Then the system is switched into automatic mode. Digital computers are often used to replace the manual adjustment process because they can be readily coded to produce complicated functions for the start-up signals. Care must also be taken when switching from manual to automatic. For example, the integrators in electronic controllers must be provided with the proper initial conditions.

### 28.7.5 Reset Windup

In practice, all actuators and final control elements have a limited operating range. For example, a motor–amplifier combination can produce a torque proportional to the input voltage over only a limited range. No amplifier can supply an infinite current; there is a maximum current and thus a maximum torque that the system can produce. The final control elements are said to be *overdriven* when they are commanded by the controller to do something they cannot do. Since the limitations of the final control elements are ultimately due to the limited rate at which they can supply energy, it is important that all system performance specifications and controller designs be consistent with the energy-delivery capabilities of the elements to be used.

Controllers using integral action can exhibit the phenomenon called *reset windup* or *integrator buildup* when overdriven, if they are not properly designed. For a step change in set point, the proportional term responds instantly and saturates immediately if the set-point change is large enough. On the other hand, the integral term does not respond as fast, It integrates the error signal and saturates some time later if the error remains large for a long enough time. As the error decreases, the proportional term no longer causes saturation. However, the integral term continues to increase as long as the error has not changed sign, and thus the manipulated variable remains saturated. Even though the output is very near its desired value, the manipulated variable remains saturated until after the error has reversed sign. The result can be an undesirable overshoot in the response of the controlled variable.

Limits on the controller prevent the voltages from exceeding the value required to saturate the actuator, and thus protect the actuator, but they do not prevent the integral build-up that causes the overshoot. One way to prevent integrator build-up is to select the gains so that saturation will never occur. This requires knowledge of the maximum input magnitude that the system will encounter. General algorithms for doing this are not available; some methods for low-order systems are presented in Ref. 1, Chap. 7, and Ref. 2, Chap. 7. Integrator build-up is easier to prevent when using digital control; this is discussed in Section 28.10.

## 28.8 COMPENSATION AND ALTERNATIVE CONTROL STRUCTURES

A common design technique is to insert a *compensator* into the system when the PID control algorithm can be made to satisfy most but not all of the design specifications. A compensator is a device that alters the response of the controller so that the overall system will have satisfactory performance. The three categories of compensation techniques generally recognized are *series compensation, parallel* (or *feedback*) *compensation,* and *feedforward compensation.* The three structures are loosely illustrated in Fig. 28.34, where we assume the final control elements have a unity transfer function. The transfer function of the controller is $G_1(s)$. The feedback elements are represented by $H(s)$, and the compensator by $G_c(s)$. We assume that the plant is unalterable, as is usually the case in control system design. The choice of compensation structure depends on what type of specifications must be satisfied. The physical devices used as compensators are similar to the pneumatic, hydraulic, and electrical devices treated previously. Compensators can be implemented in software for digital control applications.

### 28.8.1 Series Compensation

The most commonly used series compensators are the *lead,* the *lag,* and the *lead-lag* compensators. Electrical implementations of these are shown in Fig. 28.35. Other physical implementations are available. Generally, the lead compensator improves the speed of response; the lag compensator decreases the steady-state error; and the lead-lag affects both. Graphical aids, such as the root locus and frequency response plots, are usually needed to design these compensators (Ref. 1, Chap. 8; Ref. 2, Chap. 9).

### 28.8.2 Feedback Compensation and Cascade Control

The use of a tachometer to obtain velocity feedback, as in Fig. 28.24, is a case of feedback compensation. The feedback-compensation principle of Fig. 28.3 is another. Another form is *cascade control,* in which another controller is inserted within the loop of the original control system (Fig. 28.36). The new controller can be used to achieve better control of variables within the forward path of the system. Its set point is manipulated by the first controller.

Cascade control is frequently used when the plant cannot be satisfactorily approximated with a model of second order or lower. This is because the difficulty of analysis and control increases rapidly with system order. The characteristic roots of a second-order system can easily be expressed in analytical form. This is not so for third order or higher, and few general design rules are available.

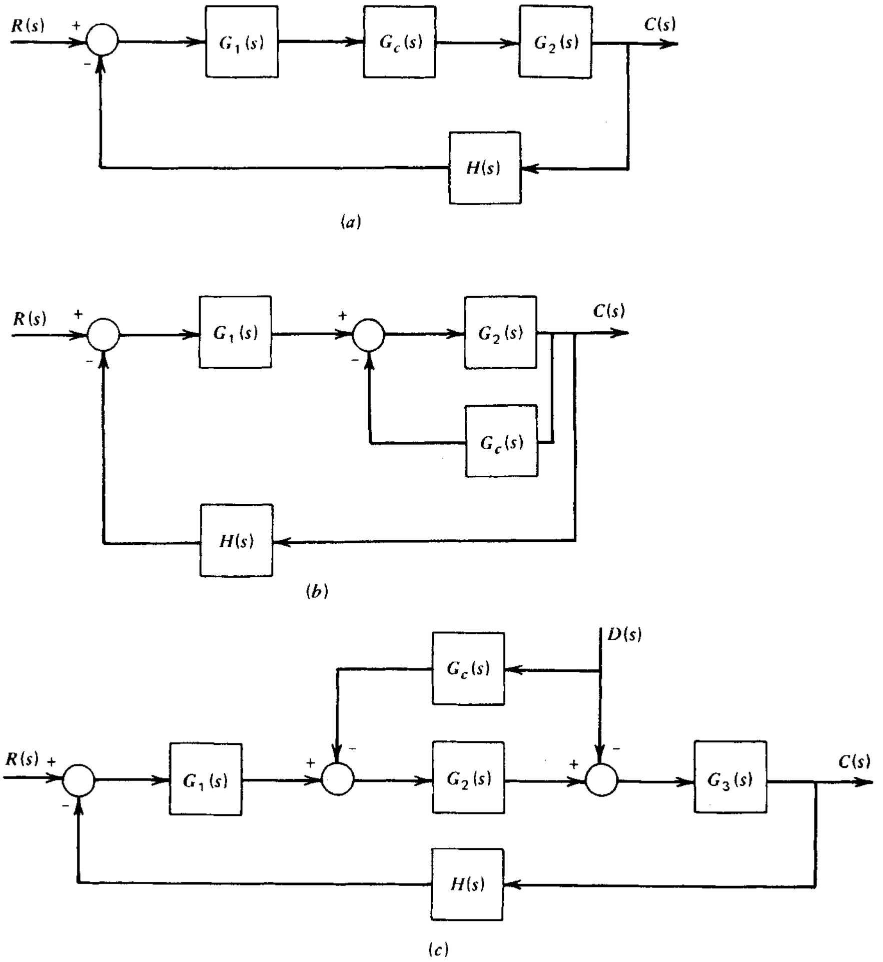


**Fig. 28.34** General structures of the three compensation types: (*a*) series; (*b*) parallel (or feedback); (c) feed-forward. The compensator transfer function is $G_c(s)$.[1]

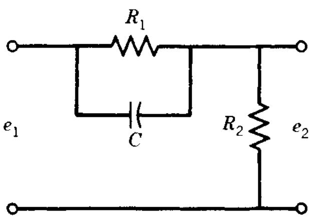
When faced with the problem of controlling a high-order system, the designer should first see if the performance requirements can be relaxed so that the system can be approximated with a low-order model. If this is not possible, the designer should attempt to divide the plant into subsystems, each of which is second order or lower. A controller is then designed for each subsystem. An application using cascade control is given in Section 28.11.

### 28.8.3 Feedforward Compensation

The control algorithms considered thus far have counteracted disturbances by using measurements of the output. One difficulty with this approach is that the effects of the disturbance must show up in the output of the plant before the controller can begin to take action. On the other hand, if we can measure the disturbance, the response of the controller can be improved by using the measurement to augment the control signal sent from the controller to the final control elements. This is the essence of feedforward compensation of the disturbance, as shown in Fig. 28.34*c*.

Feedforward compensation modified the output of the main controller. Instead of doing this by measuring the disturbance, another form of feedforward compensation utilizes the command input. Figure 28.37 is an example of this approach. The closed-loop transfer function is

$$\frac{\Omega(s)}{\Omega_r(s)} = \frac{K_f + K}{Is + c + K} \qquad (28.33)$$

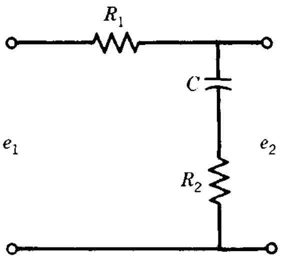


$$\frac{E_2(s)}{E_1(s)} = \frac{1}{a}\frac{1+aTs}{1+Ts}$$

$$a = \frac{R_1+R_2}{R_2}, \qquad a > 1$$

$$T = \frac{R_1R_2}{R_1+R_2}C$$

(a)

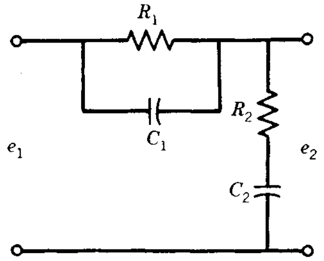


$$\frac{E_2(s)}{E_1(s)} = \frac{1+aTs}{1+Ts}$$

$$a = \frac{R_2}{R_1+R_2}, \qquad a < 1$$

$$T = C(R_1+R_2)$$

(b)



$$\frac{E_2(s)}{E_1(s)} = \frac{1+aT_1s}{1+T_1s}\frac{1+bT_2s}{1+T_2s}$$

$$aT_1 = R_1C_1, \qquad a > 1$$

$$bT_2 = R_2C_2$$

$$T_1 + T_2 = R_1C_1 + R_1C_2 + R_2C_2$$

$$b = \frac{1}{a}$$

(c)

**Fig. 28.35** Passive electrical compensators: (*a*) lead; (*b*) lag; (*c*) lead–lag.

For a unit-step input, the steady-state output is $\omega_{ss} = (K_f + K)/(c + K)$. Thus, if we choose the feedforward gain $K_f$ to be $K_f = c$, then $\omega_{ss} = 1$ as desired, and the error is zero. Note that this form of feed forward compensation does not affect the disturbance response. Its effectiveness depends on how accurately we know the value of $c$. A digital application of feedforward compensation is presented in Section 28.11.

