predictive control (GPC) algorithm of Clarke and colleagues (Clarke *et al.*, 1987). The transfer function model-based predictive control is often considered to be less effective in handling multivariable plants. A state-space formulation of GPC has been presented in Ordys and Clarke (1993).

Recent years have seen the growing popularity of predictive control design using state-space design methods (Ricker, 1991, Rawlings and Muske, 1993, Rawlings, 2000, Maciejowski, 2002). In this book, we will use state-space models, both in continuous time and discrete time for simplicity of the design framework and the direct link to the classical linear quadratic regulators.

1.2 State-space Models with Embedded Integrator

Model predictive control systems are designed based on a mathematical model of the plant. The model to be used in the control system design is taken to be a state-space model. By using a state-space model, the current information required for predicting ahead is represented by the state variable at the current time.

1.2.1 Single-input and Single-output System

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

$$x_m(k+1) = A_m x_m(k) + B_m u(k), \tag{1.1}$$

$$y(k) = C_m x_m(k), \tag{1.2}$$

where 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. Note 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.

Note that a general formulation of a state-space model has a direct term from the input signal $u(k)$ to the output $y(k)$ as

$$y(k) = C_m x_m(k) + D_m u(k).$$

However, due to the principle of receding horizon control, where a current information of the plant is required for prediction and control, we have implicitly assumed that the input $u(k)$ cannot affect the output $y(k)$ at the same time. Thus, $D_m = 0$ in the plant model.

Taking a difference operation on both sides of (1.1), we obtain that

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

Let us denote the difference of the state variable by

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

and the difference of the control variable by

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

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

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k). \tag{1.3}$$

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

$$x(k) = \left[\Delta x_m(k)^T \; y(k) \right]^T,$$

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

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k)) = C_m \Delta x_m(k+1) \\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k). \end{aligned} \tag{1.4}$$

Putting together (1.3) with (1.4) leads to the following state-space model:

$$\begin{aligned} \overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} &= \overbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k) \\ y(k) &= \overbrace{\begin{bmatrix} o_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}, \end{aligned} \tag{1.5}$$

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 1.1. Consider a discrete-time model in the following form:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \tag{1.6}$$

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 (1.5) and calculate the eigenvalues of the system matrix, A, of the augmented model.

Solution. From (1.5), $n_1 = 2$ and $o_m = [0\ 0]$. The augmented model for this plant is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k), \end{aligned} \tag{1.7}$$

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

$$\begin{aligned} \rho(\lambda) &= \det(\lambda I - A) = \det \begin{bmatrix} \lambda I - A_m & o_m^T \\ -C_m A_m & (\lambda - 1) \end{bmatrix} \\ &= (\lambda - 1) \det(\lambda I - A_m) = (\lambda - 1)^3. \end{aligned} \tag{1.8}$$

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.

1.2.2 MATLAB Tutorial: Augmented Design Model

Tutorial 1.1. *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:*

$$\begin{aligned} \dot{x}_m(t) &= \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_m(t) + \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} x_m(t). \end{aligned} \tag{1.9}$$

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 discover 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.*

1.3 Predictive Control within One Optimization Window

Upon formulation of the mathematical model, the next step in the design of a 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. This section 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 the number of samples. For simplicity, the case of single-input and single-output systems is considered first, then the results are extended to multi-input and multi-output systems.

1.3.1 Prediction of State and Output 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 more general situation where the state is not directly measured will be discussed later. 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 the 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 the 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.

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

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

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

$$\begin{aligned}
y(k_i+1 \mid k_i) &= CAx(k_i) + CB\Delta u(k_i) \qquad (1.10) \\
y(k_i+2 \mid k_i) &= CA^2x(k_i) + CAB\Delta u(k_i) + CB\Delta u(k_i+1) \\
y(k_i+3 \mid k_i) &= CA^3x(k_i) + CA^2B\Delta u(k_i) + CAB\Delta u(k_i+1) \\
&\quad + CB\Delta u(k_i+2) \\
&\vdots \\
y(k_i+N_p \mid k_i) &= CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta u(k_i) + CA^{N_p-2}B\Delta u(k_i+1) \\
&\quad + \ldots + CA^{N_p-N_c}B\Delta u(k_i+N_c-1). \qquad (1.11)
\end{aligned}$$

Note that all predicted variables are formulated in terms of current state variable information $x(k_i)$ and the future control movement $\Delta u(k_i+j)$, where $j = 0, 1, \ldots N_c-1$.

Define vectors

$$Y = \left[\, y(k_i+1 \mid k_i)\ y(k_i+2 \mid k_i)\ y(k_i+3 \mid k_i) \ldots y(k_i+N_p \mid k_i) \,\right]^T$$

$$\Delta U = \left[\, \Delta u(k_i)\ \Delta u(k_i+1)\ \Delta u(k_i+2) \ldots \Delta u(k_i+N_c-1) \,\right]^T,$$

where in the single-input and single-output case, the dimension of Y is N_p and the dimension of ΔU is N_c. We collect (1.10) and (1.11) together in a compact matrix form as

$$Y = Fx(k_i) + \Phi\Delta U, \qquad (1.12)$$

where

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

1.3.2 Optimization

For a given set-point 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 set-point 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.

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

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U, \tag{1.13}$$

where the first term is linked to the objective of minimizing the errors between the predicted output and the set-point signal while 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. $\bar{R}$ 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 (1.13) 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 (1.13) is interpreted as the situation where we would carefully consider how large the ΔU might be and cautiously reduce the error $(R_s - Y)^T (R_s - Y)$.

To find the optimal ΔU that will minimize J, by using (1.12), J is expressed as

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U. \tag{1.14}$$

From the first derivative of the cost function J:

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - Fx(k_i)) + 2(\Phi^T \Phi + \bar{R}) \Delta U, \tag{1.15}$$

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

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)), \tag{1.16}$$

with the assumption that $(\Phi^T\Phi + \bar{R})^{-1}$ exists. The matrix $(\Phi^T\Phi + \bar{R})^{-1}$ is called the Hessian matrix in the optimization literature. Note that R_s is a data vector that contains the set-point information expressed as

$$R_s = \overbrace{[1\ 1\ 1\ \ldots\ 1]^T}^{N_p} r(k_i) = \bar{R}_s r(k_i),$$

where

$$\bar{R}_s = \overbrace{[1\ 1\ 1\ \ldots\ 1]^T}^{N_p}.$$

The optimal solution of the control signal is linked to the set-point signal $r(k_i)$ and the state variable $x(k_i)$ via the following equation:

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(\bar{R}_s r(k_i) - Fx(k_i)). \tag{1.17}$$

Example 1.2. Suppose that a first-order system is described by the state equation:

$$\begin{aligned} x_m(k+1) &= ax_m(k) + bu(k) \\ y(k) &= x_m(k), \end{aligned} \tag{1.18}$$

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 that 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. The augmented state-space equation is

$$\begin{aligned} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} &= \begin{bmatrix} a & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} b \\ b \end{bmatrix} \Delta u(k) \\ y(k) &= \begin{bmatrix} 0\ 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}. \end{aligned} \tag{1.19}$$

Based on (1.12), the F and Φ matrices take the following forms:

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

The coefficients in the F and Φ matrices are calculated as follows:

$$\begin{aligned} CA &= \begin{bmatrix} s_1 & 1 \end{bmatrix} \\ CA^2 &= \begin{bmatrix} s_2 & 1 \end{bmatrix} \\ CA^3 &= \begin{bmatrix} s_3 & 1 \end{bmatrix} \\ &\vdots \\ CA^k &= \begin{bmatrix} s_k & 1 \end{bmatrix}, \end{aligned} \tag{1.20}$$

where $s_1 = a$, $s_2 = a^2 + s_1$, ..., $s_k = a^k + s_{k-1}$, and

$$\begin{aligned} CB &= g_0 = b \\ CAB &= g_1 = ab + g_0 \\ CA^2B &= g_2 = a^2b + g_1 \\ &\vdots \\ CA^{k-1}B &= g_{k-1} = a^{k-1}b + g_{k-2} \\ CA^kB &= g_k = a^kb + g_{k-1}. \end{aligned} \tag{1.21}$$

