

Condition 3: $|a_0| < a_2$

$$|0.368 + 0.066K| < 1 \quad (7.74)$$

For marginal stability

$$\begin{aligned} 0.368 + 0.066K &= 1 \\ K &= \frac{1 - 0.368}{0.066} = 9.58 \end{aligned} \quad (7.75)$$

Hence the system is marginally stable when $K = 9.58$ and 105.23 (see also Example 7.6 and Figure 7.20).

7.6.3 Root locus analysis in the z -plane

As with the continuous systems described in Chapter 5, the root locus of a discrete system is a plot of the locus of the roots of the characteristic equation

$$1 + GH(z) = 0 \quad (7.76)$$

in the z -plane as a function of the open-loop gain constant K . The closed-loop system will remain stable providing the loci remain within the unit circle.

7.6.4 Root locus construction rules

These are similar to those given in section 5.3.4 for continuous systems.

1. *Starting points* ($K = 0$): The root loci start at the open-loop poles.
2. *Termination points* ($K = \infty$): The root loci terminate at the open-loop zeros when they exist, otherwise at ∞ .
3. *Number of distinct root loci*: This is equal to the order of the characteristic equation.
4. *Symmetry of root loci*: The root loci are symmetrical about the real axis.
5. *Root locus locations on real axis*: A point on the real axis is part of the loci if the sum of the open-loop poles and zeros to the right of the point concerned is odd.
6. *Breakaway points*: The points at which a locus breaks away from the real axis can be found by obtaining the roots of the equation

$$\frac{d}{dz}\{GH(z)\} = 0$$

7. *Unit circle crossover*: This can be obtained by determining the value of K for marginal stability using the Jury test, and substituting it in the characteristic equation (7.76).

Example 7.6 (See also Appendix 1, *examp76.m*)

Sketch the root locus diagram for Example 7.4, shown in Figure 7.14. Determine the breakaway points, the value of K for marginal stability and the unit circle crossover.

Solution

From equation (7.43)

$$G(s) = K \left(1 - \frac{e^{-Ts}}{s} \right) \left\{ \frac{1}{s(s+2)} \right\} \quad (7.77)$$

and from equation (7.53), given that $T = 0.5$ seconds

$$G(z) = K \left(\frac{0.092z + 0.066}{z^2 - 1.368z + 0.368} \right) \quad (7.78)$$

Open-loop poles

$$z^2 - 1.368z + 0.368 = 0 \quad (7.79)$$

$$\begin{aligned} z &= 0.684 \pm 0.316 \\ &= 1 \text{ and } 0.368 \end{aligned} \quad (7.80)$$

Open-loop zeros

$$\begin{aligned} 0.092z + 0.066 &= 0 \\ z &= -0.717 \end{aligned} \quad (7.81)$$

From equations (7.67), (7.68) and (7.69) the characteristic equation is

$$z^2 + (0.092K - 1.368)z + (0.368 + 0.066K) = 0 \quad (7.82)$$

Breakaway points: Using Rule 6

$$\begin{aligned} \frac{d}{dz} \{GH(z)\} &= 0 \\ (z^2 - 1.368z + 0.368)K(0.092) - K(0.092z + 0.066)(2z - 1.368) &= 0 \end{aligned} \quad (7.83)$$

which gives

$$\begin{aligned} 0.092z^2 + 0.132z - 0.1239 &= 0 \\ z &= 0.647 \text{ and } -2.084 \end{aligned} \quad (7.84)$$

K for marginal stability: Using the Jury test, the values of K as the locus crosses the unit circle are given in equations (7.75) and (7.73)

$$K = 9.58 \text{ and } 105.23 \quad (7.85)$$

Unit circle crossover: Inserting $K = 9.58$ into the characteristic equation (7.82) gives

$$z^2 - 0.487z + 1 = 0 \quad (7.86)$$

The roots of equation (7.86) are

$$z = 0.244 \pm j0.97 \quad (7.87)$$

or

$$z = 1 \angle \pm 75.9^\circ = 1 \angle \pm 1.33 \text{ rad} \quad (7.88)$$

Since from equation (7.63) and Figure 7.16

$$z = |z| \angle \omega T \quad (7.89)$$

and $T = 0.5$, then the frequency of oscillation at the onset of instability is

$$\begin{aligned} 0.5\omega &= 1.33 \\ \omega &= 2.66 \text{ rad/s} \end{aligned} \quad (7.90)$$

The root locus diagram is shown in Figure 7.20.

It can be seen from Figure 7.20 that the complex loci form a circle. This is usually the case for second-order plant, where

$$\begin{aligned} \text{Radius} &= \sum |\text{open-loop poles}| \\ \text{Centre} &= (\text{Open-loop zero}, 0) \end{aligned} \quad (7.91)$$

The step response shown in Figure 7.15 is for $K = 1$. Inserting $K = 1$ into the characteristic equation gives

$$z^2 - 1.276z + 0.434 = 0$$

or

$$z = 0.638 \pm j0.164$$

This position is shown in Figure 7.20. The K values at the breakaway points are also shown in Figure 7.20.

7.7 Digital compensator design

In sections 5.4 and 6.6, compensator design in the s -plane and the frequency domain were discussed for continuous systems. In the same manner, digital compensators may be designed in the z -plane for discrete systems.

Figure 7.13 shows the general form of a digital control system. The pulse transfer function of the digital controller/compensator is written

$$\frac{U}{E}(z) = D(z) \quad (7.92)$$

and the closed-loop pulse transfer function become

$$\frac{C}{R}(z) = \frac{D(z)G(z)}{1 + D(z)GH(z)} \quad (7.93)$$

and hence the characteristic equation is

$$1 + D(z)GH(z) = 0 \quad (7.94)$$

7.7.1 Digital compensator types

In a continuous system, a differentiation of the error signal e can be represented as

$$u(t) = \frac{de}{dt}$$

Taking Laplace transforms with zero initial conditions

$$\frac{U}{E}(s) = s \quad (7.95)$$

In a discrete system, a differentiation can be approximated to

$$u(kT) = \frac{e(kT) - e(k-1)T}{T}$$

hence

$$\frac{U}{E}(z) = \frac{1 - z^{-1}}{T} \quad (7.96)$$

Hence, the Laplace operator can be approximated to

$$s = \frac{1 - z^{-1}}{T} = \frac{z - 1}{Tz} \quad (7.97)$$

Digital PID controller: From equation (4.92), a continuous PID controller can be written as

$$\frac{U}{E}(s) = \frac{K_1(T_i T_d s^2 + T_i s + 1)}{T_i s} \quad (7.98)$$

Inserting equation (7.97) into (7.98) gives

$$\frac{U}{E}(z) = \frac{K_1 \left\{ T_i T_d \left(\frac{z-1}{Tz} \right)^2 + T_i \left(\frac{z-1}{Tz} \right) + 1 \right\}}{T_i \left(\frac{z-1}{Tz} \right)} \quad (7.99)$$

which can be simplified to give

$$\frac{U}{E}(z) = \frac{K_1(b_2 z^2 + b_1 z + b_0)}{z(z-1)} \quad (7.100)$$

where

$$\begin{aligned} b_0 &= \frac{T_d}{T} \\ b_1 &= \left(1 - \frac{2T_d}{T} \right) \\ b_2 &= \left(\frac{T_d}{T} + \frac{T}{T_i} + 1 \right) \end{aligned} \quad (7.101)$$

Tustin's Rule: Tustin's rule, also called the bilinear transformation, gives a better approximation to integration since it is based on a trapezoidal rather than a rectangular area. Tustin's rule approximates the Laplace transform to

$$s = \frac{2(z-1)}{T(z+1)} \quad (7.102)$$

Inserting this value of s into the denominator of equation (7.98), still yields a digital PID controller of the form shown in equation (7.100) where

$$\begin{aligned} b_0 &= \frac{T_d}{T} \\ b_1 &= \left(\frac{T}{2T_i} - \frac{2T_d}{T} - 1 \right) \\ b_2 &= \left(\frac{T}{2T_i} + \frac{T_d}{T} + 1 \right) \end{aligned} \quad (7.103)$$

Example 7.7 (See also Appendix 1, *examp77.m*)

The laser guided missile shown in Figure 5.26 has an open-loop transfer function (combining the fin dynamics and missile dynamics) of

$$G(s)H(s) = \frac{20}{s^2(s+5)} \quad (7.104)$$

A lead compensator, see case study Example 6.6, and equation (6.113) has a transfer function of

$$G(s) = \frac{0.8(1+s)}{(1+0.0625s)} \quad (7.105)$$

- Find the z -transform of the missile by selecting a sampling frequency of at least 10 times higher than the system bandwidth.
- Convert the lead compensator in equation (7.105) into a digital compensator using the simple method, i.e. equation (7.97) and find the step response of the system.
- Convert the lead compensator in equation (7.105) into a digital compensator using Tustin's rule, i.e. equation (7.102) and find the step response of the system.
- Compare the responses found in (b) and (c) with the continuous step response, and convert the compensator that is closest to this into a difference equation.

Solution

- From Figure 6.39, lead compensator two, the bandwidth is 5.09 rad/s, or 0.81 Hz. Ten times this is 8.1 Hz, so select a sampling frequency of 10 Hz, i.e.

$T = 0.1$ seconds. For a sample and hold device cascaded with the missile dynamics

$$G(s) = \left(\frac{1 - e^{-Ts}}{s} \right) \left\{ \frac{20}{s^2(s+5)} \right\} \quad (7.106)$$

$$G(s) = (1 - e^{-Ts}) \left\{ \frac{20}{s^3(s+5)} \right\} \quad (7.107)$$

For $T = 0.1$, equation (7.107) has a z -transform of

$$G(z) = \frac{0.00296z^2 + 0.01048z + 0.0023}{z^3 - 2.6065z^2 + 2.2131z - 0.6065} \quad (7.108)$$

(b) Substituting

$$s = \frac{z-1}{Tz}$$

into lead compensator given in equation (7.105) to obtain digital compensator

$$D(z) = 0.8 \left\{ \frac{\frac{Tz+(z-1)}{Tz}}{\frac{Tz+0.0625(z-1)}{Tz}} \right\}$$

This simplifies to give

$$D(z) = \frac{5.4152z - 4.923}{z - 0.3846} \quad (7.109)$$

(c) Using Tustin's rule

$$s = \frac{2(z-1)}{T(z+1)}$$

Substituting into lead compensator

$$D(z) = 0.8 \left[\frac{\frac{T(z+1)+2(z-1)}{T(z+1)}}{\frac{T(z+1)+0.0625\{2(z-1)\}}{T(z+1)}} \right]$$

This simplifies to give

$$D(z) = \frac{7.467z - 6.756}{z - 0.111} \quad (7.110)$$

(d) From Figure 7.21, it can be seen that the digital compensator formed using Tustin's rule is closest to the continuous response. From equation (7.110)

$$\frac{U}{E}(z) = \frac{7.467 - 6.756z^{-1}}{1 - 0.111z^{-1}} \quad (7.111)$$

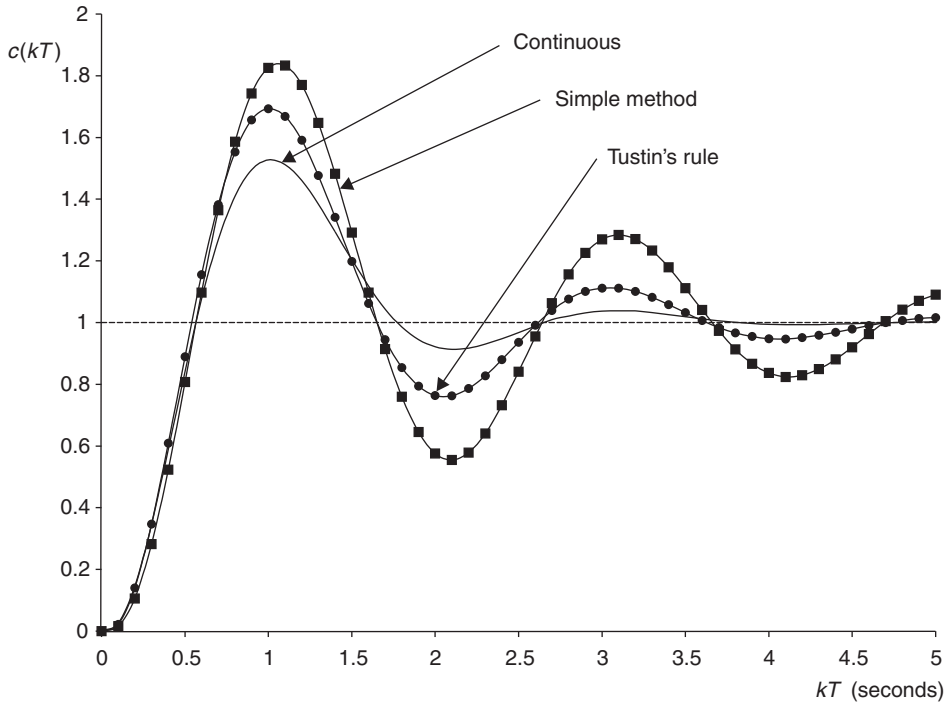


Fig. 7.21 Comparison between discrete and continuous response.

Hence the difference equation for the digital compensator is

$$u(kT) = 0.111u(k-1)T + 7.467e(kT) - 6.756e(k-1)T \quad (7.112)$$

7.7.2 Digital compensator design using pole placement

Case study

Example 7.8 (See also Appendix 1, *examp78.m*)

The continuous control system shown in Figure 7.22(a) is to be replaced by the digital control system shown in Figure 7.22(b).

- For the continuous system, find the value of K that gives the system a damping ratio of 0.5. Determine the closed-loop poles in the s -plane and hence the values of σ and ω .
- Find the closed-loop bandwidth ω_b and make the sampling frequency ω_s a factor of 10 higher. What is the value of T ?
- For the sampled system shown in Figure 7.22(b), find the open-loop pulse transfer function $G(z)$ when the sample and hold device is in cascade with the plant.
- With $D(z)$ set to the value of K found in (a), compare the continuous and discrete step responses.

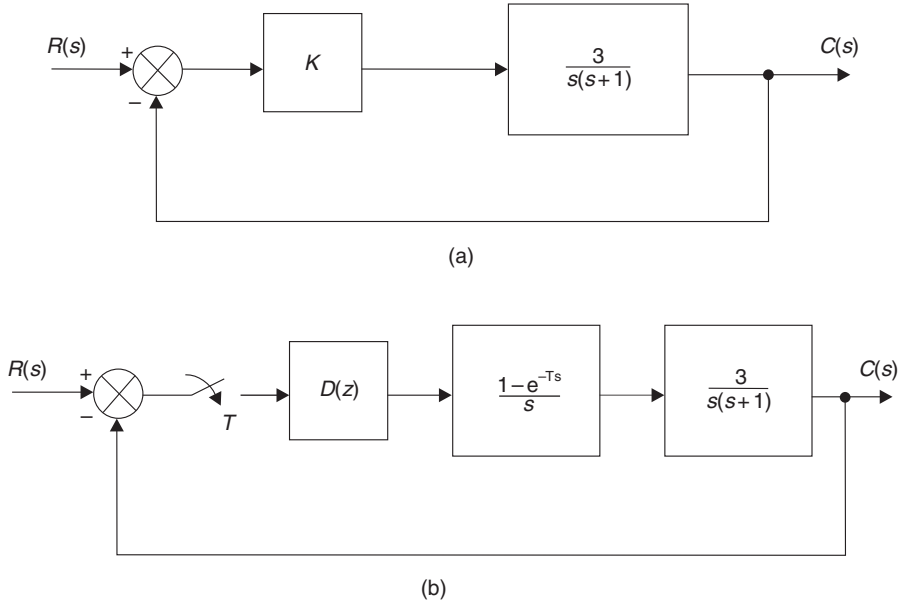


Fig. 7.22 Continuous and digital control systems.

- (e) By mapping the closed-loop poles from the s to the z -plane, design a compensator $D(z)$ such that both continuous and sampled system have identical closed-loop response, i.e. $\zeta = 0.5$.

Solution

- (a) The root-locus diagram for the continuous system is shown in Figure 7.23. From Figure 7.23 the closed-loop poles are

$$s = -0.5 \pm j0.866 \quad (7.113)$$

or

$$\sigma = -0.5, \quad \omega = 0.866 \text{ rad/s}$$

and the value of K is 0.336.

- (b) Plotting the closed-loop frequency response for the continuous system gives a bandwidth ω_b of 1.29 rad/s (0.205 Hz). The sampling frequency should therefore be a factor of 10 higher, i.e. 12.9 rad/s (2.05 Hz). Rounding down to 2.0 Hz gives a sampling time T of 0.5 seconds.

(c)
$$G(z) = (1 - z^{-1})Z\left\{\frac{3}{s^2(s+1)}\right\} \quad (7.114)$$

Using transform 7 in Table 7.1

$$G(z) = \frac{3\{(e^{-0.5} - 0.5)z + (1 - 1.5e^{-0.5})\}}{(z-1)(z-e^{-0.5})}$$

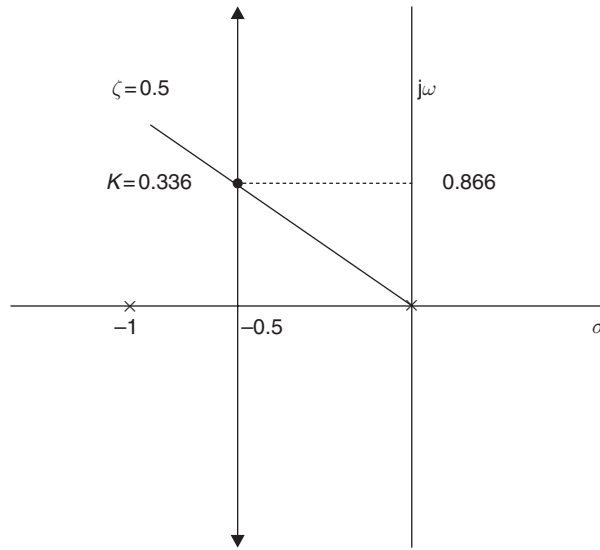


Fig. 7.23 Root locus diagram for continuous system.

Hence

$$G(z) = \frac{0.3196(z + 0.8467)}{(z - 1)(z - 0.6065)} \quad (7.115)$$

(d) With $D(z) = K = 0.336$, the difference between the continuous and discrete step response can be seen in Figure 7.24.

(e) Mapping closed-loop poles from s to z -plane

$$|z| = e^{\sigma T}$$

inserting values

$$|z| = e^{-0.5 \times 0.5} = 0.779 \quad (7.116)$$

$$\begin{aligned} \angle z &= \omega T \\ &= 0.866 \times 0.5 = 0.433 \text{ rad} \\ &= 24.8^\circ \end{aligned} \quad (7.117)$$

Converting from polar to cartesian co-ordinates gives the closed-loop poles in the z -plane

$$z = 0.707 \pm j0.327 \quad (7.118)$$

which provides a z -plane characteristic equation

$$z^2 - 1.414z + 0.607 = 0 \quad (7.119)$$

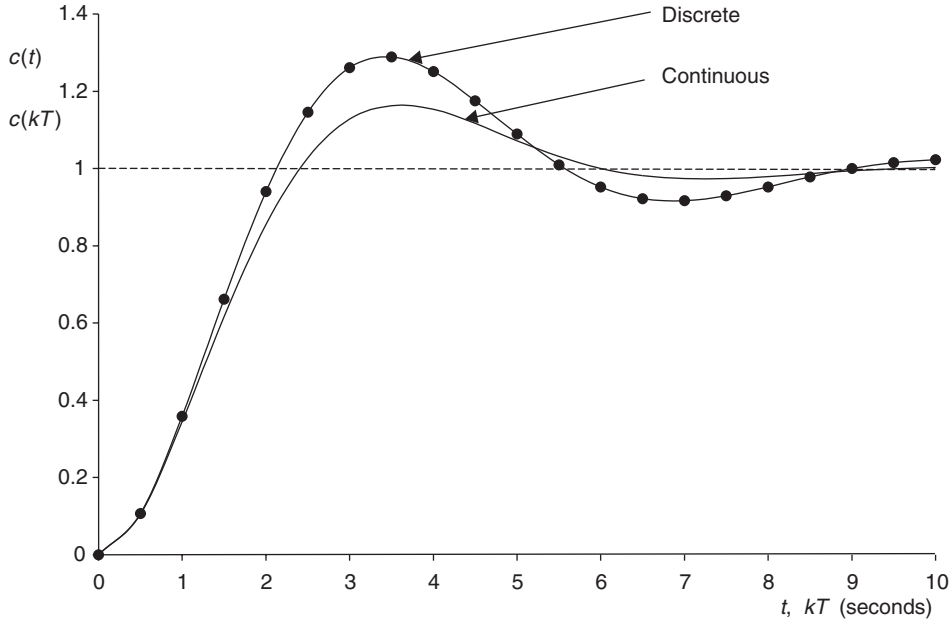


Fig. 7.24 Continuous and digital controllers set to $K = 0.336$.

The control problem is to design a compensator $D(z)$, which, when cascaded with $G(z)$, provides a characteristic equation

$$1 + D(z)G(z) = 0 \quad (7.120)$$

such that the equations (7.119) and (7.120) are identical. Let the compensator be of the form

$$D(z) = \frac{K(z - a)}{(z + b)} \quad (7.121)$$

Select the value of a so that the non-unity pole in $G(z)$ is cancelled

$$D(z)G(z) = \frac{K(z - 0.6065)}{(z + b)} \cdot \frac{0.3196(z + 0.8467)}{(z - 1)(z - 0.6065)} \quad (7.122)$$

Hence the characteristic equation (7.120) becomes

$$1 + \frac{0.3196K(z + 0.8467)}{(z + b)(z - 1)} = 0$$

which simplifies to give

$$z^2 + (0.3196K + b - 1)z + (0.2706K - b) = 0 \quad (7.123)$$

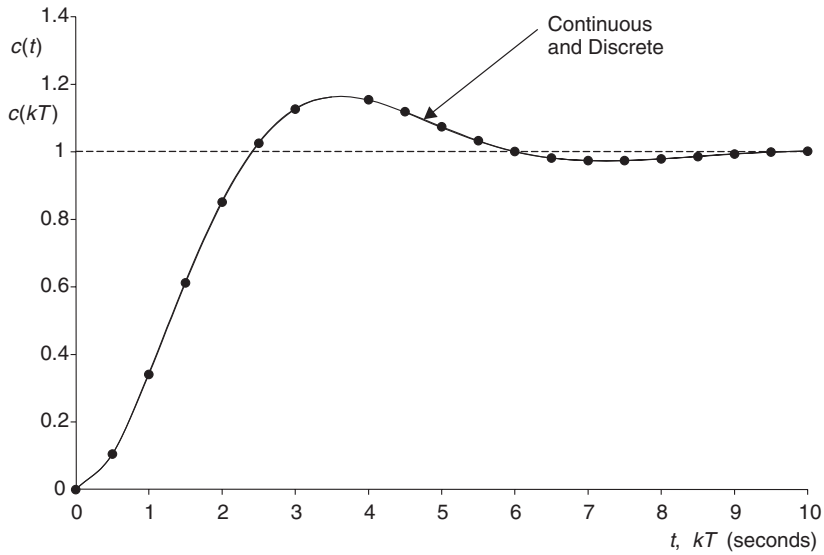


Fig. 7.25 Identical continuous and discrete step responses as a result of pole placement.

Equating coefficients in equations (7.119) and (7.123) gives

$$0.3196K + b - 1 = -1.414 \quad (7.124)$$

$$\frac{0.2706K - b}{0.5902K - 1} = 0.607 \quad (7.125)$$

$$\text{Add} \quad 0.5902K - 1 = -0.807$$

or

$$\begin{aligned} 0.5902K &= 0.193 \\ K &= 0.327 \end{aligned} \quad (7.126)$$

Inserting equation (7.126) into (7.125)

$$\begin{aligned} (0.2706 \times 0.327) - 0.607 &= b \\ b &= -0.519 \end{aligned} \quad (7.127)$$

Thus the required compensator is

$$D(z) = \frac{U}{E}(z) = \frac{0.327(z - 0.6065)}{(z - 0.519)} \quad (7.128)$$

Figure 7.25 shows that the continuous and discrete responses are identical, both with $\zeta = 0.5$. The control algorithm can be implemented as a difference equation

$$\frac{U}{E}(z) = 0.327 \frac{(1 - 0.6065z^{-1})}{(1 - 0.519z^{-1})} \quad (7.129)$$

hence

$$u(kT) = 0.327e(kT) - 0.1983e(k-1)T + 0.519u(k-1)T \quad (7.130)$$

7.8 Further problems

Example 7.9

Assuming that a sample and hold device is in cascade with the transfer function $G(s)$, determine $G(z)$ for the following

(a) $G(s) = \frac{1}{(s+1)}$, $T = 0.1$ seconds

(b) $G(s) = \frac{2}{(s+1)(s+2)}$, $T = 0.5$ seconds

(c) $G(s) = \frac{1}{s(s+0.5)}$, $T = 1.0$ seconds

Solution

(a) $G(z) = \frac{0.095}{z - 0.905}$

(b) $G(z) = \frac{0.155(z + 0.606)}{z^2 - 0.974z + 0.223}$

(c) $G(z) = \frac{0.426(z + 0.847)}{z^2 - 1.607z + 0.607}$

Example 7.10

The computer control system shown in Figure 7.26 has a sampling time of 0.5 seconds

- Find the open-loop pulse transfer function $G(z)$ and hence determine the open-loop poles and zeros for the combined sample and hold and the plant.
- From (a) evaluate the difference equation relating $c(kT)$, $c(k-1)T$, $c(k-2)T$, $u(k-1)T$ and $u(k-2)T$.
- If the computer has the control algorithm

$$u(kT) = 1.5e(kT)$$

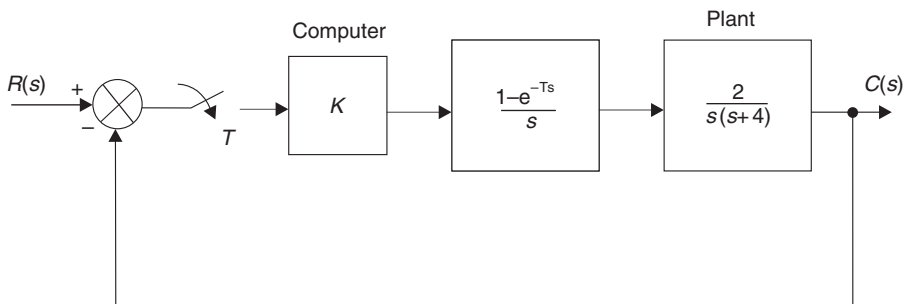


Fig. 7.26 Computer control system for Example 7.10.

using a tabular approach, calculate the system response when the input is a unit step applied at $kT = 0$ for the discrete time values of $kT = 0, 0.5, 1.0, 1.5, 2.0$ and 2.5 seconds. Assume that at kT less than zero, all values of input and output are zero.

Solution

$$(a) \quad G(z) = \frac{0.1419(z + 0.523)}{z^2 - 1.135z + 0.135}$$

poles $z = 1, 0.135$

zeros $z = -0.523$

$$(b) \quad c(kT) = 1.135c(k-1)T - 0.135c(k-2)T + 0.1419u(k-1)T + 0.0743u(k-2)T$$

(c)	kT	0	0.5	1.0	1.5	2.0	2.5
	$c(kT)$	0	0.213	0.521	0.752	0.889	0.959

Example 7.11

A unity feedback computer control system, has an open-loop pulse transfer function

$$G(z) = \frac{0.426K(z + 0.847)}{z^2 - 1.607z + 0.607}$$

- Determine the open-loop poles and zeros, the characteristic equation and break-away points.
- Using the Jury test, determine the value of K at the unit-circle crossover points.
- Find the radius and centre of the circular complex loci, and hence sketch the root locus in the z -plane.

Solution

$$(a) \quad \text{poles } z = 1, 0.607$$

$$\text{zeros } z = -0.847$$

$$z^2 + (0.426K - 1.607)z + (0.361K + 0.607) = 0$$

$$\text{breakaway points } z = 0.795, -2.5$$

$$(b) \quad K = 1.06, 47.9$$

$$(c) \quad \text{radius} = 1.607, \quad \text{centre} = -0.847, 0$$

Example 7.12

A unity feedback continuous control system has a forward-path transfer function

$$G(s) = \frac{K}{s(s+5)}$$

- Find the value of K to give the closed-loop system a damping ratio of 0.7. The above system is to be replaced by a discrete-time unity feedback control system with a forward-path transfer function

$$G(s) = D(z) \left(\frac{1 - e^{-Ts}}{s} \right) \left(\frac{1}{s(s+5)} \right)$$

- (b) If the sampling time is 0.2 seconds, determine the open-loop pulse transfer function.
- (c) The discrete-time system is to have the identical time response to the continuous system. What are the desired closed-loop poles and characteristic equations in
- the s -plane
 - the z -plane
- (d) The discrete-time compensator is to take the form

$$D(z) = \frac{K_1(z+a)}{(z+b)}$$

Find the values of K_1 and b if a is selected to cancel the non-unity open-loop pole.

Solution

(a) $K = 12.8$

(b) $G(z) = \frac{0.0147(z+0.718)}{(z-1)(z-0.368)}$

(c) $-2.48 \pm j2.56, \quad s^2 + 5s + 12.8 = 0$

$0.531 \pm j0.298, \quad z^2 - 1.062z + 0.371 = 0$

(d) $D(z) = 12.21 \frac{(z-0.368)}{(z-0.242)}$

State-space methods for control system design

8.1 The state-space-approach

The classical control system design techniques discussed in Chapters 5–7 are generally only applicable to

- (a) Single Input, Single Output (SISO) systems
- (b) Systems that are linear (or can be linearized) and are time invariant (have parameters that do not vary with time).

The state-space approach is a generalized time-domain method for modelling, analysing and designing a wide range of control systems and is particularly well suited to digital computational techniques. The approach can deal with

- (a) Multiple Input, Multiple Output (MIMO) systems, or multivariable systems
- (b) Non-linear and time-variant systems
- (c) Alternative controller design approaches.

8.1.1 The concept of state

The state of a system may be defined as: ‘The set of variables (called the state variables) which at some initial time t_0 , together with the input variables completely determine the behaviour of the system for time $t \geq t_0$ ’.

The state variables are the smallest number of states that are required to describe the dynamic nature of the system, and it is not a necessary constraint that they are measurable. The manner in which the state variables change as a function of time may be thought of as a trajectory in n dimensional space, called the *state-space*. Two-dimensional state-space is sometimes referred to as the *phase-plane* when one state is the derivative of the other.

8.1.2 The state vector differential equation

The state of a system is described by a set of first-order differential equations in terms of the state variables (x_1, x_2, \dots, x_n) and input variables (u_1, u_2, \dots, u_m) in the general form

$$\begin{aligned}\frac{dx_1}{dt} &= a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + b_{11}u_1 + \dots + b_{1m}u_m \\ \frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n + b_{21}u_1 + \dots + b_{2m}u_m \\ \frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n + b_{n1}u_1 + \dots + b_{nm}u_m\end{aligned}\quad (8.1)$$

The equations set (8.1) may be combined in matrix format. This results in the state vector differential equation

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \quad (8.2)$$

Equation (8.2) is generally called the state equation(s), where lower-case boldface represents vectors and upper-case boldface represents matrices. Thus

\mathbf{x} is the n dimensional state vector

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (8.3)$$

\mathbf{u} is the m dimensional input vector

$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \quad (8.4)$$

\mathbf{A} is the $n \times n$ system matrix

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad (8.5)$$

\mathbf{B} is the $n \times m$ control matrix

$$\begin{bmatrix} b_{11} & \dots & b_{1m} \\ b_{21} & \dots & b_{2m} \\ \vdots & & \\ b_{n1} & \dots & b_{nm} \end{bmatrix} \quad (8.6)$$

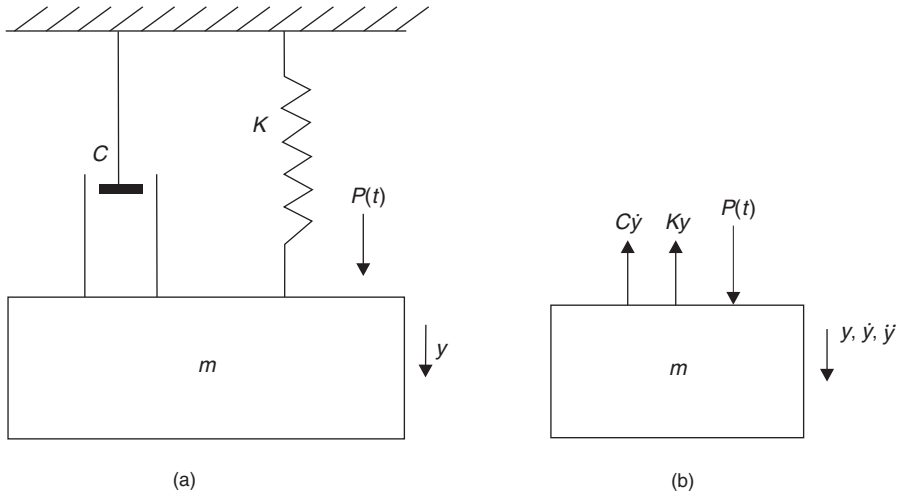


Fig. 8.1 Spring–mass–damper system and free-body diagram.

In general, the outputs (y_1, y_2, \dots, y_n) of a linear system can be related to the state variables and the input variables

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}u \quad (8.7)$$

Equation (8.7) is called the output equation(s).

Example 8.1

Write down the state equation and output equation for the spring–mass–damper system shown in Figure 8.1(a).

Solution

State variables

$$x_1 = y \quad (8.8)$$

$$x_2 = \frac{dy}{dt} = \dot{x}_1 \quad (8.9)$$

Input variable

$$u = P(t) \quad (8.10)$$

Now

$$\sum F_y = m\ddot{y}$$

From Figure 8.1(b)

$$P(t) - Ky - C\dot{y} = m\ddot{y}$$

or

$$\frac{d^2y}{dt^2} = -\frac{K}{m}y - \frac{C}{m}\dot{y} + \frac{1}{m}P(t) \quad (8.11)$$

From equations (8.9), (8.10) and (8.11) the set of first-order differential equations are

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{K}{m}x_1 - \frac{C}{m}x_2 + \frac{1}{m}u\end{aligned}\quad (8.12)$$

and the state equations become

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{m} & -\frac{C}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \quad (8.13)$$

From equation (8.8) the output equation is

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8.14)$$

State variables are not unique, and may be selected to suit the problem being studied.

Example 8.2

For the *RCL* network shown in Figure 8.2, write down the state equations when

- (a) the state variables are $v_2(t)$ and \dot{v}_2
- (b) the state variables are $v_2(t)$ and $i(t)$.

Solution

(a)

$$\begin{aligned}x_1 &= v_2(t) \\ x_2 &= \dot{v}_2 = \dot{x}_1\end{aligned}\quad (8.15)$$

From equation (2.37)

$$LC \frac{d^2 v_2}{dt^2} + RC \frac{dv_2}{dt} + v_2 = v_1(t) \quad (8.16)$$

From equations (8.15) and (8.16) the set of first-order differential equations are

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{1}{LC}x_1 - \frac{RC}{LC}x_2 + \frac{1}{LC}u\end{aligned}\quad (8.17)$$

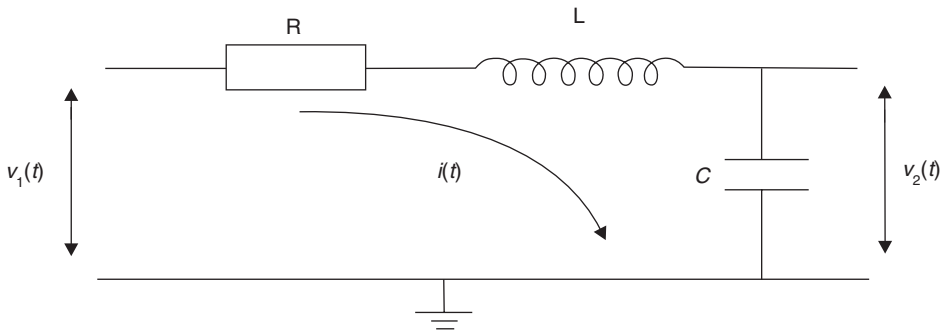


Fig. 8.2 *RCL* network.

and the state equations are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{LC} \end{bmatrix} u \quad (8.18)$$

$$\begin{aligned} (b) \quad & x_1 = v_2(t) \\ & x_2 = i(t) \end{aligned} \quad (8.19)$$

From equations (2.34) and (2.35)

$$L \frac{di}{dt} = -v_2(t) - Ri(t) + v_1(t) \quad (8.20)$$

$$C \frac{dv_2}{dt} = i(t) \quad (8.21)$$

Equations (8.20) and (8.21) are both first-order differential equations, and can be written in the form

$$\begin{aligned} \dot{x}_1 &= \frac{1}{C} x_2 \\ \dot{x}_2 &= -\frac{1}{L} x_1 - \frac{R}{L} x_2 + \frac{1}{L} u \end{aligned} \quad (8.22)$$

giving the state equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u \quad (8.23)$$

Example 8.3

For the 2 mass system shown in Figure 8.3, find the state and output equation when the state variables are the position and velocity of each mass.

Solution

State variables

$$\begin{aligned} x_1 &= y_1 & x_2 &= \dot{y}_1 \\ x_3 &= y_2 & x_4 &= \dot{y}_2 \end{aligned}$$

System outputs

$$y_1, y_2$$

System inputs

$$u = P(t) \quad (8.24)$$

For mass m_1

$$\begin{aligned} \sum F_y &= m_1 \ddot{y}_1 \\ K_2(y_2 - y_1) - K_1 y_1 + P(t) - C_1 \dot{y}_1 &= m_1 \ddot{y}_1 \end{aligned} \quad (8.25)$$

For mass m_2

$$\begin{aligned} \sum F_y &= m_2 \ddot{y}_2 \\ -K_2(y_2 - y_1) &= m_2 \ddot{y}_2 \end{aligned} \quad (8.26)$$

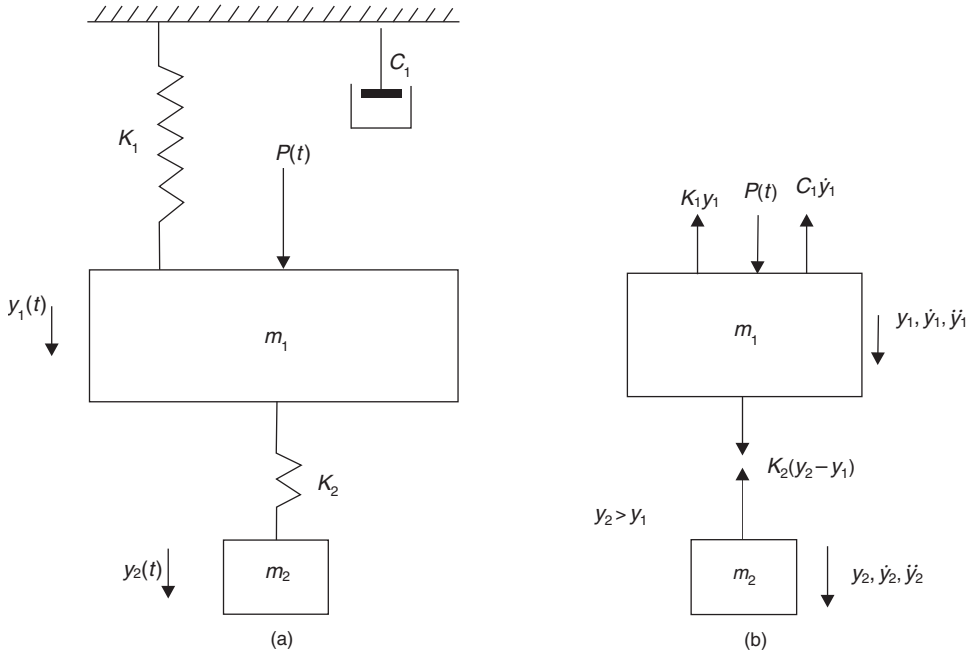


Fig. 8.3 Two-mass system and free-body diagrams.

From (8.24), (8.25) and (8.26), the four first-order differential equations are

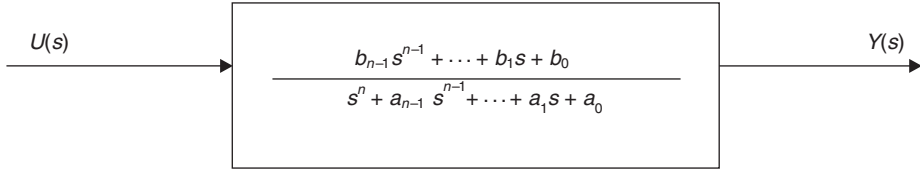
$$\begin{aligned}
 \dot{x}_1 &= x_2 \\
 \dot{x}_2 &= \left(-\frac{K_1}{m_1} - \frac{K_2}{m_1} \right) x_1 - \frac{C_1}{m_1} x_2 + \frac{K_2}{m_1} x_3 + \frac{1}{m_1} u \\
 \dot{x}_3 &= x_4 \\
 \dot{x}_4 &= \frac{K_2}{m_2} x_1 - \frac{K_2}{m_2} x_3
 \end{aligned} \tag{8.27}$$

Hence the state equations are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\left(\frac{K_1 + K_2}{m_1} \right) & -\frac{C_1}{m_1} & \frac{K_2}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{K_2}{m_2} & 0 & -\frac{K_2}{m_2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix} u \tag{8.28}$$

and the output equations are

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{8.29}$$

**Fig. 8.4** Generalized transfer function.

8.1.3 State equations from transfer functions

Consider the general differential equation

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = b_{n-1} \frac{d^{n-1} u}{dt^{n-1}} + \cdots + b_1 \frac{du}{dt} + b_0 u \quad (8.30)$$

Equation (8.30) can be represented by the transfer function shown in Figure 8.4.

Define a set of state variables such that

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= -a_0 x_1 - a_1 x_2 - \cdots - a_{n-1} x_n + u \end{aligned} \quad (8.31)$$

and an output equation

$$y = b_0 x_1 + b_1 x_2 + \cdots + b_{n-1} x_n \quad (8.32)$$

Then the state equation is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \quad (8.33)$$

The state-space representation in equation (8.33) is called the controllable canonical form and the output equation is

$$y = [b_0 \quad b_1 \quad b_2 \quad \cdots \quad b_{n-1}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \quad (8.34)$$

Example 8.4 (See also Appendix 1, *examp84.m*)

Find the state and output equations for

$$\frac{Y}{U}(s) = \frac{4}{s^3 + 3s^2 + 6s + 2}$$

Solution

State equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -6 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \quad (8.35)$$

Output equation

$$y = \begin{bmatrix} 4 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (8.36)$$

Example 8.5

Find the state and output equations for

$$\frac{Y}{U}(s) = \frac{5s^2 + 7s + 4}{s^3 + 3s^2 + 6s + 2}$$

Solution

The state equation is the same as (8.35). The output equation is

$$y = \begin{bmatrix} 4 & 7 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (8.37)$$

8.2 Solution of the state vector differential equation

Consider the first-order differential equation

$$\frac{dx}{dt} = ax(t) + bu(t) \quad (8.38)$$

where $x(t)$ and $u(t)$ are scalar functions of time. Take Laplace transforms

$$sX(s) - x(0) = aX(s) + bU(s) \quad (8.39)$$

where $x(0)$ is the initial condition. From equation (8.39)

$$X(s) = \frac{x(0)}{(s-a)} + \frac{b}{(s-a)} U(s) \quad (8.40)$$

Inverse transform

$$x(t) = e^{at}x(0) + \int_0^t e^{a(t-\tau)}bu(\tau)d\tau \quad (8.41)$$

where the integral term in equation (8.41) is the convolution integral and τ is a dummy time variable. Note that

$$e^{at} = 1 + at + \frac{a^2t^2}{2!} + \cdots + \frac{a^k t^k}{k!} \quad (8.42)$$

Consider now the state vector differential equation

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \quad (8.43)$$

Taking Laplace transforms

$$s\mathbf{X}(s) - \mathbf{x}(0) = \mathbf{AX}(s) + \mathbf{BU}(s) \quad (8.44)$$

$$(s\mathbf{I} - \mathbf{A})\mathbf{X}(s) = \mathbf{x}(0) + \mathbf{BU}(s)$$

Pre-multiplying by $(s\mathbf{I} - \mathbf{A})^{-1}$

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{BU}(s) \quad (8.45)$$

Inverse transform

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{Bu}(\tau)d\tau \quad (8.46)$$

if the initial time is t_0 , then

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}(0) + \int_{t_0}^t e^{\mathbf{A}(t-\tau)}\mathbf{Bu}(\tau)d\tau \quad (8.47)$$

The exponential matrix $e^{\mathbf{A}t}$ in equation (8.46) is called the state-transition matrix $\Phi(t)$ and represents the natural response of the system. Hence

$$\Phi(s) = (s\mathbf{I} - \mathbf{A})^{-1} \quad (8.48)$$

$$\Phi(t) = \mathcal{L}^{-1}(s\mathbf{I} - \mathbf{A})^{-1} = e^{\mathbf{A}t} \quad (8.49)$$

Alternatively

$$\Phi(t) = \mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2 t^2}{2!} + \cdots + \frac{\mathbf{A}^k t^k}{k!} \quad (8.50)$$

Hence equation (8.46) can be written

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) + \int_0^t \Phi(t - \tau)\mathbf{Bu}(\tau)d\tau \quad (8.51)$$

In equation (8.51) the first term represents the response to a set of initial conditions, whilst the integral term represents the response to a forcing function.

Characteristic equation

Using a state variable representation of a system, the characteristic equation is given by

$$|(s\mathbf{I} - \mathbf{A})| = 0 \quad (8.52)$$

8.2.1 Transient solution from a set of initial conditions

Example 8.6

For the spring–mass–damper system given in Example 8.1, Figure 8.1, the state equations are shown in equation (8.13)

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{m} & -\frac{C}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \quad (8.53)$$

Given: $m = 1$ kg, $C = 3$ Ns/m, $K = 2$ N/m, $u(t) = 0$. Evaluate,

- the characteristic equation, its roots, ω_n and ζ
- the transition matrices $\phi(s)$ and $\phi(t)$
- the transient response of the state variables from the set of initial conditions

$$\begin{aligned} y(0) &= 1.0, \\ \dot{y}(0) &= 0 \end{aligned}$$

Solution

Since $x_1 = y$ and $x_2 = \dot{y}$, then $x_1(0) = 1.0$, $x_2(0) = 0$.

Inserting values of system parameters into equation (8.53) gives

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$(a) \quad (s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} s & -1 \\ 2 & (s+3) \end{bmatrix} \quad (8.54)$$

From equation (8.52), the characteristic equation is

$$|(s\mathbf{I} - \mathbf{A})| = s(s+3) - (-2) = s^2 + 3s + 2 = 0 \quad (8.55)$$

Roots of characteristic equation

$$s = -1, -2 \quad (8.56)$$

Compare equation (8.55) with the denominator of the standard form in equation (3.43)

$$\begin{aligned} \omega_n^2 &= 2 \quad \text{i.e.} \quad \omega_n = 1.414 \text{ rad/s} \\ 2\zeta\omega_n &= 3 \quad \text{i.e.} \quad \zeta = 1.061 \end{aligned} \quad (8.57)$$

- The inverse of any matrix \mathbf{A} (see equation A2.17) is

$$\mathbf{A}^{-1} = \frac{\text{Adjoint } \mathbf{A}}{\det \mathbf{A}} \quad (8.58)$$

From equation (8.48)

$$\Phi(s) = (s\mathbf{I} - \mathbf{A})^{-1}$$

Using the standard matrix operations given in Appendix 2, equation (A2.12)

$$\text{Minors of } \Phi(s) = \begin{bmatrix} (s+3) & 2 \\ -1 & s \end{bmatrix}$$

$$\text{Co-factors of } \Phi(s) = \begin{bmatrix} (s+3) & -2 \\ 1 & s \end{bmatrix}$$

The Adjoint matrix is the transpose of the Co-factor matrix

$$\text{Adjoint of } \Phi(s) = \begin{bmatrix} (s+3) & 1 \\ -2 & s \end{bmatrix} \quad (8.59)$$

Hence, from equations (8.58) and (8.48)

$$\Phi(s) = \begin{bmatrix} \frac{(s+3)}{(s+1)(s+2)} & \frac{1}{(s+1)(s+2)} \\ \frac{-2}{(s+1)(s+2)} & \frac{s}{(s+1)(s+2)} \end{bmatrix} \quad (8.60)$$

Using partial fraction expansions

$$\Phi(s) = \begin{bmatrix} \left(\frac{2}{s+1} - \frac{1}{s+2} \right) & \left(\frac{1}{s+1} - \frac{1}{s+2} \right) \\ -2 \left(\frac{1}{s+1} - \frac{1}{s+2} \right) & \left(-\frac{1}{s+1} + \frac{2}{s+2} \right) \end{bmatrix} \quad (8.61)$$

Inverse transform equation (8.61)

$$\Phi(t) = \begin{bmatrix} (2e^{-t} - e^{-2t}) & (e^{-t} - e^{-2t}) \\ -2(e^{-t} - e^{-2t}) & (-e^{-t} + 2e^{-2t}) \end{bmatrix} \quad (8.62)$$

Note that the exponential indices are the roots of the characteristic equation (8.56).

(c) From equation (8.51), the transient response is given by

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) \quad (8.63)$$

Hence

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (2e^{-t} - e^{-2t}) & (e^{-t} - e^{-2t}) \\ -2(e^{-t} - e^{-2t}) & (-e^{-t} + 2e^{-2t}) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (8.64)$$

$$x_1(t) = (2e^{-t} - e^{-2t}) \quad (8.65)$$

$$x_2(t) = -2(e^{-t} - e^{-2t})$$

The time response of the state variables (i.e. position and velocity) together with the state trajectory is given in Figure 8.5.

Example 8.7

For the spring-mass-damper system given in Example 8.6, evaluate the transient response of the state variables to a unit step input using

- (a) The convolution integral
- (b) Inverse Laplace transforms

Assume zero initial conditions.

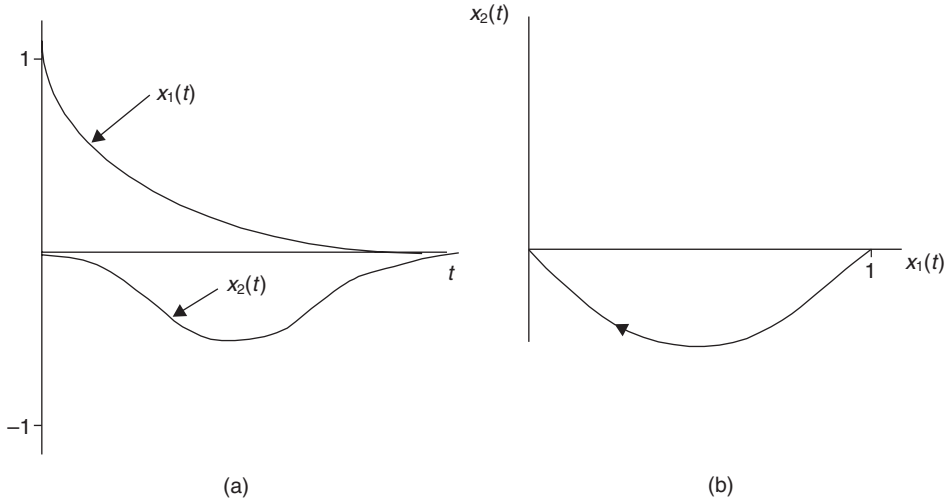


Fig. 8.5 State variable time response and state trajectory for Example 8.4.

Solution

(a) From equation (8.51)

$$\mathbf{x}(t) = \Phi(t) \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \int_0^t \begin{bmatrix} \phi_{11}(t-\tau) & \phi_{12}(t-\tau) \\ \phi_{21}(t-\tau) & \phi_{22}(t-\tau) \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \mathbf{u}(\tau) d\tau \quad (8.66)$$

Given that $u(t) = 1$ and $1/m = 1$, equation (8.66) reduces to

$$\mathbf{x}(t) = \int_0^t \begin{bmatrix} \phi_{12}(t-\tau) \\ \phi_{22}(t-\tau) \end{bmatrix} d\tau$$

Inserting values from equation (8.62)

$$\mathbf{x}(t) = \int_0^t \begin{bmatrix} e^{-(t-\tau)} - e^{-2(t-\tau)} \\ e^{-(t-\tau)} + 2e^{-2(t-\tau)} \end{bmatrix} d\tau \quad (8.67)$$

Integrating

$$\mathbf{x}(t) = \begin{bmatrix} e^{-(t-\tau)} - \frac{1}{2}e^{-2(t-\tau)} \\ e^{-(t-\tau)} + e^{-2(t-\tau)} \end{bmatrix}_0^t \quad (8.68)$$

Inserting integration limits ($\tau = t$ and $\tau = 0$)

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - e^{-t} + \frac{1}{2}e^{-2t} \\ e^{-t} - e^{-2t} \end{bmatrix} \quad (8.69)$$

(b) An alternative method is to inverse transform from an s -domain expression. Equation (8.45) may be written

$$\mathbf{X}(s) = \Phi(s)\mathbf{x}(0) + \Phi(s)\mathbf{B}\mathbf{U}(s) \quad (8.70)$$

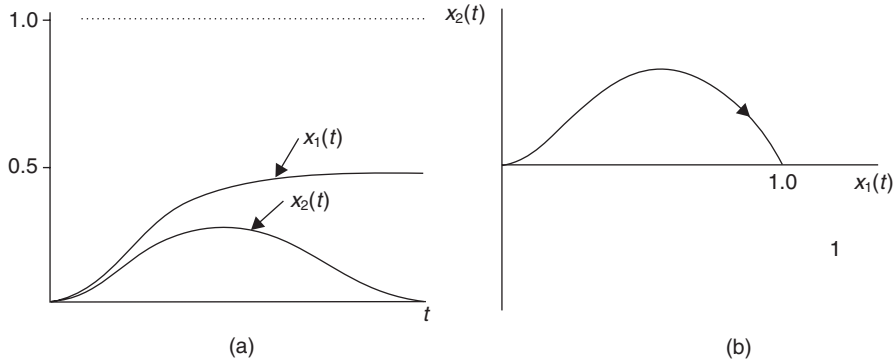


Fig. 8.6 State variable step response and state trajectory for Example 8.5.

Hence from equation (8.61)

$$\mathbf{X}(s) = \Phi(s) \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \left(\frac{2}{s+1} - \frac{1}{s+2} \right) & \left(\frac{1}{s+1} - \frac{1}{s+2} \right) \\ -2 \left(\frac{1}{s+1} - \frac{1}{s+2} \right) & \left(\frac{-1}{s+1} + \frac{2}{s+2} \right) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \frac{1}{s} \quad (8.71)$$

Simplifying

$$\mathbf{X}(s) = \begin{bmatrix} \frac{1}{s(s+1)} - \frac{1}{2} \left\{ \frac{2}{s(s+2)} \right\} \\ \frac{-1}{s(s+1)} + \frac{2}{s(s+2)} \end{bmatrix} \quad (8.72)$$

Inverse transform

$$\mathbf{x}(t) = \begin{bmatrix} (1 - e^{-t}) - \frac{1}{2}(1 - e^{-2t}) \\ -(1 - e^{-t}) + (1 - e^{-2t}) \end{bmatrix} \quad (8.73)$$

which gives

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} - e^{-t} + \frac{1}{2}e^{-2t} \\ e^{-t} - e^{-2t} \end{bmatrix} \quad (8.74)$$

Equation (8.74) is the same as equation (8.69).

The step response of the state variables, together with the state trajectory, is shown in Figure 8.6.

8.3 Discrete-time solution of the state vector differential equation

The discrete-time solution of the state equation may be considered to be the vector equivalent of the scalar difference equation method developed from a z -transform approach in Chapter 7.

The continuous-time solution of the state equation is given in equation (8.47). If the time interval $(t - t_0)$ in this equation is T , the sampling time of a discrete-time system, then the discrete-time solution of the state equation can be written as

$$\mathbf{x}[(k+1)T] = e^{\mathbf{A}T}\mathbf{x}(kT) + \left\{ \int_0^T e^{\mathbf{A}\tau} \mathbf{B} d\tau \right\} \mathbf{u}(kT) \quad (8.75)$$

Equation (8.75) can be written in the general form

$$\mathbf{x}[(k+1)T] = \mathbf{A}(T)\mathbf{x}(kT) + \mathbf{B}(T)\mathbf{u}(kT) \quad (8.76)$$

Note

$$\mathbf{A}(T) \neq \mathbf{A} \quad \text{and} \quad \mathbf{B}(T) \neq \mathbf{B}$$

Equation (8.76) is called the matrix vector difference equation and can be used for the recursive discrete-time simulation of multivariable systems.

The discrete-time state transition matrix $\mathbf{A}(T)$ may be computed by substituting $T = t$ in equations (8.49) and (8.50), i.e.

$$\mathbf{A}(T) = \Phi(T) = e^{\mathbf{A}T} \quad (8.77)$$

or

$$\mathbf{A}(T) = \mathbf{I} + \mathbf{A}T + \frac{\mathbf{A}^2 T^2}{2!} + \cdots + \frac{\mathbf{A}^k T^k}{k!} \quad (8.78)$$

Usually sufficient accuracy is obtained with $5 < k < 50$.

The discrete-time control matrix $\mathbf{B}(T)$ from equations (8.75) and (8.76) is

$$\mathbf{B}(T) = \int_0^T e^{\mathbf{A}\tau} \mathbf{B} d\tau \quad (8.79)$$

or

$$\mathbf{B}(T) = \left\{ \sum_{k=0}^{\infty} \frac{\mathbf{A}^k T^k}{(k+1)!} \right\} \mathbf{B}T$$

Put T within the brackets

$$\mathbf{B}(T) = \left\{ \sum_{k=0}^{\infty} \frac{\mathbf{A}^k T^{k+1}}{(k+1)!} \right\} \mathbf{B}$$

Hence

$$\mathbf{B}(T) = \left\{ \mathbf{I}T + \frac{\mathbf{A}T^2}{2!} + \frac{\mathbf{A}^2 T^3}{3!} + \cdots + \frac{\mathbf{A}^k T^{k+1}}{(k+1)!} \right\} \mathbf{B} \quad (8.80)$$

Example 8.8 (See also Appendix 1, *examp88.m*)

- Calculate the discrete-time transition and control matrices for the spring-mass-damper system in Example 8.6 using a sampling time $T = 0.1$ seconds.
- Using the matrix vector difference equation method, determine the unit step response assuming zero initial conditions.

Solution

(a) The exact value of $\mathbf{A}(T)$ is found by substituting $T = t$ in equation (8.62)

$$\begin{aligned}\mathbf{A}(T) = \Phi(T) &= \begin{bmatrix} (2e^{-0.1} - e^{-0.2})(e^{-0.1} - e^{-0.2}) \\ -2(e^{-0.1} - e^{-0.2})(-e^{-0.1} + 2e^{-0.2}) \end{bmatrix} \\ &= \begin{bmatrix} 0.991 & 0.086 \\ -0.172 & 0.733 \end{bmatrix}\end{aligned}\quad (8.81)$$

An approximate value of $\mathbf{A}(T)$ is found from equation (8.78), taking the series as far as $k = 2$.

$$\mathbf{A}T = \begin{bmatrix} 0 & 0.1 \\ -0.2 & -0.3 \end{bmatrix}$$

$$\frac{\mathbf{A}^2 T^2}{2!} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \frac{0.1^2}{1 \times 2} = \begin{bmatrix} -0.01 & -0.015 \\ 0.03 & 0.035 \end{bmatrix}$$

using the first 3 terms of equation (8.78)

$$\begin{aligned}\mathbf{A}(T) &\approx \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0.1 \\ -0.2 & -0.3 \end{bmatrix} + \begin{bmatrix} -0.01 & -0.015 \\ 0.03 & 0.035 \end{bmatrix} \\ &\approx \begin{bmatrix} 0.99 & 0.085 \\ -0.17 & 0.735 \end{bmatrix}\end{aligned}\quad (8.82)$$

Since in equation (8.66), $u(\tau)$ is unity, the exact value of $\mathbf{B}(T)$ can be obtained by substituting $T = t$ in equation (8.69)

$$\mathbf{B}(T) = \begin{bmatrix} \frac{1}{2} - e^{-0.1} + \frac{1}{2}e^{-0.2} \\ e^{-0.1} - e^{-0.2} \end{bmatrix}\quad (8.83)$$

$$\mathbf{B}(T) = \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix}\quad (8.84)$$

An approximate value of $\mathbf{B}(T)$ is found from equation (8.80), taking the series as far as $k = 2$.

$$\begin{aligned}\mathbf{B}(T) &\approx (\mathbf{I}T)\mathbf{B} + \left(\frac{\mathbf{A}T^2}{2!}\right)\mathbf{B} + \left(\frac{\mathbf{A}^2 T^3}{3!}\right)\mathbf{B} \\ &\approx \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.005 \\ -0.015 \end{bmatrix} + \begin{bmatrix} -0.0005 \\ 0.00117 \end{bmatrix} \\ &\approx \begin{bmatrix} 0.0045 \\ 0.08617 \end{bmatrix}\end{aligned}\quad (8.85)$$

(b) Using the values of $\mathbf{A}(T)$ and $\mathbf{B}(T)$ given in equations (8.81) and (8.84), together with the matrix vector difference equation (8.76), the first few recursive steps of the discrete solution to a step input to the system is given in equation (8.86)

$$kT = 0$$

$$\begin{bmatrix} x_1(0.1) \\ x_2(0.1) \end{bmatrix} = \begin{bmatrix} 0.991 & 0.086 \\ -0.172 & 0.733 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix} 1 = \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix}$$

$$kT = 0.1$$

$$\begin{bmatrix} x_1(0.2) \\ x_2(0.2) \end{bmatrix} = \begin{bmatrix} 0.991 & 0.086 \\ -0.172 & 0.733 \end{bmatrix} \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix} + \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix} 1 = \begin{bmatrix} 0.016 \\ 0.0148 \end{bmatrix}$$

$$kT = 0.2$$

$$\begin{bmatrix} x_1(0.3) \\ x_2(0.3) \end{bmatrix} = \begin{bmatrix} 0.991 & 0.086 \\ -0.172 & 0.733 \end{bmatrix} \begin{bmatrix} 0.016 \\ 0.148 \end{bmatrix} + \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix} 1 = \begin{bmatrix} 0.033 \\ 0.192 \end{bmatrix} \quad (8.86)$$

$$kT = 0.3$$

$$\begin{bmatrix} x_1(0.4) \\ x_2(0.4) \end{bmatrix} = \begin{bmatrix} 0.991 & 0.086 \\ -0.172 & 0.733 \end{bmatrix} \begin{bmatrix} 0.033 \\ 0.192 \end{bmatrix} + \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix} 1 = \begin{bmatrix} 0.054 \\ 0.227 \end{bmatrix}$$

$$kT = 0.4$$

$$\begin{bmatrix} x_1(0.5) \\ x_2(0.5) \end{bmatrix} = \begin{bmatrix} 0.991 & 0.086 \\ -0.172 & 0.733 \end{bmatrix} \begin{bmatrix} 0.054 \\ 0.227 \end{bmatrix} + \begin{bmatrix} 0.00453 \\ 0.0861 \end{bmatrix} 1 = \begin{bmatrix} 0.078 \\ 0.243 \end{bmatrix}$$

Example 8.9

A system has a transfer function

$$\frac{Y}{U}(s) = \frac{1}{s^2 + 2s + 1}$$

The system has an initial condition $y(0) = 1$ and is subject to a unit ramp function $u(t) = t$. Determine

- The state and output equations
- The transition matrix $\Phi(s)$
- Expressions for the time response of the state variables.

Solution

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

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$(b) \quad \Phi(s) = \begin{bmatrix} \frac{s+2}{(s+1)(s+1)} & \frac{1}{(s+1)(s+1)} \\ \frac{-1}{(s+1)(s+1)} & \frac{s}{(s+1)(s+1)} \end{bmatrix}$$

$$(c) \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3e^{-t} + 2te^{-t} - 2 + t \\ 2te^{-t} + 1 - e^{-t} \end{bmatrix}$$

8.4 Control of multivariable systems

8.4.1 Controllability and observability

The concepts of controllability and observability were introduced by Kalman (1960) and play an important role in the control of multivariable systems.

A system is said to be controllable if a control vector $\mathbf{u}(t)$ exists that will transfer the system from any initial state $\mathbf{x}(t_0)$ to some final state $\mathbf{x}(t)$ in a finite time interval.

A system is said to be observable if at time t_0 , the system state $\mathbf{x}(t_0)$ can be exactly determined from observation of the output $\mathbf{y}(t)$ over a finite time interval.

If a system is described by equations (8.2) and (8.7)

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du}\end{aligned}\tag{8.87}$$

then a sufficient condition for complete state controllability is that the $n \times n$ matrix

$$\mathbf{M} = [\mathbf{B} : \mathbf{AB} : \dots : \mathbf{A}^{n-1}\mathbf{B}]\tag{8.88}$$

contains n linearly independent row or column vectors, i.e. is of rank n (that is, the matrix is non-singular, i.e. the determinant is non-zero. See Appendix 2). Equation (8.88) is called the controllability matrix.

The system described by equations (8.87) is completely observable if the $n \times n$ matrix

$$\mathbf{N} = [\mathbf{C}^T : \mathbf{A}^T \mathbf{C}^T : \dots : (\mathbf{A}^T)^{n-1} \mathbf{C}^T]\tag{8.89}$$

is of rank n , i.e. is non-singular having a non-zero determinant. Equation (8.89) is called the observability matrix.

Example 8.10 (See also Appendix 1, *examp810.m*)

Is the following system completely controllable and observable?

$$\begin{aligned}\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} -2 & 0 \\ 3 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u} \\ y &= [1 \quad -1] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\end{aligned}$$

Solution

From equation (8.88) the controllability matrix is

$$\mathbf{M} = [\mathbf{B} : \mathbf{AB}]$$

where

$$\mathbf{AB} = \begin{bmatrix} -2 & 0 \\ 3 & -5 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$$

hence

$$\mathbf{M} = [\mathbf{B} : \mathbf{AB}] = \begin{bmatrix} 0 & -2 \\ 1 & 3 \end{bmatrix}\tag{8.90}$$

Equation (8.90) is non-singular since it has a non-zero determinant. Also the two row and column vectors can be seen to be linearly independent, so it is of rank 2 and therefore the system is controllable.

From equation (8.89) the observability matrix is

$$\mathbf{N} = [\mathbf{C}^T : \mathbf{A}^T \mathbf{C}^T]$$

where

$$\mathbf{A}^T \mathbf{C}^T = \begin{bmatrix} -2 & 3 \\ 0 & -5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \end{bmatrix}$$

hence

$$\mathbf{N} = [\mathbf{C}^T : \mathbf{A}^T \mathbf{C}^T] = \begin{bmatrix} 1 & -5 \\ -1 & 5 \end{bmatrix} \quad (8.91)$$

Equation (8.91) is singular since it has a zero determinant. Also the column vectors are linearly dependent since the second column is -5 times the first column and therefore the system is unobservable.

8.4.2 State variable feedback design

Consider a system described by the state and output equations

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \quad (8.92)$$

Select a control law of the form

$$\mathbf{u} = (\mathbf{r} - \mathbf{K}\mathbf{x}) \quad (8.93)$$

In equation (8.93), $\mathbf{r}(t)$ is a vector of desired state variables and \mathbf{K} is referred to as the state feedback gain matrix. Equations (8.92) and (8.93) are represented in state variable block diagram form in Figure 8.7.

Substituting equation (8.93) into equation (8.92) gives

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}(\mathbf{r} - \mathbf{K}\mathbf{x})$$

or

$$\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{Br} \quad (8.94)$$

In equation (8.94) the matrix $(\mathbf{A} - \mathbf{BK})$ is the closed-loop system matrix.

For the system described by equation (8.92), and using equation (8.52), the characteristic equation is given by

$$|(s\mathbf{I} - \mathbf{A})| = 0 \quad (8.95)$$

The roots of equation (8.95) are the open-loop poles or eigenvalues. For the closed-loop system described by equation (8.94), the characteristic equation is

$$|(s\mathbf{I} - \mathbf{A} + \mathbf{BK})| = 0 \quad (8.96)$$

The roots of equation (8.96) are the closed-loop poles or eigenvalues.

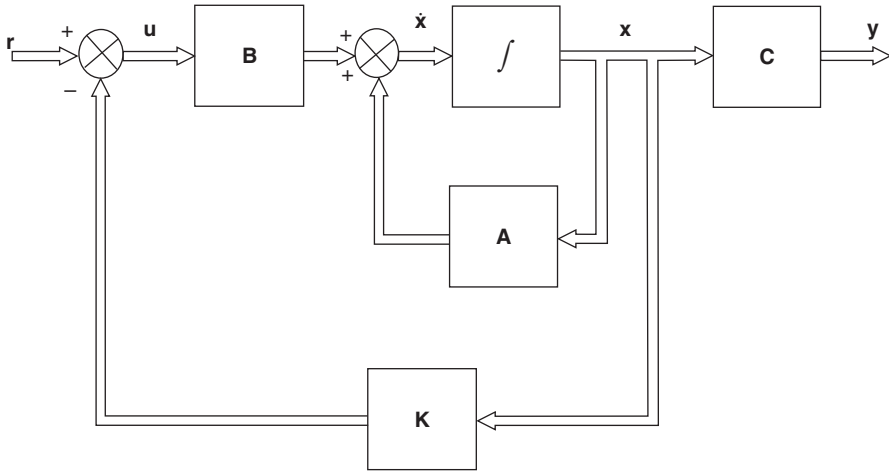


Fig. 8.7 Control using state variable feedback.

