Lecture 2: Introduction to Model Predictive Control

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

What is Model Predictive Control(MPC)

A class of algorithms that

- compute the trajectory of manipulated variable adjustment to optimise the future behaviour of a plant

Strength of the algorithms

- Multivariable control
- Constrained control
- Real-time optimisation

History of MPC

- The design principle originated in the eletrical engineering field in the end of 1960s. (See for example, (Kleinman, 1970) and (Thomas,1975)
- In the end of 1970s, chemical engineers in Shell and a couple of other petrol-chemical companies independently investigated the design ideas of MPC and brought it to industrial applications. (See for example, Cutler and Ramaker, 1979, Garcia and Morshedi, 1986)
- It is one of the few areas that has received on-going interest from researchers in both industry and universities.

Generations of MPC I

First generation of MPC– also called 'Dynamics Matrix Control' (DMC) (1970-now)

- Design based on plant step response model.
- Model structure is intuitive and attractive to the plant engineers.
- However, the structure may require many parameters to capture plant dynamic response and is limited to stable plants.
- There are still some companies using this version.

Generations of MPC

Generalised Predictive Control (GPC) 1986-current

- Design using transfer function models.
- Parsimonious model structure.
- However, it has some difficulties in dealing with multivariable systems.

Generations of MPC

Current Generation of MPC

- Design using state space model.
- Parsimonious model structure.
- Easy to deal with multivariable systems.
- Extension to nonlinear systems.
- Use of observer in the design (possibly a down side).

We will use state space model in this course.

Design Principles

The core of MPC algorithms carries the same design principles

- Design using a moving horizon window for on-line optimization
  - Solve for optimal future control trajectory within a limited time window
- Receding horizon control
  - Compute the optimal control trajectory for every sampling instant
  - Implement the first sample of the optimal control while ignoring the rest of the control trajectory.

Key Terminologies

- Moving horizon window: the time dependent window from an arbitrary time k_i to $k_i + N_p$. The length of the window N_p (measured by the number of samples) remains constant.
- Prediction horizon: equals the length of the moving horizon window N_p and dictates how 'far' we wish the future to be predicted for.

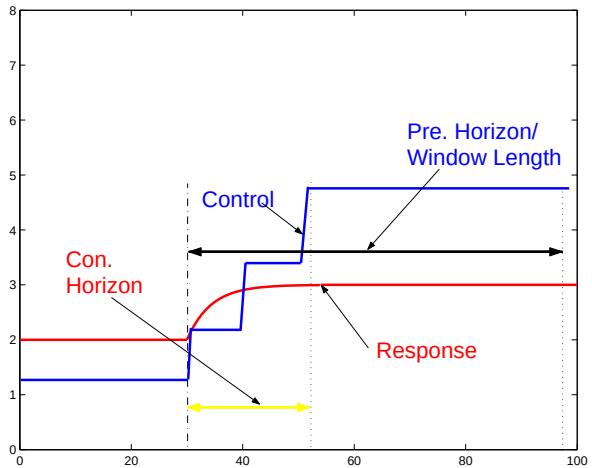


Diagram for moving horizon window

Terminologies- continue

- Receding horizon control: although the optimal trajectory of future control signal is completely captured within the moving horizon window, the actual control input to the plant only takes the first sample of the control signal while neglecting the rest of the trajectory.
- Control horizon: denoted by N_c and measured by number of samples, dictates how ‘long’ we wish the control signal to be active for before it reaches a steady state.

First Mainstream Approach

- Optimizing the incremental of control signal (DMC and GPC and other process control based algorithms)
  - elementary approaches
  - naturally embedded integral actions
  - convenient for plant implementation

Plant Model

For simplicity, we begin our study by assuming that the underlying plant is a single-input and single-output system, described by:

$$x_m(k+1) = A_m x_m(k) + B_m u(k) \quad (1)$$

$$y(k) = C_m x_m(k) \quad (2)$$

- u is the manipulated variable or input variable; y is the process output; and x_m is the state variable vector with assumed dimension n_1.
- Notice that this plant model has $u(k)$ as its input.

Thus, we need to change the model to suit our design purpose in which an integrator is embedded.

Difference of the Signals

Take a difference operation on both sides of (1), we obtain that

$$x_m(k+1) - x_m(k) = A_m(x_m(k) - x_m(k-1)) + B_m(u(k) - u(k-1)$$

Denote the difference of the state variable

$$\Delta x_m(k+1) = x_m(k+1) - x_m(k); \quad \Delta x_m(k) = x_m(k) - x_m(k-1)$$

and the difference of the control variable

$$\Delta u(k) = u(k) - u(k-1)$$

These are the increments of the variables $x_m(k)$ and $u(k)$. With this transformation, the difference of the state space equation is:

(3) $$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k)$$

Model for Design-I

Note that the input to the state-space model is $\Delta u(k)$. The next step is to connect $\Delta x_m(k)$ to the output $y(k)$. To do so, a new state variable vector is chosen to be

$$x(k) = \begin{bmatrix} \Delta x_m(k)^T & y(k) \end{bmatrix}^T$$

where superscript ^{T} indicates matrix transpose. Note that

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k)) = C_m \Delta x_m(k+1) \\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) \end{aligned} \tag{4}$$

Model for Design-II

Putting together (3) with (4) leads to the following state-space model

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} o_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{5}$$

where $o_m = \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}}^{n_1}$. The triplet (A, B, C) is called the augmented model which will be used in the design of predictive control.

Example

Consider a discrete-time model in the following form

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \tag{6}$$

where the system matrices are

$$A_m = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}; B_m = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}; C_m = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Find the triplet matrices (A, B, C) in the augmented model (5) and calculate the eigenvalues of the system matrix, A, of the augmented model.

Example Continues

Solution. From (5), $n_1 = 2$ and $o_m = [0\ 0]$. The augmented model for this plant is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{7}$$

where the augmented system matrices are

$$A = \begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix};\ B = \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix};$$

$$C = \begin{bmatrix} o_m & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

The characteristic equation of matrix A is given by

$$\begin{aligned} \rho(\lambda) &= det(\lambda I - A) = det \begin{bmatrix} \lambda I - A_m & o_m^T \\ -C_m A_m & (\lambda - 1) \end{bmatrix} \\ &= (\lambda - 1) det(\lambda I - A_m) = (\lambda - 1)^3 \end{aligned} \tag{8}$$

Therefore, the augmented state-space model has three eigenvalues at $\lambda = 1$. Among them, two are from the original integrator plant, and one is from the augmentation of the plant model.

MATLAB Tutorial

The objective of this tutorial is to demonstrate how to obtain a discrete-time state-space model from a continuous-time state-space model, and form the augmented discrete-time state-space model. Consider a continuous-time system has the state space model

$$\begin{aligned} \dot{x}_m(t) &= \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_m(t) + \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} x_m(t) \end{aligned} \tag{9}$$

Tutorial Solution

Step by step

1. Create a new file called extmodel.m. We form a continuous-time state variable model; then this continuous-time model is discretized using MATLAB function 'c2dm' with specified sampling interval Δt.
2. Enter the following program into the file:

```
Ac = [0  1  0; 3  0  1; 0 1 0 ]; Bc= [1; 1; 3]; Cc=[0 1 0];
Dc=zeros(1,1); Delta_t=1; [Ad,Bd,Cd,Dd]=c2dm(Ac,Bc,Cc,Dc,Delta_t);
```

3. The dimensions of the system matrices are determined to find out the numbers of states, inputs and outputs. The augmented state-space model is produced. Continue entering the following program into the file:

```
[m1,n1]=size(Cd); [n1,n_in]=size(Bd); A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ad; A_e(n1+1:n1+m1,1:n1)=Cd*Ad;
B_e=zeros(n1+m1,n_in); B_e(1:n1,:)=Bd; B_e(n1+1:n1+m1,:)=Cd*Bd;
C_e=zeros(m1,n1+m1); C_e(:,n1+1:n1+m1)=eye(m1,m1);
```

4. Run this program to produce the augmented state variable model for the design of predictive control.

Tutorial Exercise

A one degree of freedom mass-spring damped system is described by the differential equation

$$m\frac{dq(t)^2}{dt^2} + c\frac{dq(t)}{dt} + kq(t) = u(t)$$

where q is the position of the mass and $u(t)$ is external force. the By choosing the position and velocity of the mass as the state variables $x_1(t)$ and $x_2(t)$, we obtain the continuous-time state space model as

$$\begin{aligned} \dot{x}_m(t) &= \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} x_m(t) + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(t) \end{aligned} \tag{10}$$

where we assume $m = 3$, $k = 1$ and $c = 0.5$.

Tutorial Exercise-continues

1. Discretize this continuous-time state space model to obtain discrete-time model using sampling interval $\Delta t = 0.1$.
2. Calculate the output response $y(k)$ using the discrete-time model, where $u(k) = 0$, but the initial condition $x_m(0) = [0.5\ 1.6]^T$. Interpret your simulation results in terms of the physical mass-spring damped system.
3. Increase m to 100, and repeat the previous simulation and observe the difference between the two responses. Interpret your simulation results in terms of the physical mass-spring damped system.
4. Experiment with the variation of sampling interval. Increase your sampling interval to $\Delta t = 10$, and repeat the previous two simulations. Discuss the role of sampling interval in the discrete-time model.

Lecture 3: Prediction and Optimization in Predictive Control

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Prediction-overview

- Upon formulation of the mathematical model, the next step in the design of predictive control system is to calculate the predicted plant output with the future control signal as the adjustable variables.
- This prediction is described within an optimization window.
- We will examine in detail the optimization carried out within this window.
- Here we assume that the current time is k_i and the length of the optimization window is N_p as number of samples.

Definition of Variables

- Assuming that at the sampling instant k_i, $k_i > 0$, the state variable vector $x(k_i)$ is available through measurement, the state $x(k_i)$ provides the current plant information.
- The future control trajectory is denoted by

$$\Delta u(k_i), \Delta u(k_i + 1), \ldots, \Delta u(k_i + N_c - 1)$$

  where N_c is called control horizon dictating the number of parameters used to capture the future control trajectory.
- With given information $x(k_i)$, the future state variables are predicted for N_p number of samples, where N_p is called prediction horizon. N_p is also the length of the optimization window.
- We denote the future state variables as:

$$x(k_i + 1 \mid k_i),\ x(k_i + 2 \mid k_i),\ \ldots,\ x(k_i + m \mid k_i),\ \ldots,\ x(k_i + N_p \mid k_i)$$

  where $x(k_i + m \mid k_i)$ is the predicted state variable at $k_i + m$ with given current plant information $x(k_i)$.
- The control horizon N_c is chosen to be less than (or equal to) the prediction horizon N_p.

Calculation of the Prediction

Based on the state-space model (A, B, C), the future state variables are calculated sequentially using the set of future control parameters

$$\begin{aligned}
x(k_i+1 \mid k_i) &= Ax(k_i) + B\Delta u(k_i) \\
x(k_i+2 \mid k_i) &= Ax(k_i+1 \mid k_i) + B\Delta u(k_i+1) \\
&= A^2x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i+1) \\
&\vdots \\
x(k_i+N_p \mid k_i) &= A^{N_p}x(k_i) + A^{N_p-1}B\Delta u(k_i) + A^{N_p-2}B\Delta u(k_i+1) \\
&+ \ldots + A^{N_p-N_c}B\Delta u(k_i+N_c-1)
\end{aligned}$$

From the predicted state variables, the predicted output variables are by substitution

(1)

$$\begin{aligned}
y(k_i+1 \mid k_i) &= CAx(k_i) + CB\Delta u(k_i) \\
y(k_i+2 \mid k_i) &= CA^2x(k_i) + CAB\Delta u(k_i) + CB\Delta u(k_i+1) \\
y(k_i+3 \mid k_i) &= CA^3x(k_i) + CA^2B\Delta u(k_i) + CAB\Delta u(k_i+1) \\
&+ CB\Delta u(k_i+2) \\
&\vdots \\
y(k_i+N_p \mid k_i) &= CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta u(k_i) + CA^{N_p-2}B\Delta u(k_i+1) \\
&+ \ldots + CA^{N_p-N_c}B\Delta u(k_i+N_c-1)
\end{aligned}$$

(2)

Prediction in Matrix Form

Define vectors

$$Y = \left[\begin{array}{ccccc} y(k_i+1 \mid k_i) & y(k_i+2 \mid k_i) & y(k_i+3 \mid k_i) & \dots & y(k_i+N_p \mid k_i) \end{array} \right]^T$$

$$\Delta U = \left[\begin{array}{ccccc} \Delta u(k_i) & \Delta u(k_i+1) & \Delta u(k_i+2) & \dots & \Delta u(k_i+N_c-1) \end{array} \right]^T$$

where the dimension of Y is N_p and the dimension of ΔU is N_c. We collect (1)- (2) together in a compact matrix form as

(3) $$Y = Fx(k_i) + \Phi \Delta U$$

where

$$F = \left[\begin{array}{c} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{array} \right] ; \Phi = \left[\begin{array}{ccccc} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{array} \right]$$

Objective for Predictive Control

- For a given setpoint signal $r(k_i)$ at sample time k_i, within a prediction horizon the objective of the predictive control system is to bring the predicted output as close as possible to the setpoint signal, where we assume that the set-point signal remains constant in the optimization window.
- This objective is then translated into a design to find the 'best' control parameter vector ΔU such that an error function between the set-point and the predicted output is minimized.

Objective in Mathematical Formulation

Assuming that the data vector that contains the set-point information is

$$R_s^T = \overbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}}^{N_p} r(k_i)$$

we define the cost function J that reflects the control objective as

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U \tag{4}$$

- where the first term is linked to the objective of minimizing the errors between the predicted output and the setpoint signal,
- the second term reflects the consideration given to the size of ΔU when the objective function J is made to be as small as possible.

Weight Matrix $\bar{R}$

$\bar{R}$ is a diagonal matrix in the form that $\bar{R} = r_w I_{N_c \times N_c}$ ($r_w \geq 0$) where r_w is used as a tuning parameter for the desired closed-loop performance.

- For the case that $r_w = 0$, the cost function (4) is interpreted as the situation where we would not want to pay any attention to how large the ΔU might be and our goal would be solely to make the error $(R_s - Y)^T (R_s - Y)$ as small as possible.
- For the case of large r_w, the cost function (4) is interpreted as the situation where we would carefully consider how large the ΔU might be and cautiously reduce the error $(R_s - Y)^T (R_s - Y)$.
- The parameter r_w in $\bar{R}$ is a performance tuning parameter. For instance, if we want a faster closed-loop response, then r_w should be selected smaller.

Optimal Solution

To find the optimal ΔU that will minimize J, by using (3), J is expressed as

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U \quad (5)$$

From the first derivative of the cost function J:

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - Fx(k_i)) + 2(\Phi^T \Phi + \bar{R}) \Delta U \quad (6)$$

the necessary condition of the minimum J is obtained as

$$\frac{\partial J}{\partial \Delta U} = 0$$

from which we find the optimal solution for the control signal as

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)) \quad (7)$$

with the assumption that $(\Phi^T \Phi + \bar{R})^{-1}$ exists.

Example

Suppose that a first order system is described by the state equation

$$\begin{aligned} x_m(k+1) &= ax_m(k) + bu(k) \\ y(k) &= x_m(k) \end{aligned} \tag{8}$$

where $a = 0.8$ and $b = 0.1$ are scalars.

- Find the augmented state- space model.
- Assuming a prediction horizon $N_p = 10$ and control horizon $N_c = 4$, calculate the components which form the prediction of future output Y, and the quantities $\Phi^T\Phi$, $\Phi^T F$ and $\Phi^T \bar{R}_s$.
- Assuming that at a time k_i ($k_i = 10$ for this example), $r(k_i) = 1$ and the state vector $x(k_i) = [0.1 \;\; 0.2]^T$, find the optimal solution ΔU with respect to the cases where $r_w = 0$ and $r_w = 10$, and compare the results.