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

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

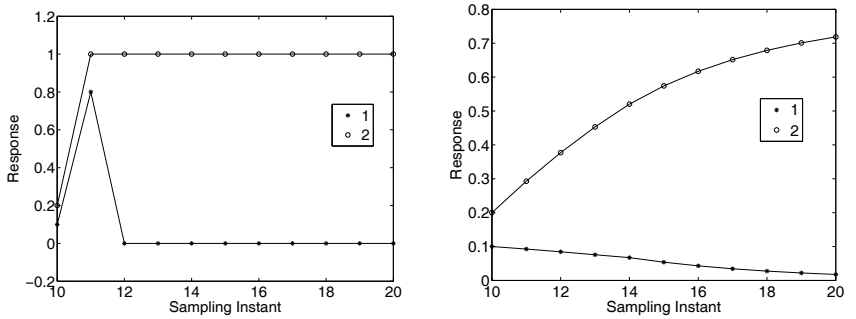
$$\Phi^T F = \begin{bmatrix} 9.2325 & 3.2147 \\ 8.3259 & 2.7684 \\ 7.2927 & 2.3355 \\ 6.1811 & 1.9194 \end{bmatrix}; \Phi^T \bar{R}_s = \begin{bmatrix} 3.2147 \\ 2.7684 \\ 2.3355 \\ 1.9194 \end{bmatrix}.$$

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 $\bar{R}_s$.

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 note 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. Figure 1.1a shows the changes of the state variables where we can see that the predicted output y has reached the desired set-point



(a) State variables with no weight on Δu

(b) State variables with weight on Δu

Fig. 1.1. Comparison of optimal solutions. Key: line (1) Δx_m; line (2) y

1 while the Δx_m decays to zero. 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,

$$\begin{aligned} \Delta U &= (\Phi^T \Phi + 10 \times I)^{-1} (\Phi^T R_s - \Phi^T F x(k_i)) \\ &= \left[0.1269 \; 0.1034 \; 0.0829 \; 0.065 \right]^T . \end{aligned} \tag{1.22}$$

With this choice, the magnitude of the first two control increments is significantly reduced, also the last two components are no longer zero. Figure 1.1b shows the optimal state variables. It is seen that the output y did not reach the set-point value of 1, however, the Δx_m approaches zero.

An observation follows from the comparison study. It seems that if we want the control to move cautiously, then it takes longer for the control signal to reach its steady state (*i.e.*, the values in ΔU decrease more slowly), because the optimal control energy is distributed over the longer period of future time. We can verify this by increasing N_c to 9, while maintaining $r_w = 10$. The result shows that the magnitude of the elements in ΔU is reducing, but they are significant for the first 8 elements:

$$\Delta U^T = \left[0.1227 \; 0.0993 \; 0.0790 \; 0.0614 \; 0.0463 \; 0.0334 \; 0.0227 \; 0.0139 \; 0.0072 \right] .$$

In comparison with the case where $N_c = 4$, we note that when $N_c = 9$, the first four parameters in ΔU are slightly different from the previous case.

Example 1.3. There is an alternative way to find the minimum of the cost function via completing the squares. This is an intuitive approach, also the minimum of the cost function becomes a by-product of the approach.
Find the optimal solution for ΔU by completing the squares of the cost function J (1.14).

Solution. From (1.14), by adding and subtracting the term

$$(R_s - Fx(k_i))^T \Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))$$

to the original cost function J, its value remains unchanged. This leads to

$$\begin{aligned} J = {} & (R_s - Fx(k_i))^T(R_s - Fx(k_i)) \\ & \overbrace{-2\Delta U^T\Phi^T(R_s - Fx(k_i)) + \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U} \\ & + \overbrace{(R_s - Fx(k_i))^T\Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))} \\ & - (R_s - Fx(k_i))^T\Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)), \end{aligned} \tag{1.23}$$

where the quantities under the $\overbrace{\cdot}$ are the completed 'squares':

$$\begin{aligned} J_0 = {} & \left(\Delta U - (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))\right)^T \\ & \times (\Phi^T\Phi + \bar{R})\left(\Delta U - (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i))\right). \end{aligned} \tag{1.24}$$

This can be easily verified by opening the squares. Since the first and last terms in (1.23) are independent of the variable ΔU (sometimes, we call this a decision variable), and $(\Phi^T\Phi + \bar{R})$ is assumed to be positive definite, then the minimum of the cost function J is achieved if the quantity J_0 equals zero, *i.e.*,

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)). \tag{1.25}$$

This is the optimal control solution. By substituting this optimal solution into the cost function (1.23), we obtain the minimum of the cost as

$$\begin{aligned} J_{min} = {} & (R_s - Fx(k_i))^T(R_s - Fx(k_i)) \\ & - (R_s - Fx(k_i))^T\Phi(\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)). \end{aligned}$$

1.3.3 MATLAB Tutorial: Computation of MPC Gains

Tutorial 1.2. *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.*

Step by Step

1. *Create a new file called mpcgain.m.*
2. *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);
```

3. *Note that the F and Phi matrices have special forms. By taking advantage of the special structure, we obtain the matrices.*
4. *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;
```