Regulator design by pole placement

The pole placement control problem is to determine a value of \mathbf{K} that will produce a desired set of closed-loop poles. With a regulator, $\mathbf{r}(t) = \mathbf{0}$ and therefore equation (8.93) becomes

$$\mathbf{u} = -\mathbf{K}\mathbf{x}$$

Thus the control $\mathbf{u}(t)$ will drive the system from a set of initial conditions $\mathbf{x}(0)$ to a set of zero states at time t_1 , i.e. $\mathbf{x}(t_1) = \mathbf{0}$.

There are several methods that can be used for pole placement.

- (a) *Direct comparison method:* If the desired locations of the closed-loop poles (eigenvalues) are

$$s = \mu_1, s = \mu_2, \dots, s = \mu_n \quad (8.97)$$

then, from equation (8.96)

$$|s\mathbf{I} - \mathbf{A} + \mathbf{BK}| = (s - \mu_1)(s - \mu_2) \dots (s - \mu_n) \quad (8.98)$$

$$= s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0 \quad (8.99)$$

Solving equation (8.99) will give the elements of the state feedback matrix.

- (b) *Controllable canonical form method:* The value of \mathbf{K} can be calculated directly using

$$\mathbf{k} = [\alpha_0 - a_0 : \alpha_1 - a_1 : \dots : \alpha_{n-2} - a_{n-2} : \alpha_{n-1} - a_{n-1}] \mathbf{T}^{-1} \quad (8.100)$$

where \mathbf{T} is a transformation matrix that transforms the system state equation into the controllable canonical form (see equation (8.33)).

$$\mathbf{T} = \mathbf{MW} \quad (8.101)$$

where \mathbf{M} is the controllability matrix, equation (8.88)

$$\mathbf{W} = \begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & 1 \\ a_2 & a_3 & \dots & 1 & 0 \\ \vdots & & & & \\ a_{n-1} & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix} \quad (8.102)$$

Note that $\mathbf{T} = \mathbf{I}$ when the system state equation is already in the controllable canonical form.

- (c) *Ackermann's formula*: As with Method 2, Ackermann's formula (1972) is a direct evaluation method. It is only applicable to SISO systems and therefore $u(t)$ and $y(t)$ in equation (8.87) are scalar quantities. Let

$$\mathbf{K} = [0 \quad 0 \quad \dots \quad 0 \quad 1] \mathbf{M}^{-1} \phi(\mathbf{A}) \quad (8.103)$$

where \mathbf{M} is the controllability matrix and

$$\phi(\mathbf{A}) = \mathbf{A}^n + \alpha_{n-1} \mathbf{A}^{n-1} + \dots + \alpha_1 \mathbf{A} + \alpha_0 \mathbf{I} \quad (8.104)$$

where \mathbf{A} is the system matrix and α_i are the coefficients of the desired closed-loop characteristic equation.

Example 8.11 (See also Appendix 1, *examp811.m*)

A control system has an open-loop transfer function

$$\frac{Y}{U}(s) = \frac{1}{s(s+4)}$$

When $x_1 = y$ and $x_2 = \dot{x}_1$, express the state equation in the controllable canonical form.

Evaluate the coefficients of the state feedback gain matrix using:

- The direct comparison method
- The controllable canonical form method
- Ackermann's formula

such that the closed-loop poles have the values

$$s = -2, s = -2$$

Solution

From the open-loop transfer function

$$\ddot{y} + 4\dot{y} = u \quad (8.105)$$

Let

$$x_1 = y \quad (8.106)$$

Then

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -4x_2 + u \end{aligned} \quad (8.107)$$

Equation (8.106) provides the output equation and (8.107) the state equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \quad (8.108)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8.109)$$

The characteristic equation for the open-loop system is

$$\begin{aligned} |s\mathbf{I} - \mathbf{A}| &= \left| \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & -4 \end{bmatrix} \right| = s^2 + 4s + 0 \\ &= s^2 + a_1s + a_0 \end{aligned} \quad (8.110)$$

Thus

$$a_1 = 4, \quad a_0 = 0$$

The required closed-loop characteristic equation is

$$(s + 2)(s + 2) = 0$$

or

$$s^2 + 4s + 4 = 0 \quad (8.111)$$

i.e.

$$s^2 + \alpha_1s + \alpha_0 = 0 \quad (8.112)$$

hence

$$\alpha_1 = 4, \quad \alpha_0 = 4$$

(a) *Direct comparison method:* From equations (8.99) and (8.111)

$$|s\mathbf{I} - \mathbf{A} + \mathbf{BK}| = s^2 + 4s + 4 \quad (8.113)$$

$$\begin{aligned} \left| \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & -4 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} [k_1 k_2] \right| &= s^2 + 4s + 4 \\ \left| \begin{bmatrix} s & -1 \\ 0 & s + 4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ k_1 & k_2 \end{bmatrix} \right| &= s^2 + 4s + 4 \\ \left| \begin{bmatrix} s & -1 \\ k_1 & s + 4 + k_2 \end{bmatrix} \right| &= s^2 + 4s + 4 \\ s^2 + (4 + k_2)s + k_1 &= s^2 + 4s + 4 \end{aligned} \quad (8.114)$$

From equation (8.114)

$$\begin{aligned} k_1 &= 4 \\ (4 + k_2) &= 4 \quad \text{i.e.} \quad k_2 = 0 \end{aligned} \quad (8.115)$$

(b) *Controllable canonical form method*: From equation (8.100)

$$\begin{aligned}\mathbf{K} &= [\alpha_0 - a_0 : \alpha_1 - a_1] \mathbf{T}^{-1} \\ &= [4 - 0 : 4 - 4] \mathbf{T}^{-1} \\ &= [4 \quad 0] \mathbf{T}^{-1}\end{aligned}\quad (8.116)$$

now

$$\mathbf{T} = \mathbf{M}\mathbf{W}$$

where

$$\mathbf{M} = [\mathbf{B} : \mathbf{A}\mathbf{B}]$$

$$\mathbf{A}\mathbf{B} = \begin{bmatrix} 0 & 1 \\ 0 & -4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$$

giving

$$\mathbf{M} = \begin{bmatrix} 0 & 1 \\ 1 & -4 \end{bmatrix}\quad (8.117)$$

Note that the determinant of \mathbf{M} is non-zero, hence the system is controllable.

From equation (8.102)

$$\mathbf{W} = \begin{bmatrix} a_1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 1 & 0 \end{bmatrix}$$

Hence

$$\mathbf{T} = \mathbf{M}\mathbf{W} = \begin{bmatrix} 0 & 1 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} 4 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}\quad (8.118)$$

Thus proving that equation (8.108) is already in the controllable canonical form. Since \mathbf{T}^{-1} is also \mathbf{I} , substitute (8.118) into (8.116)

$$\mathbf{K} = [4 \quad 0] \mathbf{I} = [4 \quad 0]\quad (8.119)$$

(c) *Ackermann's formula*: From (8.103)

$$\mathbf{K} = [0 \quad 1] \mathbf{M}^{-1} \phi(\mathbf{A})\quad (8.120)$$

From (8.117)

$$\mathbf{M}^{-1} = \frac{1}{-1} \begin{bmatrix} -4 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 1 & 0 \end{bmatrix}\quad (8.121)$$

From (8.104)

$$\phi(\mathbf{A}) = \mathbf{A}^2 + \alpha_1 \mathbf{A} + \alpha_0 \mathbf{I}$$

inserting values

$$\begin{aligned}
 \phi(\mathbf{A}) &= \begin{bmatrix} 0 & 1 \\ 0 & -4 \end{bmatrix}^2 + 4 \begin{bmatrix} 0 & 1 \\ 0 & -4 \end{bmatrix} + 4 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 0 & -4 \\ 0 & 16 \end{bmatrix} + \begin{bmatrix} 0 & 4 \\ 0 & -16 \end{bmatrix} + \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \\
 &= \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}
 \end{aligned} \tag{8.122}$$

Insert equations (8.121) and (8.122) into (8.120)

$$\begin{aligned}
 \mathbf{K} &= [0 \quad 1] \begin{bmatrix} 4 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \\
 &= [0 \quad 1] \begin{bmatrix} 16 & 4 \\ 4 & 0 \end{bmatrix} \\
 \mathbf{K} &= [4 \quad 0]
 \end{aligned} \tag{8.123}$$

These results agree with the root locus diagram in Figure 5.9, where $K = 4$ produces two real roots of $s = -2$, $s = -2$ (i.e. critical damping).

8.4.3 State observers

In section 8.4.2 where state feedback design was discussed, it was assumed that all the state variables were available for the control equation (8.93) for a regulator

$$\mathbf{u} = (\mathbf{r} - \mathbf{K}\mathbf{x})$$

when $\mathbf{r} = \mathbf{0}$

$$\mathbf{u} = -\mathbf{K}\mathbf{x} \tag{8.124}$$

Equations (8.124) requires that all state variables must be measured. In practice this may not happen for a number of reasons including cost, or that the state may not physically be measurable. Under these conditions it becomes necessary, if full state feedback is required, to observe, or estimate the state variables.

A full-order state observer estimates all of the system state variables. If, however, some of the state variables are measured, it may only be necessary to estimate a few of them. This is referred to as a reduced-order state observer. All observers use some form of mathematical model to produce an estimate $\hat{\mathbf{x}}$ of the actual state vector \mathbf{x} . Figure 8.8 shows a simple arrangement of a full-order state observer.

In Figure 8.8, since the observer dynamics will never exactly equal the system dynamics, this open-loop arrangement means that \mathbf{x} and $\hat{\mathbf{x}}$ will gradually diverge. If however, an output vector $\hat{\mathbf{y}}$ is estimated and subtracted from the actual output vector \mathbf{y} , the difference can be used, in a closed-loop sense, to modify the dynamics of the observer so that the output error ($\mathbf{y} - \hat{\mathbf{y}}$) is minimized. This arrangement, sometimes called a Luenberger observer (1964), is shown in Figure 8.9.

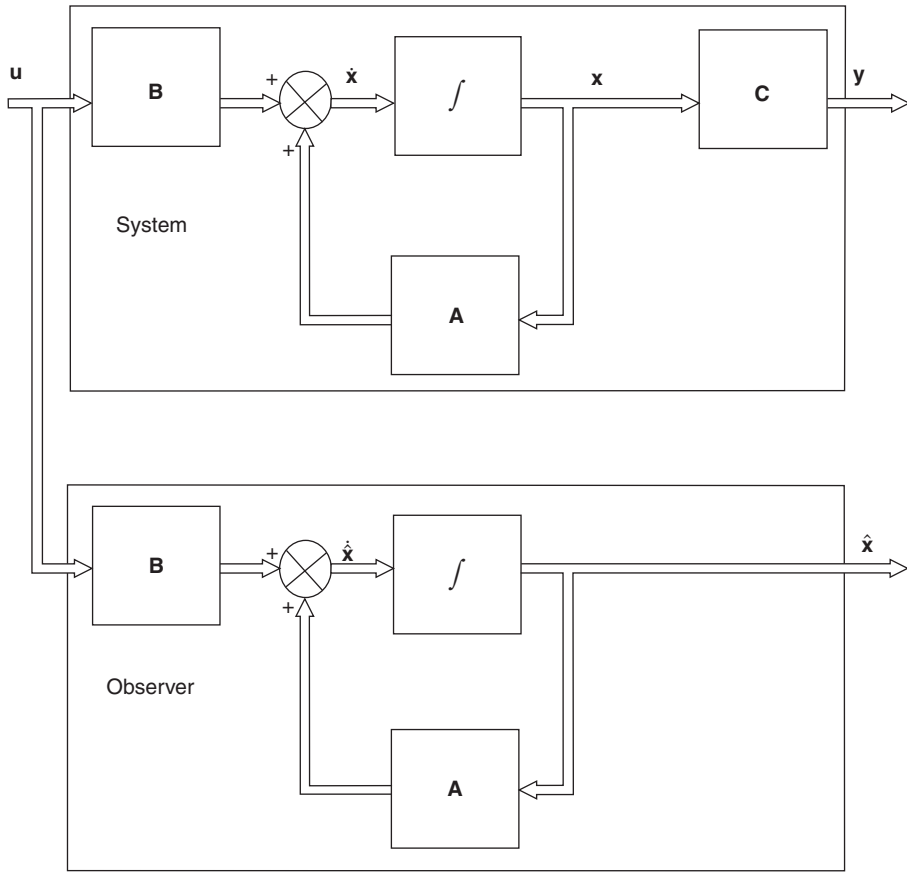


Fig. 8.8 A simple full-order state observer.

Let the system in Figure 8.9 be defined by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (8.125)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \quad (8.126)$$

Assume that the estimate $\hat{\mathbf{x}}$ of the state vector is

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{K}_e(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \quad (8.127)$$

where \mathbf{K}_e is the observer gain matrix.

If equation (8.127) is subtracted from (8.125), and $(\mathbf{x} - \hat{\mathbf{x}})$ is the error vector \mathbf{e} , then

$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\mathbf{e} \quad (8.128)$$

and, from equation (8.127), the equation for the full-order state observer is

$$\dot{\hat{\mathbf{x}}} = (\mathbf{A} - \mathbf{K}_e\mathbf{C})\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{K}_e\mathbf{y} \quad (8.129)$$

Thus from equation (8.128) the dynamic behaviour of the error vector depends upon the eigenvalues of $(\mathbf{A} - \mathbf{K}_e\mathbf{C})$. As with any measurement system, these eigenvalues

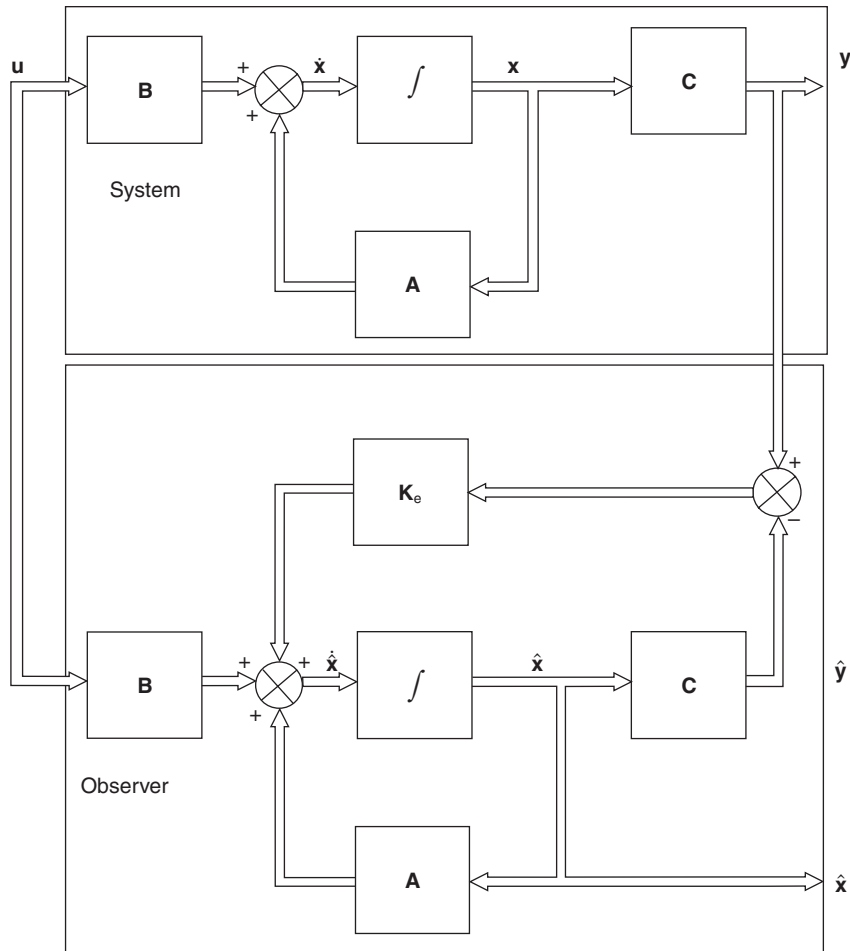


Fig. 8.9 The Luenberger full-order state observer.

should allow the observer transient response to be more rapid than the system itself (typically a factor of 5), unless a filtering effect is required.

The problem of observer design is essentially the same as the regulator pole placement problem, and similar techniques may be used.

- (a) *Direct comparison method:* If the desired locations of the closed-loop poles (eigenvalues) of the observer are

$$s = \mu_1, s = \mu_2, \dots, s = \mu_n$$

then

$$\begin{aligned} |s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C}| &= (s - \mu_1)(s - \mu_2) \dots (s - \mu_n) \\ &= s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0 \end{aligned} \quad (8.130)$$

- (b) *Observable canonical form method:* For the generalized transfer function shown in Figure 8.4, the observable form of the state equation may be written

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & \dots & 0 & -a_1 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix} u$$

$$y = [0 \quad 0 \quad \dots \quad 0 \quad 1] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (8.131)$$

Note that the system matrix of the observable canonical form is the transpose of the controllable canonical form given in equation (8.33).

The value of the observer gain matrix \mathbf{K}_e can be calculated directly using

$$\mathbf{K}_e = \mathbf{Q} \begin{bmatrix} \alpha_0 - a_0 \\ \alpha_1 - a_1 \\ \vdots \\ \alpha_{n-1} - a_{n-1} \end{bmatrix} \quad (8.132)$$

\mathbf{Q} is a transformation matrix that transforms the system state equation into the observable canonical form

$$\mathbf{Q} = (\mathbf{W}\mathbf{N}^T)^{-1} \quad (8.133)$$

where \mathbf{W} is defined in equation (8.102) and \mathbf{N} is the observability matrix given in equation (8.89). If the equation is in the observable canonical form then $\mathbf{Q} = \mathbf{I}$.

- (c) *Ackermann's formula:* As with regulator design, this is only applicable to systems where $u(t)$ and $y(t)$ are scalar quantities. It may be used to calculate the observer gain matrix as follows

$$\mathbf{K}_e = \phi(\mathbf{A})\mathbf{N}^{-1}[0 \quad 0 \quad \dots \quad 0 \quad 1]^T$$

or alternatively

$$\mathbf{K}_e = \phi(\mathbf{A}) \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad (8.134)$$

where $\phi(\mathbf{A})$ is defined in equation (8.104).

Example 8.12 (See also Appendix 1, *examp812.m*)

A system is described by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Design a full-order observer that has an undamped natural frequency of 10 rad/s and a damping ratio of 0.5.

Solution

From equation (8.89), the observability matrix is

$$\mathbf{N} = [\mathbf{C}^T : \mathbf{A}^T \mathbf{C}^T] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (8.135)$$

\mathbf{N} is of rank 2 and therefore non-singular, hence the system is completely observable and the calculation of an appropriate observer gain matrix \mathbf{K}_e realizable.

Open-loop eigenvalues:

$$|s\mathbf{I} - \mathbf{A}| = s^2 + 3s + 2 = s^2 + a_1s + a_0 \quad (8.136)$$

Hence

$$a_0 = 2, \quad a_1 = 3$$

And the open-loop eigenvalues are

$$\begin{aligned} s^2 + 3s + 2 &= 0 \\ (s + 1)(s + 2) &= 0 \\ s &= -1, \quad s = -2 \end{aligned} \quad (8.137)$$

Desired closed-loop eigenvalues:

$$\begin{aligned} s^2 + 2\zeta\omega_n s + \omega_n^2 &= 0 \\ s^2 + 10s + 100 &= s^2 + \alpha_1s + \alpha_0 = 0 \end{aligned} \quad (8.138)$$

Hence

$$\alpha_0 = 100, \quad \alpha_1 = 10$$

and the desired closed-loop eigenvalues are the roots of equation (8.138)

$$\mu_1 = -5 + j8.66, \quad \mu_2 = -5 - j8.66 \quad (8.139)$$

(a) *Direct comparison method:* From equation (8.130)

$$\begin{aligned} |s\mathbf{I} - \mathbf{A} + \mathbf{K}_e\mathbf{C}| &= s^2 + \alpha_1s + \alpha_0 \\ \left| \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} + \begin{bmatrix} k_{e1} \\ k_{e2} \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right| &= s^2 + 10s + 100 \\ \left| \begin{bmatrix} s & -1 \\ 2 & s+3 \end{bmatrix} + \begin{bmatrix} k_{e1} & 0 \\ k_{e2} & 0 \end{bmatrix} \right| &= s^2 + 10s + 100 \\ \left| \begin{bmatrix} s+k_{e1} & -1 \\ 2+k_{e2} & s+3 \end{bmatrix} \right| &= s^2 + 10s + 100 \\ s^2 + (3+k_{e1})s + (3k_{e1} + 2 + k_{e2}) &= s^2 + 10s + 100 \end{aligned} \quad (8.140)$$

From equation (8.140)

$$(3 + k_{e1}) = 10, \quad k_{e1} = 7 \quad (8.141)$$

$$\begin{aligned} (3k_{e1} + 2 + k_{e2}) &= 100 \\ k_{e2} &= 100 - 2 - 21 = 77 \end{aligned} \quad (8.142)$$

(b) *Observable canonical form method*: From equation (8.132)

$$\begin{aligned} \mathbf{K}_e &= \mathbf{Q} \begin{bmatrix} \alpha_0 - a_0 \\ \alpha_1 - a_1 \end{bmatrix} \\ &= \mathbf{Q} \begin{bmatrix} 100 - 2 \\ 10 - 3 \end{bmatrix} \\ &= \mathbf{Q} \begin{bmatrix} 98 \\ 7 \end{bmatrix} \end{aligned} \quad (8.143)$$

From equation (8.133)

$$\mathbf{Q} = (\mathbf{W}\mathbf{N}^T)^{-1}$$

and from equation (8.102)

$$\mathbf{W} = \begin{bmatrix} a_1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \quad (8.144)$$

Since from equation (8.135)

$$\mathbf{N} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{N}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (8.145)$$

Thus

$$\mathbf{W}\mathbf{N}^T = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \quad (8.146)$$

and

$$\mathbf{Q} = \frac{1}{-1} \begin{bmatrix} 0 & -1 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -3 \end{bmatrix} \quad (8.147)$$

Since $\mathbf{Q} \neq \mathbf{I}$ then \mathbf{A} is not in the observable canonical form.

From equation (8.143)

$$\mathbf{K}_e = \begin{bmatrix} 0 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 98 \\ 7 \end{bmatrix} = \begin{bmatrix} 7 \\ 77 \end{bmatrix} \quad (8.148)$$

(c) *Ackermann's Formula*: From (8.134)

$$\mathbf{K}_e = \phi(\mathbf{A}) \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Using the definition of $\phi(\mathbf{A})$ in equation (8.104)

$$\mathbf{K}_e = (\mathbf{A}^2 + \alpha_1 \mathbf{A} + \alpha_0 \mathbf{I}) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (8.149)$$

$$\begin{aligned} \mathbf{K}_e &= \begin{bmatrix} -2 & -3 \\ 6 & 7 \end{bmatrix} + \begin{bmatrix} 0 & 10 \\ -20 & -30 \end{bmatrix} + \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 98 & 7 \\ 14 & 77 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 98 & 7 \\ 14 & 77 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 77 \end{bmatrix} \end{aligned} \quad (8.150)$$

8.4.4 Effect of a full-order state observer on a closed-loop system

Figure 8.10 shows a closed-loop system that includes a full-order state observer. In Figure 8.10 the system equations are

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} \end{aligned} \quad (8.151)$$

The control is implemented using observed state variables

$$\mathbf{u} = -\mathbf{K}\hat{\mathbf{x}} \quad (8.152)$$

If the difference between the actual and observed state variables is

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$$

then

$$\hat{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{e}(t) \quad (8.153)$$

Combining equations (8.151), (8.152) and (8.153) gives the closed-loop equations

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{Ax} - \mathbf{BK}(\mathbf{x} - \mathbf{e}) \\ &= (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{BKe} \end{aligned} \quad (8.154)$$

The observer error equation from equation (8.128) is

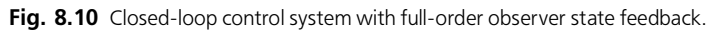
$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{K}_e \mathbf{C})\mathbf{e} \quad (8.155)$$

Combining equations (8.154) and (8.155) gives

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK} \\ \mathbf{0} & \mathbf{A} - \mathbf{B}_e \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{e} \end{bmatrix} \quad (8.156)$$

Equation (8.156) describes the closed-loop dynamics of the observed state feedback control system and the characteristic equation is therefore

$$|s\mathbf{I}\mathbf{A} + \mathbf{BK}||s\mathbf{I} - \mathbf{A} + \mathbf{K}_e \mathbf{C}| = 0 \quad (8.157)$$



Using the state vectors $\mathbf{x}(t)$ and $\hat{\mathbf{x}}(t)$ the state equations for the closed-loop system are

$$\dot{\mathbf{x}} = \mathbf{Ax} - \mathbf{BK}\hat{\mathbf{x}} \quad (8.158)$$
$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= (\mathbf{A} - \mathbf{K}_e \mathbf{C})\hat{\mathbf{x}} - \mathbf{B}\mathbf{K}\hat{\mathbf{x}} + \mathbf{K}_e \mathbf{C}\mathbf{x} \\ &= (\mathbf{A} - \mathbf{K}_e \mathbf{C} - \mathbf{B}\mathbf{K})\hat{\mathbf{x}} + \mathbf{K}_e \mathbf{C}\mathbf{x}\end{aligned}\quad (8.159)$$
$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\hat{\mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & -\mathbf{BK} \\ \mathbf{K}_e\mathbf{C} & \mathbf{A} - \mathbf{K}_e\mathbf{C} - \mathbf{BK} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \hat{\mathbf{x}} \end{bmatrix} \quad (8.160)$$

8.4.5 Reduced-order state observers

A full-order state observer estimates all state variables, irrespective of whether they are being measured. In practice, it would appear logical to use a combination of measured states from $\mathbf{y} = \mathbf{C}\mathbf{x}$ and observed states (for those state variables that are either not being measured, or not being measured with sufficient accuracy).

If the state vector is of n th order and the measured output vector is of m th order, then it is only necessary to design an $(n - m)$ th order state observer.

Consider the case of the measurement of a single state variable $x_1(t)$. The output equation is therefore

$$\mathbf{y} = x_1 = \mathbf{C}\mathbf{x} = [1 \ 0 \ \dots \ 0]\mathbf{x} \quad (8.161)$$

Partition the state vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \mathbf{x}_e \end{bmatrix} \quad (8.162)$$

where \mathbf{x}_e are the state variables to be observed.

Partition the state equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{\mathbf{x}}_e \end{bmatrix} = \begin{bmatrix} a_{11} & \mathbf{A}_{1e} \\ \mathbf{A}_{e1} & \mathbf{A}_{ee} \end{bmatrix} \begin{bmatrix} x_1 \\ \mathbf{x}_e \end{bmatrix} + \begin{bmatrix} b_1 \\ \mathbf{B}_e \end{bmatrix} u \quad (8.163)$$

If the desired eigenvalues for the reduced-order observer are

$$s = \mu_{1e}, s = \mu_{2e}, \dots, s = \mu_{(n-1)e}$$

Then it can be shown that the characteristic equation for the reduced-order observer is

$$\begin{aligned} |s\mathbf{I} - \mathbf{A}_{ee} + \mathbf{K}_e\mathbf{A}_{1e}| &= (s - \mu_{1e}) \dots (s - \mu_{(n-1)e}) \\ &= s^{n-1} + \alpha_{(n-2)e}s^{n-2} + \dots + \alpha_{1e}s + \alpha_{0e} \end{aligned} \quad (8.164)$$

In equation (8.164) \mathbf{A}_{ee} replaces \mathbf{A} and \mathbf{A}_{1e} replaces \mathbf{C} in the full-order observer.

The reduced-order observer gain matrix \mathbf{K}_e can also be obtained using appropriate substitutions into equations mentioned earlier. For example, equation (8.132) becomes

$$\mathbf{K}_e = \mathbf{Q}_e \begin{bmatrix} \alpha_{0e} - a_{0e} \\ \alpha_{1e} - a_{1e} \\ \vdots \\ \alpha_{(n-2)e} - a_{(n-2)e} \end{bmatrix} \quad (8.165)$$

where $a_{0e}, \dots, a_{(n-2)e}$ are the coefficients of the open-loop reduced order characteristics equation

$$|s\mathbf{I} - \mathbf{A}_{ee}| = s^{n-1} + a_{(n-2)e}s^{n-2} + a_{1e}s + a_{0e} \quad (8.166)$$

and

$$\mathbf{Q}_e = (\mathbf{W}_e\mathbf{N}_e^T)^{-1} \quad (8.167)$$

where

$$\mathbf{W}_e = \begin{bmatrix} a_1 & a_2 & \dots & a_{n-2} & 1 \\ a_2 & a_3 & \dots & 1 & 0 \\ \vdots & & & & \\ a_{n-2} & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix} \quad (8.168)$$

and

$$\mathbf{N}_e = [\mathbf{A}_{1e}^T : \mathbf{A}_{ee}^T \mathbf{T} \mathbf{A}_{1e}^T : \dots : (\mathbf{A}_{ee}^T \mathbf{T})^{n-2} \mathbf{A}_{1e}^T] \quad (8.169)$$

and Ackermann's formula becomes

$$\mathbf{K}_e = \phi(\mathbf{A}_{ee}) \begin{bmatrix} \mathbf{A}_{1e} \\ \mathbf{A}_{1e} \mathbf{A}_{ee} \\ \vdots \\ \mathbf{A}_{1e} \mathbf{A}_{ee}^{n-3} \\ \mathbf{A}_{1e} \mathbf{A}_{ee}^{n-2} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad (8.170)$$

where

$$\phi(\mathbf{A}_{ee}) = \mathbf{A}_{ee}^{n-1} + \alpha_{n-2} \mathbf{A}_{ee}^{n-2} + \dots + \alpha_2 \mathbf{A}_{ee} + \alpha_1 \mathbf{I} \quad (8.171)$$

Define

$$\dot{\hat{\mathbf{x}}}_{e1} = \hat{\mathbf{x}} - \mathbf{K}_e y$$

Then

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}_{e1} - \mathbf{K}_e y \quad (8.172)$$

The equation for the reduced-order observer can be shown to be

$$\dot{\hat{\mathbf{x}}}_{e1} = (\mathbf{A}_{ee} \mathbf{K}_e \mathbf{A}_{1e}) \hat{\mathbf{x}}_{e1} + \{\mathbf{A}_{e1} \mathbf{K}_e a_{11} + (\mathbf{A}_{ee} - \mathbf{K}_e \mathbf{A}_{1e}) \mathbf{K}_e\} y + (\mathbf{B}_e - \mathbf{K}_e b_1) u \quad (8.173)$$

Figure 8.11 shows the implementation of a reduced-order state observer.

Case study

Example 8.13 (See also Appendix 1, *examp813.m*)

(a) In case study Example 5.10 a control system has an open-loop transfer function

$$G(s)H(s) = \frac{1}{s(s+2)(s+5)}$$

The controller was a PD compensator of the form

$$G(s) = K_1(s+a)$$

With $K_1 = 15$ and $a = 1$, the system closed-loop poles were

$$s = -3.132 \pm j3.253$$

$$s = -0.736$$

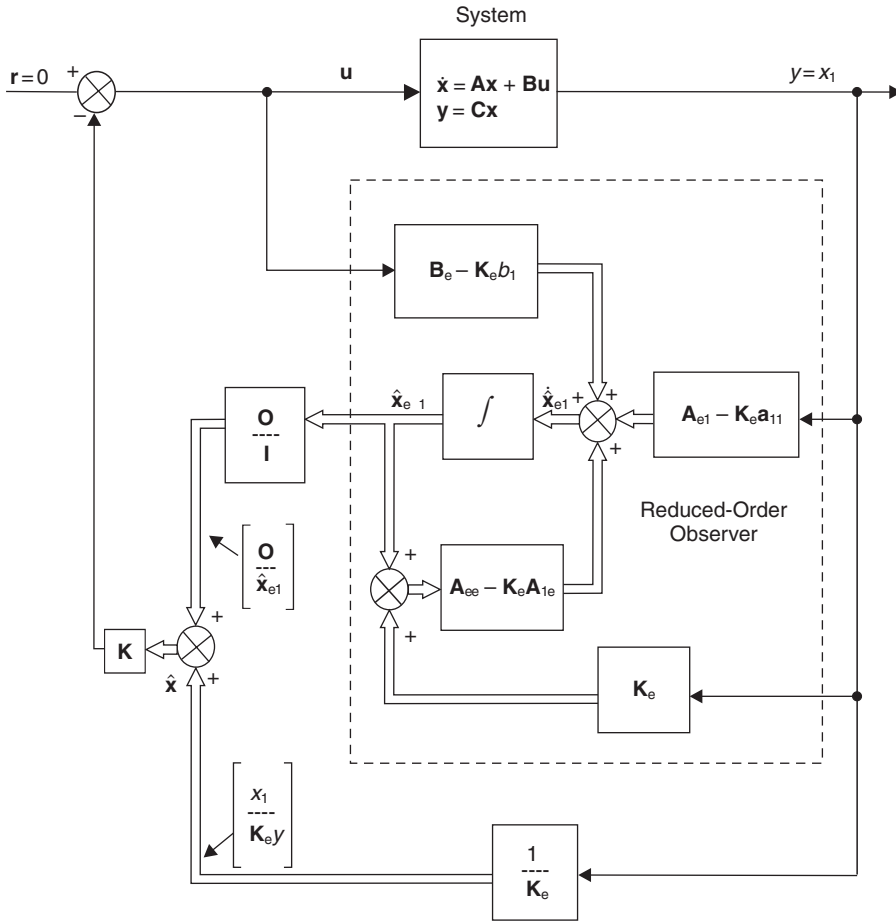


Fig. 8.11 Implementation of a reduced-order state observer.

with the resulting characteristic equation

$$s^3 + 7s^2 + 25s + 15 = 0$$

Demonstrate that the same result can be achieved using state feedback methods.

- (b) Design a reduced second-order state observer for the system such that the poles are a factor of 10 higher than the closed-loop system poles, i.e.

$$s = -31.32 \pm j32.53$$

which correspond to $\omega_n = 45.16$ rad/s and $\zeta = 0.7$.

Solution

(a)
$$G(s)H(s) = \frac{1}{s^3 + 7s^2 + 10s + 0}$$

From equations (8.33) and (8.34)

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -10 & -7 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{C} = [1 \quad 0 \quad 0] \quad (8.174)$$

Open-loop characteristic equation

$$\begin{aligned} s^3 + 7s^2 + 10s + 0 &= 0 \\ s^3 + a_2s^2 + a_1s + a_0 &= 0 \end{aligned} \quad (8.175)$$

Closed-loop characteristic equation

$$\begin{aligned} s^3 + 7s^2 + 25s + 15 &= 0 \\ s^3 + \alpha_2s^2 + \alpha_1s + \alpha_0 &= 0 \end{aligned} \quad (8.176)$$

Using direct comparison method

$$\begin{aligned} |s\mathbf{I} - \mathbf{A} + \mathbf{BK}| &= s^3 + 7s^2 + 25s + 15 \\ \left| \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -10 & -7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} [k_1 \quad k_2 \quad k_3] \right| &= s^3 + 7s^2 + 25s + 15 \\ \left| \begin{bmatrix} s & -1 & 0 \\ 0 & s & -1 \\ 0 & 10 & s+7 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ k_1 & k_2 & k_3 \end{bmatrix} \right| &= s^3 + 7s^2 + 25s + 15 \\ \left| \begin{bmatrix} s & -1 & 0 \\ 0 & s & -1 \\ k_1 & 10+k_2 & s+7+k_3 \end{bmatrix} \right| &= s^3 + 7s^2 + 25s + 15 \end{aligned} \quad (8.177)$$

Expanding the determinant in equation (8.177) gives

$$k_1 = 15, \quad k_2 = 15, \quad k_3 = 0$$

Hence

$$u = -[15 \quad 15 \quad 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (8.178)$$

since $x_2 = \dot{x}_1$, this is identical to the original PD controller

$$G(s) = 15(s+1)$$

Although the solution is the same, the important difference is with state feedback, the closed-loop poles are placed at directly the required locations. With root locus, a certain amount of trial and error in placing open-loop zeros was required to achieve the desired closed-loop locations.

(b) *Reduced-order state observer*: Partitioning the system equation **A** in (8.174) and inserting in equation (8.164)

$$\begin{aligned}
 \left| \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -10 & -7 \end{bmatrix} + \begin{bmatrix} k_{e1} \\ k_{e2} \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right| &= s^2 + 2\zeta\omega_n s + \omega_n^2 \\
 \left| \begin{bmatrix} s & -1 \\ 10 & s+7 \end{bmatrix} + \begin{bmatrix} k_{e1} & 0 \\ k_{e2} & 0 \end{bmatrix} \right| &= s^2 + 63.2s + 2039.4 \\
 \left| \begin{bmatrix} s+k_{e1} & -1 \\ 10+k_{e2} & s+7 \end{bmatrix} \right| &= s^2 + \alpha_{1e}s + \alpha_{0e} \\
 s^2 + (7+k_{e1})s + (7k_{e1} + 10 + k_{e2}) &= s^2 + 63.2s + 2039.4 \quad (8.179)
 \end{aligned}$$

Equating coefficients in equation (8.179)

$$(7 + k_{e1}) = 63.2 \quad k_{e1} = 56.2 \quad (8.180)$$

$$(7 \times 56.2 + 10 + k_{e2}) = 2039.4$$

$$k_{e2} = 1636 \quad (8.181)$$

Referring to Figure 8.11 and partitioned systems (8.174) and (8.163)

$$\begin{aligned}
 \mathbf{B}_e - \mathbf{K}_e \mathbf{b}_1 &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 56.2 \\ 1636 \end{bmatrix} 0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
 \mathbf{A}_{e1} - \mathbf{K}_e \mathbf{a}_{11} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 56.2 \\ 1636 \end{bmatrix} 0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
 \mathbf{A}_{ee} - \mathbf{K}_e \mathbf{A}_{1e} &= \begin{bmatrix} 0 & 1 \\ -10 & -7 \end{bmatrix} - \begin{bmatrix} 56.2 \\ 1636 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} -56.2 & 1 \\ -1646 & -7 \end{bmatrix}
 \end{aligned} \quad (8.182)$$

Inserting equation (8.182) into Figure 8.11 gives the complete state feedback and reduced observer system shown in Figure 8.12.

Comparing the system shown in Figure 8.12 with the original PD controller given in Example 5.10, the state feedback system may be considered to be a PD controller where the proportional term uses measured output variables and the derivative term uses observed state variables.

8.5 Further problems

Example 8.14

For the d.c. motor shown in Figure 4.14, the potential difference across the armature winding is given by equation (4.21)

$$e_a(t) - e_b(t) = L_a \frac{di_a}{dt} + R_a i_a(t)$$

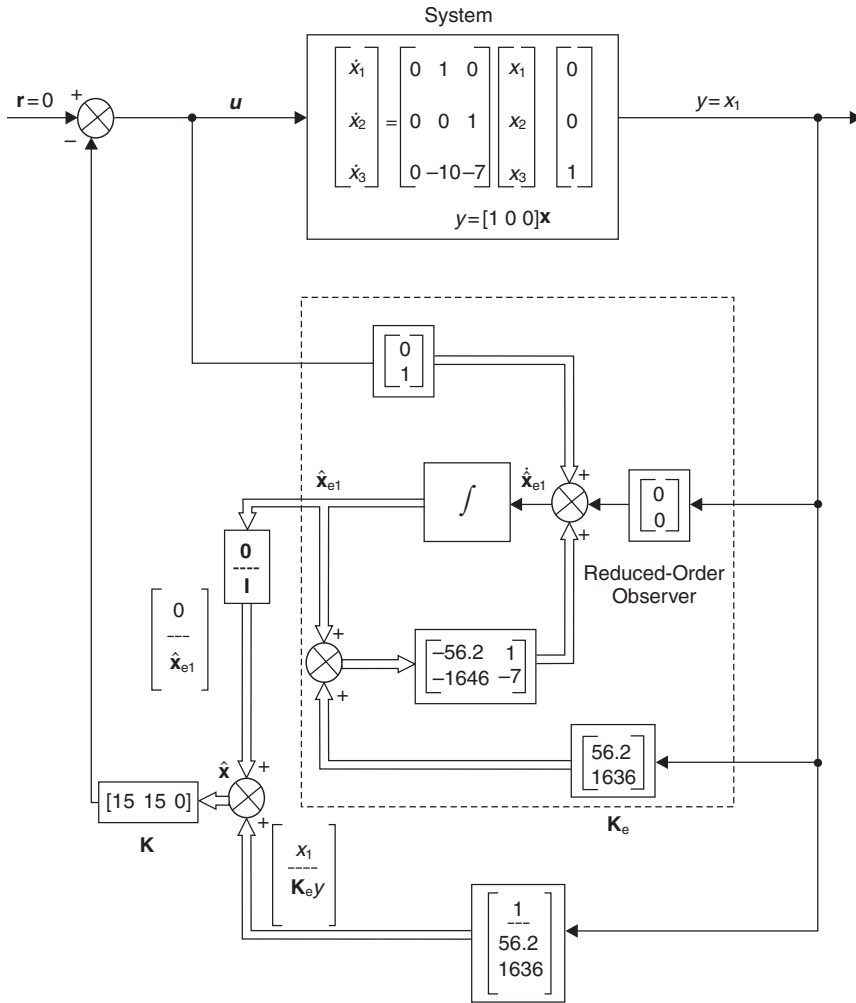


Fig. 8.12 Complete state feedback and reduced observer system for case study Example 8.11.

where, from equation (4.20)

$$e_b(t) = K_b \frac{d\theta}{dt}$$

and the torque $T_m(t)$ developed by the motor is given by equation (4.18)

$$T_m(t) = K_a i_a(t)$$

If the load consists of a rotor of moment of inertia I and a damping device of damping coefficient C , then the load dynamics are

$$T_m(t) - C \frac{d\theta}{dt} = I \frac{d^2\theta}{dt^2}$$

where θ is the angular displacement of the rotor.

- (a) Determine the state and output equations when the state and control variables are

$$x_1 = \theta, \quad x_2 = \dot{x}_1, \quad x_3 = i_a, \quad u = e_a$$

- (b) Determine the state and output equations when the state and control variables are

$$x_1 = \theta, \quad x_2 = \dot{x}_1, \quad x_3 = \dot{x}_2, \quad u = e_a$$

Solution

$$(a) \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{C}{I} & \frac{K_a}{I} \\ 0 & -\frac{K_b}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L_a} \end{bmatrix} u$$

$$\theta = [1 \quad 0 \quad 0]\mathbf{x}$$

$$(b) \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\frac{(K_a K_b + R_a C)}{L_a I} & -\left(\frac{R_a}{L_a} + \frac{C}{I}\right) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{K_a}{L_a I} \end{bmatrix} u$$

$$\theta = [1 \quad 0 \quad 0]\mathbf{x}$$

Example 8.15

Find the state and output equations for the positional servomechanism shown in Figure 8.13 when the state and control variable are

$$x_1 = c(t), \quad x_2 = \dot{x}_1, \quad u = r(t)$$

Solution

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{m} & -\frac{C}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K}{m} \end{bmatrix} u$$

$$c = [1 \quad 0]\mathbf{x}$$

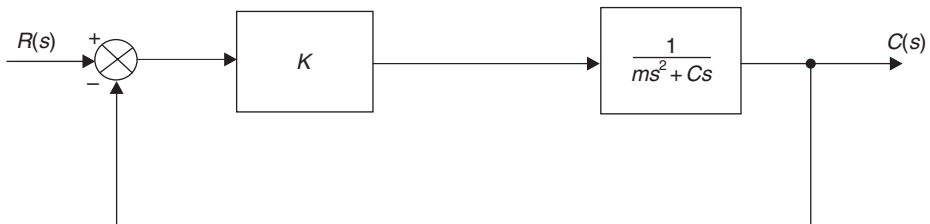


Fig. 8.13 Block diagram of positional servomechanism.

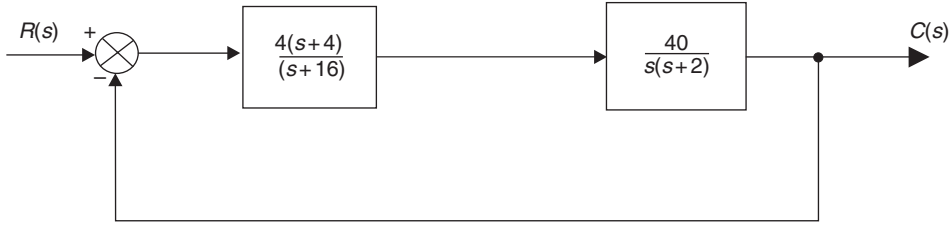


Fig. 8.14 Closed-loop control system.

Example 8.16

Find the state and output equations for the closed-loop control system shown in Figure 8.14 when the state and control variables are

$$x_1 = c(t), \quad x_2 = \dot{x}_1, \quad x_3 = \dot{x}_2, \quad u = r(t)$$

Solution

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -640 & -192 & 18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$c = [640 \quad 160 \quad 0]x$$

Example 8.17

Figure 8.15 shows the block diagram representation of a car cruise control system where $U(s)$ is the desired speed, $X(s)$ is the accelerator position and $V(s)$ is the actual speed.

(a) Find the state and output equations when the state and control variables are

$$x_1 = x(t), \quad x_2 = v(t), \quad u = u(t)$$

(b) Determine the continuous-time state transition matrix $\Phi(t)$.

(c) For a sampling time of 0.1 seconds, evaluate from $\Phi(t)$ the discrete-time state transition matrix $A(T)$.

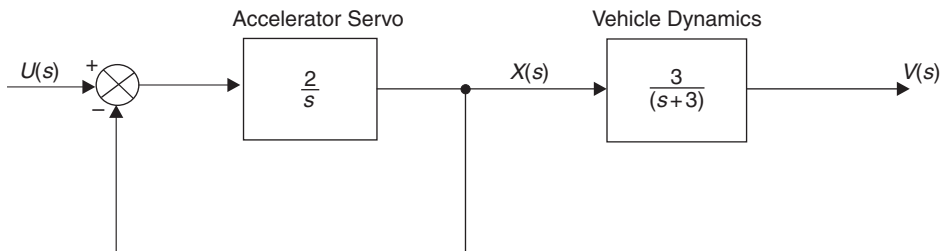


Fig. 8.15 Car cruise control system.

- (d) Using the first three terms of equation (8.80), compute the discrete-time control transition matrix $\mathbf{B}(T)$. Using the difference equations

$$\mathbf{x}(k+1)T = \mathbf{A}(T)\mathbf{x}(kT) + \mathbf{B}\mathbf{u}(kT)$$

determine values for the state variables when $\mathbf{u}(kT)$ is a piece-wise constant function of the form

kT (sec)	0	0.1	0.2	0.3	0.4
$\mathbf{u}(kT)$	10	15	20	25	30

Assume zero initial conditions.

Solution

$$(a) \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 3 & -3 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \end{bmatrix} u$$

$$v = [0 \quad 1]\mathbf{x}$$

$$(b) \Phi(t) = \begin{bmatrix} e^{-2t} & 0 \\ 3(-e^{-3t} + e^{-2t}) & e^{-3t} \end{bmatrix}$$

$$(c) \mathbf{A}(T) = \begin{bmatrix} 0.819 & 0 \\ 0.234 & 0.741 \end{bmatrix}$$

$$(d) \mathbf{B}(T) = \begin{bmatrix} 0.181 \\ 0.025 \end{bmatrix}$$

kT (sec)	0	0.1	0.2	0.3	0.4
$u(kT)$	10	15	20	25	30
x_1	0	1.81	4.197	7.057	10.305
x_2	0	0.25	0.984	2.211	3.915

Example 8.18

The ship roll stabilization system given in case-study Example 5.11 has a forward-path transfer function

$$\frac{\phi_a}{\delta_d}(s) = \frac{K}{(s+1)(s^2 + 0.7s + 2)}$$

- (a) For the condition $K = 1$, find the state and output equations when $x_1 = \phi_a(t)$, $x_2 = \dot{x}_1$, $\dot{x}_3 = \dot{x}_2$ and $u = \delta_d(t)$.
- (b) Calculate the controllability matrix \mathbf{M} and the observability matrix \mathbf{N} and demonstrate that the system is fully controllable and fully observable.
- (c) Determine the state feedback gain matrix \mathbf{K} that produces a set of desired closed-loop poles

$$s = -3.234 \pm j3.3$$

$$s = -3.2$$

- (d) Find the observer gain matrix \mathbf{K}_e for a full-order state observer that produces a set of desired closed-loop poles

$$s = -16.15 \pm j16.5$$

$$s = -16$$

- (e) If output $\phi_a(t) = x_1$ is measured, design a reduced-order state observer with desired closed-loop poles

$$s = -16.15 \pm j16.5$$

Solution

$$(a) \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -2.7 & -1.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$\phi_a = [1 \ 0 \ 0]\mathbf{x}$$

$$(b) \mathbf{M} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -1.7 \\ 1 & -1.7 & 0.19 \end{bmatrix}$$

$$\det(\mathbf{M}) = -1, \text{rank}(\mathbf{M}) = 3$$

System fully controllable.

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\det(\mathbf{N}) = 1, \text{rank}(\mathbf{N}) = 3$$

System fully observable.

$$(c) \mathbf{K} = [66.29 \ 39.34 \ 7.968]$$

$$(d) \mathbf{K}_e = [8527.2 \ 1047.2 \ 46.6]$$

$$(e) \mathbf{K}_e = [530.3 \ 30.6]$$

Optimal and robust control system design

9.1 Review of optimal control

An optimal control system seeks to maximize the return from a system for the minimum cost. In general terms, the optimal control problem is to find a control \mathbf{u} which causes the system

$$\dot{\mathbf{x}} = g(\mathbf{x}(t), \mathbf{u}(t), t) \quad (9.1)$$

to follow an optimal trajectory $\mathbf{x}(t)$ that minimizes the performance criterion, or cost function

$$J = \int_{t_0}^{t_1} h(\mathbf{x}(t), \mathbf{u}(t), t) dt \quad (9.2)$$

The problem is one of constrained functional minimization, and has several approaches.

Variational calculus, Dreyfus (1962), may be employed to obtain a set of differential equations with certain boundary condition properties, known as the Euler–Lagrange equations. The maximum principle of Pontryagin (1962) can also be applied to provide the same boundary conditions by using a Hamiltonian function.

An alternative procedure is the dynamic programming method of Bellman (1957) which is based on the principle of optimality and the imbedding approach. The principle of optimality yields the Hamilton–Jacobi partial differential equation, whose solution results in an optimal control policy. Euler–Lagrange and Pontryagin’s equations are applicable to systems with non-linear, time-varying state equations and non-quadratic, time varying performance criteria. The Hamilton–Jacobi equation is usually solved for the important and special case of the linear time-invariant plant with quadratic performance criterion (called the performance index), which takes the form of the matrix Riccati (1724) equation. This produces an optimal control law as a linear function of the state vector components which is always stable, providing the system is controllable.

9.1.1 Types of optimal control problems

- (a) *The terminal control problem:* This is used to bring the system as close as possible to a given terminal state within a given period of time. An example is an

automatic aircraft landing system, whereby the optimum control policy will focus on minimizing errors in the state vector at the point of landing.

- (b) *The minimum-time control problem:* This is used to reach the terminal state in the shortest possible time period. This usually results in a ‘bang–bang’ control policy whereby the control is set to \mathbf{u}_{\max} initially, switching to \mathbf{u}_{\min} at some specific time. In the case of a car journey, this is the equivalent of the driver keeping his foot flat down on the accelerator for the entire journey, except at the terminal point, when he brakes as hard as possible.
- (c) *The minimum energy control problem:* This is used to transfer the system from an initial state to a final state with minimum expenditure of control energy. Used in satellite control.
- (d) *The regulator control problem:* With the system initially displaced from equilibrium, will return the system to the equilibrium state in such a manner so as to minimize a given performance index.
- (e) *The tracking control problem:* This is used to cause the state of a system to track as close as possible some desired state time history in such a manner so as to minimize a given performance index. This is the generalization of the regulator control problem.

9.1.2 Selection of performance index

The decision on the type of performance index to be selected depends upon the nature of the control problem. Consider the design of an autopilot for a racing yacht.

Conventionally, the autopilot is designed for course-keeping, that is to minimise the error $\psi_e(t)$ between that desired course $\psi_d(t)$ and the actual course $\psi_a(t)$ in the presence of disturbances (wind, waves and current). Since $\psi_d(t)$ is fixed for most of the time, this is in essence a regulator problem.

Using classical design techniques, the autopilot will be tuned to return the vessel on the desired course within the minimum transient period. With an optimal control strategy, a wider view is taken. The objective is to win the race, which means completing it in the shortest possible time. This in turn requires:

- (a) Minimizing the distance off-track, or cross-track error $y_e(t)$. Wandering off track will increase distance travelled and hence time taken.
- (b) Minimizing course or heading error $\psi_e(t)$. It is possible of course to have zero heading error but still be off-track.
- (c) Minimizing rudder activity, i.e. actual rudder angle (as distinct from desired rudder angle) $\delta_a(t)$, and hence minimizing the expenditure of control energy.
- (d) Minimizing forward speed loss $u_e(t)$. As the vessel yaws as a result of correcting a track or heading error, there is an increased angle of attack of the total velocity vector, which results in increased drag and therefore increased forward speed loss.

From equation (9.2) a general performance index could be written

$$J = \int_{t_0}^{t_1} h(y_e(t), \psi_e(t), u_e(t), \delta_a(t)) dt \quad (9.3)$$

Quadratic performance indices

If, in the racing yacht example, the following state and control variables are defined

$$x_1 = y_e(t), \quad x_2 = \psi_e(t), \quad x_3 = u_e(t), \quad u = \delta_a(t)$$

then the performance index could be expressed

$$J = \int_{t_0}^{t_1} \{ (q_{11}x_1 + q_{22}x_2 + q_{33}x_3) + (r_1u) \} dt \quad (9.4)$$

or

$$J = \int_{t_0}^{t_1} (\mathbf{Q}\mathbf{x} + \mathbf{R}\mathbf{u}) dt \quad (9.5)$$

If the state and control variables in equations (9.4) and (9.5) are squared, then the performance index become quadratic. The advantage of a quadratic performance index is that for a linear system it has a mathematical solution that yields a linear control law of the form

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \quad (9.6)$$

A quadratic performance index for this example is therefore

$$J = \int_{t_0}^{t_1} \{ (q_{11}x_1^2 + q_{22}x_2^2 + q_{33}x_3^2) + (r_1u^2) \} dt \quad (9.7)$$

$$J = \int_{t_0}^{t_1} \left[\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & 0 & q_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + [u][r_1][u] \right] dt$$

or, in general

$$J = \int_{t_0}^{t_1} (\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u}) dt \quad (9.8)$$

\mathbf{Q} and \mathbf{R} are the state and control weighting matrices and are always square and symmetric. J is always a scalar quantity.

9.2 The Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) provides an optimal control law for a linear system with a quadratic performance index.

9.2.1 Continuous form

Define a functional equation of the form

$$f(\mathbf{x}, t) = \min_{\mathbf{u}} \int_{t_0}^{t_1} h(\mathbf{x}, \mathbf{u}) dt \quad (9.9)$$

where over the time interval t_0 to t_1 ,

$$f(\mathbf{x}, t_0) = f(\mathbf{x}(0))$$

$$f(\mathbf{x}, t_1) = \mathbf{0}$$

From equations (9.1) and (9.2), a Hamilton–Jacobi equation may be expressed as

$$\frac{\partial f}{\partial t} = -\min_{\mathbf{u}} \left[h(\mathbf{x}, \mathbf{u}) + \left(\frac{\partial f}{\partial \mathbf{x}} \right)^T g(\mathbf{x}, \mathbf{u}) \right] \quad (9.10)$$

For a linear, time invariant plant, equation (9.1) becomes

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \quad (9.11)$$

And if equation (9.2) is a quadratic performance index

$$J = \int_{t_0}^{t_1} (\mathbf{x}^T \mathbf{Qx} + \mathbf{u}^T \mathbf{Ru}) dt \quad (9.12)$$

Substituting equations (9.11) and (9.12) into (9.10)

$$\frac{\partial f}{\partial t} = -\min_{\mathbf{u}} \left[\mathbf{x}^T \mathbf{Qx} + \mathbf{u}^T \mathbf{Ru} + \left(\frac{\partial f}{\partial \mathbf{x}} \right)^T (\mathbf{Ax} + \mathbf{Bu}) \right] \quad (9.13)$$

Introducing a relationship of the form

$$f(\mathbf{x}, t) = \mathbf{x}^T \mathbf{P} \mathbf{x} \quad (9.14)$$

where \mathbf{P} is a square, symmetric matrix, then

$$\frac{\partial f}{\partial t} = \mathbf{x}^T \frac{\partial}{\partial t} \mathbf{P} \mathbf{x} \quad (9.15)$$

and

$$\frac{\partial f}{\partial \mathbf{x}} = 2\mathbf{Px}$$

$$\left[\frac{\partial f}{\partial \mathbf{x}} \right]^T = 2\mathbf{x}^T \mathbf{P} \quad (9.16)$$

Inserting equations (9.15) and (9.16) into (9.13) gives

$$\mathbf{x}^T \frac{\partial \mathbf{P}}{\partial t} \mathbf{x} = -\min_{\mathbf{u}} [\mathbf{x}^T \mathbf{Qx} + \mathbf{u}^T \mathbf{Ru} + 2\mathbf{x}^T \mathbf{P}(\mathbf{Ax} + \mathbf{Bu})] \quad (9.17)$$

To minimize \mathbf{u} , from equation (9.17)

$$\frac{\partial [\partial f / \partial t]}{\partial \mathbf{u}} = 2\mathbf{u}^T \mathbf{R} + 2\mathbf{x}^T \mathbf{PB} = 0 \quad (9.18)$$

Equation (9.18) can be re-arranged to give the optimal control law

$$\mathbf{u}_{\text{opt}} = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{Px} \quad (9.19)$$

or

$$\mathbf{u}_{\text{opt}} = -\mathbf{K}\mathbf{x} \quad (9.20)$$

where

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \quad (9.21)$$

Substituting equation (9.19) back into (9.17) gives

$$\mathbf{x}^T\dot{\mathbf{P}}\mathbf{x} = -\mathbf{x}^T(\mathbf{Q} + 2\mathbf{P}\mathbf{A} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P})\mathbf{x} \quad (9.22)$$

since

$$2\mathbf{x}^T\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{x}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{x}$$

then

$$\dot{\mathbf{P}} = -\mathbf{P}\mathbf{A} - \mathbf{A}^T\mathbf{P} - \mathbf{Q} + \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} \quad (9.23)$$

Equation (9.23) belongs to a class of non-linear differential equations known as the matrix Riccati equations. The coefficients of $\mathbf{P}(t)$ are found by integration in reverse time starting with the boundary condition

$$\mathbf{x}^T(t_1)\mathbf{P}(t_1)\mathbf{x}(t_1) = \mathbf{0} \quad (9.24)$$

Kalman demonstrated that as integration in reverse time proceeds, the solutions of $\mathbf{P}(t)$ converge to constant values. Should t_1 be infinite, or far removed from t_0 , the matrix Riccati equations reduce to a set of simultaneous equations

$$\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} + \mathbf{Q} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} = \mathbf{0} \quad (9.25)$$

Equations (9.23) and (9.25) are the continuous solution of the matrix Riccati equation.

9.2.2 Discrete form

From equation (8.76) the discrete solution of the state equation is

$$\mathbf{x}[(k+1)T] = \mathbf{A}(T)\mathbf{x}(kT) + \mathbf{B}(T)\mathbf{u}(kT) \quad (9.26)$$

For simplicity, if (kT) is written as (k) , then

$$\mathbf{x}(k+1) = \mathbf{A}(T)\mathbf{x}(k) + \mathbf{B}(T)\mathbf{u}(k) \quad (9.27)$$

The discrete quadratic performance index is

$$J = \sum_{k=0}^{N-1} (\mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k))T \quad (9.28)$$

The discrete solution of the matrix Riccati equation solves recursively for \mathbf{K} and \mathbf{P} in reverse time, commencing at the terminal time, where

$$\mathbf{K}(N - (k+1)) = [\mathbf{TR} + \mathbf{B}^T(T)\mathbf{P}(N-k)\mathbf{B}(T)]^{-1}\mathbf{B}^T(T)\mathbf{P}(N-k)\mathbf{A}(T) \quad (9.29)$$

and

$$\begin{aligned} \mathbf{P}(N - (k + 1)) = & [T\mathbf{Q} + \mathbf{K}^T(N - (k + 1))T\mathbf{R}\mathbf{K}(N - (k + 1))] + [\mathbf{A}(T) \\ & - \mathbf{B}(T)\mathbf{K}(N - (k + 1))]^T \mathbf{P}(N - k) [\mathbf{A}(T) - \mathbf{B}(T)\mathbf{K}(N - (k + 1))] \end{aligned} \quad (9.30)$$

As k is increased from 0 to $N - 1$, the algorithm proceeds in reverse time. When run in forward-time, the optimal control at step k is

$$\mathbf{u}_{\text{opt}}(k) = -\mathbf{K}(k)\mathbf{x}(k) \quad (9.31)$$

The boundary condition is specified at the terminal time ($k = 0$), where

$$\mathbf{x}^T(N)\mathbf{P}(N)\mathbf{x}(N) = \mathbf{0} \quad (9.32)$$

The reverse-time recursive process can commence with $\mathbf{P}(N) = \mathbf{0}$ or alternatively, with $\mathbf{P}(N - 1) = T\mathbf{Q}$.

Example 9.1 (See also Appendix 1, *examp91.m*)

The regulator shown in Figure 9.1 contains a plant that is described by

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u} \\ y &= [1 \quad 0]\mathbf{x} \end{aligned}$$

and has a performance index

$$J = \int_0^\infty \left[\mathbf{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + \mathbf{u}^2 \right] dt$$

Determine

- the Riccati matrix \mathbf{P}
- the state feedback matrix \mathbf{K}
- the closed-loop eigenvalues

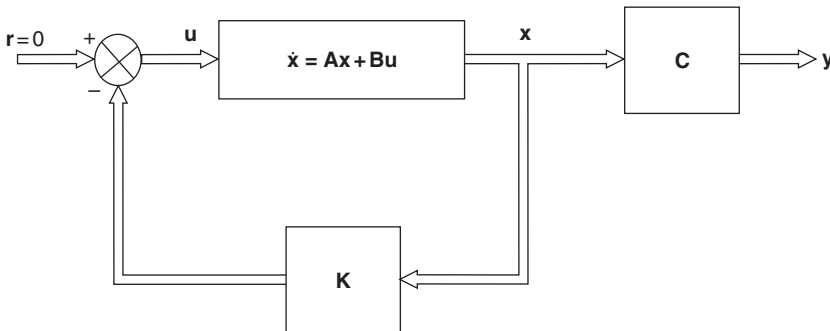


Fig. 9.1 Optimal regulator.

Solution

(a)

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{R} = \text{scalar} = 1$$

From equation (9.25) the reduced Riccati equation is

$$\mathbf{P}\mathbf{A} + \mathbf{A}^T\mathbf{P} + \mathbf{Q} - \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} = \mathbf{0} \quad (9.33)$$

$$\mathbf{P}\mathbf{A} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} = \begin{bmatrix} -p_{12} & p_{11} - 2p_{12} \\ -p_{22} & p_{21} - 2p_{22} \end{bmatrix} \quad (9.34)$$

$$\mathbf{A}^T\mathbf{P} = \begin{bmatrix} 0 & -1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} -p_{21} & -p_{22} \\ p_{11} - 2p_{21} & p_{12} - 2p_{22} \end{bmatrix} \quad (9.35)$$

$$\begin{aligned} \mathbf{PBR}^{-1}\mathbf{B}^T\mathbf{P} &= \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} 1 \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \\ &= \begin{bmatrix} p_{12} \\ p_{22} \end{bmatrix} \begin{bmatrix} p_{21} & p_{22} \end{bmatrix} \\ &= \begin{bmatrix} p_{12}p_{21} & p_{12}p_{22} \\ p_{22}p_{21} & p_{22}^2 \end{bmatrix} \end{aligned} \quad (9.36)$$

Combining equations (9.34), (9.35) and (9.36) gives

$$\begin{aligned} \begin{bmatrix} -p_{12} & p_{11} - 2p_{12} \\ -p_{22} & p_{21} - 2p_{22} \end{bmatrix} &+ \begin{bmatrix} -p_{21} & -p_{22} \\ p_{11} - 2p_{21} & p_{12} - 2p_{22} \end{bmatrix} \\ &+ \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} p_{12}p_{21} & p_{12}p_{22} \\ p_{22}p_{21} & p_{22}^2 \end{bmatrix} = \mathbf{0} \end{aligned} \quad (9.37)$$

Since \mathbf{P} is symmetric, $p_{21} = p_{12}$. Equation (9.37) can be expressed as four simultaneous equations

$$-p_{12} - p_{12} + 2 - p_{12}^2 = 0 \quad (9.38)$$

$$p_{11} - 2p_{12} - p_{22} - p_{12}p_{22} = 0 \quad (9.39)$$

$$-p_{22} + p_{11} - 2p_{12} - p_{12}p_{22} = 0 \quad (9.40)$$

$$p_{12} - 2p_{22} + p_{12} - 2p_{22} + 1 - p_{22}^2 = 0 \quad (9.41)$$

Note that equations (9.39) and (9.40) are the same. From equation (9.38)

$$p_{12}^2 + 2p_{12} - 2 = 0$$

solving

$$p_{12} = p_{21} = 0.732 \quad \text{and} \quad -2.732$$

Using positive value

$$p_{12} = p_{21} = 0.732 \quad (9.42)$$

From equation (9.41)

$$\begin{aligned} 2p_{12} - 4p_{22} + 1 - p_{22}^2 &= 0 \\ p_{22}^2 + 4p_{22} - 2.464 &= 0 \end{aligned}$$

solving

$$p_{22} = 0.542 \quad \text{and} \quad -4.542$$

Using positive value

$$p_{22} = 0.542 \quad (9.43)$$

From equation (9.39)

$$\begin{aligned} p_{11} - (2 \times 0.732) - 0.542 - (0.732 \times 0.542) &= 0 \\ p_{11} &= 2.403 \end{aligned} \quad (9.44)$$

From equations (9.42), (9.43) and (9.44) the Riccati matrix is

$$\mathbf{P} = \begin{bmatrix} 2.403 & 0.732 \\ 0.732 & 0.542 \end{bmatrix} \quad (9.45)$$

(b) Equation (9.21) gives the state feedback matrix

$$\mathbf{K} = \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P} = 1[0 \quad 1] \begin{bmatrix} 2.403 & 0.732 \\ 0.732 & 0.542 \end{bmatrix} \quad (9.46)$$

Hence

$$\mathbf{K} = [0.732 \quad 0.542]$$

(c) From equation (8.96), the closed-loop eigenvalues are

$$\begin{aligned}
 |s\mathbf{I} - \mathbf{A} + \mathbf{BK}| &= 0 \\
 \left| \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0.732 & 0.542 \end{bmatrix} \right| &= 0 \\
 \left| \begin{bmatrix} s & -1 \\ 1 & s+2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0.732 & 0.542 \end{bmatrix} \right| &= 0 \\
 \begin{vmatrix} s & -1 \\ 1.732 & s+2.542 \end{vmatrix} &= 0 \\
 s^2 + 2.542s + 1.732 &= 0 \\
 s_1, s_2 &= -1.271 \pm j0.341
 \end{aligned}$$

In general, for a second-order system, when \mathbf{Q} is a diagonal matrix and \mathbf{R} is a scalar quantity, the elements of the Riccati matrix \mathbf{P} are

$$\begin{aligned}
 p_{11} &= \left[\frac{b_2^2}{r} \right] p_{12} p_{22} - p_{22} a_{21} - p_{12} a_{22} \\
 p_{12} = p_{21} &= \frac{r}{b_2^2} \left[a_{21} \pm \sqrt{a_{21}^2 + \frac{q_{11} b_2^2}{r}} \right] \\
 p_{22} &= \frac{r}{b_2^2} \left[a_{22} \pm \sqrt{a_{22}^2 + \frac{(2p_{12} + q_{22})}{r}} \right]
 \end{aligned} \tag{9.47}$$

9.3 The linear quadratic tracking problem

The tracking or servomechanism problem is defined in section 9.1.1(e), and is directed at applying a control $\mathbf{u}(t)$ to drive a plant so that the state vector $\mathbf{x}(t)$ follows a desired state trajectory $\mathbf{r}(t)$ in some optimal manner.

