$$\frac{x_o}{x_v}(s) = \frac{\frac{K_q}{A}}{s\left\{\left(\frac{mV_t}{4\beta A^2}\right)s^2 + \left(\frac{C_p + K_c}{A^2}\right)s + 1\right\}} \tag{4.57}$$

Equation (4.57) can be written in the standard form

$$\frac{x_o}{x_v}(s) = \frac{K_h}{s\left(\frac{1}{\omega_{nh}^2}s^2 + \frac{2\zeta_h}{\omega_{nh}}s + 1\right)} \tag{4.58}$$

where

$$K_h \text{ (hydraulic gain)} = \frac{K_q}{A}$$

$$\omega_{nh} \text{ (hydraulic natural frequency)} = \sqrt{\frac{4\beta A^2}{mV_t}}$$

$$\zeta_h \text{ (hydraulic damping ratio)} = \left(\frac{C_P + K_c}{2}\right)\sqrt{\frac{4\beta}{mV_t A^2}}$$

Since the Bulk Modulus of hydraulic oil is in the order of 1.4 GPa, if m and V_t are small, a large hydraulic natural frequency is possible, resulting in a rapid response. Note that the hydraulic damping ratio is governed by C_P and K_c. To control the level of damping, it is sometimes necessary to drill small holes through the piston.

4.5 Controllers for closed-loop systems

4.5.1 The generalized control problem

A generalized closed-loop control system is shown in Figure 4.22. The control problem can be stated as: 'The control action $u(t)$ will be such that the controlled output $c(t)$ will be equal to the reference input $r_1(t)$ for all values of time, irrespective of the value of the disturbance input $r_2(t)$'.

In practice, there will always be transient errors, but the transient period should be kept as small as possible. It is usually possible to design the controller so that steady-state errors are minimized, or ideally, eliminated.

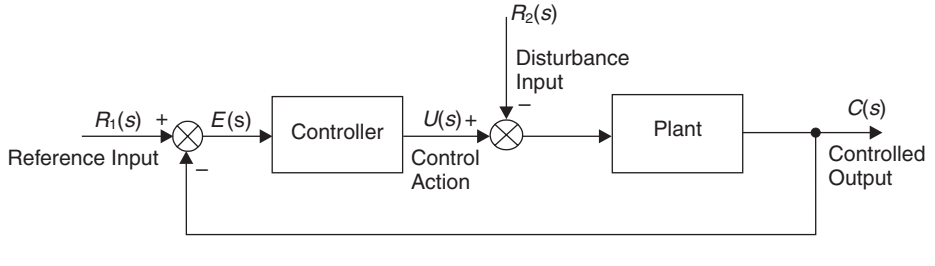


Fig. 4.22 Generalized closed-loop control system.

4.5.2 Proportional control

In this case, the control action, or signal is proportional to the error $e(t)$

$$u(t) = K_1 e(t) \tag{4.59}$$

where K_1 is the proportional gain constant.

If the plant dynamics are first-order, then Figure 4.22 can be described as shown in Figure 4.23. The plant transfer function is

$$(U(s) - R_2(s))\left(\frac{K}{1 + T\mathrm{s}}\right) = C(s) \tag{4.60}$$

And the proportional control law, from equation (4.59) becomes

$$U(s) = K_1(R_1(s) - C(s)) \tag{4.61}$$

Inserting equation (4.61) into equation (4.60) gives

$$C(s) = \frac{\{K_1(R_1(s) - C(s)) - R_2(s)\}K}{(1 + Ts)} \tag{4.62}$$

which can be written as

$$\{(1 + K_1K) + Ts\}C(s) = K_1KR_1(s) - KR_2(s) \tag{4.63}$$

Re-arranging equation (4.63) gives

$$C(s) = \frac{\left(\frac{K_1K}{1+K_1K}\right)R_1(s) - \left(\frac{K}{1+K_1K}\right)R_2(s)}{\left\{1 + \left(\frac{T}{1+K_1K}\right)s\right\}} \tag{4.64}$$

When $r_1(t)$ is a unit step, and $r_2(t)$ is zero, the final value theorem (equation (3.10)) gives the steady-state response

$$c(t) = \left(\frac{K_1K}{1 + K_1K}\right) \quad \text{as } t \to \infty.$$

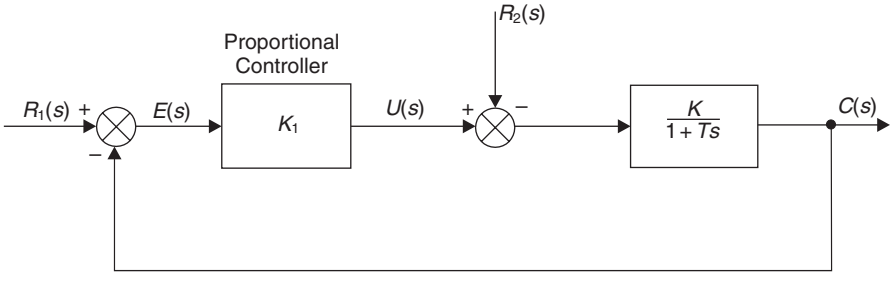


Fig. 4.23 Proportional control of a first-order plant.

When $r_2(t)$ is a unit step, and $r_1(t)$ is zero, the final value theorem (equation (3.10)) gives the steady-state response

$$c(t) = -\left(\frac{K}{1 + K_1 K}\right) \quad \text{as } t \to \infty.$$

Hence, for the system to have zero steady-state error, the terms in equation (4.64) should be

$$\left(\frac{K_1 K}{1 + K_1 K}\right) = 1$$
$$\left(\frac{K}{1 + K_1 K}\right) = 0 \tag{4.65}$$

This can only happen if the open-loop gain constant $K_1 K$ is infinite. In practice this is not possible and therefore the proportional control system proposed in Figure 4.23 will always produce steady-state errors. These can be minimized by keeping the open-loop gain constant $K_1 K$ as high as possible.

Since the closed-loop time-constant form equation (4.64) is

$$T_c = \left(\frac{T}{1 + K_1 K}\right) \tag{4.66}$$

Then maintaining $K_1 K$ at a high value will reduce the closed-loop time constant and therefore improve the system transient response.

This is illustrated in Figure 4.24 which shows a step change in $r_1(t)$ followed by a step change in $r_2(t)$.

Summary

For a first-order plant, proportional control will always produce steady-state errors. This is discussed in more detail in Chapter 6 under 'system type classification' where equations (6.63)–(6.65) define a set of error coefficients. Increasing the open-loop

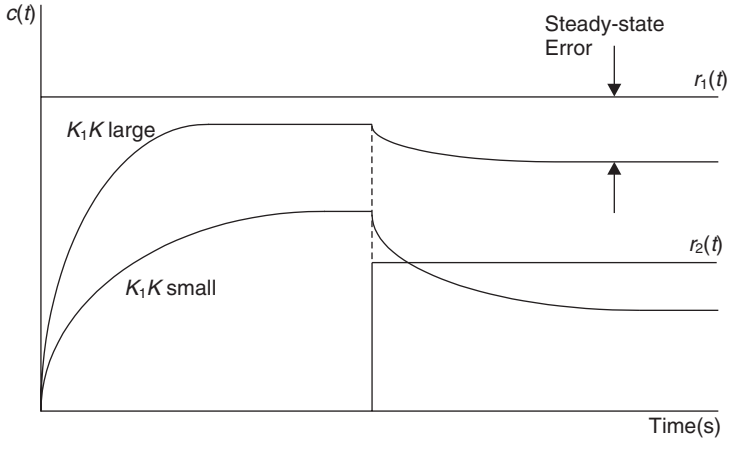


Fig. 4.24 Step response of a first-order plant using proportional control.

gain constant (which is usually achieved by increasing the controller gain K_1) will reduce, but not eliminate them. A high controller gain will also reduce the transient period. However, as will be shown in Chapters 5 and 6, high open-loop gain constants can result in the instability of higher-order plant transfer functions.

4.5.3 Proportional plus Integral (PI) control

Including a term that is a function of the integral of the error can, with the type of plant shown in Figure 4.23, eliminate steady-state errors.

Consider a control law of the form

$$u(t) = K_1 e(t) + K_2 \int e \mathrm{d}t \tag{4.67}$$

Taking Laplace transforms

$$\begin{aligned} U(s) &= \left(K_1 + \frac{K_2}{s}\right) E(s) \\ &= K_1 \left(1 + \frac{K_2}{K_1 s}\right) E(s) \\ &= K_1 \left(1 + \frac{1}{T_\mathrm{i} s}\right) E(s) \end{aligned} \tag{4.68}$$

In equation (4.68), T_i is called the integral action time, and is formally defined as: 'The time interval in which the part of the control signal due to integral action increases by an amount equal to the part of the control signal due to proportional action when the error is unchanging'. (BS 1523).

Inserting the PI control law given in equation (4.68) into the first-order plant transfer function shown in equation (4.60) gives

$$C(s) = \frac{(K_1(1 + 1/T_\mathrm{i} s)(R_1(s) - C(s)) - R_2(s))K}{(1 + Ts)} \tag{4.69}$$

which can be written as

$$\{T_\mathrm{i} T s^2 + T_\mathrm{i}(1 + K_1 K)_s + K_1 K\} C(s) = K_1 K(1 + T_\mathrm{i} s) R_1(s) - K_1 K T_\mathrm{i} s R_2(s) \tag{4.70}$$

Re-arranging gives

$$C(s) = \frac{(1 + T_\mathrm{i} s) R_1(s) - T_\mathrm{i} s R_2(s)}{\left(\frac{T_\mathrm{i} T}{K_1 K}\right) s^2 + T_\mathrm{i}\left(1 + \frac{1}{K_1 K}\right) s + 1} \tag{4.71}$$

The denominator is now in the standard second-order system form of equation (3.42). The steady-state response may be obtained using the final value theorem given in equation (3.10).

$$c(t) = (1 + 0) r_1(t) - (0) r_2(t) \quad \text{as } t \to \infty \tag{4.72}$$

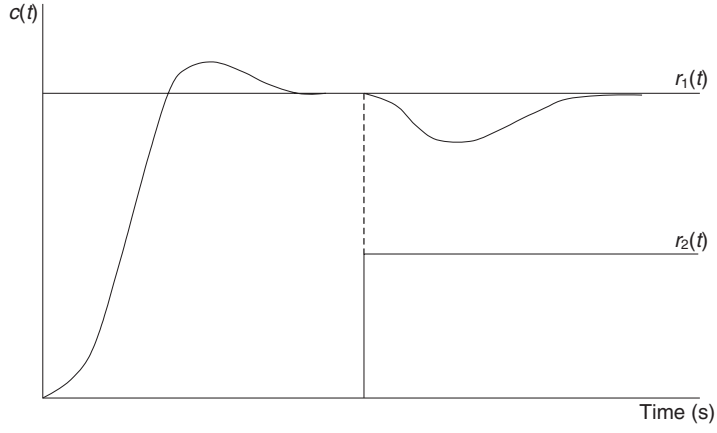


Fig. 4.25 Step response of a first-order plant using PI control.

When there are step changes in $r_1(t)$ and $r_2(t)$:

$$\begin{aligned} C(s) &= \frac{(1+0)sR_1(s)}{s} - (0)\frac{sR_2(s)}{s} \\ &= R_1(s) \\ c(t) &= r_1(t) \end{aligned} \tag{4.73}$$

Thus, when $r_1(t)$ and $r_2(t)$ are unchanging, or have step changes, there are no steady-state errors as can be seen in Figure 4.25. The second-order dynamics of the closed-loop system depend upon the values of T_i, T, K_1 and K. Again, a high value of K_1 will provide a fast transient response since it increases the undamped natural frequency, but with higher order plant transfer functions can give rise to instability.

Summary

For a first-order plant, PI control will produce a second-order response. There will be zero steady-state errors if the reference and disturbance inputs $r_1(t)$ and $r_2(t)$ are either unchanging or have step changes. The process of including an integrator within the control loop to reduce or eliminate steady-state errors is discussed in more detail in Chapter 6 under 'system type classification'.

Example 4.5 (See also Appendix 1, *examp45.m*)
A liquid-level process control system is shown in Figure 4.26. The system parameters are

$$A = 2\,\text{m}^2 \qquad R_f = 15\,\text{s/m}^2$$
$$H_1 = 1\,\text{V/m} \quad K_v = 0.1\,\text{m}^3/\text{sV} \quad K_1 = 1 \quad \text{(controller again)}$$

(a) What are the values of T_i and ζ when the undamped natural frequency ω_n is 0.1 rad/s?
(b) Find an expression for the time response of the system when there is a step change of $h_d(t)$ from 0 to 4 m. Assume zero initial conditions.