5. *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);
```

6. *Comparing the results with the answers from Example 1.2. If it is identical to what was presented there, then your program is correct.*
7. *Varying the prediction horizon and control horizon, observe the changes in these matrices.*
8. *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).*
9. *Validate the results in Example 1.2.*

1.4 Receding Horizon Control

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.

Example 1.4. We illustrate this procedure by continuing Example 1.2, 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. 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

$$x_m(11) = 0.8x_m(10) + 0.1u(10) = 0.88. \tag{1.26}$$

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

$$x_m(12) = 0.8x_m(11) + 0.1u(11) = 1. \tag{1.27}$$

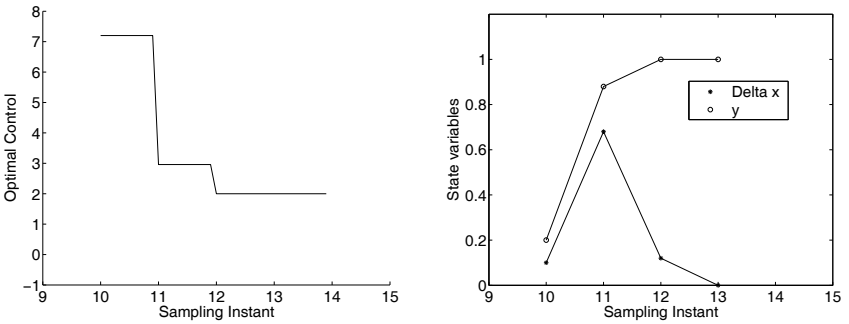
At $k_i = 12$, the new plant information is $\Delta x_m(12) = 1 - 0.88 = 0.12$ and $y(12) = 1$, which forms $x(12) = \begin{bmatrix} 0.12\ 1 \end{bmatrix}$. We obtain

$$\Delta U = (\Phi^T\Phi)^{-1}(\Phi^T R_s - \Phi^T F x(11)) = \begin{bmatrix} -0.96\ 0.000\ 0.0000\ 0.0000 \end{bmatrix}^T.$$

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

$$x_m(13) = ax_m(12) + bu(12) = 1. \tag{1.28}$$

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



(a) Optimal control (b) State variable

Fig. 1.2. Receding horizon control

$$\Delta U = (\Phi^T \Phi)^{-1}(\Phi^T R_s - \Phi^T F x(11)) = \begin{bmatrix} 0.000 & 0.000 & 0.0000 & 0.0000 \end{bmatrix}^T .$$

Figure 1.2 shows the trajectories of the state variable Δx_m and y, as well as the control signal that was used to regulate the output. This example also illustrated the differences between the ΔU parameter vectors at different time instances. We note that as the output response reaches the desired set-point signal, the parameters in the ΔU approach zero.

1.4.1 Closed-loop Control System

There is another aspect that was illustrated by Example 1.4. 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 $(\Phi^T \Phi + \bar{R})^{-1} \Phi^T R_s$ corresponds to the set-point change, while $-(\Phi^T \Phi + \bar{R})^{-1} \Phi^T F$ corresponds to the state feedback control within the framework of predictive 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

$$\begin{aligned} \Delta u(k_i) &= \overbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}}^{N_c} (\Phi^T \Phi + \bar{R})^{-1}(\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_{mpc} x(k_i), \end{aligned} \tag{1.29}$$

where 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

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

Equation (1.29) is in a standard form of linear time-invariant state feedback control. The state feedback control gain vector is K_{mpc}. Therefore, with the augmented design model

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

the closed-loop system is obtained by substituting (1.29) into the augmented system equation; changing index k_i to k, leading to the closed-loop equation

$$x(k+1) = Ax(k) - BK_{mpc}x(k) + BK_y r(k) \tag{1.30}$$

$$= (A - BK_{mpc})x(k) + BK_y r(k). \tag{1.31}$$

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

$$\det[\lambda I - (A - BK_{mpc})] = 0.$$

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_{mpc}. Noting that the state variable vector $x(k_i) = [\Delta x_m(k)^T\ y(k)]^T$, and with the definition of K_y, we can write $K_{mpc} = [K_x\ K_y]$, where K_x corresponds to the feedback gain vector related to $\Delta x_m(k)$, and K_y corresponds to the feedback gain related to $y(k)$. Then, we obtain the closed-loop block diagram for the predictive control system as in Figure 1.3 where q^{-1} denotes the backward shift operator. The diagram shows the state feedback structure for the discrete model prediction control (DMPC) with integral action in which the module $\frac{1}{1-q^{-1}}$ denotes the discrete-time integrator.

Example 1.5. This example will examine the closed-loop feedback gain matrices generated from Example 1.2 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

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

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

Hence, the eigenvalues of the closed-loop system are calculated by evaluating the eigenvalues of the closed-loop matrix $A - BK_{mpc}$, where, from Example 1.2

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

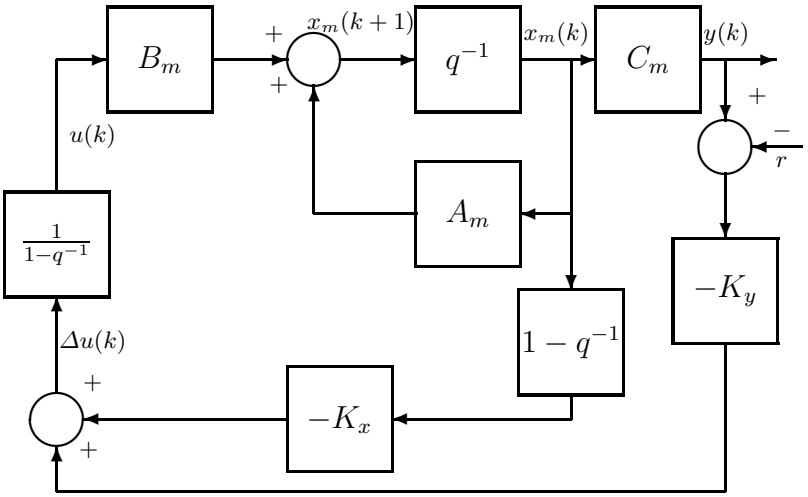


Fig. 1.3. Block diagram of discrete-time predictive control system

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.

When the weight $r_w = 10$, we have

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

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

With this gain vector, 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 one in the case when $r_w = 0$.

Example 1.6. Suppose that a continuous-time system is described by the Laplace transfer function

$$G(s) = \frac{\omega^2}{s^2 + 0.1\omega s + \omega^2},$$

where $\omega = 10$. This system is discretized using a sampling interval $\Delta t = 0.01$. Examine sensitivity issues in the selection of design parameters with $N_c = 3$, $N_p = 20$ and 200; $\bar{R} = 0.5I$.

Solution. We first use the MATLAB function script, given below, to obtain the continuous-time state-space model, then by following Tutorial 1.1 obtain the discrete-time state-space model.

```
omega=10;
numc=omega^2;
denc=[1 0.1*omega omega^2];
[Ac,Bc,Cc,Dc]=tf2ss(numc,denc);
```

Here, the augmented discrete-time state-space equation is

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

where

$$A = \begin{bmatrix} 0.9851 & -0.9934 & 0 \\ 0.0099 & 0.9950 & 0 \\ 0.9934 & 99.5021 & 1 \end{bmatrix}; B = \begin{bmatrix} 0.0099 \\ 0.0000 \\ 0.0050 \end{bmatrix}$$
$$C = [0 \quad 0 \quad 1].$$

Let us first look at the effect of the prediction horizon on the solution of ΔU. We assume that at sampling instant $k = 10$, the initial condition of $x(10) = [0.1 \quad 0.2 \quad 0.3]^T$. The solution of ΔU for $N_p = 20$ is $\Delta U = [-144.9984 \quad -65.4710 \quad 1.2037]^T$. By using the receding horizon control principle, the state feedback control gain is $K_{mpc} = [45.4168 \quad 705.6132 \quad 0.9513]$, and the closed-loop eigenvalues are 0.6974, $0.8959 \pm 0.1429j$. However, the solution of ΔU for $N_p = 200$ is $\Delta U = [-645.5885 \quad -0.4664 \quad 629.0276]^T$. In comparison with the previous case where the shorter prediction horizon $N_p = 20$ was used, the parameter vector ΔU has changed significantly. Again, from using the receding horizon control principle, the state feedback control gain is $K_{mpc} = [80.6 \quad 3190 \quad 0.79]$ and the resultant closed-loop eigenvalues are 0.9749, $0.5207 \pm j0.2919$. This comparison study illustrated the existing sensitivity in the design with respect to the choice of prediction horizon. Looking closely, we will discover that the Hessian matrix

$$\Phi^T\Phi + \bar{R}$$

is a function of the prediction horizon. For example, for $N_p = 20$

$$\Phi^T\Phi = \begin{bmatrix} 9.8796 & 8.9387 & 8.0099 \\ 8.9387 & 8.1020 & 7.2737 \\ 8.0099 & 7.2737 & 6.5425 \end{bmatrix},$$

with condition number $\kappa(\Phi^T\Phi + 0.5I) = 49.98$. However, for $N_p = 200$,

$$\Phi^T\Phi = \begin{bmatrix} 236.0557 & 235.5010 & 234.5466 \\ 235.5010 & 235.3753 & 234.8473 \\ 234.5466 & 234.8473 & 234.7473 \end{bmatrix},$$

with condition number $\kappa(\Phi^T\Phi + 0.5I) = 1410$. The condition number of the Hessian matrix has significantly increased as the prediction horizon N_p increased to 200. This large condition number of the Hessian matrix for a long

prediction horizon results in the numerical sensitivity that causes the significant difference between the short and long prediction horizon cases.

With short prediction and control horizons, the closed-loop predictive control system is not necessarily stable. Traditionally, these are the tuning parameters for closed-loop stability and performance. In Chapter 4, we will propose an approach that uses large prediction and control horizons so as to guarantee closed-loop stability.

1.4.2 MATLAB Tutorial: Implementation of Receding Horizon Control

Tutorial 1.3. *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*

$$x_m(k+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_m(k) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(k). \tag{1.32}$$

Step by Step

1. *Create a new file called reced.m*
2. *The first step is to define the plant, enter the values of prediction horizon and control horizon. The plant is the double integrator system (1.32). 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;
```