9.3.1 Continuous form

The quadratic performance index to be minimized is

$$J = \int_{t_0}^{t_1} [(\mathbf{r} - \mathbf{x})^T \mathbf{Q}(\mathbf{r} - \mathbf{x}) + \mathbf{u}^T \mathbf{R} \mathbf{u}] dt \tag{9.48}$$

It can be shown that the constrained functional minimization of equation (9.48) yields again the matrix Riccati equations (9.23) and (9.25) obtained for the LQR, combined with the additional set of reverse-time state tracking equations

$$\dot{\mathbf{s}} = (\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P})^T \mathbf{s} - \mathbf{Q}\mathbf{r} \tag{9.49}$$

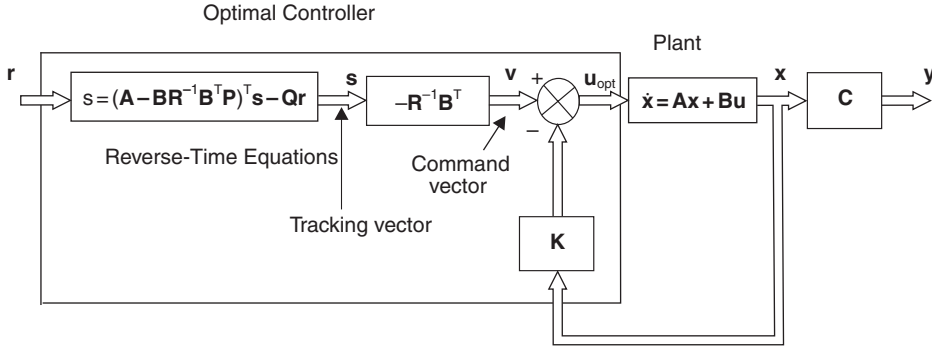


Fig. 9.2 Optimal tracking system.

where \mathbf{s} is a tracking vector, whose boundary condition is

$$\mathbf{s}(t_1) = 0 \quad (9.50)$$

and the optimal control law is given by

$$\mathbf{u}_{\text{opt}} = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{P} \mathbf{x} - \mathbf{R}^{-1} \mathbf{B}^T \mathbf{s}$$

If

$$\mathbf{v} = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{s}$$

and

$$\mathbf{K} = \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}$$

Then

$$\mathbf{u}_{\text{opt}} = \mathbf{v} - \mathbf{K} \mathbf{x} \quad (9.51)$$

Hence, if the desired state vector $\mathbf{r}(t)$ is known in advance, tracking errors may be reduced by allowing the system to follow a command vector $\mathbf{v}(t)$ computed in advance using the reverse-time equation (9.49). An optimal controller for a tracking system is shown in Figure 9.2.

9.3.2 Discrete form

The discrete quadratic performance index, writing (kT) as (k) , is

$$J = \sum_{k=0}^{N-1} [(\mathbf{r}(k) - \mathbf{x}(k))^T \mathbf{Q}(\mathbf{r}(k) - \mathbf{x}(k)) + \mathbf{u}^T(k) \mathbf{R} \mathbf{u}(k)] T \quad (9.52)$$

Discrete minimization gives the recursive Riccati equations (9.29) and (9.30). These are run in reverse-time together with the discrete reverse-time state tracking equation

$$\mathbf{s}(N - (k + 1)) = \mathbf{F}(T) \mathbf{s}(N - k) + \mathbf{G}(T) \mathbf{r}(N - k) \quad (9.53)$$

having the boundary condition

$$\mathbf{s}(N) = 0$$

$\mathbf{F}(T)$ and $\mathbf{G}(T)$ are the discrete transition and control matrices and are obtained by converting the matrices in the continuous equation (9.49) into discrete form using equations (8.78) and (8.80).

The command vector \mathbf{v} is given by

$$\mathbf{v}(N - k) = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{s}(N - k) \quad (9.54)$$

When run in forward-time, the optimal control at time (kT) is

$$\mathbf{u}_{\text{opt}}(kT) = \mathbf{v}(kT) - \mathbf{K}(kT)\mathbf{x}(kT) \quad (9.55)$$

The values of $\mathbf{x}(kT)$ are calculated using the plant state transition equation

$$\mathbf{x}(k + 1)T = \mathbf{A}(T)\mathbf{x}(kT) + \mathbf{B}(T)\mathbf{u}_{\text{opt}}(kT) \quad (9.56)$$

Example 9.2 (See also Appendix 1, *examp92.m*)

The optimal tracking system shown in Figure 9.2 contains a plant that is described by

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ \mathbf{y} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned}$$

The discrete performance index is

$$J = \sum_{k=0}^{200} \left[(r_1(kT) - x_1(kT))(r_2(kT) - x_2(kT)) \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_1(kT) - x_1(kT) \\ r_2(kT) - x_2(kT) \end{bmatrix} + u(kT)^2 \right] T$$

It is required that the system tracks the following desired state vector

$$\begin{bmatrix} r_1(kT) \\ r_2(kT) \end{bmatrix} = \begin{bmatrix} 1.0 \sin(0.6284kT) \\ 0.6 \cos(0.6284kT) \end{bmatrix}$$

over a period of 0–20 seconds. The sampling time T is 0.1 seconds.

In reverse-time, starting with $\mathbf{P}(N) = \mathbf{0}$ at $NT = 20$ seconds, compute the state feedback gain matrix $\mathbf{K}(kT)$ and Riccati matrix $\mathbf{P}(kT)$ using equations (9.29) and (9.30). Also in reverse time, use the desired state vector $\mathbf{r}(kT)$ to drive the tracking equation (9.53) with the boundary condition $\mathbf{s}(N) = 0$ and hence compute the command vector $\mathbf{v}(kT)$.

In forward-time, use the command vector $\mathbf{v}(kT)$ and state vector $\mathbf{x}(kT)$ to calculate $\mathbf{u}_{\text{opt}}(kT)$ in equation (9.55) and hence, using the plant state transition equation (9.56) calculate the state trajectories.

Solution

The reverse-time calculations are shown in Figure 9.3. Using equations (9.29) and (9.30) and commencing with $\mathbf{P}(N) = 0$, it can be seen that the solution for \mathbf{K} (and also \mathbf{P}) settle down after about 2 seconds to give steady-state values of

$$\begin{aligned}\mathbf{K}(kT) &= [2.0658 \quad 1.4880] \\ \mathbf{P}(kT) &= \begin{bmatrix} 8.0518 & 2.3145 \\ 2.3145 & 1.6310 \end{bmatrix}\end{aligned}\quad (9.57)$$

Using equation (9.49), together with equations (8.78) and (8.80), to calculate $\mathbf{F}(T)$ and $\mathbf{G}(T)$ in equation (9.53), for $T = 0.1$ seconds, the discrete reverse-time state tracking equation is

$$\begin{aligned}\begin{bmatrix} s_1(N - (k + 1)) \\ s_2(N - (k + 1)) \end{bmatrix} &= \begin{bmatrix} 0.9859 & -0.2700 \\ 0.0881 & 0.7668 \end{bmatrix} \begin{bmatrix} s_1(N - k) \\ s_2(N - k) \end{bmatrix} \\ &+ \begin{bmatrix} -0.9952 & 0.0141 \\ -0.0460 & -0.0881 \end{bmatrix} \begin{bmatrix} r_1(N - k) \\ r_2(N - k) \end{bmatrix}\end{aligned}$$

and

$$\mathbf{v}(N - k) = -1[0 \quad 1] \begin{bmatrix} s_1(N - k) \\ s_2(N - k) \end{bmatrix}\quad (9.58)$$

Then the command vector \mathbf{v} (in this case a scalar) is generated in reverse-time as shown in Figure 9.3. The forward-time response is shown in Figure 9.4.

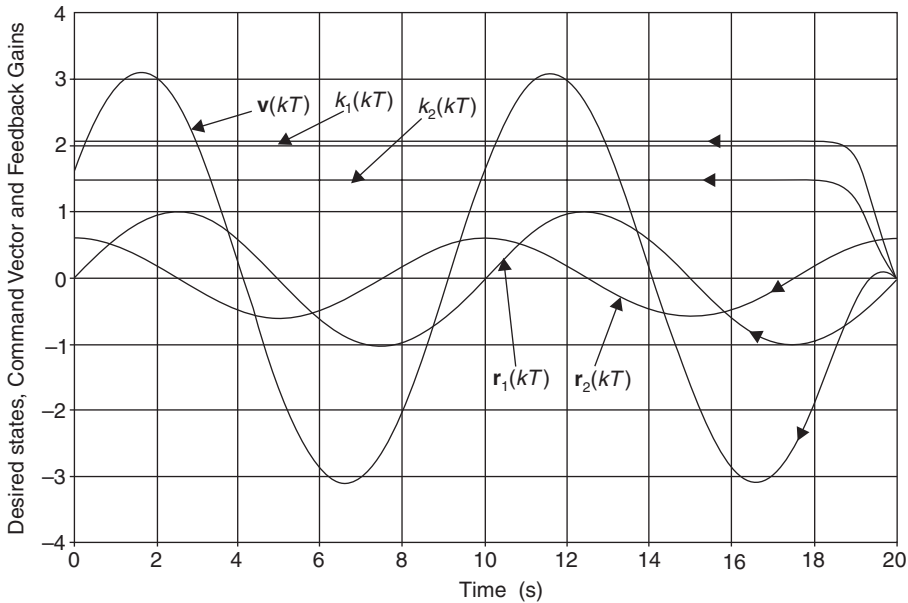


Fig. 9.3 Reverse-time solutions for a tracking system.

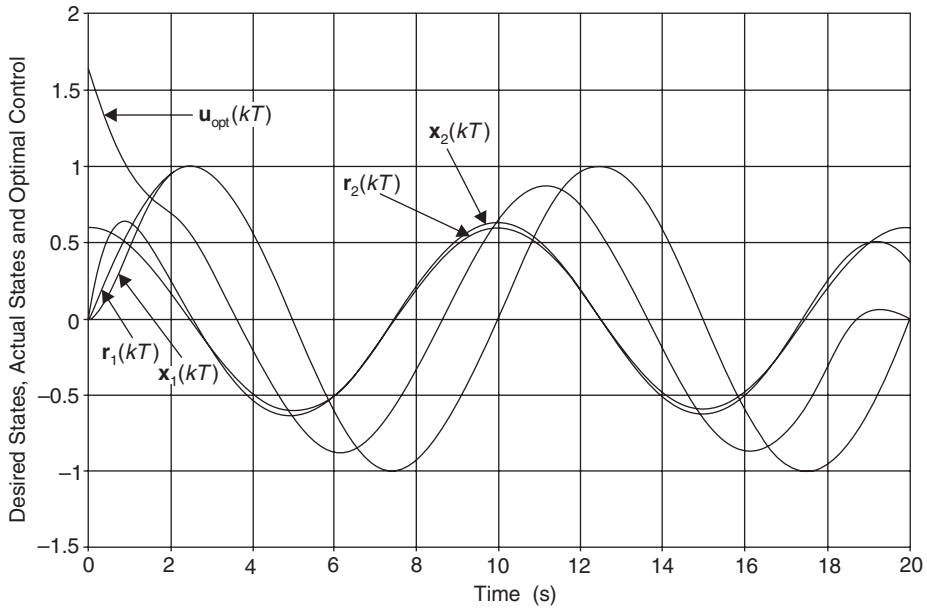


Fig. 9.4 Forward-time response of a tracking system.

The optimal control is calculated using equation (9.55) and the values of the state variables are found using the plant state transition equation (9.56)

$$\begin{bmatrix} x_1(k+1)T \\ x_2(k+1)T \end{bmatrix} = \begin{bmatrix} 0.9952 & 0.0950 \\ -0.0950 & 0.9002 \end{bmatrix} \begin{bmatrix} x_1(kT) \\ x_2(kT) \end{bmatrix} + \begin{bmatrix} 0.0048 \\ 0.0950 \end{bmatrix} u_{\text{opt}}(kT) \quad (9.59)$$

where $\mathbf{A}(T)$ and $\mathbf{B}(T)$ are calculated from equations (8.78) and (8.80). From Figure 9.4 it can be seen that after an initial transient period, the optimal control law takes the form of a phase lead compensator. Because of the weighting of the \mathbf{Q} matrix in the performance index, $x_1(kT)$ tracks $r_1(kT)$ more closely than $x_2(kT)$ tracks $r_2(kT)$.

9.4 The Kalman filter

In the design of state observers in section 8.4.3, it was assumed that the measurements $\mathbf{y} = \mathbf{C}\mathbf{x}$ were noise free. In practice, this is not usually the case and therefore the observed state vector $\hat{\mathbf{x}}$ may also be contaminated with noise.

9.4.1 The state estimation process

State estimation is the process of extracting a best estimate of a variable from a number of measurements that contain noise.

The classical problem of obtaining a best estimate of a signal by combining two noisy continuous measurements of the same signal was first solved by Wiener (1949).

His solution required that both the signal and noise be modelled as random process with known statistical properties.

This work was extended by Kalman and Bucy (1961) who designed a state estimation process based upon an optimal minimum variance filter, generally referred to as a Kalman filter.

9.4.2 The Kalman filter single variable estimation problem

The Kalman filter is a complementary form of the Weiner filter. Let A_x be a measurement of a parameter x and let its variance P_a be given by

$$P_a = E\{(A_x - \bar{A}_x)^2\} \quad (9.60)$$

where \bar{A}_x is the mean and $E\{\}$ is the expected value.

Let B_x be a measurement from another system of the same parameter and the variance P_b is

$$P_b = E\{(B_x - \bar{B}_x)^2\} \quad (9.61)$$

Assume that x can be expressed by the parametric relationship

$$x = A_x K + B_x(1 - K) \quad (9.62)$$

where K is any weighting factor between 0 and 1. The problem is to derive a value of K which gives an optimal combination of A_x and B_x and hence the best estimate of measured variable x , which is given the symbol \hat{x} (pronounced x hat).

Let P be the variance of the weighted mean

$$P = E\{(x - \bar{x})^2\} \quad (9.63)$$

The optimal value of K is the one that yields the minimum variance, i.e.

$$\frac{dP}{dK} = 0 \quad (9.64)$$

Substitution of equation (9.62) into (9.63) gives

$$P = K^2 P_A + (1 - K)^2 P_B \quad (9.65)$$

Hence K is given by

$$\frac{d}{dK} \{K^2 P_A + (1 - K)^2 P_B\} = 0$$

From which

$$K = \frac{P_B}{P_A + P_B} \quad (9.66)$$

Substitution of equation (9.66) into (9.62) provides

$$\hat{x} = A_x - \left\{ \frac{P_A}{P_A + P_B} \right\} (A_x - B_x) \quad (9.67)$$

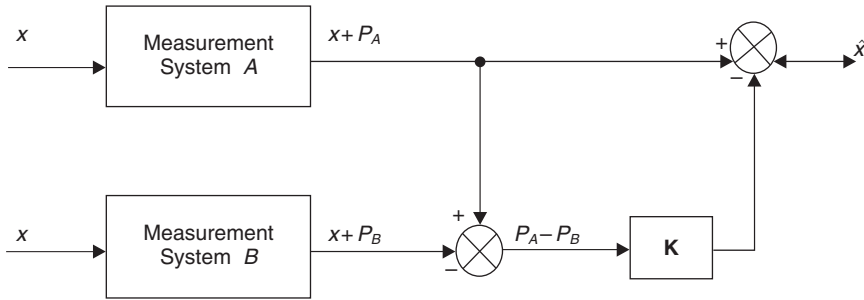


Fig. 9.5 Integration of two measurement systems to obtain optimal estimate.

or

$$\hat{x} = A_x - K(A_x - B_x) \quad (9.68)$$

K is the Kalman gain and the total error variance expected is

$$P = P_A - K(P_A - P_B) \quad (9.69)$$

so that

$$\hat{x} = x + P_A - K(P_A - P_B) \quad (9.70)$$

Equation (9.70) is illustrated in Figure 9.5.

9.4.3 The Kalman filter multivariable state estimation problem

Consider a plant that is subject to a Gaussian sequence of disturbances $w(kT)$ with disturbance transition matrix $C_d(T)$. Measurements $z(k+1)T$ contain a Gaussian noise sequence $v(k+1)T$ as shown in Figure 9.6.

The general form of the Kalman filter usually contains a discrete model of the system together with a set of recursive equations that continuously update the Kalman gain matrix K and the system covariance matrix P .

The state estimate $\hat{x}(k+1/k+1)$ is obtained by calculating the predicted state $\hat{x}(k+1/k)$ from

$$\hat{x}(k+1/k)T = A(T)\hat{x}(k/k)T + B(T)u(kT) \quad (9.71)$$

and then determining the estimated state at time $(k+1)T$ using

$$\hat{x}(k+1/k+1)T = \hat{x}(k+1/k)T + K(k+1)\{z(k+1)T - C(T)\hat{x}(k+1/k)T\} \quad (9.72)$$

The term (k/k) means data at time k based on information available at time k . The term $(k+1/k)$ means data used at time $k+1$ based on information available at time k . Similarly $(k+1/k+1)$ means data at time $k+1$ based on information available at time $k+1$.

The vector of measurements is given by

$$z(k+1)T = C(T)x(k+1)T + v(k+1)T \quad (9.73)$$

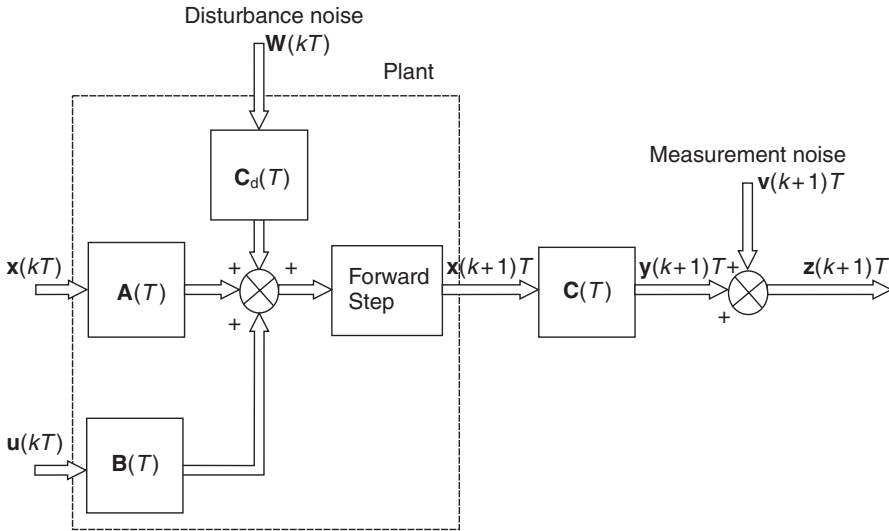


Fig. 9.6 Plant with disturbance and measurement noise.

where

$\mathbf{z}(k+1)T$ is the measurement vector

$\mathbf{C}(T)$ is the measurement matrix

$\mathbf{v}(k+1)T$ is a Gaussian noise sequence

The Kalman gain matrix \mathbf{K} is obtained from a set of recursive equations that commence from some initial covariance matrix $\mathbf{P}(k/k)$

$$\mathbf{P}(k+1/k) = \mathbf{A}(T)\mathbf{P}(k/k)\mathbf{A}^T(T) + \mathbf{C}_d(T)\mathbf{Q}\mathbf{C}_d^T(T) \quad (9.74)$$

$$\mathbf{K}(k+1) = \mathbf{P}(k+1/k)\mathbf{C}^T(T)\{\mathbf{C}(T)\mathbf{P}(k+1/k)\mathbf{C}^T(T) + \mathbf{R}\}^{-1} \quad (9.75)$$

$$\mathbf{P}(k+1/k+1) = \{\mathbf{I} - \mathbf{K}(k+1)\mathbf{C}(T)\}\mathbf{P}(k+1/k) \quad (9.76)$$

The recursive process continues by substituting the covariance matrix $\mathbf{P}(k+1/k+1)$ computed in equation (9.76) back into (9.74) as $\mathbf{P}(k/k)$ until $\mathbf{K}(k+1)$ settles to a steady value, see Appendix 1, script files *kalfilc.m* for the continuous solution and *kalfild.m* for the above discrete solution. In equations (9.74)–(9.76)

$\mathbf{C}_d(T)$ is the disturbance transition matrix

\mathbf{Q} is the disturbance noise covariance matrix

\mathbf{R} is the measurement noise covariance matrix

Equations (9.71)–(9.76) are illustrated in Figure 9.7 which shows the block diagram of the Kalman filter.

The recursive equations (9.74)–(9.76) that calculate the Kalman gain matrix and covariance matrix for a Kalman filter are similar to equations (9.29) and (9.30) that

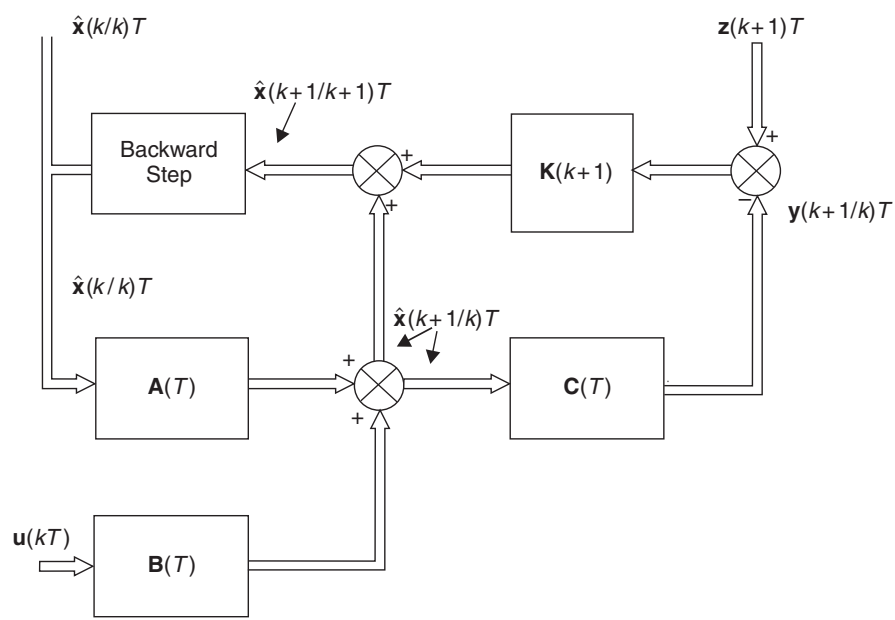


Fig. 9.7 The Kalman filter.

calculate the feedback matrix and Riccati matrix for a linear quadratic regulator. The difference is that the Kalman filter is computed in forward-time, the LQR being computed in reverse-time.

9.5 Linear Quadratic Gaussian control system design

A control system that contains a LQ Regulator/Tracking controller together with a Kalman filter state estimator as shown in Figure 9.8 is called a Linear Quadratic Gaussian (LQG) control system.

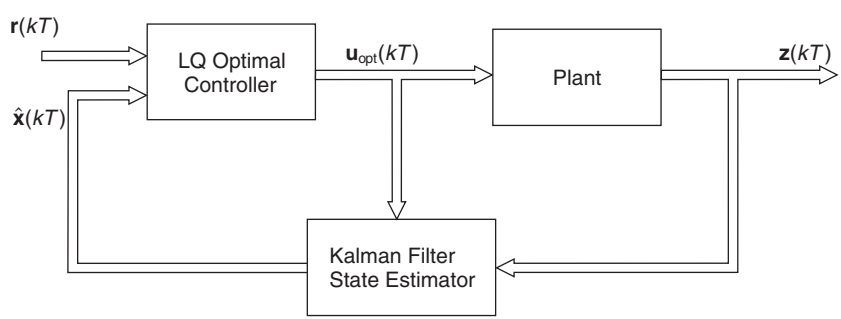


Fig. 9.8 Linear Quadratic Gaussian (LQG) control system.

Case study

Example 9.3 (See also Appendix 1, *examp93.m*)

China clay is used in the paper, ceramics and fertilizer industries, and is washed from quarry faces, by high pressure hoses. A pressing operation reduces the moisture content in the clay to about 30%, and then the clay is extruded into small cylindrical shaped noodles. The clay noodles are then passed through the band drying oven shown in Figure 9.9 at rates varying between 2 and 15 tonnes/hour. Upon exit, the moisture content of the clay should be controlled to a desired level of between 4 and 12%, with a deviation of no more than $\pm 1\%$. The process air is heated by mixing the exhaust gas from a gas burner with a large quantity of dilution air to meet the specified air flow-rate into the dryer.

An existing control arrangement uses a PID controller to control the temperature of the process air (measured by thermocouples) and the dry clay moisture content measured by samples taken by the works laboratory. If this is out of specification, then the process air temperature is raised or lowered. The dry clay moisture content can be measured by an infrared absorption analyser, but on its own, this is considered to be too noisy and unreliable.

The important process parameters are

- (a) Burner exhaust temperature $t_b(t)$ ($^{\circ}\text{C}$)
- (b) Dryer outlet temperature $t_d(t)$ ($^{\circ}\text{C}$)
- (c) Dryer outlet clay moisture content $m_f(t)$ (%)

The important control parameters are

- (i) Burner gas supply valve angle $v_a(t)$ (rad)
- (ii) Dryer clay feed-rate $f_i(t)$ (tonnes/hour)

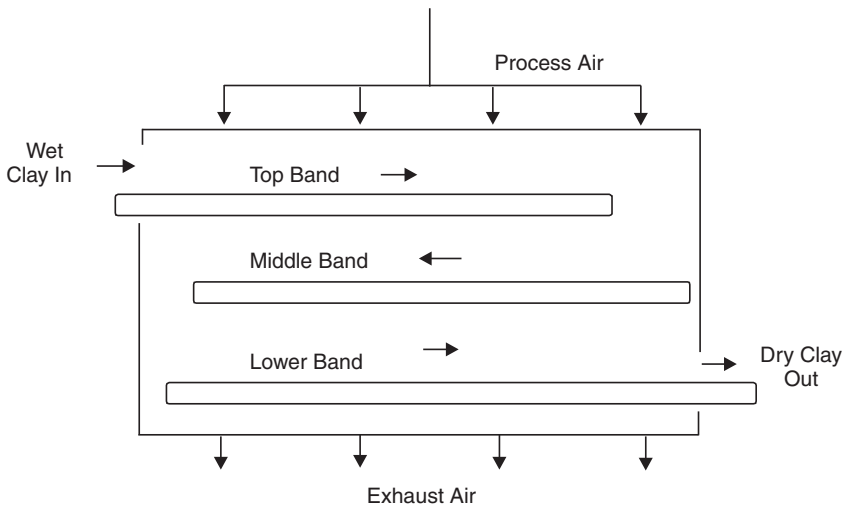


Fig. 9.9 Band drying oven.

A proposed control scheme by Drew and Burns (1992) uses an LQG design, whereby the three process parameters are controlled in an optimal manner, their values (particularly the moisture control) being estimated.

System model: The dynamic characteristics of the dryer were measured experimentally. This yielded the following transfer functions

Burner model

$$G_1(s) = \frac{T_b}{V_a}(s) = \frac{420}{1 + 47s} \quad (9.77)$$

Dryer model

$$G_2(s) = \frac{T_d}{T_b}(s) = \frac{0.119}{1 + 200s} \quad (9.78)$$

Moisture models

$$G_{31}(s) = \frac{M_f}{T_d}(s) = \frac{-0.167}{1 + 440s} \quad (9.79)$$

$$G_{32}(s) = \frac{M_f}{F_i}(s) = \frac{0.582}{1 + 440s} \quad (9.80)$$

Equations (9.79) and (9.80) can be combined to give

$$M_f(s) = \frac{-0.167T_d(s) + 0.582F_i(s)}{1 + 440s} \quad (9.81)$$

The block diagram of the system is shown in Figure 9.10.

Continuous state-space model: From equations (9.77)–(9.81)

$$\begin{aligned} 47\dot{t}_b + t_b(t) &= 420v_a(t) \\ 200\dot{t}_d + t_d(t) &= 0.119t_b(t) \\ 440\dot{m}_f + m_f(t) &= -0.167t_d(t) + 0.582f_i(t) \end{aligned} \quad (9.82)$$

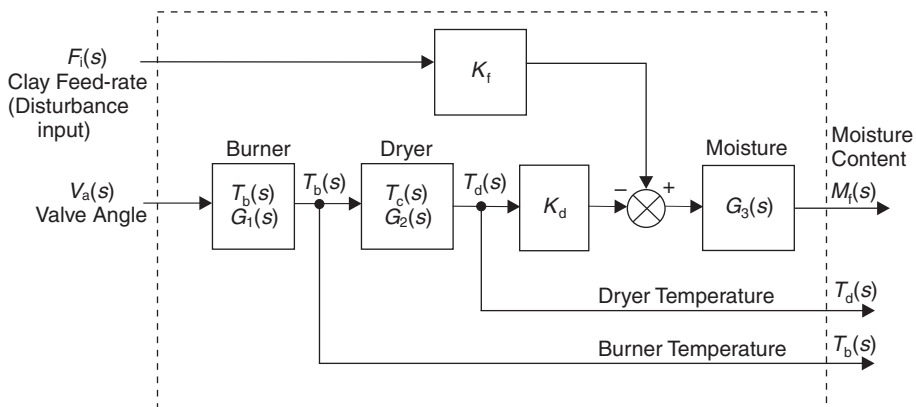


Fig. 9.10 Model of band dryer system.

Define the state variables

$$x_1 = t_b, \quad x_2 = t_d, \quad x_3 = m_f$$

and the control variables

$$u_1 = v_a$$

and the disturbance variables

$$w_1 = 0, \quad w_2 = 0, \quad w_3 = f_i$$

then equations (9.82) can now be written as

$$\begin{aligned}\dot{x}_1 &= -0.02128x_1 + 8.93617u_1 + c_{d11}w_1 \\ \dot{x}_2 &= 0.00060x_1 - 0.00500x_2 + c_{d22}w_2 \\ \dot{x}_3 &= -0.00038x_2 - 0.00227x_3 + 0.00132w_3\end{aligned}\tag{9.83}$$

where c_{d11} and c_{d22} are unknown burner and dryer disturbance coefficients, and are given an arbitrary value of 0.1. The state equations are written in the form

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{C_d} \mathbf{w}$$

$$\begin{aligned}\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} &= \begin{bmatrix} -0.0213 & 0 & 0 \\ 0.0006 & -0.005 & 0 \\ 0 & -0.00038 & -0.0023 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 8.9362 \\ 0 \\ 0 \end{bmatrix} u_1 \\ &+ \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.00132 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}\end{aligned}$$

and the output equation is

$$\begin{bmatrix} t_b \\ t_d \\ m_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\tag{9.84}$$

Discrete system model: The discrete system model (without disturbances) is given by

$$\mathbf{x}(k+1)T = \mathbf{A}(T)\mathbf{x}(kT) + \mathbf{B}(T)\mathbf{u}(kT)\tag{9.85}$$

For a sampling time of 2 seconds, from equations (8.78) and (8.80)

$$\mathbf{A}(T) = \begin{bmatrix} 0.9583 & 0 & 0 \\ 0.0012 & 0.9900 & 0 \\ 0 & -0.0008 & 0.9955 \end{bmatrix}; \quad \mathbf{B}(T) = \begin{bmatrix} 17.4974 \\ 0.0105 \\ 0 \end{bmatrix}\tag{9.86}$$

LQR Design: Using the quadratic performance index

$$J = \int_0^\infty (\mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u}) dt$$

where \mathbf{Q} and \mathbf{R} are diagonal matrices of the form

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & 0 & q_{33} \end{bmatrix}; \quad \mathbf{R} = \begin{bmatrix} r_{11} & 0 \\ 0 & r_{22} \end{bmatrix} \quad (9.87)$$

From equations (9.20) and (9.21), the optimal control law is

$$\mathbf{u}_{\text{opt}} = -\mathbf{K}\mathbf{x}$$

where

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \quad (9.88)$$

The design procedure employed was to maintain \mathbf{R} as unity scalar, and systematically vary the diagonal elements of \mathbf{Q} to achieve the performance specification. This was to maintain a dry clay moisture content of 6%, $\pm 1\%$, as the clay feed-rate varied from 6 to 10 tonnes/hour. Also the drying oven temperature t_d should not vary more than $\pm 3^\circ\text{C}$ from the set point of 50°C . At each design setting, the clay feed-rate was varied according to

$$w_3(t) = 8 + 2\sin(0.00154t) \quad (9.89)$$

Some results are presented in Table 9.1.

It was found that q_{11} had little effect, and was set to zero. From Table 9.1, the settings that meet the performance specification are

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 20 \end{bmatrix} \quad r = 1 \quad (9.90)$$

From equation (9.25), the Riccati matrix is

$$\mathbf{P} = \begin{bmatrix} 0 & 0.1 & -0.2 \\ 0.1 & 10.8 & -30 \\ -0.2 & -30 & 3670.4 \end{bmatrix} \quad (9.91)$$

which gives, from equation (9.88), the feedback gain matrix

$$\mathbf{K} = [0.0072 \quad 0.6442 \quad -1.8265] \quad (9.92)$$

The same results are also obtained from the discrete equations (9.29) and (9.30).

Table 9.1 Variations in dryer temperature and moisture content

q_{22}	q_{33}	Variation in temperature t_d ($^\circ\text{C}$)		Variation in moisture content (%)	
		Max	Min	Max	Min
3	1	0.17	0	2.09	-2.11
1	3	0.99	0	1.74	-2.13
0.5	3	1.524	0	1.5	-2.15
0.5	6	2.05	-2.05	1.27	-1.27
0.5	10	2.42	-2.42	1.1	-1.1
0.5	20	2.86	-2.86	0.89	-0.89

The closed-loop eigenvalues are

$$\begin{aligned} s &= -0.0449 \pm j0.0422 \\ s &= -0.0033 \end{aligned} \quad (9.93)$$

Implementation: The optimal control law was implemented by using

$$\mathbf{u}_1 = \mathbf{K}\mathbf{e}$$

where

$$\mathbf{e} = (\mathbf{r} - \mathbf{x}) \quad (9.94)$$

This is shown in Figure 9.11.

A discrete simulation was undertaken using equations (9.85) and (9.86) together with a disturbance transition matrix $\mathbf{C}_d(T)$, which was calculated using \mathbf{C}_d in equation (9.84) and equation (8.80) for $\mathbf{B}(T)$, with a sampling time of 2 seconds.

$$\mathbf{C}_d(T) = \begin{bmatrix} 0.1958 & 0 & 0 \\ 0.0001 & 0.199 & 0 \\ 0 & -0.0001 & 0.0026 \end{bmatrix} \quad (9.95)$$

The desired state vector was

$$\mathbf{r} = \begin{bmatrix} 450 \\ 50 \\ -6 \end{bmatrix} \quad (9.96)$$

Note that the moisture content r_3 is negative because of the moisture model in equation (9.79). The initial conditions were

$$\mathbf{x}(0) = \begin{bmatrix} 200 \\ 30 \\ -30 \end{bmatrix} \quad (9.97)$$

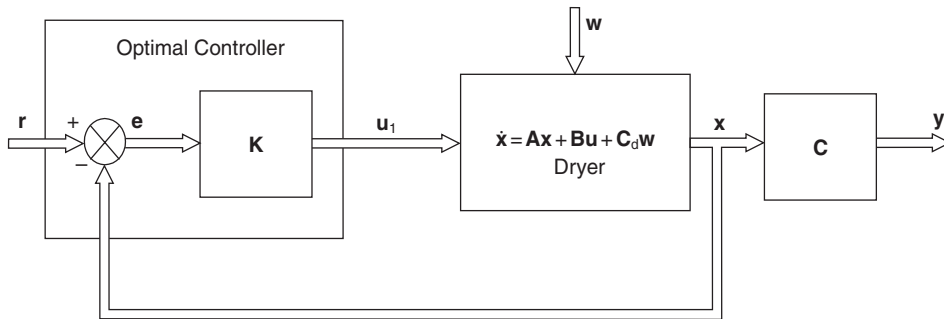


Fig. 9.11 Optimal control of band dryer.

and the disturbance vector

$$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ w_3 \end{bmatrix} \quad (9.98)$$

where w_3 , the clay feed-rate was set at a value between 6 and 10 tonnes/hour. Figure 9.12 shows the time response of $u_1(t)$, the gas-valve angle in radians. The valve angle was not allowed to have a negative value, so remained closed for the first 80 seconds of the simulation, when the dryer was cooling. The steady-state angle was 0.95 radians, or 54° .

Figure 9.13 indicates the burner temperature time response $t_b(t)$. The temperature falls from its initial value, since the gas valve is closed, and then climbs with a response indicated by the eigenvalues in equation (9.93) to a steady-state value of 400°C , or a steady-state error of 50°C .

Figure 9.14 shows the combined response of the dryer temperature $t_d(t)$ and the moisture content $m_f(t)$, the latter being shown as a positive number. The dryer temperature climbs to 48°C (steady-state error of 2°C) and the moisture falls to 6%, with no steady-state error. In this simulation the clay feed-rate $w_3(t)$ was constant at 8 tonnes/hour.

As the band dryer is a type zero system, and there are no integrators in the controller, steady-state errors must be expected. However, the selection of the elements in the \mathbf{Q} matrix, equation (9.90), focuses the control effort on control-

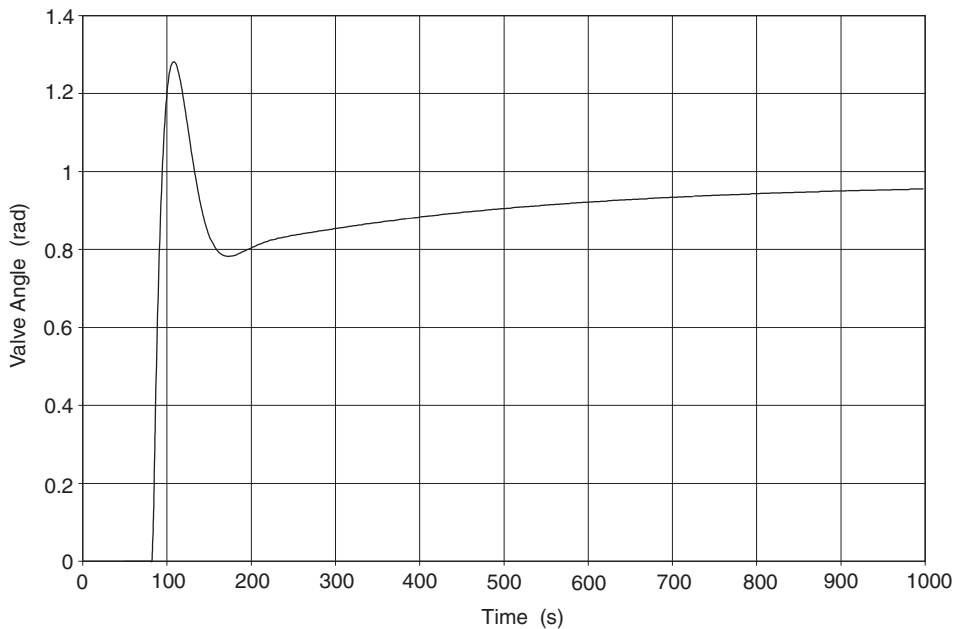


Fig. 9.12 Time response of gas-valve angle $u_1(t)$.

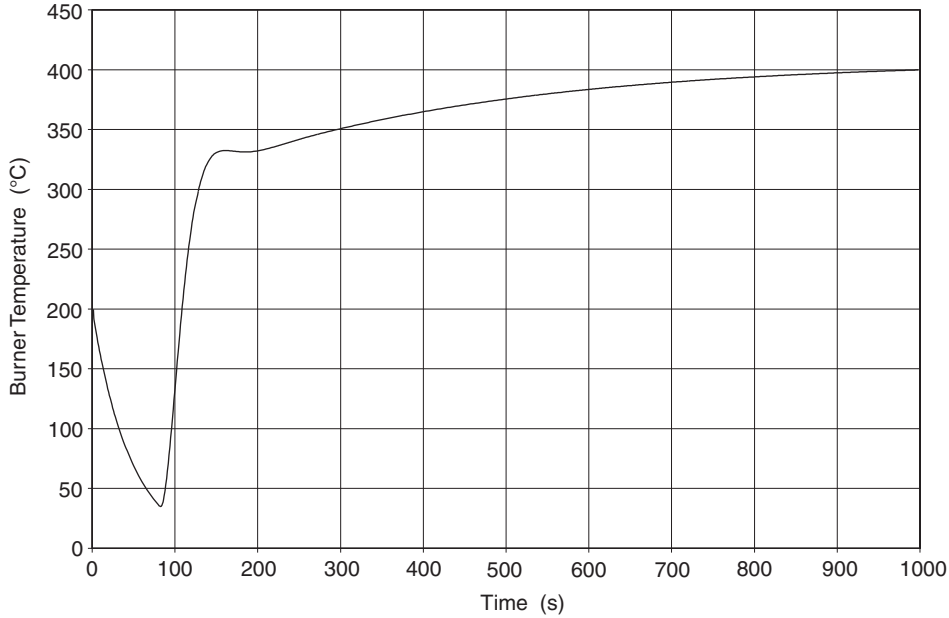


Fig. 9.13 Time response of burner temperature $t_b(t)$.

ling the moisture content, at the expense of, in particular, the burner temperature $t_b(t)$.

Figure 9.15 shows the final 500 seconds of the moisture content simulation as the clay feed-rate is varied between 6 and 10 tonnes/hour. After 1 000 seconds, as the clay leaves the dryer, the moisture content is between 5.2% and 6.8%, which is within the specification of $\pm 1\%$ of the set point of 6%.

Kalman filter design: If the three stages of the covariance matrix \mathbf{P} are written as $\mathbf{P}(k/k) = \mathbf{P}_1$; $\mathbf{P}(k+1/k) = \mathbf{P}_2$ and $\mathbf{P}(k+1/k+1) = \mathbf{P}_3$, then recursive equations (9.74), (9.75) and (9.76) become

$$\begin{aligned}\mathbf{P}_2 &= \mathbf{A}\mathbf{P}_1\mathbf{A}^T + \mathbf{C}_d\mathbf{Q}\mathbf{C}_d^T \\ \mathbf{K} &= \mathbf{P}_2\mathbf{C}^T\{\mathbf{C}\mathbf{P}_2\mathbf{C}^T + \mathbf{R}\}^{-1} \\ \mathbf{P}_3 &= \{\mathbf{I} - \mathbf{K}\mathbf{C}\}\mathbf{P}_2\end{aligned}\tag{9.99}$$

Equation set (9.99) is simpler to visualize, but remember the system matrices are the transition matrices $\mathbf{A}(T)$, $\mathbf{B}(T)$ and $\mathbf{C}_d(T)$. Before recursion can start, values for \mathbf{R} , the measurement noise covariance matrix, and \mathbf{Q} , the disturbance noise covariance matrix must be selected.

Measurement noise covariance matrix \mathbf{R} : The main problem with the instrumentation system was the randomness of the infrared absorption moisture content analyser. A number of measurements were taken from the analyser and compared with samples taken simultaneously by work laboratory staff. The errors could be approximated to a normal distribution with a standard deviation of 2.73%, or a variance of 7.46.

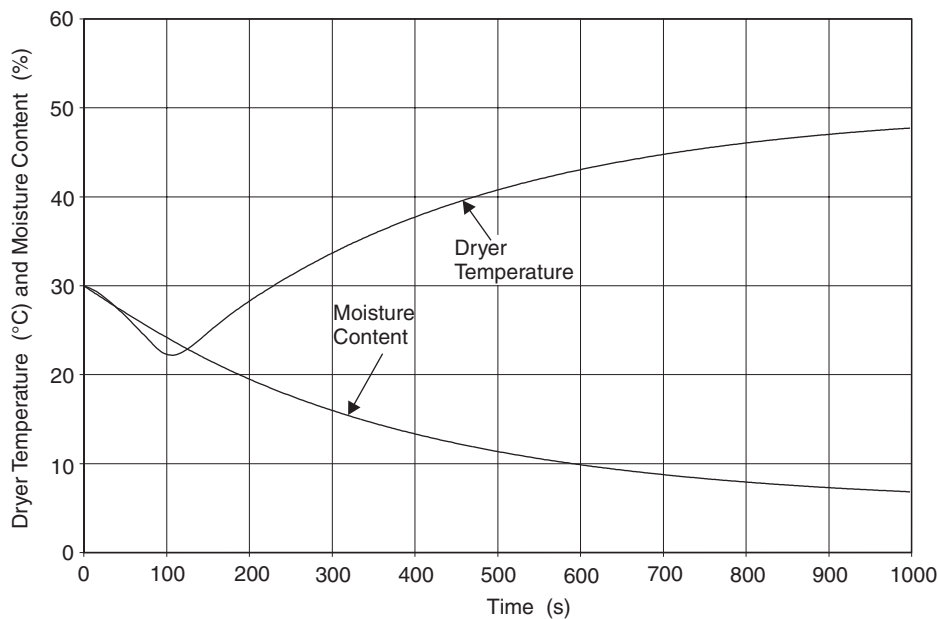


Fig. 9.14 Combined response of dryer temperature $t_d(t)$ and moisture content $m_f(t)$.

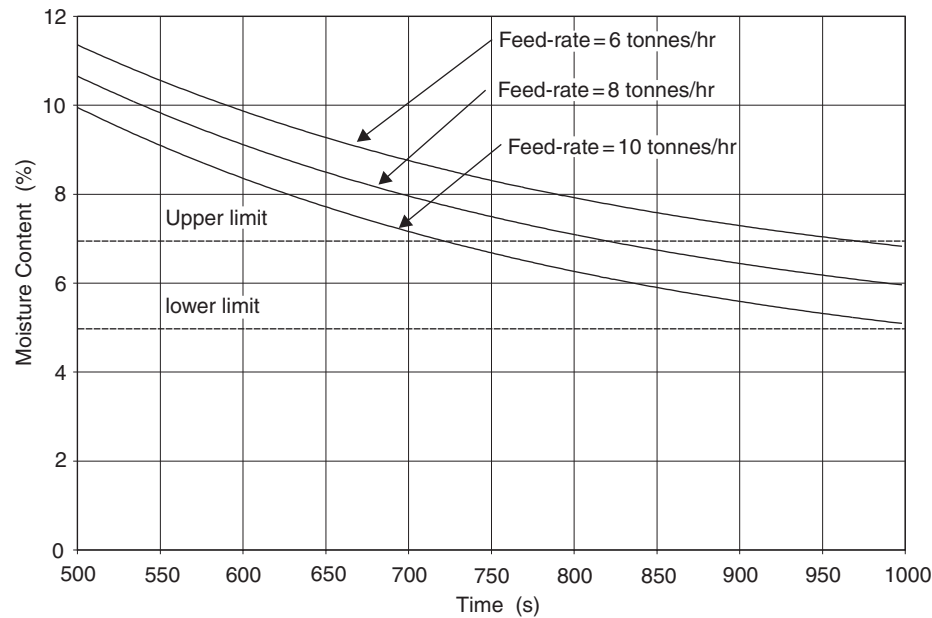


Fig. 9.15 Effect of varying clay feed-rate.

The thermocouples measuring the burner and dryer temperatures were relatively noise-free, with standard deviations in the order of 0.1°C . The variance was therefore set at 0.01, giving

$$\mathbf{R} = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 7.46 \end{bmatrix} \quad (9.100)$$

Disturbance noise covariance matrix Q: This was set as a diagonal matrix, where q_{11} and q_{22} represent changes in the burner and dryer temperatures as a result of changing heat transfer through the walls of the dryer, due to wind and variations in external temperature.

On the other hand, q_{33} is a measure of clay feed-rate variations, and a standard deviation of 0.3 tonnes/hour seemed appropriate. In the absence of any other information, standard deviations of the burner and dryer temperatures was also thought to be in the order of 0.3°C . Thus, when these values are squared, the \mathbf{Q} matrix becomes

$$\mathbf{Q} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \quad (9.101)$$

Before equations (9.99) can be run, an initial value of $\mathbf{P}(k/k)$ is required. Ideally, they should not be close to the final value, so that convergence can be seen to have taken place. In this instance, $\mathbf{P}(k/k)$ was set to an identity matrix. Figure 9.16 shows the diagonal elements of the Kalman gain matrix during the first 20 steps of the recursive equation (9.99).

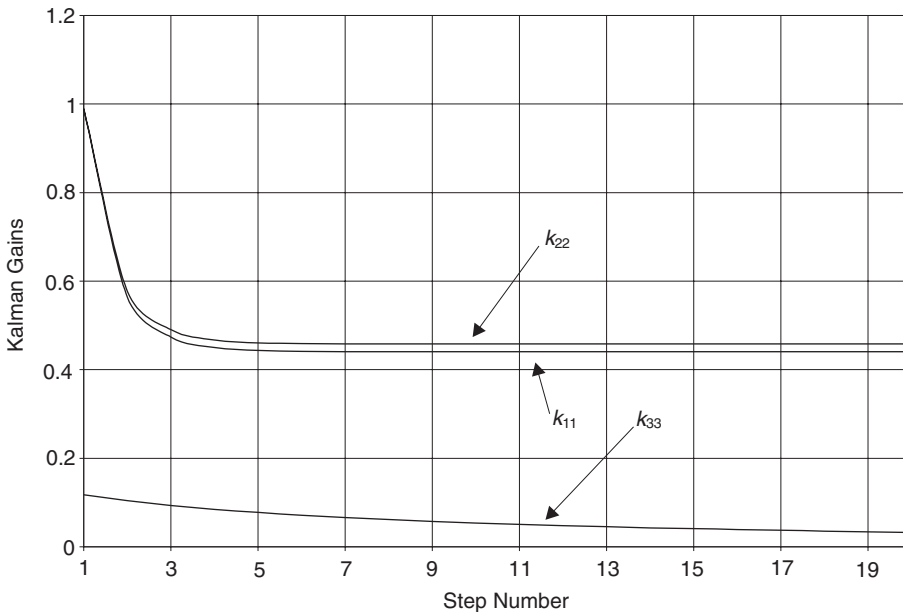


Fig. 9.16 Convergence of diagonal elements of Kalman gain matrix.

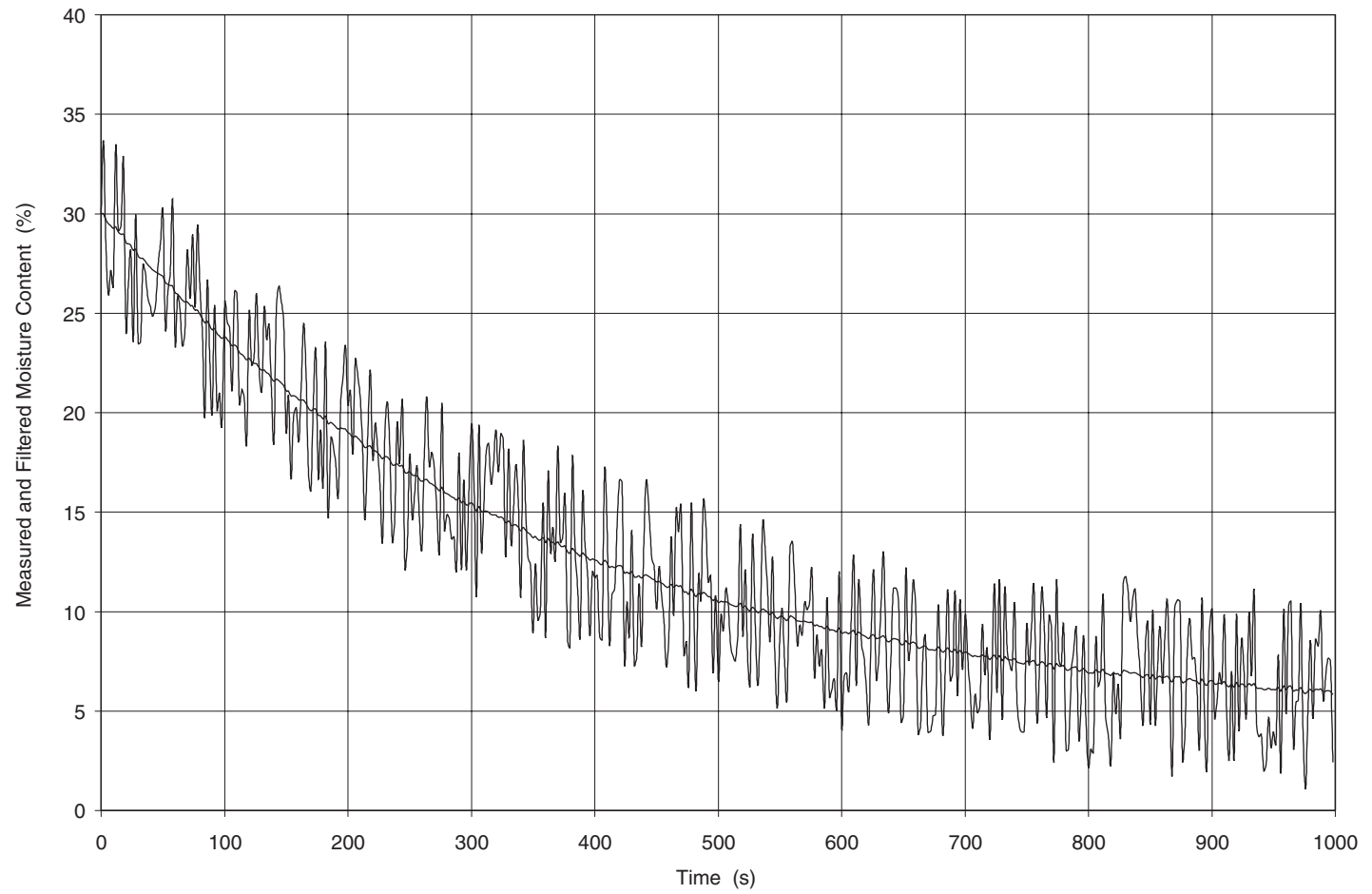


Fig. 9.17 Measured and filtered clay moisture content.

The final values of the Kalman Gain matrix \mathbf{K} and covariance matrix \mathbf{P} were

$$\mathbf{K} = \begin{bmatrix} 0.4408 & 0.0003 & 0 \\ 0.0003 & 0.4579 & 0 \\ 0 & -0.0006 & 0.0325 \end{bmatrix}; \quad \mathbf{P} = \begin{bmatrix} 0.0044 & 0 & 0 \\ 0 & 0.0046 & 0 \\ 0 & 0 & 0.2426 \end{bmatrix} \quad (9.102)$$

The full LQG system, comprising of the LQ optimal controller and Kalman filter was then constructed. Figure 9.17 shows a set of moisture content measurements $z_3(kT)$ together with the estimated moisture content $\hat{x}_3(kT)$.

9.6 Robust control

9.6.1 Introduction

The robust control problem is to find a control law which maintains system response and error signals within prescribed tolerances despite the effects of uncertainty on the system. Forms of uncertainty include

- Disturbance effects on the plant
- Measurement noise
- Modelling errors due to nonlinearities
- Modelling errors due to time-varying parameters

In previous chapters, linear models have been used to describe plant dynamics. However, in section 2.7.1 it was demonstrated that nonlinear functions could be linearized for small perturbations about an operating point. It is therefore possible to describe a nonlinear system by a series of linear models each constructed about a known operating point. If the operating point can be linked to a measurement, then a simple robust system may be constructed using an LQG approach. The feedback and Kalman gain matrices are calculated in advance for each operating point and some form of interpolation used to provide a 'Gain Scheduling Controller.'

The disturbance and measurement noise is taken into account by the Kalman filter. In the following example, undertaken by the author (1984), a non-linear simulation of a ship of length 161 m and displacement 17 000 tonnes was given a series of step changes in demanded rudder-angle at forward speeds of 2.572 m/s (5 knots), 5.145 m/s (10 knots) and 7.717 m/s (15 knots). At each forward speed a linear model was constructed and the \mathbf{Q} and \mathbf{R} matrices in an LQG implementation selected to return the closed-loop eigenvalues back to some desired value (Ackermann's formula could not be used since $\mathbf{y}(t)$ and $\mathbf{u}(t)$ were vector, not scalar quantities).

A subset of the state error variables is

$e_1(t)$ = cross-track position error

$e_2(t)$ = cross-track velocity error

$e_3(t)$ = heading error

$e_4(t)$ = heading-rate error

The feedback control is of the form

$$\mathbf{u}_{\text{opt}} = \mathbf{K}\mathbf{e}$$

where the values of \mathbf{K} for the three forward speeds are

$$\begin{aligned}\mathbf{K}_{2.572} &= [0.0121 \quad 1.035 \quad 7.596 \quad 160.26] \\ \mathbf{K}_{5.145} &= [0.0029 \quad 0.3292 \quad 1.81 \quad 25.963] \\ \mathbf{K}_{7.717} &= [0.0013 \quad 0.1532 \quad 0.8419 \quad 8.047]\end{aligned}\quad (9.103)$$

If the forward velocity of the ship is the state variable u_s , a best estimate of which is given by the Kalman filter, the gain scheduling controller can be expressed as

$$\begin{aligned}k_1 &= 0.08u_s^{-2.0} \\ k_2 &= 6.0u_s^{-1.8} \\ k_3 &= 50.0u_s^{-2.0} \\ k_4 &= 2090.0u_s^{-2.72}\end{aligned}\quad (9.104)$$

Equation set (9.104) approximates to an inverse square law, and increases the controller gains at low speeds, where the control surfaces are at their most insensitive.

In general, however, robust control system design uses an idealized, or nominal model of the plant $G_m(s)$. Uncertainty in the nominal model is taken into account by considering a family of models that include all possible variations. The control system is said to have *robust stability* if a controller can be found that will stabilize all plants within the family. However, on its own, robust stability is not enough, since there may be certain plants within the family that are on the verge of instability. A controller is said to have *robust performance* if all the plants within the family meet a given performance specification.

9.6.2 Classical feedback control

Figure 9.18 shows a classical feedback control system $D(s)$ is a disturbance input, $N(s)$ is measurement noise, and therefore

$$\begin{aligned}Y(s) &= G(s)U(s) + D(s) \\ B(s) &= Y(s) + N(s) \\ U(s) &= C(s)(R(s) - B(s))\end{aligned}\quad (9.105)$$

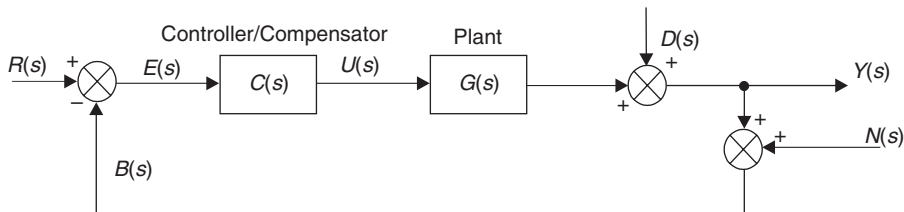


Fig. 9.18 Classical feedback control system.

Eliminating $U(s)$ and $B(s)$ from equations (9.105) gives

$$Y(s) = \frac{G(s)C(s)R(s)}{1 + G(s)C(s)} + \frac{D(s)}{1 + G(s)C(s)} - \frac{G(s)C(s)N(s)}{1 + G(s)C(s)} \quad (9.106)$$

Define a sensitivity function $S(s)$ that relates $Y(s)$ and $D(s)$ when $R(s) = N(s) = 0$

$$\frac{Y}{D}(s) = S(s) = \frac{1}{1 + G(s)C(s)} \quad (9.107)$$

and define a complementary sensitivity function

$$T(s) = 1 - S(s) = \frac{G(s)C(s)}{1 + G(s)C(s)} \quad (9.108)$$

Thus, when $N(s) = 0$, equation (9.106) may be written

$$Y(s) = T(s)R(s) + S(s)D(s) \quad (9.109)$$

If $T(s) = 1$ and $S(s) = 0$ there is perfect set-point tracking and disturbance rejection. This requires that $G(s)C(s)$ is strictly proper (has more poles than zeros), so that

$$\lim_{s \rightarrow \infty} G(s)C(s) = 0 \quad (9.110)$$

However, if $N(s) \neq 0$, then equation (9.106) becomes

$$Y(s) = T(s)R(s) + S(s)D(s) - T(s)N(s) \quad (9.111)$$

Hence, if $T(s) = 1$, there will be both perfect set-point tracking and noise acceptance. Considering the problem in the frequency domain however, it may be possible that at low frequencies $T(j\omega) \rightarrow 1$ (good set-point tracking) and at high frequencies $T(j\omega) \rightarrow 0$ (good noise rejection).

9.6.3 Internal Model Control (IMC)

Consider the system shown in Figure 9.19 $G(s)$ is the plant, $G_m(s)$ is the nominal model, $R(s)$ is the desired value, $U(s)$ is the control, $D(s)$ is a disturbance input, $Y(s)$ is the output and $N(s)$ is the measurement noise. $C(s)$ is called the IMC controller and is to be designed so that $y(t)$ is kept as close as possible to $r(t)$ at all times.

From Figure 9.19, the feedback signal $B(s)$ is

$$B(s) = G(s)U(s) + D(s) + N(s) - G_m(s)U(s)$$

or

$$B(s) = (G(s) - G_m(s))U(s) + D(s) + N(s) \quad (9.112)$$

If, in equation (9.112) the model is exact, i.e. $G_m(s) = G(s)$ and the disturbance $D(s)$ and noise $N(s)$ are both zero, then $B(s)$ is also zero and the control system is effectively open-loop. This is the condition when there is no uncertainty. However, if $G_m(s) \neq G(s)$, and $D(s)$ and $N(s)$ are not zero, then $B(s)$ expresses the uncertainty of the process.

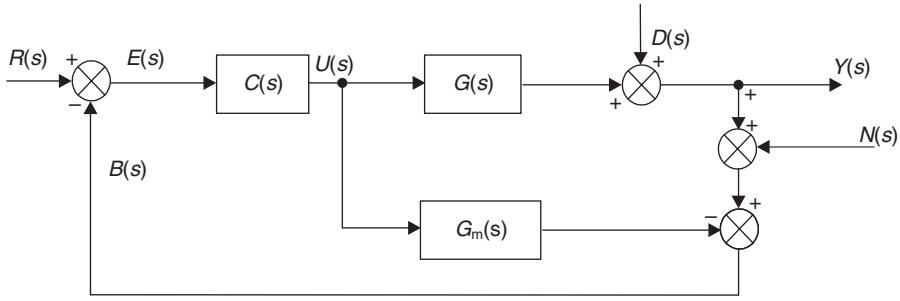


Fig. 9.19 Block diagram of an IMC system.

9.6.4 IMC performance

From Figure 9.19

$$\begin{aligned} Y(s) &= G(s)U(s) + D(s) \\ B(s) &= Y(s) + N(s) - G_m(s)U(s) \\ U(s) &= C(s)(R(s) - B(s)) \end{aligned} \quad (9.113)$$

Eliminating $U(s)$ and $B(s)$ from equations (9.113) gives

$$Y(s) = \frac{G(s)C(s)R(s)}{1 + C(s)(G(s) - G_m(s))} + \frac{(1 - C(s)G_m(s))D(s)}{1 + C(s)(G(s) - G_m(s))} - \frac{G(s)C(s)N(s)}{1 + C(s)(G(s) - G_m(s))} \quad (9.114)$$

The sensitivity function $S(s)$ that relates $Y(s)$ and $D(s)$ when $R(s) = N(s) = 0$ is

$$\frac{Y}{D}(s) = S(s) = \frac{1 - C(s)G_m(s)}{1 + C(s)(G(s) - G_m(s))} \quad (9.115)$$

and the complementary sensitivity function

$$T(s) = 1 - S(s) = \frac{C(s)G(s)}{1 + C(s)(G(s) - G_m(s))} \quad (9.116)$$

Thus, when $N(s) = 0$, equation (9.114) may be written

$$Y(s) = T(s)R(s) + S(s)D(s) \quad (9.117)$$

If $T(s) = 1$ there is perfect set-point tracking. This will occur if $G_m(s) = G(s)$ and $C(s) = 1/G(s)$. If $S(s) = 0$ there is perfect disturbance rejection. Again, this will occur if $G_m(s) = G(s)$ and $C(s) = 1/G_m(s)$.

Two degree-of-freedom IMC system

If good set-point tracking and good disturbance rejection is required when the dynamic characteristics of $R(s)$ and $D(s)$ are substantially different, then it may be

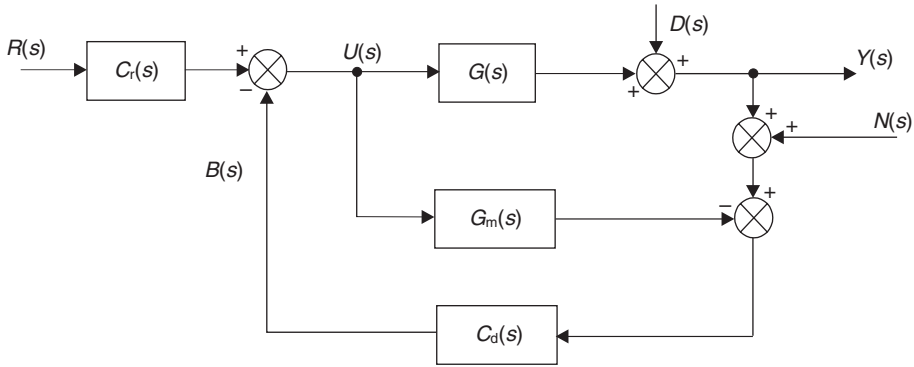


Fig. 9.20 Two degree-of-freedom IMC system.

necessary to introduce a second controller, which provides a second degree-of-freedom of control action. A two degree-of-freedom IMC system is shown in Figure 9.20.

With a two degree-of-freedom IMC system, equation (9.114) becomes

$$Y(s) = \frac{G(s)C_r(s)R(s)}{1 + C_d(s)(G(s) - G_m(s))} + \frac{(1 - C_d(s)G_m(s))D(s)}{1 + C_d(s)(G(s) - G_m(s))} - \frac{G(s)C_d(s)N(s)}{1 + C_d(s)(G(s) - G_m(s))} \quad (9.118)$$

In equation (9.118) $C_r(s)$ is designed for set-point tracking and $C_d(s)$ for disturbance rejection.

9.6.5 Structured and unstructured model uncertainty

Unstructured model uncertainty relates to unmodelled effects such as plant disturbances and are related to the nominal plant $G_m(s)$ as either additive uncertainty $\ell_a(s)$

$$G(s) = G_m(s) + \ell_a(s) \quad (9.119)$$

or multiplicative uncertainty $\ell_m(s)$

$$G(s) = (1 + \ell_m(s))G_m(s) \quad (9.120)$$

Equating (9.119) and (9.120) gives

$$\ell_a(s) = \ell_m(s)G_m(s) \quad (9.121)$$

Block diagram representations of additive and multiplicative model uncertainty are shown in Figure 9.21.

Structured uncertainty relates to parametric variations in the plant dynamics, i.e. uncertain variations in coefficients in plant differential equations.

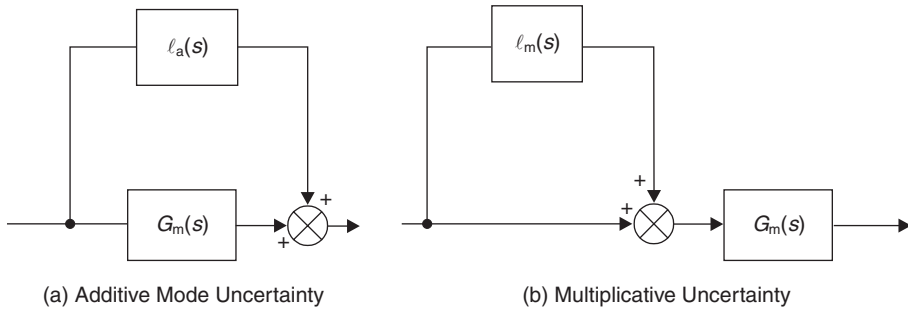


Fig. 9.21 Additive and multiplicative model uncertainty.

9.6.6 Normalized system inputs

All inputs to the control loop (changes in set-point or disturbances) are generically represented by $V(s)$. The input $V(s)$ is found by passing a mathematically bounded normalized input $V^1(s)$ through a transfer function block $W(s)$, called the input weight, as shown in Figure 9.22.

Specific inputs

$$\begin{array}{lll} \text{Impulse} & V^1(s) = 1 & W(s) = 1 \\ \text{Step} & V^1(s) = 1 & W(s) = 1/s \end{array} \quad (9.122)$$

Thus for specific inputs

$$V(s) = W(s)V^1(s) = W(s) \quad (9.123)$$

Sets of bounded inputs may be represented by

$$\|v^1(t)\|_2^2 = \int_0^\infty (v^1(t))^2 dt \leq 1 \quad (9.124)$$

The left-hand side of equation (9.124) is called ‘the 2-norm of the input signal $v^1(t)$ squared’. Norms are mathematical measures that enable objects belonging to the



Fig. 9.22 Transformation of a normalized bounded input $V^1(s)$ into an actual input $V(s)$.

same set to be compared. Using Parseval's theorem, equation (9.124) may be transformed into the frequency domain

$$\|v^1(t)\|_2^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} \left| \frac{V(j\omega)}{W(j\omega)} \right|^2 d\omega \leq 1 \quad (9.125)$$

For a realizable controller to exist, all external signals that enter the control loop must be bounded.

9.7 H_2 - and H_∞ -optimal control

9.7.1 Linear quadratic H_2 -optimal control

The scalar version of equation (9.48), when $\mathbf{u}(t)$ is not constrained, and Q is unity, is called the Integral Squared Error (ISE), i.e.

$$\text{ISE} = \int_{t_0}^{t_1} e^2(t) dt \quad (9.126)$$

The H_2 -optimal control problem is to find a controller $c(t)$ such that the 2-norm of the ISE (written $\|e(t)\|_2^2$) is minimized for just one specific input $v(t)$.

