}(k+1) = \boldsymbol{y}(k) + \boldsymbol{C}\boldsymbol{\Phi}\Delta\boldsymbol{x}(k) + \boldsymbol{C}\boldsymbol{\Gamma}\Delta\boldsymbol{u}(k). \tag{7}$$

We combine (6) and (7) into one equation to obtain

$$\begin{bmatrix} \Delta\boldsymbol{x}(k+1) \\ \boldsymbol{y}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{O} \\ \boldsymbol{C}\boldsymbol{\Phi} & \boldsymbol{I}_p \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}(k) \\ \boldsymbol{y}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{C}\boldsymbol{\Gamma} \end{bmatrix} \Delta\boldsymbol{u}(k). \tag{8}$$

We represent (5) as

$$\boldsymbol{y}(k) = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{I}_p \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}(k) \\ \boldsymbol{y}(k) \end{bmatrix}. \tag{9}$$

We now define the augmented state vector,

$$\boldsymbol{x}_a(k) = \begin{bmatrix} \Delta\boldsymbol{x}(k) \\ \boldsymbol{y}(k) \end{bmatrix}. \tag{10}$$

Let

$$\boldsymbol{\Phi}_a = \begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{O} \\ \boldsymbol{C\Phi} & \boldsymbol{I}_p \end{bmatrix}, \quad \boldsymbol{\Gamma}_a = \begin{bmatrix} \boldsymbol{\Gamma} \\ \boldsymbol{C\Gamma} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{C}_a = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{I}_p \end{bmatrix}. \tag{11}$$

Using the above notation, we represent (8) and (9) in a compact format as

$$\begin{aligned} \boldsymbol{x}_a(k+1) &= \boldsymbol{\Phi}_a \boldsymbol{x}_a(k) + \boldsymbol{\Gamma}_a \Delta \boldsymbol{u}(k) && (12) \\ \boldsymbol{y}(k) &= \boldsymbol{C}_a \boldsymbol{x}_a(k), && (13) \end{aligned}$$

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

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

$$\Delta \boldsymbol{u}(k), \Delta \boldsymbol{u}(k+1), \ldots, \Delta \boldsymbol{u}(k+N_p-1), \tag{14}$$

where N_p is the prediction horizon, such that a given cost function and constraints are satisfied. The above control sequence will result in a predicted sequence of the state vectors,

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

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

$$\boldsymbol{y}(k+1|k), \boldsymbol{y}(k+2|k), \ldots, \boldsymbol{y}(k+N_p|k). \tag{15}$$

Using the above information, we can compute the control sequence (14) 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 $\boldsymbol{u}(k+1)$, until we reach the boundary of the control horizon, N_c.

We now present an approach to construct $\boldsymbol{u}(k)$ given $\boldsymbol{x}(k)$. Using the plant model parameters and the measurement of $\boldsymbol{x}_a(k)$ we evaluate the augmented states over the prediction horizon successively applying the recursion formula (12) to obtain,

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

We represent the above set of equations in the form,

$$\begin{bmatrix} \boldsymbol{x}_a(k+1|k) \\ \boldsymbol{x}_a(k+2|k) \\ \vdots \\ \boldsymbol{x}_a(k+N_p|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_a \\ \boldsymbol{\Phi}_a^2 \\ \vdots \\ \boldsymbol{\Phi}_a^{N_p} \end{bmatrix} \boldsymbol{x}_a(k) + \begin{bmatrix} \boldsymbol{\Gamma}_a & & \\ \boldsymbol{\Phi}_a\boldsymbol{\Gamma}_a & \boldsymbol{\Gamma}_a & \\ \vdots & & \ddots \\ \boldsymbol{\Phi}_a^{N_p-1}\boldsymbol{\Gamma}_a & \cdots & \boldsymbol{\Gamma}_a \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}(k) \\ \Delta\boldsymbol{u}(k+1) \\ \vdots \\ \Delta\boldsymbol{u}(k+N_p-1) \end{bmatrix}. \tag{16}$$

We wish to design a controller that would force the plant output, $\boldsymbol{y}$, to track a given reference signal, $\boldsymbol{r}$. Using (13) and (16), we compute the sequence of predicted outputs (15),

$$\begin{aligned} \begin{bmatrix} \boldsymbol{y}(k+1|k) \\ \boldsymbol{y}(k+2|k) \\ \vdots \\ \boldsymbol{y}(k+N_p|k) \end{bmatrix} &= \begin{bmatrix} \boldsymbol{C}_a\boldsymbol{x}_a(k+1|k) \\ \boldsymbol{C}_a\boldsymbol{x}_a(k+2|k) \\ \vdots \\ \boldsymbol{C}_a\boldsymbol{x}_a(k+N_p|k) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{C}_a\boldsymbol{\Phi}_a \\ \boldsymbol{C}_a\boldsymbol{\Phi}_a^2 \\ \vdots \\ \boldsymbol{C}_a\boldsymbol{\Phi}_a^{N_p} \end{bmatrix} \boldsymbol{x}_a(k) \qquad (17) \\ &\quad + \begin{bmatrix} \boldsymbol{C}_a\boldsymbol{\Gamma}_a & & \\ \boldsymbol{C}_a\boldsymbol{\Phi}_a\boldsymbol{\Gamma}_a & \boldsymbol{C}_a\boldsymbol{\Gamma}_a & \\ \vdots & & \ddots \\ \boldsymbol{C}_a\boldsymbol{\Phi}_a^{N_p-1}\boldsymbol{\Gamma}_a & \cdots & \boldsymbol{C}_a\boldsymbol{\Gamma}_a \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}(k) \\ \Delta\boldsymbol{u}(k+1) \\ \vdots \\ \Delta\boldsymbol{u}(k+N_p-1) \end{bmatrix}. \qquad (18) \end{aligned}$$

We write the above compactly as

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{x}_a(k) + \boldsymbol{Z}\Delta\boldsymbol{U}, \tag{19}$$

where

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

and

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

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

$$J(\Delta\boldsymbol{U}) = \frac{1}{2}\left(\boldsymbol{r}_p - \boldsymbol{Y}\right)^\top \boldsymbol{Q}\left(\boldsymbol{r}_p - \boldsymbol{Y}\right) + \frac{1}{2}\Delta\boldsymbol{U}^\top \boldsymbol{R}\Delta\boldsymbol{U}, \tag{20}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^\top > 0$ and $\boldsymbol{R} = \boldsymbol{R}^\top > 0$ 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, $k, k+1, \ldots, k+N_p-1$. 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 $J(\Delta\boldsymbol{U})$,