3. *The program calls the function mpcgain.m to generate the necessary gain matrices and specifies the initial conditions for implementation of receding horizon control. The initial state variable for the plant is xm=0; and the initial state feedback variable is Xf=0; set-point signal is specified and the number of simulation points is specified as 100.*
4. *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;
```

5. *From the receding horizon control, at sample time kk, the ΔU vector is calculated using the set-point signal $r(kk)$ and the state vector Xf. Then, $\Delta u(kk)$ is taken as the first element of ΔU; and $u(kk) = u(kk-1)+\Delta u(k)$. The weight factor is selected as* 0.1.
6. *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;
```

7. *The plant state and output are simulated using the control signal generated; the state variable used in the feedback mechanism is updated as Xf.*
8. *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
```

9. *The input and output signals are plotted against samples.*
10. *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')
```

11. *Save the program in the same directory as the one that contains the function. Run the program.*
12. *Change the weight r_w in the design to* 2 *and observe that the closed-loop response speed is slower.*
13. *Create your own plant using different Ap, Bp and Cp, and experiment with different prediction and control horizons.*

1.5 Predictive Control of MIMO Systems

In the previous section, for simplicity of illustration the predictive control system was designed based on a single-input and single-output system. This design methodology can be readily extended to multi-input and multi-output systems without much additional effort, because of the state-space formulation.

1.5.1 General Formulation of the Model

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.*, $q \leq m$). 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. In the general formulation of the predictive control problem, we also take the plant noise and disturbance into consideration.

$$x_m(k+1) = A_m x_m(k) + B_m u(k) + B_d \omega(k) \tag{1.33}$$

$$y(k) = C_m x_m(k), \tag{1.34}$$

where $\omega(k)$ is the input disturbance, assumed to be a sequence of integrated white noise. This means that the input disturbance $\omega(k)$ is related to a zero-mean, white noise sequence $\epsilon(k)$ by the difference equation

$$\omega(k) - \omega(k-1) = \epsilon(k). \tag{1.35}$$

Note that from (1.33), the following difference equation is also true:

$$x_m(k) = A_m x_m(k-1) + B_m u(k-1) + B_d \omega(k-1). \tag{1.36}$$

By defining $\Delta x_m(k) = x_m(k) - x_m(k-1)$ and $\Delta u(k) = u(k) - u(k-1)$, then subtracting (1.36) from (1.33) leads to

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k) + B_d \epsilon(k). \tag{1.37}$$

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

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

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

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

$$\begin{aligned}
\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} &= \begin{bmatrix} A_m & o_m^T \\ C_m A_m & I_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \Delta u(k) \\
&\quad + \begin{bmatrix} B_d \\ C_m B_d \end{bmatrix} \epsilon(k) \\
y(k) &= \begin{bmatrix} o_m & I_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix},
\end{aligned} \tag{1.38}$$

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. In (1.38), A_m, B_m and C_m have dimension $n_1 \times n_1$, $n_1 \times m$ and $q \times n_1$, respectively.

For notational simplicity, we denote (1.38) by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) + B_\epsilon \epsilon(k) \\ y(k) &= Cx(k), \end{aligned} \tag{1.39}$$

where A, B and C are matrices corresponding to the forms given in (1.38). In the following, the dimensionality of the augmented state-space equation is taken to be n $(= n_1 + q)$.

There are two points that are worth investigating here. The first is related to the eigenvalues of the augmented design model. The second point is related to the realization of the state-space model. Both points will help us understand the model.

Eigenvalues of the Augmented Model

Note that the characteristic polynomial equation of the augmented model is

$$\begin{aligned} \rho(\lambda) &= \det \begin{bmatrix} \lambda I - A_m & o_m^T \\ -C_m A_m & (\lambda - 1) I_{q\times q} \end{bmatrix} \\ &= (\lambda - 1)^q \det(\lambda I - A_m) = 0 \end{aligned} \tag{1.40}$$

where we used the property that 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. This is the means we use to obtain integral action for the MPC systems.

Controllability and Observability of the Augmented Model

Because the original plant model is augmented with integrators and the MPC design is performed on the basis of the augmented state-space model, it is important for control system design that the augmented model does not become uncontrollable or unobservable, particularly with respect to the unstable dynamics of the system. Controllability is a pre-requisite for the predictive control system to achieve the desired closed-loop control performance and observability is a pre-requisite for a successful design of an observer. However, the conditions may be relaxed to the requirement of stabilizability and detectability, if only closed-loop stability is of concern.[1] In this book, unless it is

[1] A system is stabilizable if its uncontrollable modes, if any, are stable. Its controllable modes may be stable or unstable. A system is detectable, if its unobservable modes, if any, are stable. Its observable modes may be stable or unstable. Stable modes here means that the corresponding eigenvalues are strictly inside the unit circle.

specifically stated, we require the model to be both controllable and observable in order to achieve desired closed-loop performance. An example is given in Section 1.6 to illustrate the importance of observability for the design of observer.

Because the augmented model introduced additional integral modes, we need to examine under what conditions these additional modes become controllable. The simplest way for the investigation is based on the assumption of minimal realization of the plant model. The discussion of minimal realization, controllability and observability can be found in control textbooks (for example Kailath (1980), Bay (1999)).

Definition: A realization of transfer function $G(z)$ is any state-space triplet (A, B, C) such that $G(z) = C(zI - A)^{-1}B$. If such a set (A, B, C) exists, then $G(z)$ is said to be realizable. A realization (A, B, C) is called a minimal realization of a transfer function if no other realization of smaller dimension of the triplet exists.
A minimal realization has the distinctive feature summarized in the theorem below.

Theorem 1.1. *A minimal realization is both controllable and observable (Kailath, 1980, Bay, 1999).*

With this background information, we aim to show conditions such that the augmented model is both controllable and observable through the argument of minimal realization.

Theorem 1.2. *Assume that the plant model (A_m, B_m, C_m) is both controllable and observable having the transfer function $G_m(z)$ with minimal realization, where*

$$G_m(z) = C_m(zI - A_m)^{-1}B_m.$$

Then, the transfer function of augmented design model (1.39) has the representation

$$G(z) = \frac{z}{z-1}G_m(z), \tag{1.41}$$

and is both controllable and observable if and only if the plant model $G_m(z)$ has no zero at $z = 1$.[2]

Proof. To prove that the augmented model is controllable and observable, we need to show that (1.41) is true. After that, the results follow from the minimal structure of the augmented model without pole-zero cancellation.

Note that for a given square matrix M with the block structure

$$M = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix},$$

[2] The zeros of a MIMO transfer function are those values of z that make the matrix $G_m(z)$ lose rank.

if A_{11}^{-1} and A_{22}^{-1} exist, then

$$M^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 \\ -A_{22}^{-1}A_{21}A_{11}^{-1} & A_{22}^{-1} \end{bmatrix}. \tag{1.42}$$

By applying the equality (1.42), we obtain

$$G(z) = C(zI - A)^{-1}B, \tag{1.43}$$

where

$$(zI - A)^{-1} = \begin{bmatrix} (zI_m - A_m)^{-1} & 0 \\ (1 - z^{-1})C_m A_m (zI_m - A_m)^{-1} & (1 - z^{-1})I_q \end{bmatrix}.$$

By substituting the B and C matrices from (1.39), the transfer function of the augmented model is obtained as (1.41). Under the assumption that the plant model $G_m(z)$ has no zero at $z = 1$ and has a minimal realization, the transfer function of the augmented model has a minimal structure from (1.41), therefore it is both controllable and observable.