If, in equation set (9.105) $B(s)$ and $Y(s)$ are eliminated and $U(s)$ is written as $C(s)E(s)$, then

$$\begin{aligned} E(s) &= \frac{1}{1 + G(s)C(s)} \{R(s) - D(s) - N(s)\} \\ &= S(s) \{R(s) - D(s) - N(s)\} \end{aligned} \quad (9.127)$$

Also, from equation (9.123), for a specific input

$$V(s) = W(s) \quad (9.128)$$

Using Parseval's theorem, from equation (9.126) the H_2 -optimal control problem can be expressed in the frequency domain as

$$\min_c \|e(t)\|_2^2 = \min_c \frac{1}{2\pi} \int_{-\infty}^{\infty} |E(j\omega)|^2 d\omega \quad (9.129)$$

Substituting equations (9.127) and (9.128) into (9.129) gives

$$\min_c \|e(t)\|_2^2 = \min_c \frac{1}{2\pi} \int_{-\infty}^{\infty} |S(j\omega)W(j\omega)|^2 d\omega \quad (9.130)$$

Thus the H_2 -optimal controller minimizes the average magnitude of the sensitivity function weighted by $W(j\omega)$, where $W(j\omega)$ depends upon the specific input $V(j\omega)$. In mathematical terms, the controller minimizes the 2-norm of the sensitivity function weighted by $W(j\omega)$.

9.7.2 H_∞ -optimal control

With H_∞ -optimal control the inputs $V(j\omega)$ are assumed to belong to a set of norm-bounded functions with weight $W(j\omega)$ as given by equation (9.125). Each input $V(j\omega)$ in the set will result in a corresponding error $E(j\omega)$. The H_∞ -optimal controller is designed to minimise the worst error that can arise from any input in the set, and can be expressed as

$$\min_c \|e(t)\|_\infty = \min_c \sup_\omega |S(j\omega)W(j\omega)| \quad (9.131)$$

In equation (9.131), sup is short for supremum, which means the final result is the least upper bound. Thus the H_∞ -optimal controller minimizes the maximum magnitude of the weighted sensitivity function over frequency range ω , or in mathematical terms, minimizes the ∞ -norm of the sensitivity function weighted by $W(j\omega)$.

9.8 Robust stability and robust performance

9.8.1 Robust stability

Robust stability can be investigated in the frequency domain, using the Nyquist stability criterion, defined in section 6.4.2.

Consider a Nyquist contour for the nominal open-loop system $G_m(j\omega)C(j\omega)$ with the model uncertainty given by equation (9.119). Let $\bar{\ell}_a(\omega)$ be the bound of additive uncertainty and therefore be the radius of a disk superimposed upon the nominal Nyquist contour. This means that $G(j\omega)$ lies within a family of plants $\pi(G(j\omega) \in \pi)$ described by the disk, defined mathematically as

$$\pi = \{G: |G(j\omega) - G_m(j\omega)| \leq \bar{\ell}_a(\omega)\} \quad (9.132)$$

and therefore

$$|\ell_a(j\omega)| \leq \bar{\ell}_a(\omega) \quad (9.133)$$

If the multiplicative uncertainty in equations (9.120) and (9.121) is defined as

$$\ell_m(j\omega) = \frac{\ell_a(j\omega)}{G_m(j\omega)C(j\omega)} \quad (9.134)$$

and the bound of multiplicative uncertainty

$$\bar{\ell}_m(\omega) = \frac{\bar{\ell}_a(\omega)}{|G_m(j\omega)C(j\omega)|} \quad (9.135)$$

From equation (9.135) the disk radius (bound of uncertainty) is

$$\bar{\ell}_a(\omega) = |G_m(j\omega)C(j\omega)|\bar{\ell}_m(\omega) \quad (9.136)$$

From the Nyquist stability criterion, let $N(k, G(j\omega))$ be the net number of clockwise encirclements of a point $(k, 0)$ of the Nyquist contour. Assume that all plants in the family π , expressed in equation (9.132) have the same number (n) of right-hand plane (RHP) poles.

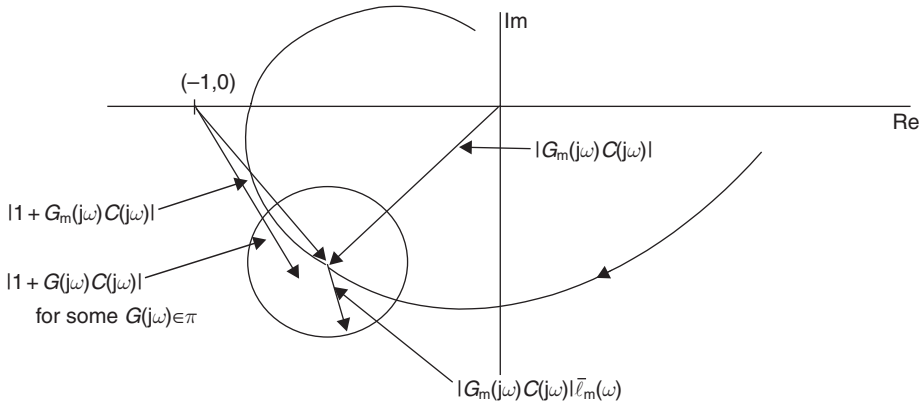


Fig. 9.23 Robust stability.

There will be robust stability of a specific controller $C(j\omega)$ if and only if

$$N(-1, G(j\omega)C(j\omega)) = -n \quad \text{for all } G(j\omega) \in \pi \quad (9.137)$$

It is also necessary for the nominal plant $G_m(j\omega)$ to be stable

$$N(-1, G_m(j\omega)C(j\omega)) = -n \quad (9.138)$$

From Figure 9.23 robust stability occurs when the vector magnitude $|1 + G_m(j\omega)C(j\omega)|$ (see also Figure 6.25) exceeds the disk radius $|G_m(j\omega)C(j\omega)|\bar{\ell}_m(\omega)$

$$|1 + G_m(j\omega)C(j\omega)| > |G_m(j\omega)C(j\omega)|\bar{\ell}_m(\omega) \quad \text{for all } \omega$$

or

$$\left| \frac{G_m(j\omega)C(j\omega)}{1 + G_m(j\omega)C(j\omega)} \right| \bar{\ell}_m(\omega) < 1 \quad (9.139)$$

Equation (9.139) uses the magnitude of the complementary sensitivity function $T(j\omega)$ as defined in equation (9.108). Thus

$$|T(j\omega)|\bar{\ell}_m(\omega) < 1 \quad \text{for all } \omega \quad (9.140)$$

Robust stability can therefore be stated as: ‘If all plants $G(s)$ in the family π have the same number of RHP poles and that a particular controller $C(s)$ stabilizes the nominal plant $G_m(s)$, then the system is robustly stable with the controller $C(s)$ if and only if the complementary sensitivity function $T(s)$ for the nominal plant $G_m(s)$ satisfies the following bound

$$\|T(j\omega)\bar{\ell}_m(\omega)\|_\infty = \sup_{\omega} |T(j\omega)\bar{\ell}_m(\omega)| < 1 \quad (9.141)$$

where the LHS of equation (9.141) is the infinity norm of $T(j\omega)\bar{\ell}_m(j\omega)$. This means that robust stability imposes a bound on the ∞ norm of the complementary sensitivity function $T(j\omega)$ weighted by $\bar{\ell}_m(\omega)$.

9.8.2 Robust performance

Robust stability provides a minimum requirement in an environment where there is plant model uncertainty. For a control system to have robust performance it should be capable of minimizing the error for the *worst* plant (i.e. the one giving the largest error) in the family $G(j\omega) \in \pi$.

For the H_∞ -control problem, from equation (9.131), the ∞ -norm of the weighted sensitivity function can be written

$$\|SW\|_\infty = \sup_{\omega} |S(j\omega)W(j\omega)| \quad (9.142)$$

If, as part of the design process, a bound is placed upon the sensitivity function

$$|S(j\omega)| < |W(j\omega)|^{-1} \quad (9.143)$$

Should an H_∞ controller be found such that

$$\|SW\|_\infty < 1 \quad (9.144)$$

then the bound in equation (9.143) is met. Hence, for robust performance

$$\|SW\|_\infty = \sup_{\omega} |S(j\omega)W(j\omega)| < 1 \quad \text{for all } G(j\omega) \in \pi \quad (9.145)$$

From Figure 9.23 representing robust stability, the actual frequency response $G(j\omega)C(j\omega)$ will always lie inside the region of uncertainty denoted by the disk, or

$$|1 + G(j\omega)C(j\omega)| \geq |1 + G_m(j\omega)C(j\omega)| - |G_m(j\omega)C(j\omega)|\bar{\ell}_m(\omega) \quad \text{for all } G(j\omega) \in \pi \quad (9.146)$$

giving

$$|S(j\omega)| = \left| \frac{1}{1 + G(j\omega)C(j\omega)} \right| \leq \frac{|S_m(j\omega)|}{1 - |T_m(j\omega)|\bar{\ell}_m(\omega)} \quad \text{for all } G(j\omega) \in \pi \quad (9.147)$$

where $S_m(j\omega)$ is the sensitivity function for the nominal plant

$$S_m(j\omega) = \frac{1}{1 + G_m(j\omega)C(j\omega)} \quad (9.148)$$

Using equation (9.147), equation (9.145) can be expressed as

$$\frac{|S_m(j\omega)W(j\omega)|}{1 - |T_m(j\omega)|\bar{\ell}_m(\omega)} < 1 \quad \text{for all } \omega \quad (9.149)$$

or

$$|T_m(j\omega)\bar{\ell}_m(\omega)| + |S_m(j\omega)W(j\omega)| < 1 \quad \text{for all } \omega \quad (9.150)$$

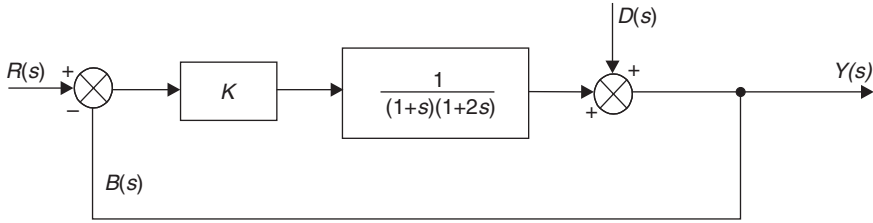


Fig. 9.24 Control system.

Robust performance then means that the closed-loop system will meet the performance specification given in equation (9.145) if and only if the nominal system is closed-loop stable (equation (9.141)) and that the sensitivity function $S_m(j\omega)$ and complementary sensitivity function $T_m(j\omega)$ for the nominal system satisfy the relationship given in equation (9.150).

Example 9.4

- For the control system shown in Figure 9.24 produce the Bode magnitude plots for the sensitivity function $|S(j\omega)|$ and the complementary sensitivity function $|T(j\omega)|$ when $K = 10$. Comment on their values.
- For a step input, let $W(s) = 1/s$. Produce Bode magnitude plots for $|S(j\omega)W(j\omega)|$ when $K = 10, 50$ and 100 and identify the optimal value using both H_2 and H_∞ criteria.

Use a frequency range of 0.01–100 rad/s for both (a) and (b).

Solution

- From equation (9.107)

$$\begin{aligned} S(s) &= \frac{1}{1 + G(s)C(s)} = \frac{1}{1 + \frac{K}{(1+s)(1+2s)}} \\ &= \frac{2s^2 + 3s + 1}{2s^2 + 3s + (1 + K)} \end{aligned} \quad (9.151)$$

From equation (9.108)

$$\begin{aligned} T(s) &= 1 - S(s) = 1 - \left\{ \frac{2s^2 + 3s + 1}{2s^2 + 3s + (1 + K)} \right\} \\ &= \frac{K}{2s^2 + 3s + (1 + K)} \end{aligned} \quad (9.152)$$

The Bode magnitude plots for $|S(j\omega)|$ and $|T(j\omega)|$ are shown in Figure 9.25 for $K = 10$. From Figure 9.25 it can be seen that up to 1 rad/s, the system has a set-point tracking error of -0.8 dB ($|T(j\omega)|$) and a disturbance rejection of -20 dB ($|S(j\omega)|$).

- (b) For a specific input of a unit step, let $W(s) = 1/s$. Hence the weighted sensitivity function is

$$S(s)W(s) = \frac{2s^2 + 3s + 1}{s\{2s^2 + 3s + (1 + K)\}} \quad (9.153)$$

The Bode magnitude plots for $|S(j\omega)W(j\omega)|$ for $K = 10, 50$ and 100 are shown in Figure 9.26.

From Figure 9.26 it can be seen that the H_2 -norm, or average value of the weighted sensitivity function (equation (9.130)) reduces as K increases and hence, using this criteria, $K = 100$ is the best value. Using the H_∞ -norm as defined in equation (9.131), the maximum magnitude of the weighted sensitivity function occurs at the lowest frequency. The least upper bound therefore is 0 dB, occurring at 0.01 rad/s when $K = 100$, so this again is the best value.

Example 9.5

A closed-loop control system has a nominal forward-path transfer function equal to that given in Example 6.4, i.e.

$$G_m(s)C(s) = \frac{K}{s(s^2 + 2s + 4)}$$

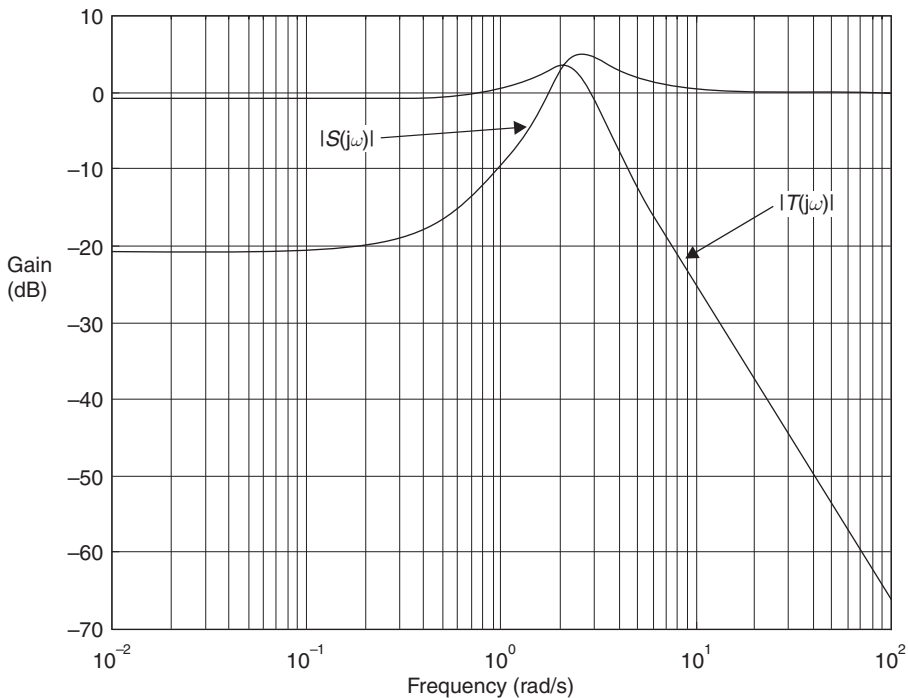


Fig. 9.25 Bode magnitude plots for $|S(j\omega)|$ and $|T(j\omega)|$.

Let the bound of the multiplicative model uncertainty be

$$\bar{\ell}_m(s) = \frac{0.5(1+s)}{(1+0.25s)}$$

What is the maximum value that K can have for robust stability?

Solution

At frequencies below 1 rad/s, $\bar{\ell}_m(\omega) \rightarrow 0.5$ and at frequencies above 4 rad/s $\bar{\ell}_m(\omega) \rightarrow 2.0$. From equation (9.141), for robust stability

$$|T(j\omega)\bar{\ell}_m(\omega)| < 1 \quad (9.154)$$

now

$$T(s) = \frac{G_m(s)C(s)}{1 + G_m(s)C(s)}$$

therefore

$$T(s) = \frac{\frac{K}{s^3+2s^2+4s}}{1 + \frac{K}{s^3+2s^2+4s}}$$

$$T(s) = \frac{K}{s^3 + 2s^2 + 4s + K}$$

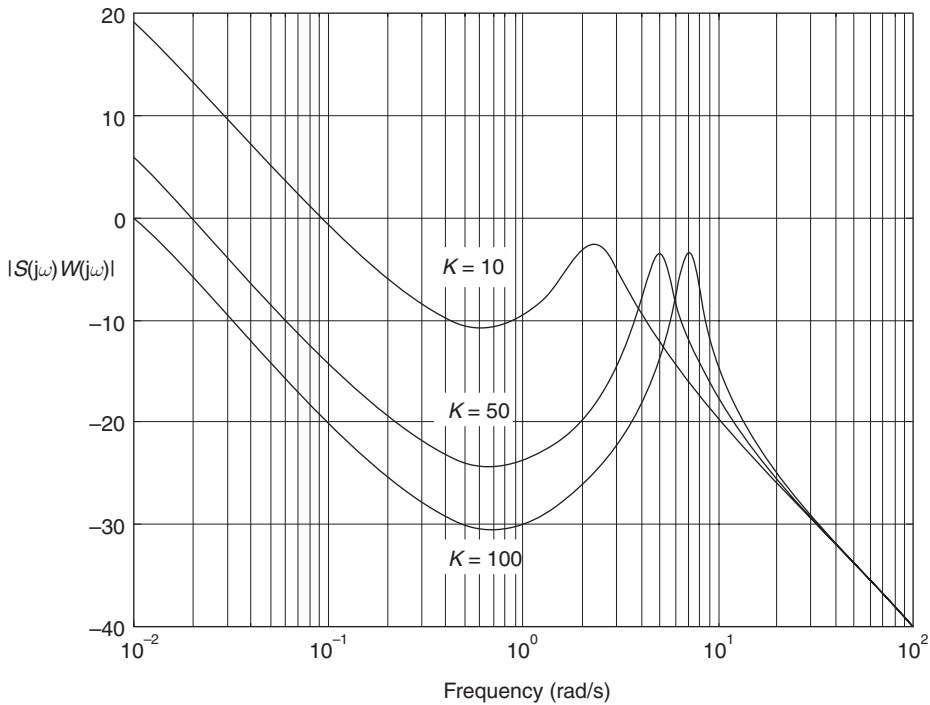


Fig. 9.26 Bode magnitude plot of weighted sensitivity function for Example 9.4.

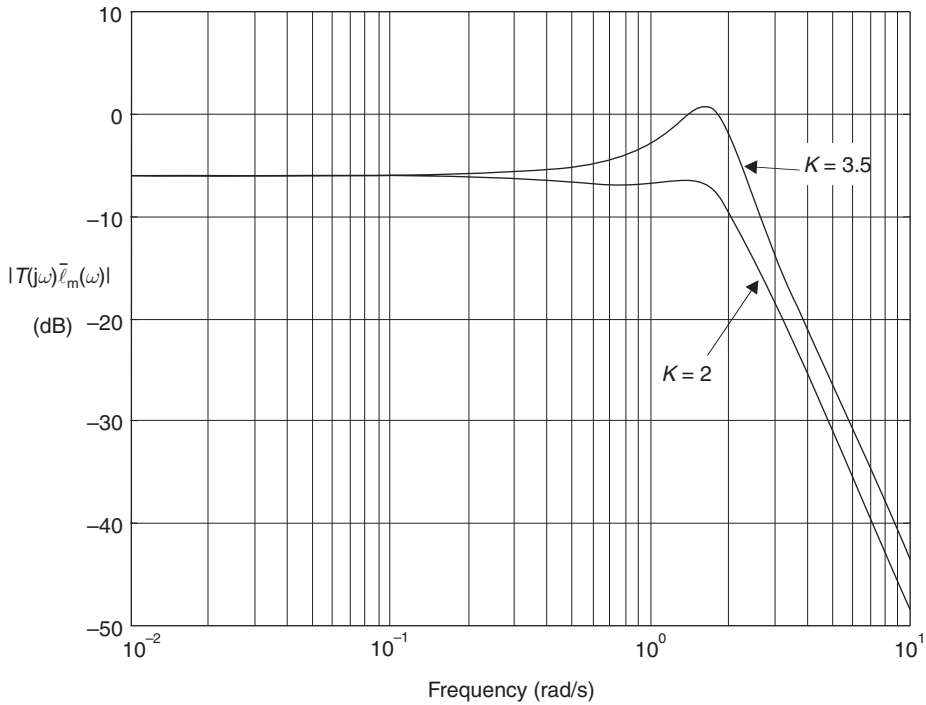


Fig. 9.27 Bode plot of $|T(j\omega)\bar{l}_m(\omega)|$ for Example 9.5.

thus

$$T(s)\bar{l}_m(s) = \frac{0.5K(1+s)}{(1+0.25s)(s^3+2s^2+4s+K)} \quad (9.155)$$

The Bode magnitude plot for equation (9.155) is shown in Figure 9.27 when $K = 2$ and 3.5.

In Example 6.4, when there was no model uncertainty, K for marginal stability was 8, and for a gain margin of 6 dB, K was 4. In this example with model uncertainty, from equation (9.154) marginal stability occurs with $K = 3.5$, so this is the maximum value for robust stability. For robust performance, equation (9.150) applies. For a specific step input let $W(s) = 1/s$ now

$$S_m(s) = \frac{s^3 + 2s^2 + 4s}{s^3 + 2s^2 + 4s + K} \quad (9.156)$$

and

$$S_m(s)W(s) = \frac{s(s^2 + 2s + 4)}{s(s^3 + 2s^2 + 4s + K)}$$

hence

$$S_m(s)W(s) = \frac{s^2 + 2s + 4}{s^3 + 2s^2 + 4s + K} \quad (9.157)$$

The Bode magnitude plot of the weighted sensitivity function is shown in Figure 9.28 for $K = 2, 2.5$ and 3.5.

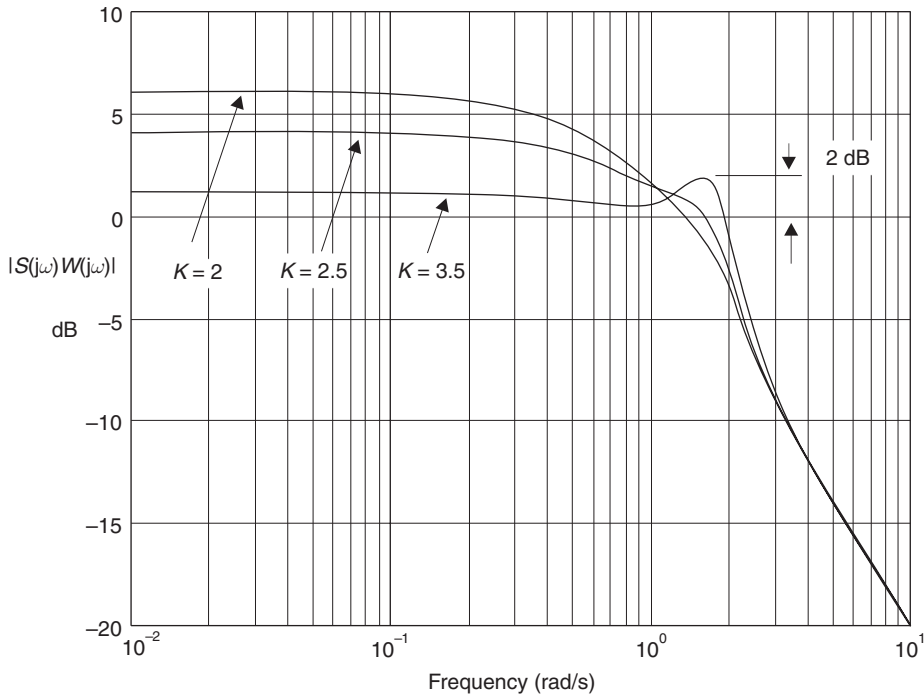


Fig. 9.28 Bode magnitude plot of weighted sensitivity function for Example 9.5.

For robust performance

$$|T_m(j\omega) + \bar{\ell}_m(\omega)| + |S_m(j\omega)W(j\omega)| < 1 \quad \text{for all } \omega \quad (9.158)$$

From Table 9.2 it can be concluded that:

- (a) The control system has robust stability up to $K = 3.5$.
- (b) The H_∞ -norm is >1 for all values of K considered. Therefore equations (9.145) and (9.150) are not met and the system cannot be considered to have robust performance.

From (b) above, it must be concluded that the controller $C(s)$ must be something more sophisticated than a simple gain constant K .

Table 9.2 Robust performance for Example 9.5

ω (rad/s)	0.01			1.5		
K	2	2.5	3.5	2	2.5	3.5
$ T_m(j\omega)\bar{\ell}_m(\omega) $	0.5 (−6 dB)	0.5 (not shown in Figure 9.27)	0.5	0.5	0.63 (not shown in Figure 9.27)	1.0
$ S_m(j\omega)W(j\omega) $	2.0 (6 dB)	1.58	1.12	0.96	1.05	1.26
Sum	2.5	2.08	1.62	1.46	1.68	2.26

9.9 Multivariable robust control

9.9.1 Plant equations

The canonical robust control problem is shown in Figure 9.29.

In Figure 9.29, \mathbf{u}_2 are the inputs to the plant \mathbf{P}_m from the controller and \mathbf{u}_1 are the disturbance and noise inputs. Also, \mathbf{y}_1 are the outputs to be controlled and \mathbf{y}_2 are the outputs that are fed back to the controller.

If $\mathbf{P}_m(s)$ and the plant uncertainty $\Delta(s)$ are combined to give $\mathbf{P}(s)$, then Figure 9.29 can be simplified as shown in Figure 9.30, also referred to as the two-port state-space representation.

The state and output equations are

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}_1\mathbf{u}_1 + \mathbf{B}_2\mathbf{u}_2 \\ \mathbf{y}_1 &= \mathbf{C}_1\mathbf{x} + \mathbf{D}_{11}\mathbf{u}_1 + \mathbf{D}_{12}\mathbf{u}_2 \\ \mathbf{y}_2 &= \mathbf{C}_2\mathbf{x} + \mathbf{D}_{21}\mathbf{u}_1 + \mathbf{D}_{22}\mathbf{u}_2\end{aligned}\tag{9.159}$$

Equation (9.159) can be combined

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B}_1 & \mathbf{B}_2 \\ \mathbf{C}_1 & \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{C}_2 & \mathbf{D}_{21} & \mathbf{D}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}\tag{9.160}$$

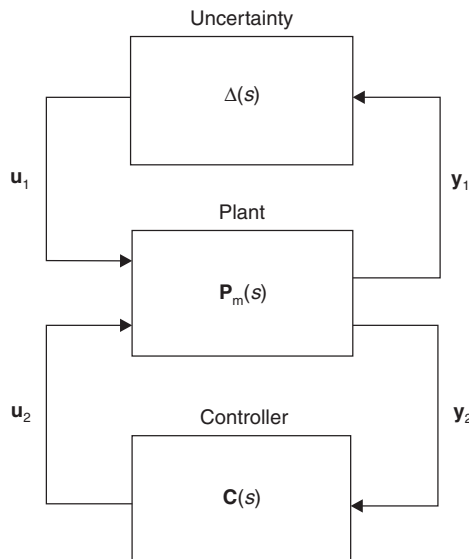


Fig. 9.29 The canonical robust control problem.

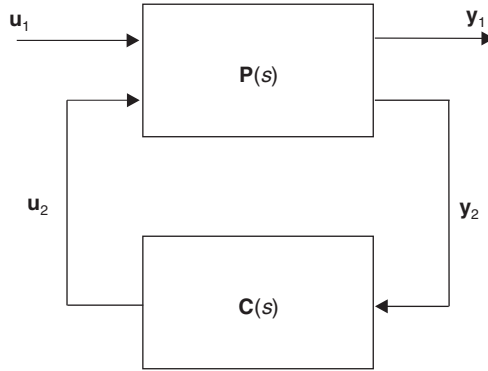


Fig. 9.30 Two-port state-space augmented plant and controller.

Hence the augmented plant matrix $\mathbf{P}(s)$ in Figure 9.30 is

$$\mathbf{P}(s) = \begin{bmatrix} \mathbf{A} & \vdots & \mathbf{B}_1 & \mathbf{B}_2 \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{C}_1 & \vdots & \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{C}_2 & \vdots & \mathbf{D}_{21} & \mathbf{D}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11} & \vdots & \mathbf{P}_{12} \\ \cdots & \cdots & \cdots \\ \mathbf{P}_{21} & \vdots & \mathbf{P}_{22} \end{bmatrix} \quad (9.161)$$

From the partitioned matrix in equation (9.161), the closed-loop transfer function matrix relating \mathbf{y}_1 and \mathbf{u}_1 is

$$\mathbf{T}\mathbf{y}_1\mathbf{u}_1 = \mathbf{P}_{11}(s) + \mathbf{P}_{12}(s)(\mathbf{I} - \mathbf{C}(s)\mathbf{P}_{22}(s))^{-1}\mathbf{C}(s)\mathbf{P}_{21}(s), \quad (9.162)$$

where

$$\mathbf{u}_2(s) = \mathbf{C}(s)\mathbf{y}_2(s) \quad (9.163)$$

9.9.2 Singular value loop shaping

The singular values of a complex $n \times m$ matrix \mathbf{A} , denoted by $\sigma_i(\mathbf{A})$ are the non-negative square-roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ ordered such that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \quad p = \min\{n, m\} \quad (9.164)$$

The maximum singular value $\bar{\sigma}$ of \mathbf{A} and the minimum singular value $\underline{\sigma}$ of \mathbf{A} are defined by

$$\begin{aligned} \bar{\sigma}(\mathbf{A}) &= \|\mathbf{A}\|_2 \\ \underline{\sigma}(\mathbf{A}) &= \|\mathbf{A}^{-1}\|_2^{-1} \quad \text{if } \mathbf{A}^{-1} \text{ exists} \end{aligned} \quad (9.165)$$

As with a SISO system, a sensitivity function may be defined

$$\mathbf{S}(s) = (\mathbf{I} + \mathbf{G}(s)\mathbf{C}(s))^{-1} \quad (9.166)$$

where $\mathbf{G}(s)$ is the non-augmented plant matrix. For good performance $\mathbf{S}(s)$ should be as small as possible. The complementary sensitivity function is

$$\mathbf{T}(s) = \mathbf{G}(s) \mathbf{C}(s) (\mathbf{I} + \mathbf{G}(s) \mathbf{C}(s))^{-1} \quad (9.167)$$

where

$$\mathbf{S}(s) + \mathbf{T}(s) = \mathbf{I} \quad (9.168)$$

The singular value of the sensitivity function $\bar{\sigma}(\mathbf{S}(j\omega))$ and of the complementary sensitivity function $\bar{\sigma}(\mathbf{T}(j\omega))$ can be displayed as Bode plots and play an important role in robust multivariable control system design.

The singular values of \mathbf{S} determine the disturbance attenuation, and thus a performance specification may be written

$$\bar{\sigma}(\mathbf{S}(j\omega)) \leq |\mathbf{W}_s^{-1}(j\omega)| \quad (9.169)$$

where $|\mathbf{W}_s^{-1}(j\omega)|$ is a desired disturbance attenuation factor. If $\Delta_m(s)$ is a diagonal matrix of multiplicative plant uncertainty as illustrated in Figure 9.29, it can be shown that the size of the smallest stable $\Delta_m(s)$ for which the system becomes unstable is

$$\bar{\sigma}(\Delta_m(j\omega)) = 1/\bar{\sigma}(\mathbf{T}(j\omega)) \quad (9.170)$$

or alternatively

$$\bar{\sigma}(\mathbf{T}(j\omega)) \leq |\mathbf{W}_T^{-1}(j\omega)| \quad (9.171)$$

where $|\mathbf{W}_T(j\omega)|$ is the size of the largest anticipated multiplicative plant uncertainty.

9.9.3 Multivariable H_2 and H_∞ robust control

The H_2 -optimal control problem is to find a stabilizing controller $\mathbf{C}(s)$ in equation (9.163) for an augmented plant $\mathbf{P}(s)$ in equation (9.161), such that the closed-loop transfer function matrix $\mathbf{T}_{y_1 u_1}$ in equation (9.162) is minimized.

Thus

$$\min_{\mathbf{C}(s)} \|\mathbf{T}_{y_1 u_1}\|_2 = \min_{\mathbf{C}(s)} \left\{ \frac{1}{\pi} \int_0^\infty \text{trace}(\mathbf{T}_{y_1 u_1}(j\omega)^T \mathbf{T}_{y_1 u_1}(j\omega)) d\omega \right\}^{1/2} \quad (9.172)$$

where \mathbf{T} is the complex conjugate transpose, and trace is the sum of the diagonal elements. The H_∞ robust control problem is to find a stabilizing controller $\mathbf{C}(s)$ for an augmented plant $\mathbf{P}(s)$, such that the closed-loop transfer function matrix $\mathbf{T}_{y_1 u_1}$ satisfies the infinity-norm inequality

$$\|\mathbf{T}_{y_1 u_1}\|_\infty = \sup_\omega \sigma_{\max}(\mathbf{T}_{y_1 u_1}(j\omega)) < 1 \quad (9.173)$$

Equation (9.173) is also called the ‘small gain’ infinity-norm control problem.

9.9.4 The weighted mixed-sensitivity approach

Multivariable loop shaping in robust control system design may be achieved using a weighted mixed sensitivity approach. As with the SISO systems described in section 9.8.2, the sensitivity function $S(s)$ given in equation (9.166) and the complementary sensitivity function $T(s)$ given in equation (9.167) may be combined with weights $W_s(s)$ and $W_T(s)$ to give

$$\mathbf{T}y_1\mathbf{u}_1 = \begin{bmatrix} W_s(s) & S(s) \\ W_T(s) & T(s) \end{bmatrix} \quad (9.174)$$

where the infinity norm of $\mathbf{T}y_1\mathbf{u}_1$ is <1 as given in equation (9.173). Equation (9.174) defines a mixed-sensitivity cost function since both $S(s)$ and $T(s)$ are penalized. Note that if $W_s(s)$ weights the error and $W_T(s)$ the output, the two-port augmented plant given in Figure 9.30 may be represented by Figure 9.31.

Example 9.6 (See also Appendix 1, *examp96.m*)

A plant has a transfer function

$$G(s) = \frac{200}{s^3 + 3s^2 + 102s + 200} \quad (9.175)$$

given the sensitivity and complementary weighting functions

$$\begin{aligned} W_s(s) &= \gamma \left(\frac{100 + s}{1 + 100s} \right) \\ W_T(s) &= \left(\frac{1 + 100s}{100 + s} \right) \end{aligned} \quad (9.176)$$

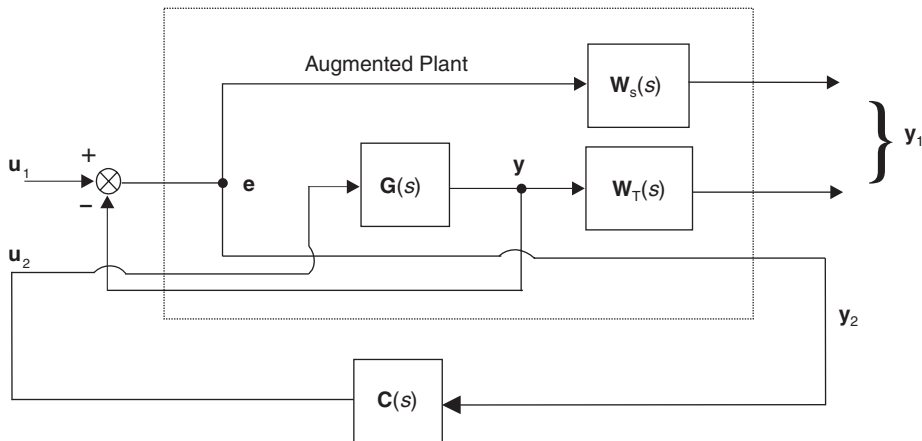


Fig. 9.31 Weighted mixed-sensitivity approach.

determine the singular value Bode magnitude plots for

- (a) the plant $G(j\omega)$
- (b) the weighting functions $W_s^{-1}(j\omega)$ and $W_T^{-1}(j\omega)$
- (c) the cost function $\mathbf{T}y_1\mathbf{u}_1(j\omega)$ at its optimal value of γ (given in $W_s(s)$)
- (d) the H_∞ -optimal controller $C(j\omega)$

Find also the state-space and transfer function expressions for the controller.

Solution

The state-space representation of the plant $G(s)$ is

$$\begin{bmatrix} \mathbf{A}_g & \mathbf{B}_g \\ \mathbf{C}_d & \mathbf{D}_g \end{bmatrix} = \begin{bmatrix} -3 & -102 & -200 & \vdots & 1 \\ 1 & 0 & 0 & \vdots & 0 \\ 0 & 1 & 0 & \vdots & 0 \\ \hdashline 0 & 0 & 200 & \vdots & 0 \end{bmatrix} \quad (9.177)$$

The singular value frequency response $G(j\omega)$ is shown in Figure 9.32.

The frequency response of the reciprocal of the weighting functions $W_s^{-1}(j\omega)$ ($\gamma = 1$) and $W_T^{-1}(j\omega)$ are given in Figure 9.33.

The optimal value of $\mathbf{T}y_1\mathbf{u}_1$ is achieved when ($\gamma = 0.13$) and its singular value frequency response is shown in Figure 9.34.

The controller single value frequency response $C(j\omega)$ is illustrated in Figure 9.35.

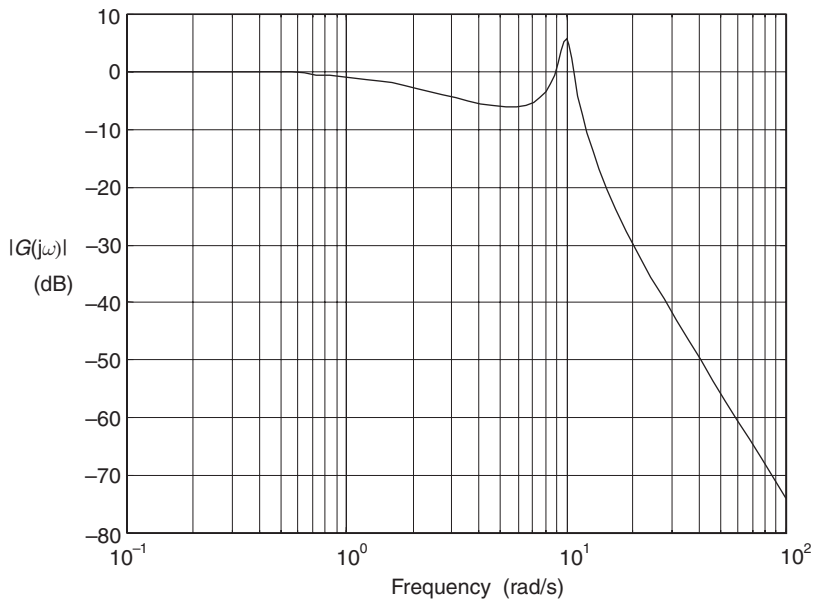


Fig. 9.32 Plant singular value Bode magnitude plot.

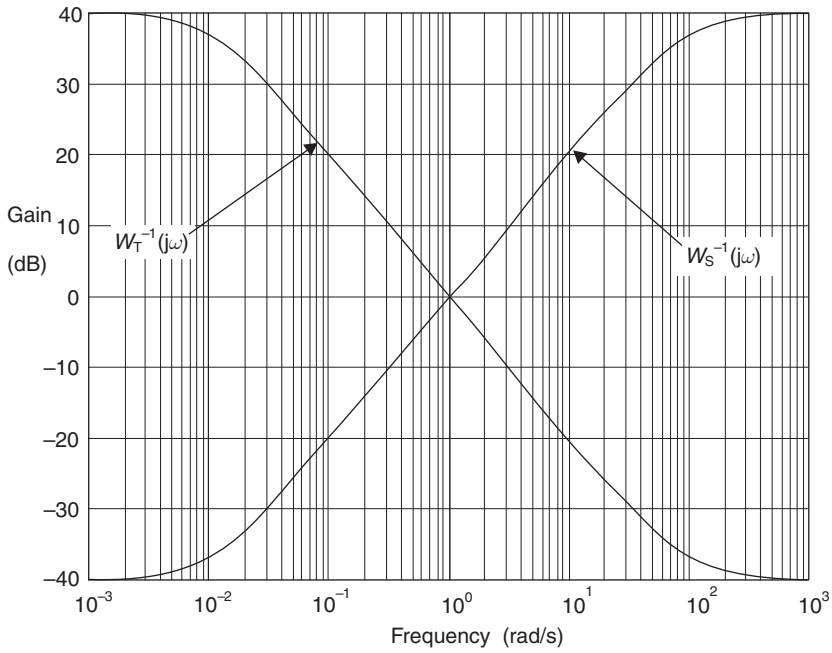


Fig. 9.33 Weighting functions Bode magnitude plots.

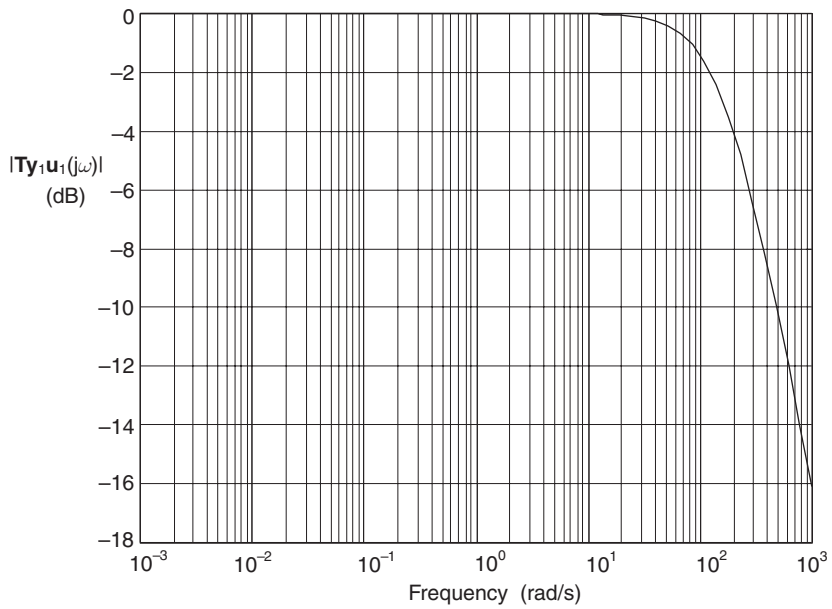


Fig. 9.34 Singular value Bode magnitude plot of $|Ty_1 u_1(j\omega)|$ when $\gamma = 0.13$.

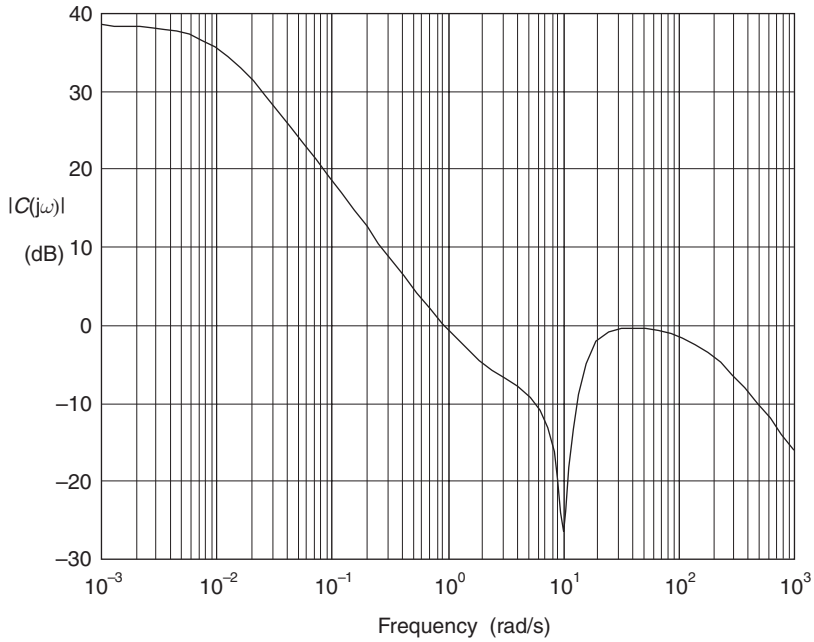


Fig. 9.35 Controller single value Bode magnitude plot $|C(j\omega)|$.

The state-space representation of the controller $C(s)$ is

$$\begin{bmatrix} \mathbf{A}_c & \mathbf{B}_c \\ \mathbf{C}_c & \mathbf{D}_c \end{bmatrix} = \begin{bmatrix} -0.01 & -0.002 & 0.004 & 0.015 & 0.137 & \vdots & -7.976 \\ -0.009 & -7.763 & 21.653 & 37.164 & 621.89 & \vdots & 0.091 \\ -0.046 & -2.032 & -3.148 & -2.234 & -81.88 & \vdots & 0.582 \\ 0.435 & 0.107 & 8.807 & -102.25 & -37.78 & \vdots & -8.8 \\ 0.676 & 0.197 & 13.558 & -3.728 & -162.51 & \vdots & -13.66 \\ -0.086 & 0.169 & 0.121 & -0.654 & -11.17 & \vdots & 0 \end{bmatrix} \quad (9.178)$$

and the controller transfer function is

$$C(s) = \frac{159s^4 + 16.4 \times 10^3 s^3 + 63.9 \times 10^3 s^2 + 1.6 \times 10^6 s + 3.18 \times 10^6}{s^5 + 275s^4 + 20.4 \times 10^3 s^3 + 324.8 \times 10^3 s^2 + 3.78 \times 10^6 s + 37.8 \times 10^3} \quad (9.179)$$

The results in this example were obtained using the MATLAB Robust Control Toolbox.

9.10 Further problems

Example 9.7

In a multivariable optimal regulator system, the plant state equations are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 4 \end{bmatrix} u$$

$$y = [1 \quad 0]\mathbf{x}$$
(9.180)

If the performance index to be minimized is

$$J = \int_0^\infty (\mathbf{x}^T \mathbf{Q} \mathbf{x} + u^2) dt$$
(9.181)

- (a) Determine, by hand, the elements of the Riccati matrix \mathbf{P} in the reduced Riccati equation

$$\mathbf{P}\mathbf{A} + \mathbf{A}^T \mathbf{P} + \mathbf{Q} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1} \mathbf{B}^T \mathbf{P} = \mathbf{0}$$
(9.182)

given that

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$
(9.183)

- (b) Find the optimal feedback matrix \mathbf{K} so that

$$\mathbf{K} = \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P}$$
(9.184)

and hence calculate the closed-loop eigenvalues.

Solutions

(a) $\mathbf{P} = \begin{bmatrix} 1.703 & 0.183 \\ 0.183 & 0.193 \end{bmatrix}$

(b) $\mathbf{K} = [0.732 \quad 0.772]$

$$s = -2.544 \pm j0.675$$

Example 9.8

A plant and measurement system are described by

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{C}_d \mathbf{w} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \\ \mathbf{z} &= \mathbf{y} + \mathbf{v} \end{aligned}$$
(9.185)

where $\mathbf{w}(t)$ is a Gaussian sequence of disturbances and $\mathbf{v}(t)$ is a Gaussian sequence of measurement noise. $\mathbf{z}(t)$ is the measured value of $\mathbf{y}(t)$ that is contaminated with measurement noise $\mathbf{v}(t)$. The plant parameters are

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -50 & -102 & -4.5 \end{bmatrix} & \mathbf{B} &= \begin{bmatrix} 0 \\ 0 \\ 100 \end{bmatrix} \\ \mathbf{C}_d &= \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 10 \end{bmatrix} & \mathbf{C} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (9.186)$$

The measurement noise and disturbance covariance matrices are

$$\mathbf{R} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 6 \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad (9.187)$$

- For a sampling time of 0.1 seconds, using equations (8.78) and (8.80) calculate the discrete-time state transition, control and disturbance matrices $\mathbf{A}(T)$, $\mathbf{B}(T)$ and $\mathbf{C}_d(T)$.
- Starting with an initial covariance matrix $\mathbf{P}(k/k)$ equal to the identity matrix, perform 20 recursions of equations (9.74), (9.75) and (9.76) to compute the Kalman gain matrix $\mathbf{K}(k+1)$ and covariance matrix $\mathbf{P}(k+1/k+1)$.

Solutions

$$\begin{aligned} \text{(a)} \quad \mathbf{A}(T) &= \begin{bmatrix} 0.993 & 0.085 & 0.004 \\ -0.199 & 0.587 & 0.067 \\ -3.370 & -7.074 & 0.284 \end{bmatrix} & \mathbf{B}(T) &= \begin{bmatrix} 0.014 \\ 0.398 \\ 6.740 \end{bmatrix} \\ \mathbf{C}_d(T) &= \begin{bmatrix} 0.050 & 0.002 & 0.001 \\ -0.004 & 0.043 & 0.040 \\ -0.100 & -0.207 & 0.674 \end{bmatrix} \\ \text{(b)} \quad \mathbf{K}(k+1) &= \begin{bmatrix} 0.040 & -0.006 & -0.002 \\ -0.006 & 0.137 & 0.003 \\ -0.133 & 0.155 & 0.218 \end{bmatrix} \\ \mathbf{P}(k+1/k+1) &= \begin{bmatrix} 0.004 & -0.001 & -0.013 \\ -0.001 & 0.014 & 0.016 \\ -0.013 & 0.016 & 1.310 \end{bmatrix} \end{aligned}$$

Example 9.9

The plant described in Example 9.8 by equations (9.185) and (9.186) is to be controlled by a Linear Quadratic Gaussian (LQG) control scheme that consists of a LQ Regulator combined with the Kalman filter designed in Example 9.8. The

quadratic performance index to be minimized for the LQ regulator is of the form given in equation (9.181) where

$$\mathbf{Q} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\mathbf{R} = 1) \quad (9.188)$$

Using the recursive equations (9.29) and (9.30), solve, in reverse time, the Riccati equation commencing with $\mathbf{P}(N) = \mathbf{0}$.

If the sampling time is 0.1 seconds, the values of the discrete-time state transition and control matrices $\mathbf{A}(T)$ and $\mathbf{B}(T)$ calculated in Example 9.8 may be used in the recursive solution.

Continue the recursive steps until the solution settles down (when $k = 50$, or $kT = 5$ seconds) and hence determine the steady-state value of the feedback matrix $\mathbf{K}(0)$ and Riccati matrix $\mathbf{P}(0)$. What are the closed-loop eigenvalues?

Solutions

$$\mathbf{K}(0) = [-0.106 \quad -0.581 \quad 0.064]$$

$$\mathbf{P}(0) = \begin{bmatrix} 11.474 & 3.406 & 0.153 \\ 3.406 & 3.952 & 0.163 \\ 0.153 & 0.163 & 0.1086 \end{bmatrix}$$

closed-loop eigenvalues = -1.230

$$-4.816 \pm j2.974$$

Example 9.10

A unity-feedback control system has a nominal plant transfer function

$$G_m(s) = \frac{1}{(s+2)(s+5)} \quad (9.189)$$

and an integral controller in the forward path

$$C(s) = \frac{K}{s} \quad (9.190)$$

If the bound of the multiplicative model uncertainty is

$$\bar{\ell}_m(s) = \frac{0.25(1+4s)}{(1+0.25s)} \quad (9.191)$$

determine:

- Expressions for the sensitivity and complementary sensitivity function $S(s)$ and $T(s)$ for the nominal plant.
- The maximum value that K can have for robust stability.

Solutions

$$(a) \quad S(s) = \frac{s^3 + 7s^2 + 10s}{s^3 + 7s^2 + 10s + K}$$

$$T(s) = \frac{K}{s^3 + 7s^2 + 10s + K}$$

$$(b) \quad K_{\max} = 4.5$$

Example 9.11

A plant has a transfer function

$$G(s) = \frac{100}{s^2 + 2s + 100}$$

and sensitivity and complementary weighting functions

$$W_s(s) = \gamma \left(\frac{s + 100}{s + 1} \right)$$

$$W_T(s) = \left(\frac{s + 1}{s + 100} \right)$$

Find the optimal value for γ and hence the state-space and transfer functions for the H_∞ -optimal controller $C(s)$.

Solutions

$$\gamma_{\text{opt}} = 0.0576$$

$$\begin{bmatrix} \mathbf{A}_c & \mathbf{B}_c \\ \mathbf{C}_c & \mathbf{D}_c \end{bmatrix} = \begin{bmatrix} -2.8 & 3.0 & -1.4 & 150.9 & \vdots & -4.55 \\ -3.0 & -22.2 & 50.1 & 258.0 & \vdots & 3.10 \\ -2.7 & 39.6 & -74.5 & 196.1 & \vdots & 2.99 \\ -3.2 & -36.3 & -23.2 & -1871.1 & \vdots & 7.44 \\ \hdashline -0.33 & 1.68 & 1.02 & -49.42 & \vdots & 0 \end{bmatrix}$$

$$C(s) = \frac{1.86 \times 10^3 s^3 + 0.1898 \times 10^6 s^2 + 0.5581 \times 10^6 s + 18.6031 \times 10^6}{s^4 + 1.97 \times 10^3 s^3 + 0.2005 \times 10^6 s^2 + 1.3531 \times 10^6 s + 1.1546 \times 10^6}$$

Intelligent control system design

10.1 Intelligent control systems

10.1.1 Intelligence in machines

According to the Oxford dictionary, the word intelligence is derived from intellect, which is the faculty of knowing, reasoning and understanding. Intelligent behaviour is therefore the ability to reason, plan and learn, which in turn requires access to knowledge.

Artificial Intelligence (AI) is a by-product of the Information Technology (IT) revolution, and is an attempt to replace human intelligence with machine intelligence. An intelligent control system combines the techniques from the fields of AI with those of control engineering to design autonomous systems that can sense, reason, plan, learn and act in an intelligent manner. Such a system should be able to achieve sustained desired behaviour under conditions of uncertainty, which include:

- (a) uncertainty in plant models
- (b) unpredictable environmental changes
- (c) incomplete, inconsistent or unreliable sensor information
- (d) actuator malfunction.

10.1.2 Control system structure

An intelligent control system, as considered by Johnson and Picton (1995), comprises of a number of subsystems as shown in Figure 10.1.

The perception subsystem

This collects information from the plant and the environment, and processes it into a form suitable for the cognition subsystem. The essential elements are:

- (a) *Sensor array* which provides raw data about the plant and the environment
- (b) *Signal processing* which transforms information into a suitable form
- (c) *Data fusion* which uses multidimensional data spaces to build representations of the plant and its environment. A key technology here is pattern recognition.

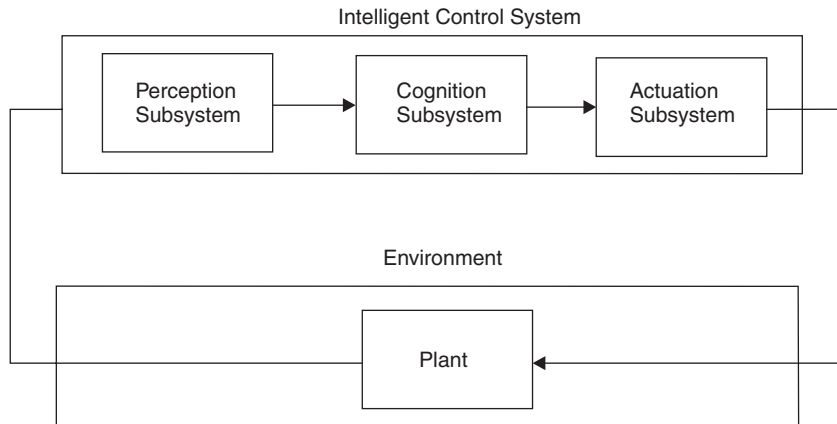


Fig. 10.1 Intelligent control system structure (adapted from Johnson and Picton).

The cognition subsystem

Cognition in an intelligent control system is concerned with the decision making process under conditions of uncertainty. Key activities include:

- (a) Reasoning, using
 - (i) knowledge-based systems
 - (ii) fuzzy logic
- (b) Strategic planning, using
 - (i) optimum policy evaluation
 - (ii) adaptive search and genetic algorithms
 - (iii) path planning
- (c) Learning, using
 - (i) supervised learning in neural networks
 - (ii) unsupervised learning in neural networks
 - (iii) adaptive learning

The actuation subsystem

The actuators operate using signals from the cognition subsystem in order to drive the plant to some desired states. In the event of actuator (or sensor) failure, an intelligent control system should be capable of being able to re-configure its control strategy.

This chapter is mainly concerned with some of the processes that are contained within the cognition subsystem.

10.2 Fuzzy logic control systems

10.2.1 Fuzzy set theory

Fuzzy logic was first proposed by Zadeh (1965) and is based on the concept of fuzzy sets. Fuzzy set theory provides a means for representing uncertainty. In general, probability theory is the primary tool for analysing uncertainty, and assumes that the

uncertainty is a random process. However, not all uncertainty is random, and fuzzy set theory is used to model the kind of uncertainty associated with imprecision, vagueness and lack of information.

Conventional set theory distinguishes between those elements that are members of a set and those that are not, there being very clear, or crisp boundaries. Figure 10.2 shows the crisp set 'medium temperature'. Temperatures between 20 and 30 °C lie within the crisp set, and have a membership value of one.

The central concept of fuzzy set theory is that the membership function μ , like probability theory, can have a value of between 0 and 1. In Figure 10.3, the membership function μ has a linear relationship with the x -axis, called the universe of discourse U . This produces a triangular shaped fuzzy set.

Fuzzy sets represented by symmetrical triangles are commonly used because they give good results and computation is simple. Other arrangements include non-symmetrical triangles, trapezoids, Gaussian and bell shaped curves.

Let the fuzzy set 'medium temperature' be called fuzzy set M . If an element u of the universe of discourse U lies within fuzzy set M , it will have a value of between 0 and 1. This is expressed mathematically as

$$\mu_M(u) \in [0, 1] \quad (10.1)$$

When the universe of discourse is discrete and finite, fuzzy set M may be expressed as

$$M = \sum_{i=1}^n \mu_M(u_i)/u_i \quad (10.2)$$

In equation (10.2) '/' is a delimiter. Hence the numerator of each term is the membership value in fuzzy set M associated with the element of the universe indicated in the denominator. When $n = 11$, equation (10.2) can be written as

$$M = 0/0 + 0/5 + 0/10 + 0.33/15 + 0.67/20 + 1/25 + 0.67/30 + 0.33/35 + 0/40 + 0/45 + 0/50 \quad (10.3)$$

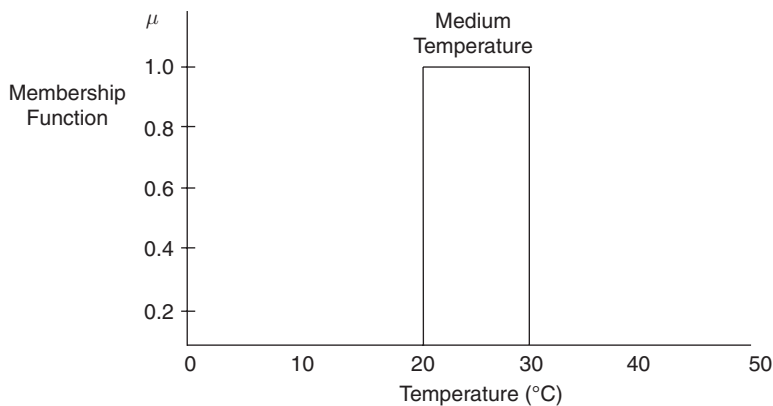


Fig. 10.2 Crisp set 'medium temperature'.

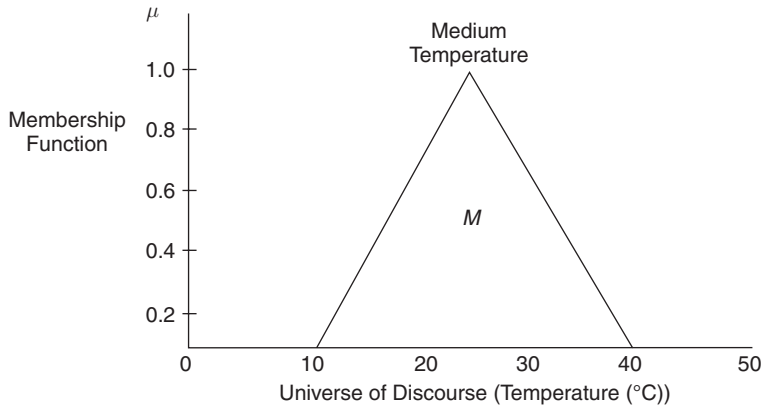


Fig. 10.3 Fuzzy set 'medium temperature'.

Note the symbol '+' is not an addition in the normal algebraic sense, but in fuzzy arithmetic denotes a union operation.

10.2.2 Basic fuzzy set operations

Let A and B be two fuzzy sets within a universe of discourse U with membership functions μ_A and μ_B respectively. The following fuzzy set operations can be defined as

Equality: Two fuzzy sets A and B are equal if they have the same membership function within a universe of discourse U.

$$\mu_A(u) = \mu_B(u) \quad \text{for all } u \in U \quad (10.4)$$

Union: The union of two fuzzy sets A and B corresponds to the Boolean OR function and is given by

$$\mu_{A \cup B}(u) = \mu_{A+B}(u) = \max\{\mu_A(u), \mu_B(u)\} \quad \text{for all } u \in U \quad (10.5)$$

Intersection: The intersection of two fuzzy sets A and B corresponds to the Boolean AND function and is given by

$$\mu_{A \cap B}(u) = \min\{\mu_A(u), \mu_B(u)\} \quad \text{for all } u \in U \quad (10.6)$$

Complement: The complement of fuzzy set A corresponds to the Boolean NOT function and is given by

$$\mu_{\neg A}(u) = 1 - \mu_A(u) \quad \text{for all } u \in U \quad (10.7)$$

Example 10.1

Find the union and intersection of fuzzy set low temperature L and medium temperature M shown in Figure 10.4. Find also the complement of fuzzy set M. Using equation (10.2) the fuzzy sets for $n = 11$ are

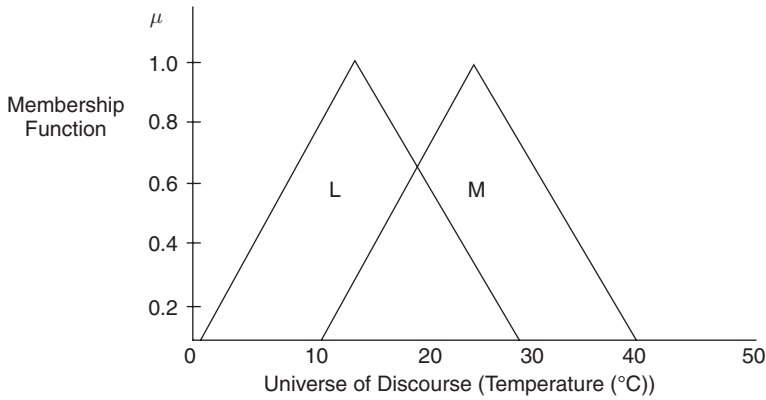


Fig. 10.4 Overlapping sets 'low' and 'medium temperature'.

$$\begin{aligned}
 L &= 0/0 + 0.33/5 + 0.67/10 + 1/15 + 0.67/20 + 0.33/25 \\
 &\quad + 0/30 + 0/35 + \dots + 0/50 \\
 M &= 0/0 + 0/5 + 0/10 + 0.33/15 + 0.67/20 + 1/25 + 0.67/30 \\
 &\quad + 0.33/35 + 0/40 + \dots + 0/50
 \end{aligned} \tag{10.8}$$

(a) *Union*: Using equation (10.5)

$$\begin{aligned}
 \mu_{L+M}(u) &= \max(0, 0)/0 + \max(0.33, 0)/5 + \max(0.67, 0)/10 \\
 &\quad + \max(1, 0.33)/15 + \max(0.67, 0.67)/20 + \max(0.33, 1)/25 \\
 &\quad + \max(0, 0.67)/30 + \max(0, 0.33)/35 + \max(0, 0)/40 + \dots \\
 &\quad + \max(0, 0)/50
 \end{aligned} \tag{10.9}$$

$$\begin{aligned}
 \mu_{L+M}(u) &= 0/0 + 0.33/5 + 0.67/10 + 1/15 + 0.67/20 + 1/25 + 0.67/30 \\
 &\quad + 0.33/35 + 0/40 + \dots + 0/50
 \end{aligned} \tag{10.10}$$

(b) *Intersection*: Using equation (10.6) and replacing 'max' by 'min' in equation (10.9) gives

$$\begin{aligned}
 \mu_{L \cap M}(u) &= 0/0 + 0/5 + 0/10 + 0.33/15 + 0.67/20 + 0.33/25 \\
 &\quad + 0/30 + \dots + 0/50
 \end{aligned} \tag{10.11}$$

Equations (10.10) and (10.11) are shown in Figure 10.5.

(c) *Complement*: Using equation (10.7)

$$\begin{aligned}
 \mu_{\neg M}(u) &= (1 - 0)/0 + (1 - 0)/5 + (1 - 0)/10 + (1 - 0.33)/15 \\
 &\quad + (1 - 0.67)/20 + (1 - 1)/25 + (1 - 0.67)/30 + (1 - 0.33)/35 \\
 &\quad + (1 - 0)/40 + \dots + (1 - 0)/50
 \end{aligned} \tag{10.12}$$

Equation (10.12) is illustrated in Figure 10.6.

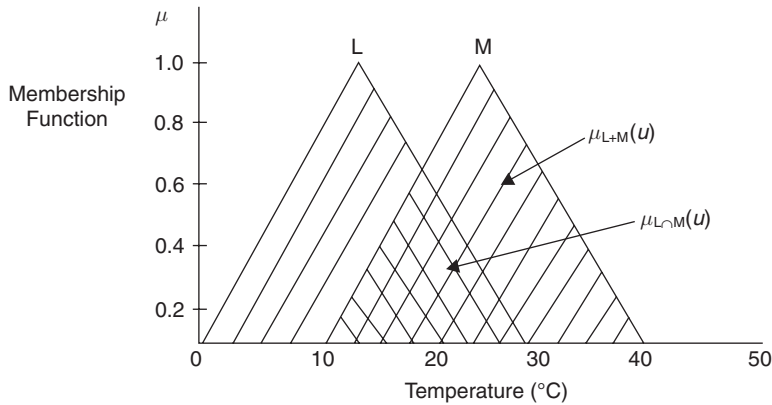


Fig. 10.5 'Union' and 'intersection' functions.

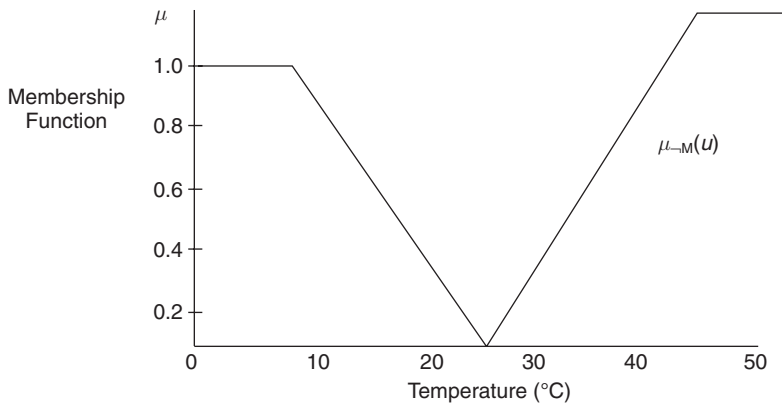


Fig. 10.6 The complement of fuzzy set M.

10.2.3 Fuzzy relations

An important aspect of fuzzy logic is the ability to relate sets with different universes of discourse. Consider the relationship

$$\text{IF } L \text{ THEN } M \quad (10.13)$$

In equation (10.13) L is known as the *antecedent* and M as the *consequent*. The relationship is denoted by

$$A = L \times M \quad (10.14)$$

or

$$L \times M = \begin{bmatrix} \min\{\mu_L(u_1), \mu_M(v_1)\} \dots \min\{\mu_L(u_1), \mu_M(v_k)\} \\ \min\{\mu_L(u_j), \mu_M(v_1)\} \dots \min\{\mu_L(u_j), \mu_M(v_k)\} \end{bmatrix} \quad (10.15)$$

where $u_1 \rightarrow u_j$ and $v_1 \rightarrow v_k$ are the discretized universe of discourse. Consider the statement

$$\text{IF } L \text{ is low THEN } M \text{ is medium} \quad (10.16)$$

Then for the fuzzy sets L and M defined by equation (10.8), for U from 5 to 35 in steps of 5

$$L \times M = \begin{bmatrix} \min(0.33, 0) & \dots & \min(0.33, 1) & \dots & \min(0.33, 0.33) \\ \min(0.67, 0) & \dots & \min(0.67, 1) & \dots & \min(0.67, 0.33) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \min(0, 0) & \dots & \min(0, 1) & \dots & \min(0, 0.33) \end{bmatrix} \quad (10.17)$$

which gives

$$L \times M = \begin{bmatrix} 0 & 0 & 0.33 & 0.33 & 0.33 & 0.33 & 0.33 \\ 0 & 0 & 0.33 & 0.67 & 0.67 & 0.67 & 0.33 \\ 0 & 0 & 0.33 & 0.67 & 1 & 0.67 & 0.33 \\ 0 & 0 & 0.33 & 0.67 & 0.67 & 0.67 & 0.33 \\ 0 & 0 & 0.33 & 0.33 & 0.33 & 0.33 & 0.33 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (10.18)$$

Several such statements would form a control strategy and would be linked by their union

$$A = A_1 + A_2 + A_3 + \dots + A_n \quad (10.19)$$

10.2.4 Fuzzy logic control

The basic structure of a Fuzzy Logic Control (FLC) system is shown in Figure 10.7.