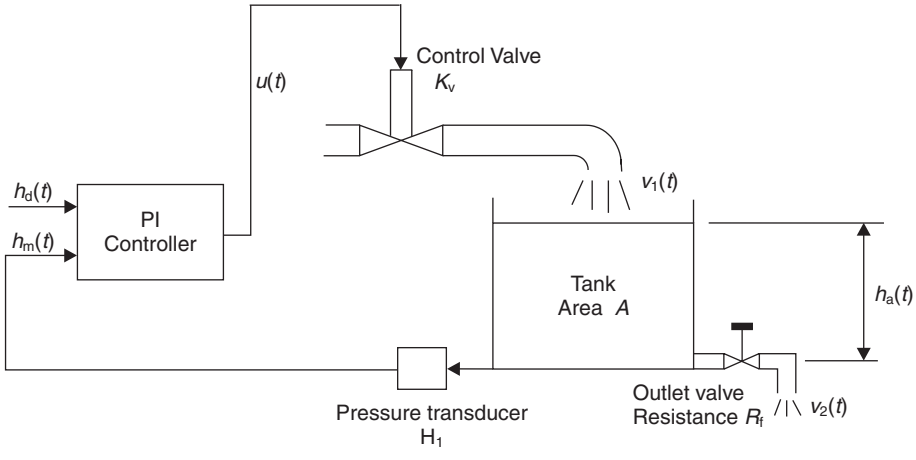


Fig. 4.26 Liquid-level process control system.

The controller is given in equation (4.68). The inflow to the tank is

$$v_1(t) = K_v u(t) \tag{4.74}$$

The tank dynamics are expressed, using equation (2.63) as

$$v_1(t) - v_2(t) = A\frac{\mathrm{d}h_a}{\mathrm{d}t} \tag{4.75}$$

and the linearized outflow is

$$v_2(t) = \frac{h_a(t)}{R_f} \tag{4.76}$$

The measured head $h_m(t)$ is obtained from the pressure transducer

$$h_m(t) = H_1 h_a(t) \tag{4.77}$$

From equations (4.75) and (4.76), the tank and outflow valve transfer function is

$$\frac{H_a}{V_1}(s) = \frac{R_f}{1 + AR_f s} \tag{4.78}$$

The block diagram for the control system is shown in Figure 4.27. From the block diagram, the forward-path transfer function $G(s)$ is

$$\begin{aligned} G(s) &= \frac{K_1 K_v R_f \left(1 + \frac{1}{T_i s}\right)}{(1 + AR_f s)} \\ &= \frac{K_1 K_v R_f (1 + T_i s)}{T_i s(1 + AR_f)} \end{aligned} \tag{4.79}$$

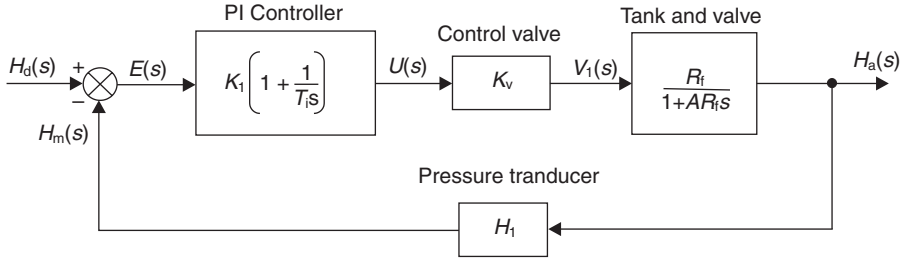


Fig. 4.27 Block diagram for liquid-level process control system.

Using equation (4.4), the closed-loop transfer function becomes

$$\frac{H_a}{H_d}(s) = \frac{\frac{K_1 K_v R_f (1+T_i s)}{(A R_f T_i s^2 + T_i s)}}{1 + \frac{K_1 K_v R_f H_1 (1+T_i s)}{(A R_f T_i s^2 + T_i s)}} \tag{4.80}$$

which simplifies to

$$\frac{H_a}{H_d}(s) = \frac{K_1 K_v R_f (1 + T_i s)}{(A R_f T_i) s^2 + T_i (1 + K_1 K_v R_f H_1) s + K_1 K_v R_f H_1} \tag{4.81}$$

Equation (4.81) can be expressed in the standard form of equation (3.42) for a second-order system.

Putting $H_1 = 1$, then

$$\frac{H_a}{H_d}(s) = \frac{(1 + T_i s)}{\left(\frac{A T_i}{K_1 K_v}\right) s^2 + T_i \left(\frac{1}{K_1 K_v R_f} + 1\right) s + 1} \tag{4.82}$$

(a) Comparing the denominator terms with the standard form given in equation (3.42)

$$\left(\frac{A T_i}{K_1 K_v}\right) = \frac{1}{\omega_n^2} \tag{4.83}$$

$$T_i \left(\frac{1}{K_1 K_v R_f} + 1\right) = \frac{2\zeta}{\omega_n} \tag{4.84}$$

From equation (4.83)

$$T_i = \frac{K_1 K_v}{\omega_n^2 A} = \frac{1 \times 0.1}{0.1^2 \times 2} = 5 \text{ seconds}$$

From equation (4.84)

$$\zeta = \frac{\omega_n T_i}{2}\left(\frac{1}{K_1 K_v R_f} + 1\right)$$
$$= \frac{0.1 \times 5}{2}\left(\frac{1}{1 \times 0.1 \times 15} + 1\right) = 0.417$$

(b) Inserting values into equation (4.82)

$$\frac{H_a}{H_d}(s) = \frac{(1+5s)}{100s^2 + 8.34s + 1} \tag{4.85}$$

For a step input of height 4 m

$$H_a(s) = \left[\frac{0.01(1+5s)}{s^2 + 0.0834s + 0.01}\right]\frac{4}{s}$$

Expanding by partial fractions using 3.2.4 (iv)

$$H_a(s) = \frac{0.04 + 0.2s}{s(s^2 + 0.0834s + 0.01)} = \frac{A}{s} + \frac{Bs + C}{s^2 + 0.0834s + 0.01} \tag{4.86}$$

Multiplying through by $s(s^2 + 0.0834s + 0.01)$

$$0.04 + 0.2s = A(s^2 + 0.0834s + 0.01) + Bs^2 + Cs$$

Equating coefficients

$$\begin{aligned}(s^2): &\quad 0 = A + B\\ (s^1): &\quad 0.2 = 0.0834A + C\\ (s^0): &\quad 0.04 = 0.01A\end{aligned}$$

giving

$$A = 4 \quad B = -4 \quad C = -0.1336$$

Substituting values back into (4.86) and complete the square to give

$$H_a(s) = \frac{4}{s} + \frac{-4s - 0.1336}{(s + 0.0417)^2 + 0.0909^2} \tag{4.87}$$

Inverse transform using Laplace transform pairs (9) and (10) in Table 3.1.

$$H_a(s) = \frac{4}{s} - \left\{\frac{4s}{(s + 0.0417)^2 + 0.0909^2}\right\} - 1.4697\left\{\frac{0.0909}{(s + 0.0417)^2 + 0.0909^2}\right\}$$

$$\begin{aligned}h_a(t) = 4 - 4e^{-0.0417t}&\left(\cos 0.0909t - \frac{0.0417}{0.0909}\sin 0.0909t\right)\\ &- 1.4697e^{-0.0417t}\sin 0.0909t\end{aligned}$$

which simplifies to give

$$h_a(t) = 4[1 - e^{-0.0417t}(\cos 0.0909t - 0.0913 \sin 0.0909t)] \tag{4.88}$$

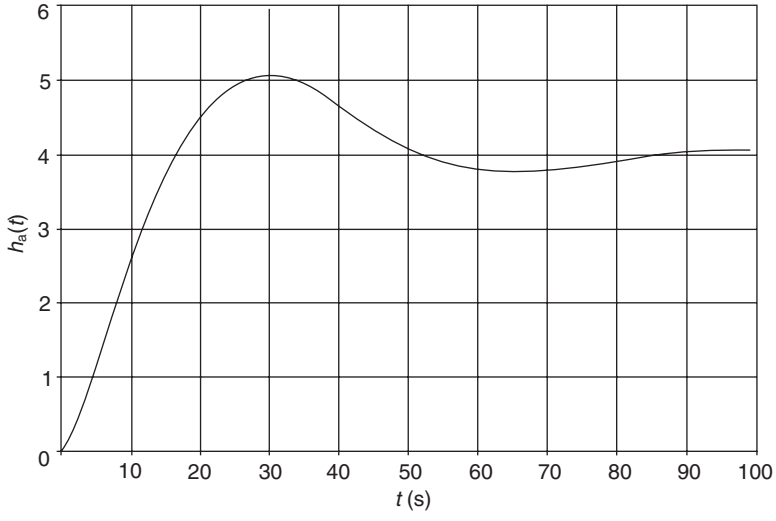


Fig. 4.28 Response of the PI controlled liquid-level system shown in Figure 4.26 to a step change in $h_d(t)$ from 0 to 4 m.

In equation (4.88) the amplitude of the sine term is small, compared with the cosine term, and can be ignored. Hence

$$h_a(t) = 4\left(1 - e^{-0.0417t}\cos 0.0909t\right) \tag{4.89}$$

The time response depicted by equation (4.89) is shown in Figure 4.28.

4.5.4 Proportional plus Integral plus Derivative (PID) control

Most commercial controllers provide full PID (also called three-term) control action. Including a term that is a function of the derivative of the error can, with high-order plants, provide a stable control solution.

Proportional plus Integral plus Derivative control action is expressed as

$$u(t) = K_1 e(t) + K_2 \int e\,dt + K_3 \frac{de}{dt} \tag{4.90}$$

Taking Laplace transforms

$$\begin{aligned} U(s) &= \left(K_1 + \frac{K_2}{s} + K_3 s\right) E(s) \\ &= K_1\left(1 + \frac{K_2}{K_1 s} + \frac{K_3}{K_1} s\right) E(s) \\ &= K_1\left(1 + \frac{1}{T_i s} + T_d s\right) E(s) \end{aligned} \tag{4.91}$$

In equation (4.91), T_d is called the derivative action time, and is formally defined as: 'The time interval in which the part of the control signal due to proportional action increases by an amount equal to the part of the control signal due to derivative action when the error is changing at a constant rate' (BS 1523).

Equation (4.91) can also be expressed as

$$U(s) = \frac{K_1(T_i T_d s^2 + T_i s + 1)}{T_i s} E(s) \tag{4.92}$$

4.5.5 The Ziegler–Nichols methods for tuning PID controllers

The selection of the PID controller parameters K_1, T_i and T_d can be obtained using the classical control system design techniques described in Chapters 5 and 6. In the 1940s, when such tools were just being developed, Ziegler and Nichols (1942) devised two empirical methods for obtaining the controller parameters. These methods are still in use.

(a) *The Process Reaction Method:* This is based on the assumption that the open-loop step response of most process control systems has an S-shape, called the process reaction curve, as shown in Figure 4.29. The process reaction curve may be approximated to a time delay D (also called a transportation lag) and a first-order system of maximum tangential slope R as shown in Figure 4.29 (see also Figure 3.13).

The Process Reaction Method assumes that the optimum response for the closed-loop system occurs when the ratio of successive peaks, as defined by equation (3.71), is 4:1. From equation (3.71) it can be seen that this occurs when the closed-loop damping ratio has a value of 0.21. The controller parameters, as a function of R and D, to produce this response, are given in Table 4.2.

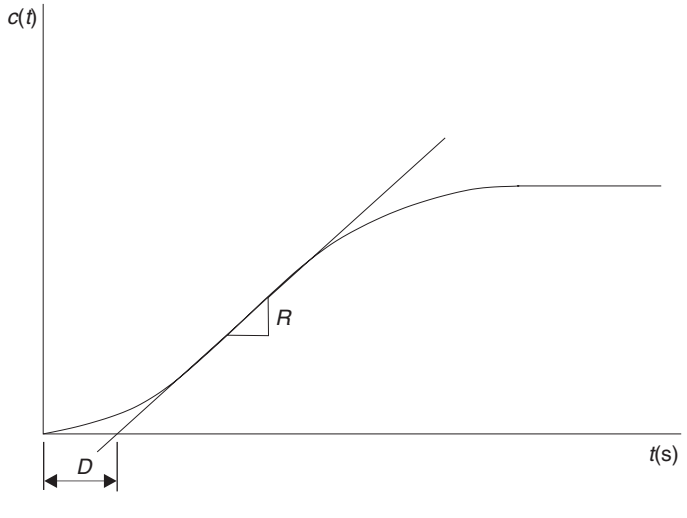


Fig. 4.29 Process reaction curve.