For a single-input, single-output system, if one of the zeros of the transfer function is at $z = 1$, then the augmented model is not controllable. For instance, if

$$G_m(z) = \frac{(z-1)}{(z-0.6)(z-0.8)},$$

then there will be a pole-zero cancellation in $G(z)$, which is

$$G(z) = \frac{z}{z-1}\frac{(z-1)}{(z-0.6)(z-0.8)}.$$

In the single-input, single-output case, the steady-state gain of the plant model is zero, and it does not permit integral control.

We emphasize that the number of inputs is greater than or equal to the number of outputs ($m \geq q$). When the number of inputs is less than the number of outputs, the augmented integral modes may become uncontrollable.

When using MATLAB, minimal realization of a state-space model is achieved through model-order reduction. For example, when a discrete-time transfer function is

$$G_m(z) = \frac{(z-0.1)}{(z-0.1)(z-0.9)},$$

there is a pole-zero cancellation at $z = 0.1$. One of the realizations of the state-space model based on $G_m(z)$ using MATLAB function (tf2ss.m) has two state variables with

$$A_m = \begin{bmatrix} 1 & -0.09 \\ 1 & 0 \end{bmatrix};\ B_m = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; C_m = \begin{bmatrix} 1 & -0.1 \end{bmatrix}.$$

This is not a minimal realization as the corresponding transfer function has a pole-zero cancellation. To obtain a minimal state-space realization, the following MATLAB script is used for this simple illustrative example.

```
numd=[1 -0.1];
dend=conv([1 -0.1],[1 -0.9]);
sys1=tf(numd,dend);
sys=ss(sys1,'min');
[Am,Bm,Cm,Dm]=ssdata(sys);
```

The minimal realization through model-order reduction is

$$A_m = 0.9;\ B_m = -0.9806;\ C_m = -1.0198,$$

which only has one state variable as we expected in a minimal realization for this example.

1.5.2 Solution of Predictive Control for MIMO Systems

The extension of the predictive control solution is quite straightforward, and we need to pay attention to the dimensions of the state, control and output vectors in a mult-input, multi-output environment. Define the vectors Y and ΔU as

$$\Delta U = \left[\, \Delta u(k_i)^T\ \Delta u(k_i+1)^T\ \ldots\ \Delta u(k_i+N_c-1)^T \,\right]^T$$

$$Y = \left[\, y(k_i+1 \mid k_i)^T\ y(k_i+2 \mid k_i)^T\ y(k_i+3 \mid k_i)^T\ \ldots\ y(k_i+N_p \mid k_i)^T \,\right]^T.$$

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

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

With the assumption that $\epsilon(k)$ is a zero-mean white noise sequence, the predicted value of $\epsilon(k_i+i \mid k_i)$ at future sample i is assumed to be zero. The prediction of state variable and output variable is calculated as the expected values of the respective variables, hence, the noise effect to the predicted values being zero. For notational simplicity, the expectation operator is omitted without confusion.

Effectively, we have

$$Y = Fx(k_i) + \Phi\Delta U, \tag{1.44}$$

where

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

The incremental optimal control within one optimization window is given by

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}(\Phi^T\bar{R}_s r(k_i) - \Phi^T F x(k_i)), \tag{1.45}$$

where matrix $\Phi^T\Phi$ has dimension $mN_c \times mN_c$ and $\Phi^T F$ has dimension $mN_c \times n$, and $\Phi^T\bar{R}_s$ 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 $r(k_i) = [r_1(k_i) \;\; r_2(k_i) \;\; \dots \;\; r_q(k_i)]^T$ as the q set-point signals to the multi-output system.

Applying the receding horizon control principle, the first m elements in ΔU are taken to form the incremental optimal control:

$$\begin{aligned} \Delta u(k_i) &= \overbrace{\begin{bmatrix} I_m & o_m & \dots & o_m \end{bmatrix}}^{N_c}(\Phi^T\Phi + \bar{R})^{-1}(\Phi^T\bar{R}_s r(k_i) - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_{mpc} x(k_i), \end{aligned} \tag{1.46}$$

where I_m and o_m are, respectively, the identity and zero matrix with dimension $m \times m$.

Further work on predictive control approaches to MIMO systems will be presented in Chapter 3, where Laguerre functions will be used in the design. In that chapter, MATLAB functions will be given for the design of MIMO predictive control systems.

1.6 State Estimation

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. One approach is to select the state variables corresponding to the input and output using a special state-space realization (see Chapter 9) and the alternative is to estimate the state variable $x(k)$ from the process measurement. The 'soft' instrument used to estimate 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 the science and engineering fields. In addition, in a noisy environment, a state observer can also act like a noise filter to reduce the effect of noise on the measurement. Our focus here is to use an observer in the design of predictive control.

1.6.1 Basic Ideas About an Observer

It would not be difficult to imagine that an observer is constructed based on a mathematical model of the plant. For instance, we assume the plant state-space model:

$$x_m(k+1) = A_m x_m(k) + B_m u(k). \tag{1.47}$$

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

$$\hat{x}_m(k+1) = A_m \hat{x}_m(k) + B_m u(k). \tag{1.48}$$

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. What could be the problems with this type of approach? Basically, it is an open-loop prediction. The error $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$ satisfies the difference equation:

$$\begin{aligned} \tilde{x}_m(k+1) &= A_m(x_m(k) - \hat{x}_m(k)) \\ &= A_m \tilde{x}_m(k). \end{aligned} \tag{1.49}$$

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

$$\tilde{x}_m(k) = A_m^k \tilde{x}_m(0). \tag{1.50}$$

Two points are discussed here. If A_m has all eigenvalues inside the unit circle, then the error system (1.50) is stable and $||\tilde{x}_m(k)|| \to 0$ as $k \to \infty$, which means that the estimated state variable $\hat{x}_m(k)$ converges to $x_m(k)$. However, if A_m has one or more eigenvalues outside the unit circle, then the error system (1.50) is unstable and $||\tilde{x}_m(k)|| \to \infty$ as $k \to \infty$, which means that the prediction $\hat{x}_m(k)$ does not converge 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. The second point is that in the case of a stable plant model A_m, we have no 'control' on the convergence rate of the error $||\tilde{x}_m(k)|| \to 0$, 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 at 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. The observer is constructed using the equation:

$$\hat{x}_m(k+1) = \overbrace{A_m \hat{x}_m(k) + B_m u(k)}^{model} + \overbrace{K_{ob}(y(k) - C_m \hat{x}_m(k))}^{correction\ term}, \tag{1.51}$$

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 the original model,

and the second term is the correction term based on the error between the measured output and the predicted output using the estimate $\hat{x}_m(k)$.

To choose the observer gain K_{ob}, we examine the closed-loop error equation. By substituting $y(k) = C_m x_m(k)$ into (1.51), with the definition of error state $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$, we obtain that

$$\begin{aligned}\tilde{x}_m(k+1) &= A_m\tilde{x}_m(k) - K_{ob}C_m\tilde{x}_m(k)\\ &= (A_m - K_{ob}C_m)\tilde{x}_m(k).\end{aligned} \tag{1.52}$$

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

$$\tilde{x}_m(k) = (A_m - K_{ob}C_m)^k\tilde{x}_m(0). \tag{1.53}$$

Comparing the observer error response given by (1.53) with the open-loop prediction (1.50), it is apparent that the observer gain K_{ob} can be used to manipulate the convergence rate of the error. If there is only a 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. The following example shows how to select the observer gain K_{ob}.

Example 1.7. The linearized equation of motion of a simple pendulum is

$$\frac{d^2\theta}{dt^2} + \omega^2\theta = u, \tag{1.54}$$

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.