The fuzzification process

Fuzzification is the process of mapping inputs to the FLC into fuzzy set membership values in the various input universes of discourse. Decisions need to be made regarding

- (a) number of inputs
- (b) size of universes of discourse
- (c) number and shape of fuzzy sets.

A FLC that emulates a PD controller will be required to minimize the error $e(t)$ and the rate of change of error de/dt , or ce .

The size of the universes of discourse will depend upon the expected range (usually up to the saturation level) of the input variables. Assume for the system about to be considered that e has a range of ± 6 and ce a range of ± 1 .

The number and shape of fuzzy sets in a particular universe of discourse is a trade-off between precision of control action and real-time computational complexity. In this example, seven triangular sets will be used.

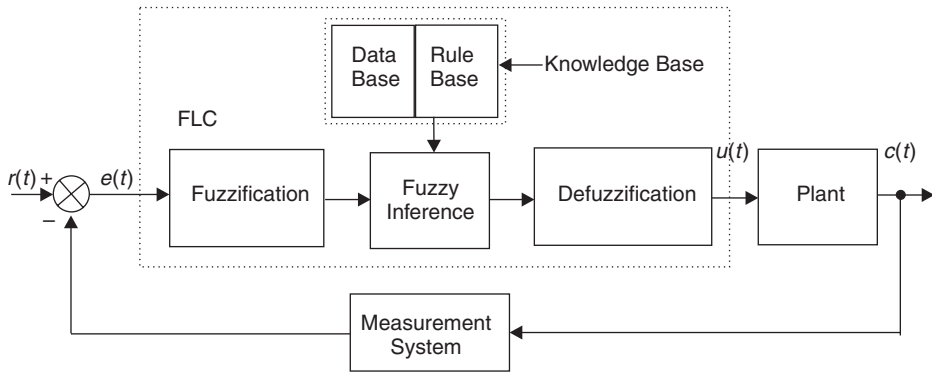


Fig. 10.7 Fuzzy Logic Control System.

Each set is given a linguistic label to identify it, such as Positive Big (PB), Positive Medium (PM), Positive Small (PS), About Zero (Z), Negative Small (NS), Negative Medium (NM) and Negative Big (NB). The seven set fuzzy input windows for e and ce are shown in Figure 10.8. If at a particular instant, $e(t) = 2.5$ and $de/dt = -0.2$, then, from Figure 10.8, the input fuzzy set membership values are

$$\begin{aligned} \mu_{PS}(e) &= 0.7 & \mu_{PM}(e) &= 0.4 \\ \mu_{NS}(ce) &= 0.6 & \mu_Z(ce) &= 0.3 \end{aligned} \quad (10.20)$$

The fuzzy rulebase

The fuzzy rulebase consists of a set of antecedent–consequent linguistic rules of the form

$$\text{IF } e \text{ is PS AND } ce \text{ is NS THEN } u \text{ is PS} \quad (10.21)$$

This style of fuzzy conditional statement is often called a ‘Mamdani’-type rule, after Mamdani (1976) who first used it in a fuzzy rulebase to control steam plant.

The rulebase is constructed using *a priori* knowledge from either one or all of the following sources:

- Physical laws that govern the plant dynamics
- Data from existing controllers
- Imprecise heuristic knowledge obtained from experienced experts.

If (c) above is used, then knowledge of the plant mathematical model is not required.

The two seven set fuzzy input windows shown in Figure 10.8 gives a possible 7×7 set of control rules of the form given in equation (10.21). It is convenient to tabulate the two-dimensional rulebase as shown in Figure 10.9.

Fuzzy inference

Figure 10.9 assumes that the output window contains seven fuzzy sets with the same linguistic labels as the input fuzzy sets. If the universe of discourse for the control signal $u(t)$ is ± 9 , then the output window is as shown in Figure 10.10.

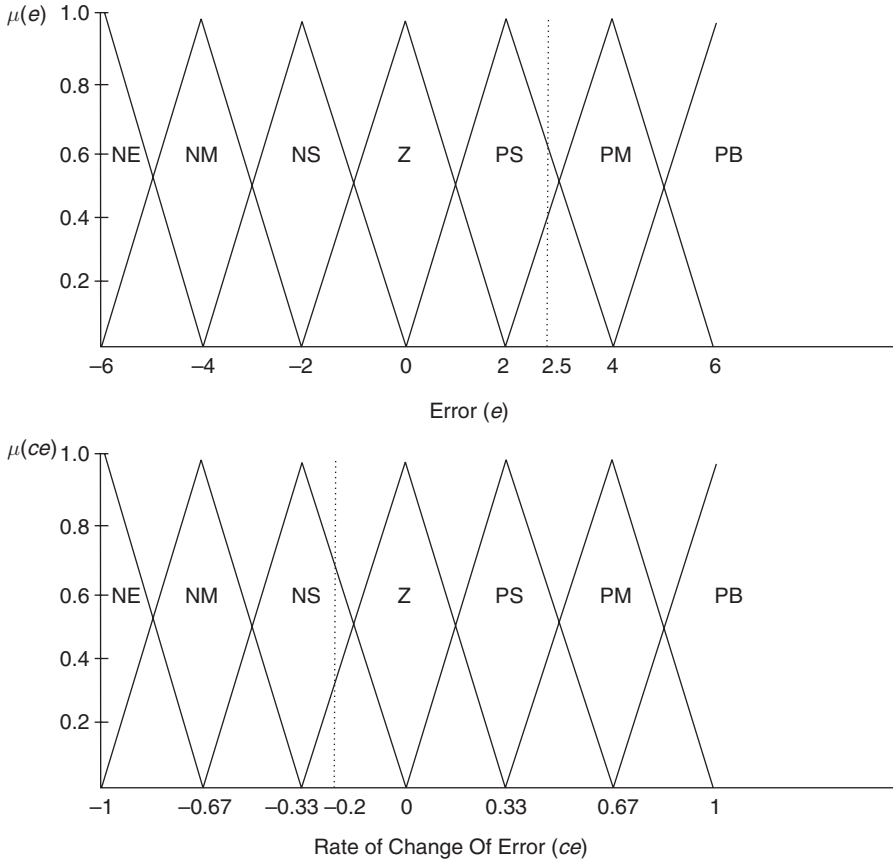


Fig. 10.8 Seven set fuzzy input windows for error (e) and rate of change of error (ce).

Assume that a certain rule in the rulebase is given by equation (10.22)

$$\text{OR IF } e \text{ is A AND } ce \text{ is B THEN } u = C \quad (10.22)$$

From equation (10.5) the Boolean OR function becomes the fuzzy max operation, and from equation (10.6) the Boolean AND function becomes the fuzzy min operation. Hence equation (10.22) can be written as

$$\mu_C(u) = \max[\min(\mu_A(e), \mu_B(ce))] \quad (10.23)$$

Equation (10.23) is referred to as the max–min inference process or max–min fuzzy reasoning.

In Figure 10.8 and equation (10.20) the fuzzy sets that were ‘hit’ in the error input window when $e(t) = 2.5$ were PS and PM. In the rate of change input window when $ce = -0.2$, the fuzzy sets to be ‘hit’ were NS and Z. From Figure 10.9, the relevant rules that correspond to these ‘hits’ are

$\begin{matrix} e \\ \backslash \\ ce \end{matrix}$		e						
		NB	NM	NS	Z	PS	PM	PB
ce	NB	NB	NB	NB	NM	Z	PM	PB
	NM	NB	NB	NB	NM	PS	PM	PB
	NS	NB	NB	NM	NS	PS	PM	PB
	Z	NB	NM	NS	Z	PS	PM	PB
	PS	NB	NM	NS	PS	PM	PB	PB
	PM	NB	NM	NS	PM	PB	PB	PB
	PB	NB	NM	Z	PM	PB	PB	PB

Fig. 10.9 Tabular structure of a linguistic fuzzy rulebase.

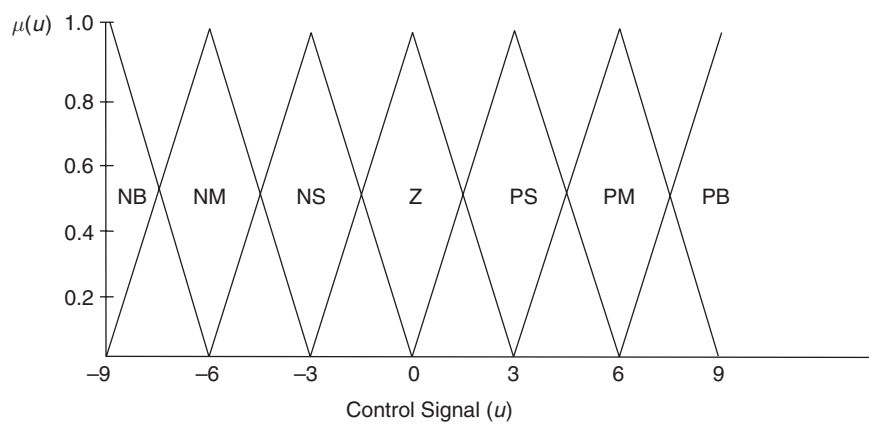


Fig. 10.10 Seven set fuzzy output window for control signal (u).

... OR IF e is PS AND ce is NS
OR IF e is PS AND ce is Z
THEN $u = \text{PS}$

(10.24)

... OR IF e is PM AND ce is NS
OR IF e is PM and ce is Z
THEN $u = \text{PM}$

(10.25)

Applying the max–min inference process to equation (10.24)

$$\mu_{PS}(u) = \max[\min(\mu_{PS}(e), \mu_{NS}(ce)), \min(\mu_{PS}(e), \mu_Z(ce))] \quad (10.26)$$

inserting values from equation (10.20)

$$\begin{aligned} \mu_{PS}(u) &= \max[\min(0.7, 0.6), \min(0.7, 0.3)] \\ &= \max[0.6, 0.3] = 0.6 \end{aligned} \quad (10.27)$$

Applying the max–min inference process to equation (10.25)

$$\mu_{PM}(u) = \max[\min(\mu_{PM}(e), \mu_{NS}(ce)), \min(\mu_{PM}(e), \mu_Z(ce))] \quad (10.28)$$

inserting values from equation (10.20)

$$\begin{aligned} \mu_{PM}(u) &= \max[\min(0.4, 0.6), \min(0.4, 0.3)] \\ &= \max[0.4, 0.3] = 0.4 \end{aligned} \quad (10.29)$$

Fuzzy inference is therefore the process of mapping membership values from the input windows, through the rulebase, to the output window(s).

The defuzzification process

Defuzzification is the procedure for mapping from a set of inferred fuzzy control signals contained within a fuzzy output window to a non-fuzzy (crisp) control signal. The *centre of area* method is the most well known defuzzification technique, which in linguistic terms can be expressed as

$$\text{Crisp control signal} = \frac{\text{Sum of first moments of area}}{\text{Sum of areas}} \quad (10.30)$$

For a continuous system, equation (10.30) becomes

$$u(t) = \frac{\int u\mu(u)du}{\int \mu(u)du} \quad (10.31)$$

or alternatively, for a discrete system, equation (10.30) can be expressed as

$$u(kT) = \frac{\sum_{i=1}^n u_i\mu(u_i)}{\sum_{i=1}^n \mu(u_i)} \quad (10.32)$$

For the case when $e(t) = 2.5$ and $ce = -0.2$, as a result of the max–min inference process (equations (10.27) and (10.29)), the fuzzy output window in Figure 10.10 is ‘clipped’, and takes the form shown in Figure 10.11.

From Figure 10.11, using the equation for the area of a trapezoid

$$\begin{aligned} \text{Area}_{PS} &= \frac{0.6(6 + 2.4)}{2} = 2.52 \\ \text{Area}_{PM} &= \frac{0.2(6 + 3.6)}{2} = 0.96 \end{aligned} \quad (10.33)$$

From equation (10.30)

$$u(t) = \frac{(2.52 \times 3) + (0.96 \times 6)}{2.52 + 0.96} = 3.83 \quad (10.34)$$

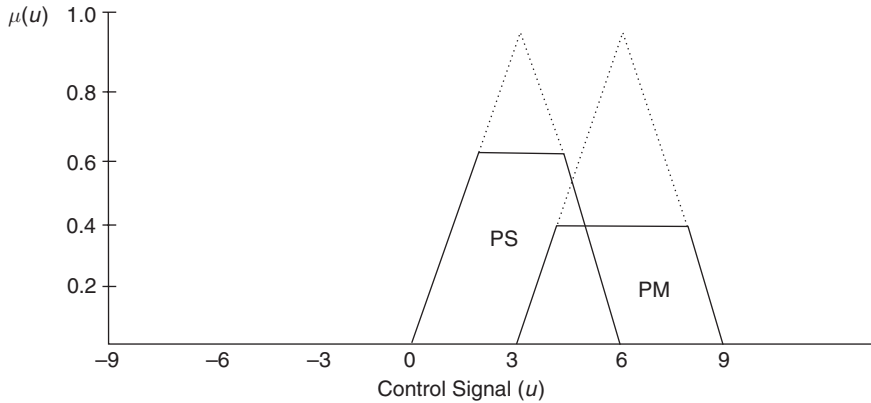


Fig. 10.11 Clipped fuzzy output window due to fuzzy inference.

Hence, for given error of 2.5, and a rate of change of error of -0.2 , the control signal from the fuzzy controller is 3.83.

Example 10.2

For the input and output fuzzy windows given in Figure 10.8 and 10.10, together with the fuzzy rulebase shown in Figure 10.9, determine

- the membership values of the input windows e and ce .
- the max-min fuzzy inference equations
- the crisp control signal $u(t)$

when $e = -3$ and $ce = 0.3$

Solution

- When $e = -3$ and $ce = 0.3$ are mapped onto the input fuzzy windows, they are referred to as fuzzy singletons. From Figure 10.8

$$e = -3 \quad \mu_{NS}(e) = 0.5 \quad \mu_{NM}(e) = 0.5 \quad (10.35)$$

$ce = 0.3$, using similar triangles

$$\begin{aligned} \frac{1}{0.33} &= \frac{\mu_Z(ce)}{(0.33 - 0.3)} \\ \mu_Z(ce) &= 0.09 \end{aligned} \quad (10.36)$$

and

$$\begin{aligned} \frac{1}{0.33} &= \frac{\mu_{PS}(ce)}{0.3} \\ \mu_{PS}(ce) &= 0.91 \end{aligned} \quad (10.37)$$

(b) The rules that are ‘hit’ in the rulebase in Figure 10.9 are

$$\begin{aligned} & \dots \text{ OR IF } e \text{ is NS and } ce \text{ is Z} \\ & \quad \text{OR IF } e \text{ is NS and } ce \text{ is PS} \\ & \quad \text{THEN } u = \text{NS} \end{aligned} \quad (10.38)$$

$$\begin{aligned} & \dots \text{ OR IF } e \text{ is NM and } ce \text{ is Z} \\ & \quad \text{OR IF } e \text{ is NM and } ce \text{ is PS} \\ & \quad \text{THEN } u = \text{NM} \end{aligned} \quad (10.39)$$

Applying max–min inference to equation (10.38)

$$\mu_{\text{NS}}(u) = \max[\min(\mu_{\text{NS}}(e), \mu_{\text{Z}}(ce)), \min(\mu_{\text{NS}}(e), \mu_{\text{PS}}(ce))] \quad (10.40)$$

Inserting values into (10.40)

$$\begin{aligned} \mu_{\text{NS}}(u) &= \max[\min(0.5, 0.09), \min(0.5, 0.91)] \\ &= \max[0.09, 0.5] = 0.5 \end{aligned} \quad (10.41)$$

and similarly with equation (10.39)

$$\begin{aligned} \mu_{\text{NM}}(u) &= \max[\min(\mu_{\text{NM}}(e), \mu_{\text{Z}}(ce)), \min(\mu_{\text{NM}}(e), \mu_{\text{PS}}(ce))] \\ &= \max[\min(0.5, 0.09), \min(0.5, 0.91)] \\ &= \max[0.09, 0.5] = 0.5 \end{aligned} \quad (10.42)$$

Using equations (10.41) and (10.42) to ‘clip’ the output window in Figure 10.10, the output window is now as illustrated in Figure 10.12.

(c) Due to the symmetry of the output window in Figure 10.12, from observation, the crisp control signal is

$$u(t) = -4.5$$

Example 10.3 (See also Appendix 1, *examp103.m*)

Design a fuzzy logic controller for the inverted pendulum system shown in Figure 10.13 so that the pendulum remains in the vertical position.

The inverted pendulum problem is a classic example of producing a stable closed-loop control system from an unstable plant.

Since the system can be modelled, it is possible to design a controller using the pole placement techniques discussed in Chapter 8. Neglecting friction at the pivot and the wheels, the equations of motion from Johnson and Picton (1995) are

$$\ddot{x} = \frac{F + m\ell(\dot{\theta}^2 \sin \theta - \ddot{\theta} \cos \theta)}{M + m} \quad (10.43)$$

$$\ddot{\theta} = \frac{g \sin \theta + \cos \theta \left(\frac{-F - m\ell \dot{\theta}^2 \sin \theta}{M + m} \right)}{\ell \left(\frac{4}{3} - \frac{m \cos^2 \theta}{M + m} \right)} \quad (10.44)$$

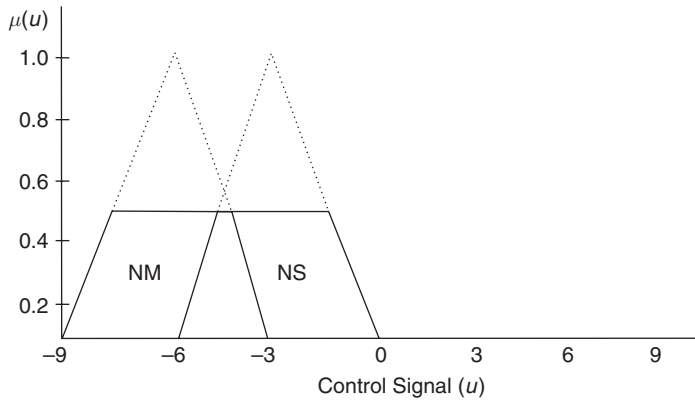


Fig. 10.12 Fuzzy output window for Example 10.2.

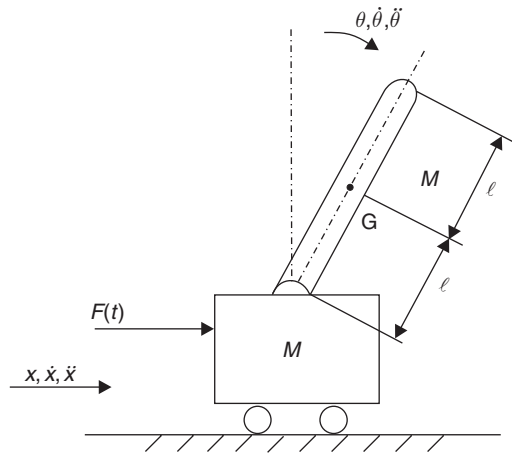


Fig. 10.13 An inverted pendulum.

In equations (10.43) and (10.44), m is the mass and ℓ is the half-length of the pendulum and M is the mass of the trolley. $F(t)$ is the applied force to the trolley in the x -direction. If it is assumed that θ is small and second-order terms ($\dot{\theta}^2$) can be neglected, then

$$\ddot{x} = \frac{F - m\ell\ddot{\theta}}{M + m} \quad (10.45)$$

$$\ddot{\theta} = \frac{g\theta + \left(\frac{-F}{M+m}\right)}{\ell\left(\frac{4}{3} - \frac{m}{M+m}\right)} \quad (10.46)$$

If the state variables are

$$x_1 = \theta, \quad x_2 = \dot{\theta}, \quad x_3 = x \quad \text{and} \quad x_4 = \dot{x}$$

and the control variable is

$$u = F(t)$$

then from equations (10.45) and (10.46), the state equations become

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ a_{41} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix} u \quad (10.47)$$

where

$$\begin{aligned} a_{21} &= \frac{3g(M+m)}{\ell\{4(M+m)-3m\}} \\ a_{41} &= \frac{-3gm}{4(M+m)-3m} \\ b_2 &= \frac{-3}{\ell\{4(M+m)-3m\}} \\ b_4 &= \left(\frac{1}{M+m}\right) \left\{1 + \frac{3m}{4(M+m)-3m}\right\} \end{aligned} \quad (10.48)$$

and the output equation is

$$\mathbf{y} = \mathbf{C}\mathbf{x} \quad (10.49)$$

where \mathbf{C} is the identity matrix. For a regulator, with a scalar control variable

$$u = -\mathbf{K}\mathbf{x}$$

The elements of \mathbf{K} can be obtained by selecting a set of desired closed-loop poles as described in section 8.4.2, and applying one of the three techniques discussed.

Data for simulation

$$\begin{aligned} \ell &= 1 \text{ m} \quad M = 1 \text{ kg} \quad m = 0.5 \text{ kg} \\ a_{21} &= \frac{3 \times 9.81(1.5)}{1\{(4 \times 1.5) - 1.5\}} = 9.81 \\ a_{41} &= \frac{-3 \times 9.81 \times 0.5}{(4 \times 1.5) - 1.5} = -3.27 \\ b_2 &= \frac{-3}{1\{(4 \times 1.5) - 1.5\}} = -0.667 \\ b_4 &= \left(\frac{1}{1.5}\right) \left\{1 + \frac{1.5}{(4 \times 1.5) - 1.5}\right\} = 0.889 \end{aligned}$$

If the required closed-loop poles are

$s = -2 \pm j2$ for the pendulum, and

$s = -4 \pm j4$ for the trolley, then the closed-loop characteristic equation is

$$s^4 + 12s^3 + 72s^2 + 192s + 256 = 0 \quad (10.50)$$

Using Ackermann's Formula in equations (8.103) and (8.104), the state feedback matrix becomes

$$\mathbf{K} = [-174.83 \quad -57.12 \quad -39.14 \quad -29.36] \quad (10.51)$$

Using the fuzzy logic approach suggested by Johnson and Picton (1995), four, three set input windows (one for each state variable) and one, three set output window has been selected as shown in Figure 10.14. Using heuristic knowledge from broom-balancing experiments, the following Mamdani-type rulebase was constructed:

1. IF θ is PB and $\dot{\theta}$ is PB then F is PB
2. IF θ is PB and $\dot{\theta}$ is Z then F is PB
3. IF θ is PB and $\dot{\theta}$ is NB then F is Z
4. IF θ is Z and $\dot{\theta}$ is PB then F is PB
5. IF θ is Z and $\dot{\theta}$ is Z then F is Z
6. IF θ is Z and $\dot{\theta}$ is NB then F is NB
7. IF θ is NB and $\dot{\theta}$ is PB then F is Z
8. IF θ is NB and $\dot{\theta}$ is Z then F is NB
9. IF θ is NB and $\dot{\theta}$ is NB then F is NB
10. IF $\dot{\theta}$ is PB then F is PB
11. IF $\dot{\theta}$ is NB then F is NB

The rulebase can be extended up to 22 rules by a further set of 11 rules replacing θ with x and $\dot{\theta}$ with \dot{x} .

For the rulebase given in equation (10.52), the fuzzy max-min inference process is

$$\begin{aligned} \mu_{PB}(u) &= \max[\mu_{PB}(\dot{\theta}), \min(\mu_{PB}(\theta), \mu_{PB}(\dot{\theta})), \min(\mu_{PB}(\theta), \mu_Z(\dot{\theta})), \min(\mu_Z(\theta), \mu_{PB}(\dot{\theta}))] \\ \mu_{NB}(u) &= \max[\mu_{NB}(\dot{\theta}), \min(\mu_Z(\theta), \mu_{NB}(\dot{\theta})), \min(\mu_{NB}(\theta), \mu_Z(\dot{\theta})), \min(\mu_{NB}(\theta), \mu_{NB}(\dot{\theta}))] \\ \mu_Z(u) &= \max[\min(\mu_{PB}(\theta), \mu_{NB}(\dot{\theta})), \min(\mu_Z(\theta), \mu_Z(\dot{\theta})), \min(\mu_{NB}(\theta), \mu_{PB}(\dot{\theta}))] \end{aligned}$$

Again, a similar inference process occurs with x and \dot{x} . Following defuzzification, a crisp control force $F(t)$ is obtained.

Figure 10.15 shows the time response of the inverted pendulum state variables from an initial condition of $\theta = 0.1$ radians. On each graph, three control strategies are shown, the 11 set rulebase of equation (10.52), the 22 set rulebase that includes x and \dot{x} , and the state feedback method given by equation (10.51).

For the pendulum angle, shown in Figure 10.15(a), the 11 set rulebase gives the best results, the state feedback being oscillatory and the 22 set rulebase diverging

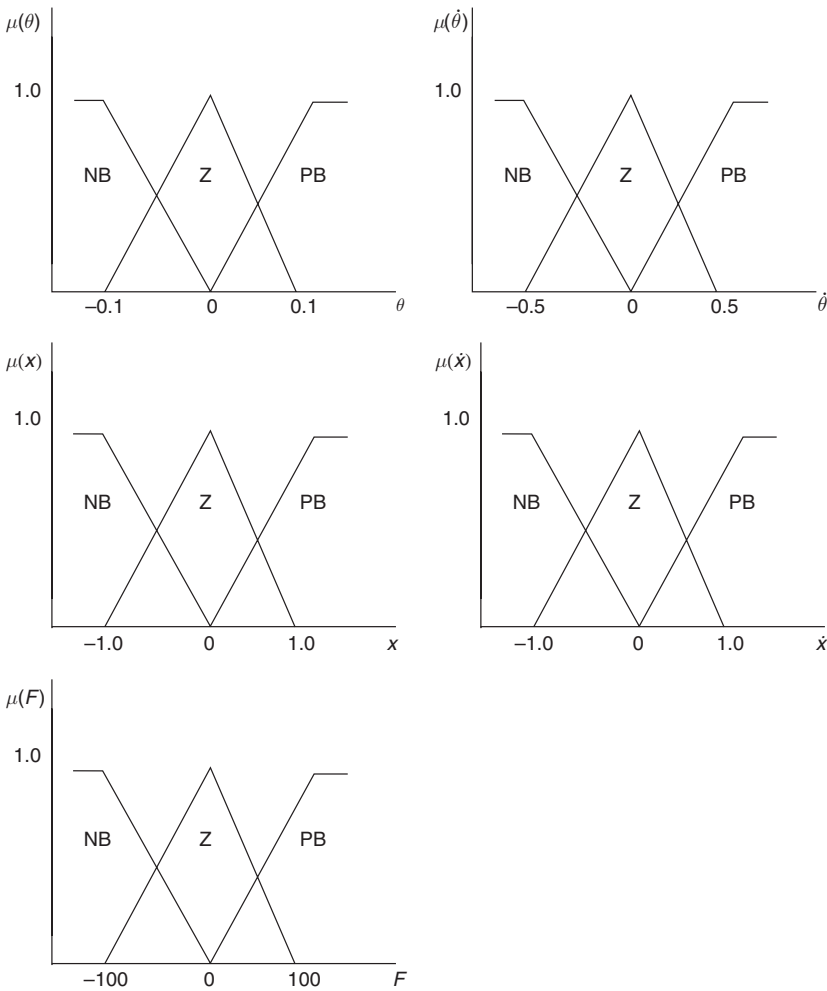
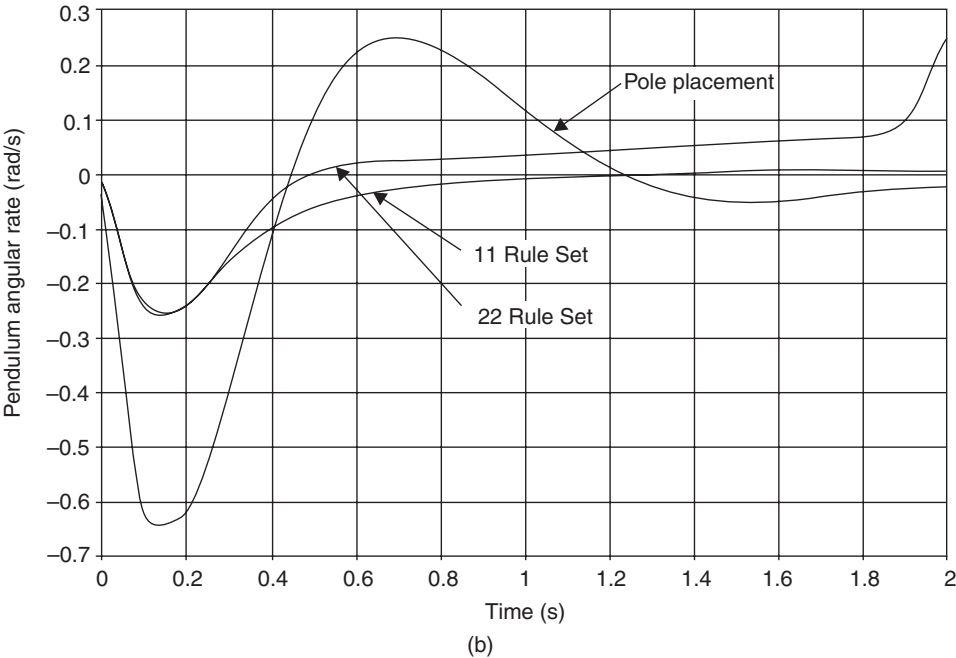
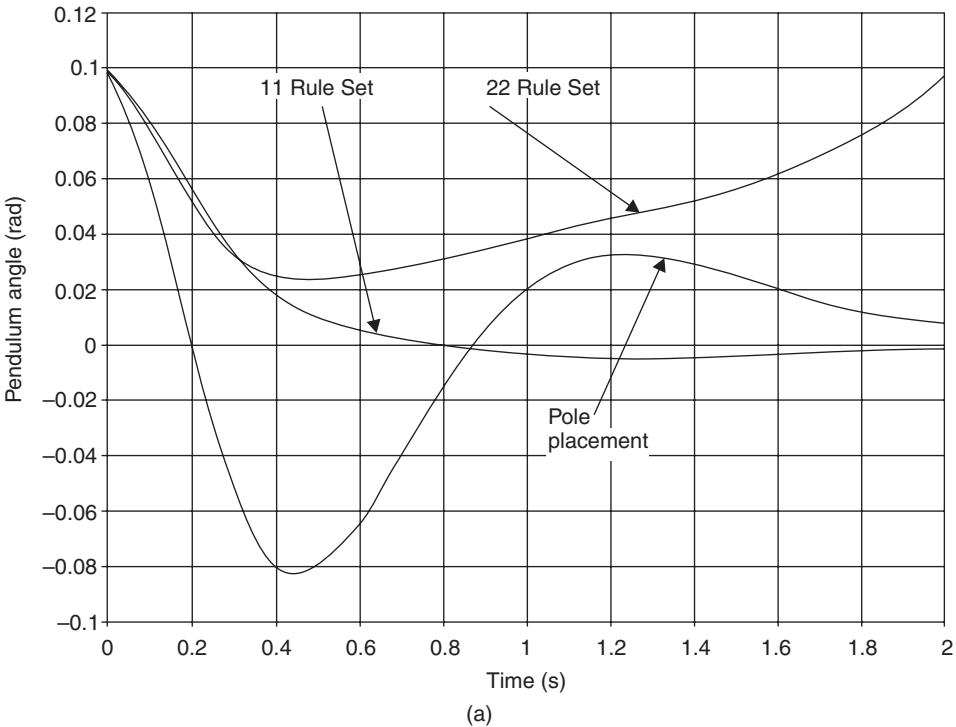


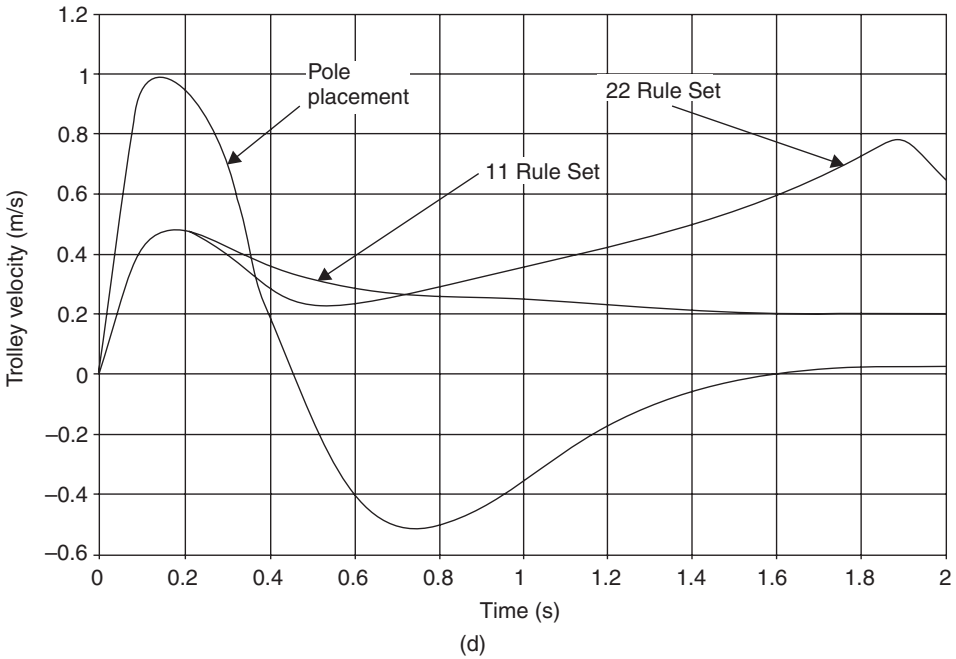
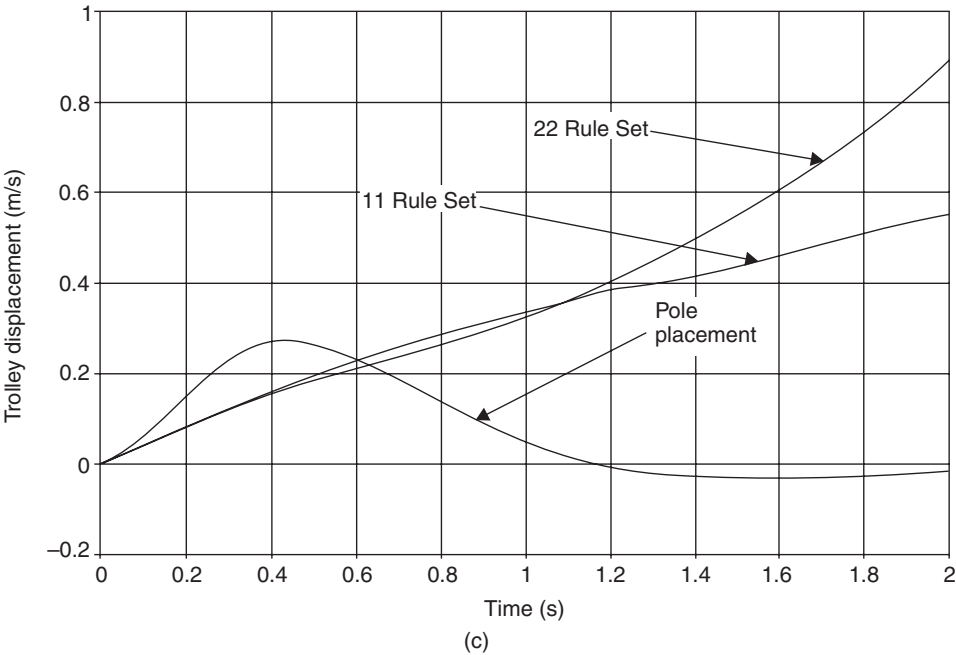
Fig. 10.14 Input and output fuzzy windows for the inverted pendulum problem.

after a while. The same comments apply to the pendulum angular rate, given in Figure 10.15(b).

With the trolley displacement and velocity shown in Figures 10.15(c) and (d), the state feedback, although oscillatory, give the best results since there is no steady-state error. The positional error for both rulebases increases with time, and there is a constant velocity steady-state error for the 11 set rulebase, and increasing error for the 22 set rulebase. Figure 10.15(e) shows the control force for each of the three strategies.

The 11 and 22 set rulebase simulations were undertaken using SIMULINK, together with the fuzzy logic toolbox for use with MATLAB. More details on the





(Fig. 10.15 continued)

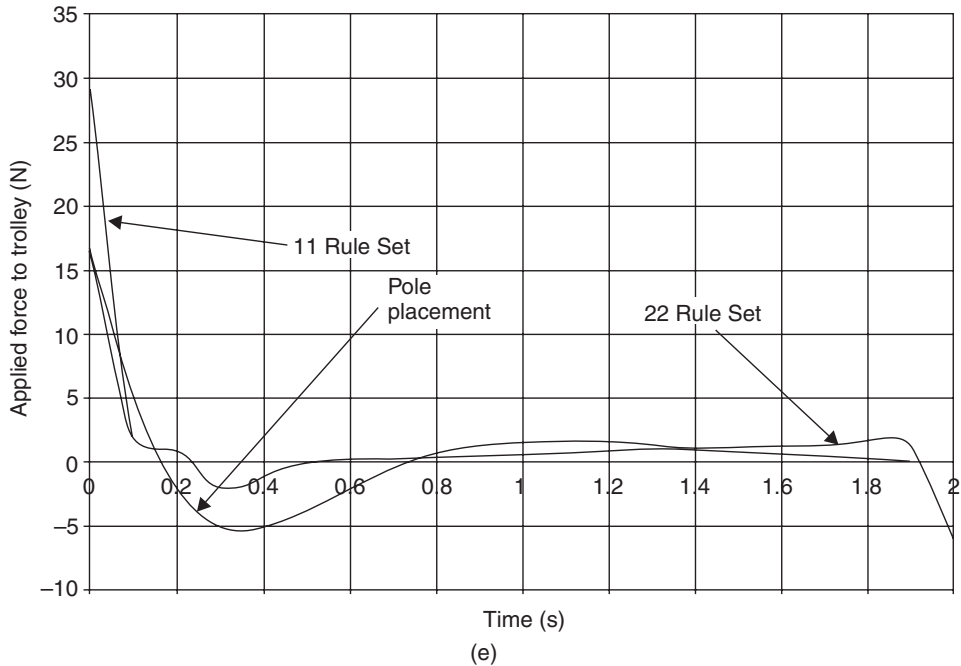


Fig. 10.15 Inverted pendulum state variable time response for three control strategies.

MATLAB Fuzzy Inference System (FIS) editor can be found in Appendix 1. Figure 10.16 shows the control surface for the 11 set rulebase fuzzy logic controller.

10.2.5 Self-organizing fuzzy logic control

Self-Organizing Fuzzy Logic Control (SOFLC) is an optimization strategy to create and modify the control rulebase for a FLC as a result of observed system performance. The SOFLC is particularly useful when the plant is subject to time-varying parameter changes and unknown disturbances.

Structure

A SOFLC is a two-level hierarchical control system that is comprised of:

- (a) a learning element at the top level
- (b) a FLC at the bottom level.

The learning element consists of a Performance Index (PI) table combined with a rule generation and modification algorithm, which creates new rules, or modifies existing ones. The structure of a SOFLC is shown in Figure 10.17. With SOFLC it is usual to express the PI table and rulebase in numerical, rather than linguistic format. So, for

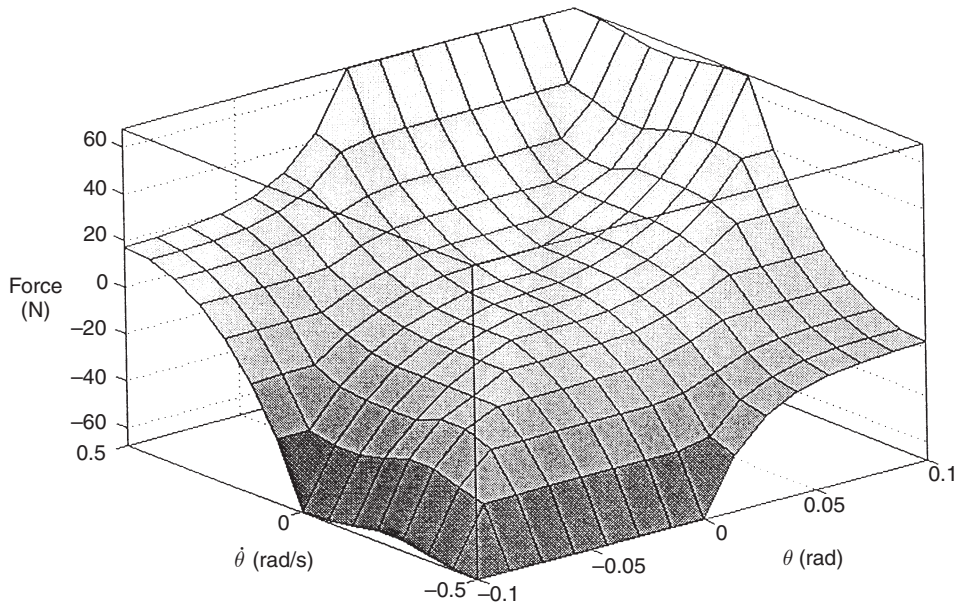


Fig. 10.16 Control surface for 11 set rulebase fuzzy logic controller.

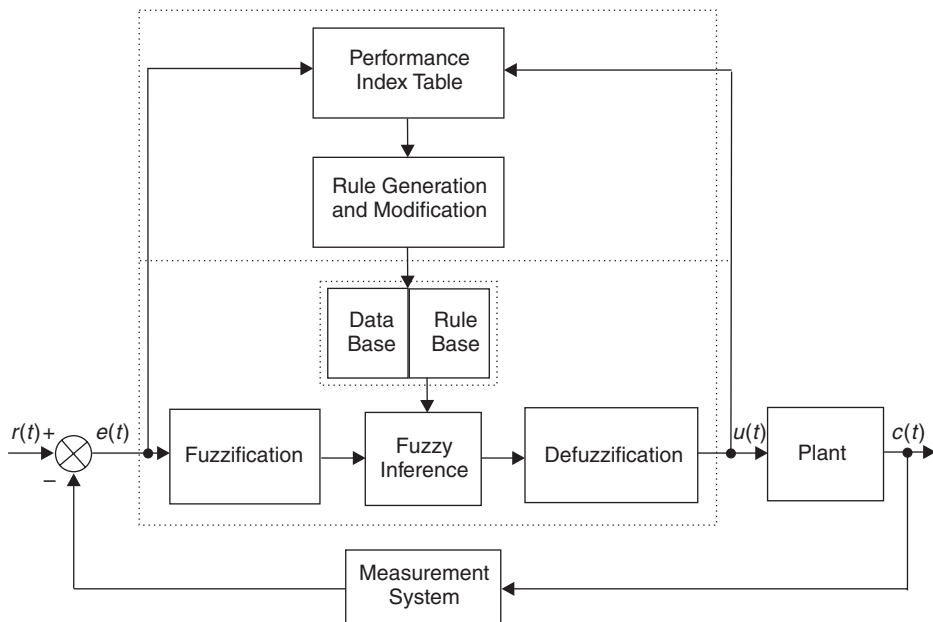


Fig. 10.17 Self-Organizing Fuzzy Logic Control system.

$\begin{matrix} e \\ \backslash \\ ce \end{matrix}$	NB	NM	NS	Z	PS	PM	PB
NB	-50	-40	-30	-20	-10	0	10
NM	-42	-32	-22	-12	-2	8	18
NS	-36	-26	-16	-6	4	14	24
Z	-30	-20	-10	0	10	20	30
PS	-24	-14	-4	6	16	26	36
PM	-18	-8	2	12	22	32	42
PB	-10	0	10	20	30	40	50

Fig. 10.18 Tabular structure of a numerical fuzzy rulebase.

example, the fuzzy rulebase in Figure 10.9, might take the form as shown in Figure 10.18.

Rulebase modification

If the numerical structure of the fuzzy rulebase does not give an acceptable response, then the values in certain cells will need to be adjusted.

Let the error, rate of change of error and control signal at time t be $e(t)$, $ce(t)$ and $u(t)$ respectively, and assume that a given PI is a function of $e(t)$ and $ce(t)$.

If there are unacceptable errors at time t , because of the dynamics of the plant, these will be as a result of control action taken d seconds previously, or at time $(t - d)$. The parameter d is a 'delay in reward' parameter and is related to the settling time of the plant, having a typical value of between $3T$ and $5T$, where T is the dominant time constant of the plant.

The value of the PI is therefore determined using $e(t - d)$ and $ce(t - d)$ and applied to $u(t)$ as a correction factor to the rulebase in the form

$$\text{IF } e(t) \text{ is } \dots \text{ and } ce(t) \text{ is } \dots \text{ THEN } u(t) = \dots + \text{PI} \quad (10.53)$$

where the PI is read from a Performance Index table of the form shown in Figure 10.19. When the values of $e(t - d)$ and $ce(t - d)$ are within an acceptable range, the PI tends to zero and the fuzzy rulebase settles down and convergence for the self-organizing process has been achieved. The PI table is usually designed heuristically, based upon an intuitive understanding of the learning process, and the trade-off between speed of learning and stability of the rulebase.

$\begin{matrix} e \\ ce \end{matrix}$	NB	NM	NS	Z	PS	PM	PB
NB	-5	-4	-3	-3	-2	-1	0
NM	-4	-3	-3	-2	-1	0	1
NS	-3	-3	-2	-1	0	1	2
Z	-3	-2	-1	0	1	2	3
PS	-2	-1	0	1	2	3	3
PM	-1	0	1	2	3	3	4
PB	0	1	2	3	3	4	5

Fig. 10.19 Performance Index table.

10.3 Neural network control systems

10.3.1 Artificial neural networks

The human brain is comprised of many millions of interconnected units, known individually as biological neurons. Each neuron consists of a cell to which is attached several dendrites (inputs) and a single axon (output). The axon connects to many other neurons via connection points called synapses. A synapse produces a chemical reaction in response to an input. The biological neuron ‘fires’ if the sum of the synaptic reactions is sufficiently large. The brain is a complex network of sensory and motor neurons that provide a human being with the capacity to remember, think, learn and reason.

Artificial Neural Networks (ANNs) attempt to emulate their biological counterparts. McCulloch and Pitts (1943) proposed a simple model of a neuron, and Hebb (1949) described a technique which became known as ‘Hebbian’ learning. Rosenblatt (1961), devised a single layer of neurons, called a Perceptron, that was used for optical pattern recognition.

One of the first applications of this technology for control purposes was by Widrow and Smith (1964). They developed an ADaptive LINear Element (ADLINE) that was taught to stabilize and control an inverted pendulum. Kohonen (1988) and Anderson (1972) investigated similar areas, looking into ‘associative’ and ‘interactive’ memory, and also ‘competitive learning’. The back propagation training algorithm was investigated by Werbos (1974) and further developed by Rumelhart (1986) and others, leading to the concept of the Multi-Layer Perceptron (MLP).

Artificial Neural Networks have the following potential advantages for intelligent control:

- They learn from experience rather than by programming.
- They have the ability to generalize from given training data to unseen data.
- They are fast, and can be implemented in real-time.
- They fail ‘gracefully’ rather than ‘catastrophically’.

10.3.2 Operation of a single artificial neuron

The basic model of a single artificial neuron consists of a weighted summer and an activation (or transfer) function as shown in Figure 10.20. Figure 10.20 shows a neuron in the j th layer, where

$x_1 \dots x_i$ are inputs

$w_{j1} \dots w_{ji}$ are weights

b_j is a bias

f_j is the activation function

y_j is the output

The weighted sum s_j is therefore

$$s_j(t) = \sum_{i=1}^N w_{ji}x_i(t) + b_j \quad (10.54)$$

Equation (10.54) can be written in matrix form

$$s_j(t) = \mathbf{W}_j \mathbf{x} + b_j \quad (10.55)$$

The activation function $f(s)$ (where s is the weighted sum) can take many forms, some of which are shown in Figure 10.21. From Figure 10.21 it can be seen that the bias b_j in equations (10.54) and (10.55) will move the curve along the s axis, i.e. effectively

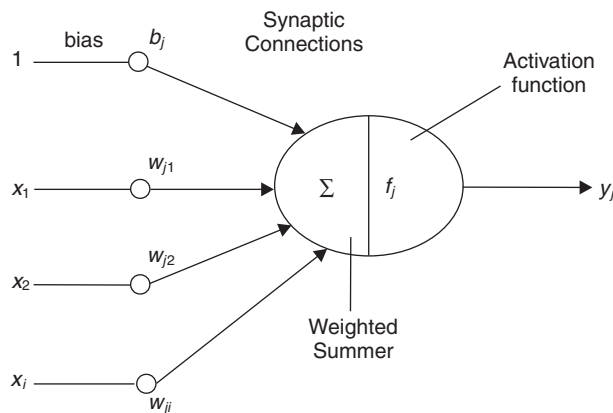


Fig. 10.20 Basic model of a single artificial neuron.

setting the threshold at which the neuron ‘fires’. So in the case of the hard-limiting function, if $b_j = 0$, the neuron will ‘fire’ when $s_j(t)$ changes from negative to positive.

The sigmoid activation function is popular for neural network applications since it is differentiable and monotonic, both of which are a requirement for the back-propagation algorithm. The equation for a sigmoid function is

$$f(s) = \frac{1}{1 + e^{-s_j}} \quad (10.56)$$

10.3.3 Network architecture

Feedforward networks

An ANN is a network of single neurons joined together by synaptic connections. Figure 10.22 shows a three-layer feedforward neural network.

The feedforward network shown in Figure 10.22 consists of a three neuron input layer, a two neuron output layer and a four neuron intermediate layer, called a hidden layer. Note that all neurons in a particular layer are fully connected to all neurons in the subsequent layer. This is generally called a fully connected multilayer network, and there is no restriction on the number of neurons in each layer, and no restriction on the number of hidden layers.

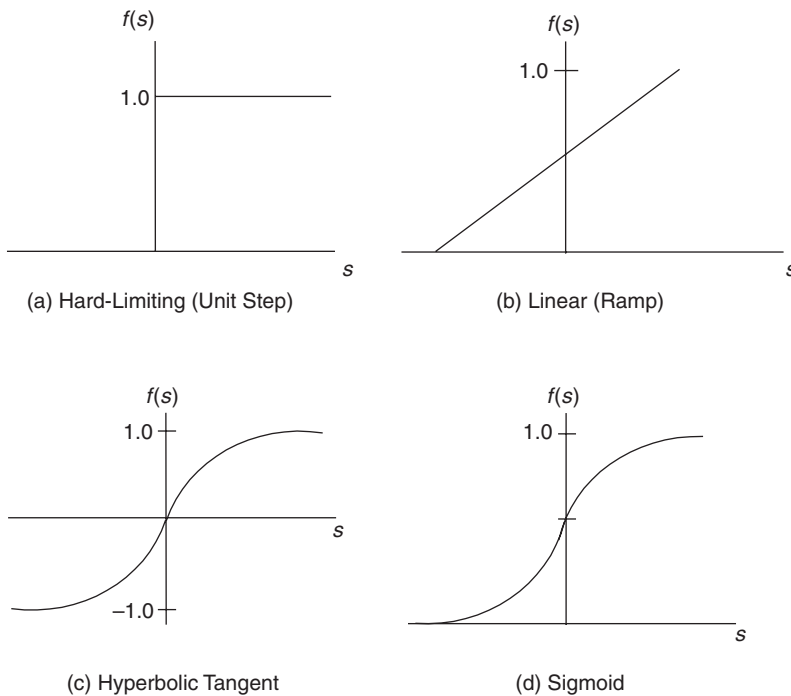


Fig. 10.21 Activation functions.

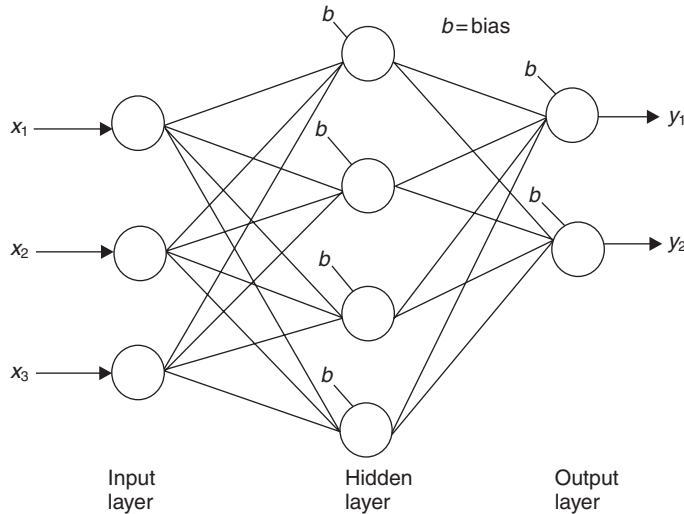


Fig. 10.22 Three-layer feedforward neural network.

Feedback (recurrent) networks

Recurrent networks are based on the work of Hopfield and contain feedback paths. Figure 10.23 shows a single-layer fully-connected recurrent network with a delay (z^{-1}) in the feedback path.

If, in Figure 10.23, the inputs occur at time (kT) and the outputs are predicted at time $(k+1)T$, then the network can be represented in matrix form by

$$\mathbf{y}(k+1)T = \mathbf{W}_1\mathbf{y}(kT) + \mathbf{W}_2\mathbf{x}(kT) \quad (10.57)$$

Equation (10.57) is in the same form as the discrete-time solution of the state equation (8.76).

10.3.4 Learning in neural networks

Learning in the context of a neural network is the process of adjusting the weights and biases in such a manner that for given inputs, the correct responses, or outputs are achieved. Learning algorithms include:

- (a) *Supervised learning*: The network is presented with training data that represents the range of input possibilities, together with the associated desired outputs. The weights are adjusted until the error between the actual and desired outputs meets some given minimum value.
- (b) *Unsupervised learning*: Also called open-loop adaption because the technique does not use feedback information to update the network's parameters. Applications for unsupervised learning include speech recognition and image compression. Important unsupervised networks include the Kohonen Self-Organizing Map (KSOM) which is a competitive network, and the Grossberg Adaptive Resonance Theory (ART), which can be used for on-line learning.

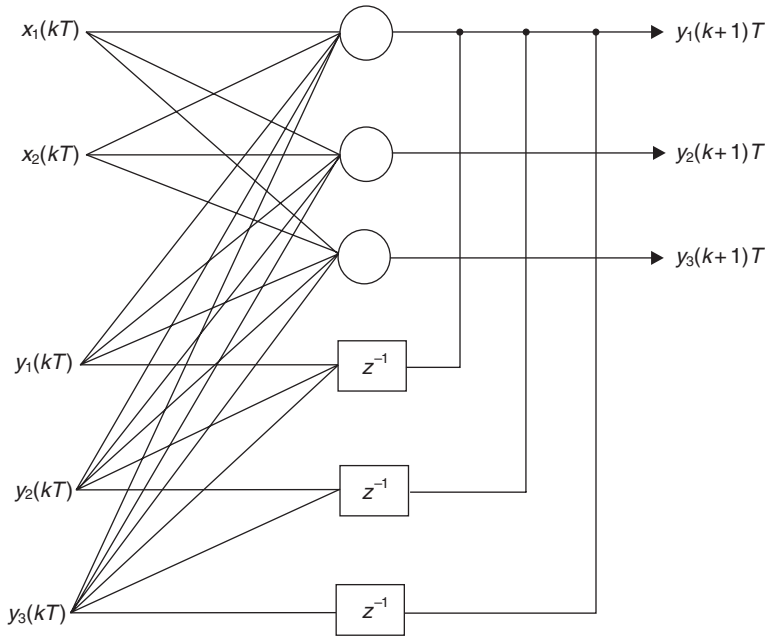


Fig. 10.23 Recurrent neural network.

10.3.5 Back-Propagation

The Back-Propagation Algorithm (BPA) is a supervised learning method for training ANNs, and is one of the most common forms of training techniques. It uses a gradient-descent optimization method, also referred to as the delta rule when applied to feedforward networks. A feedforward network that has employed the delta rule for training, is called a Multi-Layer Perceptron (MLP).

If the performance index or cost function J takes the form of a summed squared error function, then

$$J = \frac{1}{2} \sum_{j=1}^M (d_j - y_j)^2 \quad (10.58)$$

where d_j and y_j are the desired and actual network outputs respectively. Using gradient-descent, the weight increment Δw_{ji} is proportional to the (negative) slope

$$\Delta w_{ji} = -\mu \frac{\partial J}{\partial w_{ji}} \quad (10.59)$$

where μ is a constant. From equations (10.58) and (10.59),

$$\frac{\partial J}{\partial w_{ji}} = \frac{1}{M} \sum_{j=1}^M \frac{\partial}{\partial w_{ji}} (d_j - y_j)^2$$

using the chain rule,

$$\frac{\partial J}{\partial w_{ji}} = \frac{1}{M} \sum_{j=1}^M \frac{\partial}{\partial y_j} (d_j - y_j)^2 \frac{\partial y_i}{\partial w_{ji}}$$

giving

$$\frac{\partial J}{\partial w_{ji}} = -\frac{2}{M} \sum_{j=1}^M (d_j - y_j) \frac{\partial y_j}{\partial w_{ji}} \quad (10.60)$$

If the activation function is the sigmoid function given in equation (10.56), then its derivative is

$$\frac{\partial f}{\partial s_j} = -\frac{e^{-s_j}}{(1 + e^{-s_j})^2} = \frac{1}{1 + e^{-s_j}} - \left(\frac{1}{1 + e^{-s_j}} \right)^2$$

or

$$\frac{\partial f}{\partial s} = f(s) - f(s)^2 \quad (10.61)$$

Since $f(s)$ is the neuron output y_j , then equation (10.61) can be written as

$$\frac{\partial y_j}{\partial s_j} = y_j(1 - y_j) \quad (10.62)$$

From equation (10.60), again using the chain rule,

$$\frac{\partial y_j}{\partial w_{ji}} = \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_{ji}} \quad (10.63)$$

If, in equation (10.54), the bias b_j is called w_{j0} , then equation (10.54) may be written as

$$s_j = \sum_{i=0}^N w_{ji} x_i \quad (10.64)$$

thus

$$\frac{\partial s_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_{i=0}^N w_{ji} x_i = \sum_{i=0}^N \frac{\partial w_{ji}}{\partial w_{ji}} x_i = x_i \quad (10.65)$$

Substituting equations (10.62) and (10.65) into (10.63) gives

$$\frac{\partial y_i}{\partial w_{ji}} = y_j(1 - y_j)x_i \quad (10.66)$$

Putting equation (10.66) into (10.60) gives

$$\frac{\partial J}{\partial w_{ji}} = -\frac{2}{M} \sum_{j=1}^M (d_j - y_j) y_j (1 - y_j) x_i \quad (10.67)$$

or

$$\frac{\partial J}{\partial w_{ji}} = -\frac{2}{M} \sum_{j=1}^M \delta_j x_i \quad (10.68)$$

where

$$\delta_j = (d_j - y_j) y_j (1 - y_j) \quad (10.69)$$

Substituting equation (10.68) into (10.59) gives

$$\Delta w_{ji} = \eta \sum_{j=1}^M \delta_j x_i \quad (10.70)$$

where

$$\eta = \frac{2\mu}{M}$$

This leads to a weight increment, called the delta rule, for a particular neuron

$$\Delta w_{ji}(kT) = \eta \delta_j x_i \quad (10.71)$$

where η is the learning rate and has a value of between 0 and 1. Hence the new weight becomes

$$w_{ji}(kT) = w_{ji}(k-1)T + \Delta w_{ji}(kT)$$

or

$$w_{ji}(kT) = w_{ji}(k-1)T + \eta \delta_j x_i \quad (10.72)$$

Consider a three layer network. Let the input layer be layer one ($\ell = 1$), the hidden layer be layer two ($\ell = 2$) and the output layer be layer three ($\ell = 3$). The back-propagation commences with layer three where d_j is known and hence δ_j can be calculated using equation (10.69), and the weights adjusted using equation (10.71). To adjust the weights on the hidden layer ($\ell = 2$) equation (10.69) is replaced by

$$[\delta_j]_\ell = [y_j(1 - y_j)]_\ell \left[\sum_{j=1}^N w_{ji} \delta_j \right]_{\ell+1} \quad (10.73)$$

Hence the δ values for layer ℓ are calculated using the neuron outputs from layer ℓ (hidden layer) together with the summation of w and δ products from layer $\ell + 1$ (output layer). The back-propagation process continues until all weights have been adjusted. Then, using a new set of inputs, information is fed forward through the network (using the new weights) and errors at the output layer computed. The process continues until

- (i) the performance index J reaches an acceptable low value
- (ii) a maximum iteration count (number of epochs) has been exceeded
- (iii) a training-time period has been exceeded.

For either (ii) or (iii), it may well be that a local minima has been located. Under these conditions the BPA may be re-started, and if again unsuccessful, a new training set may be required.

The equations that govern the BPA can be summarized as

Single neuron summation

$$s_j = \sum_{i=1}^N w_{ji}x_i + b_j \quad (10.74)$$

Sigmoid activation function

$$y_j = \frac{1}{1 + e^{-s_j}} \quad (10.75)$$

Delta rule

$$\Delta w_{ji}(kT) = \eta \delta_j x_i \quad (10.76)$$

New weight

$$w_{ji}(kT) = w_{ji}(k-1)T + \Delta w_{ji}(kT) \quad (10.77)$$

Output layer

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \quad (10.78)$$

$$J = \frac{1}{2} \sum_{j=1}^M (d_j - y_j)^2 \quad (10.79)$$

Other layers

$$[\delta_j]_{\ell} = [y_j(1 - y_j)]_{\ell} \left[\sum_{j=1}^N w_{ji} \delta_j \right]_{\ell+1} \quad (10.80)$$

Learning with momentum

When momentum is used in the BPA, the solution stands less chance of becoming trapped in local minima. It can be included by making the current change in weight equal to a proportion of the previous weight change summed with the weight change calculated using the delta rule.

The delta rule given in equation (10.76) can be modified to include momentum as indicated in equation (10.81).

$$\Delta w_{ji}(kT) = (1 - \alpha)\eta\delta_j x_i + \alpha\Delta w_{ji}(k-1)T \quad (10.81)$$

where α is the momentum coefficient, and has a value of between 0 and 1.

Example 10.4

The neural network shown in Figure 10.24 is in the process of being trained using a BPA. The current inputs x_1 and x_2 have values of 0.2 and 0.6 respectively, and the desired output $d_j = 1$. The existing weights and biases are

Hidden layer

$$\mathbf{W}_j = \begin{bmatrix} 1.0 & 1.5 \\ 0.5 & 2.0 \\ 2.5 & 3.0 \end{bmatrix} \quad \mathbf{b}_j = \begin{bmatrix} 1.0 \\ -0.5 \\ 1.5 \end{bmatrix}$$

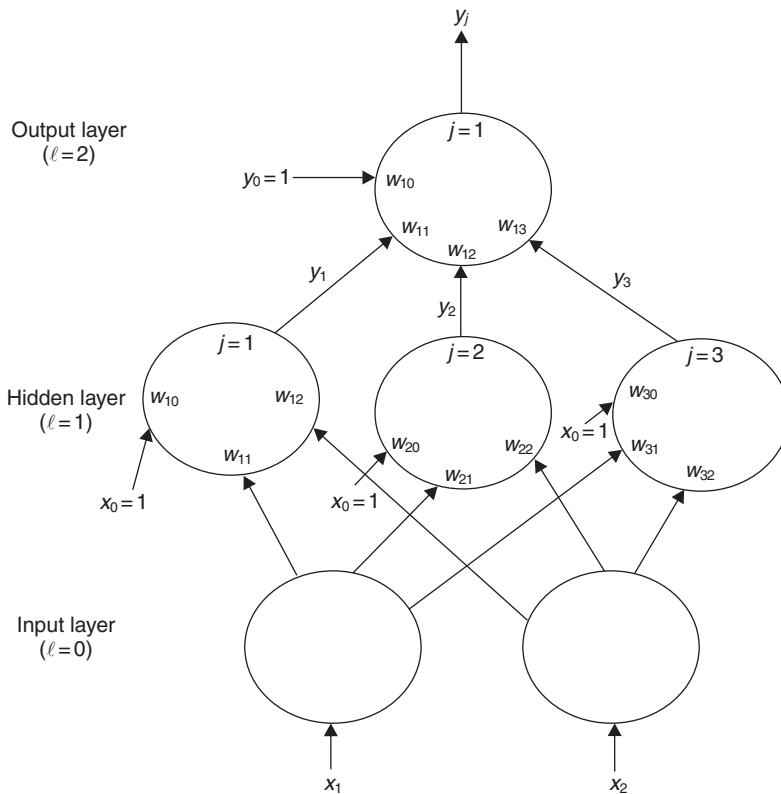


Fig. 10.24 Training using back-propagation.

Output layer

$$\mathbf{W}_j = [3.0 \quad 2.0 \quad 1.0] \quad \mathbf{b}_j = [-4.0]$$

Calculate the output y_j and hence the new values for the weights and biases. Assume a learning rate of 0.5.

Solution

Forward propagation

Hidden layer ($\ell = 1$): Single neuron summation

$$j = 1: \quad s_1 = w_{10} + w_{11}x_1 + w_{12}x_2$$

$$j = 2: \quad s_2 = w_{20} + w_{21}x_1 + w_{22}x_2$$

$$j = 3: \quad s_3 = w_{30} + w_{31}x_1 + w_{32}x_2$$

or

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} w_{10} \\ w_{20} \\ w_{30} \end{bmatrix} \quad (10.82)$$

Sigmoid activation functions ($j = 1$ to 3)

$$y_1 = \frac{1}{1 + e^{-s_1}} \quad y_2 = \frac{1}{1 + e^{-s_2}} \quad y_3 = \frac{1}{1 + e^{-s_3}} \quad (10.83)$$

Output layer ($\ell = 2$)

$$j = 1: \quad s_1 = w_{10} + w_{11}y_1 + w_{12}y_2 + w_{13}y_3 \quad (10.84)$$

$$y_j = y_1 = \frac{1}{1 + e^{-s_1}} \quad (10.85)$$

Inserting values into equations (10.82) and (10.83)

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1.0 & 1.5 \\ 0.5 & 2.0 \\ 2.5 & 3.0 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.6 \end{bmatrix} + \begin{bmatrix} 1.0 \\ -0.5 \\ 1.5 \end{bmatrix}$$

$$\begin{array}{lll} s_1 = 2.1 & s_2 = 0.8 & s_3 = 3.8 \\ y_1 = 0.891 & y_2 = 0.690 & y_3 = 0.978 \end{array}$$

Inserting values into equations (10.84) and (10.85)

$$s_1 = 1.031 \quad y_j = y_1 = 0.737 \quad (10.86)$$

Back propagation

Output layer ($\ell = 2$): From equation (10.69)

$$\delta_j = y_j(1 - y_j)(d_j - y_j)$$

Since $j = 1$

$$\begin{aligned} \delta_1 &= 0.737(1 - 0.737)(1 - 0.737) \\ &= 0.051 \end{aligned} \quad (10.87)$$

Delta rule

$$\begin{aligned}\Delta w_{ji}(kT) &= \eta \delta_j x_i \\ \Delta w_{10} &= 0.5 \times 0.051 \times 1 = 0.0255 \\ \Delta w_{11} &= 0.5 \times 0.051 \times 0.891 = 0.0227 \\ \Delta w_{12} &= 0.5 \times 0.051 \times 0.69 = 0.0176 \\ \Delta w_{13} &= 0.5 \times 0.051 \times 0.978 = 0.0249\end{aligned}$$

New weights and biases for output layer

$$\mathbf{w}_j = [3.0227 \quad 2.0176 \quad 1.0249] \quad \mathbf{b}_j = [-3.975] \quad (10.88)$$

Hidden layer ($\ell = 1$): From equation (10.80)

$$[\delta_j]_\ell = [y_j(1 - y_j)]_\ell \left[\sum_{j=1}^N w_{ji} \delta_j \right]_{\ell+1}$$

To illustrate this equation, had there been two neurons in layer ($\ell + 1$), i.e. the output layer, then values for δ_1 and δ_2 for layer ($\ell + 1$) would have been calculated. Thus, for layer ℓ (the hidden layer), the $[\delta_j]_\ell$ values would be

$$\begin{aligned}j = 1: \quad [\delta_1]_\ell &= [y_1(1 - y_1)]_\ell [w_{11}\delta_1 + w_{21}\delta_2]_{\ell+1} \\ j = 2: \quad [\delta_2]_\ell &= [y_2(1 - y_2)]_\ell [w_{12}\delta_1 + w_{22}\delta_2]_{\ell+1} \\ j = 3: \quad [\delta_3]_\ell &= [y_3(1 - y_3)]_\ell [w_{13}\delta_1 + w_{23}\delta_2]_{\ell+1}\end{aligned}$$

However, since in this example there is only a single neuron in layer ($\ell + 1$), $\delta_2 = 0$. Thus the δ values for layer ℓ are

$$\begin{aligned}j = 1: \quad [\delta_1]_\ell &= [0.891(1 - 0.891)][3.0227 \times 0.051] = 0.015 \\ j = 2: \quad [\delta_2]_\ell &= [0.690(1 - 0.690)][2.0176 \times 0.051] = 0.022 \\ j = 3: \quad [\delta_3]_\ell &= [0.978(1 - 0.978)][1.0249 \times 0.051] = 0.001\end{aligned} \quad (10.89)$$

Hence, using the delta rule, the weight increments for the hidden layer are

$$\begin{aligned}\Delta w_{10} &= 0.5 \times 0.015 \times 1 = 0.0075 \\ \Delta w_{11} &= 0.5 \times 0.015 \times 0.2 = 0.0015 \\ \Delta w_{12} &= 0.5 \times 0.015 \times 0.6 = 0.0045 \\ \Delta w_{20} &= 0.5 \times 0.022 \times 1 = 0.0110 \\ \Delta w_{21} &= 0.5 \times 0.022 \times 0.2 = 0.0022 \\ \Delta w_{22} &= 0.5 \times 0.022 \times 0.6 = 0.0066 \\ \Delta w_{30} &= 0.5 \times 0.001 \times 1 = 0.0005 \\ \Delta w_{31} &= 0.5 \times 0.001 \times 0.2 = 0.0001 \\ \Delta w_{32} &= 0.5 \times 0.001 \times 0.6 = 0.0003\end{aligned}$$

The new weights and biases for the hidden layer now become

$$\mathbf{W}_j = \begin{bmatrix} 1.0015 & 1.5045 \\ 0.5022 & 2.0066 \\ 2.5001 & 3.0003 \end{bmatrix} \quad \mathbf{b}_j = \begin{bmatrix} 1.0075 \\ -0.489 \\ 1.5005 \end{bmatrix} \quad (10.90)$$

10.3.6 Application of neural networks to modelling, estimation and control

An interesting and important feature of a neural network trained using back-propagation is that no knowledge of the process it is being trained to emulate is required. Also, since they learn from experience rather than programming, their use may be considered to be a 'black box' approach.