Table 4.2 Ziegler–Nichols PID parameters using the Process Reaction Method

Controller type	K_1	T_i	T_d
P	$1/RD$	–	–
PI	$0.9/RD$	$D/0.3$	–
PID	$1.2/RD$	$2D$	$0.5D$

Table 4.3 Ziegler–Nichols PID parameters using the Continuous Cycling Method

Controller type	K_1	T_i	T_d
P	$K_u/2$	–	–
PI	$K_u/2.2$	$T_u/1.2$	–
PID	$K_u/1.7$	$T_u/2$	$T_u/8$

Note that the Process Reaction Method cannot be used if the open-loop step response has an overshoot, or contains a pure integrator(s).

(b) *The Continuous Cycling Method*: This is a closed-loop technique whereby, using proportional control only, the controller gain K_1 is increased until the system controlled output $c(t)$ oscillates continually at constant amplitude, like a second-order system with no damping. This condition is referred to as marginal stability and is discussed further in Chapters 5 and 6. This value of controller gain is called the ultimate gain K_u, and the time period for one oscillation of $c(t)$ is called the ultimate period T_u. The controller parameters, as a function of K_u and T_u, to provide a similar closed-loop response to the Process Reaction Method, are given in Table 4.3.

The two Ziegler–Nichols PID tuning methods provide a useful 'rule of thumb' empirical approach. The control system design techniques discussed in Chapters 5 and 6 however will generally yield better design solutions.

Of the two techniques, the Process Reaction Method is the easiest and least disruptive to implement. In practice, the measurement of R and D is very subjective, and can lead to errors.

The Continuous Cycling Method, although more disruptive, has the potential to give better results. There is the risk however, particularly with high performance servo-mechanisms, that if K_u is increased by accident to slightly above the marginal stability value, then full instability can occur, resulting in damage to the system.

Actuator saturation and integral wind-up

One of the practical problems of implementing PID control is that of actuator saturation and integral wind-up. Since the range of movement in say, a control valve, has physical limits, once it has saturated, increasing the magnitude of the control signal further has no effect. However, if there is a difference between desired and measured values, the resulting error will cause a continuing increase in the integral term, referred to as integral wind-up. When the error term changes its sign, the integral term starts to 'unwind,' and this can cause long time delays and possible instability. The solution is to limit the maximum value that the integral term can have.

modified PID control schemes have proved their usefulness in providing satisfactory control, although they may not provide optimal control in many given situations.

Outline of the chapter. Section 10–1 has presented introductory material for the chapter. Section 10–2 deals with tuning methods for the basic PID control, commonly known as Ziegler–Nichols tuning rules. Section 10–3 discusses modified PID control schemes, such as PI-D control and I-PD control. Section 10–4 introduces two-degrees-of-freedom PID control schemes. Section 10–5 introduces the concept of robust control using a two-degrees-of-freedom control system as an example.

10–2 TUNING RULES FOR PID CONTROLLERS

PID control of plants. Figure 10–1 shows a PID control of a plant. If a mathematical model of the plant can be derived, then it is possible to apply various design techniques for determining parameters of the controller that will meet the transient and steady-state specifications of the closed-loop system. However, if the plant is so complicated that its mathematical model cannot be easily obtained, then analytical approach to the design of a PID controller is not possible. Then we must resort to experimental approaches to the tuning of PID controllers.

The process of selecting the controller parameters to meet given performance specifications is known as controller tuning. Ziegler and Nichols suggested rules for tuning PID controllers (meaning to set values K_p, T_i, and T_d) based on experimental step responses or based on the value of K_p that results in marginal stability when only the proportional control action is used. Ziegler–Nichols rules, which are presented in the following, are very convenient when mathematical models of plants are not known. (These rules can, of course, be applied to the design of systems with known mathematical models.)

Ziegler–Nichols rules for tuning PID controllers. Ziegler and Nichols proposed rules for determining values of the proportional gain K_p, integral time T_i, and derivative time T_d based on the transient response characteristics of a given plant. Such determination of the parameters of PID controllers or tuning of PID controllers can be made by engineers on site by experiments on the plant. (Numerous tuning rules for PID controllers have been proposed since the Ziegler–Nichols proposal. They are available in the literature. Here, however, we introduce only the Ziegler–Nichols tuning rules.)

There are two methods called Ziegler–Nichols tuning rules. In both methods, they aimed at obtaining 25% maximum overshoot in step response (see Figure 10–2).

First method. In the first method, we obtain experimentally the response of the plant to a unit-step input, as shown in Figure 10–3. If the plant involves neither inte-

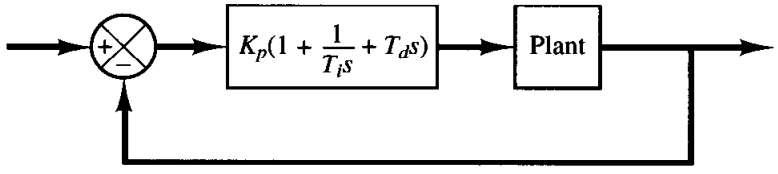


Figure 10–1
PID control of a plant.

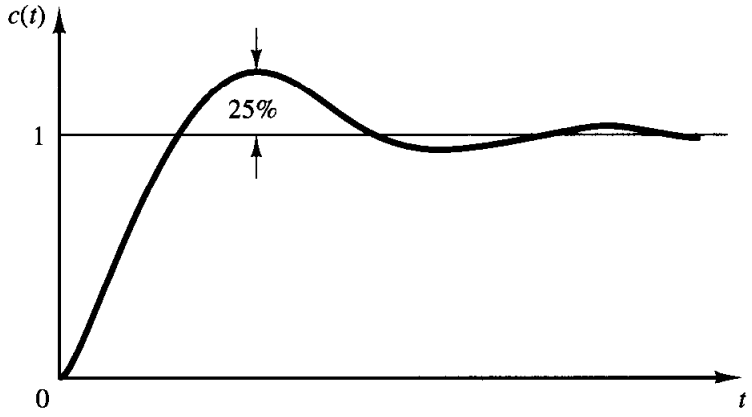


Figure 10–2
Unit-step response curve showing 25% maximum overshoot.

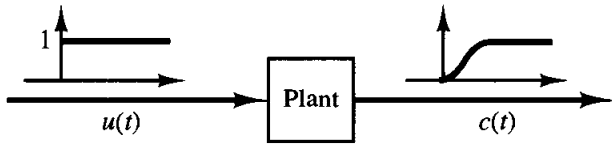


Figure 10–3
Unit-step response of a plant.

grator(s) nor dominant complex-conjugate poles, then such a unit-step response curve may look like an S-shaped curve, as shown in Figure 10–4. (If the response does not exhibit an S-shaped curve, this method does not apply.) Such step-response curves may be generated experimentally or from a dynamic simulation of the plant.

The S-shaped curve may be characterized by two constants, delay time L and time constant T. The delay time and time constant are determined by drawing a tangent line at the inflection point of the S-shaped curve and determining the intersections of the tangent line with the time axis and line $c(t) = K$, as shown in Figure 10–4. The transfer function $C(s)/U(s)$ may then be approximated by a first-order system with a transport lag as follows:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts + 1}$$

Ziegler and Nichols suggested to set the values of K_p, T_i, and T_d according to the formula shown in Table 10–1.

Notice that the PID controller tuned by the first method of Ziegler–Nichols rules gives

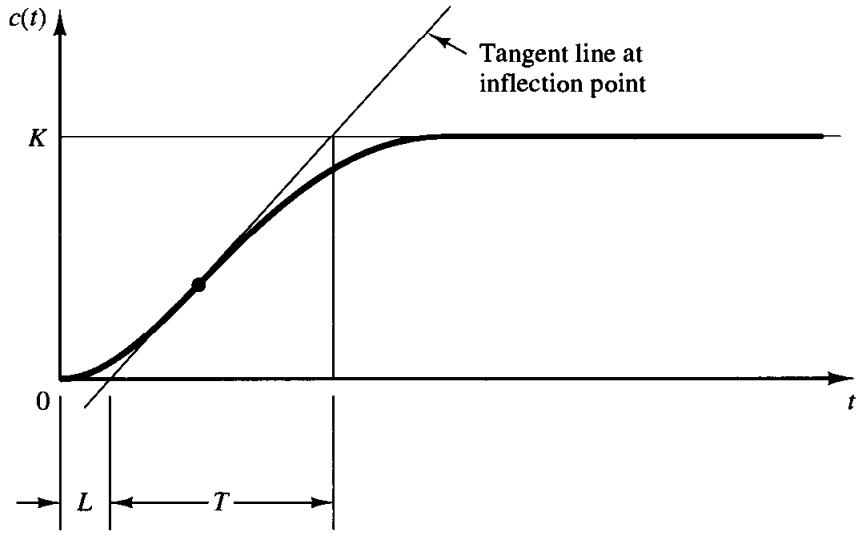


Figure 10–4
S-shaped response curve.

Table 10–1 Ziegler–Nichols Tuning Rule Based on Step Response of Plant (First Method)

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

$$= 1.2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0.5Ls\right)$$

$$= 0.6T\frac{\left(s + \frac{1}{L}\right)^2}{s}$$

Thus, the PID controller has a pole at the origin and double zeros at $s = -1/L$.

Second method. In the second method, we first set $T_i = \infty$ and $T_d = 0$. Using the proportional control action only (see Figure 10–5), increase K_p from 0 to a critical value K_{cr} where the output first exhibits sustained oscillations. (If the output does not exhibit sustained oscillations for whatever value K_p may take, then this method does not apply.) Thus, the critical gain K_{cr} and the corresponding period P_{cr} are experimentally determined (see Figure 10–6). Ziegler and Nichols suggested that we set the values of the parameters K_p, T_i, and T_d according to the formula shown in Table 10–2.

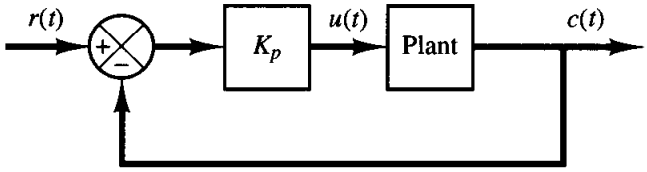


Figure 10–5
Closed-loop system with a proportional controller.

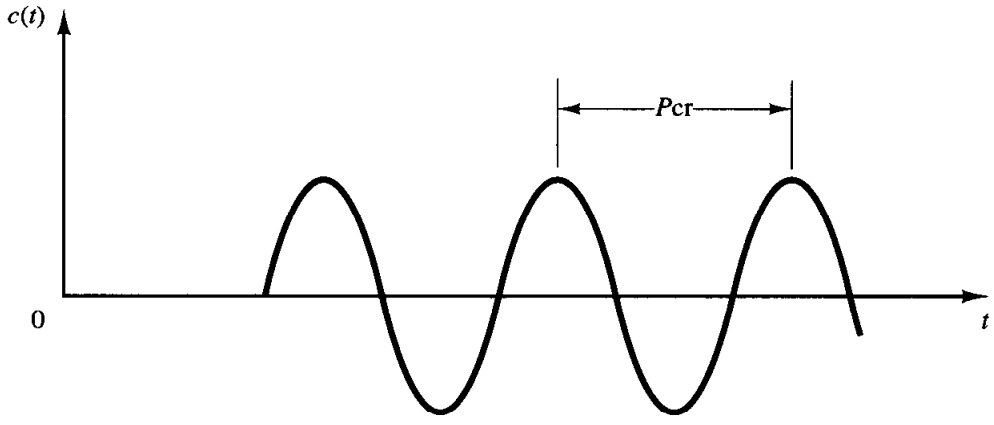


Figure 10–6
Sustained oscillation with period P_{cr}.

Table 10–2 Ziegler–Nichols Tuning Rule Based on Critical Gain K_{cr} and Critical Period P_{cr} (Second Method)

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Notice that the PID controller tuned by the second method of Ziegler–Nichols rules gives

$$\begin{aligned} G_c(s) &= K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \\ &= 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125\,P_{cr}s\right) \\ &= 0.075K_{cr}P_{cr}\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s} \end{aligned}$$

Thus, the PID controller has a pole at the origin and double zeros at $s = -4/P_{cr}$.