Solution. Let $x_1(t) = \theta$ and $x_2(t) = \dot{\theta}$, using the motion equation (1.54), the corresponding state-space model is

$$\begin{aligned}\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} &= \begin{bmatrix}0 & 1\\ -\omega^2 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}u(t)\\ y(t) &= \begin{bmatrix}0\ 1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}.\end{aligned} \tag{1.55}$$

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

$$\begin{bmatrix}x_1(k+1)\\ x_2(k+1)\end{bmatrix} = \begin{bmatrix}0.9801 & 0.0993\\ -0.3973 & 0.9801\end{bmatrix}\begin{bmatrix}x_1(k)\\ x_2(k)\end{bmatrix} + \begin{bmatrix}0.0050\\ 0.09930\end{bmatrix}u(k) \tag{1.56}$$

$$y(k) = \begin{bmatrix}0\ 1\end{bmatrix}\begin{bmatrix}x_1(k)\\ x_2(k)\end{bmatrix}. \tag{1.57}$$

To begin this study, 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 1.4a. 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 (1.48), the estimates of θ and $\dot{\theta}$ are calculated and shown in comparison to the true trajectories (see Figure 1.4a). 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 1.4a). This study demonstrated that using the model alone is not sufficient to predict the angle of the pendulum.

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

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

which is made to be equal to the desired closed-loop characteristic polynomial $(\lambda - 0.1)(\lambda - 0.2)$. 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 estimation of angle is carried out using the observer equation:

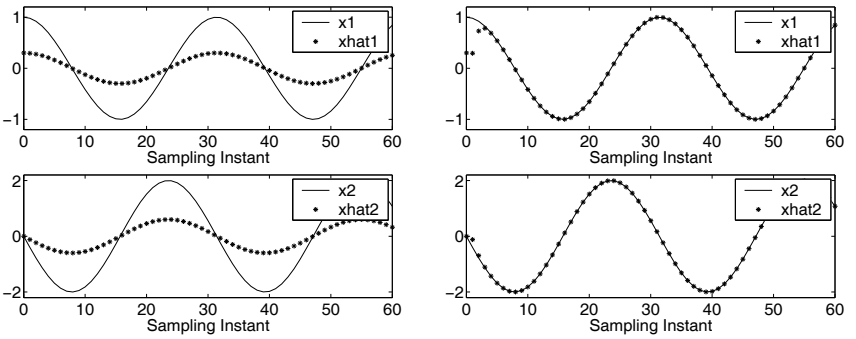
$$\begin{bmatrix} \hat{x}_1(k+1) \\ \hat{x}_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9801 & 0.0993 \\ -0.3973 & 0.9801 \end{bmatrix} \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix} + K_{ob}(x_2(k) - \hat{x}_2(k)), \quad (1.58)$$

with initial condition $\hat{x}_1(0) = 0.3$ and $\hat{x}_2(0) = 0$. Figure 1.4b shows that the estimated angle converges to the true angle in about three steps.

1.6.2 Basic Results About Observability

Definition of Observability

A state variable model of a dynamic system is said to be completely observable if, for any sample time k_0, there exists a sample time $k_1 > k_0$ such that a knowledge of the output $y(k)$ and input $u(k)$ in the time interval $k_0 \leq k \leq k_1$ is sufficient to determine the initial state $x_m(k_0)$ and as a consequence, $x_m(k)$, for all k between k_0 and k_1. A necessary and sufficient condition for a linear discrete-time system to be completely observable is, if the observability matrix



(a) Estimation without observer (b) Estimation with observer

Fig. 1.4. Observer design and implementation

$$L_o = \begin{bmatrix} C_m \\ C_m A_m \\ C_m A_m^2 \\ \vdots \\ C_m A_m^{n-1} \end{bmatrix}$$

has rank n, where n is the dimension of the state variable model.

Example 1.8. A DC motor can be described by a second-order model with an integrator and one time constant (see Figure 1.5). 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.

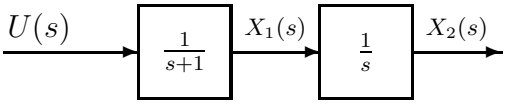


Fig. 1.5. Motor model

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:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t). \tag{1.59}$$

Suppose that we take the measurement of rotational speed, leading to

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

The model is discretized using a sampling interval $\Delta t = 0.1$ to give

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9048 & 0 \\ 0.0952 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0952 \\ 0.0048 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \tag{1.60}$$

Verify that the discrete-time model (1.60) is not observable, and as a consequence, the closed-loop observer system has a pole at 1.

Solution. The open-loop system has eigenvalues at 0.9048 and 1. The observability matrix is

$$L_o = \begin{bmatrix} C_m \\ C_m A_m \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0.9048 & 0 \end{bmatrix},$$

where $\det(L_o) = 0$. Thus, the pair (C_m, A_m) is not observable. Let us investigate this example further to discover what happens when the system is not observable. Assume that $K_{ob} = \begin{bmatrix} j_1 & j_2 \end{bmatrix}^T$. Then the closed-loop observer error system is:

$$\tilde{x}(k+1) = \left[\begin{bmatrix} 0.9048 & 0 \\ 0.0952 & 1 \end{bmatrix} - \begin{bmatrix} j_1 \\ j_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right] \tilde{x}(k). \tag{1.61}$$

Suppose that we put the desired eigenvalues of the observer system at 0.1 and 0.2.[3] Then the closed-loop characteristic polynomial is:

$$\det \begin{bmatrix} \lambda - 0.9048 + j_1 & 0 \\ -0.0952 & \lambda - 1 \end{bmatrix} = (\lambda - 0.9048 + j_1)(\lambda - 1). \tag{1.62}$$

In the design of the observer, we let the actual closed-loop characteristic polynomial equal the desired closed-loop characteristic polynomial, and solve for the observer gain vector. In this case, we would let

$$(\lambda - 0.9048 + j_1)(\lambda - 1) = (\lambda - 0.1)(\lambda - 0.2).$$

Note that the second pole at $\lambda = 1$ in (1.62) cannot be moved no matter what choice we make for j_2, simply because the closed-loop pole is independent of the observer gain. This is the consequence of the pair (C_m, A_m) being unobservable. If the pole that cannot be changed using the observer is asymptotically stable, then the system is not observable, however, it is detectable. In this particular example, the unobservable pole is on the unit circle, thus the system is not even detectable. So, if we measure the angular speed of the motor, then the angular position of the motor cannot be estimated accurately from this measurement.

[3] We also call the eigenvalues of the observer system as the closed-loop poles of the observer.

1.6.3 Kalman Filter

If the pair (A_m, C_m) is observable, then for the single-output case, as we discussed, a pole-assignment strategy can be used to determine K_{ob} such that the eigenvalues of the observer (*i.e.*, of the matrix $A_m - K_{ob}C_m$) are at desired locations. For a multi-output system, K_{ob} can be calculated recursively using a Kalman filter. Kalman filters are proposed in a stochastic setting. To this end, we assume that

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) + d(k) \\ y(k) &= C_m x_m(k) + \xi(k), \end{aligned} \tag{1.63}$$

with the covariance matrix of d and ξ, respectively, defined by

$$E\{d(k)d(\tau)^T\} = \Theta\delta(k-\tau)$$

$$E\{\xi(k)\xi(\tau)^T\} = \Gamma\delta(k-\tau),$$

where $\delta(k-\tau) = 1$, if $k = \tau$ and $\delta(k-\tau) = 0$ if $k \neq \tau$.