### 28.8.4 State-Variable Feedback

There are techniques for improving system performance that do not fall entirely into one of the three compensation categories considered previously. In some forms these techniques can be viewed as a type of feedback compensation, while in other forms they constitute a modification of the control law. *State-variable feedback* (SVFB) is a technique that uses information about all the system's state variables to modify either the control signal or the actuating signal. These two forms are illustrated in Fig. 28.38. Both forms require that the state vector $x$ be measurable or at least derivable from other information. Devices or algorithms used to obtain state variable information other than directly

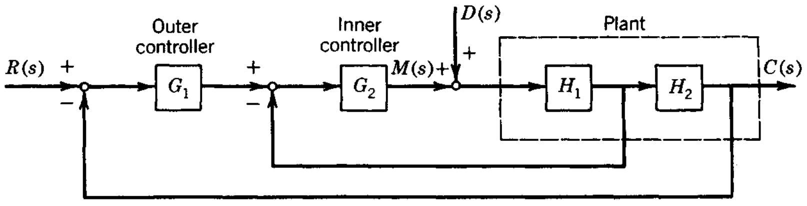


**Fig. 28.36** Cascade control structure.

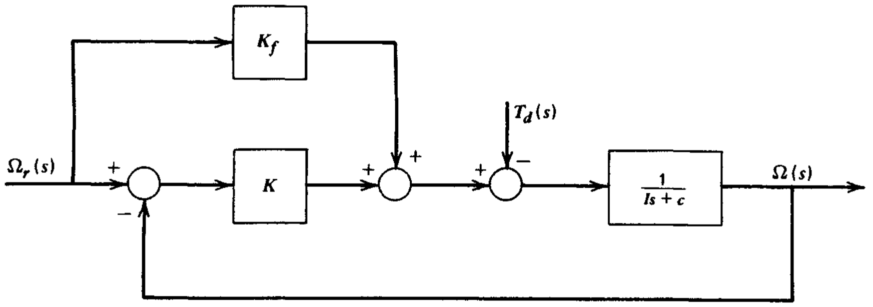


**Fig. 28.37** Feedforward compensation of the command input to augment proportional control.[2]

from measurements are variously termed *state reconstructors, estimators, observers,* or *filters* in the literature.

### 28.8.5 Pseudoderivative Feedback

*Pseudoderivative feedback* (PDF) is an extension of the velocity feedback compensation concept of Fig. 28.24.[1,2] It uses integral action in the forward path, plus an internal feedback loop whose operator $H(s)$ depends on the plant (Fig. 28.39). For $G(s) = 1/(Is + c)$, $H(s) = K_1$. For $G(s) = 1/Is^2$, $H(s) = K_1 + K_2s$. The primary advantage of PDF is that it does not need derivative action in the forward path to achieve the desired stability and damping characteristics.

## 28.9 GRAPHICAL DESIGN METHODS

Higher-order models commonly arise in control systems design. For example, integral action is often used with a second-order plant, and this produces a third-order system to be designed. Although algebraic solutions are available for third- and fourth-order polynomials, these solutions are cumbersome for design purposes. Fortunately, there exist graphical techniques to aid the designer. Frequency response plots of both the open- and closed-loop transfer functions are useful. The *Bode plot* and the *Nyquist plot* all present the frequency response information in different forms. Each form has its own advantages. The root locus plot shows the location of the characteristic roots for a range of values of some parameters, such as a controller gain. A tabulation of these plots for typical transfer functions is given in the previous chapter (Fig. 27.8). The design of two-position and other nonlinear control systems is facilitated by the *describing function,* which is a linearized approximation based on the frequency response of the controller (see Section 27.8.4). Graphical design methods are discussed in more detail in Refs. 1, 2, and 3.

### 28.9.1 The Nyquist Stability Theorem

The Nyquist stability theorem is a powerful tool for linear system analysis. If the open-loop system has no poles with positive real parts, we can concentrate our attention on the region around the point $-1 + i0$ on the polar plot of the open-loop transfer function. Figure 28.40 shows the polar plot of the open-loop transfer function of an arbitrary system that is assumed to be open-loop stable. The Nyquist stability theorem is stated as follows:

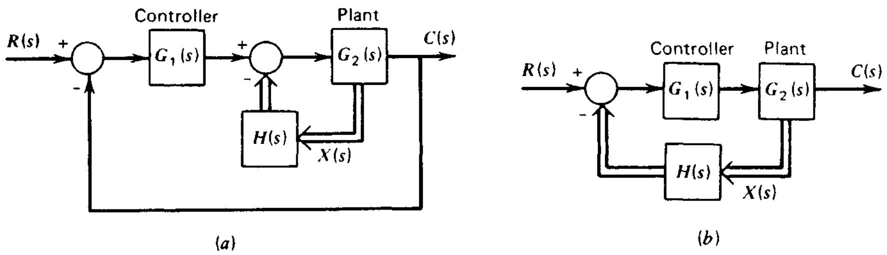


**Fig. 28.38** Two forms of state-variable feedback: (*a*) internal compensation of the control signal; (*b*) modification of the actuating signal.[1]

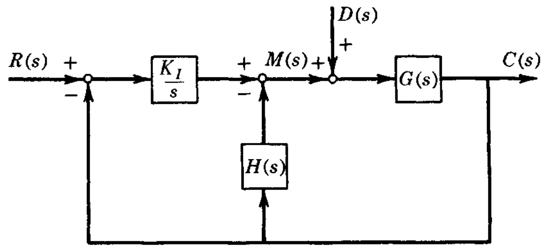


**Fig. 28.39** Structure of pseudoderivative feedback (PDF).

*A system is closed-loop stable if and only if the point $-1 + i0$ lies to the left of the open-loop Nyquist plot relative to an observer traveling along the plot in the direction of increasing frequency $\omega$.*

Therefore, the system described by Fig. 28.39 is closed-loop stable.

The Nyquist theorem provides a convenient measure of the relative stability of a system. A measure of the proximity of the plot to the $-1 + i0$ point is given by the angle between the negative real axis and a line from the origin to the point where the plot crosses the unit circle (see Fig. 28.39). The frequency corresponding to this intersection is denoted $\omega_g$. This angle is the *phase margin* (PM) and is positive when measured down from the negative real axis. The phase margin is the phase at the frequency $\omega_g$ where the magnitude ratio or "gain" of $G(i\omega)H(i\omega)$ is unity, or 0 decibels (db). The frequency $\omega_p$, the *phase crossover frequency*, is the frequency at which the phase angle is $-180°$. The *gain margin* (GM) is the difference in decibels between the unity gain condition (0 db) and the value of $|G(\omega_p)H(\omega_p)|$ db at the phase crossover frequency $\omega_p$. Thus,

$$\text{gain margin} = -|G(\omega_p)H(\omega_p)| \quad \text{(db)} \tag{28.34}$$

A system is stable only if the phase and gain margins are both positive.

The phase and gain margins can be illustrated on the Bode plots shown in Fig. 28.41. The phase and gain margins can be stated as safety margins in the design specifications. A typical set of such specifications is as follows:

$$\text{gain margin} \geq 8 \text{ db} \qquad \text{and} \qquad \text{phase margin} \geq 30° \tag{28.35}$$

In common design situations, only one of these equalities can be met, and the other margin is allowed to be greater than its minimum value. It is not desirable to make the margins too large, because this results in a low gain, which might produce sluggish response and a large steady-state error. Another commonly used set of specifications is

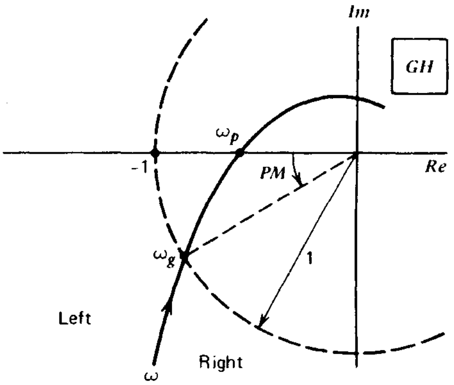


**Fig. 28.40** Nyquist plot for a stable system.[1]

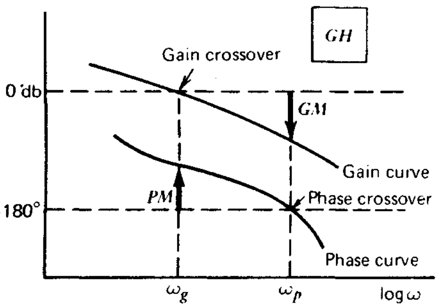


**Fig. 28.41** Bode plot showing definitions of phase and gain margin.[1]

$$\text{gain margin} \geq 6 \text{ db} \quad \text{and} \quad \text{phase margin} \geq 40° \tag{28.36}$$

The 6-db limit corresponds to the quarter amplitude decay response obtained with the gain settings given by the Ziegler–Nichols ultimate-cycle method (Table 28.2).

### 28.9.2 Systems with Dead-Time Elements

The Nyquist theorem is particularly useful for systems with dead-time elements, especially when the plant is of an order high enough to make the root-locus method cumbersome. A delay $D$ in either the manipulated variable or the measurement will result in an open-loop transfer function of the form

$$G(s)H(s) = e^{-Ds}P(s) \tag{28.37}$$

Its magnitude and phase angle are

$$|G(i\omega)H(i\omega)| = |P(i\omega)||e^{-i\omega D}| = |P(i\omega)| \tag{28.38}$$

$$\angle G(i\omega)H(i\omega)) = \angle P(i\omega) + \angle e^{-i\omega D} = \angle P(i\omega) - \omega D \tag{28.39}$$

Thus, the dead time decreases the phase angle proportionally to the frequency $\omega$, but it does not change the gain curve. This makes the analysis of its effects easier to accomplish with the open-loop frequency response plot.