Comments. Ziegler–Nichols tuning rules (and other tuning rules presented in the literature) have been widely used to tune PID controllers in process control systems where the plant dynamics are not precisely known. Over many years, such tuning rules proved to be very useful. Ziegler–Nichols tuning rules can, of course, be applied to plants whose dynamics are known. (If plant dynamics are known, many analytical and graphical approaches to the design of PID controllers are available, in addition to Ziegler–Nichols tuning rules.)

If the transfer function of the plant is known, a unit-step response may be calculated or the critical gain K_{cr} and critical period P_{cr} may be calculated. Then, using those calculated values, it is possible to determine the parameters K_p, T_i, and T_d from Table 10–1 or 10–2. However, the real usefulness of Ziegler–Nichols tuning rules (and other tuning rules) becomes apparent when the plant dynamics are not known so that no analytical or graphical approaches to the design of controllers are available.

Generally, for plants with complicated dynamics but no integrators, Ziegler–Nichols tuning rules can be applied. However, if the plant has an integrator, these rules may not be applied in some cases. To illustrate such a case where Ziegler–Nichols rules do not apply, consider the following case: Suppose that a unity-feedback control system has a plant whose transfer function is

$$G(s) = \frac{(s + 2)(s + 3)}{s(s + 1)(s + 5)}$$

Because of the presence of an integrator, the first method does not apply. Referring to

Figure 10–3, the step response of this plant will not have an S-shaped response curve; rather, the response increases with time. Also, if the second method is attempted (see Figure 10–5), the closed-loop system with a proportional controller will not exhibit sustained oscillations whatever value the gain K_p may take. This can be seen from the following analysis. Since the characteristic equation is

$$s(s + 1)(s + 5) + K_p(s + 2)(s + 3) = 0$$

or

$$s^3 + (6 + K_p)s^2 + (5 + 5K_p)s + 6K_p = 0$$

the Routh array becomes

$$\begin{array}{lcc} s^3 & 1 & 5 + 5K_p \\ s^2 & 6 + K_p & 6K_p \\ s^1 & \dfrac{30 + 29K_p + 5K_p^2}{6 + K_p} & 0 \\ s^0 & 6K_p & \end{array}$$

The coefficients in the first column are positive for all values of positive K_p. Thus, in the present case the closed-loop system will not exhibit sustained oscillations and, therefore, the critical gain value K_{cr} does not exist. Hence, the second method does not apply.

If the plant is such that Ziegler–Nichols rules can be applied, then the plant with a PID controller tuned by Ziegler–Nichols rules will exhibit approximately 10% ~ 60% maximum overshoot in step response. On the average (experimented on many different plants), the maximum overshoot is approximately 25%. (This is quite understandable because the values suggested in Tables 10–1 and 10–2 are based on the average.) In a given case, if the maximum overshoot is excessive, it is always possible (experimentally or otherwise) to make fine tuning so that the closed-loop system will exhibit satisfactory transient responses. In fact, Ziegler–Nichols tuning rules give an educated guess for the parameter values and provide a starting point for fine tuning.

EXAMPLE 10–1 Consider the control system shown in Figure 10–7 in which a PID controller is used to control the system. The PID controller has the transfer function

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

Although many analytical methods are available for the design of a PID controller for the present system, let us apply a Ziegler–Nichols tuning rule for the determination of the values of parameters K_p, T_i, and T_d. Then obtain a unit-step response curve and check to see if the designed system exhibits approximately 25% maximum overshoot. If the maximum overshoot is excessive (40% or more), make a fine tuning and reduce the amount of the maximum overshoot to approximately 25%.

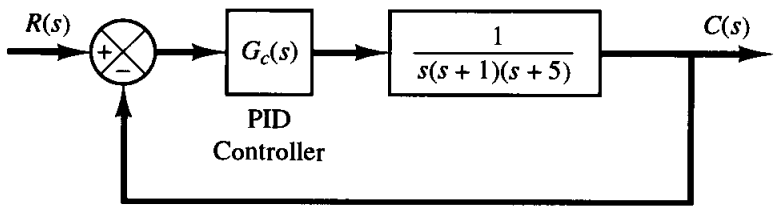


Figure 10–7 PID-controlled system.

Since the plant has an integrator, we use the second method of Ziegler–Nichols tuning rules. By setting $T_i = \infty$ and $T_d = 0$, we obtain the closed-loop transfer function as follows:

$$\frac{C(s)}{R(s)} = \frac{K_p}{s(s + 1)(s + 5) + K_p}$$

The value of K_p that makes the system marginally stable so that sustained oscillation occurs can be obtained by use of Routh's stability criterion. Since the characteristic equation for the closed-loop system is

$$s^3 + 6s^2 + 5s + K_p = 0$$

the Routh array becomes as follows:

$$\begin{array}{lcc} s^3 & 1 & 5 \\ s^2 & 6 & K_p \\ s^1 & \dfrac{30 - K_p}{6} & \\ s^0 & K_p & \end{array}$$

Examining the coefficients of the first column of the Routh table, we find that sustained oscillation will occur if $K_p = 30$. Thus, the critical gain K_{cr} is

$$K_{cr} = 30$$

With gain K_p set equal to K_{cr} (= 30), the characteristic equation becomes

$$s^3 + 6s^2 + 5s + 30 = 0$$

To find the frequency of the sustained oscillation, we substitute $s = j\omega$ into this characteristic equation as follows:

$$(j\omega)^3 + 6(j\omega)^2 + 5(j\omega) + 30 = 0$$

or

$$6(5 - \omega^2) + j\omega(5 - \omega^2) = 0$$

from which we find the frequency of the sustained oscillation to be $\omega^2 = 5$ or $\omega = \sqrt{5}$. Hence, the period of sustained oscillation is

$$P_{cr} = \frac{2\pi}{\omega} = \frac{2\pi}{\sqrt{5}} = 2.8099$$

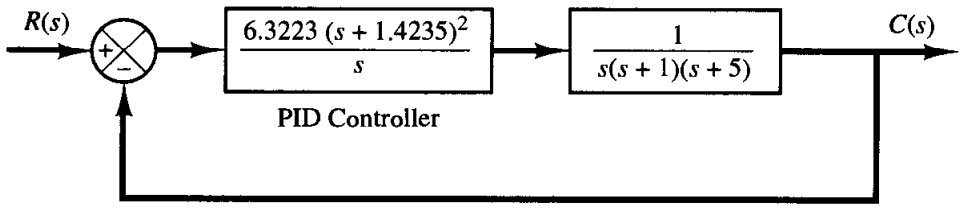
Referring to Table 10–2, we determine K_p, T_i, and T_d as follows:

$$K_p = 0.6K_{cr} = 18$$
$$T_i = 0.5P_{cr} = 1.405$$
$$T_d = 0.125P_{cr} = 0.35124$$

The transfer function of the PID controller is thus

$$\begin{aligned} G_c(s) &= K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \\ &= 18\left(1 + \frac{1}{1.405s} + 0.35124s\right) \\ &= \frac{6.3223(s + 1.4235)^2}{s} \end{aligned}$$

Figure 10–8
Block diagram of the system with PID controller designed by use of Ziegler–Nichols tuning rule (second method).



The PID controller has a pole at the origin and double zero at $s = -1.4235$. A block diagram of the control system with the designed PID controller is shown in Figure 10–8.

Next, let us examine the unit-step response of the system. The closed-loop transfer function $C(s)/R(s)$ is given by

$$\frac{C(s)}{R(s)} = \frac{6.3223s^2 + 18s + 12.811}{s^4 + 6s^3 + 11.3223s^2 + 18s + 12.811}$$

The unit-step response of this system can be obtained easily with MATLAB. See MATLAB Program 10–1. The resulting unit-step response curve is shown in Figure 10–9. The maximum overshoot in the unit-step response is approximately 62%. The amount of maximum overshoot is excessive. It can be reduced by fine tuning the controller parameters. Such fine tuning can be made on the computer. We find that by keeping $K_p = 18$ and by moving the double zero of the PID controller to $s = -0.65$, that is, using the PID controller

$$G_c(s) = 18\left(1 + \frac{1}{3.077s} + 0.7692s\right) = 13.846\,\frac{(s + 0.65)^2}{s} \tag{10–1}$$

the maximum overshoot in the unit-step response can be reduced to approximately 18% (see Figure 10–10). If the proportional gain K_p is increased to 39.42, without changing the location of the double zero ($s = -0.65$), that is, using the PID controller

$$G_c(s) = 39.42\left(1 + \frac{1}{3.077s} + 0.7692s\right) = 30.322\,\frac{(s + 0.65)^2}{s} \tag{10–2}$$

then the speed of response is increased, but the maximum overshoot is also increased to approximately 28%, as shown in Figure 10–11. Since the maximum overshoot in this case is fairly close to 25% and the response is faster than the system with $G_c(s)$ given by Equation (10–1), we may consider $G_c(s)$ as given by Equation (10–2) as acceptable. Then the tuned values of K_p, T_i, and T_d become

$$K_p = 39.42, \qquad T_i = 3.077, \qquad T_d = 0.7692$$

It is interesting to observe that these values respectively are approximately twice the values suggested by the second method of the Ziegler–Nichols tuning rule. The important thing to note here is that the Ziegler–Nichols tuning rule has provided a starting point for fine tuning.

It is instructive to note that, for the case where the double zero is located at $s = -1.4235$, increasing the value of K_p increases the speed of response, but as far as the percentage maximum

MATLAB Program 10–1

```
% ---------- Unit-step response ----------

num = [0   0   6.3223   18   12.811];
den = [1   6   11.3223   18   12.811];
step(num,den)
grid
title('Unit-Step Response')
```

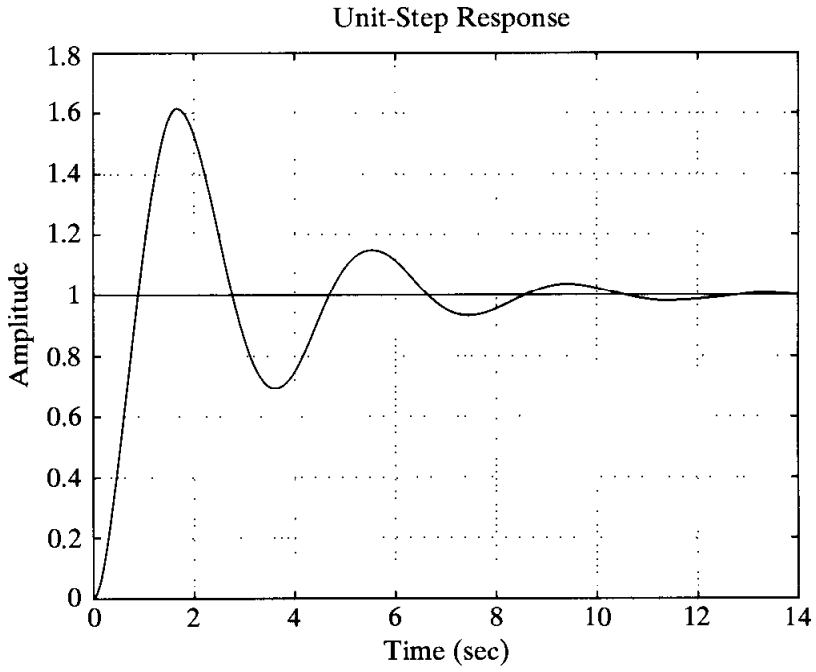


Figure 10–9
Unit-step response curve of PID-controlled system designed by use of Ziegler–Nichols tuning rule (second method).