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

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

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

Performing simple manipulations yields

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

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

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

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

$$\Delta\boldsymbol{U}^* = \left(\boldsymbol{R} + \boldsymbol{Z}^\top \boldsymbol{Q}\boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\top \boldsymbol{Q}\left(\boldsymbol{r}_p - \boldsymbol{W}\boldsymbol{x}_a\right). \tag{21}$$

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

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

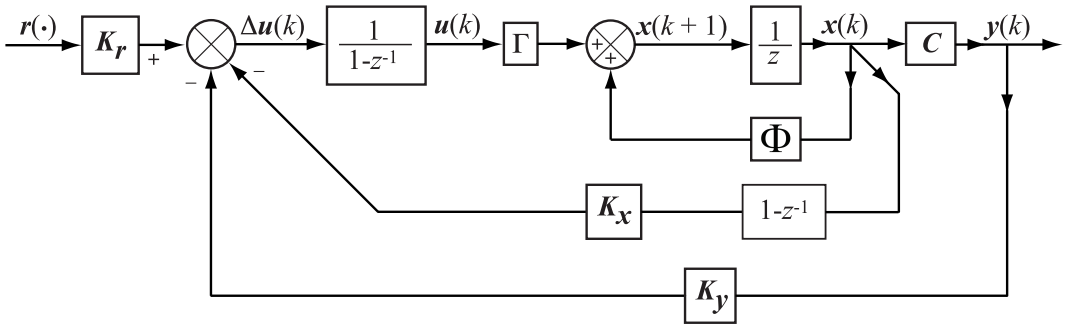

Figure 5: Discrete-time MPC for linear time-invariant systems.

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

Using (21), we compute $\Delta \boldsymbol{u}(k)$,

$$\begin{aligned} \Delta \boldsymbol{u}(k) &= \overbrace{\left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{array}\right]}^{N_p \text{ block matrices}} \left(\boldsymbol{R} + \boldsymbol{Z}^\top \boldsymbol{Q} \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\top \boldsymbol{Q} \left(\boldsymbol{r}_p - \boldsymbol{W} \boldsymbol{x}_a\right) \\ &= \boldsymbol{K_r} \boldsymbol{r}_p - \boldsymbol{K_x} \Delta \boldsymbol{x}(k) - \boldsymbol{K_y} \boldsymbol{y}(k), \end{aligned} \tag{22}$$

where

$$\boldsymbol{K_r} = \left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{array}\right] \left(\boldsymbol{R} + \boldsymbol{Z}^\top \boldsymbol{Q} \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\top \boldsymbol{Q}, \quad \boldsymbol{K_x} = \boldsymbol{K_r} \boldsymbol{W} \left[\begin{array}{c} \boldsymbol{I}_n \\ \boldsymbol{O} \end{array}\right],$$

and

$$\boldsymbol{K_y} = \boldsymbol{K_r} \boldsymbol{W} \left[\begin{array}{c} \boldsymbol{O} \\ \boldsymbol{I}_p \end{array}\right].$$

An implementation of the above controller, using a discrete-time integrator, is shown in Figure 5.

5 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.

5.1 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

$$\Delta u_i^{\min} \leq \Delta u_i(k) \leq \Delta u_i^{\max}, \quad i = 1, 2, \ldots, m \tag{23}$$

Let

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

Then, we can express (23) as

$$\Delta \boldsymbol{u}^{\min} \leq \Delta \boldsymbol{u}(k) \leq \Delta \boldsymbol{u}^{\max}. \tag{24}$$

The above can be equivalently represented as

$$\begin{bmatrix} -\boldsymbol{I}_m \\ \boldsymbol{I}_m \end{bmatrix} \Delta \boldsymbol{u}(k) \leq \begin{bmatrix} -\Delta \boldsymbol{u}^{\min} \\ \Delta \boldsymbol{u}^{\max} \end{bmatrix}. \tag{25}$$

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 $\Delta \boldsymbol{U}(k)$, by augmenting the above inequality to incorporate constraints for the remaining sampling times. That is, if the above constraints are imposed on the rate of change of the control action for all sampling times within the prediction horizon, then this can be expressed in terms of $\Delta \boldsymbol{U}(k)$ as

$$\begin{bmatrix} -\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & -\boldsymbol{I}_m & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_m & \cdots & \boldsymbol{O} & \boldsymbol{O} \\ \vdots & & & & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & -\boldsymbol{I}_m & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{I}_m & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} & -\boldsymbol{I}_m \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{I}_m \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}(k) \\ \Delta \boldsymbol{u}(k+1) \\ \vdots \\ \Delta \boldsymbol{u}(k+N_p-2) \\ \Delta \boldsymbol{u}(k+N_p-1) \end{bmatrix} \leq \begin{bmatrix} -\Delta \boldsymbol{u}^{\min} \\ \Delta \boldsymbol{u}^{\max} \\ -\Delta \boldsymbol{u}^{\min} \\ \Delta \boldsymbol{u}^{\max} \\ \vdots \\ -\Delta \boldsymbol{u}^{\min} \\ \Delta \boldsymbol{u}^{\max} \\ -\Delta \boldsymbol{u}^{\min} \\ \Delta \boldsymbol{u}^{\max} \end{bmatrix}. \tag{26}$$

On the other hand, if the constraints on the rate of change of the control are imposed only on the first component of $\Delta \boldsymbol{U}(k)$, then we express this in terms of $\Delta \boldsymbol{U}(k)$ as

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

5.2 Constraints on the Control Action Magnitude

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

$$u_i^{\min} \leq u_i(k) \leq u_i^{\max}, \quad i = 1, 2 \ldots, m \tag{27}$$

We now express constraints on the control action magnitude over the whole prediction horizon in term of $\Delta \boldsymbol{U}(k)$. To accomplish our goal, we first note that

$$\begin{aligned} \boldsymbol{u}(k) &= \boldsymbol{u}(k-1) + \Delta \boldsymbol{u}(k) \\ &= \boldsymbol{u}(k-1) + \left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{array} \right] \Delta \boldsymbol{U}(k), \end{aligned}$$

where, recall that $\Delta \boldsymbol{U}(k) = \left[\begin{array}{cccc} \Delta \boldsymbol{u}(k) & \Delta \boldsymbol{u}(k+1) & \cdots & \Delta \boldsymbol{u}(k+N_p-1) \end{array} \right]^\top$.