### 28.9.3 Open-Loop Design for PID Control

Some general comments can be made about the effects of proportional, integral, and derivative control actions on the phase and gain margins. P action does not affect the phase curve at all and thus can be used to raise or lower the open-loop gain curve until the specifications for the gain and phase margins are satisfied. If I action or D action is included, the proportional gain is selected last. Therefore, when using this approach to the design, it is best to write the PID algorithm with the proportional gain factored out, as

$$F(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)E(s) \tag{28.40}$$

D action affects both the phase and gain curves. Therefore, the selection of the derivative gain is more difficult than the proportional gain. The increase in phase margin due to the positive phase angle introduced by D action is partly negated by the derivative gain, which reduces the gain margin. Increasing the derivative gain increases the speed of response, makes the system more stable, and allows a larger proportional gain to be used to improve the system's accuracy. However, if the phase curve is too steep near $-180°$, it is difficult to use D action to improve the performance. I action also affects both the gain and phase curves. It can be used to increase the open-loop gain at low frequencies. However, it lowers the phase crossover frequency $\omega_p$ and thus reduces some of the benefits provided by D action. If required, the D-action term is usually designed first, followed by I action and P action, respectively.

The classical design methods based on the Bode plots obviously have a large component of trial and error because usually both the phase and gain curves must be manipulated to achieve an acceptable design. Given the same set of specifications, two designers can use these methods and arrive at substantially different designs. Many rules of thumb and ad hoc procedures have been developed, but a general foolproof procedure does not exist. However, an experienced designer can often obtain a good design quickly with these techniques. The use of a computer plotting routine greatly speeds up the design process.

### 28.9.4 Design with the Root Locus

The effect of D action as a series compensator can be seen with the root locus. The term $(1 + T_D s)$ in Fig. 28.32 can be considered as a series compensator to the proportional controller. The D action adds an open-loop zero at $s = -1/T_D$. For example, a plant with the transfer function $1/s(s + 1)(s + 2)$, when subjected to proportional control, has the root locus shown in Fig. 28.42*a*. If the proportional gain is too high, the system will be unstable. The smallest achievable time constant corresponds to the root $s = -0.42$, and is $\tau = 1/0.42 = 2.4$. If D action is used to put an open-loop zero at $s = -1.5$, the resulting root locus is given by Fig. 28.42*b*. The D action prevents the system from becoming unstable, and allows a smaller time constant to be achieved ($\tau$ can be made close to $1/0.75 = 1.3$ by using a high proportional gain).

The integral action in PI control can be considered to add an open-loop pole at $s = 0$, and a zero at $s = -1/T_I$. Proportional control of the plant $1/(s + 1)(s + 2)$ gives a root locus like that shown in Fig. 28.43, with $a = 1$ and $b = 2$. A steady-state error will exist for a step input. With the PI compensator applied to this plant, the root locus is given by Fig. 28.42*b*, with $T_I = 2/3$. The steady-state error is eliminated, but the response of the system has been slowed because the dominant paths of the root locus of the compensated system lie closer to the imaginary axis than those of the uncompensated system.

As another example, let the plant-transfer function be

$$G_P(s) = \frac{1}{s^2 + a_2 s + a_1} \tag{28.41}$$

where $a_1 > 0$ and $a_2 > 0$. PI control applied to this plant gives the closed-loop command transfer function

$$T_1(s) = \frac{K_P s + K_I}{s^3 + a_2 s^2 + (a_1 + K_P)s + K_I} \tag{28.42}$$

Note that the Ziegler–Nichols rules cannot be used to set the gains $K_P$ and $K_I$. The second-order plant, Eq. (28.41), does not have the S-shaped signature of Fig. 28.33, so the process-reaction method does not apply. The ultimate-cycle method requires $K_I$ to be set to zero and the ultimate gain $K_{P_u}$

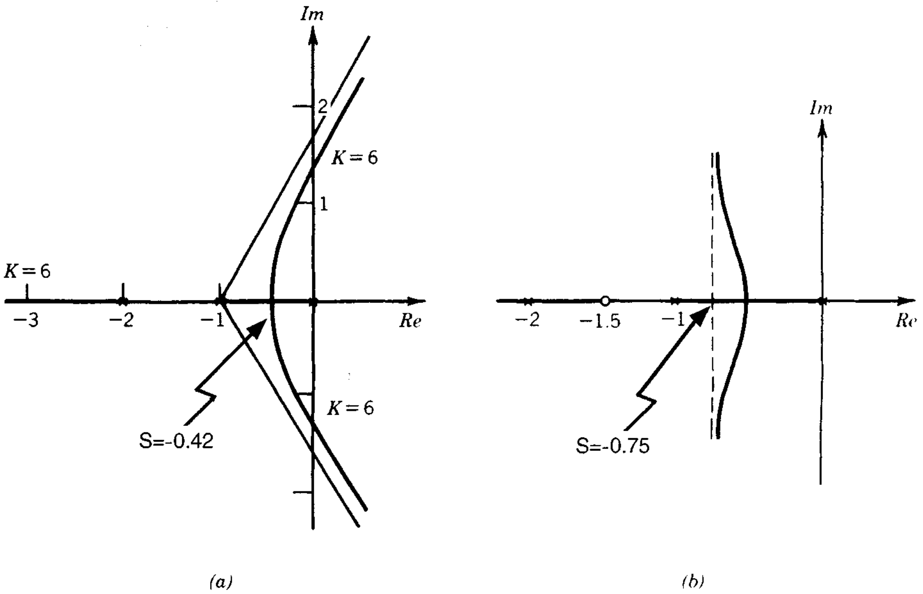


**Fig. 28.42** (*a*) Root locus plot for $s(s + 1)(s + 2) + K = 0$, for $K \geq 0$. (*b*) The effect of PD control with $T_D = 2/3$.

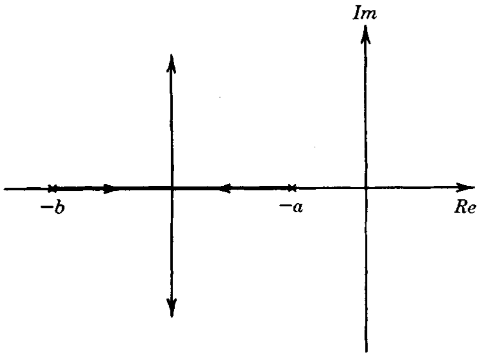


**Fig. 28.43** Root-locus plot for $(s + a)(s + b) + K = 0$.

determined. With $K_I = 0$ in Eq. (28.42) the resulting system is stable for all $K_P > 0$, and thus a positive ultimate gain does not exist.

Take the form of the PI-control law given by Eq. (28.42) with $T_D = 0$, and assume that the characteristic roots of the plant (Fig. 28.44) are real values $-r_1$ and $-r_2$ such that $-r_2 < -r_1$. In this case the open-loop transfer function of the control system is

$$G(s)H(s) = \frac{K_P(s + 1/T_I)}{s(s + r_1)(s + r_2)} \tag{28.43}$$

One design approach is to select $T_I$, and plot the locus with $K_P$ as the parameter. If the zero at $s = -1/T_I$ is located to the right of $s = -r_1$, the dominant time constant cannot be made as small as is possible with the zero located between the poles at $s = -r_1$ and $s = -r_2$ (Fig. 28.44). A large integral gain (small $T_I$ and/or large $K_P$) is desirable for reducing the overshoot due to a disturbance, but the zero should not be placed to the left of $s = -r_2$ because the dominant time constant will be larger than that obtainable with the placement shown in Fig. 28.44 for large values of $K_P$. Sketch the root-locus plots to see this. A similar situation exists if the poles of the plant are complex.

The effects of the lead compensator in terms of time-domain specifications (characteristic roots) can be shown with the root-locus plot. Consider the second-order plant with the real distinct roots $s = -\alpha$, $s = -\beta$. The root locus for this system with proportional control is shown in Fig. 28.45*a*. The smallest dominant time constant obtainable is $\tau_1$, marked in the figure. A lead compensator

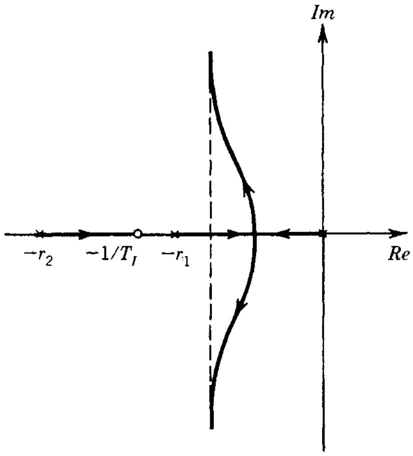


**Fig. 28.44** Root-locus plot for PI control of a second-order plant.

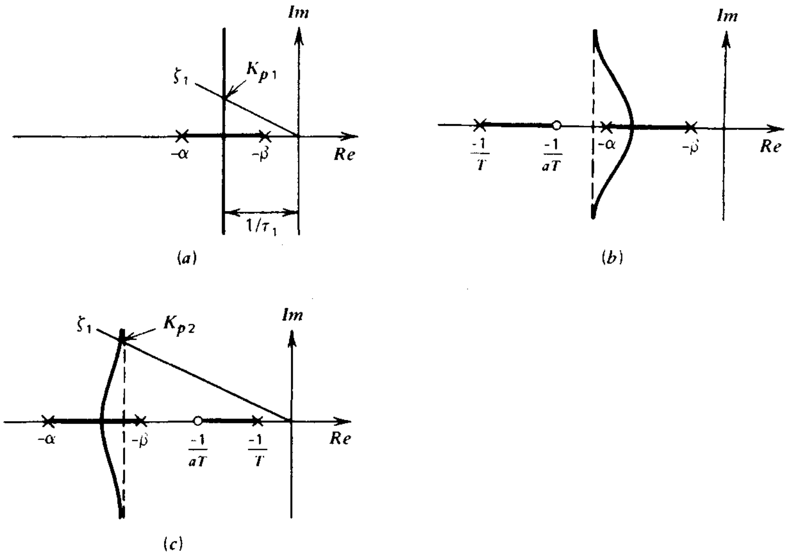


**Fig. 28.45** Effects of series lead and lag compensators: (*a*) uncompensated system's root locus; (*b*) root locus with lead compensation; (*c*) root locus with lag compensation.

introduces a pole at $s = -1/T$ and a zero at $s = -1/aT$, and the root locus becomes that shown in Fig. 28.45*b*. The pole and zero introduced by the compensator reshape the locus so that a smaller dominant time constant can be obtained. This is done by choosing the proportional gain high enough to place the roots close to the asymptotes.