Solution-I

The augmented state-space equation is

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} a & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} b \\ b \end{bmatrix} \Delta u(k)$$

$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \quad (9)$$

Based on Equation (3), the F and Φ matrices take the following forms:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \\ CA^6 \\ CA^7 \\ CA^8 \\ CA^9 \\ CA^{10} \end{bmatrix}; \Phi = \begin{bmatrix} CB & 0 & 0 & 0 \\ CAB & CB & 0 & 0 \\ CA^2B & CAB & CB & 0 \\ CA^3B & CA^2B & CAB & CB \\ CA^4B & CA^3B & CA^2B & CAB \\ CA^5B & CA^4B & CA^3B & CA^2B \\ CA^6B & CA^5B & CA^4B & CA^3B \\ CA^7B & CA^6B & CA^5B & CA^4B \\ CA^8B & CA^7B & CA^6B & CA^5B \\ CA^9B & CA^8B & CA^7B & CA^6B \end{bmatrix}$$

Solution-II

With the plant parameters $a = 0.8$ and $b = 0.1$, $N_p = 10$ and $N_c = 4$, we calculate the quantities

$$\Phi^T\Phi = \begin{bmatrix} 1.1541 & 1.0407 & 0.9116 & 0.7726 \\ 1.0407 & 0.9549 & 0.8475 & 0.7259 \\ 0.9116 & 0.8475 & 0.7675 & 0.6674 \\ 0.7726 & 0.7259 & 0.6674 & 0.5943 \end{bmatrix}$$

$$\Phi^T F = \begin{bmatrix} 9.2325 & 3.2147 \\ 8.3259 & 2.7684 \\ 7.2927 & 2.3355 \\ 6.1811 & 1.9194 \end{bmatrix}; \Phi^T \bar{R}_s = \begin{bmatrix} 3.2147 \\ 2.7684 \\ 2.3355 \\ 1.9194 \end{bmatrix}$$

- Note that the vector $\Phi^T \bar{R}_s$ is identical to the last column in the matrix $\Phi^T F$. This is because the last column of F matrix is identical to $\bar{R}_s$.

Solution -III

With the plant parameters $a = 0.8$ and $b = 0.1$, $N_p = 10$ and $N_c = 4$, we calculate the quantities

$$\Phi^T\Phi = \begin{bmatrix} 1.1541 & 1.0407 & 0.9116 & 0.7726 \\ 1.0407 & 0.9549 & 0.8475 & 0.7259 \\ 0.9116 & 0.8475 & 0.7675 & 0.6674 \\ 0.7726 & 0.7259 & 0.6674 & 0.5943 \end{bmatrix}$$

$$\Phi^T F = \begin{bmatrix} 9.2325 & 3.2147 \\ 8.3259 & 2.7684 \\ 7.2927 & 2.3355 \\ 6.1811 & 1.9194 \end{bmatrix}; \Phi^T \bar{R}_s = \begin{bmatrix} 3.2147 \\ 2.7684 \\ 2.3355 \\ 1.9194 \end{bmatrix}$$

Note that the vector $\Phi^T \bar{R}_s$ is identical to the last column in the matrix $\Phi^T F$. This is because the last column of F matrix is identical to $\bar{R}_s$.

Solution-IV

At time $k_i = 10$, the state vector $x(k_i) = [0.1 \;\; 0.2]^T$. In the first case, the error between predicted Y and R_s is reduced without any consideration to the magnitude of control changes. Namely, $r_w = 0$. Then the optimal ΔU is found through the calculation

$$\Delta U = (\Phi^T \Phi)^{-1}(\Phi^T R_s - \Phi^T F x(k_i)) = \begin{bmatrix} 7.2 & -6.4 & 0 & 0 \end{bmatrix}^T$$

We notice that without weighting on the incremental control, the last two elements $\Delta u(k_i + 2) = 0$ and $\Delta u(k_i + 3) = 0$, while the first two elements have a rather large magnitude.

Solution - V

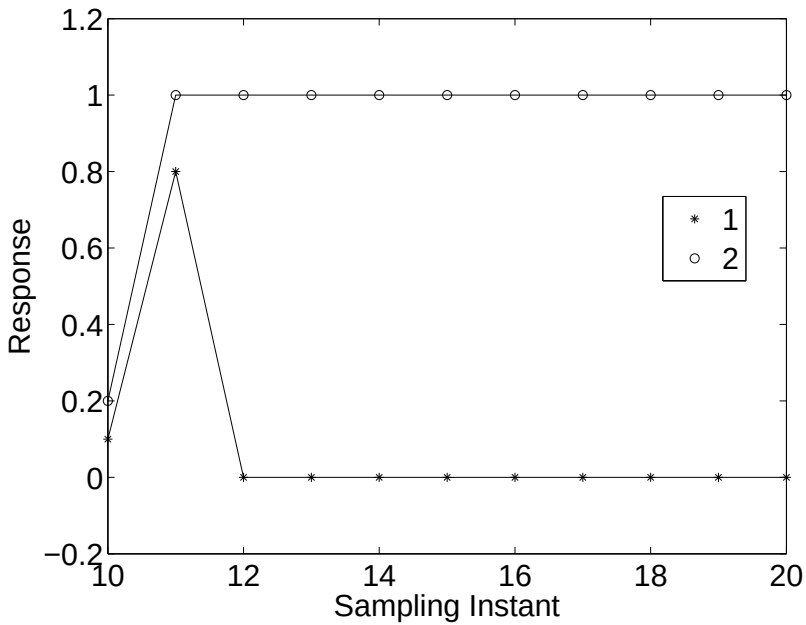
To examine the effect of the weight r_w on the optimal solution of the control, we let $r_w = 10$. The optimal solution of ΔU is given below where I is a 4×4 identity matrix.

(10)
$$\begin{aligned} \Delta U &= (\Phi^T \Phi + 10 \times I)^{-1}(\Phi^T R_s - \Phi^T F x(k_i)) \\ &= \begin{bmatrix} 0.1269 & 0.1034 & 0.0829 & 0.065 \end{bmatrix}^T \end{aligned}$$

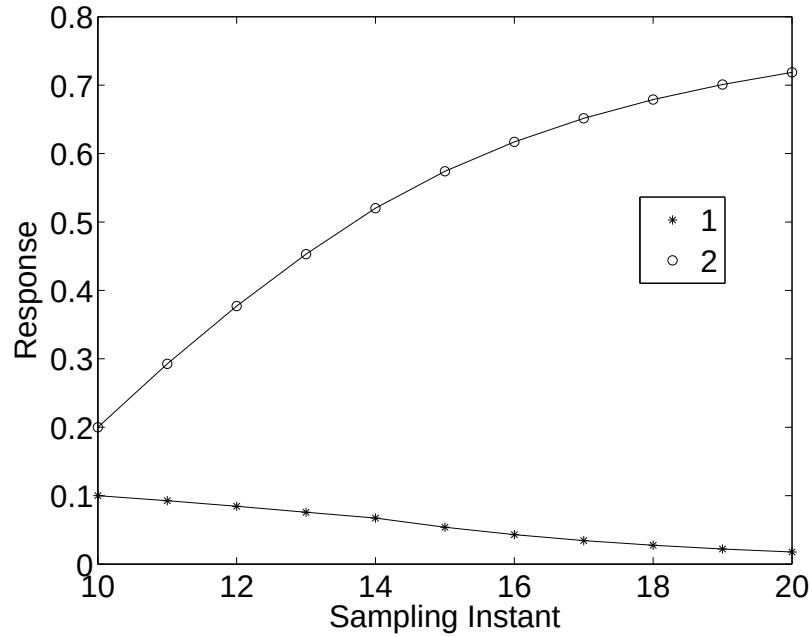
With this choice, the magnitude of the first two control increments is significantly reduced, also the last two components are no longer zero.

Solution- VI

Figure 1a shows the changes of the state variables where we can see that the predicted output $y(.)$ reached the desired setpoint 1 while the $\Delta x_m(.)$ reaches zero. Figure 1b shows the optimal state variables. It is seen that the output $y(.)$ did not reach the setpoint value of 1, however, the $\Delta x_m(.)$ approaches zero.



(a) State variables with no weight on Δu



(b) State variables with weight on Δu

Figure 1: Comparison of optimal solutions. $1 - *$ is Δx_m; $2 - o$ is y

MATLAB Tutorial: Computation of MPC Gains

The objective of this tutorial is to produce a MATLAB function for calculating $\Phi^T\Phi$, $\Phi^T F$, $\Phi^T \bar{R}_s$. The key here is to create F and Φ matrices. Φ matrix is a Toeplitz matrix, which is created by defining its first column, and the next column is obtained through shifting the previous column.

Tutorial Solution- I

Step by Step

- Create a new file called mpcgain.m.
- The first step is to create the augmented model for MPC design. The input parameters to the function are the state space model (A_p, B_p, C_p), prediction horizon N_p and control horizon N_c. Enter the following program into the file:

```
function [Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]
                  =mpcgain(Ap,Bp,Cp,Nc,Np);
[m1,n1]=size(Cp); [n1,n_in]=size(Bp); A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ap; A_e(n1+1:n1+m1,1:n1)=Cp*Ap;
B_e=zeros(n1+m1,n_in); B_e(1:n1,:)=Bp; B_e(n1+1:n1+m1,:)=Cp*Bp;
C_e=zeros(m1,n1+m1); C_e(:,n1+1:n1+m1)=eye(m1,m1);
```

- Note that the F and Phi matrices have special forms. By taking advantage of the special structure, we obtain the matrices.

Tutorial Solution -II

- Continue entering the program into the file:

```
n=n1+m1; h(1,:)=C_e; F(1,:)=C_e*A_e; for kk=2:Np
  h(kk,:)=h(kk-1,:)*A_e;
    F(kk,:)= F(kk-1,:)*A_e;
end v=h*B_e;
Phi=zeros(Np,Nc);  %declare the dimension of Phi
Phi(:,1)=v;        % first column of Phi
for i=2:Nc
   Phi(:,i)=[zeros(i-1,1);v(1:Np-i+1,1)];  %Toeplitz matrix
end
BarRs=ones(Np,1); Phi_Phi= Phi'*Phi; Phi_F= Phi'*F;
Phi_R=Phi'*BarRs;
```

- Type into the MATLAB Work Space with $Ap = 0.8$, $Bp = 0.1$, $Cp = 1$, $Nc = 4$ and $Np = 10$. Run this MATLAB function by typing

```
[Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]
        =mpcgain(Ap,Bp,Cp,Nc,Np);
```

- Comparing the results with the answers from the previous example. If it is identical to what was presented there, then your program is correct.
- Varying the prediction horizon and control horizon, observe the changes in these matrices.
- Calculate ΔU by assuming the information of initial condition on x and r. The inverse of matrix M is calculated in MATLAB as inv(M).

Tutorial Exercise I

Consider the state-space model of the mass-spring damped system

$$\begin{aligned} \dot{x}_m(t) &= \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} x_m(t) + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(t) \end{aligned} \tag{11}$$

where we assume $m = 3$, $k = 1$ and $c = 0.5$.

The objective of this tutorial is to find the optimal solution of ΔU vector that will minimize the error function

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U$$

Tutorial Exercise II

- Assume that the prediction horizon $N_p = 20$, and control horizon $N_c = 6$.
- Compute the $\Phi^T\Phi$, $\Phi^T F$ and $\Phi^T R_s$ matrices.
- Assuming that at sample time $k = 8$, $x(8) = [0.7\ 1\ 0.6]^T$, weight $r_w = 1$, and set-point signal $r = 1$, compute the control vector ΔU for this system. Present graphically ΔU and output y for $k = 8, 9, 10, \ldots, N_p$.
- Investigate how the weight r_w affects the response speed by examining the cases where $r_w = 0$ and $r_w = 10$. Compare ΔU and y for the three cases.

Lecture 4: Receding Horizon Control and Closed-loop Systems

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Receding Horizon Control:Overview

- Although the optimal parameter vector ΔU contains the controls $\Delta u(k_i)$, $\Delta u(k_i+1)$, $\Delta u(k_i+2)$, ..., $\Delta u(k_i+N_c-1)$, with the receding horizon control principle, we only implement the first sample of this sequence, i.e. $\Delta u(k_i)$, while ignoring the rest of the sequence.
- When the next sample period arrives, the more recent measurement is taken to form the state vector $x(k_i+1)$ for calculation of the new sequence of control signal.
- This procedure is repeated in real-time to give the receding horizon control law, which is a feedback control law.

Example

We illustrate this procedure by continuing the example in the previous lecture, where a first order system with the state space description

$$x_m(k+1) = 0.8x_m(k) + 0.1u(k)$$

is used in the computation. We will consider the case $r_w = 0$. The initial conditions are $x(10) = [0.1\ 0.2]^T$ and $u(9) = 0$.

Solution I

- At sample time $k_i = 10$, the optimal control was previously computed as $\Delta u(10) = 7.2$. Assuming that $u(9) = 0$, then the control signal to the plant is $u(10) = u(9) + \Delta u(10) = 7.2$ and with $x_m(10) = y(10) = 0.2$, we calculate the next simulated plant state variable

(1) $$x_m(11) = 0.8x_m(10) + 0.1u(10) = 0.88$$

- At $k_i = 11$, the new plant information is $\Delta x_m(11) = 0.88 - 0.2 = 0.68$ and $y(11) = 0.88$, which forms $x(11) = \begin{bmatrix} 0.68 & 0.88 \end{bmatrix}^T$. Then we obtain

$$\Delta U = (\Phi^T\Phi)^{-1}(\Phi^T R_s - \Phi^T F x(11)) = \begin{bmatrix} -4.24 & -0.96 & 0.0000 & 0.0000 \end{bmatrix}^T$$

This leads to the optimal control $u(11) = u(10) + \Delta u(11) = 2.96$. This new control is implemented to obtain

(2) $$x_m(12) = 0.8x_m(11) + 0.1u(11) = 1$$

Solution II

- At $k_i = 12$, the new plant information is $\Delta x_m(12) = 1 - 0.88 = 0.12$ and $y(12) = 1$, which forms $x(12) = \begin{bmatrix} 0.12 & 1 \end{bmatrix}$. We obtain

$$\Delta U = (\Phi^T \Phi)^{-1}(\Phi^T R_s - \Phi^T F x(12)) = \begin{bmatrix} -0.96 & 0.000 & 0.0000 & 0.0000 \end{bmatrix}^T$$

This leads to the control at $k_i = 12$ as $u(12) = u(11) - 0.96 = 2$. By implementing this control, we obtain the next plant output as

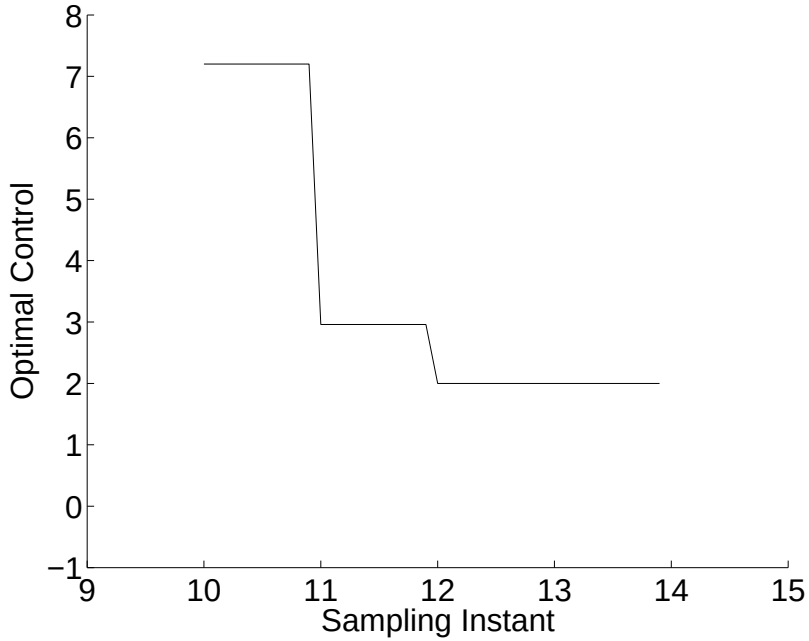
$$x_m(13) = a x_m(12) + b u(12) = 1 \tag{3}$$

- The new plant information is $\Delta x_m(13) = 1 - 1 = 0$ and $y(13) = 1$. From this information, we obtain

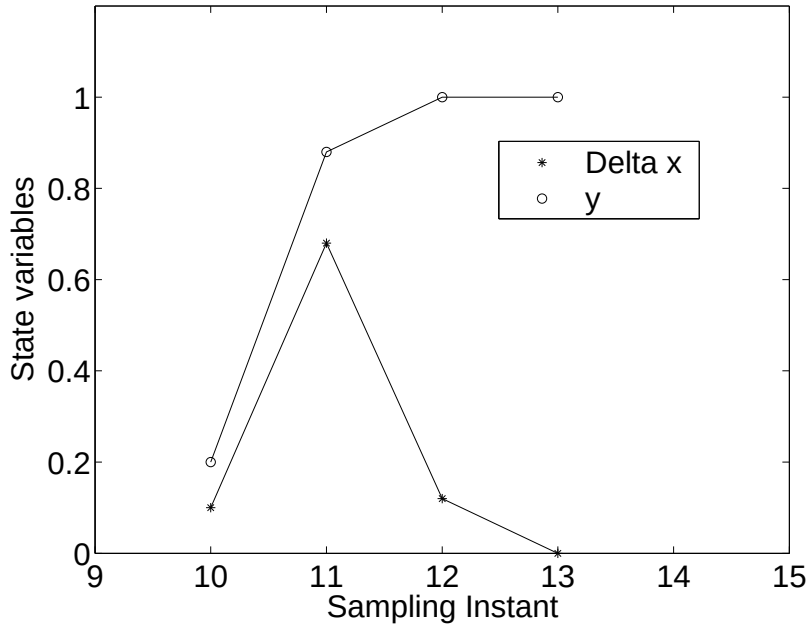
$$\Delta U = (\Phi^T \Phi)^{-1}(\Phi^T R_s - \Phi^T F x(13)) = \begin{bmatrix} 0.000 & 0.000 & 0.0000 & 0.0000 \end{bmatrix}^T$$

Solution III

Figure 1 shows the trajectories of the state variable $\Delta x_m(.)$ and $y(.)$, as well as the control signal that was used to regulate the output.