Neural networks in modelling and estimation

Providing input/output data is available, a neural network may be used to model the dynamics of an unknown plant. There is no constraint as to whether the plant is linear or nonlinear, providing that the training data covers the whole envelope of plant operation.

Consider the neural network state observer shown in Figure 10.25. This is similar in operation to the Luenberger full-order state observer given in Figure 8.9. If the neural network in Figure 10.25 is trained using back-propagation, the algorithm will minimize the PI

$$J = \sum_{k=1}^N (\mathbf{y}(kT) - \hat{\mathbf{y}}(kT))^T (\mathbf{y}(kT) - \hat{\mathbf{y}}(kT)) \quad (10.91)$$

Richter *et al.* (1997) used this technique to model the dynamic characteristics of a ship. The vessel was based on the Mariner Hull and had a length of 161 m and a displacement of 17 000 tonnes. The training data was provided by a three degree-of-freedom (forward velocity, or surge, lateral velocity, or sway and turn, or

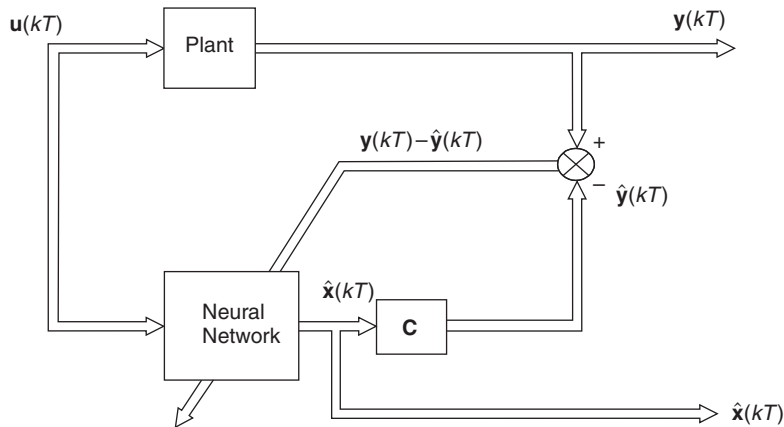
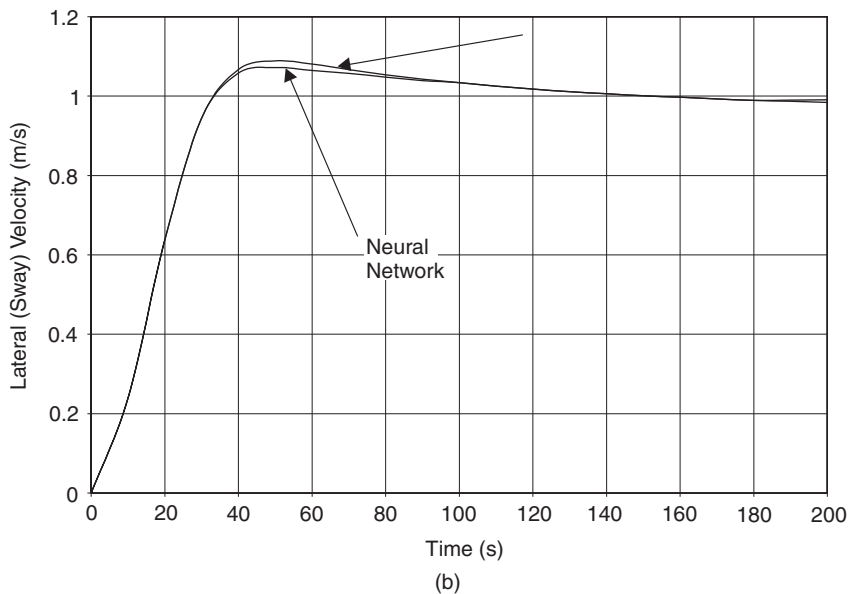
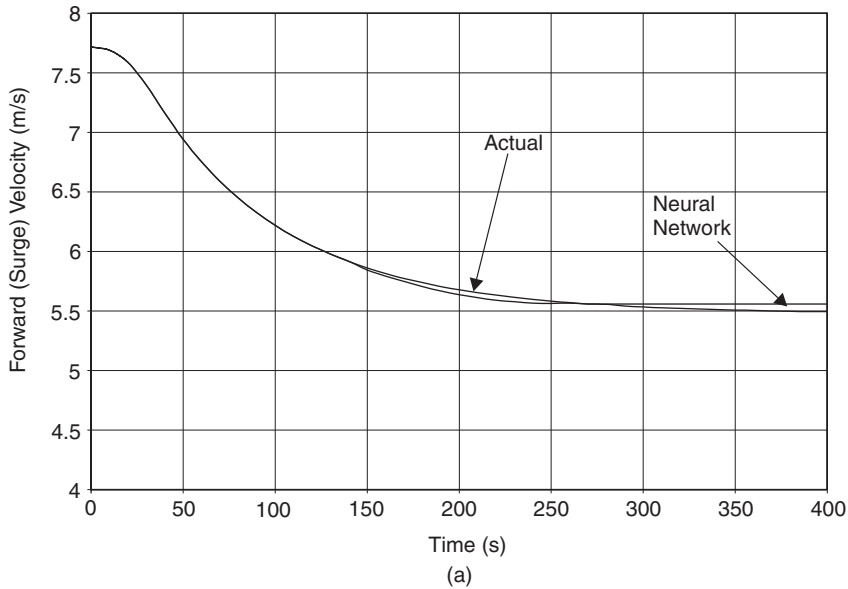


Fig. 10.25 Neural network state observer.

yaw-rate) differential equation model produced by Burns and was based on previous work by Morse and Price.

The training file consisted of input data of the form: Time elapsed $t(kT)$, Rudder angle $\delta(kT)$, Engine speed $n(kT)$ with corresponding output data Forward velocity $u(kT)$, Lateral velocity $v(kT)$, Yaw-rate $r(kT)$.

With the engine speed held constant, the rudder was given step changes of 0° , $\pm 10^\circ$, $\pm 20^\circ$ and $\pm 30^\circ$. Figure 10.26 shows training and trained data for a rudder



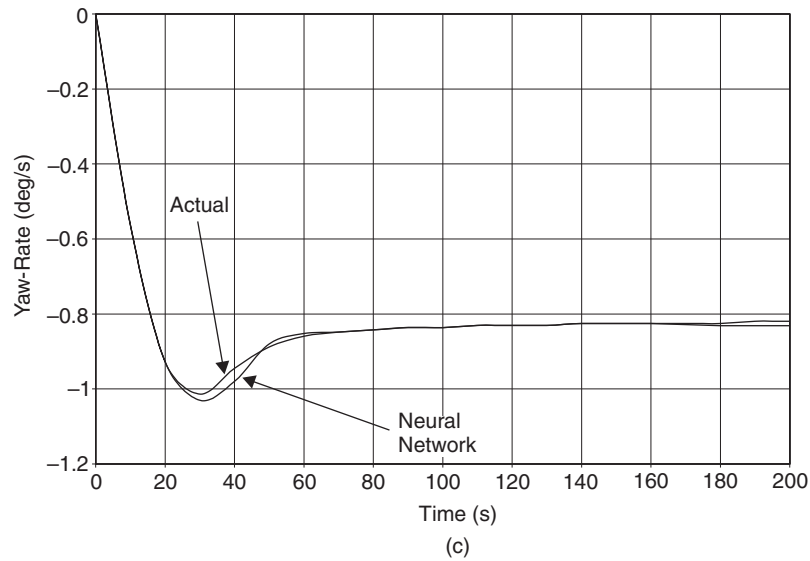


Fig. 10.26 Training and trained data for a neural network model of a Ship's Hull.

angle of $+20^\circ$ (where positive is to port, or left). The input and output data was sampled every 5 seconds, during the transient period of the turn (0–300 seconds).

The selected network had a 3-6-6-6-3 structure, i.e. input and output layers comprising 3 neurons in each, separated by three hidden layers of 6 neurons. During learning, 4 million epochs were trained. The learning rate and momentum were initially set at 0.3 and 0.8, but were reduced in three steps to final values of 0.05 and 0.4 respectively.

Inverse models

The inverse model of a plant provides a control vector $\mathbf{u}(kT)$ for a given output vector $\mathbf{y}(kT)$ as shown in Figure 10.27.

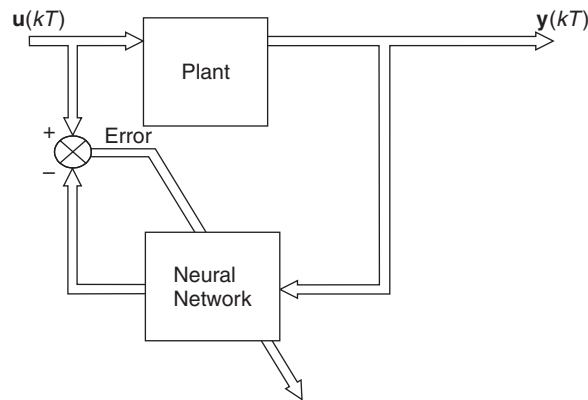


Fig. 10.27 Neural network plant inverse model.

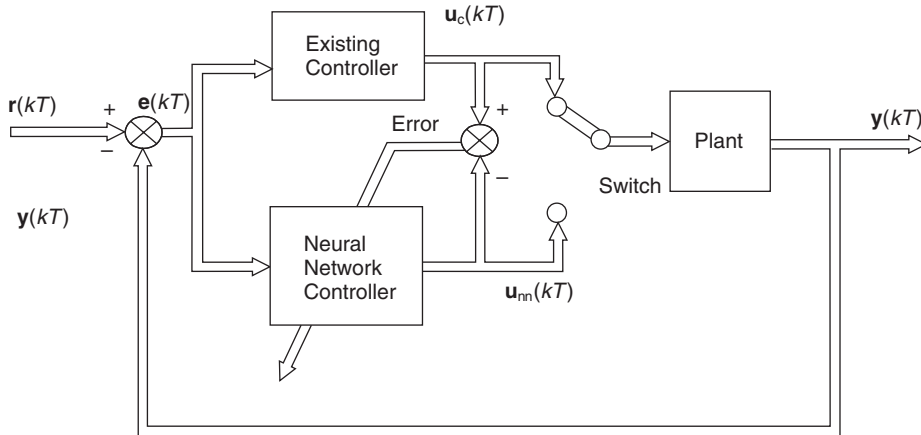


Fig. 10.28 Training a neural network controller.

So, for example, with the ship model shown in Figure 10.26, the inverse model could be trained with time, forward velocity, lateral velocity and yaw-rate as input data and rudder angle and engine speed as output data.

Neural networks in control

Controller emulation: A simple application in control is the use of neural networks to emulate the operation of existing controllers. It may be that a nonlinear plant requires several tuned PID controllers to operate over the full range of control actions. Or again, an LQ optimal controller has difficulty in running in real-time. Figure 10.28 shows how the control signal from an existing controller may be used to train, and to finally be replaced by, a neural network controller.

Error back-propagation through plant model

All closed-loop control systems operate by measuring the error between desired inputs and actual outputs. This does not, in itself, generate control action errors that may be back-propagated to train a neural network controller. If, however, a neural network of the plant exists, back-propagation through this network of the system error ($r(kT) - y(kT)$) will provide the necessary control action errors to train the neural network controller as shown in Figure 10.29.

Internal Model Control (IMC)

Internal Model Control was discussed in relation to robust control in section 9.6.3 and Figure 9.19. The IMC structure is also applicable to neural network control. The plant model $G_m(s)$ in Figure 9.19 is replaced by a neural network model and the controller $C(s)$ by an inverse neural network plant model as shown in Figure 10.30.

10.3.7 Neurofuzzy control

Neurofuzzy control combines the mapping and learning ability of an artificial neural network with the linguistic and fuzzy inference advantages of fuzzy logic. Thus

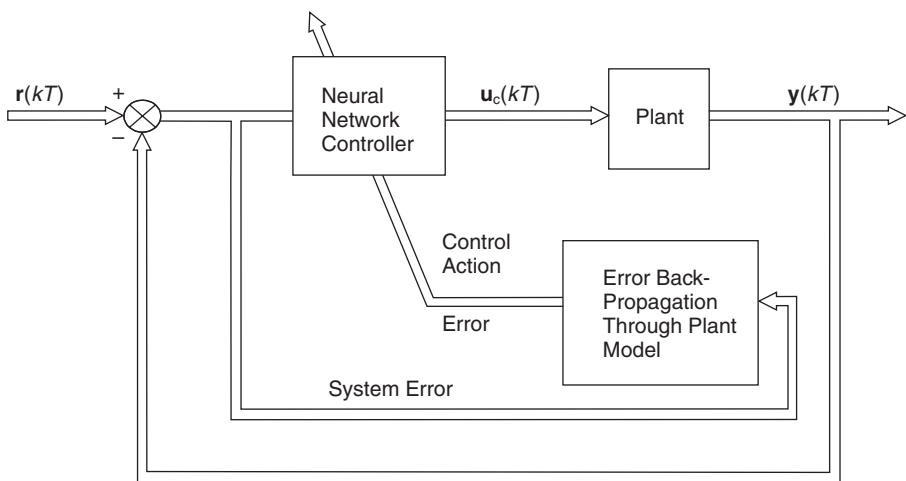


Fig. 10.29 Control action error generated by system error back-propagation through plant model.

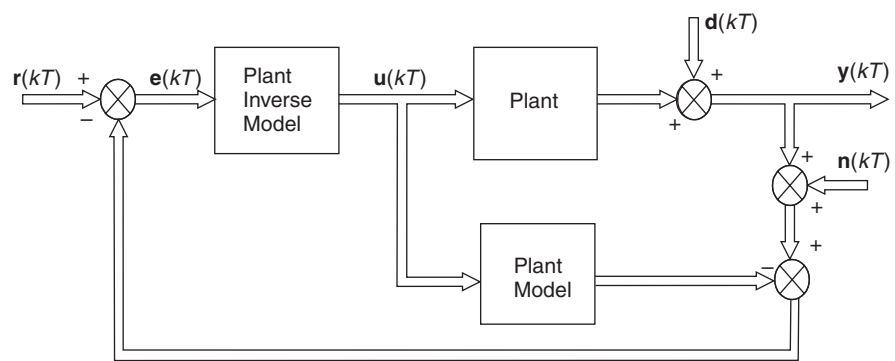


Fig. 10.30 Application of neural networks to IMC.

a neurofuzzy controller has the potential to out-perform conventional ANN or fuzzy logic controllers. The general architecture of a neurofuzzy scheme is to employ neural network learning to upgrade either the membership functions or rulebase of the fuzzy logic element.

The Adaptive Network based Fuzzy Inference System (ANFIS)

The ANFIS neurofuzzy controller was implemented by Jang (1993) and employs a Takagi–Sugeno–Kang (TSK) fuzzy inference system. The basic ANFIS architecture is shown in Figure 10.31.

Square nodes in the ANFIS structure denote parameter sets of the membership functions of the TSK fuzzy system. Circular nodes are static/non-modifiable and perform operations such as product or max/min calculations. A hybrid learning rule is used to accelerate parameter adaption. This uses sequential least squares in the forward pass to identify consequent parameters, and back-propagation in the backward pass to establish the premise parameters.

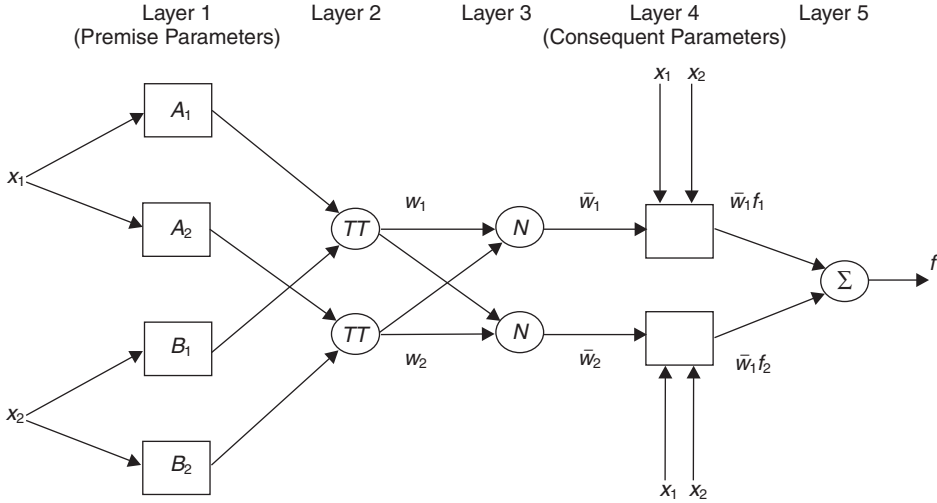


Fig. 10.31 The Adaptive Network based Fuzzy Inference System (ANFIS) architecture (after Craven).

If the fuzzy inference system has inputs x_1 and x_2 and output f as shown in Figure 10.31, then a first-order TSK rulebase might be

- Rule 1: If x_1 is A_1 and x_2 is B_1
 then $f_1 = p_1x_1 + q_1x_2 + r_1$
- Rule 2: If x_1 is A_2 and x_2 is B_2
 then $f_2 = p_2x_1 + q_2x_2 + r_2$
- Rule n : If x_1 is A_n and x_2 is B_n
 then $f_n = p_nx_1 + q_nx_2 + r_n$ (10.92)

Where $A_1 \dots A_n, B_1 \dots B_n$ are membership functions and $p_1 \dots p_n, q_1 \dots q_n$ and $r_1 \dots r_n$ are constants within the consequent functions.

Layer 1 contains adaptive nodes that require suitable premise membership functions (triangular, trapezoidal, bell, etc). Hence

$$y_{1,i} = \mu_{8i}(x_i) \quad (10.93)$$

Layer 2 undertakes a product or T-norm operation.

$$y_{2,i} = w_i = \mu_{Ai}(x_1)\mu_{Bi}(x_2) \dots \mu_{pi}(x_n) \quad i = 1, 2, \dots, n \quad (10.94)$$

Layer 3 calculates the ratio of the firing strength of the rules

$$y_{3,i} = \bar{w}_i = \frac{w_i}{\sum_{i=1}^n w_i} \quad (10.95)$$

Layer 4 generates the linear consequent functions as given in equation (10.92). Layer 5 sums all incoming signals

$$y_{5,i} = f = \sum_{i=1}^n \bar{w}_i f_i = \frac{\sum_{i=1}^n w_i f_i}{\sum_{i=1}^n w_i} \quad (10.96)$$

A limitation of the ANFIS technique is that it cannot be employed on multivariable systems. The Co-active ANFIS (CANFIS) developed by Craven (1999) extends the ANFIS architecture to provide a flexible multivariable control environment. This was employed to control the yaw and roll channels of an Autonomous Underwater Vehicle (AUV) simultaneously.

Predictive Self-Organizing Fuzzy Logic Control (PSOFLC)

This is an extension of the SOFLC strategy discussed in section 10.2.5 and illustrated in Figure 10.17. Predictive Self-Organizing Fuzzy Logic Control is particularly useful when the plant dynamics are time-varying, and the general architecture is shown in Figure 10.32.

In Figure 10.30 the predictive neural network model tracks the changing dynamics of the plant. Following a suitable time delay, $e_m(kT)$ is passed to the performance index table. If this indicates poor performance as a result of changed plant dynamics, the rulebase is adjusted accordingly. Richter (2000) demonstrated that this technique could improve and stabilize a SOFLC when applied to the autopilot of a small motorized surface vessel.

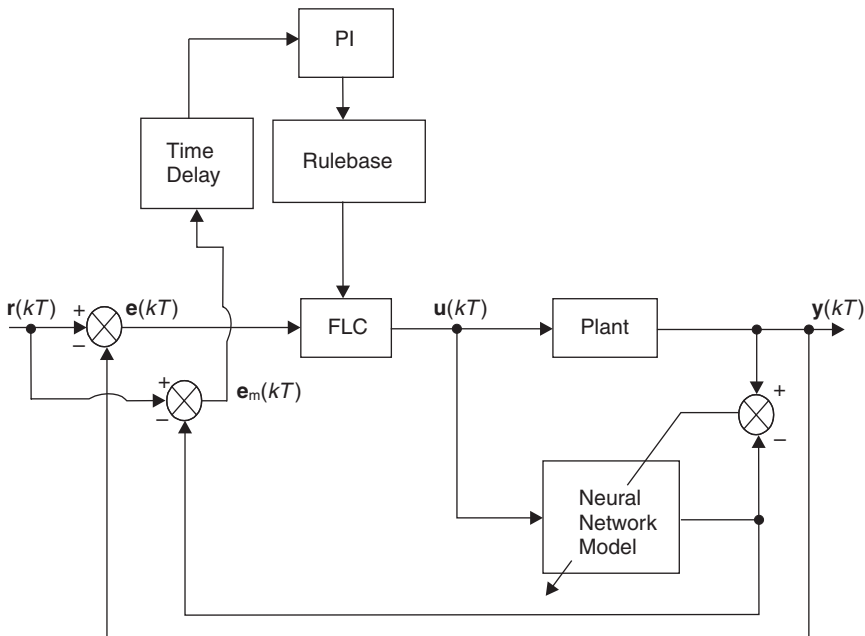


Fig. 10.32 Predictive Self-Organizing Fuzzy Logic Control (PSOFLC).

10.4 Genetic algorithms and their application to control system design

10.4.1 Evolutionary design techniques

In any design problem there is a multi-dimensional space of possible solutions. Some of these solutions may be acceptable, but not the best (local optima) and there may exist a single best solution (global optimum).

It has been shown in Chapter 9 that it is possible to obtain an optimal mathematical solution for a control system with linear plant dynamics. An alternative approach is to use 'heuristics', or knowledge acquired through experience, to search for optimal solutions. One such technique is to employ a Genetic Algorithm (GA). This is a search algorithm based upon the evolutionary process of natural selection of the fittest members of a given population to breed the next generation.

10.4.2 The genetic algorithm (GA)

In the early 1960s Rechenburg (1965) conducted studies at the Technical University of Berlin in the use of an evolutionary strategy to minimize drag on a steel plate. Genetic algorithms were used by Holland (1975) and his students at the University of Michigan in the late 1970s and early 1980s to analyse a range of engineering problems. In particular, Goldberg (1983) used GAs to optimize the design of gas pipeline systems.

The basic element of a GA is the chromosome. This contains the genetic information for a given solution and is typically coded as a binary string. For example, an eight digit binary number such as 11001001 represents a chromosome that contains eight genes. Initially, a population of chromosomes, created randomly, represent a number of solutions to a given problem.

A 'fitness function', which is in effect a performance index, is used to select the best solutions in the population to be parents to the offsprings that will comprise the next generation. The fitter the parent, the greater the probability of selection. This emulates the evolutionary process of 'survival of the fittest'. Parents are selected using a roulette wheel method as shown in Figure 10.33. Here there are four candidate parents P1, P2, P3 and P4, having selection probabilities (from the fitness function) of 0.5, 0.3, 0.15 and 0.05 respectively. For the example in Figure 10.33, if the roulette wheel is spun four times, P1 may be selected twice, P2 and P3 once, and P4 not at all.

Offsprings are produced by selecting parent chromosomes for breeding, and crossing over some of the genetic material. The amount of genetic material passed from parent to offspring is dictated by the random selection of a crossover point as indicated in Figure 10.34, where P1 and P2 are the parents, and O1 and O2 the offsprings.

Mutation is allowed to occur in some of the offsprings, the amount being controlled by the mutation rate, typically a very small number. This results in the random change in a gene in an offspring, i.e. from 0 to 1.

The breeding of successive generations continues until all offsprings are acceptably fit. In some cases, all offsprings will eventually have the same genetic structure,

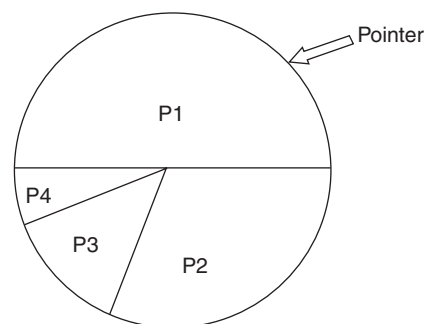


Fig. 10.33 Roulette wheel selection.

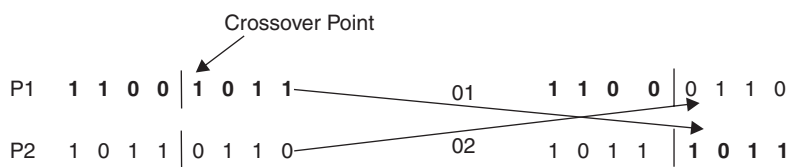


Fig. 10.34 Genetic material transfer during breeding.

representing a global optimum, or in other cases, several solutions, called clustering may evolve. In this latter case, the system designer makes the decision as to which is the best solution.

Example 10.5

A system has a fitness function

$$J = 1 + \sin ax \tag{10.97}$$

as shown in Figure 10.35. Assume that the solution space has 31 values and that each solution can be represented by a five digit binary string ranging from 00000 to 11111. The value of a in equation (10.97) is therefore 11.6° (0.203 rad). If the population has four members, spinning a coin (heads = 1, tails = 0) produced the following initial population

00101 11110 00001 00011

Determine the offsprings from the initial generation and the subsequent generation.

Solution

From Figure 10.35 it can be seen that the optimal solution occurs when $x_{10} = 8$, or $x_2 = 01000$.

Table 10.1 shows the selection of parents for mating from the initial population. If a random number generator is used to generate numbers between 0.0 and 1.0, then the cumulative probability values in Table 10.1 is used as follows:

- Values between 0 and 0.342, Parent 1 selected
- Values between 0.343 and 0.488, Parent 2 selected

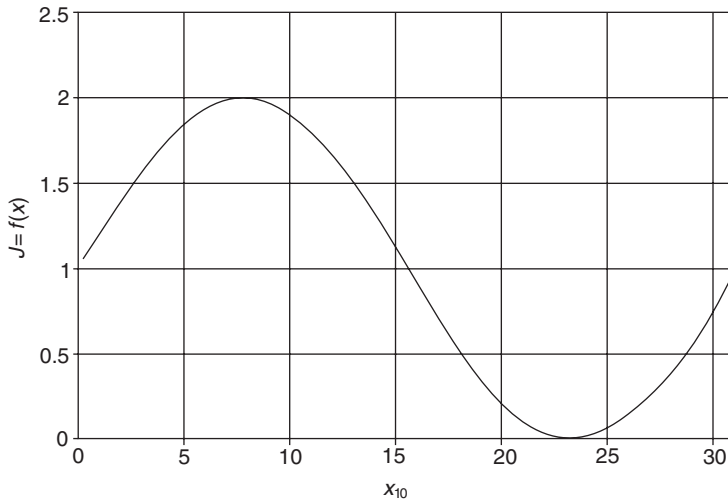


Fig. 10.35 Fitness function for Example 10.5.

Table 10.1 Selection of parents for mating from initial population

Parent	x_2	x_{10}	$J = f(x)$	$p = J/\Sigma J$	Cumulative probability	Roulette wheel hits
1	00101	5	1.848	0.342	0.342	2
2	11110	30	0.792	0.146	0.488	1
3	00001	1	1.201	0.222	0.710	0
4	00011	3	1.571	0.290	1.000	1
Total			5.412	1.000		4
Mean			1.353	0.250		1
Maximum			1.848	0.342		2

Values between 0.489 and 0.710, Parent 3 selected

Values between 0.711 and 1.000, Parent 4 selected

The random number generator produced the following values: 0.326, 0.412, 0.862 and 0.067. Hence Parent 1 was selected twice, Parents 2 and 4 once and Parent 3 not at all. The selected parents were randomly mated with random choice of crossover points. The fitness of the first generation of offsprings is shown in Table 10.2.

From Tables 10.1 and 10.2 the total fitness of the initial population was 5.412, whereas the total fitness of their offsprings was 5.956, an increase of 10%. Note that if each offspring was perfect, they would have a value of 8_{10} , or 01000_2 , thus giving the maximum fitness that any population could have of $2 \times 4 = 8.0$.

The next spin of the random number generator produced values: 0.814, 0.236, 0.481 and 0.712, giving the roulette wheel hits shown in Table 10.2.

The second generation of offsprings is shown in Table 10.3. From Table 10.3 the total fitness of the second generation of offsprings is 7.204, or an increase of 33% above the initial population. As things stand, since the two most significant binary digits in the second generation offsprings are 00, subsequent breeding will not

Table 10.2 Fitness of first generation of offsprings

Parent	Breeding	Offspring	x_2	x_{10}	$J = f(x)$	$p = J / \sum J$	Cumulative probability	Roulette wheel hits
1	001101	1	00110	6	1.937	0.325	0.325	1
2	11110	2	11101	29	0.600	0.101	0.426	0
1	00101	3	00011	3	1.571	0.264	0.690	1
4	00011	4	00101	5	1.848	0.310	1.000	2
Total					5.956	1.000		
Mean					1.489	0.250		
Maximum					1.937	0.325		

Table 10.3 Fitness of second generation of offsprings

Parent	Breeding	Offspring	x_2	x_{10}	$J = f(x)$
1	001101	1	00101	5	1.848
4	00101	2	00110	6	1.937
3	00011	3	00101	5	1.848
4	00101	4	00011	3	1.571
Total					7.204
Mean					1.801
Maximum					1.937

produce strings greater than 00111, or 7₁₀. If all four offsprings had this value, then a total fitness for the population would be 7.953, which is close to, but not the ideal fitness of 8.0 as explained earlier. However, if mutation in a particular offspring changed one of the two most significant digits from 0 to 1, a perfect optimal value could still be achieved. Also to bear in mind, is the very limited size of the population used in this example.

Example 10.6

The block diagram for an angular positional control system is shown in Figure 10.36. The system parameters are:

Amplifier gain $K_2 = 3.5$

Servomotor constant $K_3 = 15 \text{ Nm/A}$

Field resistance $R_f = 20 \Omega$

Gear ratio $n = 5$

Equivalent moment of inertia of motor and load $I_e = 1.3 \text{ kgm}^2$

Sampling time $T = 0.05$ seconds.

Using a GA with a population of 10 members, find the values of the controller gain K_1 and the tachogenerator constant K_4 that maximizes the fitness function

$$J = 100 \left/ \sum_{k=0}^{N-1} \{(\theta_i(kT) - \theta_0(kT))^2 T\} \right. \quad (10.98)$$

when the system is subjected to a unit step at time $kT = 0$. Perform the summation over a time period of 2 seconds ($N = 40$). Allow a search space of 0–15 for K_1 and

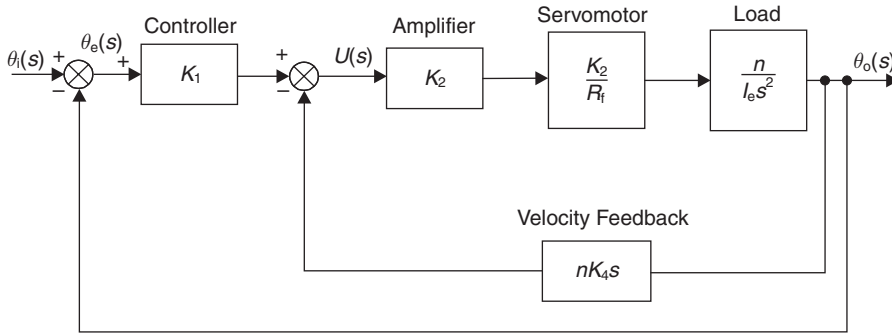


Fig. 10.36 Angular positional control system.

0–1 for K_4 . Assume that the solution space has 255 values and that each solution can be represented by an eight digit binary string, the first four digits representing K_1 and the second four representing K_4 .

Solution

The plant transfer function is

$$\frac{\theta_0}{U}(s) = \frac{K_2 K_3 n}{R_f I_e s^2} \quad (10.99)$$

If the state variables are $x_1 = \theta_0(t)$ and $x_2 = \dot{\theta}_0(t) = \dot{x}_1$, then the state equations become

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_2 K_3 n}{R_f I_e} \end{bmatrix} u$$

and the output equations

$$\begin{bmatrix} \theta_0 \\ \dot{\theta}_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Inserting values into the state equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 10.096 \end{bmatrix} u \quad (10.100)$$

For a sampling time of 0.05 seconds, the discrete form of equation (10.100) is

$$\begin{bmatrix} x_1(k+1)T \\ x_2(k+1)T \end{bmatrix} = \begin{bmatrix} 1 & 0.05 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(kT) \\ x_2(kT) \end{bmatrix} + \begin{bmatrix} 0.0126 \\ 0.5048 \end{bmatrix} u(kT) \quad (10.101)$$

From Figure 10.36 the control law is

$$u(kT) = K_1(\theta_i(kT) - x_1(kT)) - nK_4 x_2(kT) \quad (10.102)$$

where $\theta_i(kT) = 1.0$ for all $kT > 0$.

Hence values of K_1 and K_4 generated by the GA are inserted into equation (10.102) and the control $u(kT)$ used to drive the discrete plant equation (10.101). The fitness function J is updated at each sampling instant to give an overall value at the end of each simulation. For a population of 10 members, 10 simulations are required per generation.

Since the required search space is K_1 0–15, K_4 0–1, the following are examples of population membership

K_1	K_4		
1111	1111	$K_1 = 15$	$K_4 = 1$
0011	0111	$K_1 = 3$	$K_4 = 7/15 = 0.467$
1001	1100	$K_1 = 9$	$K_4 = 12/15 = 0.8$

Table 10.4 shows the parent selection for mating from a randomly seeded initial population. The random numbers (0–1) from the roulette wheel spins were: 0.145, 0.422, 0.977, 0.339, 0.607, 0.419, 0.075, 0.027, 0.846, 0.047.

The fitness of the first generation of offsprings is given in Table 10.5. Further breeding produced the sixth generation of offsprings given in Table 10.6. Inspection of Table 10.6 reveals that values of $K_1 = 15$, $K_4 = 0.333$ produces a global maximum of $J = 1065$. Figure 10.37 shows the unit step response of the system in Example 10.6 for

- (a) the maximum of the first generation of offsprings ($J = 841$)
- (b) the global maximum of the sixth generation of offsprings ($J = 1065$)

Use of Schemata (Similarity templates): As the progression through generations of solutions takes place, there evolves certain similarities between genes within chromosomes. These similarities can be exploited using a similarity template or schema, that sits within a schemata framework.

A schema employs a don't care symbol '*', so, for example, the sixth generation of offsprings in Table 10.6 could have employed the template

$$111**1**$$

The use of schemata will aid the speed of convergence.

Table 10.4 Parent selection from initial population for Example 10.6

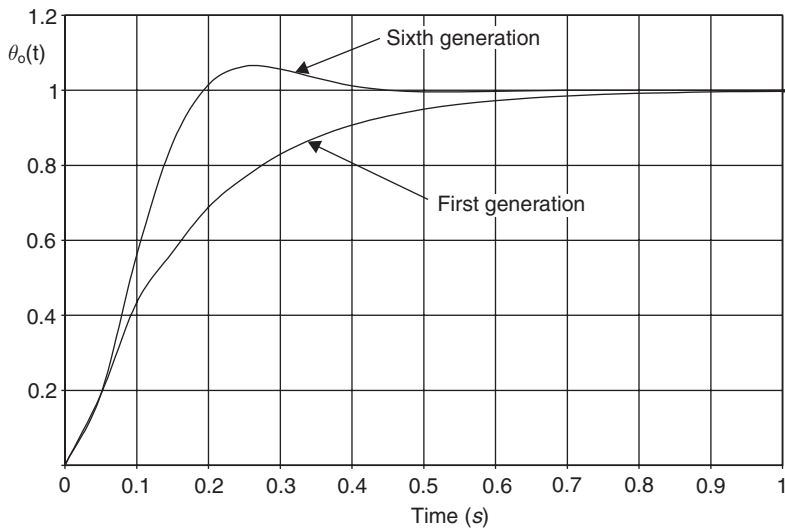
Parent	K_1	K_4	J	$p = J / \sum J$	Cumulative probability	Roulette wheel hits
1011 1001	11	0.600	647	0.157	0.157	4
0100 1010	4	0.667	233	0.056	0.213	0
0001 0111	1	0.400	112	0.027	0.240	0
1001 1010	9	0.667	497	0.122	0.362	1
1110 0111	14	0.467	923	0.224	0.586	2
0101 0001	5	0.067	375	0.092	0.678	1
1001 1101	9	0.867	18	0.004	0.682	0
0110 1101	6	0.867	46	0.011	0.693	0
1101 1010	13	0.667	691	0.168	0.861	1
0101 0100	5	0.267	573	0.139	1.000	1
Total			4115	1.000		10
Mean			411.5	0.100		1
Maximum			923	0.224		4

Table 10.5 Fitness of first generation of offsprings for Example 10.6

<i>Parent</i>	<i>Offspring</i>	K_1	K_4	J
1011 1001	1010 0111	10	0.467	712
1110 0111	1111 1001	15	0.600	841
0101 0001	0101 0100	5	0.267	573
0101 0100	0101 0001	5	0.067	375
1011 1001	1011 1010	11	0.667	596
1001 1010	1001 1001	9	0.600	541
1101 1010	1101 1001	13	0.600	747
1011 1001	1011 1010	11	0.667	596
1011 1001	1010 0111	10	0.467	713
1110 0111	1111 1001	15	0.600	841
Total				6535
Mean				653.5
Maximum				841

Table 10.6 Fitness of sixth generation of offsprings for Example 10.6

<i>Offspring</i>	K_1	K_4	J
1110 0101	14	0.333	1030
1111 0110	15	0.400	1029
1110 0100	14	0.267	1021
1111 0101	15	0.333	1065
1111 0111	15	0.467	970
1111 0100	15	0.267	1041
1110 1000	14	0.533	858
1110 0111	14	0.467	923
1111 1000	15	0.533	905
1111 0101	15	0.333	1065
Total			9907
Mean			990.7
Maximum			1065

**Fig. 10.37** Comparison between best first generation and best sixth generations solutions for Example 10.6.

Other applications of Genetic Algorithms

- (a) *Optimal control:* Optimal control problems such as the linear quadratic regulator discussed in section 9.2 can also be solved using GAs. The discrete quadratic performance index given in equation (9.28) can be employed as a (minimum) fitness function directly. Alternatively, as shown in Example 10.6 the reciprocal of equation (9.28) provides a (maximum) fitness function.
- (b) *Self-Organizing Fuzzy Logic Control:* Genetic Algorithms may be used to adapt both membership functions and rulebase structures in a SOFLC system.

Figure 10.38 shows an input window with three triangular fuzzy sets NB, Z and PB. Each set is positioned in its regime of operation by the centre parameter c so that, for example, NB can only operate on the negative side of the universe of discourse. The width of each set is controlled by parameter w .

The chromosome string could take the form

$$[c_1 \ w_1 \ c_2 \ w_2 \ c_3 \ w_3] \tag{10.103}$$

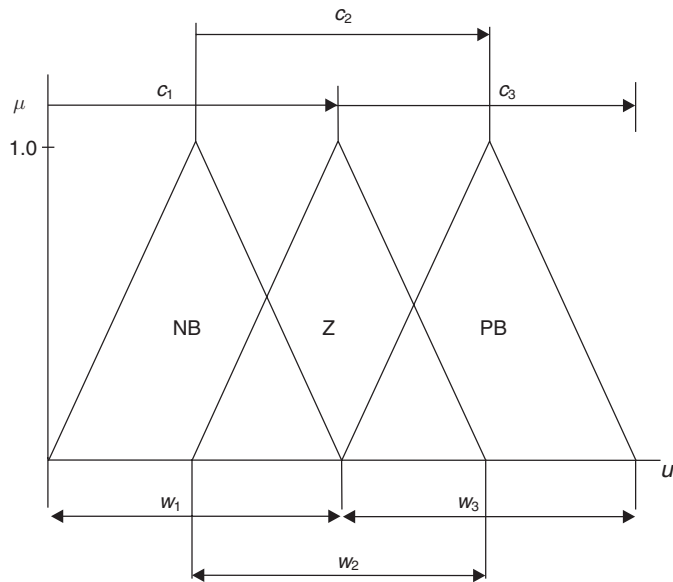


Fig. 10.38 Adaption of membership function features using genetic algorithms.

If each parameter is a 4-bit string, the configuration in Figure 10.38 would be represented by

$$[100011111000111110001111] \tag{10.104}$$

10.4.3. Alternative search strategies

In addition to evolutionary search strategies such as GAs, there are a number of other search techniques that are employed for design optimization.

Simulated annealing

The physical annealing process consists of heating a metal up to a prescribed temperature and then allowing it to cool slowly. During cooling, the molecules form crystals which are in a minimum energy condition. The metal, therefore, settles to a global minimum energy state.

With simulated annealing, an energy term E is defined, which then becomes the performance index to be minimized. For a given energy change ΔE at temperature T , the probability P of accepting a solution is

$$P = e^{-\Delta E/kT} \quad (10.105)$$

where k is the Boltzmann constant. If the energy of the system has decreased, then the system may move to the new state, based on the probability given by equation (10.105). As cooling proceeds along a prescribed schedule, the system avoids local minima, and settles down to a global minimum energy state.

Tabu search

Like simulated annealing, tabu search is a technique designed to avoid the problem of becoming trapped in local optima. The procedure is basically hill-climbing, which commences at an initial solution and searches the neighbourhood for a better solution. However, the process will recognize, and avoid areas of the solution space that it has already encountered, thus making these areas 'tabu'. The tabu moves are kept in a finite list, which is updated as the search proceeds.

10.5 Further problems

Example 10.7

A fuzzy logic controller has input and output fuzzy windows as shown in Figure 10.39. The fuzzy rulebase is given in Figure 10.40. If defuzzification is by the centre of area method, calculate crisp control signals $u(t)$ when the error $e(t)$ and the rate of change of error $ce(t)$ have the following values:

	$e(t)$	$ce(t)$	[Answer $u(t)$]
(i)	0	-2	-6.67
(ii)	-4	-1	-2.66
(iii)	1	1.5	3.01
(iv)	4	-1.5	0.106

Note: For the centre of area method, use only those values inside the dotted lines in the output window.

Example 10.8

The angular positional control system shown by the block diagram in Figure 10.36 is to have the velocity feedback loop removed and controller K_1 replaced by a fuzzy logic controller (FLC) as demonstrated by Barrett (1992). The inputs to the FLC

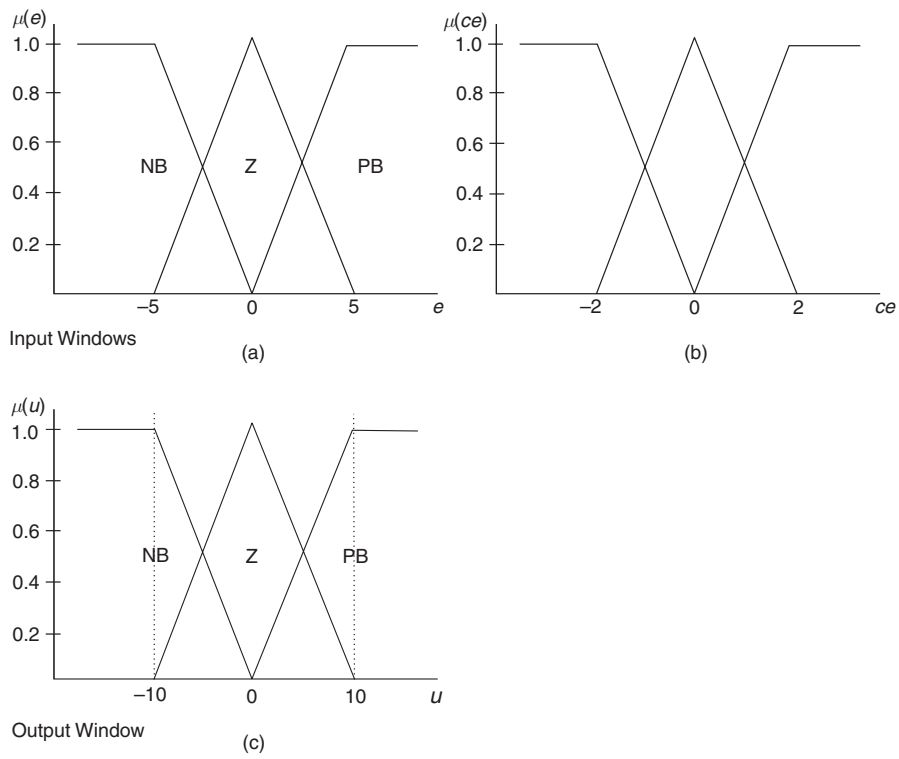


Fig. 10.39 Input and output fuzzy windows for Example 10.7.

$e \backslash ce$	NB	Z	PB
NB	NB	NB	Z
Z	NB	Z	PB
PB	Z	PB	PB

Fig. 10.40 Fuzzy rulebase for Example 10.7.

are angular error $e(kT)$ and rate-of-change of angular error $ce(kT)$. The output from the FLC is a control signal $u(kT)$. The input and output fuzzy windows have the same number of fuzzy sets, but different scales for the universes of discourse as shown in Figure 10.41. The fuzzy rulebase is given in Figure 10.42. If the state variables are

$$\begin{aligned}x_1 &= \theta_0(t) \\x_2 &= \dot{\theta}_0(t) = \dot{x}_1\end{aligned}$$

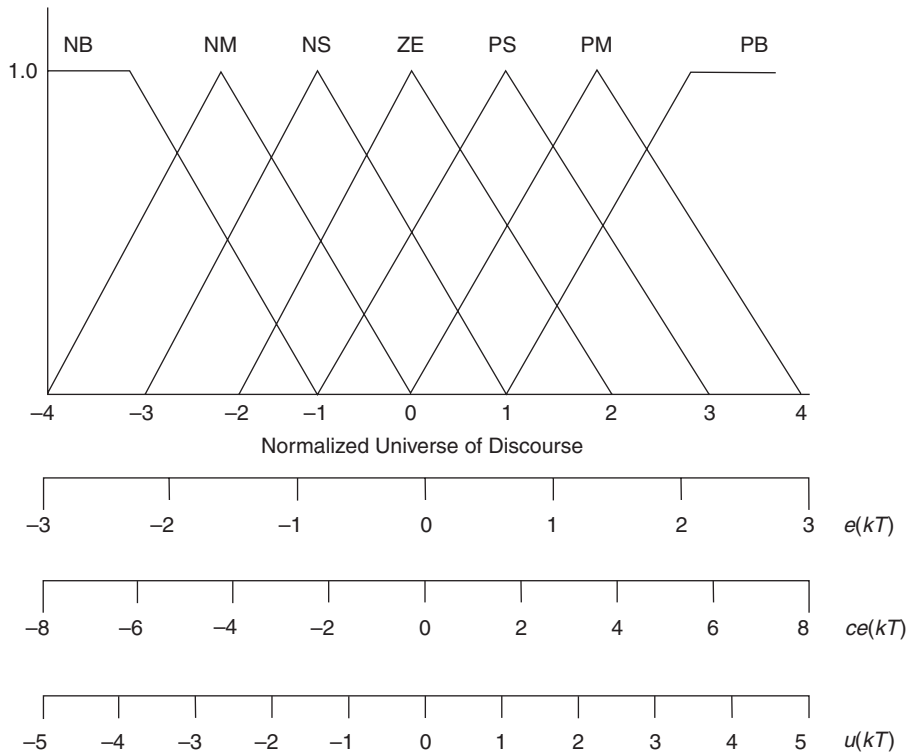


Fig. 10.41 Input and output fuzzy windows for Example 10.8.

e							
ce	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NM	NS	ZE	PM	PB
NM	NB	NB	NM	NM	PS	PM	PB
NS	NM	NS	NS	NS	PS	PM	PM
ZE	NM	NS	NS	ZE	PS	PS	PM
PS	NM	NM	NS	PS	PS	PS	PM
PM	NB	NM	NS	PM	PM	PB	PB
PB	NB	NM	ZE	PS	PM	PB	PB

Fig. 10.42 Fuzzy rulebase for Example 10.8.

then the state and output equations for the plant are as given in equation (10.100)

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 10.096 \end{bmatrix} u \\ \begin{bmatrix} \theta_1 \\ \dot{\theta}_2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned} \quad (10.106)$$

and, for a sampling time of 0.2 seconds, the discrete form of equation (10.106) is

$$\begin{bmatrix} x_1(k+1)T \\ x_2(k+1)T \end{bmatrix} = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(kT) \\ x_2(kT) \end{bmatrix} + \begin{bmatrix} 0.2019 \\ 2.0192 \end{bmatrix} u(kT) \quad (10.107)$$

If the system, which is initially at rest when $kT = 0$, is given a unit step input at this time, determine $e(kT)$, $ce(kT)$, $u(kT)$, $x_1(kT)$ and $x_2(kT)$ for values of $kT = 0$, 0.2, 0.4 and 0.6 seconds.

Assume that

$$e(kT) = (\theta_i - \theta_0) = 1 - x_1(kT)$$

$$ce(kT) = d/dt(\theta_i - \theta_0)$$

$$= \frac{d\theta_i}{dt} - \frac{d\theta_0}{dt}$$

$$= 0 - \frac{d\theta_0}{dt} = -x_2(kT)$$

Solution

Table 10.7 Solution to Example 10.8

kT	$x_1(kT)$	$x_2(kT)$	$e(kT)$	$ce(kT)$	$u(kT)$	$x_1(k+1)T$	$x_2(k+1)T$
0	0	0	1	0	1.083	0.2187	2.187
0.2	0.2187	2.187	0.7813	-2.187	0.666	0.7906	3.532
0.4	0.7906	3.532	0.2094	-3.532	-1.32	1.2305	0.867
0.6	1.2305	0.867	-0.2305	-0.867	-0.511	1.3007	-0.165

Example 10.9

A neural network has a structure as shown in Figure 10.43. Assuming that all the activation functions are sigmoids, calculate the values of y_{12} and y_{22} when the inputs are

(a) $x_1 = 0.3$, $x_2 = 0.5$

(b) $x_1 = 0.9$, $x_2 = 0.1$

when the weights and biases are

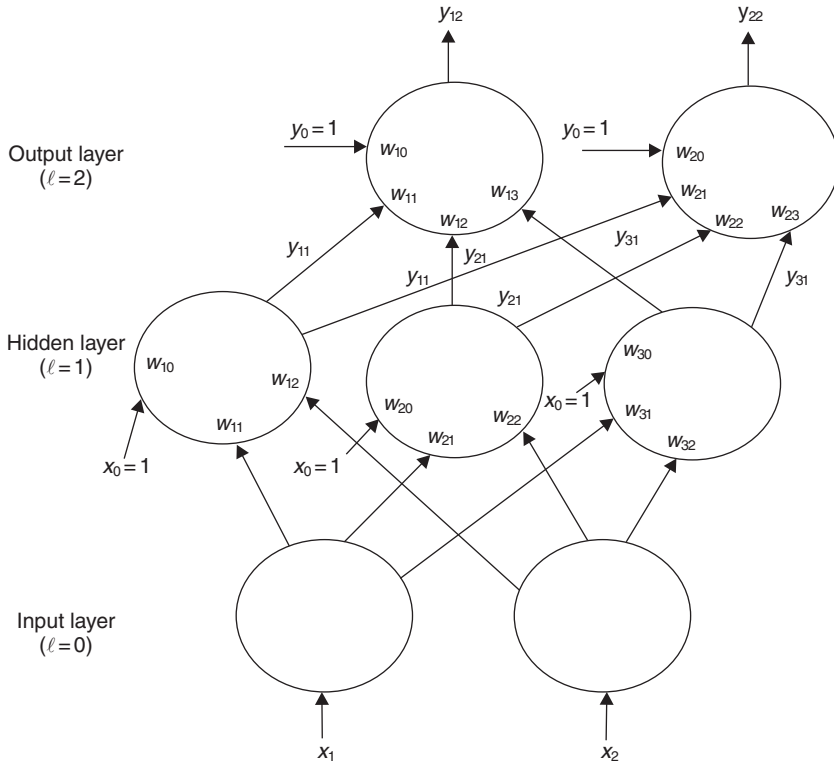


Fig. 10.43 Neural network structure for Example 10.9.

Hidden layer

$$\mathbf{W}_{j1} = \begin{bmatrix} 1 & 2 \\ 3 & 3 \\ 2 & 1 \end{bmatrix} \quad \mathbf{b}_{j1} = \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix}$$

Output layer

$$\mathbf{W}_{j2} = \begin{bmatrix} -3 & -2 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad \mathbf{b}_{j2} = \begin{bmatrix} 3 \\ -3 \end{bmatrix}$$

Solution

- (a) $y_{12} = 0.450, y_{22} = 0.862$
 (b) $y_{12} = 0.405, y_{22} = 0.906$

Example 10.10

In Example 10.9(b), if the target values for the outputs are $d_{12} = 0$ and $d_{22} = 1$, calculate new values for the weights and biases using the back-propagation algorithm. Assume a learning rate of 0.5 with no momentum term.

Solution

Output layer: $\delta_1 = -0.098$, $\delta_2 = 0.008$ (Equation 10.78)

$$\mathbf{W}_{j2} = \begin{bmatrix} -3.047 & -2.036 & 0.954 \\ 1.004 & 2.003 & 3.004 \end{bmatrix} \quad \mathbf{b}_{j2} = \begin{bmatrix} 2.951 \\ -2.996 \end{bmatrix}$$

Hidden layer: $\delta_1 = 0.0016$, $\delta_2 = 0.042$, $\delta_3 = -0.003$ (Equation 10.80)

$$\mathbf{W}_{j1} = \begin{bmatrix} 1.0007 & 2.00008 \\ 3.019 & 3.0021 \\ 1.999 & 0.9999 \end{bmatrix} \quad \mathbf{b}_{j1} = \begin{bmatrix} 2.0008 \\ -1.979 \\ 0.9985 \end{bmatrix}$$

Example 10.11

A system has a parabolic fitness function

$$J = -x^2 + 2x \quad (10.108)$$

Using a genetic algorithm, find the value of x that maximizes J in equation (10.108). Assume that the solution space has 31 values in the range $x = 0$ to 2 and that each solution can be represented by a five digit binary string ranging from 00000 to 11111. Let the population have four members and it is initially seeded by spinning a coin (heads = 1, tails = 0).

Solution

$x(2)$	01110	01111	10000	10001
$x(10)$	14	15	16	17
x	0.903	0.968	1.032	1.097
J	0.990591	0.998976	0.998976	0.990591

Hence $x = 0.968$ and 1.032 both give optimum solutions.

Example 10.12

The closed-loop control system analysed by the root-locus method in Example 5.8 can be represented by the block diagram shown in Figure 10.44. Using root-locus, the best setting for K_1 was found to be 11.35, representing a damping ratio of 0.5.

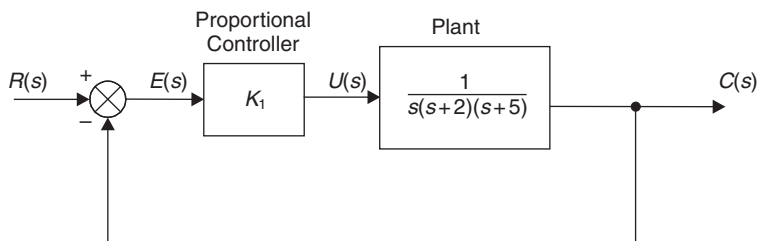


Fig. 10.44 Block diagram for Example 5.8.

Validate, or otherwise, using a genetic algorithm that this value of K_1 maximizes the fitness function

$$J = 100 \left/ \sum_{k=0}^{N-1} \{ (r(kT) - c(kT))^2 T \} \right. \quad (10.109)$$

when the system is subjected to a unit step at time $kT = 0$. Using a sampling time $T = 0.05$ seconds, perform the summation over a time period of 10 seconds ($N = 200$). Allow a search space of 0–100 for K_1 . Assume that the solution space has 255 values and that each solution can be represented by an eight digit binary string chromosome. Use a randomly seeded initial population of 10 members.

If the state variables are

$$\begin{aligned} x_1 &= c(t) \\ x_2 &= \dot{c}(t) = \dot{x}_1 \\ x_3 &= \ddot{c}(t) = \dot{x}_2 \end{aligned}$$

From Figure 10.44, the plant differential equation may be written as

$$\dot{x}_3 = 0x_1 - 10x_2 - 7x_3 + u$$

The plant state and output equations are then

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -10 & -7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$C = [1 \quad 0 \quad 0] \mathbf{x} \quad (10.110)$$

For a sampling time of 0.05 seconds, the discrete form of equation (10.110) is

$$\begin{bmatrix} x_1(k+1)T \\ x_2(k+1)T \\ x_3(k+1)T \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0498 & 0.0011 \\ 0.0 & 0.9889 & 0.0420 \\ 0.0 & -0.4201 & 0.6948 \end{bmatrix} \begin{bmatrix} x_1(kT) \\ x_2(kT) \\ x_3(kT) \end{bmatrix} + \begin{bmatrix} 0.00002 \\ 0.00111 \\ 0.04201 \end{bmatrix} u(kT) \quad (10.111)$$

Solution

$K_1(2)$	00011101	00101001	00110011	00111000	10000001	10110011
$K_1(10)$	29	41	51	56	129	179
K_1	11.37	16.08	20.0	21.96	50.59	70.20
J	112.99	124.51	126.63	125.82	48.18	7.77

Thus the value of K_1 that maximizes the fitness function in equation (10.109) is $K_1 = 20.0$. This produces an overshoot of 38% when $kT = 2.0$ seconds, and represents an ω_n of 1.847 rad/s and a ζ of 0.303.

Appendix 1

Control system design using MATLAB

A1.1 Introduction

MATLAB, its Toolboxes and SIMULINK have become, over a number of years, the industry standard software package for control system design. The purpose of this Appendix is to introduce the reader to some of the more useful aspects of MATLAB, and to illustrate how the software may be used to solve examples given in the main text of the book.

A1.1.1 Getting started

The examples given in this Appendix were generated using MATLAB Version 5.3. Once the software has been installed, MATLAB is most easily entered by clicking the MATLAB icon. Alternatively, in a Windows environment, MATLAB can be entered by clicking the following sequence

Start → Programs → MATLAB for Windows → MATLAB 5.3

The user should now be in the MATLAB command window, which contains some helpful comments together with the MATLAB prompt `»`. MATLAB commands are typed after the prompt, and entered using 'Return' (or 'Enter'). Terminating the command with `;` will suppress the result of the command being printed in the command window. Comments are preceded by the `'%'` symbol.

A1.2 Tutorial 1: Matrix operations

This tutorial introduces the reader to matrix operations using MATLAB. All text in courier font is either typed into, or printed into the command window.

```
» % Matrix Operations
» % To enter a matrix
» A=[1 3;5 9];
» B=[4 -7;10 0];
» A
```

```

A=
    1    3
    5    9
»B
B=
    4   -7
   10    0
»% Note that MATLAB is case sensitive
»% Matrix Addition
»A+B % Terminating with ';' will suppress 'ans'
ans=
    5   -4
   15    9
»% Matrix Multiplication
»A*B
ans=
   34   -7
  110  -35
»% Determinant of a Matrix
»det(A)
ans=
   -6
»% Inverse of a Matrix
»inv(A)
ans=
   -1.5000    0.5000
    0.8333   -0.1667
»% Check
»C=inv(A);
»A*C
ans=
    1.0000    0.0000
    0.0000    1.0000
»% Solve  $A^{-1}B$ 
»A\B
ans=
   -1.0000   10.5000
    1.6667   -5.8333
»% Solve  $A*B^{-1}$ 
»A/B
ans=
   -0.4286    0.2714
   -1.2857    1.0143
»% Eigenvalues of a Matrix
»eig(A)
ans=
   -0.5678
   10.5678
»% Coefficients of Characteristic Equation ( $as^2+bs+c$ )
»poly(A)
ans=
    1.0000  -10.0000   -6.0000
»% Roots of Characteristic Equation ( $as^2+bs+c=0$ )
»ce=[1 -10 -6];
»roots(ce)

```

```

ans=
    10.5678
   -0.5678
» % Note that roots(ce) and eig(A) give the same result
» % Transpose of a Matrix
» A'
ans=
     1     5
     3     9
» % Rank of a Matrix
» rank(A)
ans=
     2
» % Create an Identity Matrix
» I=eye(3);
» I
I=
     1     0     0
     0     1     0
     0     0     1
» % Condition of a Matrix
» % The higher the Condition Number, the more ill-conditioned the
» % matrix is
» % Log10 of Condition Number gives approx. number of decimal
» % places
» % lost due to round-off errors
» cond(A)
ans=
    19.2815
» % Tutorial End

```

The above session may be printed by clicking

File → Print

MATLAB may be closed by clicking

File → Exit MATLAB

A1.3 Tutorial 2: Time domain analysis

This tutorial introduces the reader to time domain analysis using MATLAB. It uses commands from the Control System Toolbox. A list of the commands can be found using

```
» help control
```

More information on individual commands can be obtained, for example

```
» help step
```

will provide more detail on how to use the step command.

Script files: A script file is an ASCII text file of MATLAB commands, that can be created using

- (a) a text editor
- (b) the MATLAB editor/debugger
- (c) a word processor that can save as pure ASCII text files.

A script file should have a name that ends in '.m', and is run by typing the name of the file (without '.m') after the MATLAB prompt, or by typing the sequence

File → Run Script → enter file name

The advantage of a script file is that it only needs to be created once and saves the labour of continually typing lists of commands at the MATLAB prompt.

The examples given in this tutorial relate to those solved in Chapter 3. Consider a first-order transfer function

$$G(s) = 1/1 + s$$

The impulse response function (Example 3.4, Figure 3.11) can be created by the following script file

File name: examp34.m

```
%impulse response of transfer function G(s)=num(s)/den(s)
%num and den contain polynomial coefficients
%in descending powers of s
clf
num=[1];
den=[1 1];
impz(num,den);
grid;
printsys(num,den,'s');
```

This shows how a transfer function is entered into MATLAB, where num=[1] is the numerator ($K=1$) and den=[1 1] represents the 's' coefficient and the 's⁰' coefficient respectively. 'Impz (num, den)' computes and plots the impulse response and grid produces a rectangular grid on the plot. Printsys (num, den, 's') prints the transfer function at the MATLAB prompt. A hard copy can be obtained by selecting, from the screen plot

File → Print



The step response of a first-order system (Example 3.5, Figure 3.13) is obtained using the step command




File name: examp35.m

```
% step response of transfer function G(s)=num(s)/den(s)
%num and den contain polynomial coefficients
%in descending powers of s
clf
num=[1];
den=[1 1];
step(num,den);
grid;
printsys(num,den,'s');
```

SIMULINK: The Control System Toolbox does not possess a ‘ramp’ command, but the ramp response of a first-order system (Example 3.6, Figure 3.15) can be obtained using SIMULINK, which is an easy to use Graphical User Interface (GUI). SIMULINK allows a block diagram representation of a control system to be constructed and real-time simulations performed.

With MATLAB Version 5.3, typing `simulink` at the MATLAB prompt brings up the SIMULINK Library Browser. Clicking on the ‘Create new model’ icon in the top left-hand corner creates a new window called ‘untitled’.

Clicking on the  icon attached to SIMULINK lists the SIMULINK options, and  Continuous lists the continuous systems options. To obtain a transfer function block, click and hold left mouse button on the ‘Transfer Fcn’ icon under ‘Continuous’, and drag to ‘untitled’ window.

Click on  to close down ‘Continuous’ and click on  ‘Sources’ to drag ‘Ramp’ from Browser to ‘untitled’ window. Close down ‘Sources’ and click on  ‘Sinks’ to drag ‘Scope’ to ‘untitled’. Holding down left mouse button connect ‘Ramp’, ‘Transfer Fcn’ and ‘Scope’ together as shown in Figure A1.1.

Double click on ‘Scope’ to bring up scope screen, and, in ‘untitled’, click ‘Simulation’ and ‘Start’. The ramp response should appear on the scope screen. Click on the scope screen and choose autoscale (binoculars). Click print icon to obtain a hard copy of the ramp response. In the ‘untitled’ window, click on ‘File’ and ‘Save As’ and save as a ‘.mdl’ (model) file in a directory of your choice (i.e. *examp36.mdl* in ‘work’).

The transfer function for a second-order system can easily be obtained in terms of ω_n and ζ using the `ord2` command. This has ω_n and ζ as input arguments and generates the numerator and denominator of the equivalent transfer function. The script file *sec_ord.m* shows how Figure 3.19 can be generated using the `ord2` and step MATLAB commands

File name: sec_ord.m

```
%Second-order system
t=[0:0.1:15];
wn=1;
zeta=0.2;
[num,den]=ord2(wn,zeta);
[y,x,t]=step(num,den,t);
zeta=0.4;
[num1,den1]=ord2(wn,zeta);
[y1,x,t]=step(num1,den1,t);
zeta=0.6;
[num2,den2]=ord2(wn,zeta);
[y2,x,t]=step(num2,den2,t);
plot(t,y,t,y1,t,y2);
grid;
```

In *sec_ord.m* a user supplied common time base ($t = 0$ –15 seconds in 0.1 second intervals) has been set up. The `plot` command superimposes the step responses for $\zeta = 0.2, 0.4$ and 0.6 .

The step response for Example 3.8, the resistance thermometer and valve, shown in Figure 3.23 can be generated with script file *examp38.m*.

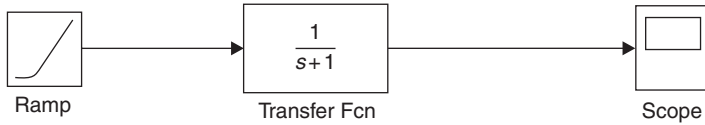


Fig. A1.1 SIMULINK model for the ramp response of a first-order system.

File name: examp38.m

```
%Step response of a third-order system
%G(s)=12.5/(s+0.5)(s^2+s+25)
clf
num=[12.5];
den=conv([1 0.5],[1 1 25]);
step(num,den);
grid;
```

The transfer function is based on equation (3.75), with the input term $X_i(s) = 1/s$ removed. The denominator is input using the `conv` command (short for convolution), which multiplies two polynomial expressions together.

A1.4 Tutorial 3: Closed-loop control systems

This tutorial shows how MATLAB can be used to build up and test closed-loop control systems. The examples given relate to those solved in Chapter 4. The commands used in MATLAB to create a single model from the elements in the control loop are

Command	Operation
<code>series</code>	Combines blocks in cascade, see Transformation 1, Table 4.1
<code>parallel</code>	Combines blocks in parallel, see Transformation 2, Table 4.1
<code>feedback</code>	Eliminates a feedback loop, see Transformation 4, Table 4.1
<code>cloop</code>	Closes the loop with unity feedback.

Example 4.5 is a PI controlled liquid-level system shown in Figure 4.26. In the block diagram representation, Figure 4.27, the system parameters are

$$K_1 = 1; \quad T_i = 5 \text{ seconds}; \quad K_v = 0.1 \text{ m}^3/\text{sV}$$

$$R_f = 15 \text{ s/m}^2; \quad A = 2 \text{ m}^2; \quad H_1 = 1$$

The PI controller and control valve transfer function is therefore

$$G_c(s) = \frac{0.5s + 0.1}{5s}$$

and the plant transfer function is

$$G_p(s) = \frac{15}{30s + 1}$$

Since $H_1 = 1$, the system has unity feedback, and the closed-loop transfer function and step response is given by

Filename: examp45.m

```
%Example 4.5 (Liquid-Level Process Control System)
%Use of series and cloop
numc=[ 0.5  0.1];
denc=[ 5  0];
nump=[15];
denp=[30  1];
[numol,denol]=series(numc,denc,nump,denp);
[numcl,denc1]=cloop(numol,denol);
printsys(numcl,denc1,'s');
step(numcl,denc1);
grid;
```

Case study Example 4.6.1 is a CNC machine-tool positional control system, whose block diagram representation is shown in Figure 4.31. When system parameter values are inserted, the block diagram is as shown in Figure A1.2.