With reference to the proportional control system whose root locus is shown in Fig. 28.45*a*, suppose that the desired damping ratio $\zeta_1$ and desired time constant $\tau_1$ are obtainable with a proportional gain of $K_{P1}$, but the resulting steady-state error $\alpha\beta/(\alpha\beta + K_{P1})$ due to a step input is too large. We need to increase the gain while preserving the desired damping ratio and time constant. With the lag compensator, the root locus is as shown in Fig. 28.45*c*. By considering specific numerical values, one can show that for the compensated system, roots with a damping ratio $\zeta_1$ correspond to a high value of the proportional gain. Call this value $K_{P2}$. Thus $K_{P2} > K_{P1}$, and the steady-state error will be reduced. If the value of $T$ is chosen large enough, the pole at $s = -1/T$ is approximately canceled by the zero at $s = -1/aT$, and the open-loop transfer function is given approximately by

$$G(s)H(s) = \frac{aK_P}{(s + \alpha)(s + \beta)} \tag{28.44}$$

Thus, the system's response is governed approximately by the complex roots corresponding to the gain value $K_{P2}$. By comparing Fig. 28.45*a* with 28.45*c*, we see that the compensation leaves the time constant relatively unchanged. From Eq. (28.44) it can be seen that since $a < 1$, $K_P$ can be selected as the larger value $K_{P2}$. The ratio of $K_{P1}$ to $K_{P2}$ is approximately given by the parameter $a$.

Design by pole–zero cancellation can be difficult to accomplish because a response pattern of the system is essentially ignored. The pattern corresponds to the behavior generated by the canceled pole and zero, and this response can be shown to be beyond the influence of the controller. In this example, the canceled pole gives a stable response because it lies in the left-hand plane. However, another input not modeled here, such as a disturbance, might excite the response and cause unexpected behavior. The designer should therefore proceed with caution. None of the physical parameters of the system are known exactly, so exact pole–zero cancellation is not possible. A root-locus study of the effects of parameter uncertainty and a simulation study of the response are often advised before the design is accepted as final.

## 28.10 PRINCIPLES OF DIGITAL CONTROL

Digital control has several advantages over analog devices. A greater variety of control algorithms is possible, including nonlinear algorithms and ones with time-varying coefficients. Also, greater accuracy is possible with digital systems. However, their additional hardware complexity can result in lower reliability, and their application is limited to signals whose time variation is slow enough to be handled by the samplers and the logic circuitry. This is now less of a problem because of the large increase in the speed of digital systems.

### 28.10.1 Digital Controller Structure

Sampling, discrete-time models, the $z$-transform, and pulse transfer functions were outlined in the previous chapter. The basic structure of a single-loop controller is shown in Fig. 28.46. The computer with its internal clock drives the *digital-to-analog* (D/A) and *analog-to-digital* (A/D) converters. It compares the command signals with the feedback signals and generates the control signals to be sent to the final control elements. These control signals are computed from the control algorithm stored in the memory. Slightly different structures exist, but Fig. 28.46 shows the important aspects. For example, the comparison between the command and feedback signals can be done with analog elements, and the A/D conversion made on the resulting error signal. The software must also provide for *interrupts,* which are conditions that call for the computer's attention to do something other than computing the control algorithm.

The time required for the control system to complete one loop of the algorithm is the time $T$, the *sampling time* of the control system. It depends on the time required for the computer to calculate the control algorithm, and on the time required for the interfaces to convert data. Modern systems are capable of very high rates, with sample times under 1 $\mu$s.

In most digital control applications, the plant is an analog system, but the controller is a discrete-time system. Thus, to design a digital control system, we must either model the controller as an analog system or model the plant as a discrete-time system. Each approach has its own merits, and we will examine both.

If we model the controller as an analog system, we use methods based on *differential* equations to compute the gains. However, a digital control system requires *difference* equations to describe its behavior. Thus, from a strictly mathematical point of view, the gain values we will compute will not give the predicted response exactly. However, if the sampling time is small compared to the smallest time constant in the system, then the digital system will act like an analog system, and our designs will work properly. Because most physical systems of interest have time constants greater than 1 ms, and controllers can now achieve sampling times less than 1 $\mu$s, controllers designed with analog methods will often be adequate.

### 28.10.2 Digital Forms of PID Control

There are a number of ways that PID control can be implemented in software in a digital control system, because the integral and derivative terms must be approximated with formulas chosen from a variety of available algorithms. The simplest integral approximation is to replace the integral with a sum of rectangular areas. With this rectangular approximation, the error integral is calculated as

$$\int_0^{(k+1)T} e(t)\,dt \approx Te(0) + Te(t_1) + Te(t_2) + \cdots + Te(t_k) = T\sum_{i=0}^{k} e(t_i) \tag{28.45}$$

where $t_k = kT$ and the width of each rectangle is the sampling time $T = t_{i+1} - t_i$. The times $t_i$ are the times at which the computer updates its calculation of the control algorithm after receiving an updated command signal and an updated measurement from the sensor through the A/D interfaces. If the time T is small, then the value of the sum in (28.45) is close to the value of the integral. After the control algorithm calculation is made, the calculated value of the control signal $f(t_k)$ is sent to the actuator via the output interface. This interface includes a D/A converter and a *hold* circuit that "holds" or keeps the analog voltage corresponding to the control signal applied to the actuator until

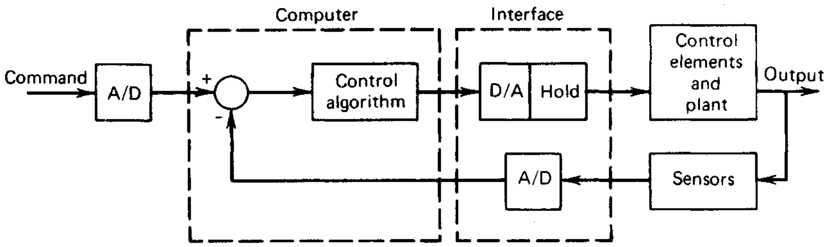


Fig. 28.46 Structure of a digital control system.[1]

the next updated value is passed along from the computer. The simplest digital form of PI control uses (28.45) for the integral term. It is

$$f(t_k) = K_P e(t_k) + K_I T \sum_{i=0}^{k} e(t_i) \tag{28.46}$$

This can be written in a more efficient form by noting that

$$f(t_{k-1}) = K_P e(t_{k-1}) + K_I T \sum_{i=0}^{k-1} e(t_i)$$

and subtracting this from (28.46) to obtain

$$f(t_k) = f(t_{k-1}) + K_P[e(t_k) - e(t_{k-1})] + K_I T e(t_k) \tag{28.47}$$

This form—called the *incremental* or *velocity* algorithm—is well suited for incremental output devices such as stepper motors. Its use also avoids the problem of integrator buildup, the condition in which the actuator saturates but the control algorithm continues to integrate the error.

The simplest approximation to the derivative is the first-order difference approximation

$$\frac{de}{dt} \approx \frac{e(t_k) - e(t_{k-1})}{T} \tag{28.48}$$

The corresponding PID approximation using the rectangular integral approximation is

$$f(t_k) = K_P e(t_k) + K_I T \sum_{i=0}^{k} e(t_i) + \frac{K_D}{\mathrm{T}} [e(t_k) - e(t_{k-1})] \tag{28.49}$$

The accuracy of the integral approximation can be improved by substituting a more sophisticated algorithm, such as the following trapezoidal rule.

$$\int_0^{(k+1)T} e(t)\, dt \approx T \sum_{i=0}^{k} \frac{1}{2} [e(t_{i+1} + e(t_i)] \tag{28.50}$$

The accuracy of the derivative approximation can be improved by using values of the sampled error signal at more instants. Using the four-point central difference method (Refs. 1 and 2), the derivative term is approximated by

$$\frac{de}{dt} \approx \frac{1}{6T} [e(t_k) + 3e(t_{k-1}) - 3e(t_{k-2}) - e(t_{k-3})]$$

The derivative action is sensitive to the resulting rapid change in the error samples that follows a step input. This effect can be eliminated by reformulating the control algorithm as follows (Refs. 1 and 2):

$$\begin{aligned} f(t_k) = {} & f(t_{k-1}) + K_P[c(t_{k-1}) - c(t_k)] \\ & + K_I T[r(t_k) - c(t_k)] \\ & + \frac{K_D}{T} [-c(t_k) + 2c(t_{k-1}) - c(t_{k-2})] \end{aligned} \tag{28.51}$$

where $r(t_k)$ is the command input and $c(t_k)$ is the variable being controlled. Because the command input $r(t_k)$ appears in this algorithm only in the integral term, we cannot apply this algorithm to PD control; that is, the integral gain $K_I$ must be nonzero.

## 28.11 UNIQUELY DIGITAL ALGORITHMS

Development of analog control algorithms was constrained by the need to design physical devices that could implement the algorithm. However, digital control algorithms simply need to be programmable, and are thus less constrained than analog algorithms.

### 28.11.1 Digital Feedforward Compensation

Classical control system design methods depend on linear models of the plant. With linearization we can obtain an approximately linear model, which is valid only over a limited operating range. Digital control now allows us to deal with nonlinear models more directly, using the concepts of feedforward compensation discussed in Section 28.8.