(a) Optimal control



(b) State variable

Figure 1: Receding horizon control

Feedback Gains

- If we examine this example carefully, then we find that at a given time k_i, the optimal parameter vector ΔU is solved using

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1}(\Phi^T R_s - \Phi^T F x(k_i))$$

where $(\Phi^T \Phi + \bar{R})^{-1}\Phi^T \bar{R}_s$ corresponds to the setpoint change while $-(\Phi^T \Phi + \bar{R})^{-1}\Phi^T F$ corresponds to the state feedback control.

- Both depend on the system parameters, hence are constant matrices for a time- invariant system. Because of the receding horizon control principle, we only take the first element of ΔU at time k_i as the incremental control, thus

$$\begin{aligned} \Delta u(k_i) &= \overbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}}^{N_c}(\Phi^T \Phi + \bar{R})^{-1}(\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_x x(k_i) \end{aligned} \tag{4}$$

where K_y is the first element of

$$(\Phi^T \Phi + \bar{R})^{-1}\Phi^T \bar{R}_s$$

and K_x is the first row of

$$(\Phi^T \Phi + \bar{R})^{-1}\Phi^T F$$

Closed-loop Control System

- Equation (4) is in a standard form of linear time-invariant state feedback control. The state feedback control gain vector is K_x.
- Therefore, with the augmented design model

$$x(k+1) = Ax(k) + B\Delta u(k)$$

the closed-loop system is obtained by substituting (4) into the augmented system equation; changing index k_i to k, leading to the closed-loop equation

$$x(k+1) = Ax(k) - BK_x x(k) + BK_y r(k) \quad (5)$$

$$= (A - BK_x)x(k) + BK_y r(k) \quad (6)$$

Closed-loop Eigenvalues

The closed-loop eigenvalues can be evaluated through the closed-loop characteristic equation:

$$\det[\lambda I - (A - BK_x)] = 0$$

We can use the closed-loop eigenvalues to determine the closed-loop stability and closed-loop response speed of the predictive control system.

Closed-loop Feedback Diagram

Because of the special structures of the matrices C and A, the last column of F is identical to $\bar{R}_s$, which is $[1\ 1\ \ldots,\ 1]^T$, therefore K_y is identical to the last element of K_x. Furthermore, noting that the state variable vector $x(k_i) = [\Delta x_m(k)^T\ y(k)]^T$.

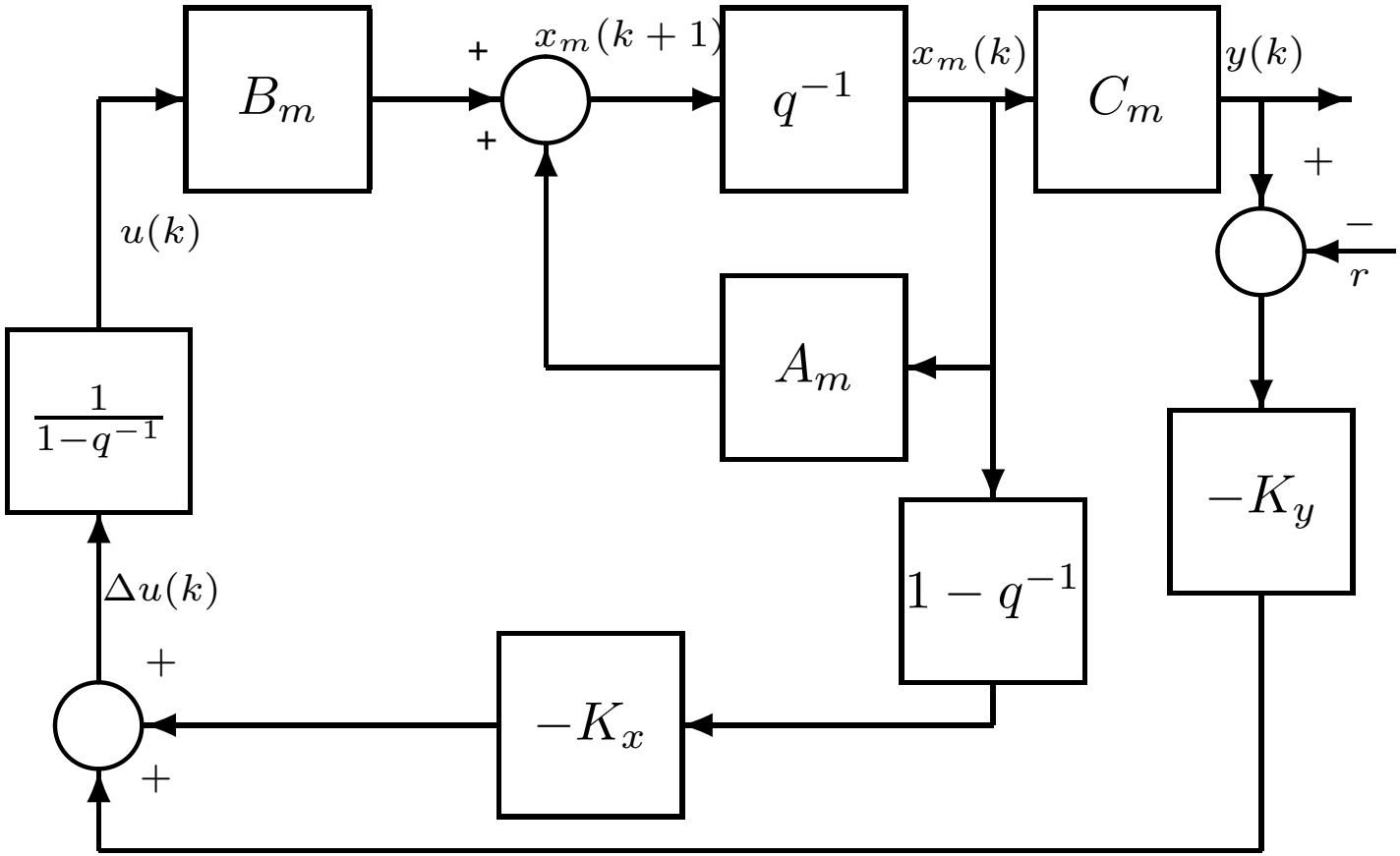


Figure 2: Block diagram of discrete-time predictive control system

Example

This example will examine the closed-loop feedback gain matrices generated from the previous example and the eigenvalues of the closed-loop system with weight $r_w = 0$ and $r_w = 10$.

Solution. When the weight $r_w = 0$, we have

$$K_y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}) = 10$$

$$K_x = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T F) = \begin{bmatrix} 8 & 10 \end{bmatrix}$$

Hence the eigenvalues of the closed-loop system are calculated by evaluating the eigenvalues of the closed-loop matrix $A - BK_x$, where from the previous example

$$A = \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}; \; B = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

They are $\lambda_1 = -6.409 \times 10^{-7}$ and $\lambda_2 = 6.409 \times 10^{-7}$, approximately on the origin of the complex plane.

Example-continues

When the weight $r_w = 10$, we have

$$K_y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}) = 0.2453$$

$$K_x = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T F) = \begin{bmatrix} 0.6939 & 0.2453 \end{bmatrix}$$

With the value of K_x, the eigenvalues of the closed-loop system are $\lambda_{1,2} = 0.8530 \pm j0.0542$, indicating that the dynamics of the closed-loop system have a much slower response than the case when $r_w = 0$.

MATLAB Tutorial:Implementation of Predictive Control

The objective of this tutorial is to learn how to implement a predictive control system using receding horizon control. The plant state-space model is given by

$$
\begin{aligned}
x_m(k+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_m(k) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(k) \\
y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(k)
\end{aligned} \tag{7}
$$

A set-point signal $r = 1$ is used in the simulation-implementation study.

Tutorial Solution-I

Step by Step

- Create a new file called reced.m
- The first step is to define the plant, enter the values of prediction horizon and control horizon. The plant is the double integrator system (7). Control horizon is selected to be $N_c = 4$ and prediction horizon is $N_p = 20$. Enter the following program into the file:

```
Ap=[1 1;0 1]; Bp=[0.5;1]; Cp=[1 0]; Dp=0; Np=20; Nc=4;
```

- The program calls the function mpcgain.m to generate the necessary gain matrices and specifies the initial conditions for implementation of receding horizon control. Initial state variable for the plant is xm=0; and the initial state feedback variable is Xf=0; set-point signal is specified and number of simulation point is specified as 100.

Tutorial Solution -II

- Continue entering the following program into the file:

```
[Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]=
 mpcgain(Ap,Bp,Cp,Nc,Np);
[n,n_in]=size(B_e); xm=[0;0];
 Xf=zeros(n,1);
  N_sim=100;
r=ones(N_sim,1);
u=0;  % u(k-1) =0
y=0;
```

- From receding horizon control, at sample time kk, the ΔU vector is calculated using the setpoint signal $r(kk)$ and the state vector Xf. Then $\Delta u(kk)$ is taken as the first element of ΔU; and $u(kk) = u(kk-1) + \Delta u(k)$. Weight factor r_w is selected as 0.1.

Tutorial Solution -III

- Continue entering the following program into the file:

```
for kk=1:N_sim;
DeltaU=inv(Phi_Phi+0.1*eye(Nc,Nc))*(Phi_R*r(kk)-Phi_F*Xf
deltau=DeltaU(1,1); u=u+deltau; u1(kk)=u; y1(kk)=y;
```

- The plant state and output are simulated using the control signal generated; the state variable used in the feedback mechanism is updated as Xf.
- Continue entering the following program into the file:

```
    xm_old=xm;
     xm=Ap*xm+Bp*u;
     y=Cp*xm;
     Xf=[xm-xm_old;y];
end
```

Tutorial Solution -IV

- The input and output signals are plotted against samples.
- Continue entering the following program into the file:

```
k=0:(N_sim-1);
figure subplot(211) plot(k,y1)
xlabel('Sampling Instant') legend('Output')
 subplot(212) plot(k,u1)
 xlabel('Sampling Instant') legend('Control')
```

- Save the program in the same directory as the one that contains the function. Run the program.
- Change the weight r_w in the design to 2 and observe that the closed-loop response speed is slower.

Tutorial Exercise

Predictive control of a one degree of freedom mass-spring damped system described by the state-space model

$$\begin{aligned} \dot{x}_m(t) &= \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} x_m(t) + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(t) \end{aligned} \tag{8}$$

where $y(t)$ is the position of the mass and $u(t)$ is external force, and we assume $m = 6$, $k = 1$ and $c = 0.5$. The sampling interval is selected as $\Delta t = 0.5$. The design parameters are

- The control horizon is $N_c = 8$, and prediction horizon is $N_p = 38$.
- The set-point signal r is a unit step signal. Namely, we want to the position of the mass follows the set-point signal change from 0 to 1 unit.
- Simulate the closed-loop response of the position of mass and the external force for 500 samples.
- Present your results graphically.

Lecture 5: Observer Design and Implementation

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Motivation

- In the design of model predictive controllers, we assumed that the information $x(k_i)$ is available at the time k_i.
- This assumes that all the state variables are measurable.
- In reality, with most applications, not all state variables are measured (or available). Some of them may be impossible to measure.
- The alternative is to construct the state variable $x(k)$ from the process measurement.
- The 'soft' instrument used to construct unknown state variables based on process measurement, in a control engineering context, is called an observer.
- The concept of an observer has been widely used in science and engineering fields, which is often called a 'soft' sensor.

Observer in Open-loop Formulation

- It would not be hard to imagine that an observer is constructed based on a mathematical model of the plant.
- We assume the plant state space model:

$$\text{(1)} \qquad x_m(k+1) \quad = \quad A_m x_m(k) + B_m u(k)$$

- Then we can use this model to calculate the state variable $\hat{x}_m(k)$, $k = 1, 2, \ldots,$ with an initial state condition $\hat{x}_m(0)$ and input signal $u(k)$ as

$$\text{(2)} \qquad \hat{x}_m(k+1) \quad = \quad A_m \hat{x}_m(k) + B_m u(k)$$

- $\hat{x}_m(k)$ is the estimated state using the input data $u(k)$. This is an open-loop formulation of an observer.
- This approach in fact would work after some transient time, if the plant model is stable and our guess of the initial condition is nearly correct.

Problems with Observers in Open-loop Operation-I

- The error $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$ satisfies the difference equation:

$$\begin{aligned} \tilde{x}_m(k+1) &= A_m(x_m(k) - \hat{x}_m(k)) \\ &= A_m\tilde{x}_m(k) \end{aligned} \tag{3}$$

- For a given initial error state $\tilde{x}_m(0) \neq 0$, we have

$$\tilde{x}_m(k) = A_m^k\tilde{x}_m(0) \tag{4}$$

- If A_m has all eigenvalues inside the unit circle, then the error system (4) is stable and $||\tilde{x}_m(k)|| \rightarrow 0$ as $k \rightarrow \infty$, which means that the estimated state variable $\hat{x}_m(k)$ converges to $x_m(k)$.
- If A_m has one or more eigenvalues on the unit circle, the error states $||\tilde{x}_m(k)||$ will not converge to zero.
- if A_m has one or more eigenvalues outside the unit circle, then the error system (4) is unstable and $||\tilde{x}_m(k)|| \rightarrow \infty$ as $k \rightarrow \infty$, which means that the prediction $\hat{x}_m(k)$ does not converge to $x_m(k)$.

Problems with Observers in Open-loop Operation-II

- In the case of stable plant model A_m, we have no 'control' on the convergence rate of the error $||\tilde{x}_m(k)|| \rightarrow 0$, which is dependent on the location of the plant poles.
- Namely, if the plant poles are close to the origin of the complex plane, then the error converges in a fast rate to zero; otherwise, the convergence rate could be slow.
- The question is how to improve the estimate of $x_m(k)$. The solution is to use a feedback principle where an error signal is deployed to improve the estimation.

Observer in Closed-loop Formulation

The observer is constructed using the equation:

$$\hat{x}_m(k+1) = \overbrace{A_m x_m(k) + B_m u(k)}^{model} + \overbrace{K_{ob}(y(k) - C_m \hat{x}_m(k))}^{correction\ term} \tag{5}$$

where K_{ob} is the observer gain matrix.

- In the observer form, the state variable estimate $\hat{x}_m(k+1)$ consists of two terms.
  - The first term is from the response of the original model'
  - The second term is the correction term based on the error between the measured output and the predicted output using the estimate $\hat{x}_m(k)$. This is the deployment of feedback principle.

Observer Design

- To choose the observer gain K_{ob}, we examine the closed-loop error equation. By substituting $y(k) = C_m x_m(k)$ into (5), with the definition of error state $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$, we obtain that

$$\begin{aligned} \tilde{x}_m(k+1) &= A_m \tilde{x}_m(k) - K_{ob} C_m \tilde{x}_m(k) \\ &= (A_m - K_{ob} C_m) \tilde{x}_m(k) \end{aligned} \tag{6}$$

  Now, with given initial error $\tilde{x}_m(0)$, we have

$$\tilde{x}_m(k) = (A_m - K_{ob} C_m)^k \tilde{x}_m(0) \tag{7}$$

- Comparing the observer error response given by (7) with the open-loop prediction (4), it is apparent that the observer gain K_{ob} can be used to manipulate the convergence rate of the error.
- If there is only single output, a commonly used approach is to place the closed-loop eigenvalues of the error system matrix $A_m - K_{ob} C_m$ at a desired location of the complex plane.