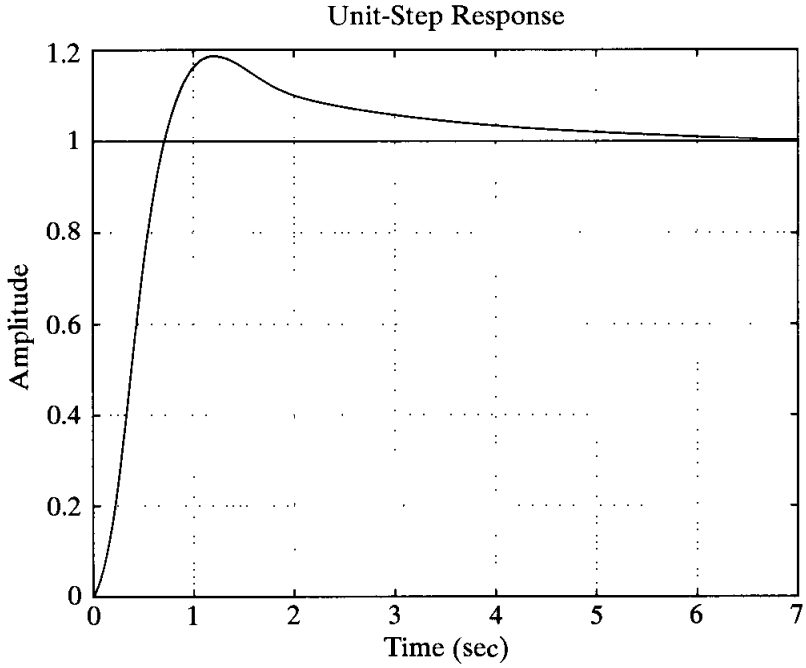


Figure 10–10
Unit-step response of the system shown in Figure 10–7 with PID controller having parameters $K_p = 18$, $T_i = 3.077$, and $T_d = 0.7692$.

overshoot is concerned, varying gain K_p has very little effect. The reason for this may be seen from the root-locus analysis. Figure 10–12 shows the root-locus diagram for the system designed by use of the second method of Ziegler–Nichols tuning rules. Since the dominant branches of root loci are along the $\zeta = 0.3$ lines for a considerable range of K, varying the value of K (from 6 to 30) will not change the damping ratio of the dominant closed-loop poles very much. However, varying the location of the double zero has a significant effect on the maximum overshoot, because the damping ratio of the dominant closed-loop poles can be changed significantly. This can also be seen from the root-locus analysis. Figure 10–13 shows the root-locus diagram for the system where the PID controller has the double zero at $s = -0.65$. Notice the change of the root-locus configuration. This change in the configuration makes it possible to change the damping ratio of the dominant closed-loop poles.

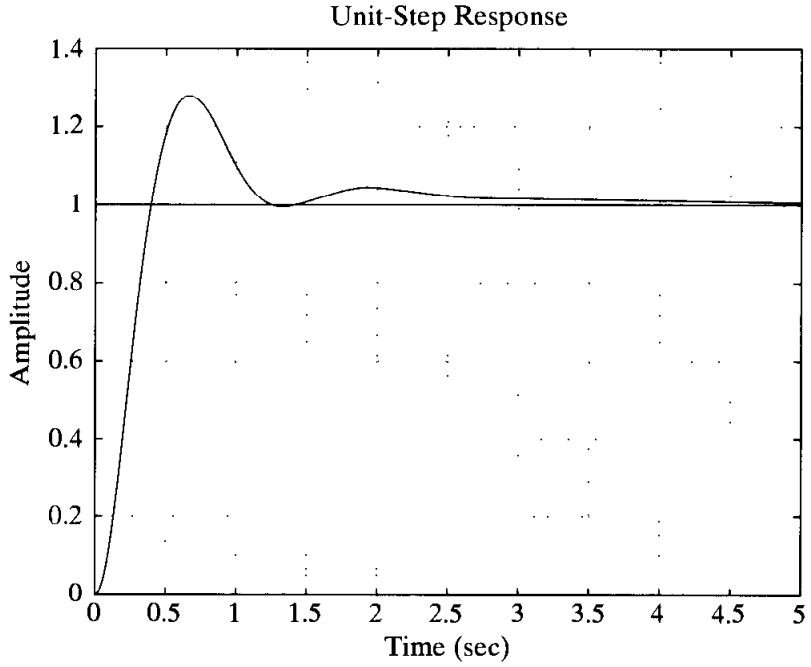


Figure 10–11
Unit-step response of the system shown in Figure 10–7 with PID controller having parameters $K_p = 39.42$, $T_i = 3.077$, and $T_d = 0.7692$.

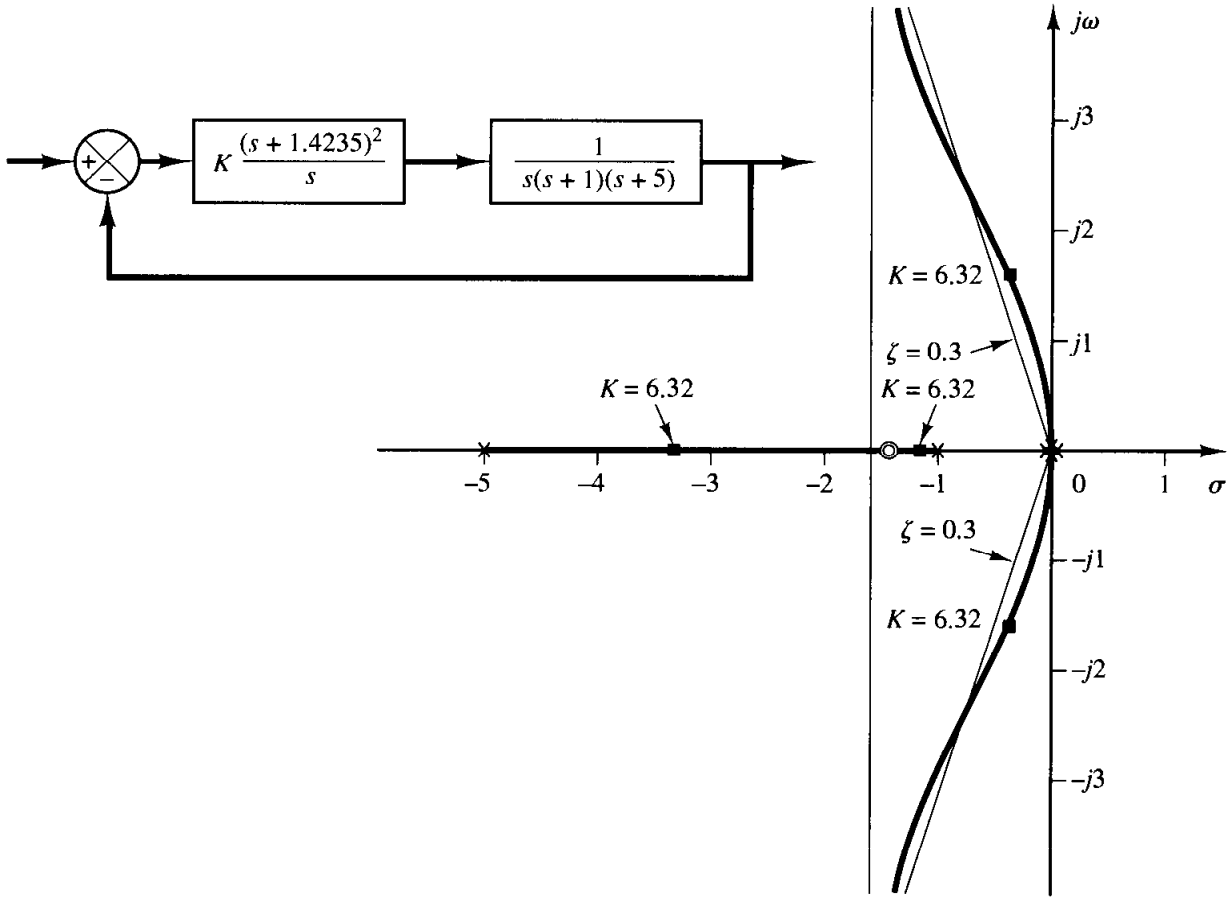


Figure 10–12
Root-locus diagram of system when PID controller has double zero at $s = -1.4235$.

In Figure 10–13, notice that, in the case where the system has gain $K = 30.322$, the closed-loop poles at $s = -2.35 \pm j4.82$ act as dominant poles. Two additional closed-loop poles are very near the double zero at $s = -0.65$, with the result that these closed-loop poles and the double zero almost cancel each other. The dominant pair of closed-loop poles indeed determines the nature of the response. On the other hand, when the system has $K = 13.846$, the closed-loop poles at $s = -2.35 \pm j2.62$ are not quite dominant because the two other closed-loop poles near the dou-

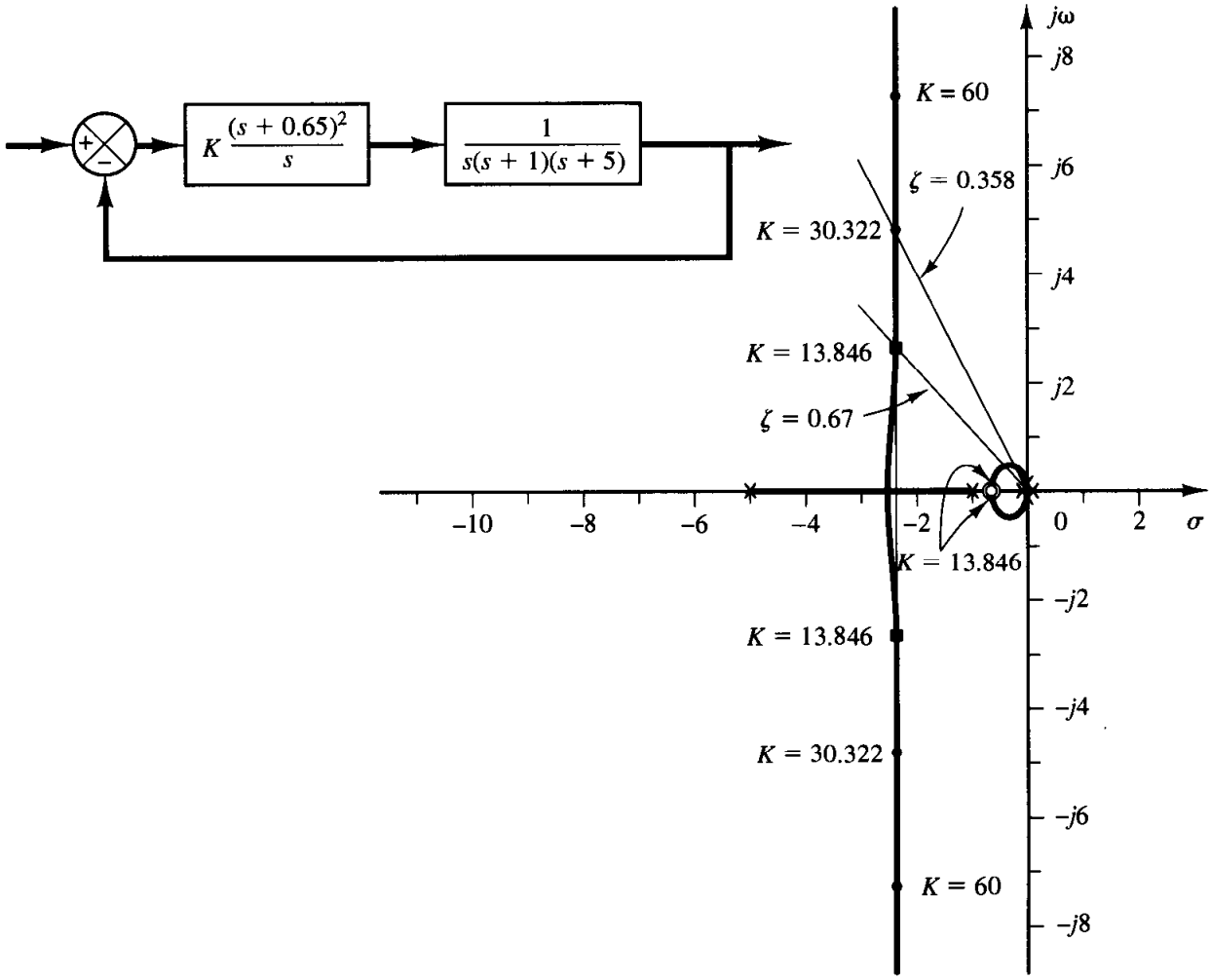


Figure 10–13
Root-locus diagram of system when PID controller has double zero at $s = -0.65$. $K = 13.846$ corresponds to $G_c(s)$ given by Equation (10–1) and $K = 30.322$ corresponds to $G_c(s)$ given by Equation (10–2).

ble zero at $s = -0.65$ have considerable effect on the response. The maximum overshoot in the step response in this case (18%) is much larger than the case where the system is of second-order having only dominant closed-loop poles. (In the latter case the maximum overshoot in the step response would be approximately 6%.)

10–3 MODIFICATIONS OF PID CONTROL SCHEMES

Consider the basic PID control system shown in Figure 10–14(a), where the system is subjected to disturbances and noises. Figure 10–14(b) is a modified block diagram of the same system. In the basic PID control system such as the one shown in Figure 10–14(b), if the reference input is a step function, then, because of the presence of the derivative term in the control action, the manipulated variable $u(t)$ will involve an impulse function (delta function). In an actual PID controller, instead of the pure derivative term $T_d s$ we employ

$$\frac{T_d s}{1 + \gamma T_d s}$$

where the value of γ is somewhere around 0.1. Therefore, when the reference input is a step function, the manipulated variable $u(t)$ will not involve an impulse function, but will involve a sharp pulse function. Such a phenomenon is called *set-point kick.*

7

Digital control system design

7.1 Microprocessor control

As a result of developments in microprocessor technology, the implementation of control algorithms is now invariably through the use of embedded microcontrollers rather than employing analogue devices. A typical system using microprocessor control is shown in Figure 7.1.