#### Computed Torque Method

Figure 28.47 illustrates a variation of feedforward compensation of the disturbance called the *computed torque method.* It is used to control the motion of robots. A simple model of a robot arm is the following nonlinear equation.

$$I\ddot{\theta} = T - mgL \sin \theta \tag{28.52}$$

where $\theta$ is the arm angle, $I$ is its inertia, $mg$ is its weight, $L$ is the distance from its mass center to the arm joint where the motor acts. The motor supplies the torque $T$. To position the arm at some desired angle $\theta_r$, we can use PID control on the angle error $\theta_r - \theta$. This works well if the arm angle $\theta$ is never far from the desired angle $\theta_r$ so that we can linearize the plant model about $\theta_r$. However, the controller will work for large-angle excursions if we compute the nonlinear gravity torque term $mgL \sin \theta$ and add it to the PID output. That is, part of the motor torque will be computed specifically to cancel the gravity torque, in effect producing a linear system for the PID algorithm to handle. The nonlinear torque calculations required to control multidegree-of-freedom robots are very complicated, and can be done only with a digital controller.

#### Feedforward Command Compensation

Computers can store lookup tables, which can be used to control systems that are difficult to model entirely with differential equations and analytical functions. Figure 28.48 shows a speed-control system for an internal combustion engine. The fuel flow rate required to achieve a desired speed depends in a complicated way on many variables not shown in the figure, such as temperature, humidity, and so on. This dependence can be summarized in tables stored in the control computer and can be used to estimate the required fuel flow rate. A PID algorithm can be used to adjust the estimate based on the speed error. This application is an example of feedforward compensation of the command input, and it requires a digital computer.

### 28.11.2 Control Design in the *z*-Plane

There are two common approaches to designing a digital controller:

1. The performance is specified in terms of the desired continuous-time response, and the controller design is done entirely in the *s*-plane, as with an analog controller. The resulting control law is then converted to discrete-time form, using approximations for the integral and derivative terms. This method can be successfully applied if the sampling time is small. The technique is widely used for two reasons. When existing analog controllers are converted to

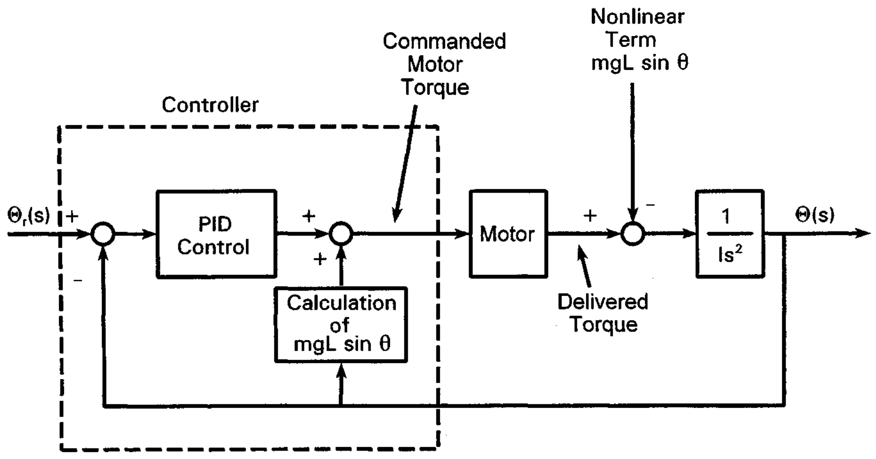


**Fig. 28.47** The computed torque method applied to robot arm control.

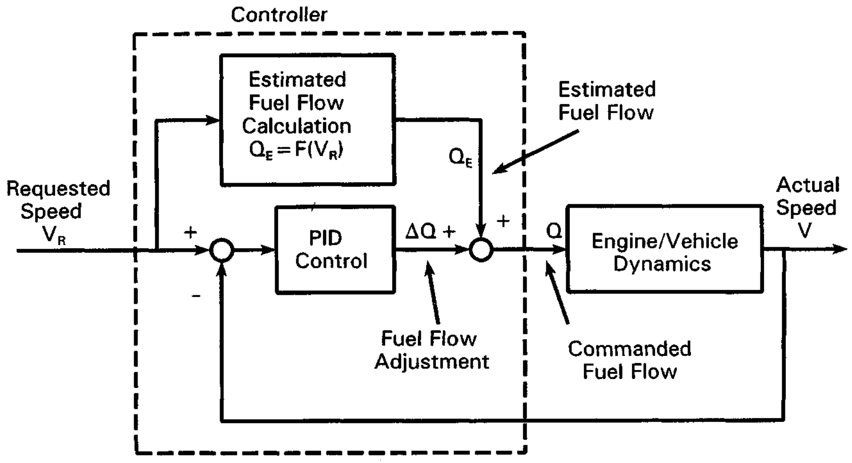


**Fig. 28.48** Feedforward compensation applied to engine control.

digital control, the form of the control law and the values of its associated gains are known to have been satisfactory. Therefore, the digital version can use the same control law and gain values. Second, because analog design methods are well established, many engineers prefer to take this route and then convert the design into a discrete-time equivalent.

2. The performance specifications are given in terms of the desired continuous-time response and/or desired root locations in the $s$-plane. From these the corresponding root locations in the $z$-plane are found and a discrete control law is designed. This method avoids the derivative and integral approximation errors that are inherent in the first method, and is the preferred method when the sampling time $T$ is large. However, the algebraic manipulations are more cumbersome.

The second approach uses the $z$-transform and pulse transfer functions, which were outlined in the previous chapter. If we have an analog model of the plant, with its transfer function $G(s)$, we can obtain its pulse transfer function $G(z)$ by finding the $z$-transform of the impulse response $g(t) = \mathcal{L}^{-1}[G(s)]$; that is, $G(z) = \mathcal{Z}[g(t)]$. Table 27.12 in the previous chapter facilitates this process; see also Refs. 1 and 2. Figure 28.49*a* shows the basic elements of a digital control system. Part (*b*) of the figure is an equivalent diagram with the analog transfer functions inserted. Part (*c*) represents the same system in terms of pulse transfer functions. From the diagram we can find the closed-loop pulse transfer function. It is

$$\frac{C(z)}{R(z)} = \frac{G(z)P(z)}{1 + G(z)P(z)} \tag{28.53}$$

The variable $z$ is related to the Laplace variable $s$ by

$$z = e^{sT} \tag{28.54}$$

If we know the desired root locations and the sampling time $T$, we can compute the $z$ roots from this equation.

### Digital PI Control Design

For example, the first-order plant $1/(2s + 1)$ with a zero-order hold has the following pulse transfer function (Refs. 1 and 2).

$$P(z) = \frac{1 - e^{-0.5T}}{z - e^{-0.5T}} \tag{28.55}$$

Suppose we use a control algorithm described by the following pulse transfer function:

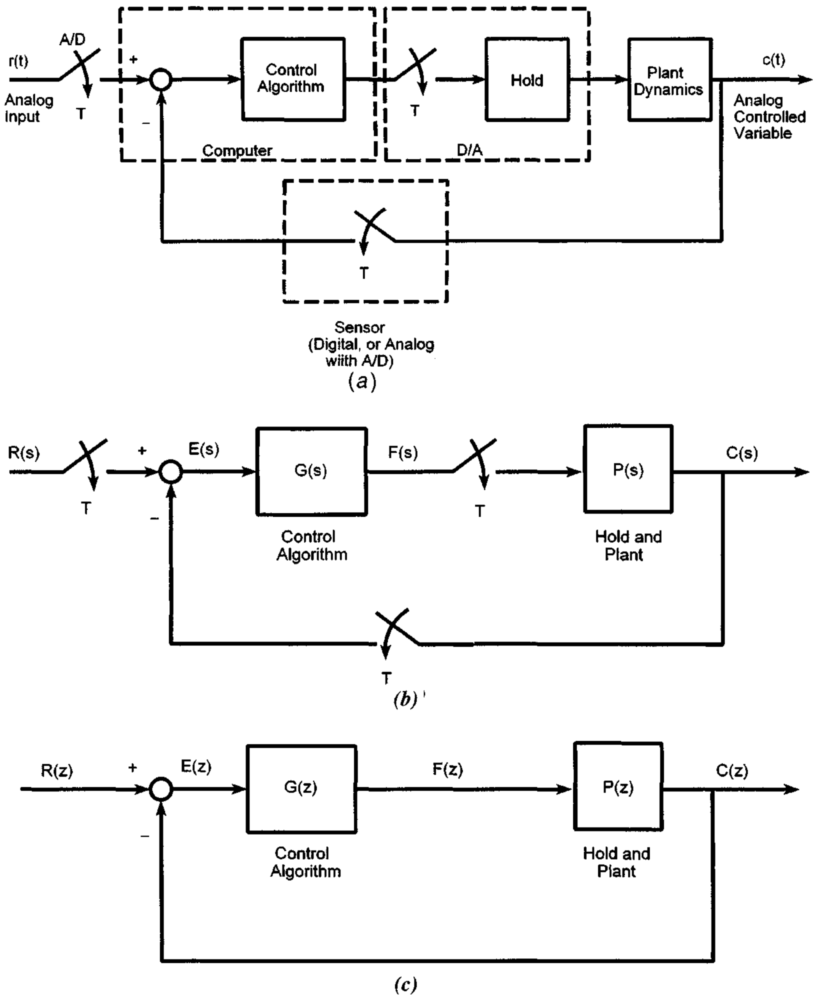


**Fig. 28.49** Block diagrams of a typical digital controller. (*a*) Diagram showing the components. (*b*) Diagram of the *s*-plane relations. (*c*) Diagram of the *z*-plane relations.

$$G(z) = \frac{F(z)}{E(z)} = \frac{K_1 z + K_2}{z - 1} = \frac{K_1 + K_2 z^{-1}}{1 - z^{-1}} \tag{28.56}$$

The corresponding difference equation that the control computer must implement is

$$f(t_k) = f(t_{k-1}) + K_1 e(t_k) + K_2 e(t_{k-1}) \tag{28.57}$$

where $e(t_k) = r(t_k) - c(t_k)$. By comparing (28.57) with (28.47), it can be seen that this is the digital equivalent of PI control, where $K_P = -K_2$ and $K_I = (K_1 + K_2)/T$. Using the form of $G(z)$ given by (28.56), the closed loop transfer function is

$$\frac{C(z)}{R(z)} = \frac{(1 - b)(K_1 z + K_2)}{z^2 + (K_1 - 1 - b - bK_1)z + b + K_2 - bK_2} \tag{28.58}$$

where $b = e^{-05T}$.