Similarly,

$$\begin{aligned} \boldsymbol{u}(k+1) &= \boldsymbol{u}(k) + \Delta \boldsymbol{u}(k+1) \\ &= \boldsymbol{u}(k-1) + \left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{array} \right] \Delta \boldsymbol{U}(k) + \Delta \boldsymbol{u}(k+1) \\ &= \boldsymbol{u}(k-1) + \left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{O} \end{array} \right] \Delta \boldsymbol{U}(k). \end{aligned}$$

Continuing in this manner, we obtain

$$\left[\begin{array}{c} \boldsymbol{u}(k) \\ \boldsymbol{u}(k+1) \\ \vdots \\ \boldsymbol{u}(k+N_p-1) \end{array} \right] = \left[\begin{array}{c} \boldsymbol{I}_m \\ \boldsymbol{I}_m \\ \vdots \\ \boldsymbol{I}_m \end{array} \right] \boldsymbol{u}(k-1) + \left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{O} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{I}_m \end{array} \right] \left[\begin{array}{c} \Delta \boldsymbol{u}(k) \\ \Delta \boldsymbol{u}(k+1) \\ \vdots \\ \Delta \boldsymbol{u}(k+N_p-1) \end{array} \right]. \quad (28)$$

Let

$$\boldsymbol{E} = \left[\begin{array}{c} \boldsymbol{I}_m \\ \boldsymbol{I}_m \\ \vdots \\ \boldsymbol{I}_m \end{array} \right] \quad \text{and} \quad \boldsymbol{H} = \left[\begin{array}{cccc} \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{O} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{I}_m & \boldsymbol{I}_m & \cdots & \boldsymbol{I}_m \end{array} \right].$$

Then, we can represent (28) as

$$\boldsymbol{U}(k) = \boldsymbol{E}\boldsymbol{u}(k-1) + \boldsymbol{H}\Delta \boldsymbol{U}(k). \quad (29)$$

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

$$\boldsymbol{U}^{\min} \leq \boldsymbol{U}(k) \leq \boldsymbol{U}^{\max}.$$

The above constraints can be equivalently represented as

$$\left[\begin{array}{c} -\boldsymbol{U}(k) \\ \boldsymbol{U}(k) \end{array} \right] \leq \left[\begin{array}{c} -\boldsymbol{U}^{\min} \\ \boldsymbol{U}^{\max} \end{array} \right],$$

Taking into account (29), we write the above as

$$\begin{bmatrix} -(\boldsymbol{E}\boldsymbol{u}(k-1)+\boldsymbol{H}\Delta\boldsymbol{U}(k)) \\ \boldsymbol{E}\boldsymbol{u}(k-1)+\boldsymbol{H}\Delta\boldsymbol{U}(k) \end{bmatrix} \leq \begin{bmatrix} -\boldsymbol{U}^{\min} \\ \boldsymbol{U}^{\max} \end{bmatrix}. \tag{30}$$

The above, in turn, can be represented as

$$\begin{bmatrix} -\boldsymbol{H} \\ \boldsymbol{H} \end{bmatrix} \Delta\boldsymbol{U}(k) \leq \begin{bmatrix} -\boldsymbol{U}^{\min}+\boldsymbol{E}\boldsymbol{u}(k-1) \\ \boldsymbol{U}^{\max}-\boldsymbol{E}\boldsymbol{u}(k-1) \end{bmatrix}. \tag{31}$$

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

$$\begin{bmatrix} -\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O} \end{bmatrix} \Delta\boldsymbol{U}(k) \leq \begin{bmatrix} -\boldsymbol{U}^{\min}+\boldsymbol{E}\boldsymbol{u}(k-1) \\ \boldsymbol{U}^{\max}-\boldsymbol{E}\boldsymbol{u}(k-1) \end{bmatrix}. \tag{32}$$

5.3 Constraints on the Plant Output

Recall from (19) that the predicted plant output is, $\boldsymbol{Y}(k) = \boldsymbol{W}\boldsymbol{x}_a(k) + \boldsymbol{Z}\Delta\boldsymbol{U}(k)$. Suppose now that the following constraints are imposed on the predicted plant's output,

$$\boldsymbol{Y}^{\min} \leq \boldsymbol{Y}(k) \leq \boldsymbol{Y}^{\max}.$$

We represent the above as

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

Taking into the account the expression for $\boldsymbol{Y}$ given by (19), we represent the above as

$$\begin{bmatrix} -\boldsymbol{Z} \\ \boldsymbol{Z} \end{bmatrix} \Delta\boldsymbol{U}(k) \leq \begin{bmatrix} -\boldsymbol{Y}^{\min}+\boldsymbol{W}\boldsymbol{x}_a(k) \\ \boldsymbol{Y}^{\max}-\boldsymbol{W}\boldsymbol{x}_a(k) \end{bmatrix}. \tag{33}$$

The above discussion clearly demonstrates that we need an effective method of minimizing a function of many variables, $J(\Delta\boldsymbol{U})$, subject to inequality constraints such as given by (25), (30), and (33), or their combination. In the following, we present a powerful method for solving such optimization problems. This is the method that we shall use to implement our MPCs.

6 An Optimizer for Solving Constrained Optimization Problems

If follows from the discussion in the previous section that at each sampling time the MPC calls for a solution to a constrained optimization problem of the form,

$$\begin{aligned} \text{minimize} \quad & J(\Delta \boldsymbol{U}) \\ \text{subject to} \quad & \boldsymbol{g}(\Delta \boldsymbol{U}) \leq \boldsymbol{0}, \end{aligned}$$

where $\boldsymbol{g}(\Delta \boldsymbol{U}) \leq \boldsymbol{0}$ contains inequality constrains given by (25), (30), and (33), or their combination. In this section, we present an iterative methods for solving the above optimization problems. To proceed, we first present the descent gradient method, followed by the Newton's method, for solving unconstrained optimization problems of the form,

$$\text{minimize } J(\Delta \boldsymbol{U})$$

6.1 Gradient Descent Method

For the sake of simplicity, we denote the argument of a function of many variables as $\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^N$, and the function that we will be minimizing will be denoted as f, where $f : \mathbb{R}^N \to \mathbb{R}$.

The method of the gradient descent is based on the following property of the gradient of a differentiable function, f, on $\mathbb{R}^N$:

Theorem 1 *At a given point $\boldsymbol{x}^{(0)}$, the vector*

$$\boldsymbol{v} = -\nabla f\left(\boldsymbol{x}^{(0)}\right)$$

points in the direction of most rapid decrease of f and the rate of increase of f at $\boldsymbol{x}^{(0)}$ in the direction $\boldsymbol{v}$ is $-\left\|\nabla f\left(\boldsymbol{x}^{(0)}\right)\right\|$, equivalently, the rate of decrease of f at $\boldsymbol{x}^{(0)}$ in the direction $\boldsymbol{v}$ is $\left\|\nabla f\left(\boldsymbol{x}^{(0)}\right)\right\|$.