In Figure 7.1

- RAM is Random Access Memory and is used for general purpose working space during computation and data transfer.
- ROM, PROM, EPROM is Read Only Memory, Programmable Read Only Memory and Erasable Programmable Read Only Memory and are used for rapid sources of information that seldom, or never need to be modified.
- A/D Converter converts analogue signals from sensors into digital form at a given sampling period T seconds and given resolution (8 bits, 16 bits, 24 bits, etc.)
- D/A Converter converts digital signals into analogue signals suitable for driving actuators and other devices.

The elements of a microprocessor controller (microcontroller) are shown in Figure 7.2. Figure 7.2 shows a Central Processing Unit (CPU) which consists of

- the Arithmetic Logic Unit (ALU) which performs arithmetic and logical operations on the data

and a number of registers, typically

- Program Counter – incremented each time an instruction is executed
- Accumulator(s) – can undertake arithmetic operations
- Instruction register – holds current instruction
- Data address register – holds memory address of data

Control algorithms are implemented in either high level or low level language. The lowest level of code is executable machine code, which is a sequence of binary words that is understood by the CPU. A higher level of language is an assembler, which employs meaningful mnemonics and names for data addresses. Programs written in assembler are rapid in execution. At a higher level still are languages

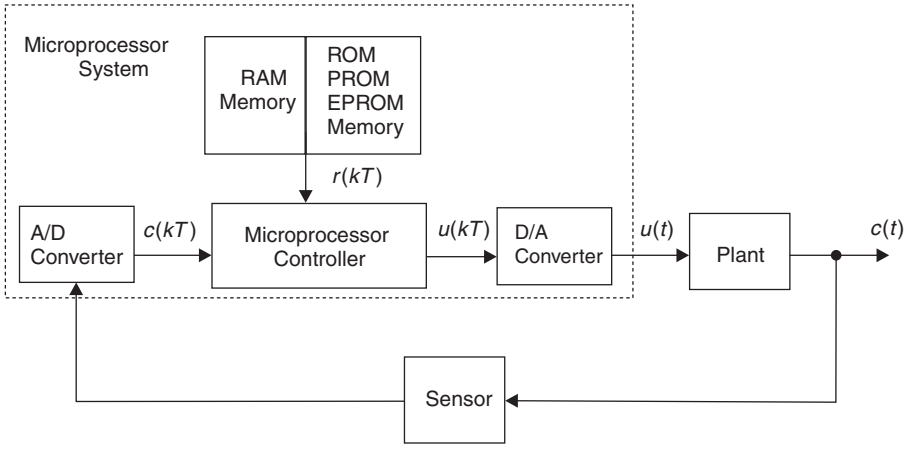


Fig. 7.1 Microprocessor control of a plant.

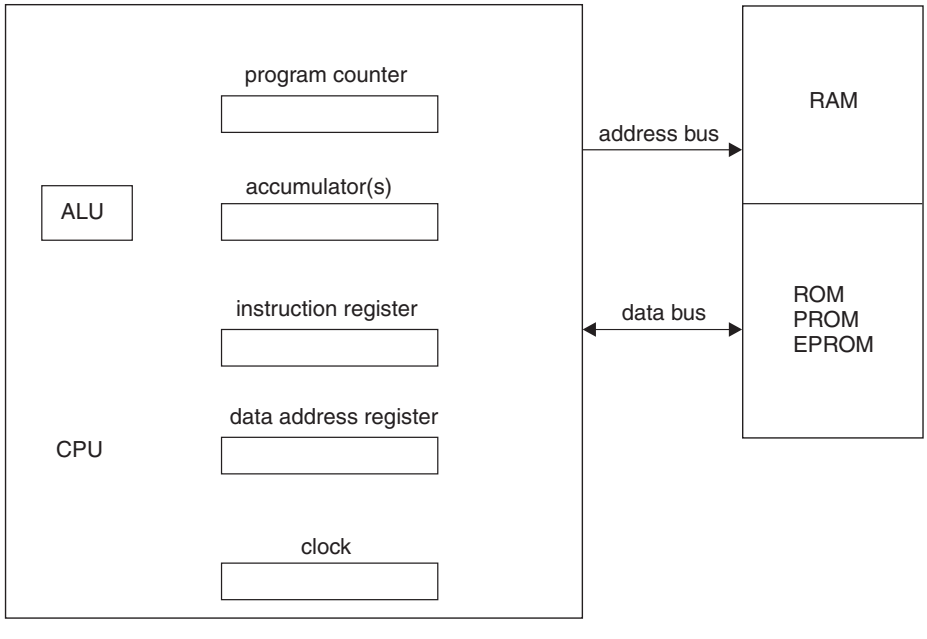


Fig. 7.2 Elements of a microprocessor controller.

such as C and C++, which are rapidly becoming industry standard for control software.

The advantages of microprocessor control are

- Versatility – programs may easily be changed
- Sophistication – advanced control laws can be implemented.

The disadvantages of microprocessor control are

- Works in discrete time – only snap-shots of the system output through the A/D converter are available. Hence, to ensure that all relevant data is available, the frequency of sampling is very important.

7.2 Shannon's sampling theorem

Shannon's sampling theorem states that 'A function $f(t)$ that has a bandwidth ω_b is uniquely determined by a discrete set of sample values provided that the sampling frequency is greater than $2\omega_b$'. The sampling frequency $2\omega_b$ is called the Nyquist frequency.

It is rare in practise to work near to the limit given by Shannon's theorem. A useful rule of thumb is to sample the signal at about ten times higher than the highest frequency thought to be present.

If a signal is sampled below Shannon's limit, then a lower frequency signal, called an alias may be constructed as shown in Figure 7.3.

To ensure that aliasing does not take place, it is common practice to place an anti-aliasing filter before the A/D converter. This is an analogue low-pass filter with a break-frequency of $0.5\omega_s$ where ω_s is the sampling frequency ($\omega_s > 10\omega_b$). The higher ω_s is in comparison to ω_b, the more closely the digital system resembles an analogue one and as a result, the more applicable are the design methods described in Chapters 5 and 6.

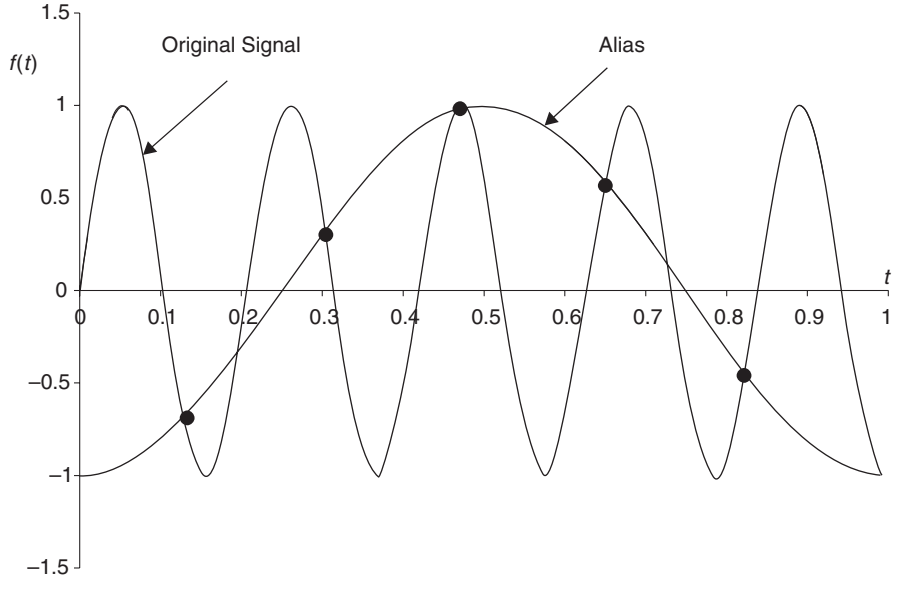


Fig. 7.3 Construction of an alias due to undersampling.

7.3 Ideal sampling

An ideal sample $f^*(t)$ of a continuous signal $f(t)$ is a series of zero width impulses spaced at sampling time T seconds apart as shown in Figure 7.4.

The sampled signal is represented by equation (7.1).

$$f^*(t) = \sum_{k=-\infty}^{\infty} f(kT)\delta(t - kT) \tag{7.1}$$

where $\delta(t - kT)$ is the unit impulse function occurring at $t = kT$.

A sampler (i.e. an A/D converter) is represented by a switch symbol as shown in Figure 7.5. It is possible to reconstruct $f(t)$ approximately from $f^*(t)$ by the use of a hold device, the most common of which is the zero-order hold (D/A converter) as shown in Figure 7.6. From Figure 7.6 it can be seen that a zero-order hold converts a series of impulses into a series of pulses of width T. Hence a unit impulse at time t is converted into a pulse of width T, which may be created by a positive unit step at time t, followed by a negative unit step at time $(t - T)$, i.e. delayed by T.

The transfer function for a zero-order hold is

$$\begin{aligned} \mathscr{L}[f(t)] &= \frac{1}{s} - \frac{1}{s}e^{-Ts} \\ G_h(s) &= \frac{1 - e^{-Ts}}{s} \end{aligned} \tag{7.2}$$

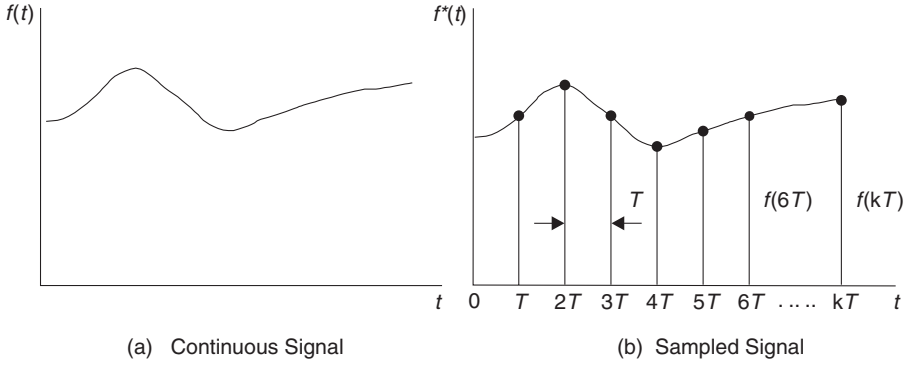


Fig. 7.4 The sampling process.

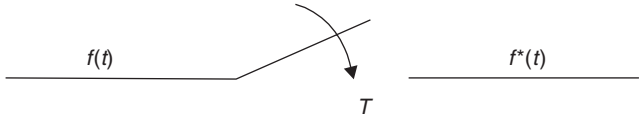


Fig. 7.5 A sampler.

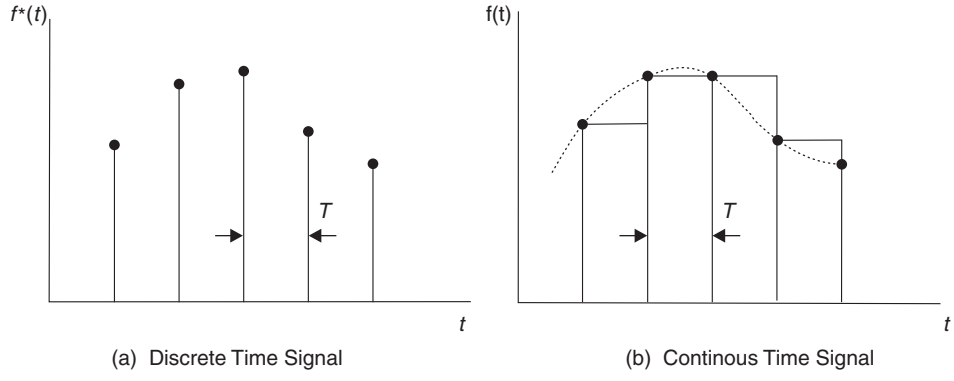


Fig. 7.6 Construction of a continuous signal using a zero-order hold.

7.4 The *z*-transform

The z-transform is the principal analytical tool for single-input–single-output discrete-time systems, and is analogous to the Laplace transform for continuous systems.

Conceptually, the symbol z can be associated with discrete time shifting in a difference equation in the same way that s can be associated with differentiation in a differential equation.