If the design specifications call for $\tau = 1$ and $\zeta = 1$, then the desired $s$ roots are $s = -1, -1$, and the analog PI gains required to achieve these roots are $K_P = 3$ and $K_I = 2$. Using a sampling time of $T = 0.1$, the $z$ roots must be $z = e^{-0.1}, e^{-0.1}$. To achieve these roots, the denominator of the transfer function (28.58) must be $z^2 - 2e^{-0.1}z + e^{-0.2}$. Thus the control gains must be $K_1 = 2.903$ and $K_2 = -2.717$. These values of $K_1$ and $K_2$ correspond to $K_P = 2.72$ and $K_I = 1.86$, which are close to the PI gains computed for an analog controller. If we had used a sampling time smaller than 0.1, say $T = 0.01$, the values of $K_P$ and $K_I$ computed from $K_1$ and $K_2$ would be $K_P = 2.97$ and $K_I = 1.98$, which are even closer to the analog gain values. This illustrates the earlier claim that analog design methods can be used when the sampling time is small enough.

### Digital Series Compensation

Series compensation can be implemented digitally by applying suitable discrete-time approximations for the derivative and integral to the model represented by the compensator's transfer function $G_c(s)$. For example, the form of a lead or a lag compensator's transfer function is

$$G_c(s) = \frac{M(s)}{F(s)} = K\frac{s + c}{s + d} \tag{28.59}$$

where $m(t)$ is the actuator command and $f(t)$ is the control signal produced by the main (PID) controller. The differential equation corresponding to (28.59) is

$$\dot{m} + dm = K(\dot{f} + cf) \tag{28.60}$$

Using the simplest approximation for the derivative, Eq. (28.48), we obtain the following difference equation that the digital compensator must implement:

$$\frac{m(t_k) - m(t_{k-1})}{T} + dm(t_k) = K\left[\frac{f(t_k) - f(t_{k-1})}{T} + cf(t_k)\right]$$

In the $z$-plane, the equation becomes

$$\frac{1 - z^{-1}}{T}M(z) + dM(z) = K\left[\frac{1 - z^{-1}}{T}F(z) + cF(z)\right] \tag{28.61}$$

The compensator's pulse transfer function is thus seen to be

$$G_c(z) = \frac{M(z)}{F(z)} = \frac{K(1 - z^{-1}) + cT}{1 - z^{-1} + dT}$$

which has the form

$$G_c(z) = K_c\frac{z + a}{z + b} \tag{28.62}$$

where $K_c$, $a$, and $b$ can be expressed in terms of $K$, $c$, $d$, and $T$ if we wish to use analog design methods to design the compensator. When using commercial controllers, the user might be required to enter the values of the gain, the pole, and the zero of the compensator. The user must ascertain whether these values should be entered as $s$-plane values (i.e., $K$, $c$, and $d$) or as $z$-plane values ($K_c$, $a$, and $b$).

Note that the digital compensator has the same number of poles and zeros as the analog compensator. This is a result of the simple approximation used for the derivative. Note that Eq. (28.61) shows that when we use this approximation, we can simply replace $s$ in the analog transfer function with $1 - z^{-1}$. Because the integration operation is the inverse of differentiation, we can replace $1/s$ with $1/(1 - z^{-1})$ when integration is used. (This is equivalent to using the rectangular approximation for the integral, and can be verified by finding the pulse transfer function of the incremental algorithm (28.47) with $K_P = 0$.)

Some commercial controllers treat the PID algorithm as a series compensator, and the user is expected to enter the controller's values not as PID gains, but as pole and zero locations in the $z$-plane. The PID transfer function is

$$\frac{F(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s \tag{28.63}$$

Making the indicated replacements for the $s$ terms, we obtain

$$\frac{F(z)}{E(z)} = K_P + \frac{K_I}{1 - z^{-1}} + K_D(1 - z^{-1})$$

which has the form

$$\frac{F(z)}{E(z)} = K_c \frac{z^2 - az + b}{z - 1} \tag{28.64}$$

where $K_c$, $a$, and $b$ can be expressed in terms of $K_P$, $K_I$, $K_D$, and $T$. Note that the algorithm has two zeros and one pole, which is fixed at $z = 1$. Sometimes the algorithm is expressed in the more general form

$$\frac{F(z)}{E(z)} = K_c \frac{z^2 - az + b}{z - c} \tag{28.65}$$

to allow the user to select the pole as well.

Digital compensator design can be done with frequency response methods, or with the root-locus plot applied to the $z$-plane rather than the $s$-plane. However, when better approximations are used for the derivative and integral, the digital series compensator will have more poles and zeros than its analog counterpart. This means that the root-locus plot will have more root paths, and the analysis will be more difficult. This topic is discussed in more detail in Refs. 1, 2, 3, and 7.

### 28.11.3 Direct Design of Digital Algorithms

Because almost any algorithm can be implemented digitally, we can specify the desired response and work backward to find the required control algorithm. This is the *direct design* method. If we let $D(z)$ be the desired form of the closed loop transfer function $C(z)/R(z)$, and solve (28.53) for the controller transfer function $G(z)$, we obtain

$$G(z) = \frac{D(z)}{P(z)[1 - D(z)]} \tag{28.66}$$

We can pick $D(z)$ directly or obtain it from the specified input transform $R(z)$ and the desired output transform $C(z)$, because $D(z) = C(z)/R(z)$.

#### Finite Settling Time Algorithm

This method can be used to design a controller to compensate for the effects of process dead time. A plant having such a response can often be approximately described by a first-order model with a dead-time element; that is,

$$G_P(s) = K\frac{e^{-Ds}}{\tau s + 1} \tag{28.67}$$

where $D$ is the dead time. This model also approximately describes the S-shaped response curve used with the Ziegler–Nichols method (Figure 28.33). When combined with a zero-order hold, this plant has the following pulse transfer function:

$$P(z) = Kz^{-n}\frac{1 - a}{z - a} \tag{28.68}$$

where $a = \exp(-T/\tau)$ and $n = D/T$. If we choose $D(z) = z^{-(n+1)}$, then with a step command input, the output $c(k)$ will reach its desired value in $n + 1$ sample times, one more than is in the dead time $D$. This is the fastest response possible. From (28.66) the required controller transfer function is

$$G(z) = \frac{1}{K(1 - a)}\frac{1 - az^{-1}}{1 - z^{-(n+1)}} \tag{28.69}$$

The corresponding difference equation that the control computer must implement is

$$f(t_k) = f(t_{k-n-1}) + \frac{1}{K(1 - a)}[e(t_k) - ae(t_{k-1})] \tag{28.70}$$

This algorithm is called a *finite settling time* algorithm, because the response reaches its desired value in a finite, prescribed time. The maximum value of the manipulated variable required by this algorithm occurs at $t = 0$ and is $1/K(1 - a)$. If this value saturates the actuator, this method will not work as predicted. Its success depends also on the accuracy of the plant model.

#### Dahlin's Algorithm

This sensitivity to plant modeling errors can be reduced by relaxing the minimum response time requirement. For example, choosing $D(z)$ to have the same form as $P(z)$, namely,

$$D(z) = K_d z^{-n} \frac{1 - a_d}{z - a_d} \tag{28.71}$$

we obtain from (28.66) the following controller transfer function:

$$G(z) = \frac{K_d(1 - a_d)}{K(1 - a)} \frac{1 - az^{-1}}{1 - a_d z^{-1} - K_d(1 - a_d)z^{-(n+1)}} \tag{28.72}$$

This is *Dahlin's algorithm*.[3] The corresponding difference equation that the control computer must implement is

$$\begin{aligned} f(t_k) &= a_d f(t_{k-1}) + K_d(1 - a_d) f(t_{k-n-1}) \\ &+ \frac{K_d(1 - a_d)}{K(1 - a)} [e(t_k) - a e(t_{k-1})] \end{aligned} \tag{28.73}$$

Normally we would first try setting $K_d = K$ and $a_d = a$, but since we might not have good estimates of $K$ and $a$, we can use $K_d$ and $a_d$ as tuning parameters to adjust the controller's performance. The constant $a_d$ is related to the time constant $\tau_d$ of the desired response: $a_d = \exp(-T/\tau_d)$. Choosing $\tau_d$ smaller gives faster response.

Algorithms such as these are often used for system startup, after which the control mode is switched to PID, which is more capable of handling disturbances.

## 28.12 HARDWARE AND SOFTWARE FOR DIGITAL CONTROL

This section provides an overview of the general categories of digital controllers that are commercially available. This is followed by a summary of the software currently available for digital control and for control system design.

### 28.12.1 Digital Control Hardware

Commercially available controllers have different capabilities, such as different speeds and operator interfaces, depending on their targeted application.

#### Programmable Logic Controllers (PLCs)

These are controllers that are programmed with relay ladder logic, which is based on Boolean algebra. Now designed around microprocessors, they are the successors to the large relay panels, mechanical counters, and drum programmers used up to the 1960s for sequencing control and control applications requiring only a finite set of output values (for example, opening and closing of valves). Some models now have the ability to perform advanced mathematical calculations required for PID control, thus allowing them to be used for modulated control as well as finite state control. There are numerous manufacturers of PLCs.

#### Digital Signal Processors (DSPs)

A modern development is the *Digital Signal Processor* (DSP), which has proved useful for feedback control as well as signal processing.[8] This special type of processor chip has separate buses for moving data and instructions, and is constructed to perform rapidly the kind of mathematical operations required for digital filtering and signal processing. The separate buses allow the data and the instructions to move in parallel rather than sequentially. Because the PID control algorithm can be written in the form of a digital filter, DSPs can also be used as controllers.

The DSP architecture was developed to handle the types of calculations required for digital filters and discrete Fourier transforms, which form the basis of most signal processing operations. DSPs usually lack the extensive memory-management capabilities of general-purpose computers, because they need not store large programs or large amounts of data. Some DSPs contain A/D and D/A converters, serial ports, timers, and other features. They are programmed with specialized software that runs on popular personal computers. Low-cost DSPs are now widely used in consumer electronics and automotive applications, with Texas Instruments being a major supplier.