The closed-loop transfer function and step response is given by

Filename: examp461.m

```
%Case Study Example 4.6.1 (CNC Machine-Tool Positional Control)
%Use of series and feedback
nump=[80];
denp=[ 0.45  0];
numpl=[0.365];
denpl=[ 1  0];
numvf=[0.697];
denvf=[1];
numpf=[60];
denpf=[1];
[num,den]=feedback(nump,denp,numvf,denvf);
[numfp,denfp]=series(num,den,numpl,denpl);
[numcl,denc1]=feedback(numfp,denfp,numpf,denpf);
printsys(numcl,denc1,'s');
step(numcl,denc1);
grid;
```

In this example, the inner loop is solved first using feedback. The controller and integrator are cascaded together (numpl,denpl) and then series is used to find the forward-path transfer function (numfp,denfp). Feedback is then used again to obtain the closed-loop transfer function.

The time response of the CNC control system is also obtained using SIMULINK as shown in Figure A1.3. Note that the variables t and x_0 have been sent to workspace, an area of memory that holds and saves variables. The commands who and whos lists the variables in the workspace. The system time response can be obtained by the command

```
plot(t,x0)
```

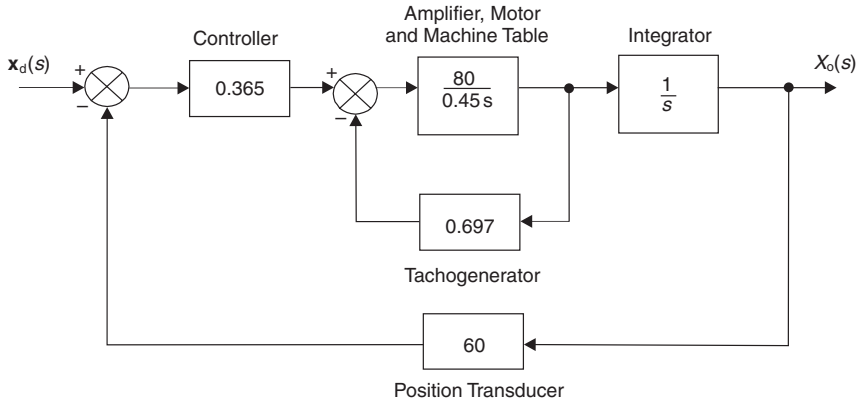


Fig. A1.2 CNC machine-tool positional control system.

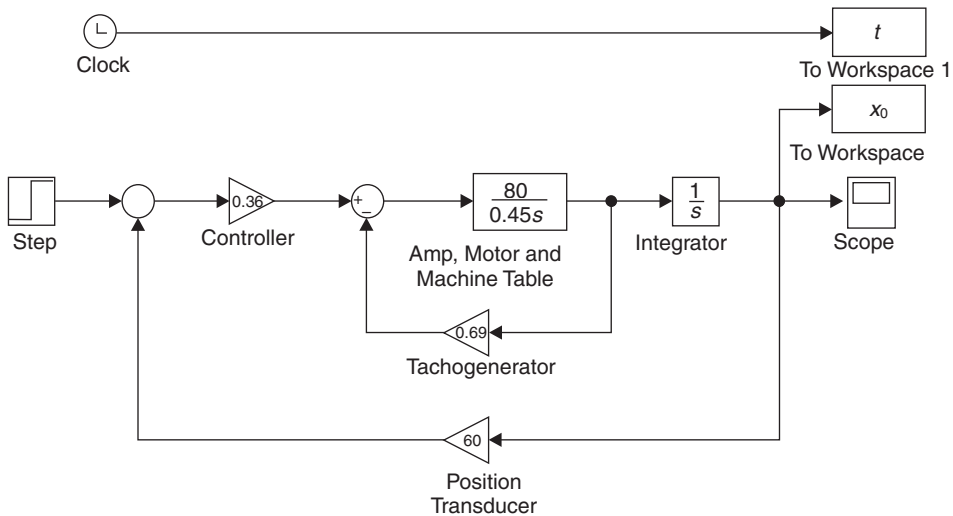


Fig. A1.3 SIMULINK simulation of CNC machine-tool positional control system.

Case study Example 4.6.2 is a PID temperature control system, and is represented by the block diagram in Figure 4.33. Here, the PID control transfer function is not ‘proper’, and the `series` command does not work. The forward path transfer function, with values inserted, is

$$G_{fp}(s) = \frac{29.412s^2 + 39.216s + 13.072}{3s(4s + 1)(8s + 1)}$$

The closed-loop transfer function and step response is given by

Filename: *examp462.m*

```
%Case study example 4.6.2 (Temperature Control)
%Use of feedback
```

```

numfp=[ 29.412  39.216  13.072 ];
denfp=conv([ 3  0],conv([ 4  1],[ 8  1]));
numtf=[1];
dentf=[1];
[numcl,denc1]=feedback(numfp,denfp,numtf,dentf);
printsys(numcl,denc1,'s');
step(numcl,denc1);
grid;

```

Case study Example 4.6.3 is a ship autopilot control system. The block diagram representation is given by Figure 4.37. Inserting system parameters gives a forward-path transfer function

$$G_{fp}(s) = \frac{0.2}{20s^2 + 2s}$$

The closed-loop transfer function and step response is given by

Filename: examp463.m

```

%Case study example 4.6.3 (Ship Autopilot Control System)
%Use of feedback
numfp=[0.2];
denfp=[ 20  2  0];
numhf=[1];
denhf=[1];
[numcl,denc1]=feedback(numfp,denfp,numhf,denhf);
printsys(numcl,denc1,'s');
step(numcl,denc1);
grid;

```

A1.5 Tutorial 4: Classical design in the s-plane

The tutorial demonstrates how MATLAB is used to generate root locus diagrams, and hence how to design control systems in the s -plane. Examples given in Chapter 5 are used to illustrate the MATLAB commands. The roots of the characteristic equation (or any polynomial) can be found using the `roots` command.

Example 5.1

Check the stability of the system that has the characteristic equation

$$s^4 + 2s^3 + s^2 + 4s + 2 = 0$$

At the MATLAB prompt type

```

»ce=[ 1  2  1  4  2 ];
»roots(ce)
ans=
-2.1877
 0.3516+1.2843i
 0.3516-1.2843i
-0.5156

```

Two roots have positive real parts, hence the system is unstable.

Example 5.2

What values of K_1 make the following system unstable

$$G(s)H(s) = \frac{8K_1}{s(s^2 + s + 2)}$$

Filename: examp52.m

```
%Roots of denominator of closed-loop transfer function
k1=0.25
num=[8*k1];
den=conv([1 0],[1 1 2]);
[numcl,denc1]=cloop(num,den);
roots(denc1)
```

Running *examp52.m* with K_1 set to 0.25, 0.15 and 0.35 gives

```
» examp52
k1=
    0.2500

ans=
    0.0000+1.4142i
    0.0000-1.4142i
   -1.0000

» examp52
k1=
    0.1500

ans=
   -0.1630+1.3243i
   -0.1630-1.3243i
   -0.6740

» examp52
k1=
    0.3500

ans=
    0.1140+1.5057i
    0.1140-1.5057i
   -1.2280
```

Hence, with $K_1 = 0.15$, the system is stable, $K_1 = 0.25$, the system has marginal stability and $K_1 = 0.35$, the system is unstable.

Example 5.8

The root locus for

$$G(s)H(s) = \frac{K}{s(s+2)(s+5)}$$

can be drawn using the `rlocus` command as shown in *examp58.m*

Filename: examp58.m

```
%Example 5.8 Simple root locus
%G(s)=K/s(s+2)(s+5)
clf
num=[1];
den=conv(conv([1 0],[1 2]),[1 5]);
rlocus(num,den);
axis([-9 2 -4 4]);
%axis('square');
```

With no axis command, the axes are generated automatically. Alternatively, user defined axes can be generated with `axis ([xmin xmax ymin ymax])`. Using the command 'square' generates the same scales on both x and y axes. With *examp58.m* it is easy to experiment with all three methods.

Using MATLAB to design a system, it is possible to superimpose lines of constant ζ and ω_n on the root locus diagram. It is also possible, using a cursor in the graphics window, to select a point on the locus, and return values for open-loop gain K and closed-loop poles using the command

```
[k,poles]=relocfind(num,den)
```

The script file *examp58a.m* shows how this is achieved

Filename: examp58a.m

```
%Example 5.8 Root locus
%G(s)=K/s(s+2)(s+5)
clf
num=[1];
den=conv(conv([1 0],[1 2]),[1 5]);
rlocus(num,den);
axis([-9 2 -4 4]);
zeta=0.5;
wn=2:2:8;
sgrid(zeta,wn);
[k,poles]=relocfind(num,den)
```

Here a line of $\zeta = 0.5$ ($\beta = 60^\circ$) is drawn together with four ω_n circles ($\omega_n = 2, 4, 6$ and 8 rad/s.) At the MATLAB prompt, the user is asked to select a point in the graphics window. If the intersection of the complex locus with the $\zeta = 0.5$ line is selected (see Figure 5.14), the following response is obtained

```
» examp58a
Select a point in the graphics window

selected point=
-0.7143+1.2541i
k=
11.5754
poles=
-5.5796
-0.7102+1.2531i
-0.7102-1.2531i
```

Example 5.9 is solved using *examp59.m* to create Figure 5.16.

Filename : *examp59.m*

```
%Example 5.9 Root Locus
%G(s)=K/s(s^2+4s+13)
clf
num=[1];
den=conv([1 0],[1 4 13]);
rlocus(num,den);
axis([-9 2 -4 4]);
zeta=0.25;
wn=3;
sgrid(zeta,wn);
k=rlocfind(num,den) %Select k from graphic window
```

When run, the program invites the user to select a point in the graphics window, which may be used to find the value of K when $\zeta = 0.25$. If the last line of *examp59.m* is typed at the MATLAB prompt, the cursor re-appears, and a further selection can be made, in this case to select the value of K for marginal stability. This is demonstrated below

```
» examp59

Select a point in the graphics window
selected_point=
    -0.7429+2.9175i
k=
    22.9452
» k=rlocfind(num,den)
Select a point in the graphics window
selected_point=
    0+3.6304i
k=
    52.7222
»
```

Example 5.10 uses the root locus method to design a PD controller that will allow the system

$$G(s)H(s) = \frac{1}{s(s+2)(s+5)}$$

To meet a given time-domain specification. In *examp510.m*, the PD controller takes the form

$$G_c(s) = K_1(s+2)$$

Filename: *examp510.m*

```
%Example 5.10 Root Locus
%G(s)=K(s+2)/s(s+2)(s+5)
clf
num=[1 2];
den=conv(conv([1 0],[1 2]),[1 5]);
rlocus(num,den);
axis([-9 2 -4 4]);
zeta=0.7;
wn=3.5;
sgrid(zeta,wn);
k=rlocfind(num,den) %Select k from screen
```

```

num=[k k*2];
[numcl,denc1]=cloop(num,den);
step(numcl,denc1);
grid;

```

This produces the (pole-zero cancellation) root locus plot shown in Figure 5.18. When run, *examp510.m* allows the user to select the value of K_1 that corresponds to $\zeta = 0.7$, and then uses this selected value to plot the step response. The text that appears in the command window is

```

» examp510
Select a point in the graphics window
selected_point=
    -2.4857+2.5215i
k=
    12.6077
»

```

Case study Example 5.11 uses root locus to design a ship roll stabilization system. The script file *examp511.m* considers a combined PD and PID (PIDD) controller of the form

$$G_c(s) = \frac{K_1(s+2)(s^2+4s+8)}{s}$$

Filename: examp511.m

```

%Example 5.11 Root Locus
%G(s)=K(s+2)(s^2+4s+8)/s(s+1)(s^2+0.7s+2)
clf
num=conv([1 2],[1 4 8]);
den=conv(conv([1 0],[1 1]),[1 0.7 2]);
rlocus(num,den);
axis([-9 2 -4 4]);
zeta=0.7;
wn=4.2;
sgrid(zeta,wn);
k=rlocfind(num,den) % Select k from screen
num=k*conv([1 2],[1 4 8]);
[numcl,denc1]=cloop(num,den);
step(numcl,denc1);
grid;

```

This script file produces the root locus shown in Figure 5.24 and allows the user to select the value of K furthest from the imaginary axis that corresponds to $\zeta = 0.7$. The command window text is

```

» examp511
Select a point in the graphics window
selected_point=
    -3.2000+3.2607i
k=
    10.2416
»

```

When K has been selected, the roll angle step response, as shown in Figure 5.25 is plotted.

A1.6 Tutorial 5: Classical design in the frequency domain

This tutorial shows how MATLAB can be used to construct all the classical frequency domain plots, i.e. Bode gain and phase diagrams, Nyquist diagrams and Nichols charts. Control system design problems from Chapter 6 are used as examples.

Example 6.1

Construct the Bode diagram for

$$G(s) = \frac{2}{1 + 0.5s}$$

Filename: *examp61.m*

```
%Example 6.1 Bode Diagram
%First-order system
num=[2];
den=[0.5 1];
bode(num,den);
```

The Nyquist diagram for the same system is

Filename: *examp61a.m*

```
%Example 6.1(a) Nyquist Diagram
%First-order system
num=[2];
den=[0.5 1];
nyquist(num,den);
```

The script file *examp61b.m* shows how it is possible to customize a Bode diagram

Filename: *examp61b.m*

```
%Example 6.1(b)
%Customizing a Bode Diagram
clf
num=[2];
den=[0.5 1];
w=logspace(-1,2,200); %w from 10^-1 to 10^2, 200 points
[mag,phase,w]=bode(num,den,w) % cal mag and phase
semilogx(w,20*log10(mag)),grid;
xlabel('Frequency (rad/s)'),ylabel('Gain dB');
```

The `logspace` command allows a vector of frequencies to be specified by the user, in this case, 200 points between 10^{-1} and 10^2 (rad/s). The `bode` command has a left-hand argument `[mag,phase,w]` which allows the magnitude and phase to be calculated, but not displayed. The `semilogx` command plots the magnitude (converted to decibels) against log frequency, and `grid` provides a log-linear grid. Script file *examp62.m* produces a second-order Bode gain diagram for Example 6.2.

Filename: *examp62.m*

```
%Example 6.2 Bode Diagram
%Second-order system
clf
num=[4];
den=[0.25 0.2 1];
w=logspace(-1,2,200);
[mag,phase,w]=bode(num,den,w);
semilogx(w,20*log10(mag)),grid;
xlabel('Frequency (rad/s)'),ylabel('Gain dB')
```

The Nyquist diagram (ω varying from $-\infty$ to $+\infty$) is produced by *examp64.m* where

$$G(s)H(s) = \frac{K}{s(s^2 + 2s + 4)}$$

Filename: *examp64.m*

```
%Example 6.4 Nyquist Diagram
%Third-order type one system
num=[1];
den=conv([1 0],[1 2 4]);
nyquist(num,den);
axis([-0.6 0.1 -0.6 0.1]);
```

Note that to obtain a reasonable diagram, it is usually necessary for the user to define the scales of the x and y axes using the axis command. The script file *examp64a.m* produces the Bode gain diagrams for the same system when $K = 4$ and 8, see Figure 6.23(a).

Filename: *examp64a.m*

```
%Example 6.4(a)
G(s)=K/s(s^2+2s+4)
%Creates Bode Gain Diagrams for K=4 and 8
clf
num=[4];
den=conv([1 0],[1 2 4]);
w=logspace(-1,1,200);
[mag,phase,w]=bode(num,den,w);
semilogx(w,20*log10(mag));
num=[8];
[mag1,phase1,w]=bode(num,den,w);
semilogx(w,20*log10(mag),w,20*log10(mag1)),grid;
xlabel('Frequency (rad/s)'),ylabel('Gain (dB)')
```

Using the command `margin(mag,phase,w)` gives

- Bode gain and phase diagrams showing, as a vertical line, the gain and phase margins.
- a print-out above the plots of the gain and phase margins, and their respective frequencies.

This is illustrated by running *examp64b.m*, which confirms the values given in Figure 6.23, when $K = 4$.

Filename: *examp64b.m*

```
%Example 6.4(b)
%G(s)=K/s(s^2+2s+4)
%Creates a Bode Gain and Phase Diagrams for K=4
%Determines Gain and Phase Margins
clf
num=[4];
den=conv([1 0],[1 2 4]);
w=logspace(-1,1,200);
[mag,phase,w]=bode(num,den,w);
margin(mag,phase,w);
xlabel('Frequency(rad/s)');
```

Script file *fig627.m* produces the Nichols chart for Example 6.4 when $K = 4$, as illustrated in Figure 6.27. The command *ngrid* produces the closed-loop magnitude and phase contours and axis provides user-defined axes. Some versions of MATLAB appear to have problems with the *nichols* command.

Filename: *fig627.m*

```
%Example 6.4 as displayed in Figure 6.27
%G(s)=K/s(s^2+2s+4) where K=4
%Nichols Chart
num=[4];
den=conv([1 0],[1 2 4]);
nichols(num,den);
ngrid;
axis([-210 0 -30 40]);
```

Running script file *fig629.m* will produce the closed-loop frequency response gain diagrams shown in Figure 6.29 for Example 6.4 when $K = 3.8$ and 3.2 (value of K for best flatband response).

Filename: *fig629.m*

```
%Example 6.4 as displayed in Figure 6.29
%G(s)=K/s(s^2+2s+4)
%Creates closed-loop Bode Gain Diagrams for K=3.8 and 3.2
%Prints in Command Window Mp,k,wp and bandwidth
clf
num=[3.8];
den=conv([1 0],[1 2 4]);
w=logspace(-1,1,200);
[numcl,denc1]=cloop(num,den);
[mag,phase,w]=bode(numcl,denc1,w);
[Mp,k]=max(20*log10(mag))
wp=w(k)
n=1;
while 20*log10(mag(n))>=-3,n=n+1;end
bandwidth=w(n)
num=[3.2];
[num2cl,den2cl]=cloop(num,den);
[mag1,phase1,w]=bode(num2cl,den2cl,w);
semilogx(w,20*log10(mag),w,20*log10(mag1)),grid;
xlabel('Frequency(rad/s)'),ylabel('Gain)dB)')
```

The command `cloop` is used to find the closed-loop transfer function. The command `max` is used to find the maximum value of $20 \log_{10}(\text{mag})$, i.e. M_p and the frequency at which it occurs i.e. $\omega_p = \omega(k)$. A `while` loop is used to find the -3 dB point and hence `bandwidth` $= \omega(n)$. Thus, in addition to plotting the closed-loop frequency response gain diagrams, *fig629.m* will print in the command window:

```
»
Mp=
    3.1124
k=
    121
wp=
    1.6071
bandwidth=
    2.1711
```

Case study

Example 6.6

This is a laser guided missile with dynamics

$$G(s)H(s) = \frac{20}{s^2(s+5)}$$

The missile is to have a series compensator (design one) of the form

$$G(s) = \frac{K(1+s)}{(1+0.25s)}$$

Figure 6.34 is generated using *fig634.m* and shows the Nichols Chart for the uncompensated system. Curve (a) is when the compensator gain $K = 1$, and curve (b) is when $K = 0.537$ (a gain reduction of 5.4 dB).

Filename: *fig634.m*

```
%Nichols Chart for Case Study Example 6.6
%Lead Compensator Design One, Figure 6.34
clf
%Uncompensated System
num=[20];
den=[1 5 0 0];
w=logspace(-1,1,200);
[mag,phase,w]=nichols(num,den,w);
%Compensator Gain set to unity
num1=[20 20];
den1=conv([0.25 1],[1 5 0 0]);
[mag1,phase1,w]=nichols(num1,den1,w);
%Compensator Gain reduced by 5.4dB
num2=[10.74 10.74];
den2=conv([0.25 1],[1 5 0 0]);
[mag2,phase2,w]=nichols(num2,den2,w);
plot(phase,20*log10(mag),phase1,20*log10(mag1),
phase2,20*log10(mag2))
grid;
axis([-240 -60 -20 30]);
```

A1.7 Tutorial 6: Digital control system design

This tutorial looks at the application of MATLAB to digital control system design, using the problems in Chapter 7 as design examples.

Example 7.3

To obtain the z -transform of a first-order sampled data system in cascade with a zero-order hold (zoh), as shown in Figure 7.10.

Filename: *examp73.m*

```
%Example 7.3 Transfer Function to z-Transform
%Continuous and Discrete Step Response
num=[1];
den=[1 1];
Ts=0.5;
[numd,dend]=c2dm(num,den,Ts,'zoh');
printsys(num,den,'s')
printsys(numd,dend,'z')
subplot(121), step(num,den);
subplot(122), dstep(numd,dend);
```

The continuous to discrete command `c2dm` employs a number of conversion methods in the last term of the right-hand argument. These include

'zoh' zero-order hold
'foh' first-order hold
'tustin' tustin's rule (see equation 7.102)

The command `subplot (mnp)` creates an m -by- n array of equal sized graphs, the argument p indicates the current graph. Thus `subplot (12p)` produces a single row, two column array (i.e. side by side graphs). When $p = 1$, the left-hand graph is produced and when $p = 2$, the right-hand one is produced. Running *examp73.m* generates step response plots of both continuous and discrete systems, with the following text in the command window

```
»
num/den=
    1
   s+1
num/den=
    0.39347
   z-0.60653
```

Example 7.4

This is a closed-loop digital control system as shown in Figure 7.14, with a plant transfer function

$$G_p(s) = \frac{1}{s(s+2)}$$

Filename: *examp74.m*

```
%Example 7.4 Open and Closed-Loop Pulse Transfer Functions
%Discrete Step Response
```

```

num=[1];
den=conv([1 0],[1 2]);
Ts=0.5;
[numd,dend]=c2dm(num,den,Ts,'zoh');
[numcld,dencld]=cloop(numd,dend);
printsys(num,den,'s')
printsys(numd,dend,'z')
printsys(numcld,dencld,'z')
dstep(numcld,dencld);

```

Note that the command `cloop` works with both continuous and discrete systems. Running *examp74.m* will produce a step response plot of the closed-loop discrete system, and the open and closed-loop pulse transfer functions will be written in the command window

```

»
num/den=
      1
  s^2+2s
num/den=
      0.09197z+0.06606
  z^2-1.3679z+0.36788    %see equation (7.53)
num/den=
      0.09197z+0.06606
  z^2-1.2759z+0.43394    %see equation (7.55)

```

The script file *examp75.m* simulates the Jury stability test undertaken in Example 7.5. With the controller gain K in Example 7.5 (Figure 7.14) set to 9.58 for marginal stability see equation (7.75), the roots of the denominator of the closed-loop pulse transfer function are calculated, and found to lie on the unit circle in the z -plane.

Filename: examp75.m

```

%Example 7.5 Stability in the z-plane
K=9.58
num=[K];
den=conv([1 0],[1 2]);
Ts=0.5;
[numd,dend]=c2dm(num,den,Ts,'zoh');
[numcld,dencld]=cloop(numd,dend);
printsys(numcld,dencld,'z')
roots(dencld)

```

Running *examp75.m* produces command window text

```

»
K=
    9.5800
num/den=
      0.88107z+0.63286
  z^2-0.48681z+1.0007
ans=
    0.2434+0.9703i    %lies on unit circle i.e.  $\sqrt{R^2+I^2}=1$ 
    0.2434-0.9703i    %see equation (7.87)

```

Example 7.6 constructs the root-locus in the z -plane for the digital control system in Example 7.4 (Figure 7.14).

Filename: examp76.m

```
%Example 7.6 Root Locus Analysis in the z-plane
num=[1];
den=conv([1 0],[1 2]);
Ts=0.5;
[numd,dend]=c2dm(num,den,Ts,'zoh');
rlocus(numd,dend);
axis('square')
zgrid;
k=rlocfind(numd,dend)
```

Note that `rlocus` and `rlocfind` works for both continuous and discrete systems. The statement `'square'` provides square axes and so provides a round unit circle. The command `zgrid` creates a unit circle together with contours of constant natural frequency and damping, within the unit circle. When *examp76.m* has been run, using `rlocfind` at the MATLAB prompt allows points on the loci to be selected and values of K identified (see Figure 7.20)

```
»Select a point in the graphics window
selected_point=
    0.2212+0.9591i
k=
    9.7078
»k=rlocfind(numd,dend)
Select a point in the graphics window
selected_point=
   -2.0876-0.0117i
k=
   60.2004
k=rlocfind(numd,dend)
Select a point in the graphics window
selected_point=
   -1.0092-0.0117i
k=
   103.3290
```

Example 7.7

A laser guided missile has dynamics

$$G(s)H(s) = \frac{20}{s^2(s+5)}$$

and a compensator

$$G_c(s) = \frac{0.8(1+s)}{(1+0.0625s)}$$

Script file *examp77.m* plots the closed-loop step responses of both the continuous system and discrete system (see Figure 7.21). In the latter case the plant pulse transfer function uses `zoh`, and the compensator is converted into discrete form using

Tustin's rule. **Subplot (211)** and **(212)** creates a 2 row, single column plot matrix (i.e. produces 2 plots, one beneath the other).

Filename: examp77.m

```
%Example 7.7 Digital Compensator using Tustin's Rule
%Laser Guided Missile
num=[20];
den=conv([1 0 0],[1 5]);
Ts=0.1;
ncomp=0.8*[1 1];
dcomp=[0.0625 1];
[nol,dol]=series(ncomp,dcomp,num,den);
[ncl,dcl]=cloop(nol,dol);
[numd,dend]=c2dm(num,den,Ts,'zoh');
[ncomd,dcomd]=c2dm(ncomp,dcomp,Ts,'tustin');
printsys(num,den,'s')
printsys(numd,dend,'z')
printsys(ncomp,dcomp,'s')
printsys(ncomd,dcomd,'z')
[nold,dold]=series(ncomd,dcomd,numd,dend);
[nclld,dclld]=cloop(nold,dold);
subplot(211), step(ncl,dcl);
subplot(212), dstep(nclld,dclld);
```

The open continuous transfer functions and pulse transfer functions for the plant and compensator are printed in the command window

```
»
num/den=
      20
  -----
  s^3+5s^2
num/den=
  0.0029551z^2+0.010482z+0.002302
  -----
  z^3-2.6065z^2+2.2131z-0.60653      %see equation (7.108)
num/den=
  0.8s+0.8
  -----
  0.0625s+1
num/den=
  7.4667z-6.7556
  -----
  z-0.11111      %see equation (7.110)
```

Example 7.8

Here pole placement is used to design a digital compensator that produces exactly the step response of the continuous system.

Filename: examp78.m

```
%Example 7.8 Digital Compensator Design using Pole Placement
K=0.336
numk=K*[3];
num=[3];
```

```

den=conv([1 0],[1 1]);
Ts=0.5;
[ncl,dcl]=cloop(numk,den);
ncomd=0.327*[1-0.6065];
dcomd=[1-0.519];
[numd,dend]=c2dm(num,den,Ts,'zoh');
[nold,dold]=series(ncomd,dcomd,numd,dend);
[nclld,dclld]=cloop(nold,dold);
printsys(num,den,'s')
printsys(numd,dend,'z')
printsys(ncomd,dcomd,'z')
printsys(ncl,dcl,'s')
printsys(nclld,dclld,'z')
subplot(211), step(ncl,dcl);
subplot(212), dstep(nclld,dclld);

```

The continuous and discrete closed-loop systems are shown in Figures 7.22(a) and (b). The digital compensator is given in equation (7.128). Script file *examp78.m* produces the step response of both systems (Figure 7.25) and prints the open and closed-loop continuous and pulse transfer functions in the command window

```

»
K=
    0.3360
num/den=
      3
    -----
    s^2+s
num/den=
    0.31959z+0.27061
    -----
    z^2-1.6065z+0.60653           %see equation (7.115)
num/den=
    0.327z-0.19833
    -----
    z-0.519                     %see equation (7.128)
num/den=
    1.008
    -----
    s^2+s+1.008
num/den=
    0.10451z^2+0.025107z-0.053669
    -----
    z^3-2.021z^2+1.4654z-0.36846

```

A1.8 Tutorial 7: State-space methods for control system design

This tutorial looks at how MATLAB commands are used to convert transfer functions into state-space vector matrix representation, and back again. The discrete-time response of a multivariable system is undertaken. Also the controllability and observability of multivariable systems is considered, together with pole placement design techniques for both controllers and observers. The problems in Chapter 8 are used as design examples.

Example 8.4

This converts a transfer function into its state-space representation using `tf2ss(num, den)` and back again using `ss2tf(A,B,C,D,iu)` when `iu` is the `i`th input `u`, normally 1.

Filename: examp84.m

```
%Example 8.4 transfer function to state space representation
%And back again
num=[4];
den=[1 3 6 2];
printsys(num,den,'s')
[A,B,C,D]=tf2ss(num,den)
[num1,den1]=ss2tf(A,B,C,D,1);
printsys(num1,den1,'s')
condition_number=cond(A)
```

The print-out in the command window is

```
» examp84
num/den=
      4
-----
s^3+3s^2+6s+2
A=
-3   -6   -2
 1    0    0
 0    1    0
B=
 1
 0
 0
C=
 0    0    4
D=
 0
num/den=
 4.441e-016s^2+5.329e-015s+4
-----
s^3+3s^2+6s+2
condition_number=
24.9599
```

Comparing the output with equation (8.35) it will be noticed that the top and bottom rows of **A** have been swapped, which means that the state variables x_1 and x_3 have also been exchanged. This means that if the user is expecting the state variables to represent particular parameters (say position, velocity and acceleration), then the rows of **A** and **B**, and the columns of **C** might have to be re-arranged.

The conversion from state-space to transfer function has produced some small erroneous numerator terms, which can be neglected. These errors relate to the condition of **A**, and will increase as the condition number increases.

Example 8.8 uses `c2d` to calculate the discrete-time state and control transition matrices **A**(*T*) and **B**(*T*) as given by equations (8.78), (8.80), (8.82) and (8.85). The

matrix-vector difference equation (8.76) is then used to calculate the first five discrete values of the state variables when responding to a unit step input.

Filename: examp88.m

```
%Calculate state and control transition matrices
A=[ 0  1;-2 -3]
B=[0; 1]
Ts=0.1
[AT,BT]=c2d(A,B,Ts)
%Compute discrete step response
kT=0
x=[0; 0]
u=1
for i=1:5
xnew=AT*x+BT*u;      %Matrix-vector difference equation (8.76)
kT=kT+Ts
x=xnew
end
```

The command window text is

```
»examp88
A=
    0    1
   -2   -3
B=
    0
    1
Ts=
    0.1000
AT=
    0.9909    0.0861
   -0.1722    0.7326
BT=
    0.0045
    0.0861
kT=
    0
x=
    0
    0
u=
    1
kT=
    0.1000
x=
    0.0045
    0.0861
kT=
    0.2000
x=
    0.0164
    0.1484
```

```

kT=
    0.3000
x=
    0.0336
    0.1920
kT=
    0.4000
x=
    0.0543
    0.2210
kT=
    0.5000
x=
    0.0774
    0.2387
»

```

The for-end loop in *examp88.m* that employs equation (8.76), while appearing very simple, is in fact very powerful since it can be used to simulate the time response of any size of multivariable system to any number and manner of inputs. If **A** and **B** are time-varying, then $\mathbf{A}(T)$ and $\mathbf{B}(T)$ should be calculated each time around the loop. The author has used this technique to simulate the time response of a 14 state-variable, 6 input time-varying system. Example 8.10 shows the ease in which the controllability and observability matrices **M** and **N** can be calculated using `ctrb` and `obsv` and their rank checked.

Filename: examp810.m

```

%Example 8.10
%Controllability and observability
A=[-2  0; 3  -5]
B=[1; 0]
C=[1  -1]
D=[0]
M=ctrb(A,B)
rank_of_M=rank(M)
system_order=length(A)
N=(obsv(A,C))'
Rank_of_N=rank(N)

```

The command window will display

```

» examp810
A=
    -2     0
     3    -5
B=
     1
     0
C=
     1    -1
D=
     0

```

```

M=
    1  -2
    0   3
rank_of_M=
    2
system_order=
    2           %rank of M=system order. System completely
                %controllable.

N=
    1  -5
   -1   5
rank_of_N=
    1           %rank of N< system order. System is unobservable.

```

Example 8.11 uses Ackermann's formula (`acker`) to calculate the elements of a regulator feedback matrix **K**, that places the closed-loop poles at some desired location in the *s*-plane.

Filename: examp811.m

```

%Example 8.11
%Regulator design using pole placement
%G(s)=1/s(s+4)
num=1;
den=conv([1 0],[1 4]);
printsys(num,den,'s')
%convert to state space
[A,B,C,D]=tf2ss(num,den);
%Check for controllability;
rank_of_M=rank(ctrb(A,B))
system_order=length(A)
%Enter desired characteristic equation
chareqn=[1 4 4]
%Calculate desired closed-loop poles
desiredpoles=roots(chareqn)
%Calculate feedback gain matrix using Ackermann's formula
K=acker(A,B,desiredpoles)
%Closed-loop state feedback system
Asf=A-B*K; Bs=B; Csf=C; Dsf=0;
[numsf,densf]=ss2tf(Asf,Bsf,Csf,Dsf);
densf
roots(densf)

```

In *examp811.m*, the closed-loop transfer function is obtained and the roots of the denominator calculated to check that the closed-loop poles are at the desired locations. The output at the command window is

```

» examp811
num/den=
    1
   s^2+4s
rank_of_M=
    2
system_order={
    2           % Rank of M = system order hence system
                % is controllable

```

```

chareqn=
    1  4  4
desiredpoles=
   -2
   -2
K=
    0  4
densf=
    1  4  4
ans=
   -2
   -2      % Actual closed-loop poles = desired closed-loop poles

```

Example 8.12 shows how `acker` uses the transpose of the **A** and **C** matrices to design a full-order state observer.

Filename: examp812.m

```

%Example 8.12
%Full-order observer design using pole placement
A=[0 1;-2 -3]
B=[0;1]
C=[1 0]
D=[0]
%Check for observability;
N=obsv(A,C)
rank_or_N=rank(N)
system_order=length(A)
%Enter desired characteristic equation
chareqn=[1 10 100]
%Calculate desired observer eigenvalues
desired_eigenvalues=roots(chareqn)
%Calculate observer gain matrix using Ackermann's formula
Ke=acker(A',C',desired_eigenvalues)

```

The command window text is

```

» examp812
A=
    0    1
   -2   -2
B=
    0
    1
C=
    1    0
D=
    0
N=
    1    0
    0    1
rank_of_N=
    2
system_order=
    2      % Rank of N = system order hence system
           % is observable

```

```

chareqn=
    1 10 100
desired_eigenvalues=
    -5.0000+8.6603i
    -5.0000-8.6603i
Ke=
    7.0000 77.0000      % Agrees with equation (8.150)

```

Example 8.13 illustrates the design of a regulator combined with a reduced-order state observer.

Filename: examp813.m

```

%Example 8.13
%Regulator and reduced-order observer design
%Using pole placement
%G(s)=1/s(s+2)(s+5)
%Input in state-space format directly
A=[0 1 0;0 0 1;0 -10 -7]
B=[0;0;1]
C=[1 0 0]
D=[0]
%Regulator design
%Check for controllability;
rank_of_M=rank(ctrb(A,B))
%Enter desired characteristic equation
chareqn=[1 7 25 15]
%Calculate desired closed-loop poles
desiredpoles=roots(chareqn);
%Calculate feedback gain matrix using Ackermann's formula
K=acker(A,B,desiredpoles)
%Reduced-order observer design
A1E=[1 0]
AEE=[0 1;-10 -7] % See equation (8.163)
obchareqn=[1 63.2 2039.4]
observerpoles=roots(obchareqn);
%Calculate observer matrix using Ackermann's formula
Ke=acker(AEE', A1E', observerpoles)

```

Note that the transposes of the partitioned matrices \mathbf{A}_{1e} and \mathbf{A}_{ee} from equation (8.163) are used in `acker` to calculate \mathbf{K}_e for the reduced-order state observer. The command window output is

```

» examp813
A=
    0     1     0
    0     0     1
    0   -10    -7
B=
    0
    0
    1
C=
    1     0     0
D=
    0

```

```

chareqn=
    1    7   25   15
K=
    15.0000   15.0000    0           % Regulator feedback matrix
AlE=
    1    0
AEE=
    0    1
   -10   -7
obchareqn=
    1.0e+003*
    0.0010   0.0632   2.0394
Ke=
    1.0e+003*
    0.0562   1.6360           %Reduced-order observer matrix

```

A1.9 Tutorial 8: Optimal and robust control system design

This tutorial uses the MATLAB Control System Toolbox for linear quadratic regulator, linear quadratic estimator (Kalman filter) and linear quadratic Gaussian control system design. The tutorial also employs the Robust Control Toolbox for multivariable robust control system design. Problems in Chapter 9 are used as design examples.

Example 9.1

This example uses the MATLAB command `lqr` to provide the continuous solution of the reduced matrix Riccati equation (9.25)

Filename: examp91.m

```

%Example 9.1
%Continuous Optimal Linear Quadratic Regulator (LQR) Design
A=[0 1;-1 -2]
B=[0;1]
Q=[2 0;0 1]
R=[1]
[K,P,E]=lqr(A,B,Q,R)

```

The output at the command window is

```

» examp91
A=
    0    1
   -1   -2
B=
    0
    1

```

```

Q=
    2   0
    0   1          %State weighting matrix, see equation (9.8)
R=
    1          %Control weighting matrix, see equation (9.8)
K=
    0.7321  0.5425  %State Feedback gain matrix, see equations
                    %(9.20) and (9.46)
P=
    2.4037  0.7321
    0.7321  0.5425  %Riccati matrix, see equation (9.45)
E=
   -1.2712+0.3406i  %Closed-loop eigenvalues
   -1.2712-0.3406i

```

Example 9.2

This example solves the discrete Riccati equation using a reverse-time recursive process, commencing with $\mathbf{P}(n) = \mathbf{0}$. Also tackled is the discrete state-tracking problem which solves an additional set of reverse-time state tracking equations (9.49) to generate a command vector \mathbf{v} .

Filename: examp92.m

```

%Example 9.2
%Discrete Solution of Riccati Equation
%Optimal Tracking Control Problem
A=[0 1;-1 -1];
B=[0;1];
Q=[10 0;0 1];
R=[1];
F=[0.9859 -0.27;0.08808 0.76677];
G=[-0.9952 0.01409;-0.04598 -0.08808];
S=[0;0];          %Initialize
T=0;
V=0;
Ts=0.1;
[AD,BD]=c2d(A,B,Ts);
P=[0 0;0 0];
H=BD'*P;
X=Ts*R+H*BD;
Y=H*AD;
K=X\Y;
%Discrete reverse-time solution of the Riccati equations (9.29)
%and (9.30)
%Discrete reverse-time solution of the state tracking equations
%(9.53) and (9.54)
for i=1:200
L=Ts*Q+K'*Ts*R*K;
M=AD-BD*K;
PP1=L+M'*P*M;          %Value of Riccati matrix at time (N-(k+1))T
RIN=[-sin(0.6284*T);0.6*cos(0.6284*T)];
SP1=F*S+G*RIN;
V=-B'*SP1;              %Value of command vector at time (N-(k+1))T
S=SP1;

```



```

T=T+Ts ;
P=PP1 ;
H=BD'*P ;
X=Ts*R+H*BD ;
Y=H*AD ;
K=X\Y ;                                %Value of feedback gain matrix at time
                                        % (N-(k+1))T
end

```

Checking values in the command window gives

```

» examp92
» A
A=
    0    1
   -1   -1
» B
B=
    0
    1
» Q
Q=
    10    0
    0    1                %State weighting matrix
» R
R=
    1                    %Control weighting matrix
» Ts
Ts=
    0.1000
» AD
AD=
    0.9952    0.0950
   -0.0950    0.9002    %Discrete-time state transition matrix
» BD
BD=
    0.0048
    0.0950                %Discrete-time control transition matrix
» K
K=
    2.0658    1.4880    %Discrete-time steady-state feedback gain
                        % matrix, after 200 reverse-time iterations
» P
P=
    8.0518    2.3145    %Steady-state value of Riccati matrix
    2.3145    1.6310

```

The reverse-time process is shown in Figure 9.3. The discrete-time steady-state feedback matrix could also have been found using `lqrd`, but this would not have generated the command vector \mathbf{v} . The forward-time tracking process is shown in Figure 9.4 using $\mathbf{K}(kT)$ and $\mathbf{v}(kT)$ to generate $\mathbf{u}_{\text{opt}}(kT)$ in equation (9.55). The script file *kalfilc.m* uses the MATLAB command `lqe` to solve the continuous linear quadratic estimator, or Kalman filter problem.

Filename: *kalfilc.m*

```
%Continuous Linear Quadratic Estimator (Kalman Filter)
A=[0 1;-1 -2]
B=[0;1]
C=[1 0;0 1]
D=[0]
R=[0.01 0;0 1]
Cd=[0.1 0;0 0.01]
Q=[0.1 0;0 0.1]
[K,P,E]=lqe(A,Cd,C,Q,R)
```

The command window text is

```
»kalfilc
A=
    0    1
   -1   -2
B=
    0
    1
C=
    1    0
    0    1
D=
    0
R=
    0.0100    0
         0    1.0000    %Measurement noise covariance matrix
Cd=
    0.1000    0
         0    0.0100    %Disturbance matrix
Q=
    0.1000    0
         0    0.1000    %Disturbance noise covariance matrix
K=
    0.1136 -0.0004
   -0.0435    0.0002    %Kalman gain matrix
P=
    0.0011 -0.0004
   -0.0004    0.0002    %Estimation error covariance matrix
E=
   -1.0569+0.2589i
   -1.0569-0.2589i    %Closed-loop estimator eigenvalues
```

The script file *kalfild.m* solves, in forward-time, the discrete solution of the Kalman filter equations, using equations (9.74), (9.75) and (9.76) in a recursive process. The MATLAB command *lqed* gives the same result.

Filename: *kalfild.m*

```
%Discrete Linear Quadratic Estimator (Kalman Filter)
%The algorithm uses discrete transition matrices A(T) and Cd(T)
A=[0 1;-2 -3]
```

```

Cd=[ 0.1  0; 0  0.1]
C=[ 1  0; 0  1]
ID=eye(2);
Ts=0.1
[AT,CD]=c2d(A,Cd,Ts)
R=[ 0.01  0; 0  1.0]
Q=[ 0.1  0; 0  0.1]
%Discrete solution of Kalman filter equations
%Initialize
P1=ID;           %Initial covariance matrix=identity matrix
P2=(AT*P1*AT')+(CD*Q*CD');
X=P2*C';
Y=(C*P2*C')+R;
K=X/Y;
P3=(ID-(K*C))*P2;
%Solve recursive equations
for i=1:20
P1=P3;
P2=(AT*P1*AT')+(CD*Q*CD');
X=P2*C';
Y=(C*P2*C')+R;
K=X/Y;
P3=(ID-(K*C))*P2;
end

```

Checking results in the command window

```

»kalfield
A=
    0    1
   -2   -3
Cd=
    0.1000    0
         0  0.1000    %Disturbance matrix
C=
    1    0
    0    1
Ts=
    0.1000
AT=
    0.9909    0.0861
   -0.1722    0.7326    %Discrete-time state transition matrix
CD=
    0.0100    0.0005
   -0.0009    0.0086    %Discrete-time disturbance transition
                        %matrix
R=
    0.0100    0
         0  1.0000    %Measurement noise covariance matrix
Q=
    0.1000    0
         0  0.1000    %Disturbance noise covariance matrix
» K
K=
    0.0260   -0.0002
   -0.0181    0.0002    %Kalman gain matrix after 21 iterations

```

Case study

Example 9.3

This is a china clay band drying oven. The state and output variables are burner temperature, dryer temperature and clay moisture content. The control parameters are burner gas supply valve angle and clay feed-rate. A linear quadratic Gaussian control strategy is to be implemented. Script file *examp93.m* calculates the optimal feedback control matrix \mathbf{K} and also the Kalman gain matrix \mathbf{K}_e using the recursive equation (9.99).

Filename: *examp93.m*

```
%Linear Quadratic Gaussian (LQG) Design
%Case Study Example 9.3 Clay Drying Oven
%Optimal Controller
A=[-0.02128 0 0;0.0006 -0.005 0;0 -0.00038 -0.00227]
B=[8.93617;0;0]
Cd=[0.1 0 0;0 0.1 0;0 0 0.00132]
C=[1 0 0;0 1 0;0 0 1]
D=[0]
Ts=2;
[AT, BT]=c2d(A,B, Ts)
Q=[0 0 0;0 0.5 0;0 0 20]
R=[1]
[K,P,E]=lqr(A,B,Q,R)
%Kalman Filter
Re=[0.01 0 0;0 0.01 0;0 0 7.46]
Qe=[0.1 0 0;0 0.1 0;0 0 0.1]
[AT,CD]=c2d(A,Cd,Ts)
%Initialize
%Implement equations (9.99)
ID=eye(3);
P1=ID; %Initial covariance matrix=identity matrix
P2=(AT*P1*AT')+(CD*Qe*CD');
X=P2*C';
Y=(C*P2*C')+Re;
Ke=X/Y;
P3=(ID-(Ke*C))*P2;
%Solve recursive equations
for i=1:19
P1=P3;
P2=(AT*P1*AT')+(CD*Qe*CD');
X=P2*C';
Y=(C*P2*C')+Re;
Ke=X/Y;
P3=(ID-(Ke*C))*P2;
End
Ke
P3
```

The output at the command window is

```
» examp93
A=
```

```

-0.0213      0      0
0.0006 -0.0050      0
B=      0 -0.0004 -0.0023
      8.9362
      0
      0
Cd=
      0.1000      0      0
      0 0.1000      0 %Disturbance matrix, equation
      0      0 0.0013
C=
      1 0 0
      0 1 0
      0 0 1
D=
      0
AT=
      0.9583      0      0
      0.0012 0.9900      0 %A(T), Ts=2 seconds
      0.0000 -0.0008 0.9955
BT=
      17.4974 %B(T), equation (9.86)
      0.0105
      0.0000
Q=
      0      0      0
      0      0.5000      0 %State weighting matrix
      0      0 20.0000
R=
      1 %Control weighting matrix
K=
      0.0072 0.6442 -1.8265 %Feedback gain matrix,
      %equation (9.92)
P=
      1.0e+003*
      0.0000 0.0001 -0.0002
      0.0001 0.0108 -0.0300 %Riccati matrix, equation
      -0.0002 -0.0300 3.6704 %(9.91)
E=
      -0.0449+0.0422i %Closed-loop eigenvalues,
      %equation (9.93)
      -0.0449-0.0422i
      -0.0033
Re=
      0.0100      0      0
      0 0.0100      0 %Measurement noise covariance
      0      0 7.4600 %matrix, equation (9.100)
Qe=
      0.1000      0      0
      0 0.1000      0 %Disturbance noise covariance
      0      0 0.1000 %matrix, equation(9.101)
CD=
      0.1958      0      0 %Discrete-time disturbance
      0.0001 0.1990      0 %transition matrix, equation
      %(9.95)

```

```

Ke=      0.0000  -0.0001  0.0026
      0.4408   0.0003  0.0000   %Kalman gain matrix after 20
                                     %iterations,
      0.0003   0.4579  0.0000   %equation (9.102)
      0.0000  -0.0006  0.0325   % and Figure 9.16
P3=      0.0044   0.0000  0.0000   %Estimation error covariance
                                     %matrix,
      0.0000   0.0046  0.0000   %after 20 iterations,
      0.0000   0.0000  0.2426   %equation (9.102)

```

The implementation of the LQG design is shown in Figures 9.12 through to 9.17.

Example 9.6

This is a multivariable robust control problem that calculates the optimal H_∞ controller. The MATLAB command `hinftopt` undertakes a number of iterations by varying a parameter γ until a best solution, within a given tolerance, is achieved.

Filename: *examp96.m*

```

%Example 9.6
%Multivariable robust control using H infinity
%Singular value loop shaping using the weighted mixed
%sensitivity approach
nug=200;
dng=[1 3 102 200];
[ag,bg,cg,dg]=tf2ss(nug,dng);
ss_g=mksys(ag,bg,cg,dg);
w1=[1 100;100 1];
nw1neg=[100 1];
dw1neg=[1 100];
w2=[1;1];
w3=[100 1;1 100];
nw3neg=[1 100];
dw3neg=[100 1];
[TSS_1]=augtf(ss_g,w1,w2,w3);
[ap,bp,cp,dp]=branch(TSS_1);
[gamopt,ss_f,ss_cl]=hinftopt(TSS_1,1);
[acp,bcp,ccp,dcp]=branch(ss_f);
[acl,bcl,ccl,dcl]=branch(ss_cl);
[numcp,dencp]=ss2tf(acp,bcp,ccp,dcp,1);
printsys(nug,dng,'s')
printsys(nw1neg,dw1neg,'s')
printsys(nw3neg,dw3neg,'s')
w=logspace(-3,3,200);
%[mag,phase,w]=bode(nw1neg,dw1neg,w);
%[mag1,phase1,w]=bode(nw3neg,dw3neg,w);
%semilogx(w,20*log10(mag),w,20*log10(mag1)),grid;
[sv,w]=sigma(ss_g);
%[sv,w]=sigma(ss_cl);
%[sv,w]=sigma(ss_f);
semilogx(w,20*log10(sv)),grid;
ag
bg

```

```
cg
dg
acp
bcp
ccp
dcp
printsys(numcp,dencp,'s')
```

The command `mksys` packs the plant state-space matrices **ag**, **bg**, **cg** and **dg** into a tree structure `ss_g`.

The command `augtf` augments the plant with the weighting functions as shown in Figure 9.31. The `branch` command recovers the matrices **ap**, **bp**, **cp** and **dp** packed in `TSS_`. The `hinftopt` command produces the following output in the command window

No	Gamma	D11<=1	P-Exist	P>=0	S-Exist	s>0	lam(PS)<1	C.L.
1	1.0000e+000	OK	OK	FAIL	OK	OK	OK	UNST
2	5.0000e-001	OK	OK	FAIL	OK	OK	OK	UNST
3	2.5000e-001	OK	OK	FAIL	OK	OK	OK	UNST
4	1.2500e-001	OK	OK	OK	OK	OK	OK	STAB
5	1.8750e-001	OK	OK	FAIL	OK	OK	OK	UNST
6	1.5625e-001	OK	OK	FAIL	OK	OK	OK	UNST
7	1.4063e-001	OK	OK	FAIL	OK	OK	OK	UNST
8	1.3281e-001	OK	OK	FAIL	OK	OK	OK	UNST
9	1.2891e-001	OK	OK	OK	OK	OK	OK	STAB
10	1.3086e-001	OK	OK	FAIL	OK	OK	OK	UNST
11	1.2988e-001	OK	OK	OK	OK	OK	OK	STAB

Iteration no. 11 is your best answer under the tolerance: 0.0100.

After eleven iterations, `hinftopt` identifies that γ in equation (9.176) has a best value of 0.13. The command `sigma` calculates the data for a singular value Bode diagram as shown in Figures 9.32, 9.34 and 9.35. Other information printed in the command window is given below

```
num/den=
      200
-----
s^3+3s^2+102s+200      %Plant transfer function

num/den=
  100s+1
-----
s+100      %Ws-1(s)

num/den=
  s+100
-----
100s+1      %WT-1(s)

ag=
  -3  -102  -200
   1    0    0
   0    1    0
```

```

bg=
    1
    0
    0
cg=
    0    0    200
dg=
    0
acp=
   -0.0100   -0.0022    0.0039    0.0148    0.1367    %Controller
   -0.0086   -7.7634   21.6533   37.1636   621.8932    %matrices,
   -0.0461   -2.0317   -3.1477   -2.2338   -81.8751    %See equation
    0.4353    0.1067    8.8071  -102.2531   -37.7767    %(9.178)
    0.6756    0.1970   13.5575   -3.7282  -162.5129
bcp=
   -7.9756
    0.0906
    0.5820
   -8.8009
  -13.6595
ccp=
   -0.0861    0.1698    0.1213   -0.6544   -11.1708
dcp=
    0
num/den=
      159.1184s^4+16389.1905s^3+63965.5785s^2+1654830.8855s+3182367.0874
      s^5+275.687s^4+20439.8141s^3+324835.0198s^2+3782679.1393s+37794.3283

      %Controller transfer function,
      %See equation (9.179)

```

A1.10 Tutorial 9: Intelligent control system design

This tutorial uses the MATLAB Control System Toolbox, the Fuzzy Logic Toolbox and the Neural Network Toolbox. Problems in Chapter 10 are used as design examples.

Example 10.3

This is the inverted pendulum control problem and, as a benchmark, is initially solved as a multivariable control problem, using pole placement (Ackermann's formula) to calculate the feedback gain matrix in *examp103.m*.

Filename: examp103m

```

%Regulator design using pole placement
%Inverted pendulum problem
A=[0 1 0 0;9.81 0 0 0;0 0 0 1;-3.27 0 0 0]
B=[0;-0.667;0;0.889]
C=[1 0 0 0]
D=0
%Check for controllability;
rank_of_M=rank(ctrb(A,B))

```



```
%Enter desired characteristic equation
chareqn=[1 12 72 192 256]
%Calculate desired closed-loop poles
desiredpoles=roots(chareqn)
%Calculate feedback gain matrix using Ackermann's formula
K=acker(A,B,desiredpoles)
```

The output at the command window is

```
» exampl03
A=
    0    1.0000    0    0
   9.8100    0    0    0
    0    0    0    1.0000
  -3.2700    0    0    0
B=
    0
  -0.6670
    0
   0.8890
C=
    1    0    0    0
D=
    0
rank_of_M=
    4
chareqn=
    1    12    72    192    256
desiredpoles=
  -4.0000+4.0000i
  -4.0000-4.0000i
  -2.0000+2.0000i
  -2.0000-2.0000i
K=
  -174.8258  -57.1201  -39.1437  -29.3578
```

The state-variable feedback solution was implemented in SIMULINK as shown in Figure A1.4.

Inverted Pendulum, Fuzzy Logic Controller Design: The fuzzy logic controller used for the inverted pendulum problem has four input windows and one output window as shown in Figure 10.14. One version of the controller has a Mamdani-type rulebase consisting of 11 rules as given in equation (10.52). The MATLAB Fuzzy Inference System (FIS) Editor can be entered by typing at the MATLAB prompt

```
» fuzzy
```

This brings up the main menu screen as shown in Figure A1.5. Double-clicking on the 'theta' icon brings up the membership function editor. Figure A1.6 shows the NB fuzzy set being given the trapizoidal (trapmf) membership function $[-1 \ -1 \ -0.10]$. Other membership functions include gaussmf (Gaussian), trimf (triangular) and gbellmf (generalized bell). For further information type

```
» help fuzzy
```

Returning to the main menu and double-clicking on the centre box entitled 'Pendulum 11' (mamdani) will bring up the FIS rule editor, as shown in Figure A1.7. Highlighted are the antecedents and the consequent of Rule 1.

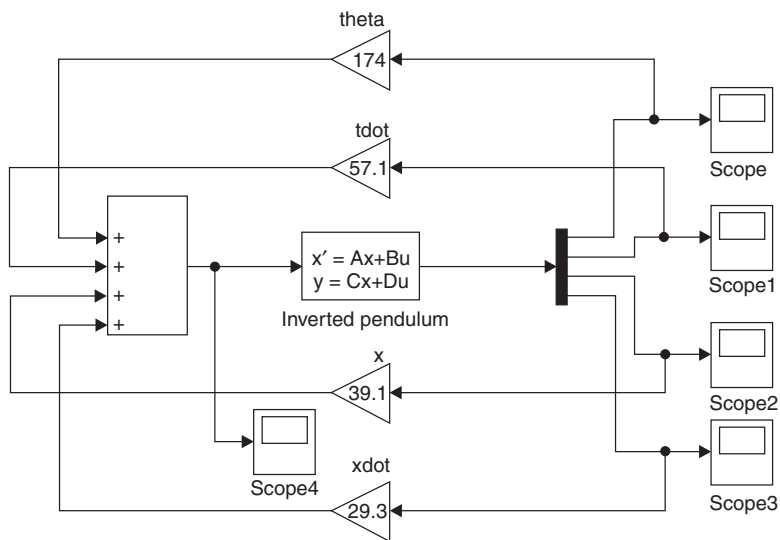


Fig. A1.4 Inverted pendulum control system design using pole placement

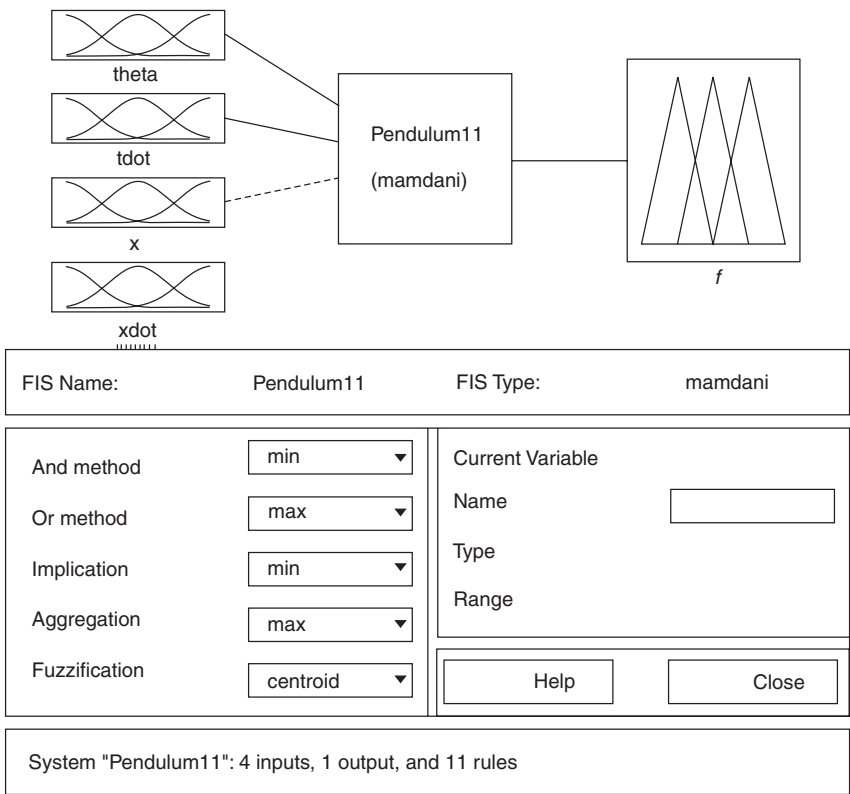


Fig. A1.5 Main menu of Fuzzy Inference System (FIS) editor.

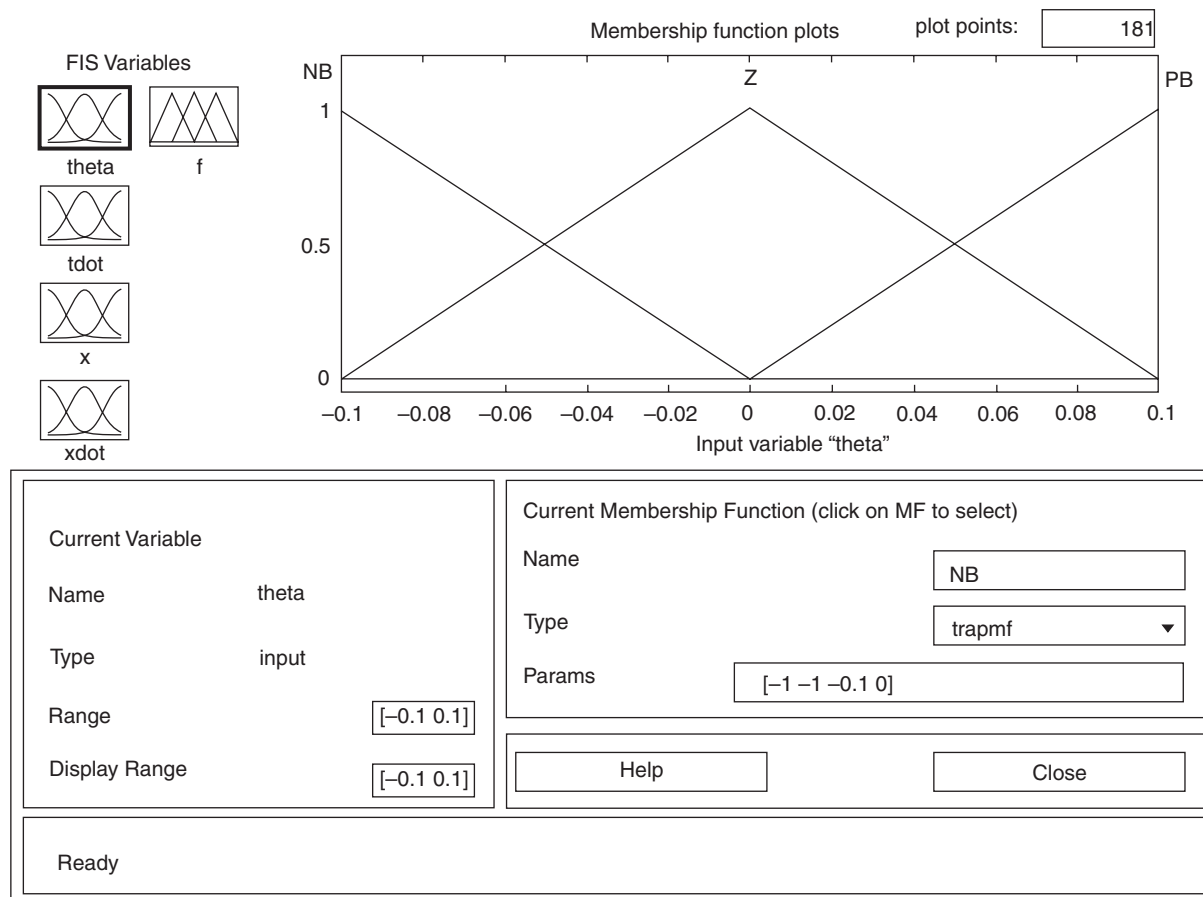


Fig. A1.6 FIS membership function editor.

1. If (theta is PB) and (tdot is PB) then (f is PB) (1)
2. If (theta is PB) and (tdot is Z) then (f is PB) (1)
3. If (theta is PB) and (tdot is NB) then (f is Z) (1)
4. If (theta is Z) and (tdot is PB) then (f is PB) (1)
5. If (theta is Z) and (tdot is Z) then (f is Z) (1)
6. If (theta is Z) and (tdot is NB) then (f is NB) (1)
7. If (theta is NB) and (tdot is PB) then (f is Z) (1)
8. If (theta is NB) and (tdot is Z) then (f is NB) (1)
9. If (theta is NB) and (tdot is NB) then (f is NB) (1)
10. If (tdot is PB) then (f is PB) (1)
11. If (tdot is NB) then (f is NB) (1)

If

theta is

NB

Z

PB

none

not

and

tdot is

NB

Z

PB

none

not

and

x is

NB

Z

PB

none

not

and

xdot is

NB

Z

PB

none

not

Then

f is

NB

Z

PB

none

not

Connection

☐ or
☒ and

Weight

1

Delete rule

Add rule

Change rule

FIS Name: Pendulum11

Help

Close

Fig. A1.7 FIS rule editor.

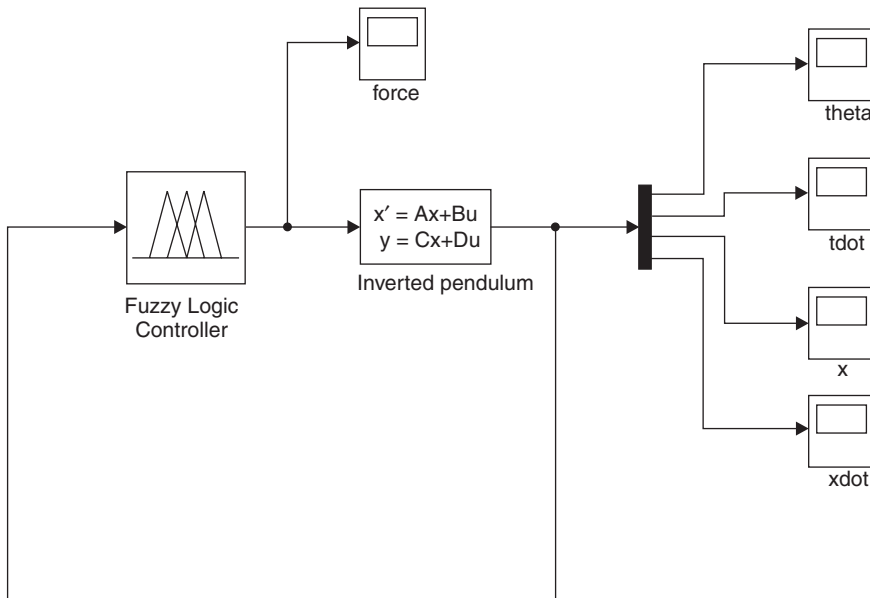


Fig. A1.8 Simulink implementation of inverted pendulum fuzzy logic control problem.

When editing is complete, the controller is stored as a *.fis* file, in this case *pendulum11.fis*. The inverted pendulum fuzzy logic control problem can now be implemented in SIMULINK as shown in Figure A1.8. The fuzzy logic icon is obtained by opening the fuzzy logic toolbox within the Simulink Library Browser, and dragging it across. Running the simulation as it stands in Figure A1.8 will bring up a MATLAB error because the properties of the fuzzy logic controller have not been defined. At the MATLAB prompt, type

```
» fismat=readfis
```

This will allow you to select from a directory a suitable stored filename, i.e. *pendulum11.fis*. The system will respond with

```
fismat=
name:      'Pendulum11'
type:      'mamdani'
andMethod: 'min'
orMethod:  'max'
defuzzMethod: 'centroid'
impMethod: 'min'
aggMethod: 'max'
input:     [1x4 struct]
output:    [1x1 struct]
rule:      [1x11 struct]
```

This means that the fuzzy logic controller parameters have been placed in the work-space under 'fismat', and that the simulation can now proceed. More details on the properties of the fuzzy logic controller can be found by typing

```
» out=getfis(fismat)
```

The system will respond with

```
Name = Pendulum11
Type = mamdani
NumInputs=4
InLabels=
    theta
    tdot
    x
    xdot
NumOutputs=1
OutLabels=
    f
NumRules    =11
AndMethod   =min
OrMethod    =max
ImpMethod   =min
AggMethod   =max
DefuzzMethod=centroid
Out=
Pendulum 11
```

The results of the pole placement and the 11 rule and 22 rule fuzzy logic controllers are compared in Figure 10.15. Figure 10.16 shows the $\theta-\dot{\theta}$ control surface, also generated using the FIS Editor.

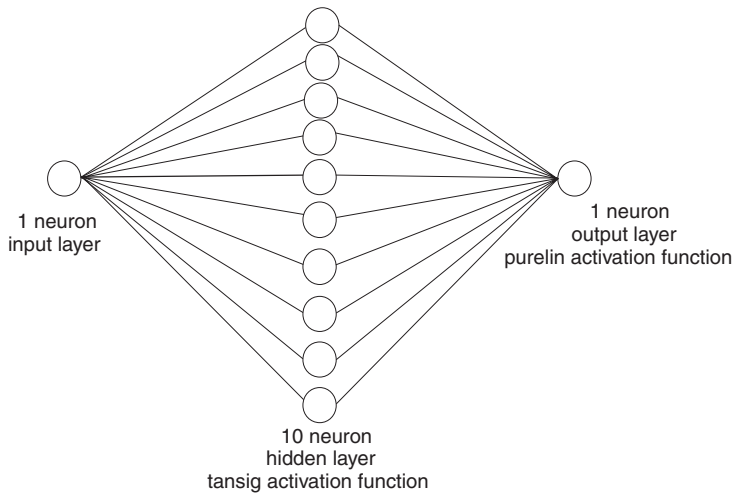


Fig. A1.9 Demonstration neural network.

The MATLAB Neural Network Toolbox: This Toolbox was not used in the Examples given in Chapter 10. For details on the Toolbox, type

```
» help nnet
```

To demonstrate how the Toolbox is used, consider a neural network with a structure shown in Figure A1.9.

The script file *neural1.m* trains and runs the network.

Filename: neural1.m

```
%Feedforward Back-propagation Neural Network
%Network structure: 1:10(tansig):1(purelin)
%P=Input values
P=[0 1 2 3 4 5 6 7 8];
%T=Target values
T=[0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
plot(P,T,'o');
%Use Levenberg-Marquardt back-propagation training
net=newff([0 8],[10 1],{'tansig','purelin'},'trainlm');
Y1=sim(net,P);
plot(P,T,P,Y1,'o');
net.trainParam.epochs=50;
net=train(net,P,T);
Y2=sim(net,P);
plot(P,T,P,Y2,'o');
```

P is a vector of inputs and **T** a vector of target (desired) values. The command `newff` creates the feed-forward network, defines the activation functions and the training method. The default is Levenberg–Marquardt back-propagation training since it is fast, but it does require a lot of memory. The `train` command trains the network, and in this case, the network is trained for 50 epochs. The results before and after training are plotted.

Appendix 2

Matrix algebra

A2.1 Definitions

Matrix: An $m \times n$ matrix \mathbf{A} is an array of elements with m rows and n columns, and is written as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad (\text{A2.1})$$

If $m = n$, \mathbf{A} is a square matrix.