Thus, if we wish to minimize a differentiable function, f, then moving in the direction of the negative gradient is a good direction. The gradient descent algorithm rests on the above observation and has the form,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha \nabla f\left(\boldsymbol{x}^{(k)}\right),$$

where $\alpha > 0$ is a step size.

6.2 Second-Order Necessary Conditions for a Minimum

Suppose we are given a function f of one variable x. Recall a well-known theorem referred to as the second-order Taylor's formula or the extended law of mean [8, p. 1]:

Theorem 2 *Suppose that $f(x)$, $f'(x)$, $f''(x)$ exist on the closed interval $[a,b] = \{x \in \mathbb{R} : a \leq x \leq b\}$. If x^*, x are any two different points of $[a,b]$, then there exists a point z strictly between x^* and x such that*

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + \frac{f''(z)}{2}(x - x^*)^2.$$

Using the above formula we observe that if

- $f'(x^*) = 0$, and
- $f''(x^*) > 0$,

then

$$f(x) = f(x^*) + \text{ a positive number}$$

for all x "close" to x^*. Indeed, if $f''(x)$ is continuous at x^* and $f''(x^*) > 0$, then $f''(x) > 0$ for all x in some neighborhood of x^*. Therefore,

$$f(x) > f(x^*) \quad \text{for all } x \text{ close to } x^*,$$

which means that x^* is a strict local minimizer of f.

We now extend the above result for functions of many variables. To proceed, we need the second-order Taylor's formula for such functions that can be found in [8, p. 11].

Theorem 3 *Suppose that $\boldsymbol{x}^*$, $\boldsymbol{x}$ are points in $\mathbb{R}^N$ and that f is a function of N variables with continuous first and second partial derivatives on some open set containing the line segment*

$$[\boldsymbol{x}^*, \boldsymbol{x}] = \left\{\boldsymbol{w} \in \mathbb{R}^N : \boldsymbol{w} = \boldsymbol{x}^* + t(\boldsymbol{x} - \boldsymbol{x}^*); 0 \leq t \leq 1\right\}$$

joining $\boldsymbol{x}^*$ and $\boldsymbol{x}$. Then there exists a $\boldsymbol{z} \in [\boldsymbol{x}^*, \boldsymbol{x}]$ such that

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*)^\top (\boldsymbol{x} - \boldsymbol{x}^*) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^\top \boldsymbol{F}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{x}^*), \tag{34}$$

where $\boldsymbol{F}(\cdot)$ is the Hessian of f, that is, the second derivative of f.

Hence if

- $\boldsymbol{x}^*$ is a critical point, that is, $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$, and

- $\boldsymbol{F}(\boldsymbol{x}^*) > 0$,

then using (34), we conclude that

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^*) + 0 + \text{a positive number}$$

for all $\boldsymbol{x}$ in a neighborhood of $\boldsymbol{x}^*$. Therefore for all $\boldsymbol{x} \neq \boldsymbol{x}^*$ in some neighborhood of $\boldsymbol{x}^*$, we have

$$f(\boldsymbol{x}) > f(\boldsymbol{x}^*),$$

which implies that $\boldsymbol{x}^*$ is a strict local minimizer.

6.3 Newton's Method

The idea behind Newton's method for function minimization is to minimize the quadratic approximation rather than the function itself as illustrated in Figure 6. Newton's method

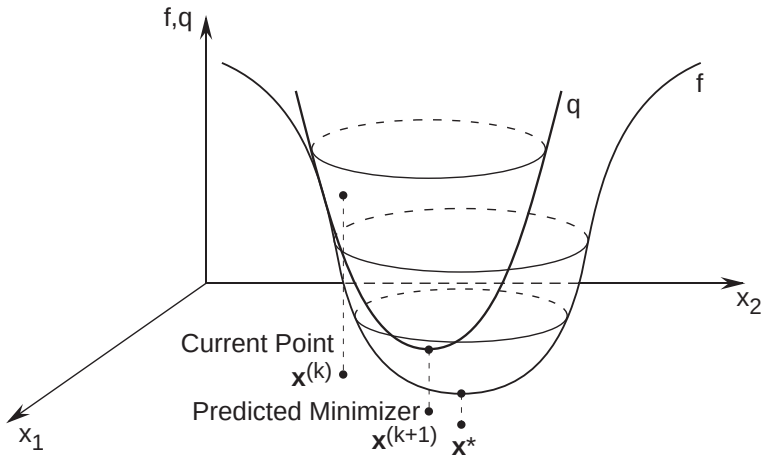

Figure 6: Newton's method minimizes the quadratic approximation of the objective function that utilizes first and second derivatives of the optimized function.

seeks a critical point, $\boldsymbol{x}^*$, of a given function. If at this critical point we have $\boldsymbol{F}(\boldsymbol{x}^*) > 0$, then $\boldsymbol{x}^*$ is a strict local minimizer of f.

We can obtain a quadratic approximation q of f at $\boldsymbol{x}^*$ from the second-order Taylor series expansion of f about $\boldsymbol{x}^*$,

$$q(\boldsymbol{x}) = f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*)^\top (\boldsymbol{x} - \boldsymbol{x}^*) + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}^*)^\top \boldsymbol{F}(\boldsymbol{x}^*) (\boldsymbol{x} - \boldsymbol{x}^*).$$

Note that

$$q(\boldsymbol{x}^*) = f(\boldsymbol{x}^*), \quad \nabla q(\boldsymbol{x}^*) = \nabla f(\boldsymbol{x}^*),$$

as well as their Hessians, that is, their second derivatives evaluated at $\boldsymbol{x}^*$ are equal.

A critical point of q can be obtained by solving the algebraic equation,

$$\nabla q(\boldsymbol{x}) = \mathbf{0},$$

that is, by solving the equation

$$\nabla q(\boldsymbol{x}) = \nabla f\left(\boldsymbol{x}^*\right) + \boldsymbol{F}\left(\boldsymbol{x}^*\right)\left(\boldsymbol{x} - \boldsymbol{x}^*\right) = \mathbf{0}.$$

Suppose now that we have a quadratic approximation of f at a point $\boldsymbol{x}^{(k)}$, that is,

$$q(\boldsymbol{x}) = f\left(\boldsymbol{x}^{(k)}\right) + \nabla f\left(\boldsymbol{x}^{(k)}\right)^\top \left(\boldsymbol{x} - \boldsymbol{x}^{(k)}\right) + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^{(k)}\right)^\top \boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)\left(\boldsymbol{x} - \boldsymbol{x}^{(k)}\right).$$

We assume that $\det \boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right) \neq 0$. Denoting the solution to the above equation as $\boldsymbol{x}^{(k+1)}$, we obtain

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)^{-1} \nabla f\left(\boldsymbol{x}^{(k)}\right)$$

The above is known as the Newton's method for minimizing a function of many variables f. Note that $\boldsymbol{x}^{(k+1)}$ is a critical point of the quadratic function q that approximates f at $\boldsymbol{x}^{(k)}$.