### Motion Controllers

*Motion controllers* are specialized control systems that provide feedback control for one or more motors. They also provide a convenient operator interface for generating the commanded trajectories. Motion controllers are particularly well suited for applications requiring coordinated motion of two or more axes, and for applications where the commanded trajectory is complicated. A higher-level host computer might transmit required distance, speed, and acceleration rates to the motion controller, which then constructs and implements the continuous position profile required for each motor. For example, the host computer would supply the required total displacement, the acceleration and deceleration times, and the desired slew speed (the speed during the zero acceleration phase). The motion controller would generate the commanded position versus time for each motor. The motion controller also has the task of providing feedback control for each motor, to ensure that the system follows the required position profile.

Figure 28.50 shows the functional elements of a typical motion controller, such as those built by Galil Motion Control, Inc. Provision for both analog and digital input signals allows these controllers to perform other control tasks besides motion control. Compared to DSPs, such controllers generally have greater capabilities for motion control and have operator interfaces that are better suited for such applications. Motion controllers are available as plug-in cards for most computer bus types. Some are available as stand-alone units.

Motion controllers use a PID control algorithm to provide feedback control for each motor (some manufacturers call this algorithm a "filter"). The user enters the values of the PID gains (some manufacturers provide preset gain values, which can be changed; others provide tuning software that assists in selecting the proper gain values). Such controllers also have their own language for programming a variety of motion profiles and other applications.[15] For example, they provide for linear and circular interpolation for 2D coordinated motion, motion smoothing (to eliminate jerk), contouring, helical motion, and electronic gearing. The latter is a control mode that emulates mechanical gearing in software, in which one motor (the slave) is driven in proportion to the position of another motor (the master) or an encoder.

### Process Controllers

*Process controllers* are designed to handle inputs from sensors, such as thermocouples, and outputs to actuators, such as valve positioners, that are commonly found in process control applications. Figure 28.51 illustrates the input–output capabilities of a typical process controller such as those manufactured by Honeywell, which is a major supplier of such devices. This device is a stand-alone

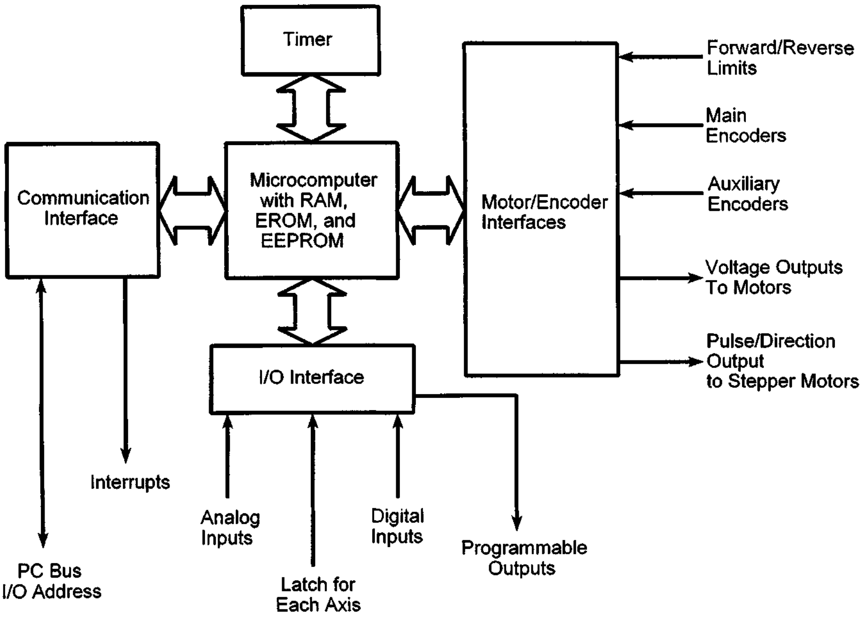


**Fig. 28.50** Functional diagram of a motion controller.

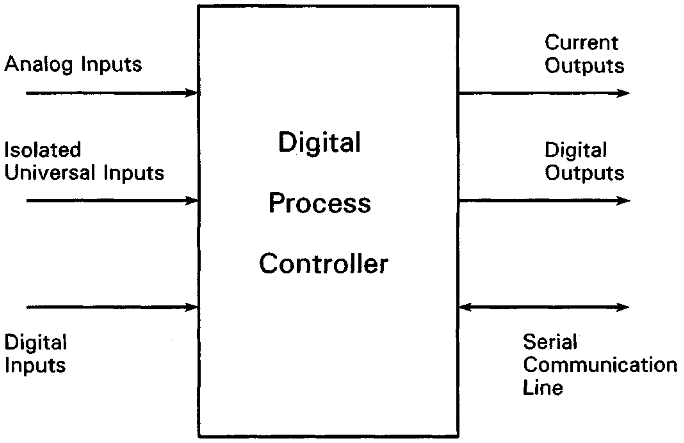

Analog Inputs
Isolated Universal Inputs
Digital Inputs
Digital Process Controller
Current Outputs
Digital Outputs
Serial Communication Line

**Fig. 28.51** Functional diagram of a digital process controller.

unit designed to be mounted in an instrumentation panel. The voltage and current ranges of the analog inputs are those normally found with thermocouple-based temperature sensors. The current outputs are designed for devices like valve positioners, which usually require 4–20 ma signals.

The controller contains a microcomputer with built-in math functions normally required for process control, such as thermocouple linearization, weighted averaging, square roots, ratio/bias calculations, and the PID control algorithm. These controllers do not have the same software and memory capabilities as desktop computers, but they are less expensive. Their operator interface consists of a small keypad with typically fewer than 10 keys, a small graphical display for displaying bargraphs of the setpoints and the process variables, indicator lights, and an alphanumeric display for programming the controller.

The PID gains are entered by the user. Some units allow multiple sets of gains to be stored; the unit can be programmed to switch between gain settings when certain conditions occur. Some controllers have an adaptive tuning feature that is supposed to adjust the gains to prevent overshoot in startup mode, to adapt to changing process dynamics, and to adapt to disturbances. However, at this time, adaptive tuning cannot claim a 100% success rate, and further research and development in adaptive control is needed.

Some process controllers have more than one PID control loop for controlling several variables. Figure 28.52 illustrates a boiler feedwater control application for a controller with two PID loops arranged in a cascade control structure. Loop 1 is the main or outer-loop controller for maintaining the desired water volume in the boiler. It uses sensing of the steam flow rate to implement feedforward compensation. Loop 2 is the inner-loop controller that directly controls the feedwater control valve.

### 28.12.2 Software for Digital Control

The software available to the modern control engineer is quite varied and powerful, and can be categorized according to the following tasks:

1. Control algorithm design, gain selection, and simulation
2. Tuning
3. Motion programming
4. Instrumentation configuration
5. Real-time control functions

Many analysis and simulation packages now contain algorithms of specific interest to control system designers. *Matlab* is one such package that is widely used. It contains built-in functions for generating root-locus and frequency-response plots, system simulation, digital filtering, calculation of control gains, and data analysis. It can accept model descriptions in the form of transfer functions or as state variable equations.

Other popular control system design and simulation packages include *Program CC*, *Matrix*$_X$, and *ACSL*. Some manufacturers provide software to assist the engineer in sizing and selecting components. An example is the *Motion Component Selector* (MCS) sold by Galil Motion Control, Inc. It

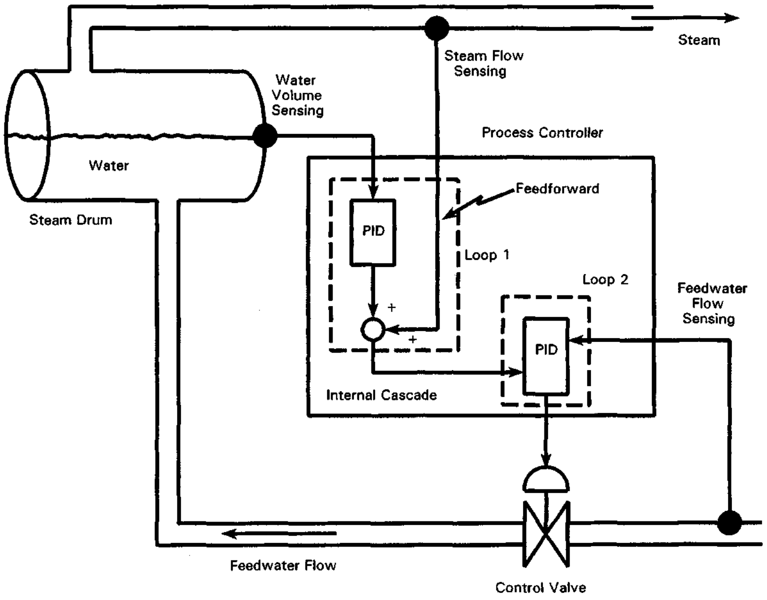


**Fig. 28.52** Application of a two-loop process controller for feedwater control.

assists the engineer in computing the load inertia, including the effects of the mechanical drive, and then selects the proper motor and amplifier based on the user's description of the desired motion profile.

Some hardware manufacturers supply software to assist the engineer in selecting control gains and modifying (*tuning*) them to achieve good response. This might require that the system to be controlled be available for experiments prior to installation. Some controllers, such as some Honeywell process controllers, have an autotuning feature that adjusts the gains in real time to improve performance.

Motion programming software supplied with motion controllers was mentioned previously. Some packages, such as Galil's, allow the user to simulate a multi-axis system having more than one motor and to display the resulting trajectory.

Instrumentation configuration software, such as *LabView* and *Dadisp,* provides specialized programming languages for interacting with instruments and for creating graphical real-time displays of instrument outputs.

Until recently, development of real-time digital control software involved tedious programming, often in assembly language. Even when implemented in a higher-level language, such as Fortran or C, programming real-time control algorithms can be very challenging, partly because of the need to provide adequately for interrupts. Software packages are now available that provide real-time control capability, usually a form of the PID algorithm, that can be programmed through user-friendly graphical interfaces. Examples include the Galil motion controllers and the add-on modules for Labview and Dadisp.