Example I

The linearized equation of motion of a simple pendulum is

$$\frac{d^2\theta}{dt^2} + \omega^2\theta = u \tag{8}$$

where θ is the angle of the pendulum. Design an observer that reconstructs the angle of the pendulum given measurements of $\frac{d\theta}{dt}$. Assume $\omega = 2$ rad/sec and sampling interval $\Delta t = 0.1$ (sec). The desired observer poles are chosen to be 0.1 and 0.2. Compare the open-loop estimation with the observer based estimation.

Example II-Solution

Let $x_1(t) = \theta$ and $x_2(t) = \dot{\theta}$, using the motion equation (8), the corresponding state space model is

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{9}$$

With $\omega = 2$ rad/sec and sampling interval $\Delta t = 0.1$ (sec), the corresponding discrete-time state space model is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9801 & 0.0993 \\ -0.3973 & 0.9801 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0050 \\ 0.09930 \end{bmatrix} u(k) \tag{10}$$

$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{11}$$

Example III-Solution

Simulation of Open-loop formulation

- We investigate what happens if the pendulum model alone is used for the prediction of the angle $\theta(x_1)$.
- Assume that the input signal $u(k) = 0$ and the initial conditions of the state variables are $\theta(0) = x_1(0) = 1$ and $\dot{\theta}(0) = x_2(0) = 0$. The trajectories of movement for both θ and $\dot{\theta}$ are shown in Figure 1. Both θ and $\dot{\theta}$ are sinusoidal signals.
- Now suppose that we take a guess at the initial conditions of the state variables as $\hat{x}_1(0) = 0.3$ and $\hat{x}_2(0) = 0$. By using the state space model (2), the estimates of θ and $\dot{\theta}$ are calculated and shown in comparison to the true trajectories (see Figure 1).

Example IV- Solution

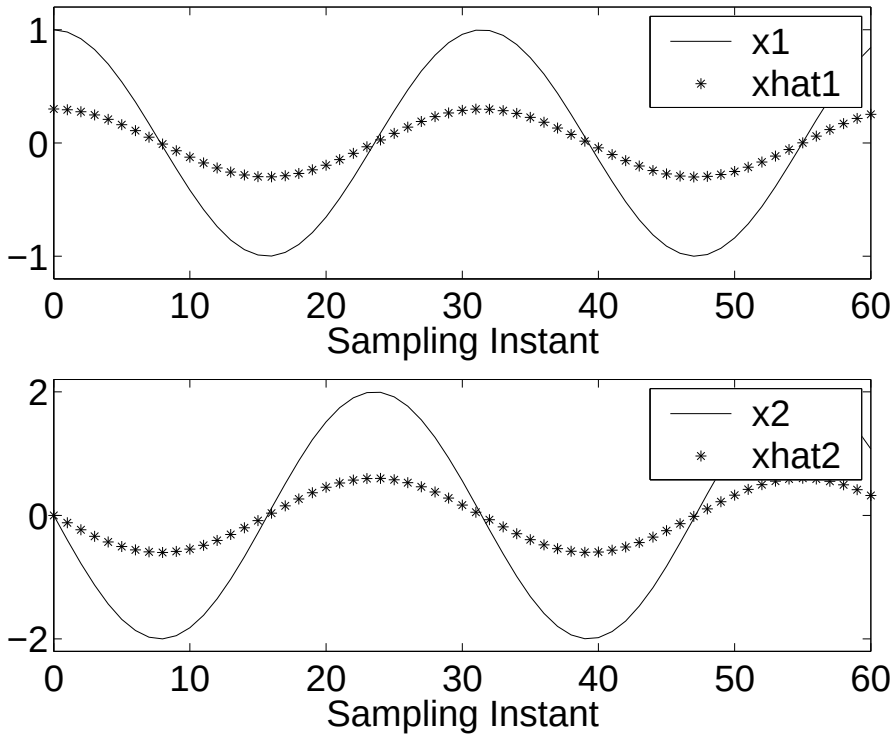


Figure 1: Estimation using Open-loop Formulation

It is seen that the estimate of θ, denoted $\hat{x}_1$, is not close to the true θ (see the top plot of Figure 1). This study demonstrated that using the model alone is not sufficient to predict the angle of the pendulum.

Example V- Solution

Simulation of observer (closed-loop formulation)

- Let us design and implement an observer to predict the angle of the pendulum.
- Assume that the observer gain $K_{ob} = [j_1 \;\; j_2]^T$. The closed loop characteristic polynomial for the observer is

$$\begin{aligned} & \det(\lambda I - \begin{bmatrix} 0.9801 & 0.0993 - j_1 \\ -0.3973 & 0.9801 - j_2 \end{bmatrix}) \\ = \quad & (\lambda - 0.9801)(\lambda + j_2 - 0.9801) - 0.3973 \times (j_1 - 0.0993) \end{aligned}$$

which is made to be equal to the desired closed-loop characteristic polynomial $(\lambda - 0.1)(\lambda - 0.2)$.

Example VI -Solution

Namely,

$$(\lambda - 0.9801)(\lambda + j_2 - 0.9801) - 0.3973 \times (j_1 - 0.0993) = (\lambda - 0.1)(\lambda - 0.2)$$

Solution of the polynomial equation gives us the observer gain as $j_1 = -1.6284$ and $j_2 = 1.6601$.

The essence of this design is to put the closed-loop observer poles at the desired locations. This is called pole-assignment observer design.

Example VII -Solution

The estimation of angle is carried out using the observer equation:

$$\begin{bmatrix} \hat{x}_1(k+1) \\ \hat{x}_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9801 & 0.0993 \\ -0.3973 & 0.9801 \end{bmatrix} \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix} + K_{ob}(x_2(k) - \hat{x}_2(k))$$

with initial condition $\hat{x}_1(0) = 0.3$ and $\hat{x}_2(0) = 0$. Figure 2 shows that the estimated angle converges to the true angle in about three steps.

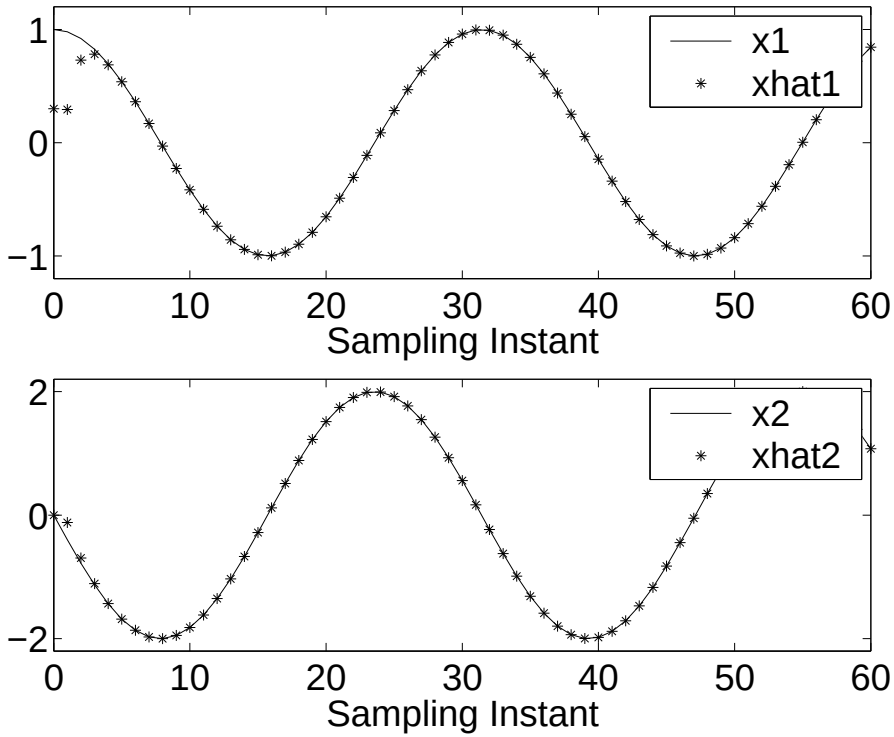


Figure 2: Estimation using observer

MATLAB Tutorial:Observer Design and Implementation

The objective of this tutorial is to learn how to design and implement an observer. We will use the pendulum model in the tutorial. Here the continuous-time model is given by

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{12}$$

where $\omega = 2$ rad/sec and sampling interval $\Delta t = 0.1$.

Tutorial Solution-I

Step by Step

- Create a new file called obser.m
- The first step is to define the continuous-time plant, and discretize it to obtain the discrete-time model. Enter the following program into the file:

```
w=2;
Ap=[0 1;-w*w 0];
Bp=[0;1];
Cp=[0 1];
 Dp=0;
 h=0.1;
[Am,Bm,Cm,Dm]=c2dm(Ap,Bp,Cp,Dp,h);
```

- We design an observer using pole-assignment technique in order to find the observer gain K_{ob}. We specify the desired closed-loop poles at 0.1 and 0.2. Continue entering the following program into the file:

```
pole=[0.1 0.2];
Kob=place(Am',Cm', pole)'
```

Tutorial Solution -II

- Simulate the estimated state variables using the observer designed. First we simulate the plant output $y(k)$ with zero initial conditions. Continue entering the following program into the file:

```
u=0; x=[1;0];
x1(1)=1;
x2(1)=0;
y(1)=x2(1);
for k=1:60
    x=Am*x;
    x1(k+1)=x(1,1);
    x2(k+1)=x(2,1);
    y(k+1)=x2(k+1);
end
```

- Specify the initial conditions for the estimated state.

```
xhat=[0.3;0.0]; xhat2(1)=0;
 xhat1(1)=0.3;
```

Tutorial Solution -III

- Simulate the response of observer. Continue entering the following program into the file:

```
for k=1:60 xhat=Am*xhat+Kob*(y(k)-xhat2(k));
    xhat1(k+1)=xhat(1,1);
    xhat2(k+1)=xhat(2,1);
end
 kk=0:60;
figure subplot(211)
plot(kk,x1,kk,xhat1,'.')
set(gca,'FontSize',20,'FontName','helvetica');
 xlabel('Sampling Instant')
 legend('x1','xhat1')
 subplot(212)
plot(kk,x2,kk,xhat2,'.')
set(gca,'FontSize',20,'FontName','helvetica');

xlabel('Sampling Instant') legend('x2','xhat2')
```

Tutorial Exercise

A DC motor can be described by a second-order model with an integrator and one time constant (see Figure 3). The input is the voltage to the motor and the output is the shaft position. The time constant is due to the mechanical parts of the system. The dynamics due to the electrical parts are neglected because they have small time constants.

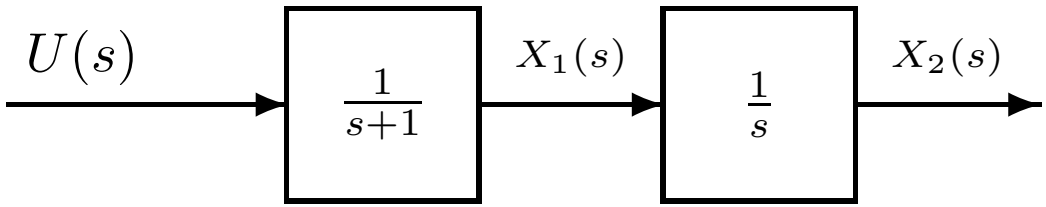


Figure 3: Motor model

Tutorial Exercise-Continue

- By choosing x_1 as the angular velocity and x_2 as the angular position of the motor shaft, we obtain the continuous-time state space equation

(13)
$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t)$$

- Suppose that we take the measurement of position, leading to

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

- Design an observer to estimate model angular speed by using measurement of position. The design specifications are: sampling interval $\Delta t = 0.1$ and the closed-loop observer poles are 0.2 and 0.3.
- The simulation of the plant conditions is set as $u = 0$, $x_1(0) = 1$ (an initial speed) and $x_2(0) = 0$. The initial condition of the estimated states is set as $\hat{x}_1(0) = 0$ and $\hat{x}_2(0) = 0$.
- Compare the estimated states with the true plant states.

Lecture 6: State Estimate Predictive Control

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Basic Ideas

- In the implementation of predictive control, an observer is used for the cases where the state variable $x(k_i)$ at time k_i is not measurable.
- Essentially, the state variable $x(k_i)$ is estimated via an observer of the form:

(1) $$\hat{x}(k_i+1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i))$$

- Note that in the implementation of predictive control using an observer, the control signal is $\Delta u(.)$ and the matrices (A, B, C) come from the augmented model used for the predictive control design.

Optimal Solution of Control Parameter Vector

- With the information of $\hat{x}(k_i)$ replacing $x(k_i)$, the predictive control law is then modified to find ΔU by minimizing

$$\begin{aligned} J &= (R_s - F\hat{x}(k_i))^T(\bar{R}_s r(k_i) - F\hat{x}(k_i)) - 2\Delta U^T \Phi^T (R_s - F\hat{x}(k_i)) \\ &+ \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U \end{aligned} \tag{2}$$

  where $\bar{R}_s$, F, Φ , $\bar{R}$ and ΔU were the matrices and vectors, defined in the previous sections.

- The minimization of the objective function leads to the optimal solution ΔU obtained as

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - F\hat{x}(k_i)) \tag{3}$$

- Note that ΔU is a function of the estimated state $\hat{x}(k_i)$.

State Estimate Receding Horizon Control

- By only applying the first element of the control parameter vector ΔU at time k_i, we obtain the optimal solution of the control signal $\Delta u(k_i)$ at time k_i:

(4) $$\Delta u(k_i) = K_y r(k_i) - K_{mpc}\hat{x}(k_i)$$

- K_y is the first element of

$$(\Phi^T\Phi + \bar{R})^{-1}\Phi^T\bar{R}_s$$

and K_{mpc} is the first row of

$$(\Phi^T\Phi + \bar{R})^{-1}\Phi^T F$$

Closed-loop State Estimate Feedback Structure

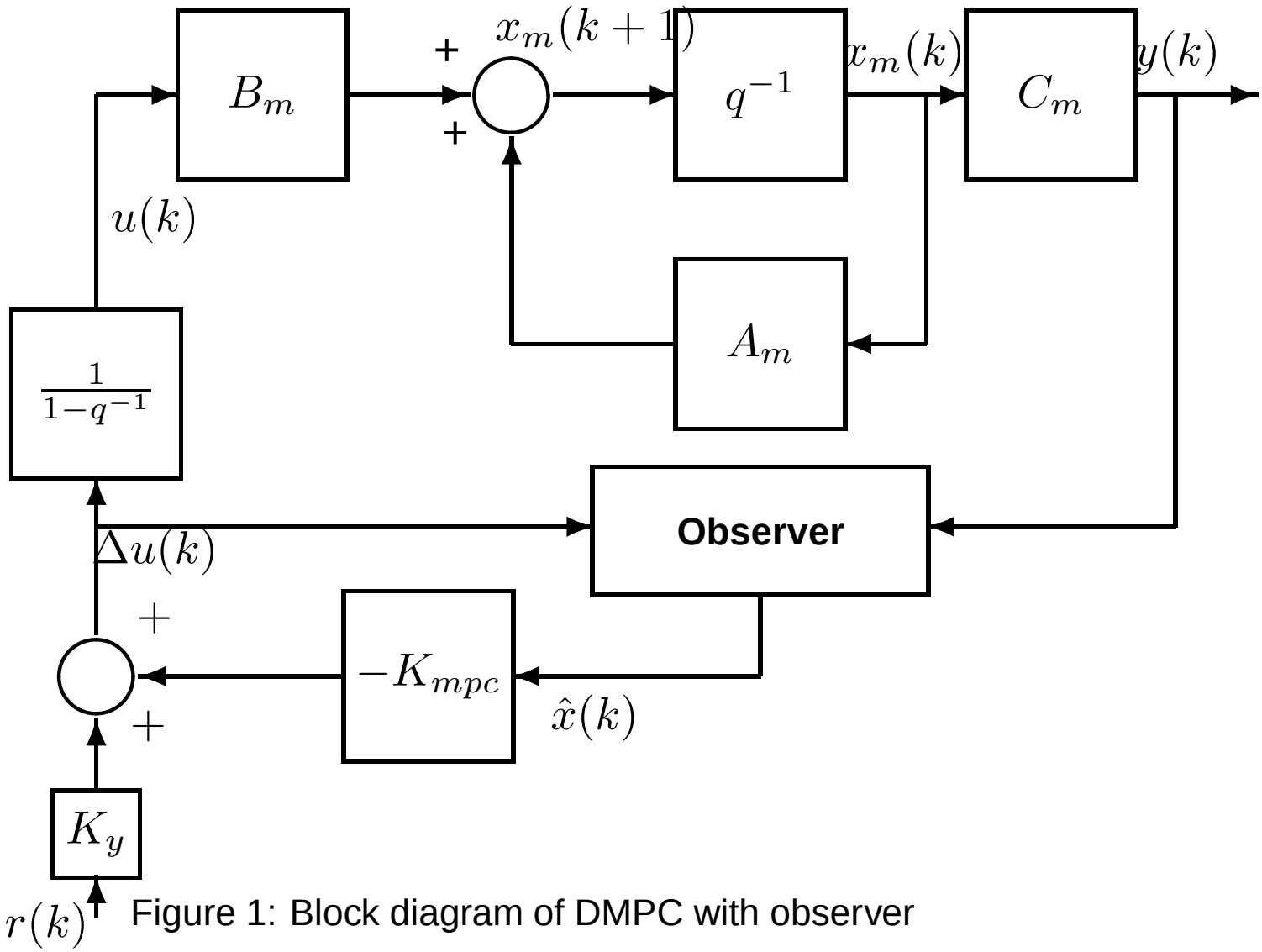


Figure 1: Block diagram of DMPC with observer

Closed-loop Equation

- We form the closed-loop state space equation as below

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ &= Ax(k) + BK_y r(k) - BK_x \hat{x}(k) \end{aligned} \tag{5}$$

where we substituted $\Delta u(k)$ with (4).