A computationally efficient representation of Newton's algorithm has the form,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \Delta\boldsymbol{x}^{(k)},$$

where $\Delta\boldsymbol{x}^{(k)}$ is obtained by solving the equation,

$$\boldsymbol{F}\left(\boldsymbol{x}^{(k)}\right)\Delta\boldsymbol{x}^{(k)} = \nabla f\left(\boldsymbol{x}^{(k)}\right).$$

7 Minimization Subject to Equality Constraints

We now discuss the problem of finding a point $\boldsymbol{x} \in \mathbb{R}^N$ that minimizes $f(\boldsymbol{x})$ subject to equality constraints,

$$\left.\begin{array}{rcl} h_1(\boldsymbol{x}) &=& 0 \\ h_2(\boldsymbol{x}) &=& 0 \\ &\vdots& \\ h_M(\boldsymbol{x}) &=& 0 \end{array}\right\}$$

where $M \leq N$. We write the above equality constraints in a compact form as

$$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0},$$

where $\boldsymbol{h} : \mathbb{R}^N \to \mathbb{R}^M$. We refer to the set of points satisfying the above constraints as the surface.

We now introduce the notion of the tangent plane to the surface $S = \{\boldsymbol{x} : \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}\}$ at a point $\boldsymbol{x}^* \in S$. First, we define a curve on the surface S as a family of points $\boldsymbol{x}(t) \in S$ continuously parameterized by t for $t \in [a, b]$. The curve is differentiable if $\dot{\boldsymbol{x}}(t) = d\boldsymbol{x}(t)/dt$ exists. A curve is said to pass through the point $\boldsymbol{x}^* \in S$ if $\boldsymbol{x}^* = \boldsymbol{x}(t^*)$ for some $t^* \in [a, b]$. The tangent plane to the surface $S = \{\boldsymbol{x} : \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}\}$ at $\boldsymbol{x}^*$ is the collection of the derivatives at $\boldsymbol{x}^*$ of all differentiable curves on S that pass through $\boldsymbol{x}^*$.

To proceed, we need the notion of a regular point of the constraints. We say that $\boldsymbol{x}^*$ satisfying the constraints, that is, $\boldsymbol{h}(\boldsymbol{x}^*) = \boldsymbol{0}$, is a regular point of the constraints if the gradient vectors,

$$\nabla h_1(\boldsymbol{x}^*), \ldots, \nabla h_M(\boldsymbol{x}^*)$$

are linearly independent.

One can show, see, for example [4, p. 298], that the tangent space at a regular point $\boldsymbol{x}^*$, denoted $T(\boldsymbol{x}^*)$, to the surface $\{\boldsymbol{x} : \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}\}$ at the regular point $\boldsymbol{x}^*$ is

$$T(\boldsymbol{x}^*) = \left\{ \boldsymbol{y} : \begin{bmatrix} \nabla h_1(\boldsymbol{x}^*)^\top \\ \vdots \\ \nabla h_M(\boldsymbol{x}^*)^\top \end{bmatrix} \boldsymbol{y} = \boldsymbol{0} \right\}. \tag{35}$$

With the above notions in place, we are ready to state and prove the following result which is known as the first-order necessary condition (FONC) for function minimization subject to equality constraints.

Theorem 4 *Let $\boldsymbol{x}^*$ be a local minimizer (or maximizer) of f subject to the constraints $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$ and let $\boldsymbol{x}^*$ be a regular point of the constraints. Then there exists a vector $\boldsymbol{\lambda}^*$ such that*

$$\nabla f(\boldsymbol{x}^*) + \begin{bmatrix} \nabla h_1(\boldsymbol{x}^*) & \cdots & \nabla h_M(\boldsymbol{x}^*) \end{bmatrix} \boldsymbol{\lambda}^* = \boldsymbol{0}.$$

Proof Let $\boldsymbol{x}(t)$ be a differentiable curve passing through $\boldsymbol{x}^*$ on the surface $S = \{\boldsymbol{x} : \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}\}$ such that $\dot{\boldsymbol{x}}(t^*) = \boldsymbol{y}$ where $t^* \in [a, b]$. Note that the tangent plane is given by (35). Because $\boldsymbol{x}^*$ is a local minimizer of f on S, we have

$$\frac{d}{dt} f(\boldsymbol{x}(t)) \bigg|_{t=t^*} = 0.$$

Applying the chain rule to the above gives

$$\nabla f\left(\boldsymbol{x}^*\right)^\top \boldsymbol{y} = 0.$$

Thus $\nabla f\left(\boldsymbol{x}^*\right)$ is orthogonal to the tangent space $T\left(\boldsymbol{x}^*\right)$. That is, $\nabla f\left(\boldsymbol{x}^*\right)$ is a linear combination of the gradients $\nabla h_1\left(\boldsymbol{x}^*\right), \ldots, \nabla h_M\left(\boldsymbol{x}^*\right)$. This fact can be expressed as

$$\nabla f\left(\boldsymbol{x}^*\right) + \left[\begin{array}{ccc} \nabla h_1\left(\boldsymbol{x}^*\right) & \cdots & \nabla h_M\left(\boldsymbol{x}^*\right) \end{array}\right] \boldsymbol{\lambda}^* = \mathbf{0}$$

for some constant vector $\boldsymbol{\lambda}^* \in \mathbb{R}^M$.

□

The vector $\boldsymbol{\lambda}^*$ is called the vector of Lagrange multipliers.