## 28.13 FUTURE TRENDS IN CONTROL SYSTEMS

Microprocessors have rejuvenated the development of controllers for mechanical systems. Currently, there are several applications areas in which new control systems are indispensable to the product's success:

1. Active vibration control
2. Noise cancellation
3. Adaptive optics
4. Robotics
5. Micromachines
6. Precision engineering

Most of the design techniques presented here comprise "classical" control methods. These methods are widely used because when they are combined with some testing and computer simulation, an experienced engineer can rapidly achieve an acceptable design. Modern control algorithms, such as state variable feedback and the linear-quadratic optimal controller, have had some significant mechanical engineering applications—for example, in the control of aerospace vehicles. The current approach to multivariable systems like the one shown in Fig. 28.53 is to use classical methods to design a controller for each subsystem, because they can often be modeled with low-order linearized models. The coordination of the various low-level controllers is a nonlinear problem. High-order, nonlinear, multivariable systems that cannot be controlled with classical methods cannot yet be handled by modern control theory in a general way, and further research is needed.

In addition to the improvements, such as lower cost, brought on by digital hardware, microprocessors have allowed designers to incorporate algorithms of much greater complexity into control systems. The following is a summary of the areas currently receiving much attention in the control systems community.

### 28.13.1 Fuzzy Logic Control

In classical set theory, an object's membership in a set is clearly defined and unambiguous. *Fuzzy logic control* is based on a generalization of classical set theory to allow objects to belong to several sets with various degrees of membership. Fuzzy logic can be used to describe processes that defy precise definition or precise measurement, and thus it can be used to model the inexact and subjective aspects of human reasoning. For example, room temperature can be described as cold, cool, just right, warm, or hot. Development of a fuzzy logic temperature controller would require the designer to specify the membership functions that describe "warm" as a function of temperature, and so on. The control logic would then be developed as a linguistic algorithm that models a human operator's decision process (for example, if the room temperature is "cold," then "greatly" increase the heater output; if the temperature is "cool," then increase the heater output "slightly").

Fuzzy logic controllers have been implemented in a number of applications. Proponents of fuzzy logic control point to its ability to convert a human operator's reasoning process into computer code. Its critics argue that because all the controller's fuzzy calculations must eventually reduce to a specific output that must be given to the actuator (e.g., a specific voltage value or a specific valve position), why not be unambiguous from the start, and define a "cool" temperature to be the range between 65° and 68°, for example? Perhaps the proper role of fuzzy logic is at the human operator interface. Research is active in this area, and the issue is not yet settled.[10,11]

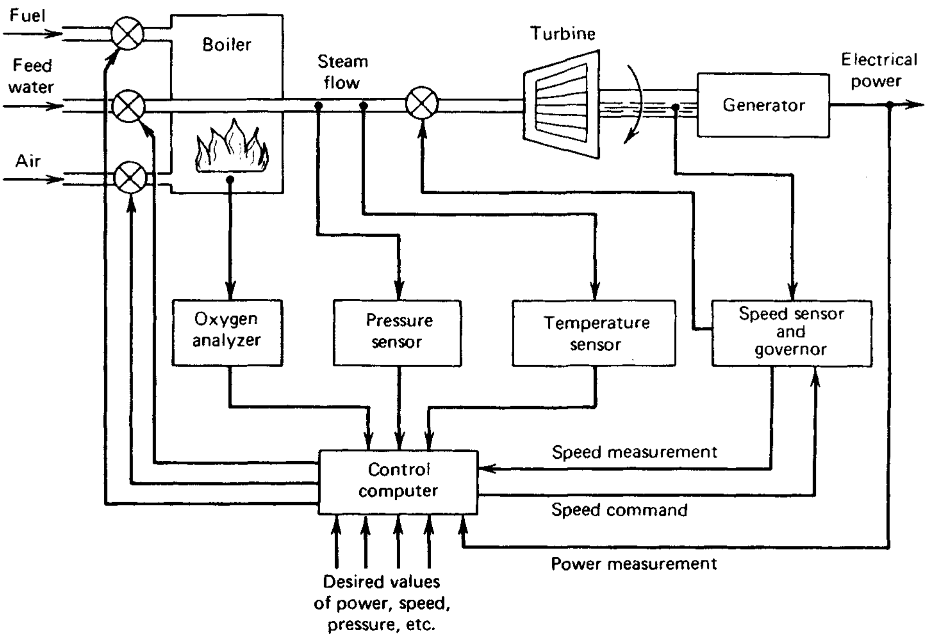


**Fig. 28.53** Computer control system for a boiler-generator. Each important variable requires its own controller. The interaction between variables calls for coordinated control of all loops.[1]

### 28.13.2 Neural Networks

Most digital computers contain a small number of sophisticated processors capable of executing large sets of instructions. In contrast, *neural networks* are devices that contain a large number of simple processors, called *nodes* or *artificial neurons.*[11,12] In conventional computers, processing and memory functions are performed by distinct elements that communicate with each other via a common bus. In neural networks, processing, memory, and communication are all handled by the nodes. This allows for faster computation, while the nonlinear behavior of the nodes provides more versatile computational capabilities.

Few neural networks have yet been implemented in hardware; most are simulations using conventional digital computers. They show promise for control systems applications requiring pattern recognition, such as image processing, tactile sensing, and nonlinear control of robot arms.

### 28.13.3 Nonlinear Control

Most real systems are nonlinear, which means that they must be described by nonlinear differential equations. Control systems designed with the linear control theory described in this chapter depend on a linearized approximation to the original nonlinear model. This linearization can be explicitly performed, or implicitly made, as when we use the small-angle approximation: $\sin\theta \approx \theta$. This approach has been enormously successful because a well-designed controller will keep the system in the operating range where the linearization was done, thus preserving the accuracy of the linear model. However, it is difficult to control some systems accurately in this way, because their operating range is too large. Robot arms are a good example.[13,14] Their equations of motion are very nonlinear, due primarily to the fact that their inertia varies greatly as their configuration changes.

Nonlinear systems encompass everything that is "not linear," and thus there is no general theory for nonlinear systems. There have been many nonlinear control methods proposed; too many to summarize here.[9] *Liapunov's stability theory* and Popov's method play a central role in many such schemes. Adaptive control is a subcase of nonlinear control (see below).

The high speeds of modern digital computers now allow us to implement nonlinear control algorithms not possible with earlier hardware. An example is the computed-torque method for controlling robot arms, which was discussed in Section 28.11 (see Fig. 28.47).

### 28.13.4 Adaptive Control

The term *adaptive control,* which unfortunately has been loosely used, describes control systems that can change the form of the control algorithm or the values of the control gains in real time, as the controller improves its internal model of the process dynamics or in response to unmodeled disturbances.[16] Constant control gains do not provide adequate response for some systems that exhibit large changes in their dynamics over their entire operating range, and some adaptive controllers use several models of the process, each of which is accurate within a certain operating range. The adaptive controller switches between gain settings that are appropriate for each operating range. Adaptive controllers are difficult to design and are prone to instability. Most existing adaptive controllers change only the gain values, and not the form of the control algorithm. Many problems remain to be solved before adaptive control theory becomes widely implemented.

### 28.13.5 Optimal Control

A rocket might be required to reach orbit using minimum fuel, or it might need to reach a given intercept point in minimum time. These are examples of potential applications of *optimal control theory.* Optimal control problems often consist of two subproblems. For the rocket example, these subproblems are (1) the determination of the minimum-fuel (or minimum-time) trajectory, and the open-loop control outputs (e.g., rocket thrust as a function of time) required to achieve the trajectory, and (2) the design of a feedback controller to keep the system near the optimal trajectory.

Many optimal control problems are nonlinear, and thus no general theory is available. Two classes of problems that have achieved some practical successes are the *bang-bang control* problem, in which the control variable switches between two fixed values (e.g., on and off, or open and closed),[5] and the *linear-quadratic-regulator* (LQG), discussed in Section 28.7, which has proven useful for high-order systems.[1,5]

Closely related to optimal control theory are methods based on stochastic process theory, including *stochastic control theory,*[17] *estimators, Kalman filters,* and *observers.*[1,5,17]

## REFERENCES

1. W. J. Palm III, *Modeling, Analysis, and Control of Dynamic Systems,* Wiley, New York, 1983.
2. W. J. Palm III, *Control Systems Engineering,* Wiley, New York, 1986.
3. D. E. Seborg, T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control,* Wiley, New York, 1989.
4. D. McCloy and H. Martin, *The Control of Fluid Power,* 2nd ed., Halsted Press, London, 1980.

5. A. E. Bryson and Y. C. Ho, *Applied Optimal Control,* Blaisdell, Waltham, MA, 1969.
6. F. Lewis, *Optimal Control,* Wiley, New York, 1986.
7. K. J. Astrom and B. Wittenmark, *Computer Controlled Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1984.
8. Y. Dote, *Servo Motor and Motion Control Using Digital Signal Processors,* Prentice-Hall, Englewood Cliffs, NJ, 1990.
9. J. Slotine and W. Li, *Applied Nonlinear Control,* Prentice-Hall, Englewood Cliffs, NJ, 1991.
10. G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic,* Prentice-Hall, Englewood Cliffs, NJ, 1995.
11. B. Kosko, *Neural Networks and Fuzzy Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1992.
12. A. Cichocki, and R. Unbehauen, *Neural Networks for Optimization and Signal Processing,* Wiley, New York, 1994.
13. J. Craig, *Introduction to Robotics,* 2nd ed., Addison-Wesley, Reading, MA, 1989.
14. M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control,* Wiley, New York, 1989.
15. J. Tal, *Step by Step Design of Motion Control Systems,* Galil Motion Control, Sunnyvale, CA, 1994.
16. K. J. Astrom, *Adaptive Control,* Addison-Wesley, Reading, MA, 1989.
17. R. Stengel, *Stochastic Optimal Control,* Wiley, New York, 1986.