The optimal observer gain K_{ob} is solved recursively for $i = 0, 1, \ldots$, using

$$K_{ob}(i) = A_m P(i) C_m^T (\Gamma + C_m P(i) C_m^T)^{-1}, \tag{1.64}$$

and

$$P(i+1) = A_m\{P(i) - P(i)C_m^T(\Gamma + C_m P(i) C_m^T)^{-1} C_m P(i)\}A_m^T + \Theta. \tag{1.65}$$

More specifically, $P(0)$ satisfies

$$E\{[x(0) - \hat{x}(0)][x(0) - \hat{x}(0)]^T\} = P(0).$$

Assuming that the system (C_m, A_m) is detectable from the output $y(k)$ (*i.e.*, there are no unstable states whose response can not be 'seen' from the output) and $(A_m, \Theta^{1/2})$ is stabilizable, then, as $k \to \infty$, the steady-state solutions of (1.64) and (1.65) satisfy the discrete-time algebraic Riccati equation:

$$P(\infty) = A_m\{P(\infty) - P(\infty)C_m^T(\Gamma + C_m P(\infty) C_m^T)^{-1} C_m P_m(\infty)\}A_m^T + \Theta, \tag{1.66}$$

and

$$K_{ob}(\infty) = A_m P(\infty) C_m^T (\Gamma + C_m P(\infty) C_m^T)^{-1}. \tag{1.67}$$

Also, the eigenvalues of $A_m - K_{ob}(\infty)C_m$ are guaranteed to be inside the unit circle (*i.e.* stable). To avoid confusion, it is emphasized that the iterative solution of the Riccati equation (1.65) is not required in real time. The observer gain is calculated off-line for predictive control applications.

Kalman filters can be found in the textbooks (see for example, Anderson and Moore, 1979, Grimble and Johnson 1988b, Goodwin, *et al.*, 2000).

1.6.4 Tuning Observer Dynamics

It is often the case that the covariance matrices Θ and Γ, corresponding to the characteristics of the disturbances, are unknown. Thus, in practice, we choose Θ, Γ and an initial $P(0)$ to calculate an observer gain K_{ob} by solving the Riccati equation iteratively until the solution converges to a constant matrix. Then, the closed-loop system obtained is analyzed with respect to the location of eigenvalues contained in $A_m - K_{ob}C_m$, the transient response of the observer, robustness and effect of noise on the response. The elements of the covariance matrices are modified until a desired result is obtained. Such a trial-and-error procedure can be time consuming and frustrating, and is one of the challenges we face when using Kalman-filter-based multivariable system design. In some circumstances, however, it is possible to specify a region in which the closed-loop observer error system poles should reside and to enforce this in the solution. We propose a simple approach, along similar lines to the classic approach in Anderson and Moore (1979), in which the closed-loop observer poles are assigned inside a circle with a pre-specified radius α ($0 < \alpha < 1$). The procedure is summarized as follows. Let the error of the estimated state $\tilde{x}(k) = x(k) - \hat{x}(k)$. Then the observer error system is:

$$\tilde{x}(k+1) = (A_m - K_{ob}C_m)\tilde{x}(k). \tag{1.68}$$

We perform the transformation $\hat{A}_m = \frac{A_m}{\alpha}$ and $\hat{C}_m = \frac{C_m}{\alpha}$ where $0 < \alpha < 1$, leading to a transformed system:

$$\tilde{x}_t(k+1) = \frac{1}{\alpha}(A_m - \hat{K}_{ob}C_m)\tilde{x}_t(k) = (\hat{A}_m - \hat{K}_{ob}\hat{C}_m)\tilde{x}_t(k). \tag{1.69}$$

Solving the iterative equations (1.64) and (1.65), or the steady-state Riccati equation (1.66) by using $\hat{A}_m$ and $\hat{C}_m$ to replace A_m and C_m matrices, and then the eigenvalues of $\hat{A}_m - \hat{K}_{ob}(\infty)\hat{C}_m$ are guaranteed to be inside the unit circle (*i.e.*, stable). The resultant observer gain $\hat{K}_{ob}$ is then applied to the original observer system (1.68), leading to the closed-loop characteristic equation:

$$\det(zI - (A_m - \hat{K}_{ob}C_m)) = \det(zI - (\hat{A}_m - \hat{K}_{ob}\hat{C}_m) \times \alpha) = 0. \tag{1.70}$$

Therefore, we conclude that the eigenvalues of $(A_m - \hat{K}_{ob}C_m)$ are equal to the eigenvalues of $\hat{A}_m - \hat{K}_{ob}\hat{C}_m$ multiplied by the factor α, which guarantees that the eigenvalues of the observer error system with $\hat{K}_{ob}$ reside inside the circle of radius α. This procedure makes a direct connection to the observer dynamics via the choice of α. The trial-and-error procedure can be reduced to choose a suitable α along with Θ and Γ to achieve the desired closed-loop performance.

1.7 State Estimate Predictive Control

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:

$$\hat{x}(k_i+1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i)). \tag{1.71}$$

Note that in the implementation of predictive control using an observer, the control signal is $\Delta u(k_i)$ and the matrices (A, B, C) come from the augmented model used for the predictive control design. With the information of $\hat{x}(k_i)$ replacing $x(k_i)$, the predictive control law is then modified to find ΔU by minimizing

$$\begin{aligned} J = &(R_s - F\hat{x}(k_i))^T(\bar{R}_s r(k_i) - F\hat{x}(k_i)) - 2\Delta U^T \Phi^T(R_s - F\hat{x}(k_i)) \\ &+ \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U, \end{aligned} \tag{1.72}$$

where $\bar{R}_s$, F, Φ , $\bar{R}$ and ΔU were defined in (1.44 and 1.45).

The optimal solution ΔU is obtained as,

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - F\hat{x}(k_i)). \tag{1.73}$$

Furthermore, application of the receding horizon control principle leads to the optimal solution of $\Delta u(k_i)$ at time k_i:

$$\Delta u(k_i) = K_y r(k_i) - K_{mpc}\hat{x}(k_i), \tag{1.74}$$

which is a standard state feedback control law with estimated $x(k_i)$. The closed-loop state feedback structure is illustrated in Figure 1.6.

What about the closed-loop characteristic equation, and hence the eigenvalues of the closed-loop system? To investigate these issues, we form the closed-loop state-space equation as below:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ &= Ax(k) + BK_y r(k) - BK_{mpc}\hat{x}(k), \end{aligned} \tag{1.75}$$

where we substituted $\Delta u(k)$ with (1.74). Note that the closed-loop observer error equation is:

$$\tilde{x}(k+1) = (A - K_{ob}C)\tilde{x}(k), \tag{1.76}$$

where $\tilde{x}(k) = x(k) - \hat{x}(k)$. Replacing $\hat{x}(k)$ by $x(k) - \tilde{x}(k)$, (1.75) is rewritten as,

$$x(k+1) = (A - BK_{mpc})x(k) - BK_{mpc}\tilde{x}(k) + BK_y r(k). \tag{1.77}$$

Combination of (1.76) with (1.77) leads to:

$$\begin{bmatrix} \tilde{x}(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} A - K_{ob}C & o_{n\times n} \\ -BK_{mpc} & A - BK_{mpc} \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ x(k) \end{bmatrix} + \begin{bmatrix} o_{n\times m} \\ BK_y \end{bmatrix} r(k), \tag{1.78}$$

where $o_{n\times n}$ is a $n \times n$ zero matrix and $o_{n\times m}$ is a $n \times m$ zero matrix. The characteristic equation of the closed-loop state-space system is determined by