We now introduce the Lagrangian associated with the constrained optimization problem,

$$l(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{h}(\boldsymbol{x}).$$

Then the FONC can be expressed as

$$\begin{aligned} \nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \boldsymbol{\lambda}) &= \mathbf{0} \\ \nabla_{\boldsymbol{\lambda}} l(\boldsymbol{x}, \boldsymbol{\lambda}) &= \mathbf{0}. \end{aligned}$$

Note that the second of the above condition is equivalent to $\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$, that is,

$$\nabla_{\boldsymbol{\lambda}} l(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}.$$

Equivalently the FONC can be written as

$$\begin{aligned} \nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \boldsymbol{\lambda}) &= \mathbf{0} \\ \boldsymbol{h}(\boldsymbol{x}) &= \mathbf{0}. \end{aligned}$$

We now apply Newton's method to solve the above system of equations iteratively,

$$\left[\begin{array}{c} \boldsymbol{x}^{(k+1)} \\ \boldsymbol{\lambda}^{(k+1)} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{x}^{(k)} \\ \boldsymbol{\lambda}^{(k)} \end{array}\right] + \left[\begin{array}{c} \boldsymbol{d}^{(k)} \\ \boldsymbol{y}^{(k)} \end{array}\right],$$

where $\boldsymbol{d}^{(k)}$ and $\boldsymbol{y}^{(k)}$ are obtained by solving the matrix equation,

$$\left[\begin{array}{cc} \boldsymbol{L}\left(\boldsymbol{x}^{(k)}, \boldsymbol{\lambda}^{(k)}\right) & D\boldsymbol{h}\left(\boldsymbol{x}^{(k)}\right)^\top \\ D\boldsymbol{h}\left(\boldsymbol{x}^{(k)}\right) & \boldsymbol{O} \end{array}\right] \left[\begin{array}{c} \boldsymbol{d}^{(k)} \\ \boldsymbol{y}^{(k)} \end{array}\right] = \left[\begin{array}{c} -\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \boldsymbol{\lambda}) \\ -\boldsymbol{h}\left(\boldsymbol{x}^{(k)}\right) \end{array}\right],$$

where $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is the Hessian of $l(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$, and $D\boldsymbol{h}(\boldsymbol{x})$ is the Jacobian matrix of $\boldsymbol{h}(\boldsymbol{x})$, that is,

$$D\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} \nabla h_1(\boldsymbol{x})^\top \\ \vdots \\ \nabla h_M(\boldsymbol{x})^\top \end{bmatrix}.$$

The above algorithm is also referred to as in the literature as sequential quadratic programming (SQP); see, for example [7, Section 15.5].

8 A Lagrangian Algorithm for Equality Constraints

The first-order Lagrangian algorithm for the optimization problem involving minimizing f subject to the equality constraints, $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$, has the form,

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \alpha_k \left(\nabla f\left(\boldsymbol{x}^{(k)}\right) + D\boldsymbol{h}\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{\lambda}^{(k)} \right) \\ \boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \beta_k \boldsymbol{h}\left(\boldsymbol{x}^{(k)}\right), \end{aligned}$$

where α_k and β_k are positive constants. Note that the update for $\boldsymbol{x}^{(k)}$ is a descent gradient for minimizing the Lagrangian with respect to $\boldsymbol{x}$, while the update for $\boldsymbol{\lambda}^{(k)}$ is a gradient ascent for maximizing the Lagrangian with respect to $\boldsymbol{\lambda}$. For an analysis of the above algorithm, we recommend [2, pp. 522–524]

9 Minimization Subject to Inequality Constraints

We now discuss the problem of finding a point $\boldsymbol{x} \in \mathbb{R}^N$ that minimizes $f(\boldsymbol{x})$ subject to inequality constraints,

$$\left.\begin{aligned} g_1(\boldsymbol{x}) &\leq 0 \\ g_2(\boldsymbol{x}) &\leq 0 \\ &\vdots \\ g_P(\boldsymbol{x}) &\leq 0 \end{aligned}\right\}$$

We write the above inequality constraints in a compact form as

$$\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0},$$

where $\boldsymbol{h} : \mathbb{R}^N \to \mathbb{R}^P$.

Let $\boldsymbol{x}^*$ be a point satisfying the constraints, that is, $\boldsymbol{g}(\boldsymbol{x}^*) \leq \mathbf{0}$, and let J be the set of indices j for which $g_j(\boldsymbol{x}^*) = 0$. Then $\boldsymbol{x}^*$ is said to be a regular point of the constraints $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$ if the gradient vectors,

$$\nabla g_j(\boldsymbol{x}^*), \quad j \in J,$$

are linearly independent.

We say that a constraint $g_j(\boldsymbol{x}) \leq 0$ is active at $\boldsymbol{x}^*$ if $g_j(\boldsymbol{x}^*) = 0$. Thus the index set J, defined above, contains indices of active constraints.

Theorem 5 *Let $\boldsymbol{x}^*$ be a regular point and a local minimizer of f subject to $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$. Then there exists a vector $\boldsymbol{\mu}^* \in \mathbb{R}^P$ such that*

1. $\boldsymbol{\mu}^* \geq 0$,
2. $\nabla f(\boldsymbol{x}^*) + \begin{bmatrix} \nabla g_1(\boldsymbol{x}^*) & \cdots & \nabla g_P(\boldsymbol{x}^*) \end{bmatrix} \boldsymbol{\mu}^* = \mathbf{0}$,
3. $\boldsymbol{\mu}^{*\top} \boldsymbol{g}(\boldsymbol{x}^*) = 0$.

Proof First note that because $\boldsymbol{x}^*$ is a relative minimizer over the constraint set $\{\boldsymbol{x} : \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}\}$, it is also a minimizer over a subset of the constraint set obtained by setting the active constraints to zero. Therefore, for the resulting equality constrained problem, by Theorem 4, we have

$$\nabla f(\boldsymbol{x}^*) + \begin{bmatrix} \nabla g_1(\boldsymbol{x}^*) & \cdots & \nabla g_P(\boldsymbol{x}^*) \end{bmatrix} \boldsymbol{\mu}^* = \mathbf{0}, \tag{36}$$

where $\mu_j^* = 0$ if $g_j(\boldsymbol{x}^*) < 0$. This means that $\boldsymbol{\mu}^{*\top} \boldsymbol{g}(\boldsymbol{x}^*) = 0$. Thus μ_i may be non-zero only if the corresponding constraint is active, that is, $g_i(\boldsymbol{x}^*) = 0$.