- Note that the closed-loop observer error equation is

$$\tilde{x}(k+1) = (A - K_{ob}C)\tilde{x}(k) \tag{6}$$

where $\tilde{x}(k) = x(k) - \hat{x}(k)$.

- Replacing $\hat{x}(k)$ by $x(k) - \tilde{x}(k)$, (5) is rewritten as

$$x(k+1) = (A - BK_x)x(k) - BK_x\tilde{x}(k) + BK_y r(k) \tag{7}$$

Combination of (6) with (7) leads to

$$\begin{bmatrix} \tilde{x}(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} A - K_{ob}C & o_n \\ -BK_x & A - BK_x \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ x(k) \end{bmatrix} + \begin{bmatrix} o_1 \\ BK_y \end{bmatrix} r(k) \tag{8}$$

where o_n is a $n \times n$ zero matrix and o_1 is a $n \times m$ zero matrix.

Closed-loop Characteristic Equation

- The characteristic equation of the closed- loop state space system is determined by

$$\det\left[\lambda I - \left[\begin{array}{cc} A - K_{ob}C & o_n \\ -BK_x & A - BK_x \end{array}\right]\right] = 0$$

which is equivalent to

$$\det(\lambda I - (A - K_{ob}C))\det(\lambda I - (A - BK_x)) = 0$$

because the system matrix in (8) has a lower block triangular structure.

- This effectively means that the closed-loop model predictive control system with state estimate has two independent characteristic equations

$$\det(\lambda I - (A - K_{ob}C)) = 0 \quad (9)$$

$$\det(\lambda I - (A - BK_x)) = 0 \quad (10)$$

Closed-loop Eigenvalues of State Estimate Predictive Control System

- The closed-loop eigenvalues are the solutions of the characteristic equations.
- Two independent characteristic equations indicate that the set of eigenvalues of the closed-loop system with observer consists of predictive control-loop eigenvalues and observer-loop eigenvalues.
- This means that the design of the predictive control law and the observer can be carried out independently (or separately), yet when they are put together this way, the eigenvalues remain unchanged.

Example

The augmented model for a double integrated plant is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{11}$$

where $A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$; $B = \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix}$; $C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$

- We will design a state estimate predictive control system and simulate the closed-loop response for a set-point change.
- The design specifications are $N_c = 5$, $N_p = 30$, the weight on the control signal is $r_w = 10$.
- The observer is designed using pole- assignment method where the closed-loop observer poles are $0.01,\ \ 0.0105,\ \ 0.011$, corresponding to a fast dynamic response speed from the observer.

Solution–I

- The open-loop plant has three eigenvalues at 1 where two of these were from the double integrated plant and one from the predictive controller structure.
- We use the MATLAB command 'place' and write a few lines of MATLAB program to produce the observer gain vector K_{ob}.

```
Pole=[0.01 0.0105 0.011];
 K_ob=place(A',C',Pole)';
```

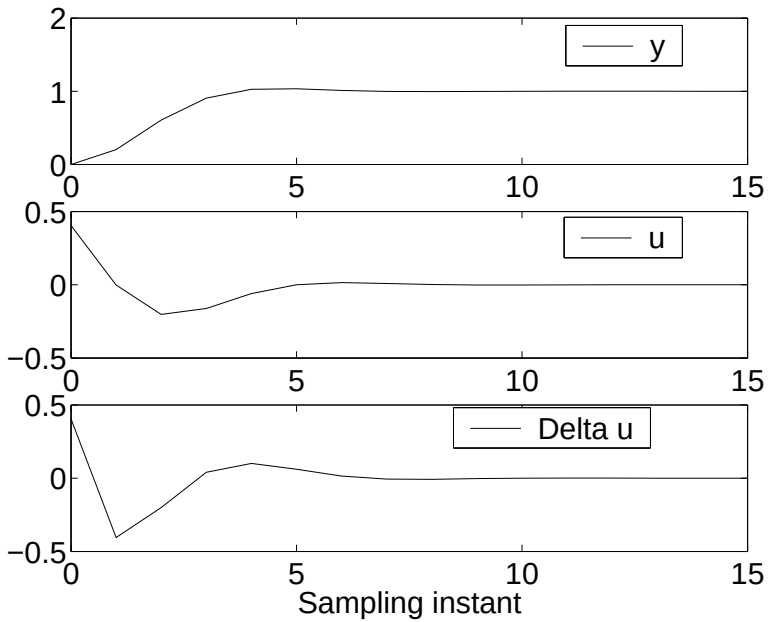
  where $A^{'}$, $C^{'}$ are the transposed matrices A and C. The transposes are needed because the MATLAB program 'place' was written for controller design.
- By using this program, we have used the dual relationship between controller and observer. The resultant observer gain is

$$K_{ob} = [1.9685\ 0.9688\ 2.9685]^T$$

Solution – II

- With this set of performance parameters specified, the state feedback control gain is $K_x = [0.8984\ 1.3521\ 0.4039]$ which effectively yields a set of closed-loop eigenvalues at $0.3172 \pm j0.4089$ and 0.3624.

Figure 2 shows that the closed-loop response for a step set-point change. It is seen that the closed-loop output response follows the set-point change, and the control signal converges to zero as the plant has integrators.



MATLAB Tutorial: State Estimate Predictive Control

- The objective of this tutorial is to learn how to design and implement a state estimate predictive control system. We will use the double integrator model in the tutorial.
- The design and implementation is based on a combination of two previous tutorials.

Tutorial Solution–I

Step by Step

- Create a new file called predcont.m, and copy the contents of the m-file built for receding horizon control 'reced.m' into this file.
- Make the following changes to the design parameters:

```
Np=30;
Nc=4;
```

- Insert observer design after calling the function 'mpcgain'. Entering the following program after that line:

```
Pole=[0.01 0.0105 0.011];
 K_ob=place(A_e',C_e',Pole)';
```

Tutorial Solution –II

- In the original program 'reced.m', 'Xf' is the state feedback variable and is simulated from plant state measurement. Here 'Xf' needs to be constructed using observer.
- Delete the line:

  ```
  Xf=[xm-xm_old;y];
  ```

- Find the line

  ```
  u=u+deltau;
  ```

  and add the computation of estimated states below it:

  ```
  Xf=A_e*Xf+Kob*(y-C_e*Xf)+B_e*deltau;
  ```

- We also need to change the value of r_w to 10 in this tutorial. Modify the solution of ΔU to reflect this change:

  ```
  DeltaU=inv(Phi_Phi+10*eye(Nc,Nc))*(Phi_R*r(kk)-Phi_F*Xf)
  ```

- Run this program and compare the results with the example.

Tutorial Exercise

Design and implement predictive control of motor where the measurement of angular position is used while the angular speed is estimated using an observer.

- The model of the motor is:

(12)
$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

- Design an observer to estimate angular speed by using measurement of position. The design specifications are: sampling interval $\Delta t = 0.1$ and the closed-loop observer poles are 0.2 and 0.3.
- The design parameters for the predictive control are $N_c = 4$, $N_p = 20$, $r_w = 3$.
- Simulate the closed-loop predictive control system with a unit step set-point change.

Lecture 7: Introduction to Constrained Control

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Constraints on $\Delta u(k)$–I

Suppose that for a single-input system, the upper limit is Δu^{max} and the lower limit is Δu^{min}.

(1) $$\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max}.$$

Note that we use *less than plus equal to*, where the equality will play the critical role in the solution of the constrained control problem.

- The rate of change constraints can be used to impose directional movement constraints on the control variables; for instance, if $u(k)$ can only increase, not decrease, then possibly, we select

$$0 \leq \Delta u(k) \leq \Delta u^{max}.$$

Constraints on $\Delta u(k)$–II

- The constraint on $\Delta u(k)$ can be used to cope with the cases where the rate of change of the control amplitude is restricted or limited in value. For example, in a control system implementation, assuming that the control variable $u(k)$ is only permitted to increase or decrease in a magnitude less 0.1 unit, then the operational constraint is

$$-0.1 \leq \Delta u(k) \leq 0.1.$$

- The constraints are also used in protection, avoiding sudden large spikes in the control signal.

Constraints on $u(k)$

- These are the most commonly encountered constraints among all constraint types. For instance, we cannot expect a valve to open more than 100 percent nor a voltage to go beyond a given range. These are the physical hard constraints on the system.
- Simply, we demand that

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

- Here, we need to pay particular attention to the fact that $u(k)$ is an incremental variable, not the actual physical variable. The actual physical control variable equals the incremental variable u plus its steady-state value u_{ss}.
- For instance, if a valve is allowed to open in the range between 15% and 80% and the valve's normal operating value is at 30%, then $u^{min} = 15\% - 30\% = -15\%$ and $u^{max} = 80\% - 30\% = 50\%$.

Output Constraints–I

- We can also specify the operating range for the plant output. For instance, supposing that the output $y(k)$ has an upper limit y^{max} and a lower limit y^{min}, then the output constraints are specified as

$$y^{min} \leq y(k) \leq y^{max}. \tag{2}$$

- Output constraints are often implemented as 'soft' constraints in the way that a slack variable $s_v > 0$ is added to the constraints, forming

$$y^{min} - s_v \leq y(k) \leq y^{max} + s_v. \tag{3}$$

Output Constraints–II

- Output constraints often cause large changes in both the control and incremental control variables when they are enforced (we term them *become active*). When that happens, the control or incremental control variables can violate their own constraints and the problem of constraint conflict occurs.
- In the situations where the constraints on the control variables are more essential to plant operation, the output constraints are often relaxed by selecting a larger slack variable s_v to resolve the conflict problem.

Constraints in an MIMO Setting–I

- Suppose that the constraints are given for the upper limits as

$$\begin{bmatrix} \Delta u_1^{max} & \Delta u_2^{max} & \dots & \Delta u_m^{max} \end{bmatrix},$$

and lower limits as

$$\begin{bmatrix} \Delta u_1^{min} & \Delta u_2^{min} & \dots & \Delta u_m^{min} \end{bmatrix}.$$

Each variable with rate of change is specified as

$$\begin{aligned} \Delta u_1^{min} \leq \quad & \Delta u_1(k) \quad \leq \Delta u_1^{max} \\ \Delta u_2^{min} \leq \quad & \Delta u_2(k) \quad \leq \Delta u_2^{max} \\ & \vdots \\ \Delta u_m^{min} \leq \quad & \Delta u_m(k) \quad \leq \Delta u_m^{max}. \end{aligned} \tag{4}$$

Constraints in an MIMO Setting–II

- Similarly, suppose that the constraints are given for the upper limit of the control signal as

$$\begin{bmatrix} u_1^{max} & u_2^{max} & \dots & u_m^{max} \end{bmatrix},$$

and lower limit as

$$\begin{bmatrix} u_1^{min} & u_2^{min} & \dots & u_m^{min} \end{bmatrix}.$$

Then, the amplitude of each control signal is required to satisfy the constraints:

$$\begin{aligned} u_1^{min} \leq \quad & u_1(k) \quad \leq u_1^{max} \\ u_2^{min} \leq \quad & u_2(k) \quad \leq u_2^{max} \\ & \vdots \\ u_m^{min} \leq \quad & u_m(k) \quad \leq u_m^{max}. \end{aligned} \tag{5}$$

Constraints as Part of the Optimal Solution

- Having formulated the constraints as part of the design requirements, the next step is to translate them into linear inequalities, and relate them to the original model predictive control problem.
- The key here is to parameterize the constrained variables using the same parameter vector ΔU as the ones used in the design of predictive control. Therefore, the constraints are expressed in a set of linear equations based on the parameter vector ΔU.
- The vector ΔU is often called the *decision variable* in optimization literature.
- Since the predictive control problem is formulated and solved in the framework of receding horizon control, the constraints are taken into consideration for each moving horizon window. This allows us to vary the constraints at the beginning of each optimization window, and also gives us the means to tackle the constrained control problem numerically.

Example–I

In the control of a DC motor, suppose that the input voltage variation is limited to 2 V and 6 V. The steady state of the control signal is at 4 V. Assuming that the control horizon is selected to be $N_c = 4$, express the constraint on $\Delta u(k_i)$ and $\Delta u(k_i + 1)$ in terms of ΔU for the first two sample times.

Solution. The parameter vector to be optimized in the predictive control system at time k_i is

$$\Delta U = \begin{bmatrix} \Delta u(k_i) & \Delta u(k_i+1) & \Delta u(k_i+2) & \Delta u(k_i+3) \end{bmatrix}^T.$$

Note that

$$\begin{aligned} u(k_i) &= u(k_i-1) + \Delta u(k_i) = u(k_i-1) + \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \Delta U \quad (6) \\ u(k_i+1) &= u(k_i) + \Delta u(k_i+1) = u(k_i-1) + \Delta u(k_i) + \Delta u(k_i+1) \\ &= u(k_i-1) + \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \Delta U. \quad (7) \end{aligned}$$

Example–II

With the limits on the control variables, by subtracting the steady-state value of the control, as $u^{min} = 2 - 4 = -2$ and $u^{max} = 6 - 4 = 2$, the constraints are expressed as

$$\text{(8)} \quad \begin{bmatrix} -2 \\ -2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(k_i - 1) + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \Delta U \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

Constraints as Inequalities

- As the optimal solutions will be obtained using quadratic programming, the constraints need to be decomposed into two parts to reflect the upper limit, and the lower limit with opposite sign. Namely, for instance, the constraints

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

will be expressed by two inequalities:

$$-\Delta U \leq -\Delta U^{min} \tag{9}$$

$$\Delta U \leq \Delta U^{max}. \tag{10}$$

- In the case of a manipulated variable constraint, we write:

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \\ \vdots \\ u(k_i+N_c-1) \end{bmatrix} = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix} u(k_i-1) + \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ I & I & 0 & \dots & 0 \\ I & I & I & \dots & 0 \\ \vdots & & & & \\ I & I & \dots & I & I \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix}. \tag{11}$$

$$-(C_1 u(k_i-1) + C_2 \Delta U) \leq -U^{min} \tag{12}$$

$$(C_1 u(k_i-1) + C_2 \Delta U) \leq U^{max}, \tag{13}$$

where U^{min} and U^{max} are column vectors with N_c elements of u^{min} and u^{max}.

The Essential Computation of MPC

- Model predictive control in the presence of hard constraints is proposed as finding the parameter vector ΔU that minimizes

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U, \tag{14}$$

- subject to the inequality constraints

$$M \Delta U \leq \gamma, \tag{15}$$

  where M is a matrix reflecting the constraints, with its number of rows equal to the number of constraints and number of columns equal to the dimension of ΔU.

- When the constraints are fully imposed, the number of constraints is equal to $4 \times m \times N_c + 2 \times q \times N_p$, where m is the number of inputs and q is the number of outputs.
- The total number of constraints is, in general, greater than the dimension of the decision variable ΔU.
- The matrix $\Phi^T \Phi + \bar{R}$ is the Hessian matrix and is assumed to be positive definite.