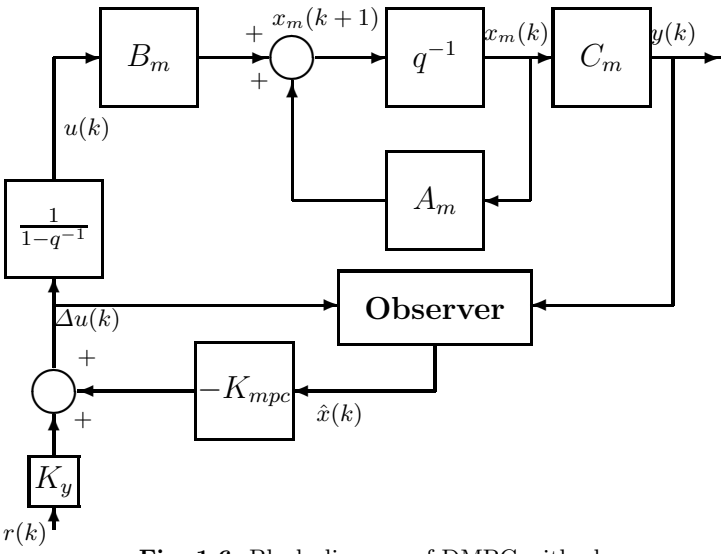


Fig. 1.6. Block diagram of DMPC with observer

$$\det\left[\lambda I - \begin{bmatrix} A - K_{ob}C & o_{n\times n} \\ -BK_{mpc} & A - BK_{mpc} \end{bmatrix}\right] = 0,$$

which is equivalent to

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

because the system matrix in (1.78) 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

$$\det(\lambda I - (A - K_{ob}C)) = 0 \tag{1.79}$$
$$\det(\lambda I - (A - BK_{mpc})) = 0. \tag{1.80}$$

Since the closed-loop eigenvalues are the solutions of the characteristic equations, (1.79) and (1.80) indicate that the set of eigenvalues of the combined closed-loop system 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 in this way, the eigenvalues remain unchanged.

Example 1.9. The augmented model for a double integrated plant (see Example 1.1) is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k), \end{aligned} \tag{1.81}$$

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 the 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. 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^T and C^T. 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

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

With this set of performance parameters specified, the state feedback control gain is $K_{mpc} = [0.8984\ 1.3521\ 0.4039]$, which effectively yields a set of closed-loop eigenvalues at $0.3172 \pm j0.4089$ and 0.3624. Figure 1.7 shows 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.

1.8 Summary

This chapter has discussed the basic ideas about discrete-time model predictive control. The key terms are: moving horizon window, prediction horizon and control horizon. With the current plant information represented by the state variable vector $x(k_i)$, the prediction of the future behaviour of the plant output relies on the state-space model where the optimal control trajectory is captured by the set of parameters that define the incremental control movement as: $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$. Within the optimization window, the objective of the control system is expressed in terms of the error function between the desired set-point signal and the predicted output signal. With a specific choice of an error function as the measurement of the objective, an optimal solution is obtained for the set of incremental control

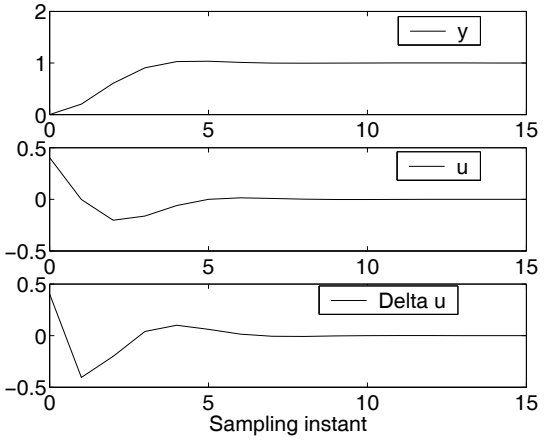


Fig. 1.7. Predictive control of double integrator plant

movement: $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$. Although the optimal control trajectory is calculated for N_c future samples, the implementation of the predictive control uses only the first sample, $\Delta u(k_i)$ while ignoring the rest of the trajectory. The optimization procedure repeats itself when the next sample period arrives. This is based on the receding horizon control principle, where feedback is naturally incorporated in the control system design.

The design model used here is an augmented plant model with embedded integrator(s). By doing so, the control signal to be optimized is the sequence of $\Delta u(k_i + m)$, $m = 0, 1, 2, \ldots$, instead of the sequence of the control signal $u(k_i + m)$. An integrator is naturally embedded into the design, leading to the predictive control system tracking constant references and rejecting constant disturbances without steady-state errors. Another significant advantage of this approach is that in implementation, it neither requires the steady-state information about the control ($u(k) = u(k-1) + \Delta u(k)$) nor the information about the steady state of the state variable x_m (Δx_m at steady state is zero). This simplified information requirement becomes more important for a plant having many inputs and many outputs. Because of this formulation of embedding integrators in the design, the model used for prediction has at least one eigenvalue on the unit circle. As a result, it inherits a numerical instability problem when the prediction horizon N_p becomes large. Stability cannot be guaranteed with a small prediction horizon and control horizon parameters, although it can be checked. Parameters N_p and N_c are used as tuning parameters. With some small modifications, as shown in the later chapters of this book (see Chapter 4), this numerical problem is overcome and stability is guaranteed.

There are several major reviews published for MPC that clarify the economic benefits of this class of control algorithms when applied to process industry

(Garcia *et al.*, 1989, Richalet, 1993, Qin and Badgwell, 1996, Morari and Lee, 1999, Mayne *et al.*, 2000). There are also several excellent tutorial papers published in the area of model predictive control (Ricker, 1991, Shah, 1995, Rawlings, 2000, Allgower *et al.*, 1999). Books about predictive control include 'Adaptive Optimal Control the thinking man's GPC' by Bitmead *et al.*, in 1990, 'Predictive Control' by Camacho and Bordons in 2004, 'Predictive Control with Constraints' by Maciejowski in 2002, and 'Model-based Predictive Control, a Practical Approach' by Rossiter in 2003. There is also a book about receding horizon control by Kwon and Han published in 2005.

Problems

1.1. Draw a road map for the contents of this chapter. List the key equations and key concepts at the corresponding locations.

1.2. Imagine yourself as a driver who is trying to steer a vehicle at a bend. Describe your activity in terms of a predictive control system.

1.3. Assume a discrete-time system with input $u(k)$ and output $y(k)$. The system has a constant input disturbance d. Find the augmented state-space model with input $\Delta u(k)$ and output $y(k)$ for the plant model given as below:

$$x_m(k+1) = A_m x_m(k) + B_m u(k) + B_d d;\ y(k) = C_m x_m(k) \tag{1.82}$$

where $A_m = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & -0.1 \\ 0 & 0 & 0.8 \end{bmatrix}$; $B_m = \begin{bmatrix} 0.5 \\ 1 \\ -0.6 \end{bmatrix}$; $C_m = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$; $B_d = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$.

1. Calculate the plant transfer function that relates the input $u(k)$ to the output $y(k)$, and the transfer function of the augmented state-space model that relates the input $\Delta u(k)$ to the output $y(k)$.
2. Compare the poles and zeros of these transfer functions.

1.4. Assume that the augmented mathematical model for a discrete-time system is given by

$$x(k+1) = Ax(k) + B\Delta u(k);\ y(k) = Cx(k), \tag{1.83}$$

where $A = \begin{bmatrix} 0.6 & 0 \\ 0.6 & 1 \end{bmatrix}$; $B = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix}$; $C = \begin{bmatrix} 0 & 1 \end{bmatrix}$.

1. At time $k_i = 0$, assuming control horizon $N_c = 4$, $N_p = 10$ and the initial state variable $x(0) = [0.1\ 0.2]^T$, express the predicted output

$$y(k_i + 1 \mid k_i),\ y(y_i + 2 \mid k_i),\ \ldots,\ y(k_i + N_p \mid k_i)$$

in terms of ΔU where

$$\Delta U = [\Delta u(k_i)\ \Delta u(k_i + 1)\ \Delta u(k_i + 2)\ \Delta u(k_i + 3)]^T.$$