It thus remains to show that $\mu^* \geq \mathbf{0}$. We prove this by contraposition. We suppose that for some $k \in J$, we have $\mu_k^* < 0$. Consider next a surface formed by all other active constraints, that is, the surface

$$\{\boldsymbol{x} : g_j(\boldsymbol{x}) = 0, j \in J, j \neq k\}. \tag{37}$$

Also consider a tangent space to the above surface at $\boldsymbol{x}^*$. By assumption $\boldsymbol{x}^*$ is a regular point of the active constraints. Therefore, there exists a vector $\boldsymbol{y}$ such that

$$\nabla g_k(\boldsymbol{y})^\top \boldsymbol{y} < 0.$$

Let now $\boldsymbol{x}(t)$, $t \in [-a, a]$, $a > 0$, be a curve on the surface (37) such that

$$\dot{\boldsymbol{x}}(0) = \boldsymbol{y}.$$

Note that for small t, the curve $\boldsymbol{x}(t)$ is feasible. We apply the transposition operator to the both sides of (36) to obtain

$$\nabla f\left(\boldsymbol{x}^*\right)^\top + \sum_{i=1}^{P} \mu_i^* \nabla g_i\left(\boldsymbol{x}^*\right)^\top = \mathbf{0}^\top.$$

Post-multiplying the above by $\boldsymbol{y}$ an taking into account the fact that $\boldsymbol{y}$ belongs to the tangent space to the surface (37) gives

$$\begin{aligned} \nabla f\left(\boldsymbol{x}^*\right)^\top \boldsymbol{y} &= -\mu_k^* \nabla g_k\left(\boldsymbol{x}^*\right)^\top \boldsymbol{y} \\ &< 0. \end{aligned}$$

Suppose, without loss of generality, that $\|\boldsymbol{y}\|_2 = 1$. Then, we have that

$$\begin{aligned} \frac{df(\boldsymbol{x}(t))}{dt} &= \nabla f\left(\boldsymbol{x}^*\right)^\top \boldsymbol{y} \\ &< 0, \end{aligned}$$

that is, the rate of increase of f at $\boldsymbol{x}^*$ in the direction $\boldsymbol{y}$ is negative. This would mean that we could decrease the value of f moving just slightly away from $\boldsymbol{x}^*$ along $\boldsymbol{y}$ while, at the same time, preserving feasibility. But this contradicts the minimality of $f\left(\boldsymbol{x}^*\right)$. In summary, if $\boldsymbol{x}^*$ is a relative minimizer then we also have the components of $\boldsymbol{\mu}^*$ all non-negative.

□

The vector $\boldsymbol{\mu}^*$ is called the vector of the Karush-Kuhn-Tucker (KKT) multipliers.

10 A Lagrangian Algorithm for Inequality Constraints

We now present a first-order Lagrangian algorithm for the optimization problem involving inequality constraints,

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}, \end{aligned}$$

where $\boldsymbol{g} : \mathbb{R}^N \to \boldsymbol{g} : \mathbb{R}^P$. The Lagrangian function is

$$l(\boldsymbol{x}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \boldsymbol{\mu}^\top \boldsymbol{g}(\boldsymbol{x}).$$

The first-order Lagrangian algorithm for the above optimization problem involving minimizing f subject to the inequality constraints, $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$, has the form,

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \alpha_k \left(\nabla f\left(\boldsymbol{x}^{(k)}\right) + D\boldsymbol{g}\left(\boldsymbol{x}^{(k)}\right)^\top \boldsymbol{\mu}^{(k)} \right) \\ \boldsymbol{\mu}^{(k+1)} &= \left[\boldsymbol{\mu}^{(k)} + \beta_k \boldsymbol{g}\left(\boldsymbol{x}^{(k)}\right) \right]_+, \end{aligned}$$

where the operation $[\cdot]_+ = \max(\cdot, 0)$ is applied component-wise.

For an analysis of the above algorithm, we recommend [2, pp. 524–528]

Example 1 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,

$$\left.\begin{array}{rcl} \dot{\boldsymbol{x}} &=& \left[\begin{array}{cc} -0.1 & -3.0 \\ 1 & 0 \end{array}\right] \boldsymbol{x} + \left[\begin{array}{c} 1 \\ 0 \end{array}\right] u \\ y &=& \left[\begin{array}{cc} 0 & 10 \end{array}\right] \boldsymbol{x}. \end{array}\right\} \tag{38}$$

The system with the transfer function as the above system was used by Wang [9, 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 (20) 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 (32). Thus, in our example, the constraints can be written as,

$$\left[\begin{array}{ccc} -1 & 0 & 0 \\ 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta u(k) \\ \Delta u(k+1) \\ \Delta u(k+2) \end{array}\right] \leq \left[\begin{array}{c} 0.3 + u(k-1) \\ 0.5 - u(k-1) \end{array}\right].$$

We employed the first-order Lagrangian algorithm for inequality constraints presented in Section 10, where $\alpha_k = \beta_k = 0.005$. We set zero initial conditions. Plots of the control effort and the plant's output are shown in Figure 7. 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

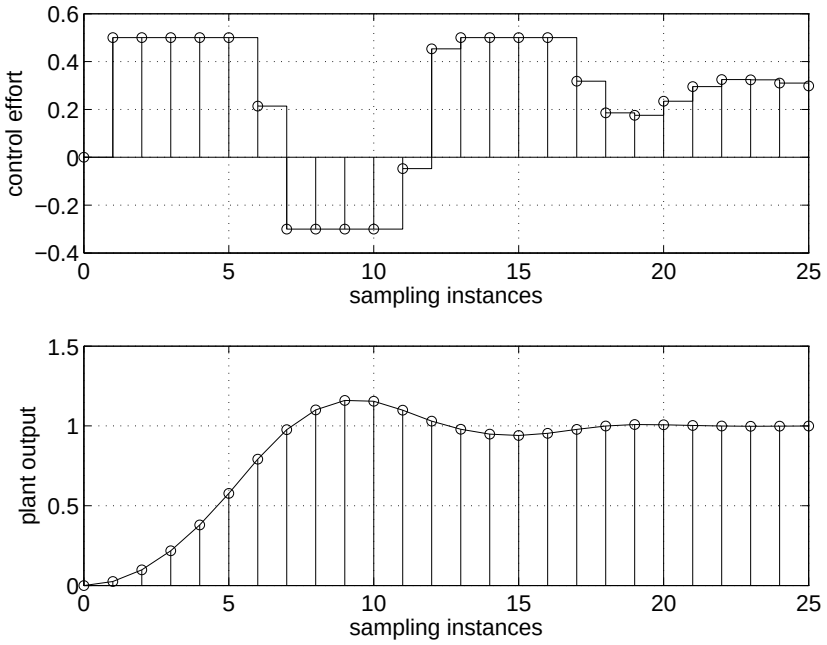

Figure 7: Plots of control effort and the plant's output versus time of the closed-loop system of Example 1 for the case when the constraint was imposed only on the first component of $\boldsymbol{U}(k)$.