Lecture 8A: Predictive Control of Multi-input and Multi-out Systems

Liuping Wang

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Overview

- The MPC design methodology can be readily extended to multi-input and multi-output systems without much additional effort, because of the state-space formulation.
- We assume that the plant has m inputs, q outputs and n_1 states. We also assume that the number of outputs is less than or equal to the number of inputs (*i.e.*, $q \leq m$).
- The key reason for this assumption is that integrators are embedded into the MPC design. If the number of outputs is greater than the number of inputs, we cannot hope to control each of the measured outputs independently with zero steady-state errors.
- This means that we will choose the outputs that will have desired reference signals.

Augmented Model (i)

- We assume that the MIMO plant is described by

$$x_m(k+1) = A_m x_m(k) + B_m u(k) \quad (1)$$

$$y(k) = C_m x_m(k). \quad (2)$$

- Note that from (1), the following difference equation is also true:

$$x_m(k) = A_m x_m(k-1) + B_m u(k-1). \quad (3)$$

- By defining $\Delta x_m(k) = x_m(k) - x_m(k-1)$ and $\Delta u(k) = u(k) - u(k-1)$, then subtracting (3) from (1) leads to

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k). \quad (4)$$

- In order to relate the output $y(k)$ to the state variable $\Delta x_m(k)$, we deduce that

$$\Delta y(k+1) = C_m \Delta x_m(k+1) = C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k)$$

where $\Delta y(k+1) = y(k+1) - y(k)$.

Augmented Model (ii)

- Choosing a new state variable vector $x(k) = [\Delta x_m(k)^T \;\; y(k)^T]^T$, we have:

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & I_{q\times q} \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u($$

$$y(k) = \overbrace{\begin{bmatrix} o_m & I_{q\times q} \end{bmatrix}}^{C} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)},$$

- where $I_{q\times q}$ is the identity matrix with dimensions $q \times q$, which is the number of outputs; and o_m is a $q \times n_1$ zero matrix.

Eigenvalues of A Matrix

- The characteristic polynomial equation of the augmented model is

$$\begin{aligned} \rho(\lambda) &= \det \begin{bmatrix} \lambda I - A_m & o_m^T \\ -C_m A_m & (\lambda - 1) I_{q \times q} \end{bmatrix} \\ &= (\lambda - 1)^q \det(\lambda I - A_m) = 0 \end{aligned} \tag{6}$$

- where the determinant of a block lower triangular matrix equals the product of the determinants of the matrices on the diagonal.
- Hence, the eigenvalues of the augmented model are the union of the eigenvalues of the plant model and the q eigenvalues, $\lambda = 1$. This means that there are q integrators embedded into the augmented design model.

Prediction of State Variables

- Define the vectors Y and ΔU as

$$\Delta U = \left[\begin{array}{cccc} \Delta u(k_i)^T & \Delta u(k_i+1)^T & \ldots & \Delta u(k_i+N_c-1)^T \end{array} \right]^T$$

$$Y = \left[\begin{array}{cccc} y(k_i+1 \mid k_i)^T & y(k_i+2 \mid k_i)^T & \ldots & y(k_i+N_p \mid k_i)^T \end{array} \right]^T.$$

- The prediction of state variables is formulated as

$$\begin{aligned} x(k_i+1 \mid k_i) &= Ax(k_i) + B\Delta u(k_i) \\ x(k_i+2 \mid k_i) &= Ax(k_i+1 \mid k_i) + B\Delta u(k_i+1) \\ &= A^2 x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i+1) \\ &\vdots \\ x(k_i+N_p \mid k_i) &= A^{N_p} x(k_i) + A^{N_p-1} B\Delta u(k_i) + A^{N_p-2} B\Delta u(k_i+1) \\ &+ A^{N_p-N_c} B\Delta u(k_i+N_c-1). \end{aligned}$$

Prediction of Output

$$Y = Fx(k_i) + \Phi \Delta U, \tag{7}$$

where

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \Phi = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{bmatrix}.$$

Optimal Solution without Constraints

- The incremental optimal control within one optimization window is given by

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1}(\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)), \tag{8}$$

- Matrix $\Phi^T \Phi$ has dimension $mN_c \times mN_c$ and $\Phi^T F$ has dimension $mN_c \times n$.
- $\Phi^T \bar{R}_s$ equals the last q columns of $\Phi^T F$.
- The weight matrix $\bar{R}$ is a block matrix with m blocks and has its dimension equal to the dimension of $\Phi^T \Phi$.
- The set-point signal is $r(k_i) = [r_1(k_i) \;\; r_2(k_i) \;\; \ldots \;\; r_q(k_i)]^T$ as the q set-point signals to the multi-output system.

Receding Horizon Control

$$\Delta u(k_i) = \overbrace{\begin{bmatrix} I_m & o_m & \dots & o_m \end{bmatrix}}^{N_c}(\Phi^T\Phi+\bar{R})^{-1}(\Phi^T\bar{R}_s r(k_i)-\Phi^T F x(k_i))$$

MATLAB Tutorial (i)

The objective of this tutorial is to extend the mpcgain function written for single-input and single output system to multi-input and multi-output system.

Step by step

- Create a new file called mpcgainMIMO.m. Enter the following program into the file.

```
function [Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]
=mpcgainMIMO(Ap,Bp,Cp,Nc,Np);
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
%nl-- dimension of the state variables;
%ml-- number of outputs;
%n_in-- number of inputs.
```

MATLAB Tutorial (ii)

- Generate augmented system matrices with appropriate dimensions by considering the number of inputs and number of outputs. Enter the following program into the file.

```
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ap;
A_e(n1+1:n1+m1,1:n1)=Cp*Ap;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bp;
B_e(n1+1:n1+m1,:)=Cp*Bp;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);
%dimension of the extended state space
n=n1+m1;
```

MATLAB Tutorial (ii)

- Create F and Φ with appropriate dimensions by taking consideration of number inputs and number of outputs. Enter the following program into the file.

```
h(1:m1,:)=C_e;
F(1:m1,:)=C_e*A_e;   % row vector (1xn) one row all column
for kk=2:Np
    h((kk-1)*m1+1:kk*m1,:)=h((kk-2)*m1+1:(kk-1)*m1,:)*A_e;
    F((kk-1)*m1+1:kk*m1,:)=F((kk-2)*m1+1:(kk-1)*m1,:)*A_e;
end
v=h*B_e;
Phi=zeros(Np*m1,Nc*n_in);  %declare the dimension of Phi
Phi(:,1:n_in)=v;        % first column
for i=2:Nc
Phi(:,(i-1)*n_in+1:i*n_in)=[zeros((i-1)*m1,n_in);v(1:Np*m1+(-i+1)*m1,1:n_in)];  %
end
```

MATLAB Tutorial (iii)

- Generate the $\Phi^T\Phi$, $\Phi^T F$ and $\Phi^T \bar{R}_s$ matrices. Enter the following program into the file.

```
BarRs=F(:,n1+1:n);
Phi_Phi= Phi'*Phi;
Phi_F= Phi'*F;
Phi_R=Phi'*BarRs;
```

Lecture 7: Introduction to Constrained Control

Liuping Wang

liuping.wang@rmit.edu.au

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Constraints on $\Delta u(k)$–I

Suppose that for a single-input system, the upper limit is Δu^{max} and the lower limit is Δu^{min}.

(1) $$\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max}.$$

Note that we use *less than plus equal to*, where the equality will play the critical role in the solution of the constrained control problem.

- The rate of change constraints can be used to impose directional movement constraints on the control variables; for instance, if $u(k)$ can only increase, not decrease, then possibly, we select

$$0 \leq \Delta u(k) \leq \Delta u^{max}.$$

Constraints on $\Delta u(k)$–II

- The constraint on $\Delta u(k)$ can be used to cope with the cases where the rate of change of the control amplitude is restricted or limited in value. For example, in a control system implementation, assuming that the control variable $u(k)$ is only permitted to increase or decrease in a magnitude less 0.1 unit, then the operational constraint is

$$-0.1 \leq \Delta u(k) \leq 0.1.$$

- The constraints are also used in protection, avoiding sudden large spikes in the control signal.

Constraints on $u(k)$

- These are the most commonly encountered constraints among all constraint types. For instance, we cannot expect a valve to open more than 100 percent nor a voltage to go beyond a given range. These are the physical hard constraints on the system.
- Simply, we demand that

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

- Here, we need to pay particular attention to the fact that $u(k)$ is an incremental variable, not the actual physical variable. The actual physical control variable equals the incremental variable u plus its steady-state value u_{ss}.
- For instance, if a valve is allowed to open in the range between 15% and 80% and the valve's normal operating value is at 30%, then $u^{min} = 15\% - 30\% = -15\%$ and $u^{max} = 80\% - 30\% = 50\%$.

Output Constraints–I

- We can also specify the operating range for the plant output. For instance, supposing that the output $y(k)$ has an upper limit y^{max} and a lower limit y^{min}, then the output constraints are specified as

$$y^{min} \leq y(k) \leq y^{max}. \tag{2}$$

- Output constraints are often implemented as 'soft' constraints in the way that a slack variable $s_v > 0$ is added to the constraints, forming

$$y^{min} - s_v \leq y(k) \leq y^{max} + s_v. \tag{3}$$

Output Constraints–II

- Output constraints often cause large changes in both the control and incremental control variables when they are enforced (we term them *become active*). When that happens, the control or incremental control variables can violate their own constraints and the problem of constraint conflict occurs.
- In the situations where the constraints on the control variables are more essential to plant operation, the output constraints are often relaxed by selecting a larger slack variable s_v to resolve the conflict problem.

Constraints in an MIMO Setting–I

- Suppose that the constraints are given for the upper limits as

$$\begin{bmatrix} \Delta u_1^{max} & \Delta u_2^{max} & \dots & \Delta u_m^{max} \end{bmatrix},$$

and lower limits as

$$\begin{bmatrix} \Delta u_1^{min} & \Delta u_2^{min} & \dots & \Delta u_m^{min} \end{bmatrix}.$$

Each variable with rate of change is specified as

$$\begin{aligned} \Delta u_1^{min} \leq \quad & \Delta u_1(k) \quad \leq \Delta u_1^{max} \\ \Delta u_2^{min} \leq \quad & \Delta u_2(k) \quad \leq \Delta u_2^{max} \\ & \vdots \\ \Delta u_m^{min} \leq \quad & \Delta u_m(k) \quad \leq \Delta u_m^{max}. \end{aligned} \tag{4}$$

Constraints in an MIMO Setting–II

- Similarly, suppose that the constraints are given for the upper limit of the control signal as

$$\begin{bmatrix} u_1^{max} & u_2^{max} & \dots & u_m^{max} \end{bmatrix},$$

and lower limit as

$$\begin{bmatrix} u_1^{min} & u_2^{min} & \dots & u_m^{min} \end{bmatrix}.$$

Then, the amplitude of each control signal is required to satisfy the constraints:

$$\begin{aligned} u_1^{min} \leq \quad & u_1(k) \quad \leq u_1^{max} \\ u_2^{min} \leq \quad & u_2(k) \quad \leq u_2^{max} \\ & \vdots \\ u_m^{min} \leq \quad & u_m(k) \quad \leq u_m^{max}. \end{aligned} \tag{5}$$

Constraints as Part of the Optimal Solution

- Having formulated the constraints as part of the design requirements, the next step is to translate them into linear inequalities, and relate them to the original model predictive control problem.
- The key here is to parameterize the constrained variables using the same parameter vector ΔU as the ones used in the design of predictive control. Therefore, the constraints are expressed in a set of linear equations based on the parameter vector ΔU.
- The vector ΔU is often called the *decision variable* in optimization literature.
- Since the predictive control problem is formulated and solved in the framework of receding horizon control, the constraints are taken into consideration for each moving horizon window. This allows us to vary the constraints at the beginning of each optimization window, and also gives us the means to tackle the constrained control problem numerically.

Example–I

In the control of a DC motor, suppose that the input voltage variation is limited to 2 V and 6 V. The steady state of the control signal is at 4 V. Assuming that the control horizon is selected to be $N_c = 4$, express the constraint on $\Delta u(k_i)$ and $\Delta u(k_i + 1)$ in terms of ΔU for the first two sample times.

Solution. The parameter vector to be optimized in the predictive control system at time k_i is

$$\Delta U = \begin{bmatrix} \Delta u(k_i) & \Delta u(k_i+1) & \Delta u(k_i+2) & \Delta u(k_i+3) \end{bmatrix}^T.$$

Note that

$$\begin{aligned} u(k_i) &= u(k_i-1) + \Delta u(k_i) = u(k_i-1) + \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \Delta U \quad (6) \\ u(k_i+1) &= u(k_i) + \Delta u(k_i+1) = u(k_i-1) + \Delta u(k_i) + \Delta u(k_i+1) \\ &= u(k_i-1) + \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \Delta U. \quad (7) \end{aligned}$$

Example–II

With the limits on the control variables, by subtracting the steady-state value of the control, as $u^{min} = 2 - 4 = -2$ and $u^{max} = 6 - 4 = 2$, the constraints are expressed as

$$\text{(8)} \quad \begin{bmatrix} -2 \\ -2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(k_i - 1) + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \Delta U \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

Constraints as Inequalities

- As the optimal solutions will be obtained using quadratic programming, the constraints need to be decomposed into two parts to reflect the upper limit, and the lower limit with opposite sign. Namely, for instance, the constraints

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

will be expressed by two inequalities:

$$-\Delta U \leq -\Delta U^{min} \tag{9}$$

$$\Delta U \leq \Delta U^{max}. \tag{10}$$

- In the case of a manipulated variable constraint, we write:

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \\ \vdots \\ u(k_i+N_c-1) \end{bmatrix} = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix} u(k_i-1) + \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ I & I & 0 & \dots & 0 \\ I & I & I & \dots & 0 \\ \vdots & & & & \\ I & I & \dots & I & I \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \\ \vdots \\ \Delta u(k_i+N_c-1) \end{bmatrix}. \tag{11}$$

$$-(C_1 u(k_i-1) + C_2 \Delta U) \leq -U^{min} \tag{12}$$

$$(C_1 u(k_i-1) + C_2 \Delta U) \leq U^{max}, \tag{13}$$

where U^{min} and U^{max} are column vectors with N_c elements of u^{min} and u^{max}.