Vector: A column vector \mathbf{x} is

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \quad (\text{A2.2})$$

A row vector \mathbf{y} is

$$\mathbf{y} = [y_1 \ y_2 \ \dots \ y_n] \quad (\text{A2.3})$$

Null matrix: This has all elements equal to zero.

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (\text{A2.4})$$

Diagonal matrix: This is a square matrix with all elements off the diagonal equal to zero.

$$\mathbf{C} = \begin{bmatrix} c_{11} & 0 & 0 \\ 0 & c_{22} & 0 \\ 0 & 0 & c_{33} \end{bmatrix} \quad (\text{A2.5})$$

Identity matrix: This is a diagonal matrix with all diagonal elements equal to unity, and is normally denoted by **I**.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{A2.6})$$

Symmetric matrix: This is a square matrix where elements $a_{ij} = a_{ji}$.

$$\mathbf{S} = \begin{bmatrix} 5 & 8 & 9 \\ 8 & 4 & 2 \\ 9 & 2 & 3 \end{bmatrix} \quad (\text{A2.7})$$

A2.2 Matrix operations

Transpose of a matrix: The transpose of a matrix **A**, denoted by \mathbf{A}^T , is formed by interchanging its rows and columns.

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 3 & 1 & 9 \\ 6 & 8 & 4 \end{bmatrix} \quad (\text{A2.8})$$

$$\mathbf{A}^T = \begin{bmatrix} 2 & 3 & 6 \\ 4 & 1 & 8 \\ 5 & 9 & 4 \end{bmatrix} \quad (\text{A2.9})$$

Determinant of a matrix: Given a 2×2 matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (\text{A2.10})$$

Its determinant is

$$\det \mathbf{A} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12} \quad (\text{A2.11})$$

Minors of an Element: The minor M_{ij} of element a_{ij} of $\det \mathbf{A}$ is the determinant formed by removing the i th row and the j th column from $\det \mathbf{A}$.

$$\det \mathbf{A} = \begin{vmatrix} 8 & 6 & 2 \\ 4 & 3 & 1 \\ 9 & 5 & 7 \end{vmatrix}$$

$$M_{23} = \begin{vmatrix} 8 & 6 \\ 9 & 5 \end{vmatrix} = -14 \quad (\text{A2.12})$$

Cofactor of an Element: The cofactor C_{ij} of element a_{ij} of $\det \mathbf{A}$ is defined as

$$C_{ij} = (-1)^{(i+j)} M_{ij}$$

In general, the determinant of a square matrix is given by

$$\det \mathbf{A} = \sum_{k=1}^n a_{ik} C_{ik} \text{ (expanding along the } i\text{th row)}$$

or

$$\det \mathbf{A} = \sum_{k=1}^m a_{kj} C_{kj} \text{ (expanding along the } j\text{th column)} \quad (\text{A2.13})$$

Given

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & 3 & 0 \\ 6 & 8 & 9 \end{bmatrix}$$

Expanding along the first column gives

$$\begin{aligned} \det \mathbf{A} &= 2 \begin{vmatrix} 3 & 0 \\ 8 & 9 \end{vmatrix} - 1 \begin{vmatrix} 4 & 5 \\ 8 & 9 \end{vmatrix} + 6 \begin{vmatrix} 4 & 5 \\ 3 & 0 \end{vmatrix} \\ &= 2(27) - 1(-4) + 6(-15) \\ &= -32 \end{aligned} \quad (\text{A2.14})$$

Singular matrix: A matrix is singular if its determinant is zero.

Nonsingular matrix: A matrix is nonsingular if its determinant is not zero.

Adjoint of a matrix: The adjoint of an $n \times n$ matrix is the transpose of the matrix when all elements have been replaced by their cofactors.

$$\text{adj } \mathbf{A} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \dots & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix}^T \quad (\text{A2.15})$$

Inverse of a matrix: An $n \times n$ matrix \mathbf{A} has an inverse \mathbf{A}^{-1} , which satisfies the relationship

$$\mathbf{A}^{-1} \mathbf{A} = \mathbf{A} \mathbf{A}^{-1} = \mathbf{I} \quad (\text{A2.16})$$

The inverse of \mathbf{A} is given by

$$\mathbf{A}^{-1} = \frac{\text{adj } \mathbf{A}}{\det \mathbf{A}} \quad (\text{A2.17})$$

Addition and subtraction of matrices: The sum of two matrices

$$\mathbf{A} + \mathbf{B} = \mathbf{C}$$

is given by

$$a_{ij} + b_{ij} = c_{ij} \quad (\text{A2.18})$$

the difference between two matrices

$$\mathbf{A} - \mathbf{B} = \mathbf{C}$$

is given by

$$a_{ij} - b_{ij} = c_{ij} \quad (\text{A2.19})$$

Multiplication of matrices: The product of two matrices

$$\mathbf{AB} = \mathbf{C}$$

is given by

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{ki} \quad (\text{A2.20})$$

Multiplication by a constant

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$k\mathbf{A} = \begin{bmatrix} ka_{11} & ka_{12} \\ ka_{21} & ka_{22} \end{bmatrix} \quad (\text{A2.21})$$

Note that in general, multiplication is not commutative, i.e. $\mathbf{AB} \neq \mathbf{BA}$.

Trace of a matrix: The trace of an $n \times n$ matrix is the sum of the diagonal elements of the matrix.

$$\text{tr } \mathbf{A} = a_{11} + a_{22} + \cdots + a_{nn} \quad (\text{A2.22})$$

Rank of a matrix: The rank of a matrix is equal to the number of linearly independent rows or columns. The rank can be found by determining the largest square submatrix that is nonsingular.

If

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 1 \\ 1 & 2 & 3 \end{bmatrix} \quad (\text{A2.23})$$

since $\det \mathbf{A}$ is zero, the 3×3 matrix is singular. Consider a submatrix

$$\mathbf{A}_{\text{sub}} = \begin{bmatrix} 1 & 2 \\ 0 & 4 \end{bmatrix} \quad (\text{A2.24})$$

The determinant of \mathbf{A}_{sub} is 4, hence the rank of \mathbf{A} is the size of \mathbf{A}_{sub} , i.e. 2.

References and further reading

- Ackermann, J. (1972) Der Entwurf Linearer Regelungssysteme im Zustandsraum, *Regelungstechnik und Prozessdatenverarbeitung*, **7**, pp. 297–300.
- Anderson, J.A. (1972) A Simple Neural Network Generating an Interactive Memory, *Mathematical Biosciences*, **14**, pp. 197–220.
- Anderson, B.D.O. and Moore, J.B. (1979) *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ.
- Anderson, B.D.O. and Moore, J.B. (1990) *Optimal Control*, Prentice-Hall, Englewood Cliffs, NJ.
- Åström, K.J. and Wittenmark, B. (1984) *Computer Controlled Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Åström, K.J. and Wittenmark, B. (1989) *Adaptive Control*, Addison-Wesley, Reading, Mass.
- Athans, M. (1971) The Role and Use of the Stochastic Linear-Quadratic-Gaussian Problem in Control System Design, *IEEE Trans. on Automatic Control* AC-16, **6**, pp. 529–551.
- Atherton, D.P. (1982) *Nonlinear Control Engineering*, Van Nostrand Reinhold, London.
- Bellman, R. (1957) *Dynamic Programming*, Princeton University Press, Princeton, NJ.
- Bennett, S. (1984) Nicholas Minorsky and the automatic steering of ships, *IEEE Control Systems Magazine*, **4**, pp. 10–15.
- Bode, H.W. (1945) *Network Analysis and Feedback Amplifier Design*, Van Nostrand, Princeton, NJ.
- Brown, M. and Harris, C. (1994) *Neurofuzzy Adaptive Modelling and Control*, Prentice-Hall International (UK) Hemel Hempstead, UK.
- Burns, R.S. (1984) The Automatic Control of Large Ships in Confined Waters, PhD Thesis, University of Plymouth (then Plymouth Polytechnic).
- Burns, R.S. (1989) Application of the Riccati Equation in the Control and Guidance of Marine Vehicles. In: *The Riccati Equation in Control, Systems, and Signals*, Pitagora Editrice, Bologna, Italy, pp. 18–23.
- Burns, R.S. (1990) The Design, Development and Implementation of an Optimal Guidance System for Ships in Confined Waters, *Proc: Ninth Ship Control Systems Symposium*, Naval Sea Systems Command, Department of the Navy, Bethesda, USA, 9–14 September, **3**, pp. 386–401.
- Burns, R.S. (1991) A Multivariable Mathematical Model for Simulating the Total Motion of Surface Ships. In: *Proc. European Simulation Multiconference*, The Society for Computer Simulation International, Copenhagen, Denmark, 17–19 June.
- Burns, R.S. (1995) The Use of Artificial Networks for the Intelligent Optimal Control of Surface Ships, *IEEE Journal of Oceanic Engineering*, **20**(1); Special Issue: *Advanced Control & Signal Processing for Oceanic Applications*, **20**(1), pp. 66–72.
- Burns, R.S. (1997) The Application of Artificial Intelligence Techniques to Modelling and Control of Surface Ships. In: *Proc. 11th Ship Control Systems Symposium*, Southampton UK, April, **1**, 77–83.

- Burns, R.S. (1997) Intelligent Manufacturing, *Journal of Aircraft Engineering and Aerospace Technology*, MCB University Press, **69**(5), pp. 440–446.
- Burns, R.S. and Richter, R. (1995) A Neural Network Approach to the Control of Surface Ships, *Journal of Control Engineering Practice*, International Federation of Automatic Control, Elsevier Science, **4**(3), pp. 411–416.
- Burns, R.S., Richter, R. and Polkinghorne, M.N. (1995) A Multivariable Neural Network Ship Mathematical Model. In: *Marine Technology and Transportation*, Graczyk, T., Jastrzebski, T., Brebbia, C.A. and Burns R.S. (eds.), Southampton U.K., Computational Mechanics.
- Burns, R.S., Sutton, R. and Craven, P.J. (2000) A Multivariable Online Intelligent Autopilot Design Study. In: *The Ocean Engineering Handbook*, El-Hawary, F. (ed.), CRC Press, Boca Raton, Florida, pp. 252–259.
- Cadzow, J.A. and Martens, H.R. (1970) *Discrete-Time and Computer Control Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J.
- Chiang, R.Y. (1988) Modern Robust Control Theory, PhD Dissertation, USC.
- Chiang, R.Y. and Safonov, M.G. (1992) *Robust Control Toolbox for Use with MATLAB. Users Guide*, MathWorks.
- Chu, C.C. (1985) H_∞ -Optimization and Robust Multivariable Control, PhD Thesis, University of Minnesota, Minneapolis, MN.
- Craven, P.J. (1999) Intelligent Control Strategies for an Autonomous Underwater Vehicle, PhD Thesis, Department of Mechanical and Marine Engineering, University of Plymouth, UK.
- Craven, P.J., Sutton, R. and Burns, R.S. (1997) Intelligent Course Changing Control of an Autonomous Underwater Vehicle. In: *Twelfth International Conference on Systems Engineering*, Coventry, UK, September, **1**, pp. 159–164.
- Demuth, H. and Beale, M. (1993) *Neural Network Toolbox for Use with MATLAB – User's Guide*, The MathWorks, Inc., Natick, Mass.
- Dorato, P. (ed.) (1987) *Robust Control*, IEEE Press, New York.
- Dorf, R.C. (1992) *Modern Control Systems*, 6th ed., Addison-Wesley, Reading, Mass.
- Dove, M.J., Burns, R.S. and Evison, J.L. (1986) The use of Kalman Filters in Navigation Systems – Current Status and Future Possibilities. In: *Proc. of the Int. Conf. on Computer Aided Design, Manf. and Operation in the Marine and Offshore Industries*, Keramidas, G.A. and Murthy, T.K.S. (eds.), Springer-Verlag, Washington, DC, pp. 361–374.
- Drew, B.L. and Burns, R.S. (1992) Simulation of Optimal Control and Filtering of an Industrial China Clay Dryer. In: *European Simulation Symposium on Simulation and AI in Computer Aided Techniques*, The Society for Computer Simulation, Dresden, 5–8 November, pp. 531–535.
- Dreyfus, S.E. (1962) Variational Problems with Inequality Constraints, *J. Math. Anal. Appl.*, **4**, p. 297.
- Dutton, K., Thompson, S. and Barraclough, B. (1997) *The Art of Control Engineering*, Addison-Wesley-Longman, Harlow, Essex.
- Evans, W.R. (1948) Graphical Analysis of Control Systems, *Transactions of the AIEE*, **67**, pp. 547–551.
- Francis, B.A. (1987) *A Course in H_∞ Control Theory*, Springer-Verlag, New York.
- Franklin, G.F., Powell, J.D. and Workman, M.L. (1990) *Digital Control of Dynamic Systems*, 2nd ed., Addison-Wesley, Menlow Park, CA.
- Goldberg, D.E. (1983) Computer-aided gas pipeline operation using genetic algorithms and rule learning (Doctoral dissertation, University of Michigan). *Dissertation Abstracts International*, **44**(10), p. 3174B (University Microfilms No. 8402282).
- Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, Mass.
- Grace, A., Lamb, A.J., Little, J.N. and Thompson, C.M. (1992) *Control System Toolbox for Use with MATLAB – User's Guide*, The MathWorks, Inc., Natick, Mass.

- Grimble, M.J., Patton, R.J. and Wise, D.A. (1979) The design of dynamic ship positioning control systems using extended Kalman filtering techniques, *IEEE Conference, Oceans '79*, CA, San Diego.
- Grimble, M.J. and Johnson, M.A. (1988) *Optimal Control and Stochastic Estimation: Theory and Application, Vols 1 and 2*, John Wiley & Sons, Chichester, UK.
- Healey, M. (1967) *Principles of Automatic Control*, The English Universities Press, London.
- Hebb, D.O. (1949) *The Organization of Behaviour*, Wiley, New York.
- Holland, J.H. (1975) *Adaption in natural and artificial systems*, The University of Michigan Press, Ann Arbor.
- Hughs, T.P. (1971) *Elmer Sperry: Inventor and Engineer*, Johns Hopkins Press, Baltimore. pp. 232–233.
- James, H.M., Nichols, N.B. and Phillips, R.S. (1947) *Theory of Servomechanisms*, McGraw-Hill, New York.
- Jang, J.S.R. (1993) ANFIS: Adaptive Network-based Fuzzy Inference System, *IEEE Transactions on Systems, Man and Cybernetics*, **23**, pp. 665–685.
- Johnson, J. and Picton, P. (1995) *Designing Intelligent Machines, Vol. 2, Concepts in Artificial Intelligence*, Butterworth-Heinemann (in association with The Open University), Oxford, UK.
- Jury, E.I. (1958) *Sampled-Data Control Systems*, John Wiley & Sons, New York.
- Kalman, R.E. (1961) On the General Theory of Control Systems. In: *Proceedings of the First International Congress IFAC, Moscow 1960: Automatic and Remote Control*, Butterworth & Co., London, pp. 481–492.
- Kalman, R.E. and Bucy, R.S. (1961) New Results in Linear Filtering and Prediction Theory, *J. of Basic Eng., Trans. of the Am. Soc. of Mech. Eng.*, pp. 95–108.
- Kohonen, T. (1988) *Self-Organization and Associative Memory*, Springer-Verlag, Berlin.
- Kuo, B.C. (1980) *Digital Control Systems*, Holt, Rinehart and Winston, Inc., New York.
- Kurzweil, R. (1990) *The Age of Intelligent Machines*, M.I.T. Press, Cambridge, Mass.
- Laub, A.J. (1979) A Schur Method for Solving Riccati Equations, *IEEE Trans. on Automat. Contr.*, **AC-24**, pp. 913–921.
- Lehtomaki, N.A., Sandell, Jr., N.R. and Athans, M. (1981) Robustness Results in Linear-Quadratic Gaussian Based Multivariable Control Designs, *IEEE Trans. on Automat. Contr.*, **AC-26**(1), pp. 75–92.
- Leigh, J.R. (1985) *Applied Digital Control*, Prentice-Hall International, Englewood Cliffs, NJ.
- Luenberger, D.G. (1964) Observing the State of a Linear System, *IEEE Trans. Military Electronics*, **MIL-8**, pp. 74–80.
- Lyapunov, A.M. (1907) Problème général de la stabilité du mouvement. In: *Ann. Fac. Sci., Toulouse*, **9**, pp. 203–474 (Translation of the original paper published in 1893 in *Comm. Soc. Math. Kharkow*).
- MacFarlane, A.G.J. and Kouvaritakis, B. (1977) A design technique for linear multivariable feedback systems, *International Journal of Control*, **25**, pp. 837–874.
- Mamdani, E.H. (1976) Advances in linguistic synthesis of fuzzy controllers, *Int. J. Man & Mach. Studies*, **8**(6), pp. 669–678.
- Maxwell, J.C. (1868) On Governors. In: *Proceedings of the Royal Society of London*, **16**.
- Mayr, O. (1970) *The Origins of Feedback Control*, M.I.T. Press, Cambridge, Mass.
- McCulloch, W.S. and Pitts, W.H. (1943) A logical calculus of ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, **5**, pp. 115–133.
- Merritt, H.E. (1967) *Hydraulic Control Systems*, John Wiley and Sons, New York.
- Minorski, N. (1922) Directional stability of automatically steered bodies, *J. American Society of Naval Engineers*, **34**, pp. 280–309.
- Minorski, N. (1941) Note on the angular motion of ships. *Trans. American Society of Mech. Eng.*, **63**, pp. 111–120.

- Morari, M. and Zafiriou, E. (1989) *Robust Process Control*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Mościński, J. and Ogonowski, Z. (eds.) (1995) *Advanced Control with MATLAB and SIMULINK*, Ellis Horwood, Hemel Hempstead, UK.
- Nyquist, H. (1932) Regeneration Theory, *Bell System Technical Journal*, **11**, pp. 126–147.
- Ogata, K. (1995) *Discrete-Time Control Systems*, 2nd ed., Prentice-Hall, Inc., Upper Saddle River, NJ.
- Ogata, K. (1997) *Modern Control Engineering*, 3rd ed., Prentice-Hall, Inc., Upper Saddle River, NJ.
- Payne, H.J. and Silverman, L.M. (1973) On the Discrete Time Algebraic Riccati Equation, *IEEE Trans. Automatic Control*, **AC-18**, pp. 226–234.
- Pearson, A.R., Sutton, R., Burns, R.S. and Robinson, P. (2000) A Kalman Filter Approach to Fault Tolerance Control in Autonomous Underwater Vehicles. In: *Proc. 14th International Conference on Systems Engineering*, Coventry, 12–14 September, **2**, pp. 458–463.
- Phillips, C.L. and Harbor, R.D. (2000) *Feedback Control Systems*, 4th ed., Prentice-Hall, Inc., Upper Saddle River, NJ.
- Polkinghorne, M.N. (1994) A Self-Organising Fuzzy Logic Autopilot for Small Vessels, PhD Thesis, School of Manufacturing, Materials and Mechanical Engineering, University of Plymouth, UK.
- Polkinghorne, M.N., Roberts, G.N. and Burns, R.S. (1994) The Implementation of a Fuzzy Logic Marine Autopilot. In: *Proc. IEE Control 94*, Warwick, UK, March **2**, pp. 1572–1577.
- Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V. and Mishchenko, E.F. (1962) *The Mathematical Theory of Optimal Processes*, John Wiley & Sons, New York.
- Postlethwaite, I., Edward J.M. and MacFarlane, A.G.J. (1981) Principal gains and principal phases in the analysis of linear multivariable feedback systems, *IEEE Transactions on Automatic Control*, **AC-26**, pp. 32–46.
- Raven, F. (1990) *Automatic Control Engineering*, 2nd ed., McGraw-Hill, New York.
- Rechenburg, I. (1965) *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment Translation No. 1122, B.F. Toms (trans.), Farnborough, Hants, Ministry of Aviation, Royal Aircraft Establishment.
- Rezevski, G. (ed.) (1995) *Designing Intelligent Machines, Vol. 1, Perception, Cognition and Execution*, Butterworth-Heinemann (in association with The Open University), Oxford, UK.
- Riccati, J.F. (1724) Animadversiones In Aequationes Differentiales, *Acta Eruditorum Lipsiae*. Re-printed by Bittanti, S. (ed.) (1989) *Count Riccati and the Early Days of the Riccati Equation*, Pitagora Editrice, Bologna, Milano.
- Richter, R. (2000) A Predictive Fuzzy-Neural Autopilot for the Guidance of Small Motorised Marine Craft, PhD Thesis, Department of Mechanical and Marine Engineering, University of Plymouth, UK.
- Richter, R., Burns, R.S., Polkinghorne, M.N. and Nurse, P. (1997) A Predictive Ship Control using a Fuzzy-Neural Autopilot. In: *Eleventh Ship Control Systems Symposium, Southampton, UK*, 14–18 April, **1**, pp. 161–172.
- Rosenblatt, F. (1961) *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Press, Washington, DC.
- Rosenbrock, H.H. (1974) *Computer Aided Control System Design*, Academic Press, New York.
- Routh, E.J. (1905) *Dynamics of a System of Rigid Bodies*, Macmillan & Co., London.
- Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) Learning internal representations by error propagation. In: *Parallel Distributed Processing*, Rumelhart, D.E. and McClelland, J.L. (eds.), M.I.T. Press, Cambridge, Mass.
- Safanov, M.G. (1980) *Stability and Robustness of Multivariable Feedback Systems*, M.I.T. Press, Cambridge, Mass.
- Schwarzenbach, J. and Gill, K.F. (1979) *System Modelling and Control*, Edward Arnold, London.

- Shannon, C.E. and Weaver, W. (1949) *The mathematical theory of communication*, University of Illinois Press, Urbana.
- Smith, O.J.M. (1957) Closer Control of Loops with Dead Time, *Chem. Eng. Progress*, **53**(5), pp. 217–219.
- Sperry, E.A. (1922) Automatic Steering. *Trans. Soc. Naval Arch. & Marine Eng.*, **XXX**, pp.53–57.
- Sugeno, M. (ed.) (1985) *Industrial Applications of Fuzzy Control*, Elsevier Science Publishers BV, North-Holland.
- Sutton, R. and Jess, I.M. (1991) A Design Study of a Self-Organising Fuzzy Autopilot for Ship Control. In: *IMEchE, Proc. Instn. Mech. Engrs.*, **205**, pp. 35–47.
- Sutton, R. and Marsden, G.D. (1997) A Fuzzy Autopilot Optimized using a Genetic Algorithm, *Journal of Navigation*, **50**(1), pp. 120–131.
- Sutton, R., Burns, R.S. and Craven, P.J. (2000) Intelligent Steering Control of an Autonomous Underwater Vehicle, *Journal of Navigation*, **53**(3), pp. 511–525.
- The MathWorks Inc. (1993) *SIMULINK Numerical Simulation Software – Reference Guide*, The MathWorks Inc., Natick, Mass.
- Tong, R.M. (1978) Synthesis of fuzzy models for industrial processes, *Int. J. General Systems*, **4**, pp. 143–162.
- Van Dyke Parunak, H. (1990) Focus on Intelligent Control, *Inter. J. of Integrated Manufacturing*, pp. 1–5.
- Werbos, P. (1974) Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences, Thesis in Applied Mathematics, Harvard University.
- Widrow, B. (1987) The Original Adaptive Neural Net Broom-Balancer. In: *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 351–357.
- Widrow, B. and Hoff, M.E. (1960) Adaptive switching circuits. In: *IEEE WESCON Convention Record, IRE*, New York, pp. 96–104.
- Widrow, B. and Smith, F.W. (1964) Pattern recognising control systems. In: *Computer and Information Sciences*, Ton, J.T. and Wilcox, R.H. (eds.), Spartan Books, Cleaver Hume Press, pp. 288–317.
- Wiener, N. (1949) *The Extrapolation, Interpolation and Smoothing of Stationary Time Series*, John Wiley, New York.
- Yan, J., Ryan, M. and Power, J. (1994) *Using fuzzy logic – Towards intelligent systems*, Prentice-Hall International (UK), Hemel Hempstead, UK.
- Zadeh, L.A. (1965) Fuzzy Sets. In: *Information and Control*, **8**, pp. 338–353.
- Zames, G. (1966) On the Input–Output Stability of Time-Varying Non-Linear Feedback Systems. Parts I and II. *IEEE Trans. on Automat. Contr.*, **AC-11**(2 & 3), pp. 228–238, 465–476.
- Zames, G. (1981) Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms and Approximate Inverses, *IEEE Trans. on Automat. Contr.*, **AC-26**, pp. 301–320.
- Ziegler, J.G. and Nichols, N.B. (1942) *Optimum Settings for Automatic Controllers*, trans. ASME, **64**, pp. 759–768.

Index

- A/D converter 198
- Acceleration 14, 17
- Acceleration due to gravity 29
- Acceleration error coefficient 169
- Accelerator pedal 13
- Acceptable transient response in the
s-plane 122
- Accumulator(s) 198
- Acker 405
- Ackermann's formula 251, 253, 257, 259,
263, 299, 340
- Activation function 348
- Activation functions
 - hard-limiting 349
 - hyperbolic tangent 349
 - linear 349
 - sigmoid 349
- Active:
 - lead compensation 179
 - lead compensation element 196
 - phase lag compensator 189
- Actual:
 - heading 9, 101
 - rudder angle 9
 - temperature 7
 - value 10
- Actuating:
 - device 10
 - element 10
 - force 8
- Actuation subsystem 326
- Actuator saturation 91
- Adaptive linear element (ADLINE) 347
- Adaptive network based fuzzy inference
system (ANFIS) 362
- Adaptive search and genetic algorithms 326
- Addition and subtraction of matrices 426
- Additive uncertainty 303
- Address bus 199
- Adjoint matrix 242
- Adjoint of a matrix 426
- Aerodynamic drag 13
- Aerodynamic forces 8
- Ailerons 7
- Air gap flux 72
- Aircraft elevator control 7
- Airliner 12
- Aliasing 200
- Amplitude ratio 145
- Amplitudes of vibration 193
- Analogue:
 - circuits 10
 - control signal 9
 - signal 10
- Anderson 347
- Angle criterion 123
- Angle of departure 131, 142
- Angles of departure and arrival 126
- Angular:
 - displacement 33, 61
 - position 19
 - positional control system 106
 - velocity 19, 60
- Antecedent 330
- Antecedent-consequent linguistic rules 332
- Anti-aliasing filter 200
- Applied torque 19
- Approximate linear relationship 28
- Argand Diagram 148
- Argument 149–150
- Arithmetic logic unit (ALU) 198
- Armature:
 - constant 73
 - control 71
 - controlled dc servo-motor 75
 - current 72
 - excitation voltage 73
 - inductance 73
 - resistance 73
 - winding 73
- Artificial intelligence (AI) 3, 325
- Artificial neural networks (ANNs) 3, 347
- ASCII 382
- Assembler 198

- Asymptote angles 127, 131, 137
- Asymptote intersection 126, 127, 131, 137
- Asymptote intersection (break frequency) 159
- Asymptote intersection: first-order system 154
- Asymptote intersection: second-order system 155
- Asymptotes 142
- Asymptotic approximation: Bode diagrams 153
- Asymptotic construction: Bode plot 154
- Athans 3
- Augmented plant matrix $\mathbf{P}(s)$ 315
- Augtf 416
- Author 299
- Autonomous systems 325
- Autopilot 273
- Autoscale 384
- Auxiliary polynomial 117
- Average magnitude of sensitivity function 305
- Axis 390
- Axon 347

- Back emf 73
- Back-propagation 356, 362
- Back-propagation algorithm (BPA) 347, 351, 377
- Band drying oven 289
- Bandwidth 175–179, 183, 186, 188–189, 191–192, 194–197, 200, 222
- Bandwidth (ω_B) 172
- Bang-bang control 7, 273
- Barrett 373
- Bell Telephone Laboratories 2
- Bellman 3, 272
- Best estimate of measured variable 285
- Best flatband response 175–178
- Bias 348
- Bilinear transformation 222
- Bill of materials 4
- Binary word 198
- Biological neurons 347
- Black box approach 358
- Block diagram 4
 - form 63
 - manipulation 67
 - reduction 66
 - transformation theorems 67, 68
 - with interaction 68
- Blocks in cascade 207
- Bode 2
- Bode 393
- Bode diagram 151–152, 165,
 - first-order system 154, 159
 - lead compensator 185
 - pure integrator 158
 - second-order system 157, 160
 - active lead network 180
- Boltzmann constant 373
- Boolean AND function 333
- Boolean OR function 333
- Bound of multiplicative uncertainty 306
- Bound of the multiplicative model uncertainty 311, 323
- Boundary of a system 4
- Bounded inputs 304
- Branch 416
- Breakaway points 121, 126, 127, 138, 142, 218–20, 230
- Break-frequency 200
- Breeding of successive generations 365
- Bucy 3, 285
- Bulk modulus 77
- Burner 6
 - exhaust temperature $t_b(t)$ 289
 - gas supply valve angle $v_a(t)$ 289
 - model 290
- Burns 290, 299, 359

- C and C++ 199
- C2d 402
- C2dm 397
- Canonical robust control problem 314
- Capacitance 21
- Capacitive element 22, 25
- Capacity requirements planning 4
- Car cruise control system 269
- Cartesian co-ordinates 226
- Cascade 65
- Cauchy's theorem 161
- Central processing unit (CPU) 198
- Centre of area method 335
- Centrifugal speed governor 1
- Chain rule 352
- Characteristic equation 2, 50, 114–116, 120, 127–128, 131, 138, 220, 227, 230, 240–241, 260, 262, 264

- Characteristic equation: sampled-data system 215
- Chiang 3
- Chromosome 365
- C1f 383
- Clipped fuzzy output window 335–336
- Cloop 385
- Closed form 203
- Closed-loop 6
 - bandwidth 224
 - characteristic equation 114, 252, 265
 - control system 63
 - damping ratio 126
 - dynamics of observed state feedback control system 260
 - eigenvalues 249, 277, 280, 321, 323
 - eigenvalues: band dryer 293
 - equations 260
 - frequency response 172–173, 178, 186, 189, 225
 - modulus 172
 - peak magnitude M_p 194
 - peak M_p 189, 191
 - poles 118, 225, 249
 - pulse transfer function 209–210, 212, 220
 - state equations 261
 - system matrix ($\mathbf{A} - \mathbf{BK}$) 249
 - temperature control system 107
 - time-constant 83, 108
 - transfer function 63, 107, 194
 - transfer function matrix 315
 - zeros 118
- Clustering 366
- CNC machine-tool positional control 92
- Co-active ANFIS (CANFIS) 364
- Coefficient of discharge 29, 78
- Cofactor of an element 425
- Co-factors 242
- Cognition subsystem 325
- Combined transfer function 65
- Combining blocks in cascade 67
- Combining blocks in parallel 67
- Command vector $\mathbf{v}(kT)$ 282–283
- Command vector $\mathbf{v}(t)$ 281
- Compensated modulus crossover frequency 190
- Compensator 133
 - characteristics 133
 - design 133
 - design in the frequency domain 178
 - gain constant 197
- Competitive learning 347
- Complementary sensitivity function 307
 - multivariable system 316
 - $T(s)$ (classical control) 301
 - $T(s)$ (internal model control) 302
 - $T_m(j\omega)$ 309
- Complete state feedback 266
- Completing the square 53, 88
- Complex:
 - dominant loci 135
 - exponential form 145
 - frequency approach 147
 - plane 119
 - quantity 148
 - space 148
 - variables 2
 - zeros 138
- Compressibility effects 77
- Computer numerically controlled (CNC) machine tool 3, 8
- Cond 382
- Conditions of uncertainty 325
- Conformal mapping 161
- Conical pendulum 1
- Conjugate 147
- Consequent 330
- Consistent right-hand system of coordinates 101
- Constant:
 - acceleration input 42
 - amplitude harmonic forcing 193
 - multiplication 38
 - position input 41
 - velocity 14
 - velocity input 42, 107
- Constrained functional minimization 272
- Continuity equation 31, 77
- Continuous 384
 - cycling method 91
 - solution of the matrix Riccati equation 276
 - system 225
 - time solution of state equation 245
 - time state transition matrix $\phi(t)$ 269
- Control:
 - algorithm 198, 228
 - column 7, 8
 - engineering 4
 - equation 254
 - fits 32, 143

- Control: (*continued*)
 - input 4, 5
 - law 140
 - matrix 233
 - moment 9
 - of an autonomous underwater vehicles (AUV) 364
 - signal 6, 7, 9, 10
 - strategy 140
 - surfaces 7, 8
 - system 6, 10
 - system design 10
 - system designer 12
 - system toolbox 408, 417
 - variable (inverted pendulum) 339
 - variables 291
 - vector $\mathbf{u}(t)$ 248
 - weighting matrix \mathbf{R} 274
- Controllability 248
 - matrix \mathbf{M} 248, 251, 270
- Controllable 248–249
 - canonical form 238, 251, 253
 - canonical form method 250, 251, 253
- Controlled variable 4, 5, 12
- Controller 5–7, 10, 93
 - emulation (neural network) 361
 - transfer function 320
- Controlling device 10
- Conv 385
- Conventional set theory 327
- Convolution integral 38, 239, 242
- Cost function 272
- Course changing 9
- Course-keeping 9, 273
- Covariance matrix 288
 - \mathbf{P} : band dryer 299
 - $\mathbf{P}(k + 1/k + 1)$ 322
- Craven 364
- Create new model 384
- Crisp control signal 335–336
- Crisp set 327
- Critical damping 51–52, 61, 112, 193, 254
 - coefficient 51
- Crossover point 365
- Cross-sectional area 29–30
- Cross-track:
 - error 273
 - position error 299
 - velocity error 299
- Ctrb 404
- Current 4, 9, 22
- D/A converter 198
- Damped natural frequency 54, 59, 62
- Damping 15–16
 - coefficient 17, 19, 33, 61, 193, 267
 - element 16
 - ratio 49, 51–52, 55, 62, 100, 133, 144, 193, 230, 258, 378
- Dashpot 16, 92
- Data:
 - address register 198
 - bus 199
 - for Nyquist diagram 167
 - fusion 325
- dc motor 266
- dc servo-motor 9, 71, 93
- Decibels 151
- Defuzzification process 335
- Degrees-of-freedom 100
- Delay in reward parameter 346
- Delimiter 327
- Delta rule 351, 353–355, 357
- Demanded rudder angle 9
- Dendrites 347
- Derivative action time 90
- Design:
 - in the s-plane 132
 - of control systems 12
 - procedure: lag compensation 190
- Desired:
 - closed-loop characteristic equation 251
 - closed-loop eigenvalues 258
 - closed-loop poles 231, 271
 - eigenvalues 262
 - heading 9, 101
 - set of closed-loop poles 250
 - state trajectory $\mathbf{r}(t)$ 280
 - state vector: band dryer 293
 - state vector $\mathbf{r}(kT)$ 282
 - state vector $\mathbf{r}(t)$ 281
 - temperature 6–7
 - value 4, 6, 10
 - value of closed-loop eigenvalues 299
- Det (\mathbf{A}) 381
- Determinant of a matrix 425
- Determination of:
 - \mathbf{K} on root loci 126
 - points on root loci 126
 - real closed-loop pole 130
- Diagonal matrix 424

- Difference equation 202, 206, 208, 210, 212, 228–229
 - digital compensator 224
 - method 205
- Differential equation 2–3, 14, 19, 21, 26, 30, 32–34, 36, 39, 202
 - with constant coefficients 15
- Digital:
 - circuits 10
 - compensator 216, 222–223
 - compensator design 220
 - compensator design using pole placement 224
 - compensator types 221
 - control system design 198, 210
 - error signal 8
 - feedback signal 8
 - format 8
 - PID controller 221, 222
 - signal 10
- Direct comparison method 250–252, 256, 258, 265
- Direct construction method 174
- Discrete:
 - performance index 282
 - quadratic performance index 276, 281
 - reverse-time state tracking equation 281, 283
 - solution of the matrix Riccati equation 276
 - solution of the state equation 276
 - system model: band dryer 291
- Discrete-time 200
 - compensator 231
 - control matrix $\mathbf{B}(T)$ 245
 - control transition matrix $\mathbf{B}(T)$ 270
 - response 204, 210
 - solution of state equation 244
 - state transition matrix $\mathbf{A}(T)$ 245, 269
 - system 231
 - unity feedback control system 230
- Discretized universe of discourse 331
- Discriminant 50
- Disturbance:
 - attenuation 316
 - attenuation factor 316
 - effects 12
 - input 4–5
 - noise covariance matrix \mathbf{Q} 287
 - noise covariance matrix \mathbf{Q} : band dryer 297
 - rejection 303
 - transition matrix $\mathbf{C}_d(T)$ 286–287
 - transition matrix $\mathbf{C}_d(T)$: band dryer 293
 - variables 291
- Disturbances 9
- Dominant:
 - closed-loop poles 142
 - first-order response 134
- Don't care symbol '*' 370
- Drew 290
- Dreyfus 272
- Dryer:
 - clay feed-rate $f_i(t)$ 289
 - model 290
 - outlet clay moisture content $m_f(t)$ 289
 - outlet temperature $t_d(t)$ 289
- Drying oven 34
- Dummy time variable 239
- Dynamic:
 - behaviour 13–14
 - characteristics 10, 12, 14
 - programming 3, 272
 - response 10
 - system 35
- $\text{Eig}(\mathbf{A})$ 381
- Eigenvalues 255
- Elastic element 15
- Electrical:
 - networks 23
 - noise 12
 - system 13, 21, 25, 27
- Electrohydraulic servovalve 7
- Electromagnetic force 31
- Electrostatic equation 22, 26
- Elements of the Riccati matrix \mathbf{P} 280
- Elevator 7–8
 - control 7
- Eliminating:
 - a feedback loop 67
 - a forward loop 67
- Embedded microcontrollers 198
- Empirical approach 91
- Energy output 5
- Engine torque 13
- Envelope of exponential form 110
- Epoch 360
- Equating:
 - imaginary parts 128, 131, 138
 - real parts 128, 131, 138

- Equation:
 - of a circle 173–174
 - of motion 18–20
- Equivalent:
 - block diagram 67
 - damping coefficient 21
 - mass 94
 - moment of inertia 21
 - time constant 58–59
- Erasable programmable read only memory (EPROM) 198
- Error 7
 - back-propagation through plant model 361
 - coefficients 83
 - detector 106
 - signal 64
 - actuated system 6
- Estimate $\hat{\mathbf{x}}$ of actual state vector \mathbf{x} 254
- Euler-Lagrange equations 272
- Evans 2, 119
- Evolutionary process 365
- Executable machine code 198
- Exhaust pressure 76
- Experimental determination of system time constant using step response 46
- Exponential:
 - indices 242
 - matrix e^{At} 240
- External forces 17
- Eye 382

- Factored roots 39
- Feedback 385
- Feedback:
 - amplifier 2
 - control system 6, 63
 - gain matrix: band dryer 292
 - path 6
 - sensor 10
- Feedforward neural network 349
- Feel simulator 8
- Field:
 - coil 33
 - constant 74
 - control 74
 - controlled d.c. motor 33
 - controlled dc servo-motor 75–76, 104
 - current 33
- File 384

- Filtering effect 256
- Fin:
 - positional control system 32, 143
 - time constant 135
- Final:
 - lead compensator 188
 - state $\mathbf{x}(t)$ 248
 - value theorem 38, 82–84, 168, 212
- Firebrick walls 34
- First-order:
 - differential equation 15, 23, 27, 43, 60, 236
 - element 62
 - lag systems 153
 - lead system 155
 - linear system 15
 - sampled-data system 208
 - system 18
 - type zero system 169
- Fismat 422
- Fitness function 365–366, 370, 379
 - positional control system 368
- Fixed control strategy 12
- Flow:
 - gain 79
 - pressure coefficient 79
 - rate 27
- Fluid:
 - damper 33, 61, 104
 - systems 27
- Flyball governor 1
- Flywheel 19
- Foh 397
- Forms of uncertainty 299
- Forward:
 - propagation 356
 - speed 13, 28
 - speed loss 273
 - velocity 4
- Forward-path 6
 - transfer function 63, 65, 107
- Fourier 2
- Fourier's Law 25
- Free-body diagram 17–18, 20, 28, 52
- Frequency:
 - of maximum phase advance 196
 - of transient oscillation 56
- Frequency domain 2, 145
 - analysis 145, 175
 - performance specifications 172
 - specification 179, 192

- Frequency response 2
 - characteristics: lag compensator 190
 - diagram 150
 - diagrams 151
- Frictional moment 19
- Full-order:
 - observer 258
 - state observer 254–255, 260, 271
- Fully connected multilayer network 349
- Function of time 35
- Fuzzification process 331
- Fuzzy 418
- Fuzzy:
 - conditional statement 332
 - inference 332
 - input window 332
 - logic 3, 326
 - max operation 333
 - max-min inference 340
 - min operation 333
 - output window 334
 - relations 330
 - rulebase 332, 336, 374
- Fuzzy logic:
 - control (FLC) 331
 - controller 3, 373
 - toolbox 341, 417
- Fuzzy set operations:
 - complement 328, 329
 - equality 328
 - intersection 328, 329
 - union 328, 329
- Fuzzy set theory 326–327
- Fuzzy sets 326

- Gain:
 - margin (GM) 165–166, 175–178, 184, 188–189, 196
 - reduction 184
 - reduction factor 177
 - scheduling controller 299–300
- Gas:
 - burner 34, 97
 - fire 5
 - solenoid valve 6, 97
 - solenoid valve and burner 10
- Gaussian noise sequence $\mathbf{v}(k+1)$ 286–287
- Gearbox 92–93
- General:
 - form of a digital control system 220
 - performance index 273
 - second-order transfer function 40
- Generalized:
 - control problem 81
 - second-order system response to a unit step input 52
 - second-order transfer function 122
 - transfer function 238
- Generic elements 10
- Genetic:
 - algorithm (GA) 365
 - information 365
 - material 365
 - structure 365
- Getfis 422
- Global:
 - minimum energy state 373
 - optimum 365
- Goldberg 365
- Governor 2
- Gradient-descent optimization 351
- Graphical
 - addition 151
 - user interface (GUI) 384
- Grid 383
- Grimble 3
- Grossberg adaptive resonance theory (ART) 350
- Gyro-compass 9, 102

- H_∞
 - norm 3, 310
 - control problem 308
 - optimal control 306
- H_2
 - norm 310
 - optimal control problem 305
- Hall chart 174
- Hamiltonian function 272
- Hamilton–Jacobi
 - equation 275
 - partial differential equation 272
- Harmonic
 - input 145
 - response diagram 146–148
- Head 29–30
- Heading 4
 - error 273, 299
 - rate error 299

- Heat 26
 - flow 25
 - loss 7
 - sink 26–27
 - source 26
- Heave 4
- Hebb 3, 347
- Hebbian learning 347
- Help 382
- Heuristics 365
- Hidden layer 349
- High:
 - frequency noise 179
 - level language 198
 - frequency (HF) asymptote: first-order system 153
 - speed jet 7
- Hinfopt 415
- Holland 365
- Hospital operating theatres 7
- Hull 9
 - dynamics 101
 - transfer function 102
- Human vagueness 3
- Hunting 2
- Hurwitz 2, 112
 - determinants 113
- Hydraulic:
 - actuators 75
 - cylinder 8
 - damping ratio 81
 - gain 81
 - natural frequency 81
 - servomechanism 8, 106
- Hydrodynamic resistance 28

- Ideal sampling 201
- Identity matrix 425
- Imaginary axis:
 - crossover 126, 128, 131, 138
 - crossover point 142
- Imaginary part 148
- Implementation of reduced-order state
 - observer 263
- Impulse 383
- Impulse:
 - function 41
 - response of first-order system 44
- Inductance 21
- Inductive element 22

- Industrial standards rooms 7
- Infinite power series method 204
- Infinity norm 307
- Inflow 29–30
- Information technology (IT) 325
- Infrared absorption analyser 289
- Initial:
 - conditions 39
 - state $\mathbf{x}(t_0)$ 248
 - value theorem 38
- Input:
 - function 36
 - fuzzy set membership values 332
 - layer 349
 - variable 234
 - vector 233
 - weight 304
- Inputs 4
- Insertion loss 195
- Instability 2
- Instantaneous change 14
- Instruction register 198
- Integral:
 - action time 84
 - control action 170
 - squared error (ISE) 305
 - wind-up 91
- Intelligent
 - behaviour 325
 - control system 3, 325
- Internal model control (IMC) 301, 361
- International Federation of Automatic Control (IFAC) 3
- Inv(A) 381
- Inverse:
 - Laplace transforms 242
 - model 360
 - of a matrix 426
 - square law 300
 - transform 38, 54, 59, 243
 - z-transformation 204
- Inverted:
 - pendulum 347
 - pendulum system 337

- James Watt 1
- Jang 362
- Johnson 325, 337, 340

- Jury 215
 - stability test 215
 - test 218, 230
- Jury's array 216–217
- Kalman 3, 248, 276, 285
 - filter 3, 285, 287–288
 - filter design: band dryer 295
 - gain 286
 - gain matrix **K** 286–288
 - gain matrix **K**: band dryer 299
 - gain matrix **K**($k + 1$) 322
- Kang 362
- Kinematic relationships 21
- Knowledge-based systems 326
- Kohonen 3, 347
 - self-organizing map (KSOM) 350
- Lag compensator 191
- Laplace 2
 - and z-transforms 204
 - transform 36, 39, 79, 94, 102, 202
 - transform of a time derivative 37
 - transform pairs 37–38, 88
 - transforms 98, 147, 239–240
- Laser guided missile 32, 143, 182, 222
- Lead:
 - compensator 222
 - compensator design 183, 186
- Lead-lag compensation 92
- Lead-screw 9, 92–93
- Leakage:
 - coefficient 78
 - flow-rate 77
- Learning rate 353, 360, 377
- Levenberg-Marquardt back-propagation training 423
- Line of constant damping ratio 129, 132
- Linear:
 - bearings 31
 - differential equation 15, 27–28
 - displacement transducer 92
 - lumped parameter elements 15
 - phase-log frequency plot 151
 - relationship 14, 29
 - systems 2, 15
- Linear quadratic:
 - Gaussian (LQG) control scheme 288, 322
 - H_2 -optimal control 305
 - regulator (LQR) 274, 288
 - tracking problem 280
- Linearity 37
- Linearization of nonlinear functions 27
- Linearize 27
 - hydrodynamic resistance 28
- Linearized:
 - equation 31
 - outflow 86
 - relationship 30
 - spool-valve analysis 78
 - valve resistance 29
- Linearly independent 249
- Linear-quadratic-Gaussian (LQG) control 3
- Lines of constant:
 - settling time 214
 - transient frequency 214
- Linguistic label 332
- Liquid-level process control system 85
- Load:
 - damping coefficient 106
 - flow-rate 76, 79
 - moment of inertia 106
 - on hydraulic actuator 80
 - pressure 76
- Local:
 - minima 354
 - optima 365
- Log modulus plot 153
- Log modulus-log frequency plot 151
- Logarithmic decrement 56
- Logspace 393
- Low frequency (LF) asymptote 159
 - first-order system 153
- Low level language 198
- Low-cost applications 7
- Low-pass filter 179
- LQ control 3
- Lqe 410
- Lqed 411
- Lqr 408
- Lqrd 410
- Luenberger observer 254
- Lyapunov 2
- M* and *N*:
 - circles 174
 - contours 175

- Machine:
 - intelligence 325
 - table 8, 9, 93
- Magnetic compass 9
- Magnitude criterion 124, 129, 132
- Major loop 64
- Mamdani 3, 332
 - type rule 332
 - type rulebase (inverted pendulum) 340
- Mapping:
 - closed-loop poles from s to z -plane 226
 - from s to z -plane 213
 - from s to z -plane 214
- Margin 394
- Marginal stability 91, 126, 218–219, 312
- Margins of stability 164
- Mariner Hull 103, 358
- Mass 13, 15, 17, 26
- Master production schedule 3
- Material requirements planning (MRP)
 - system 3–4
- Mathematical model 4, 13–15, 21, 25, 27, 35–36, 55, 254
- MATLAB 109, 133, 161, 341
 - command window
 - control system toolbox 382
 - editor/debugger 383
 - fuzzy inference system (FIS) editor 344
 - robust control toolbox 320
 - toolboxes 380
- Matrix 424
 - of multiplicative plant uncertainty 316
 - Riccati equation 272, 276, 280, 323
 - vector difference equation 245–246
- Matrix operations: 380
 - $A + B$ 381
 - $A * B$ 381
 - $A \setminus B$ 381
 - A/B 381
 - A' 382
- Max 396
- Maximum:
 - closed-loop peak modulus M_p 195
 - closed-loop peak M_p 196–197
 - magnitude of weighted sensitivity function
 - infinity norm 306
 - phase advance ϕ_m 187, 195
 - principle 3, 272
- Max-min:
 - fuzzy inference 336
 - fuzzy reasoning 333
 - inference 337
 - inference process 333, 335
- Maxwell 2
- McCulloch 347
- Measured:
 - elevator angle 7
 - open-loop frequency response data 164
 - value 6, 10
- Measurement:
 - matrix $C(T)$ 287
 - noise covariance matrix R 287
 - noise covariance matrix R : band
 - dryer 295
 - vector $z(k+1)T$ 287
- Mechanical:
 - levers 10
 - system 13, 15–17
- Membership function μ 327
- Microcomputers 10
- Microcontroller 198
- Microprocessor control 198
- Mid-frequency (MF) asymptote 155
- Minimum energy control problem 273
- Minimum-time control problem 273
- Minor control loop 9, 64
- Minors 242
- Minors of an element 425
- Mixed-sensitivity cost function 317
- Mksys 416
- Model:
 - of a single artificial neuron 348
 - of band dryer system 290
- Modulus 145, 147–148, 150–152
 - and phase for a first-order system 149
 - and phase for a second-order system 150
 - attenuation 190–191
 - crossover frequency 182–184, 186–187, 191
- Moisture models 290
- Moment of inertia 17, 19, 33, 61, 104, 143, 267
- Momentum (learning with) 354, 360
- Momentum coefficient 355
- Morse 359
- Motor shaft 20
- Motor vehicle 13–14
- Moving a:
 - summing point ahead of a block 67–68
 - summing point beyond a block 67
 - take-off point ahead of a block 67
 - take-off point beyond a block 67

- μ -synthesis theory 3
- Multi-layer perceptron (MLP) 347, 351
- Multiple:
 - input, multiple output (MIMO) systems 232
 - loop systems 64
- Multiplication:
 - by a constant 427
 - of matrices 427
- Multiplicative uncertainty 303, 306
- Multivariable:
 - H_∞ robust control 316
 - H_2 robust control 316
 - loop shaping 317
 - robust control 314
 - systems 232, 245, 248
- Mutation rate 365

- Networks 23
- Neural network
 - state observer 358
 - toolbox 417
- Neurofuzzy control 361
- Newff 423
- Newton's second law of motion 17, 29, 52
- Ngrid 395
- Nichols 395
- Nichols chart 175–178, 196–197
- Nichols 2
- Nichols chart:
 - lead compensator 184, 188
 - uncompensated laser guided missile 183
- Nominal plant model $G_m(s)$ 300
- Non-linear 27, 232
 - function 27
- Non-singular 248–249, 258
 - matrix 426
- Non-zero determinant 248–249
- Normal cross sectional area 25
- Normalized system inputs 304
- Null matrix 424
- Number of:
 - clockwise encirclements 162
 - distinct root loci 125, 218
 - pure integrations in the open-loop transfer function 168
 - roots with positive real parts 141
- Nyquist 393
- Nyquist 2, 162
 - contour 163
 - diagram 164–168, 170, 174, 194
 - frequency 200
 - stability criterion 162, 164, 306

- Observability 248
 - matrix \mathbf{N} 248–249, 257, 270
- Observable 248
 - canonical form 257, 259
 - canonical form method 257, 259
- Observed state variables 260
- Observer:
 - design 256
 - gain matrix \mathbf{K}_e 255, 257, 271
 - transient response 256
- Obsv 404
- Offspring chromosomes 365
- Ohm's Law 22, 26
- On-governors 2
- On-off control 7
- Onset of instability 142
- Open-loop
 - characteristic equation 265
 - control system 5
 - eigenvalues 249, 258
 - frequency response data 195
 - frequency response locus 176
 - frequency response test 194
 - gain constant 83, 115, 119, 124, 142
 - poles 119–120, 127, 130, 219, 249
 - pulse transfer function 210, 212, 224, 229–231
 - reduced order characteristic equation 262
 - time constant 108
 - transfer function 63–64, 115, 166, 191, 194, 222
 - zeros 119–120, 127, 130, 144, 219
- Operating point 27
- Operational amplifier 179
- Optical pattern recognition 347
- Optimal:
 - control law 3, 272, 275, 281
 - control law: band dryer 292
 - control policy 272
 - control problem 272
 - control system 272
 - feedback matrix \mathbf{K} 321
 - filter 3
 - minimum variance filter 285
 - trajectory $\mathbf{x}(t)$ 272

- Optimum:
 - design 3
 - policy evaluation 326
- Ord2 384
- Orifice theory 78
- Oscillation 7
- Oscillatory second-order response 134
- Other applications of genetic algorithms:
 - optimal control 372
 - self-organizing fuzzy logic control 372
- Outflow 29–30
- Output:
 - equation 234–235, 237–239
 - equation (inverted pendulum) 339
 - equation: band dryer 291
 - error ($y - \hat{y}$) 254
 - layer 349
 - shaft 20
 - $y(t)$ 248
- Outputs 4
- Overall:
 - closed-loop transfer function 64, 66, 68, 95
 - differential equation 31
- Overdamped 61
- Overdamping 112
- Overshoot 57, 62 133, 137, 193

- Parabolic:
 - fitness function 378
 - function 42
- Parallel 385
- Parametric relationship 285
- Parent chromosomes 365
- Partial fraction 39, 47, 53, 88
 - expansion 211, 242
- Passive
 - electrical network 34, 60
 - lead compensation 179
 - phase lag compensator 189
- PD 133, 179
 - cascade compensation 179
 - compensator 263
 - control 92
 - controller 331
- Peak:
 - closed-loop modulus M_p 183
 - frequency (ω_p) 172
- Peak modulus (M_p) 172, 175–176, 179, 186, 188
- Percentage overshoot 104, 141, 176, 193
- Percentage overshoot of first peak 55
- Perception subsystem 325
- Perceptron 347
- Perfect set-point tracking 301–302
- Performance:
 - criterion 3, 272
 - index 272–273, 321, 351, 354
 - index (PI) table 344, 346
 - specification 10, 137
- Periodic time 96
- Phase 152
 - advance 186
 - advance ϕ_m 182
 - angle 147–150
 - lag compensation 189
 - lead compensation 179
 - lead compensation network 195
 - lead compensator 284
 - margin (PM) 165–166, 175–178, 182–184, 187–190, 194–196
 - relationship 145
 - plane 232
- Physical:
 - characteristics 13
 - laws 10
 - system 13–14
- PI 133, 170
 - control 85
 - control law 84
- Picton 325, 337, 340
- PID 97, 133, 170, 179
 - control 138
 - control action 89
 - controller 108, 289
 - controllers for closed-loop systems 178
- PIDD control 140
- Piece-wise constant function 270
- Pilot 8
- Piston 8
- Pitch 4
 - moment of inertia 32
- Pitts 347
- Plant 4–6, 10
 - state transition equation 282
- Plot 384
- Pneumatic elements 10
- Polar co-ordinates 148
 - plot 147, 150–151

- Pole:
 - angles 123
 - locations: s and z-planes 215
 - vector magnitudes 124
- Pole/zero cancellation 134
- Pole-zero format 143
- Polynomial (A) 381
- Pontryagin 3, 272
- Population of chromosomes 365
- Position error coefficient 168
- Positional:
 - feedback 106
 - servomechanism 268
- Positive feedback 70
- Potentiometer 6
- Power:
 - amplifier 93
 - series method 206
- Predictive self-organizing fuzzy logic control (PSOFLC) 364
- Pressure 27
 - difference 8
 - flow-rate characteristics for a spool-valve 80
- Price 359
- Primary feedback signal 64
- Principle of optimality 272
- Principle of Superposition 69, 71
- Printsys 383
- Probability
 - P of accepting a solution 373
 - of selection 365
 - theory 326
- Process:
 - air 289
 - plant 191
 - reaction curve 99
 - reaction method 90, 108
- Program counter 198
- Programmable read only memory (PROM) 198
- Propeller 28, 60
 - angular velocity 28
- Proper 387
- Proportional:
 - control 7, 82, 137, 142
 - control of a first-order plant 82
 - controller 133
 - controller gain 116
 - gain constant 82
 - plus derivative (PD) control 92
 - plus integral (PI) control 84
 - plus integral plus derivative (PID) control 89
- Pulse transfer function 206–207, 220
- Pure integrator 138, 158, 170
 - asymptote 170
- Quadratic performance index 274–275, 291
- Ramp 384
 - function 36, 42
 - response of first-order system 47
- Random
 - access memory (RAM) 198
 - number generator 367
- Rank 248–249
- Rank 382
- Rank of a matrix 427
- Ratio of successive peaks 99, 109
- RC network 2
- RCL network 235
- Read only memory (ROM) 198
- Readfis 422
- Real:
 - part 148
 - shift theorem 38
 - time computational complexity 331
- Rearranging summing points 67
- Recent approaches 12
- Rechenburg 365
- Recommended compensator 135
- Rectangular plot 150, 175
- Recursive:
 - discrete-time simulation 245
 - steps 246
- Reduced:
 - block diagram 69
 - matrix Riccati equation 276, 321
 - observer system 266
 - Riccati equation 278
 - second-order state observer 264
 - order observer gain matrix K_e 262
 - order state observer 254, 262, 266, 271
- Reduction gearbox 19
- Regeneration Theory 2
- Regulator 254
 - control problem 273
 - regulator problem 273

- Relationship between
 - ϕ_m and the spacing of $1/T_1$ and $1/T_2$ 182
 - frequency response and time response:
 - closed-loop systems 191
 - open-loop and closed-loop frequency response 172
- Relative modulus 156
- Relocfind 390
- Removing a block from a
 - feedback loop 67
 - forward path 67
- Repeated roots 39
- Resistance 13, 21
 - thermometer 6, 58
- Resistive element 22, 25
- Resolution 10, 198
- Reverse-time:
 - recursive process 277
 - state tracking equations 280
- Riccati matrix:
 - band dryer 292
 - P** 277, 279, 288
 - P**(kT) 282
- Richter 358, 364
- Rise time 57, 104, 141, 176
- Rlocus 389
- Robust:
 - control 3
 - control problem 299
 - control toolbox 408
 - design 3
 - multivariable control system design 316
 - performance 300, 308–309, 312–313
 - stability 300, 306–307, 311–313, 324
- Roll 4
 - dynamics 137
- Room temperature control system 6
- Root locus:
 - analysis: z-plane 218
 - asymptotes 125
 - construction rules 125
 - construction rules: z-plane 218
 - locations on real axis 126, 218
 - method 119
 - diagram 119
- Roots 381
- Roots:
 - of characteristic equation 2, 49, 51, 112, 118, 120–122, 126, 163, 218, 241–242
 - with positive real parts 114
- Rosenblatt 3, 347
- Rotation about yaw axis 101
- Rotational:
 - damper 16
 - spring 16
 - l system 17
- Roulette wheel selection 365
- Routh 2, 112
 - stability criterion 166–167
- Routh's array 113, 116, 128, 167
 - special cases: a zero in the first column 117
 - special cases: all elements in a row are zero 117
- Routh–Hurwitz stability criterion 113, 126, 141, 215
- Rudder 9–10
 - activity 273
 - angle 101
 - angle sensor 9
- Rule-of-thumb 58, 91, 200
- Rumelhart 347
- Rzevski 3
- S domain 36
 - algebraic equations 36
- Safanov 3
- Sampled-data system 204
- Sampling:
 - frequency (ω_s) 200, 214, 222, 224
 - period T 198
 - time T 201, 225
- Satellite control 273
- Save as 384
- Schema 370
- Schemata 370
- Scope 384
- Script files 382–383
- Second-order:
 - complex roots 39
 - differential equation 15, 24–25
 - element 62
 - linear system 15
 - real roots 39
 - system 18, 58, 170, 280
 - system closed-loop frequency response 172
 - transfer functions 61
 - type one system 169

- Selection of:
 - parents for mating 367
 - performance index 273
- Self-organizing fuzzy logic control (SOFLC) 344
- Semilogx 393
- Sensitivity function
 - multivariable system 315
 - $S(s)$ (classical control) 301
 - $S(s)$ (internal model control) 302
 - $S_m(j\omega)$ 308
- Sensor 5–6
 - array 325
- Series 385
- Servomechanism 7, 107
- Servomotor 9
- Set heading 9
- Set of:
 - differential equations 272
 - first-order differential equations 233, 235
- Set-point tracking 303
- Settling time 57, 96, 104, 133, 137, 141, 176, 191, 193
- Several inputs acting simultaneously 69
- Sgrid 390
- Shaft encoder 8–9
- Shannon's sampling theorem 200
- Ship 28, 60
 - autopilot 100
 - autopilot control system 9, 102
 - roll damping ratio 137
 - roll natural frequency 135
 - roll stabilization system 135, 270
 - steady-state gain 137
- Sigma 416
- Sigmoid:
 - activation function 354
 - function 352
- Signal:
 - processing 325
 - take-off point 64
- Simplified Nyquist stability criterion 164
- Simulated annealing 373
- Simulation 384
- SIMULINK 341, 380, 384
 - library browser 384
- Simultaneous equations 59
- Single input, single output (SISO) systems 232
- Single neuron summation 354
- Single-layer fully-connected recurrent network 350
- Singular 249
 - matrix 426
 - value loop shaping 315
 - values of a complex matrix 315
- Sinks 384
- Slope:
 - 12 dB per octave (–40 dB per decade) 155
 - 6 dB per octave (–20 dB per decade) 153, 158
- Small:
 - gain infinity-norm control problem 316
 - perturbation theory 28
 - perturbations 27, 79
 - perturbations of spool valve and actuator 75
- Smith 347
- Solenoid valve 31
- Solution:
 - of the state vector differential equation 239
 - space 369
- Sources 384
- Specific:
 - heat at constant pressure 26
 - inputs 304
- Specification 96
- Speed:
 - control system 104
 - of convergence 370
- Spool-valve:
 - controlled linear actuator 75–76
 - movement 7
- Spring stiffness 16
- Spring-mass-damper system 17–18, 51, 193, 234, 241–242, 245
- Square 390
- ss_g 416
- ss2tf 402
- Stability 2
 - criteria 2
 - in the frequency domain 161
 - in the z-plane 213
 - of a closed-loop system 114
 - on the Bode diagram 170–171
- Stabilize 9
- Stable and unstable time responses 111
- Stable systems 110

- Standard:
 - components 39
 - form of transfer function for a first-order system 44
 - forms of transfer functions for a second-order system 49
 - methodology 12
- Start 384
- Starting point 125, 218
- State equation 235–239, 241, 252, 233
 - (inverted pendulum) 339
- State equations:
 - band dryer 291
 - from transfer functions 238
- State feedback:
 - gain matrix 249, 251
 - gain matrix $\mathbf{K}(kT)$ 282
 - matrix 250
 - matrix (inverted pendulum) 340
 - matrix \mathbf{K} 277, 279
- State:
 - estimation 284
 - observer 254, 261
 - of a system 232–233
 - trajectory 242–244
- State variable:
 - step response 244
 - time response 243
- State variables 232–236, 238, 242, 244, 254, 291
 - inverted pendulum 339
- State vector 233
 - differential equation 233, 240
 - $\mathbf{x}(kT)$ 282
- State weighting matrix \mathbf{Q} 274
- State-space 232
 - approach 232
 - formulation 3
 - representation of controller 320
- State-transition matrix $\phi(t)$ 240
- Static characteristics 10
- Steady-state:
 - error 36, 48, 62, 81, 108, 132
 - error coefficients 168
 - gain 52
 - gain constant 44, 49, 62
 - period 36
 - response 35
 - terms 110
- Steam mass flow-rate 1
- Steering gear 9–10, 101
- Step 383
- Step:
 - function 41
 - input function 62
- Step response 137, 220
 - analysis 55
 - function 55
 - of first-order system 45
 - performance specification 57
- Stiffness 15, 33, 61
- Structured model uncertainty 303
- Subplot 397
- Sugeno 3, 362
- Summation of
 - log modulus 153
 - phase 153
 - system elements 152
- Summed squared error function 351
- Summing
 - point 6
 - point symbol 64
- Supervised
 - learning 350
 - learning in neural networks 326
- Supply pressure 76
- Survival of the fittest 365
- Sutton 3
- Symmetric matrix 425
- Symmetry of root loci 125, 218
- Synapses 347
- Synaptic reaction 347
- System 4
 - covariance matrix \mathbf{P} 286
 - dynamics 15, 35
 - frequency domain performance 189
 - inputs 236
 - matrix 233
 - outputs 236
 - performance 141
 - response 4, 36
- System type:
 - and steady-state errors 168
 - classification 83, 85, 168–169
- Systems with multiple inputs 69
- Tabu search 373
- Tachogenerator 9, 95, 104
- Takagi 362
- Temperature 5
 - control 7

- control system 97
- differential 25
- rise 26
- Term:
 - (k/k) 286
 - $(k + 1/k)$ 286
 - $(k + 1/k + 1)$ 286
- Terminal control problem 272
- Termination point 125, 218
- Tf2ss 402
- Thermal:
 - capacitance 26, 34, 97, 108
 - conductivity 25
 - dynamics of room 97
 - resistance 26, 34, 108
 - resistance of walls 98
 - systems 25
- Thermocouple 6
- Thermometer 98, 108
- Thermostatic control 7
- Thickness 25
- Third-order:
 - differential equation 15
 - linear system 15
 - transient response 59
 - type two system 169
- Three-term control action 89
- Time constant 44, 48, 59, 60, 62
- Time domain 35, 36
 - analysis 35
 - differential equations 36
 - performance specifications 172
- Time invariant 232
- Time response 35–36, 59, 60, 62, 191
 - of burner temperature: band dryer 295
 - of gas-valve angle: band dryer 294
 - of higher-order systems 58
- Time-variant 12, 232
- Tolerance band 193
- Tong 3
- Torsional spring 33, 61
- Total fitness of:
 - offsprings 367
 - population 367
- Total response 36
- Trace of a matrix 427
- Tracking
 - control problem 273
 - vector $\mathbf{s}(t)$ 281
- Train 423
- Training
 - data 358
 - file 359
- Transfer fcn 384
- Transfer function 39–40, 60, 62
 - approach 41, 63
- Transformation matrix \mathbf{T} 250
- Transient:
 - analysis of discrete systems 213
 - errors 36, 81
 - period 36, 58
 - response 35, 242
 - response of a second-order system 50
 - terms 110
- Transition matrix $\phi(s)$ 247
- Translational:
 - damper 16
 - spring 15
 - system 17
- Transportation lag 90
- Transpose of a matrix 425
- Trial point 144
- TSS_ 416
- Turbulent flow conditions 27
- Tustin 397
- Tustin's rule 222, 223
- Two degree-of-freedom IMC system 302
- Two-mass system 237
- Two-port state-space representation 314
- Type number 168
- Ultimate:
 - gain 91
 - period 91
- Undamped natural frequency 49, 52, 62, 193, 258
- Underdamped 61, 112
 - second-order system 55
- Unit circle 214, 218
 - crossover 218–219, 230
- Unit impulse 41
 - function 201
- Unit parabolic function 42
- Unit ramp 48
 - function 42, 208–247
- Unit step:
 - function 37, 41
 - input 52, 62
 - response 54, 60–61
 - response function 58

- Unity feedback:
 - computer control system 230
 - continuous control system 230
 - control system 196
- Unity gain second-order system 62
- Universe of discourse U 327, 332
- Unstable systems 110
- Unstructured model uncertainty 303
- Unsupervised
 - learning 350
 - learning in neural networks 326
- Valve 58
 - flow area 29
- Variance of the weighted mean P 285
- Variational calculus 272
- Vector 424
 - of measurements $z(k+1)T$ 286
- Vehicle 14
 - model 14
- Velocity 16
 - error coefficient 169
 - feedback 92, 106
- Vessel 9
- Voltage 22
- Volume of hydraulic fluid 76
- Volumetric
 - flow-rate 29
 - strain 77
 - stress 77
- Walking beam feedback linkage 106
- Wall 26
- Watt governor 1
- Waves 4, 9
- Weighted
 - mixed-sensitivity approach 317
 - sensitivity function 310, 312
 - summer 348
- Weights and biases 357–358, 376–377
- Werbos 347
- Wheel traction force 13
- While 396
- Who 386
- Whos 386
- Widrow 347
- Widrow-Hoff 3
- Wiener 3, 284
- Wind 4, 9
- Xlabel 393
- Yaw hydrodynamics coefficients 101
- Zadeh 3
- Zero:
 - angles 123
 - determinant 249
 - modulus crossover 183
 - steady-state error 137
 - vector magnitudes 124
 - order hold 201, 202
- Zgrid 399
- Ziegler-Nichols 99, 108
 - methods 90
- Zoh 397
- Z-plane 217, 220
 - characteristic equation 226
- Z-transform 202
 - theorems: initial value theorem, final value theorem 204
 - theorems: linearity 203