on all the elements of $\boldsymbol{U}(k)$. We use (31) to express the constraints, where

$$\begin{bmatrix} -1 & 0 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & -1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \Delta u(k+2) \end{bmatrix} \leq \begin{bmatrix} 0.3 + u(k-1) \\ 0.3 + u(k-1) \\ 0.3 + u(k-1) \\ 0.5 - u(k-1) \\ 0.5 - u(k-1) \\ 0.5 - u(k-1) \end{bmatrix}.$$

Example 2 In this example, we apply the discrete MPC controller from Example 1 to the continuous plant modeled by (38). The MPC controller's objective is to force the plant's 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 9, 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

$$-0.3 \leq u(k) \leq 0.5$$

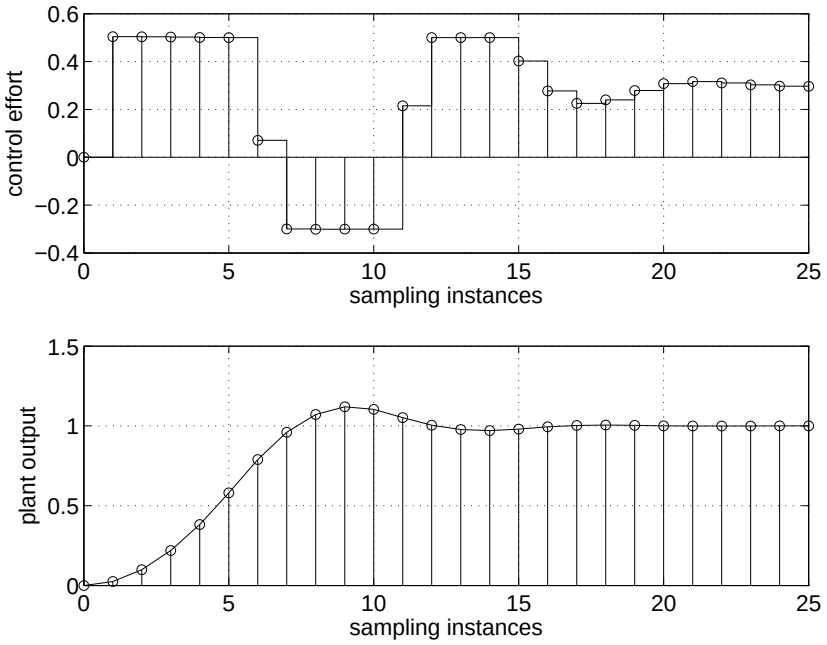

Figure 8: Plots of control effort and the plant's output versus time of the closed-loop system of Example 1 for the case when the constraints were imposed on all the components of $\boldsymbol{U}(k)$.

11 Observer-Based MPC

If the plant state is not accessible, we need to use an observer in the controller implementation as shown in Figure 10.

12 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 [3]. For some impressive industrial applications of MPCs see [3, 1]. Nonlinear model predictive controllers are presented in [5, 6].

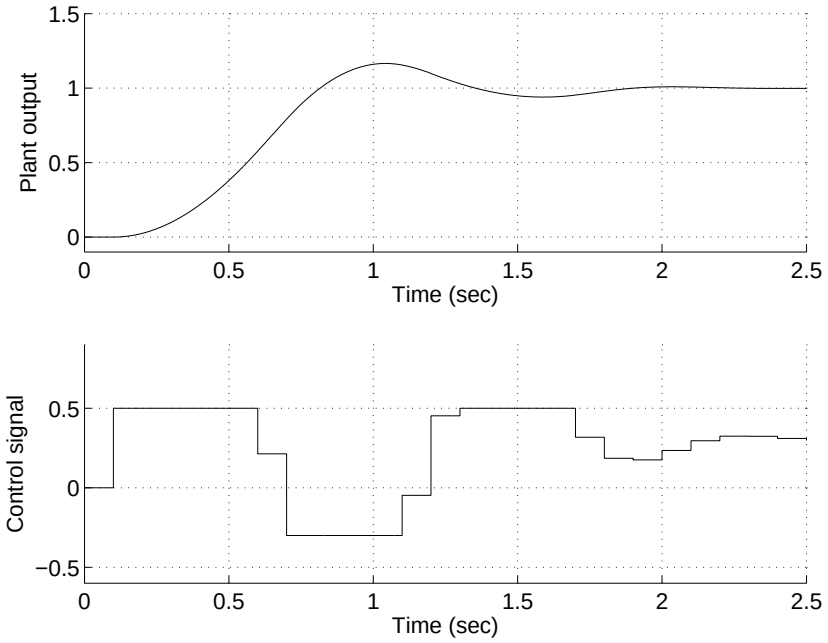

Figure 9: Plots of control effort and the plant's output versus time of the closed-loop system of Example 2 for the case when the constraints were imposed only on the first component of $\boldsymbol{U}(k)$.

References

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

[2] E. K. P. Chong and S. H. Żak. *An Introduction to Optimization*. Wiley-Interscience, John Wiley & Sons, Inc., New York, third edition, 2008.

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

[4] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing, Reading, Massachusetts, second edition, 1984.

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

[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.

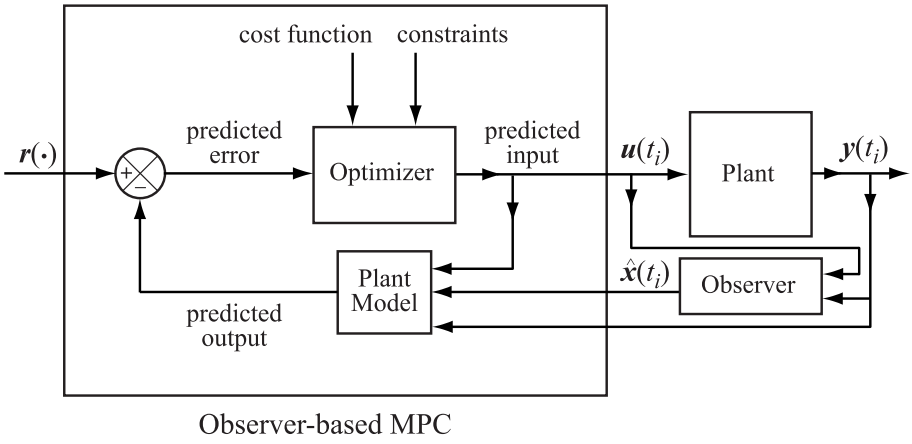

Figure 10: MPC implementation when the plant's state is not available.

[7] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, New York, 1996.

[8] A. L. Peressini, F. E. Sullivan, and J. J. Uhl, Jr. *The Mathematics of Nonlinear Programming*. Springer-Verlag, New York, 1988.

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