MPC With Constraints on $\Delta u(k)$

- Previously, we assumed no constraints on states or control
- What if the rate of change of the control, $\Delta u(k)$, is bounded?
- **Solution:** if $\Delta u^{\min} \leq \Delta u(k) \leq \Delta u^{\max}$, then:

$$\begin{bmatrix} -I_m \\ I_m \end{bmatrix} \Delta u(k) \leq \begin{bmatrix} -\Delta u^{\min} \\ \Delta u^{\max} \end{bmatrix}$$

- For a prediction horizon N_p, we have:

$$\begin{bmatrix} -I_m & O & \dots & O & O \\ I_m & O & \dots & O & O \\ O & -I_m & \dots & O & O \\ O & I_m & \dots & O & O \\ \vdots & & & & \vdots \\ O & O & \dots & O & -I_m \\ O & O & \dots & O & I_m \end{bmatrix} \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix}}_{\Delta U} \leq \begin{bmatrix} -\Delta u^{\min} \\ \Delta u^{\max} \\ -\Delta u^{\min} \\ \Delta u^{\max} \\ \vdots \\ -\Delta u^{\min} \\ \Delta u^{\max} \end{bmatrix}$$

MPC With Constraints on $u(k)$

- What if the control, $u(k)$, is bounded?
- **Solution:** We know that:

$$u(k) = u(k-1) + \Delta u(k) = u(k-1) + \begin{bmatrix} I_m & O & \dots & O \end{bmatrix} \Delta U(k)$$

- Similarly:

$$u(k+1) = u(k) + \Delta u(k+1) = u(k-1) + \begin{bmatrix} I_m & I_m & O & \dots & O \end{bmatrix} \Delta U(k)$$

- Or:

$$\begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N_p-1) \end{bmatrix} = \begin{bmatrix} I_m \\ I_m \\ \vdots \\ I_m \end{bmatrix} u(k-1) + \begin{bmatrix} I_m & & & \\ I_m & I_m & & \\ \vdots & \vdots & \ddots & \\ I_m & I_m & \dots & I_m \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_p-1) \end{bmatrix}$$

- Therefore, we can write:

$$U(k) = Eu(k-1) + H\Delta U(k)$$

MPC With Control Constraints

- Suppose that we have the following constraints:

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

- We can represent the above constraints as:

$$\begin{bmatrix} -U(k) \\ U(k) \end{bmatrix} \leq \begin{bmatrix} -u^{\min} \\ u^{\max} \end{bmatrix}$$

- Recall that

$$U(k) = Eu(k-1) + H\Delta U(k)$$

- Since $u(k-1)$ is know, we obtain an $Ax \leq b$-like inequality:

$$\begin{bmatrix} -H \\ H \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -u^{\min} + Eu(k-1) \\ u^{\max} - Eu(k-1) \end{bmatrix}$$

- Input-Constrained MPC—a quadratic program:

$$\begin{aligned} \text{minimize} \quad & J(\Delta U) = \frac{1}{2}(r-Y)^\top Q(r-Y) + \frac{1}{2}\Delta U^\top R \Delta U \\ \text{subject to} \quad & \begin{bmatrix} -H \\ H \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -u^{\min} + Eu(k-1) \\ u^{\max} - Eu(k-1) \end{bmatrix} \end{aligned}$$

MPC With Output Constraints

- Suppose that we require the output to be bounded:

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

- Hence, we can write:

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

- Recall that $Y(k) = Wx_a(k) + Z\Delta U(k)$
- Similar to the input-constraints, we obtain:

$$\begin{bmatrix} -Z \\ Z \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -y^{\min} + Wx_a(k) \\ y^{\max} - Wx_a(k) \end{bmatrix}$$

- Output-Constrained MPC—a quadratic program:

$$\begin{aligned} \text{minimize} \quad & J(\Delta U) = \frac{1}{2}(r - Y)^\top Q(r - Y) + \frac{1}{2}\Delta U^\top R \Delta U \\ \text{subject to} \quad & \begin{bmatrix} -Z \\ Z \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} -y^{\min} + Wx_a(k) \\ y^{\max} - Wx_a(k) \end{bmatrix} \end{aligned}$$

Constrained MPC as an Optimization Problem

- As we saw in the previous 3–4 slides, MPC problem can be written as:

$$\begin{aligned} \text{minimize} \quad & J(\Delta U) \text{ (quadratic function)} \\ \text{subject to} \quad & g(\Delta U) \leq 0 \text{ (linear constraints)} \end{aligned}$$

- Hence, we solve a constrained optimization problem (possibly convex) for each time-horizon
- Linear constraints can include constraints on: input, output, or rate of change (or their combination)
- Plethora of methods to solve such optimization problems
- How about nonlinear constraints? Can be included too!

The Essential Computation of MPC

- Model predictive control in the presence of hard constraints is proposed as finding the parameter vector ΔU that minimizes

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U, \tag{14}$$

- subject to the inequality constraints

$$M \Delta U \leq \gamma, \tag{15}$$

  where M is a matrix reflecting the constraints, with its number of rows equal to the number of constraints and number of columns equal to the dimension of ΔU.

- When the constraints are fully imposed, the number of constraints is equal to $4 \times m \times N_c + 2 \times q \times N_p$, where m is the number of inputs and q is the number of outputs.
- The total number of constraints is, in general, greater than the dimension of the decision variable ΔU.
- The matrix $\Phi^T \Phi + \bar{R}$ is the Hessian matrix and is assumed to be positive definite.

Lecture 8: Review of Quadratic Programming Algorithms

Liuping Wang

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Overview of MPC with Constraints

- Formulation of constraints
- Essence of constrained control
- Quadratic programming
- Hildreth’s quadratic programming procedure

Review of Quadratic Programming

- To be consistent with the literatures of quadratic programming, the decision variable is denoted by x.
- The objective function J and the constraints are expressed as

$$J = \frac{1}{2}x^T Ex + x^T F \quad (1)$$

$$Mx \leq \gamma, \quad (2)$$

  where E, F, M and γ are compatible matrices and vectors in the quadratic programming problem.
- Without loss of generality, E is assumed to be symmetric and positive definite.

QP for Equality Constraints:Example

Minimize

$$J = (x_1 - 2)^2 + (x_2 - 2)^2,$$

subject to

$$x_1 + x_2 = 1.$$

Solution. The global minimum, without constraint, is at

$$x_1 = 2; x_2 = 2.$$

The feasible solutions are the combinations of x_1 and x_2 that satisfy the linear equality. From the constraint, the feasible solution x_2 is expressed as

$$x_2 = 1 - x_1.$$

Substituting this into the objective function, we obtain

$$\begin{aligned} J &= (x_1 - 2)^2 + (1 - x_1 - 2)^2 \\ &= 2x_1^2 - 2x_1 + 5. \end{aligned} \tag{3}$$

In order to minimize J, the derivative $\frac{\partial J}{\partial x_1} = 4x_1 - 2 = 0$, giving the minimizing solution $x_1 = 0.5$. Now, from the constraint equation, $x_2 = 1 - x_1 = 0.5$.

Example -continues

Figure 1 illustrates the optimal solution on (x_1,x_2) plane, where the equality constraint defines the straight line and the positive definite quadratic function has its level surface as circles and the constrained minimum is located at the point of tangency between the straight line and the minimizing circle, which is $x_1 = 0.5$ and $x_2 = 0.5$.

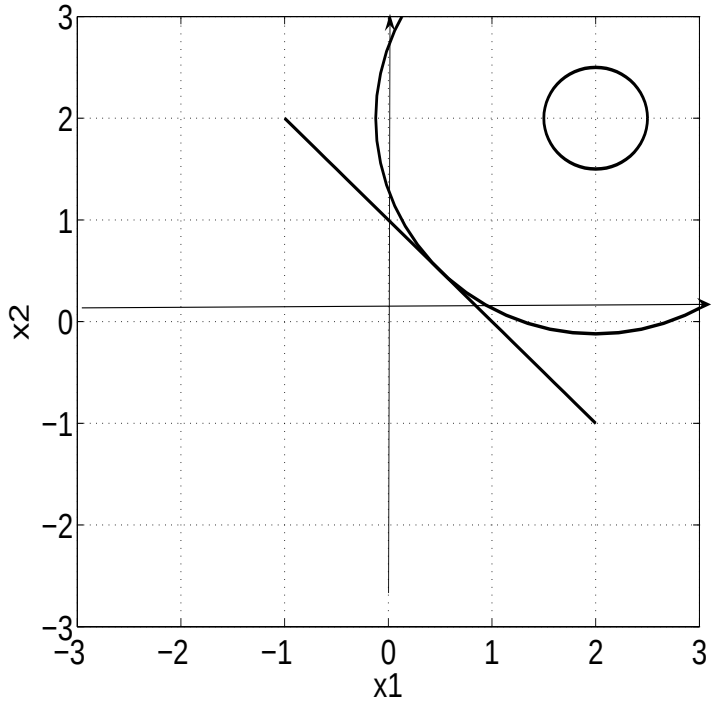


Figure 1: Illustration of constrained optimal solution

Lagrange Multipliers–I

- To minimize the objective function subject to equality constraints, let us consider the so-called Lagrange expression

$$J = \frac{1}{2}x^T Ex + x^T F + \lambda^T (Mx - \gamma). \tag{4}$$

- It is easy to see that the value of (4) subject to the equality constraints $Mx = \gamma$ being satisfied is the same as the original objective function.
- We now consider (4) as an objective function in $n + m$ variables x and λ, where n is the dimension of x and m is the dimension of λ.

Lagrange Multipliers–II

- The procedure of minimization is to take the first partial derivatives with respect to the vectors x and λ, and then equate these derivatives to zero.
- This gives us the results

$$\frac{\partial J}{\partial x} = Ex + F + M^T\lambda = 0 \quad (5)$$

$$\frac{\partial J}{\partial \lambda} = Mx - \gamma = 0. \quad (6)$$

- The linear equations (5) together with (6) contain $n + m$ variables x and λ, which are the necessary conditions for minimizing the objective function with equality constraints.
- The elements of the vector λ are called Lagrange multipliers.

Lagrange Multipliers–III

- The optimal λ and x are found via the set of linear equations defined by (5) and (6) where

$$\lambda = -(ME^{-1}M^T)^{-1}\left(\gamma + ME^{-1}F\right) \quad (7)$$

$$x = -E^{-1}(M^T\lambda + F). \quad (8)$$

- It is interesting to note that (8) can be written as two terms:

$$x = -E^{-1}F - E^{-1}M^T\lambda = x^0 - E^{-1}M^T\lambda,$$

where the first term $x^0 = -E^{-1}F$ is the global optimal solution that will give a minimum of the original cost function J without constraints, and the second term is a correction term due to the equality constraints.

Example–I

Minimize

$$J = \frac{1}{2} x^T E x + x^T F,$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 3x_1 - 2x_2 - 3x_3 &= 1, \end{aligned} \tag{9}$$

where $E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; F = \begin{bmatrix} -2 \\ -3 \\ -1 \end{bmatrix}.$

Solution. Without the equality constraints, the optimal solution is

$$x^0 = -E^{-1} F = \begin{bmatrix} 2 & 3 & 1 \end{bmatrix}^T.$$

Example–II

Writing the two equality constraints given by (9) in matrix form, we obtain the M and γ matrices as

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 3 & -2 & -3 \end{bmatrix}; \ \gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

To use (7) the following quantities are required

$$ME^{-1}M^T = \begin{bmatrix} 3 & -2 \\ -2 & 22 \end{bmatrix}; \ ME^{-1}F = \begin{bmatrix} -6 \\ 3 \end{bmatrix}.$$

Note that the determinant $\det(ME^{-1}M^T) = 62$, thus the matrix $ME^{-1}M^T$ is invertible. The λ vector is

$$\lambda = -(ME^{-1}M^T)^{-1}\left(\gamma + ME^{-1}F\right) = \begin{bmatrix} 1.6452 \\ -0.0323 \end{bmatrix}.$$

Example–III

The x vector that minimizes the objective function is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x^0 - E^{-1}M^T\lambda = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 3 \\ 1 & -2 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 1.6452 \\ -0.0323 \end{bmatrix}$$

$$= \begin{bmatrix} 0.4516 \\ 1.2903 \\ -0.7419 \end{bmatrix}.$$

Summary of Minimization with Equality Constraints

- The linear equality constraints are required to be linearly independent.
- The number of equality constraints is required to be less than or equal to the number of decision variables (*i.e.*, x).
- If the number of equality constraints equals the number of decision variables, the only feasible solution is the one that satisfies the constraints and there is no additional variable in x that can be used to optimize the original objective function.
- If the number of equality constraints is greater than the number of decision variables, then there is no feasible solution to satisfy the constraints. Alternatively, the situation is called infeasible.

Minimization with Inequality Constraints

- In the minimization with inequality constraints, the number of constraints could be larger than the number of decision variables.
- The inequality constraints $Mx \leq \gamma$ may comprise active constraints and inactive constraints.
- An inequality $M_i x \leq \gamma_i$ is said to be active if $M_i x = \gamma_i$ and inactive if $M_i x < \gamma_i$, where M_i together with γ_i form the ith inequality constraint and are the ith row of M matrix and the ith element of γ vector, respectively.

Kuhn-Tucker conditions

The necessary conditions for this optimization problem (Kuhn-Tucker conditions) are

$$\begin{aligned} Ex + F + M^T\lambda &= 0 \\ Mx - \gamma &\leq 0 \\ \lambda^T(Mx - \gamma) &= 0 \\ \lambda &\geq 0, \end{aligned} \tag{10}$$

where the vector λ contains the Lagrange multipliers.

Active Constraints–I

- Kuhn-Tucker conditions can be expressed in a simpler form in terms of the set of active constraints.
- Let S_{act} denote the index set of active constraints. Then the necessary conditions become

$$
\begin{aligned}
Ex + F + \sum_{i \in S_{act}} \lambda_i M_i^T &= 0 \\
M_i x - \gamma_i &= 0 \quad i \in S_{act} && (11)\\
M_i x - \gamma_i &< 0 \quad i \notin S_{act} && (12)\\
\lambda_i &\geq 0 \quad i \in S_{act} && (13)\\
\lambda_i &= 0 \quad i \notin S_{act}, && (14)
\end{aligned}
$$

  where M_i is the ith row of the M matrix.
- In other words, (11) says that for the ith row, $M_i x - \gamma_i = 0$ means that this is an equality constraint, hence an active constraint.

Active Constraints–II

- In contrast, $M_i x - \gamma_i < 0$ (see (12)) means that the constraint is satisfied, hence it is an inactive constraint.
- For an active constraint, the corresponding Lagrange multiplier is non-negative (see (13)), whilst the Lagrange multiplier is zero if the constraint is inactive (see (14)).

In the Case of Known Active Constraints

- It is clear that if the active set were known, the original problem could be replaced by the corresponding problem having equality constraints only.
- Explicitly, supposing that M_{act} and λ_{act} are given, the optimal solution with inequality solution has the closed-form

$$\lambda_{act} = -(M_{act}E^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E^{-1}F) \quad (15)$$

$$x = -E^{-1}(F + M_{act}^T\lambda_{act}). \quad (16)$$

Example

We assume that the objective function

$$J = \frac{1}{2} x^T E x + x^T F, \tag{17}$$

where the matrices $E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $F = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ and the constraints are

$$x_1 + x_2 \leq 1 \tag{18}$$

$$2x_1 + 2x_2 \leq 6. \tag{19}$$

Solution. Clearly the set of variables that satisfy inequality (18) will also satisfy the inequality (19). This is illustrated in Figure 2. Thus, the constraint (18) is an active constraint, while the constraint (19) is an inactive constraint. We find the constrained optimum by minimizing J subject to equality constraint: $x_1 + x_2 = 1$, which is $x_1 = 0.5$ and $x_2 = 0.5$. We verify that indeed with this set of x_1 and x_2 values, inequality (19) is satisfied.

Figure

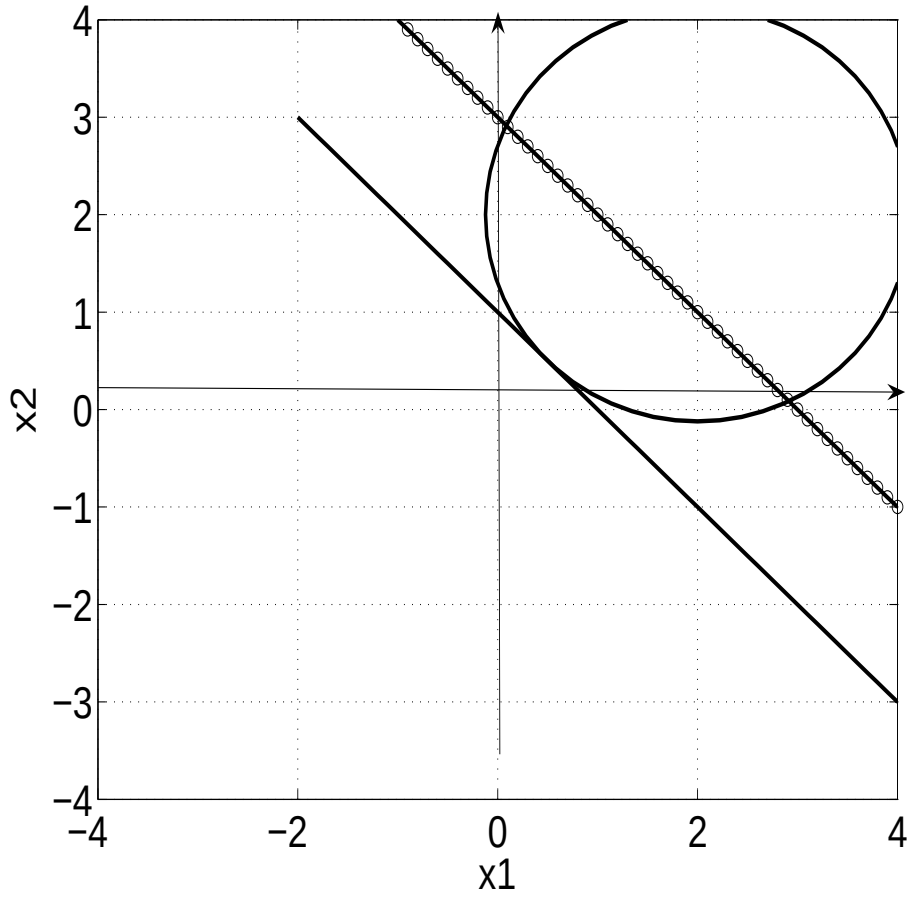


Figure 2: Illustration of the constrained optimization problem with inequality constraints. Solid-line $x_1 + x_2 = 1$; darker-solid-line $2x_1 + 2x_2 = 6$

Summary of Minimization with Inequality Constraints

- In the case of equality constraints, the maximum number of equality constraints equals the number of decision variables. In contrast, in the case of inequality constraints, the number of inequality constraints is permitted to be larger than the number of decision variables, as long as they are not all active. In this example, only one constraint becomes active so it becomes an equality constraint. Once the optimal solution is found against this active constraint, the rest of the inequalities are automatically satisfied.
- It is clear that an iterative procedure is required to solve the optimization problem with inequality constraints, because we did not know which constraints would become active constraints.
- Note that the conditions for the inequality constraints are more relaxed than the case of imposing equality constraints. For instance, the number of constraints is permitted to be greater than the number of decision variables, and the set of inequality constraints is permitted to be linearly dependent. However, these relaxations are only permitted to the point that the active constraints need to be linearly independent and the number of active constraints needs to be less than or equal to the number of decision variables.