Taking Laplace transforms of equation (7.1), which is the ideal sampled signal, gives

$$F^*(s) = \mathscr{L}[f^*(t)] = \sum_{k=0}^{\infty} f(kT)\mathrm{e}^{-kTs} \tag{7.3}$$

or

$$F^*(s) = \sum_{k=0}^{\infty} f(kT)\left(\mathrm{e}^{sT}\right)^{-k} \tag{7.4}$$

Define z as

$$z = \mathrm{e}^{sT} \tag{7.5}$$

then

$$F(z) = \sum_{k=0}^{\infty} f(kT)z^{-k} = Z[f(t)] \tag{7.6}$$

In 'long-hand' form equation (7.6) is written as

$$F(z) = f(0) + f(T)z^{-1} + f(2T)z^{-2} + \cdots + f(kT)z^{-k} \tag{7.7}$$

Example 7.1
Find the z-transform of the unit step function $f(t) = 1$.

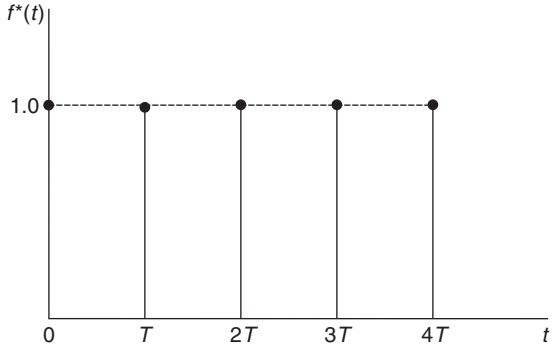


Fig. 7.7 z-Transform of a sampled unit step function.

Solution
From equations (7.6) and (7.7)

$$Z[1(t)] = \sum_{k=0}^{\infty} 1(kT)z^{-k} \tag{7.8}$$

or

$$F(z) = 1 + z^{-1} + z^{-2} + \ldots + z^{-k} \tag{7.9}$$

Figure 7.7 shows a graphical representation of equation (7.9).
Equation (7.9) can be written in 'closed form' as

$$Z[1(t)] = \frac{z}{z-1} = \frac{1}{1-z^{-1}} \tag{7.10}$$

Equations (7.9) and (7.10) can be shown to be the same by long division

$$\begin{array}{r} 1 + z^{-1} + z^{-2} + \cdots \\ z-1 \overline{)\, z \quad 0 \quad\quad 0} \\ \underline{z-1} \\ 0+1 \\ \quad\quad \underline{1-z^{-1}} \\ \quad\quad 0+z^{-1} \\ \quad\quad\quad\; z^{-1} - z^{-2} \end{array} \tag{7.11}$$

Table 7.1 gives Laplace and z-transforms of common functions.
z-transform Theorems:

(a) Linearity

$$Z[f_1(t) \pm f_2(t)] = F_1(z) \pm F_2(z) \tag{7.12}$$

Table 7.1 Common Laplace and z-transforms

	$f(t)$ or $f(kT)$	$F(s)$	$F(z)$
1	$\delta(t)$	1	1
2	$\delta(t-kT)$	e^{-kTs}	z^{-k}
3	$1(t)$	$\frac{1}{s}$	$\frac{z}{z-1}$
4	t	$\frac{1}{s^2}$	$\frac{Tz}{(z-1)^2}$
5	e^{-at}	$\frac{1}{(s+a)}$	$\frac{z}{z-\mathrm{e}^{-aT}}$
6	$1-\mathrm{e}^{-at}$	$\frac{a}{s(s+a)}$	$\frac{z(1-\mathrm{e}^{-aT})}{(z-1)(z-\mathrm{e}^{-aT})}$
7	$\frac{1}{a}(at-1+\mathrm{e}^{-at})$	$\frac{a}{s^2(s+a)}$	$\frac{z\{(aT-1+\mathrm{e}^{-aT})z+(1-\mathrm{e}^{-aT}-aT\mathrm{e}^{-aT})\}}{a(z-1)^2(z-\mathrm{e}^{-aT})}$
8	$\sin\omega t$	$\frac{\omega}{s^2+\omega^2}$	$\frac{z\sin\omega T}{z^2-2z\cos\omega T+1}$
9	$\cos\omega t$	$\frac{s}{s^2+\omega^2}$	$\frac{z(z-\cos\omega T)}{z^2-2z\cos\omega T+1}$
10	$\mathrm{e}^{-at}\sin\omega t$	$\frac{\omega}{(s+a)^2+\omega^2}$	$\frac{z\mathrm{e}^{-aT}\sin\omega T}{z^2-2z\mathrm{e}^{-aT}\cos\omega T+\mathrm{e}^{-2aT}}$
11	$\mathrm{e}^{-at}\cos\omega t$	$\frac{(s+a)}{(s+a)^2+\omega^2}$	$\frac{z^2-z\mathrm{e}^{-aT}\cos\omega T}{z^2-2z\mathrm{e}^{-aT}\cos\omega T+\mathrm{e}^{-2aT}}$

(b) Initial Value Theorem

$$f(0)=\lim_{z\to\infty} F(z) \tag{7.13}$$

(c) Final Value Theorem

$$f(\infty)=\lim_{z\to 1}\left[\left(\frac{z-1}{z}\right)F(z)\right] \tag{7.14}$$

7.4.1 Inverse transformation

The discrete time response can be found using a number of methods.

(a) Infinite power series method

Example 7.2
A sampled-data system has a transfer function

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

If the sampling time is one second and the system is subject to a unit step input function, determine the discrete time response. (N.B. normally, a zero-order hold would be included, but, in the interest of simplicity, has been omitted.) Now

$$X_o(z) = G(z)X_i(z) \tag{7.15}$$

from Table 7.1

$$X_o(z) = \left(\frac{z}{z - e^{-T}}\right)\left(\frac{z}{z-1}\right) \tag{7.16}$$

for $T = 1$ second

$$\begin{aligned} X_o(z) &= \left(\frac{z}{z-0.368}\right)\left(\frac{z}{z-1}\right) \\ &= \frac{z^2}{z^2 - 1.368z + 0.368} \end{aligned} \tag{7.17}$$

By long division

$$\begin{array}{r} 1 + 1.368z^{-1} + 1.503z^{-2} + \cdots \\ z^2 - 1.368z + 0.368 \overline{)\, z^2 \quad 0 \quad 0 \quad 0} \\ \underline{z^2 - 1.368z + 0.368} \\ 0 + 1.368z - 0.368 \\ \underline{1.368z - 1.871 + 0.503z^{-1}} \\ 0 + 1.503 - 0.503z^{-1} \\ \underline{1.503 - 2.056z^{-1} + 0.553z^{-2}} \end{array} \tag{7.18}$$

Thus

$$x_o(0) = 1$$

$$x_o(1) = 1.368$$

$$x_o(2) = 1.503$$

(b) Difference equation method

Consider a system of the form

$$\frac{X_o}{X_i}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots} \tag{7.19}$$

Thus

$$(1 + a_1 z^{-1} + a_2 z^{-2} + \cdots)X_o(z) = (b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots)X_i(z) \tag{7.20}$$

or

$$X_o(z) = (-a_1 z^{-1} - a_2 z^{-2} - \cdots)X_o(z) + (b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots)X_i(z) \tag{7.21}$$

Equation (7.21) can be expressed as a difference equation of the form

$$\begin{aligned} x_o(kT) = &-a_1 x_o(k-1)T - a_2 x_o(k-2)T - \cdots \\ &+ b_0 x_i(kT) + b_1 x_i(k-1)T + b_2 x_i(k-2)T + \cdots \end{aligned} \tag{7.22}$$

In Example 7.2

$$\frac{X_o}{X_i}(s) = \frac{1}{1+s}$$
$$= \frac{z}{z - e^{-T}} = \frac{z}{z - 0.368} \tag{7.23}$$

Equation (7.23) can be written as

$$\frac{X_o}{X_i}(z) = \frac{1}{1 - 0.368z^{-1}} \tag{7.24}$$

Equation (7.24) is in the same form as equation (7.19). Hence

$$(1 - 0.368z^{-1})X_o(z) = X_i(z)$$

or

$$X_o(z) = 0.368z^{-1}X_o(z) + X_i(z) \tag{7.25}$$

Equation (7.25) can be expressed as a difference equation

$$x_o(kT) = 0.368x_o(k-1)T + x_i(kT) \tag{7.26}$$

Assume that $x_o(-1) = 0$ and $x_i(kT) = 1$, then from equation (7.26)

$$x_o(0) = 0 + 1 = 1, \quad k = 0$$
$$x_o(1) = (0.368 \times 1) + 1 = 1.368, \quad k = 1$$
$$x_o(2) = (0.368 \times 1.368) + 1 = 1.503, \quad k = 2 \quad \text{etc.}$$

These results are the same as with the power series method, but difference equations are more suited to digital computation.

7.4.2 The pulse transfer function

Consider the block diagrams shown in Figure 7.8. In Figure 7.8(a) $U^*(s)$ is a sampled input to $G(s)$ which gives a continuous output $X_o(s)$, which when sampled by a

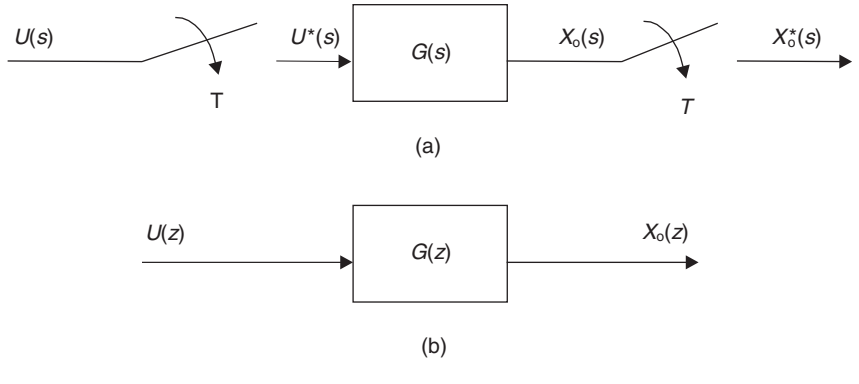


Fig. 7.8 Relationship between $G(s)$ and $G(z)$.

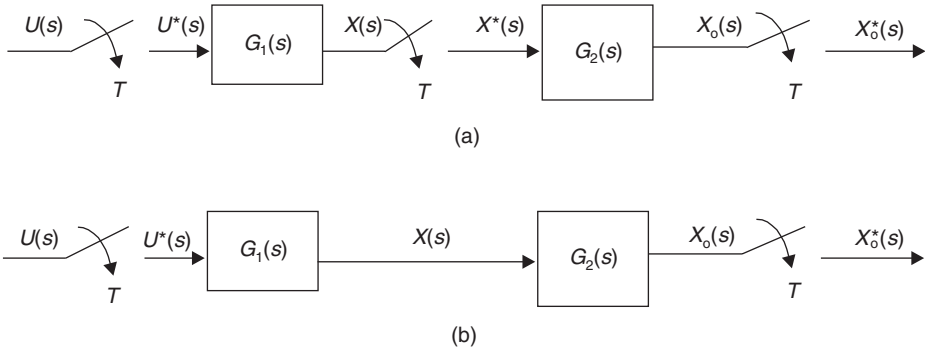


Fig. 7.9 Blocks in cascade.

synchronized sampler becomes $X_o^*(s)$. Figure 7.8(b) shows the pulse transfer function where $U(z)$ is equivalent to $U^*(s)$ and $X_o(z)$ is equivalent to $X_o^*(s)$.

From Figure 7.8(b) the pulse transfer function is

$$\frac{X_o}{U}(z) = G(z) \tag{7.27}$$

Blocks in Cascade: In Figure 7.9(a) there are synchronized samplers either side of blocks $G_1(s)$ and $G_2(s)$. The pulse transfer function is therefore

$$\frac{X_o}{U}(z) = G_1(z)G_2(z) \tag{7.28}$$

In Figure 7.9(b) there is no sampler between $G_1(s)$ and $G_2(s)$ so they can be combined to give $G_1(s)G_2(s)$, or $G_1G_2(s)$. Hence the output $X_o(z)$ is given by

$$X_o(z) = Z\{G_1G_2(s)\}U(z) \tag{7.29}$$

and the pulse transfer function is

$$\frac{X_o}{U}(z) = G_1G_2(z) \tag{7.30}$$

Note that $G_1(z)G_2(z) \neq G_1G_2(z)$.