Primal- Dual Approach

- Primal variables– means the decision variables, i.e. x.
- Dual variables– are the Lagrange multipliers, i.e. λ
- Primal-Dual approach used here will first solve the quadratic programming problem in dual variables λ and then go back to solve the decision variables x .
- It is particularly suitable for constrained control application because the solution procedure will allow us to detect
- the conflict constraints (dependent active constraints)
- over-specified constraints (number of active constraints is bigger than the number of decision variables).

The Dual Problem

- Assuming feasibility (*i.e.*, there is an x such that $Mx < \gamma$), the primal problem is equivalent to

$$\max_{\lambda \geq 0} \min_{x} [\frac{1}{2} x^T E x + x^T F + \lambda^T (Mx - \gamma)]. \tag{20}$$

- The minimization over x is unconstrained and is attained by

$$x = -E^{-1}(F + M^T \lambda). \tag{21}$$

- Substituting this in (20), the dual problem is written as

$$\max_{\lambda \geq 0} (-\frac{1}{2} \lambda^T H \lambda - \lambda^T K - \frac{1}{2} F^T E^{-1} F), \tag{22}$$

- where the matrices H and K are given by

$$H = M E^{-1} M^T \tag{23}$$

$$K = \gamma + M E^{-1} F. \tag{24}$$

The Dual Problem–continues

The dual is also a quadratic programming problem with λ as the decision variable. Equation (22) is equivalent to

$$\min_{\lambda \geq 0} (\frac{1}{2}\lambda^T H \lambda + \lambda^T K + \frac{1}{2}\gamma^T E^{-1}\gamma). \tag{25}$$

Note that the dual problem may be much easier to solve than the primal problem because the constraints are simpler. Here the simplified solution is found by using Hildreth's QP procedure.

Hildreth's QP Procedure– I

- It is a constrained gradient minimization algorithm.
- In this algorithm, the direction vectors were selected to be equal to the basis vectors $e_i = (0, 0, \ldots, 1, \ldots, 0, 0)$. Then the λ vector can be varied one component at a time. Thus it is a one-dimensional search algorithm.
- It is extremely easy for implementation.

Hildreth's QP Procedure– II

- At a given step in the process, we fix our attention on a single component λ_i.
- The objective function may be regarded as a quadratic function in this single component.
- We adjust λ_i to minimize the objective function. If that requires $\lambda_i < 0$, we set $\lambda_i = 0$.
- In any case, the objective function is decreased. Then we consider the next component λ_{i+1}.

Hildreth's QP Procedure– III

- The initial conditions for the search are set at zero.
- If we consider one complete cycle through the components to be one iteration taking the vector λ^m to λ^{m+1}, the method can be expressed explicitly as

$$\lambda_i^{m+1} = max(0, w_i^{m+1})$$

$$w_i^{m+1} = -\frac{1}{h_{ii}}[k_i + \sum_{j=1}^{i-1} h_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} h_{ij}\lambda_j^m]$$

- The positive Lagrange multipliers, denoted as λ_{act}, are then used in the solution of the optimal parameter vector η:

$$\eta = -E^{-1}(F + M_{act}^T \lambda_{act})$$

Example- I

- Minimize the cost function

$$J = \frac{1}{2} x^T E x + F^T x$$

where $E = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}; F = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$.

- The constraints are $0 \leq x_1$, $0 \leq x_2$ and $3x_1 + 2x_2 \leq 4$
- We form the linear inequality constraints

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}$$

Example -II

- The global optimal solution without constraints is

$$\begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix} = - \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- We notice that the inequality constraints are violated, with respect to the third constraint (i.e. 3+2>4).

Example –III

- We form

$$H = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 0 & 3 \\ 0 & -1 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & -5 \\ 1 & 2 & -7 \\ -5 & -7 & 29 \end{bmatrix}$$

$$K = \gamma + ME^{-1}F = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Example–IV

- We use Hildreth Programming procedure to identify the active constraint.
- The iteration starts at $k = 0$, with the initial conditions of $\lambda_1^0 = \lambda_2^0 = \lambda_3^0 = 0$.
- At $k = 1$,

$$\begin{aligned} w_1^1 + 1 &= 0 \\ 2w_2^1 + 1 &= 0 \\ 29w_3^1 - 1 &= 0 \end{aligned}$$

This gives $\lambda_1^1 = max(0, w_1^1) = 0$, $\lambda_2^1 = max(0, w_2^1) = 0$ and $\lambda_3^1 = max(0, w_3^1) = 0.0345$.

Example- V

- At $k = 2$,

$$\begin{aligned} w_1^2 + w_2^1 - 5w_3^1 + 1 &= 0 \\ w_1^1 + 2w_2^2 - 7w_3^1 + 1 &= 0 \\ 29w_3^2 - 1 &= 0 \end{aligned}$$

- This gives $\lambda_1^2 = max(0, w_1^2) = 0$, $\lambda_2^2 = max(0, w_2^2) = 0$ and $\lambda_3^2 = max(0, w_3^2) = 0.0345$.
- Since $\lambda^2 = \lambda^1$, the iterative procedure has converged. The optimal solution of λ is $\lambda_1^* = 0$, $\lambda_2^* = 0$ and $\lambda_3^* = 0.0345$.

Example–VI

- The optimal solution of x is given by

$$
\begin{aligned}
x^* &= \begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix} - E^{-1} M_{act}^T \lambda_{act}^* \\
&= \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{5}{29} \\ \frac{7}{29} \end{bmatrix} = \begin{bmatrix} 0.8276 \\ 0.7586 \end{bmatrix}
\end{aligned}
$$

- It is seen that the constrained optimal solution consists of two parts. The first part is the global optimal solution, and the second part is a correction term due to the active constraint.

MATLAB Tutorial–I

The objective of this tutorial is to demonstrate how to solve the constrained optimization problem using Hildreth programming. The problem is written as minimizing either

$$J = \frac{1}{2}\eta^T H \eta + \eta^T f, \tag{26}$$

$$\hat{J} = \eta^T H \eta + 2\eta^T f, \tag{27}$$

subject to constraints

$$A_{cons}\eta \leq b. \tag{28}$$

Step by Step

- Create a new file called $QPhild$.m.
- The program finds the global optimal solution and checks if all the constraints are satisfied. If so, the program returns the optimal solution η. If not, the program then begins to calculate the dual variable λ.

Tutorial–II

- Enter the following program into the file:

```
function eta=QPhild(H,f,A_cons,b);
% E=H;
% F=f;
% M=A_cons;
% gamma=b;
% eta =x
[n1,m1]=size(A_cons);
 eta=-H\f;
  kk=0;
   for i=1:n1
    if(A_cons(i,:)*eta>b(i)) kk=kk+1;
 else
kk=kk+0;
 end
end if (kk==0) return; end
```

Tutorial– III

- Note that in the quadratic programming procedure, the ith Lagrange multiplier λ_i becomes zero if the corresponding constraint is not active. Otherwise it is positive. We need to calculate the Lagrange multipliers iteratively. We will first set-up the matrices of the dual quadratic programming, followed by the computation of the Lagrange multipliers.
- Continue entering the following program into the file:

```
P=A_cons*(H\A_cons');
 d=(A_cons*(H\f)+b);
 [n,m]=size(d);
x_ini=zeros(n,m); lambda=x_ini;
```

Tutorial– IV

```
 al=10;
  for km=1:38
%find the elements in the solution vector one by one
% km could be larger if the Lagranger multiplier has
% a slow convergence rate.
lambda_p=lambda;
 for i=1:n
 w= P(i,:)*lambda-P(i,i)*lambda(i,1);
w=w+d(i,1); la=-w/P(i,i); lambda(i,1)=max(0,la);
 end
al=(lambda-lambda_p)'*(lambda-lambda_p);
 if (al<10e-8); break;
  end
end
```

Tutorial– V

- We can directly use the λ vector and the constraint equation to calculate the changes in η due to the active constraints, because the elements in λ are either positive or zero.
- Continue entering the following program into the file:

```
eta=-H\f -H\A_cons'*lambda;
```

Lecture 9: Constrained Model Predictive Control

Liuping Wang

School of Electrical and Computer Engineering

Royal Melbourne Institute of Technology University

Australia

Constrained Control

- MPC with constraints
- MATLAB Tutorial on Constrained Control

Example of Constrained Control

A continuous-time plant is described by a transfer function model,

$$G(s) = \frac{10}{s^2 + 0.1s + 3}, \tag{1}$$

which has a pair of poles at $-0.0500 \pm 1.7313j$. Suppose that the system is sampled with an interval $\Delta t = 0.01$. Design a discrete-time model predictive control with control horizon $N_c = 3$, $N_p = 20$, and $\bar{R} = 0.01 \times I$. The limit on the rate of change on the control signal is specified as

$$-1.5 \leq \Delta u(k) \leq 3.$$

For this example, we only consider the case of imposing the constraints on the first element of ΔU.

Solution–I

We obtain the objective function:

$$J = \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U - 2\Delta U^T\Phi^T(R_s - F\hat{x}(k_i)), \tag{2}$$

where

$$\Phi^T\Phi = \begin{bmatrix} 0.1760 & 0.1553 & 0.1361 \\ 0.1553 & 0.1373 & 0.1204 \\ 0.1361 & 0.1204 & 0.1057 \end{bmatrix}; \Phi^T F = \begin{bmatrix} 0.1972 & -0.1758 & 1.4187 \\ 0.1740 & -0.1552 & 1.2220 \\ 0.1522 & -0.1359 & 1.0443 \end{bmatrix},$$

and $R_s = \begin{bmatrix} 1.4187 \\ 1.2220 \\ 1.0443 \end{bmatrix} r(k_i)$. $r(k_i)$ and $\hat{x}(k_i)$ are the set-point signal and the estimated state variable at time k_i, respectively.

Solution–II

For simplicity, we assume that the observer poles are selected at $0, 0, 0$.
The closed-loop system without constraints has its eigenvalues located at $0.6851, 0.9109 \pm 0.1070j$, and $0, 0, 0$.
The constraints are translated to the two linear inequalities as

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \end{bmatrix} \leq \begin{bmatrix} 3.0000 \\ 1.5000 \end{bmatrix}. \tag{3}$$

Solution–III

To demonstrate how the solution evolves, we illustrate the first two steps in the computational procedure.

- Assume that the initial observer state is zero, and a set-point signal at time $k_i = 1$, is 1. Then, without constraints, the global optimal solution of ΔU is $[6.1083 \;\; 2.3334 \;\; -0.5861]^T$. Thus, the constraint is violated. The problem is solved using Hildreth's quadratic programming to obtain

$$\Delta U = [3.0000 \;\; 4.2751 \;\; 1.0494]^T.$$

- With the first component $\Delta u(1) = 3$, and $y(1) = 0$ the estimated state variable is updated to yield $\hat{x}(2) = [3.0000 \;\; 0 \;\; 0.0015]^T$.
- Again, the global optimal solution is $\Delta U = [4.6329 \;\; 1.1265 \;\; -1.5559]^T$, which violates the constraint. The optimization problem is again solved using the Hildreth's quadratic programming to obtain

$$\Delta U = [3.0000 \; 2.1466 \;\; -0.6967]^T.$$

Solution–IV

- With updated information on $y(2) = 0.0015$ and $\Delta u(2) = 3$, the estimated state variable is updated with value $\hat{x}(3) = [8.9961 \;\; 3.0000 \;\; 0.0075]^T$.
- The global optimal solution is $\Delta U = [2.9338 \quad -0.2126 \quad -2.5828]^T$, which satisfies the constraint.
- The computation continues simultaneously in closed-loop simulation and development of constrained control.

MATLAB Tutorial: Constrained Predictive Control

- The objective of this tutorial is to learn how to design and implement a state estimate predictive control system with constraints. We will use a DC motor model as an example in the study.

Tutorial Solution–I

Step by Step

- Create a new file called DCmotor.m. The first step is to enter the discrete-time state space model for the DC motor. Enter the following program into the file.

```
Ap=[0.9048 0;0.0952 1];
Bp=[0.0952;0.0048];
 Cp=[0 1];
 Dp=0;
```

- Choose the prediction horizon $N_p = 60$ and control horizon $N_c = 5$, we use the mpcgain program in the previous part to design the predictive controller, as well as the augmented model. Continue to enter the following program into the file:

Tutorial Solution–II

- ```
  Nc=5;
   Np=60;

  [Phi_Phi,Phi_F,PhiR,A_e,B_e,C_e]=mpcgain(Ap,Bp,Cp,Nc,Np)
  ```
- The observer is designed using pole-assignment technique. Continue entering the following program into the file:

  ```
  pole=[0.1 0.2 0.3];
   Kob=place(A_e',C_e', pole)';
  ```
- Specify the operational constraints for the DC motor. Continue entering the following program into the file:

  ```
  u_min=-0.2;
   u_max=0.6;
    deltau_min=-0.2;
    deltau_max=0.2;
  ```
  ```

Tutorial Solution–III

- Prepare the parameters for simulation, such as the initial conditions of the state variables, estimated state variables and set-point signal. Continue entering the following program into the file:

```
[n,n_in]=size(B_e);
 xm=[0;0];
 Xf=zeros(n,1);
 N_sim=100;
  r=ones(1,N_sim);
u=0;
 y=0;
  rw=1;
 Omega=Phi_Phi+rw*eye(Nc);
 Psi=Phi_F;
  Xr=zeros(n,1);
   Xr(n,1)=r(1);
```

Tutorial Solution–IV

- Prepare the matrices used in the inequality constraints. Continue entering the following program into the file:

```
L0=zeros(Nc,1);
  L0(1,1)=1;
  L1=zeros(Nc,1);
   L1(2,1)=1;
 A_cons_d=[L0';-L0'; L1'; -L1'];
 A_cons_a=[L0';-L0'; L0'+L1';-(L0'+L1')];
A_cons=[A_cons_d;A_cons_a];
 b_d=[deltau_max;-deltau_min; deltau_max;
-deltau_min];
```

Tutorial Solution–V

- We perform simulation of predictive control with constraints. We will first update the constraints on the control signal with u

```
for kk=1:N_sim;
b_a=[u_max-u;-u_min+u; u_max-u;-u_min+u];
b=[b_d;b_a];
```

- The set-point signal $r(kk)$ enters the simulation. Continue entering the following program into the file:

```
Xr(n,1)=r(kk);
Xfq=Xf-Xr;
```

- We find ΔU using constrained optimization algorithm (QP), and take the first sample as $\Delta u(kk)$. Continue entering the following program into the file:

```
DeltaU=QPhild(Omega,Psi*Xfq,A_cons,b);
 deltau=DeltaU(1,1);
```

Tutorial Solution–VI

- Update the observer and the control signal using the current $\Delta u(kk)$. Continue entering the following program into the file:

```
Xf=A_e*Xf+B_e*deltau+ Kob*(y-C_e*Xf);
 u=u+deltau;
  u1(kk)=u;
y1(kk)=y;
```

- Use the current control signal $u(kk)$ to generate the output $y(kk+1)$, and after that the simulate repeats for the next sample period. Continue entering the following program into the file:

```
xm=Ap*xm+Bp*u;
y=Cp*xm;
 end
```

Tutorial Solution–VII

- Presents the simulation results. Continue entering the following program into the file:

```
k=0:(N_sim-1);
 figure
 subplot(211)
  plot(k,y1,k,r,'r')
xlabel('Sampling Instant')
 legend('Output')
 subplot(212)
  plot(k,u1)
xlabel('Sampling Instant')
 legend('Control')
```

- Run this program. Impose different constraints on the control signal, for example, $-0.1 \leq \Delta u(k) \leq 0.1$; and $0 \leq u(k) \leq 0.6$, and compare the closed-loop performance.