Example 7.3 (See also Appendix 1, *examp73.m*)
A first-order sampled-data system is shown in Figure 7.10.
Find the pulse transfer function and hence calculate the response to a unit step and unit ramp. $T = 0.5$ seconds. Compare the results with the continuous system response $x_o(t)$. The system is of the type shown in Figure 7.9(b) and therefore

$$G(s) = G_1G_2(s)$$

Inserting values

$$G(s) = (1 - e^{-Ts})\left\{\frac{1}{s(s+1)}\right\} \tag{7.31}$$

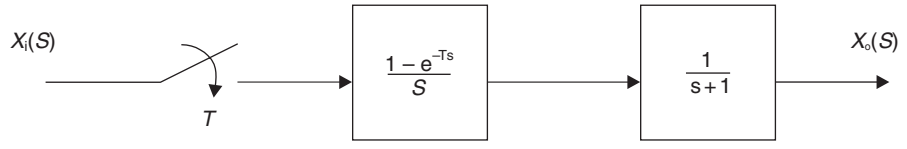


Fig. 7.10 First-order sampled-data system.

Taking z-transforms using Table 7.1.

$$G(z) = (1 - z^{-1})\left\{\frac{z(1 - e^{-T})}{(z - 1)(z - e^{-T})}\right\} \tag{7.32}$$

or

$$G(z) = \left(\frac{z - 1}{z}\right)\left\{\frac{z(1 - e^{-T})}{(z - 1)(z - e^{-T})}\right\} \tag{7.33}$$

which gives

$$G(z) = \left(\frac{1 - e^{-T}}{z - e^{-T}}\right) \tag{7.34}$$

For $T = 0.5$ seconds

$$G(z) = \left(\frac{0.393}{z - 0.607}\right) \tag{7.35}$$

hence

$$\frac{X_o}{X_i}(z) = \left(\frac{0.393z^{-1}}{1 - 0.607z^{-1}}\right) \tag{7.36}$$

which is converted into a difference equation

$$x_o(kT) = 0.607x_o(k - 1)T + 0.393x_i(k - 1)T \tag{7.37}$$

Table 7.2 shows the discrete response $x_o(kT)$ to a unit step function and is compared with the continuous response (equation 3.29) where

$$x_o(t) = (1 - e^{-t}) \tag{7.38}$$

From Table 7.2, it can be seen that the discrete and continuous step response is identical. Table 7.3 shows the discrete response $x(kT)$ and continuous response $x(t)$ to a unit ramp function where $x_o(t)$ is calculated from equation (3.39)

$$x_o(t) = t - 1 + e^{-t} \tag{7.39}$$

In Table 7.3 the difference between $x_o(kT)$ and $x_o(t)$ is due to the sample and hold. It should also be noted that with the discrete response $x(kT)$, there is only knowledge of the output at the sampling instant.

Table 7.2 Comparison between discrete and continuous step response

k	kT (seconds)	$x_i(kT)$	$x_o(kT)$	$x_o(t)$
−1	−0.5	0	0	0
0	0	1	0	0
1	0.5	1	0.393	0.393
2	1.0	1	0.632	0.632
3	1.5	1	0.776	0.776
4	2.0	1	0.864	0.864
5	2.5	1	0.918	0.918
6	3.0	1	0.950	0.950
7	3.5	1	0.970	0.970
8	4.0	1	0.982	0.982

Table 7.3 Comparison between discrete and continuous ramp response

k	kT (seconds)	$x_i(kT)$	$x_o(kT)$	$x_o(t)$
−1	−0.5	0	0	0
0	0	0	0	0
1	0.5	0.5	0	0.107
2	1.0	1.0	0.304	0.368
3	1.5	1.5	0.577	0.723
4	2.0	2.0	0.940	1.135
5	2.5	2.5	1.357	1.582
6	3.0	3.0	1.805	2.050
7	3.5	3.5	2.275	2.530
8	4.0	4.0	2.757	3.018

7.4.3 The closed-loop pulse transfer function

Consider the error sampled system shown in Figure 7.11. Since there is no sampler between $G(s)$ and $H(s)$ in the closed-loop system shown in Figure 7.11, it is a similar arrangement to that shown in Figure 7.9(b). From equation (4.4), the closed-loop pulse transfer function can be written as

$$\frac{C}{R}(z) = \frac{G(z)}{1 + GH(z)} \tag{7.40}$$

In equation (7.40)

$$GH(z) = Z\{GH(s)\} \tag{7.41}$$

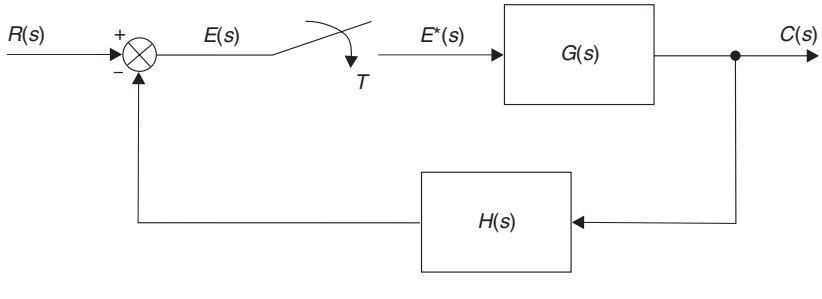


Fig. 7.11 Closed-loop error sampled system.

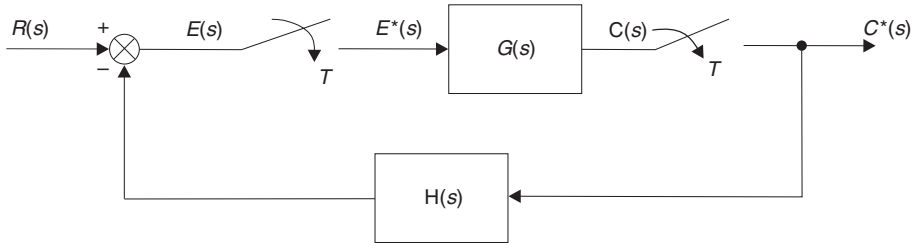


Fig. 7.12 Closed-loop error and output sampled system.

Consider the error and output sampled system shown in Figure 7.12. In Figure 7.12, there is now a sampler between $G(s)$ and $H(s)$, which is similar to Figure 7.9(a). From equation (4.4), the closed-loop pulse transfer function is now written as

$$\frac{C}{R}(z) = \frac{G(z)}{1 + G(z)H(z)} \tag{7.42}$$

7.5 Digital control systems

From Figure 7.1, a digital control system may be represented by the block diagram shown in Figure 7.13.

Example 7.4 (See also Appendix 1, *examp74.m*)
Figure 7.14 shows a digital control system. When the controller gain K is unity and the sampling time is 0.5 seconds, determine

(a) the open-loop pulse transfer function
(b) the closed-loop pulse transfer function
(c) the difference equation for the discrete time response
(d) a sketch of the unit step response assuming zero initial conditions
(e) the steady-state value of the system output

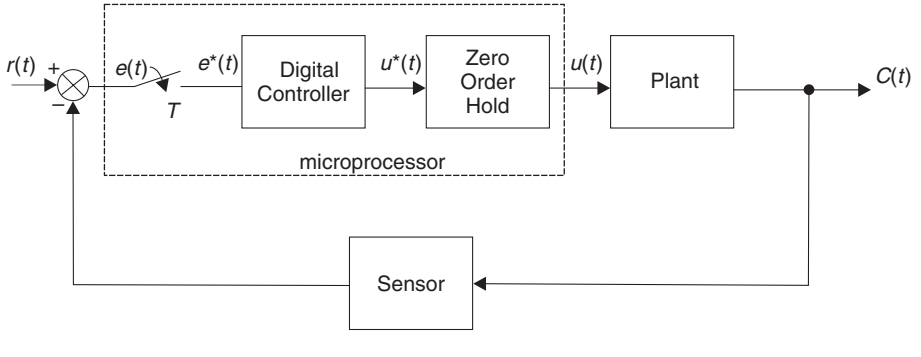


Fig. 7.13 Digital control system.

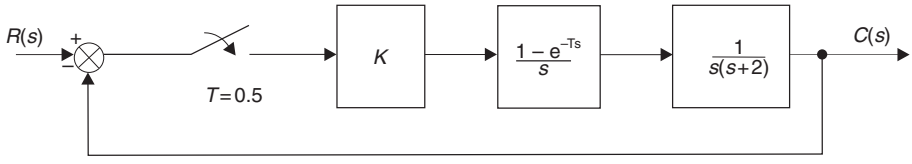


Fig. 7.14 Digital control system for Example 7.4.

Solution

$$\text{(a)} \quad G(s) = K\left(\frac{1-\mathrm{e}^{-Ts}}{s}\right)\left\{\frac{1}{s(s+2)}\right\} \tag{7.43}$$

Given $K = 1$

$$G(s) = \left(1-\mathrm{e}^{-Ts}\right)\left\{\frac{1}{s^2(s+2)}\right\} \tag{7.44}$$

Partial fraction expansion

$$\frac{1}{s^2(s+2)} = \left\{\frac{A}{s} + \frac{B}{s^2} + \frac{C}{(s+2)}\right\} \tag{7.45}$$

or

$$1 = s(s+2)A + (s+2)B + s^2\mathrm{C} \tag{7.46}$$

Equating coefficients gives

$$A = -0.25$$
$$B = 0.5$$
$$C = 0.25$$

Substituting these values into equation (7.44) and (7.45)

$$G(s) = \left(1-\mathrm{e}^{-Ts}\right)\left\{\frac{-0.25}{s} + \frac{0.5}{s^2} + \frac{0.25}{(s+2)}\right\} \tag{7.47}$$

or

$$G(s) = 0.25\left(1-\mathrm{e}^{-Ts}\right)\left\{-\frac{1}{s} + \frac{2}{s^2} + \frac{1}{(s+2)}\right\} \tag{7.48}$$

Taking z-transforms

$$G(z) = 0.25\left(1-z^{-1}\right)\left\{\frac{-z}{(z-1)} + \frac{2Tz}{(z-1)^2} + \frac{z}{(z-\mathrm{e}^{-2T})}\right\} \tag{7.49}$$

Given $T = 0.5$ seconds

$$G(z) = 0.25\left(\frac{z-1}{z}\right)z\left\{\frac{-1}{(z-1)} + \frac{2\times 0.5}{(z-1)^2} + \frac{1}{(z-0.368)}\right\} \tag{7.50}$$

Hence

$$G(z) = 0.25(z-1)\left\{\frac{-1(z-1)(z-0.368)+(z-0.368)+(z-1)^2}{(z-1)^2(z-0.368)}\right\} \tag{7.51}$$

$$G(z) = 0.25\left\{\frac{-z^2+1.368z-0.368+z-0.368+z^2-2z+1}{(z-1)(z-0.368)}\right\} \tag{7.52}$$

which simplifies to give the open-loop pulse transfer function

$$G(z) = \left(\frac{0.092z+0.066}{z^2-1.368z+0.368}\right) \tag{7.53}$$

Note: This result could also have been obtained at equation (7.44) by using z-transform number 7 in Table 7.1, but the solution demonstrates the use of partial fractions.

(b) The closed-loop pulse transfer function, from equation (7.40) is

$$\frac{C}{R}(z) = \frac{\left(\frac{0.092z+0.066}{z^2-1.368z+0.368}\right)}{\left(1+\frac{0.092z+0.066}{z^2-1.368z+0.368}\right)} \tag{7.54}$$

which simplifies to give the closed-loop pulse transfer function

$$\frac{C}{R}(z) = \frac{0.092z+0.066}{z^2-1.276z+0.434} \tag{7.55}$$

or

$$\frac{C}{R}(z) = \frac{0.092z^{-1}+0.066z^{-2}}{1-1.276z^{-1}+0.434z^{-2}} \tag{7.56}$$

(c) Equation (7.56) can be expressed as a difference equation

$$c(kT) = 1.276c(k-1)T - 0.434c(k-2)T + 0.092r(k-1)T + 0.066r(k-2)T \tag{7.57}$$

(d) Using the difference equation (7.57), and assuming zero initial conditions, the unit step response is shown in Figure 7.15.

Note that the response in Figure 7.15 is constructed solely from the knowledge of the two previous sampled outputs and the two previous sampled inputs.

(e) Using the final value theorem given in equation (7.14)

$$c(\infty) = \lim_{z\to 1}\left[\left(\frac{z-1}{z}\right)\frac{C}{R}(z)R(z)\right] \tag{7.58}$$

$$c(\infty) = \lim_{z\to 1}\left[\left(\frac{z-1}{z}\right)\left\{\frac{0.092z+0.066}{1-1.276z+0.434}\right\}\left(\frac{z}{z-1}\right)\right] \tag{7.59}$$

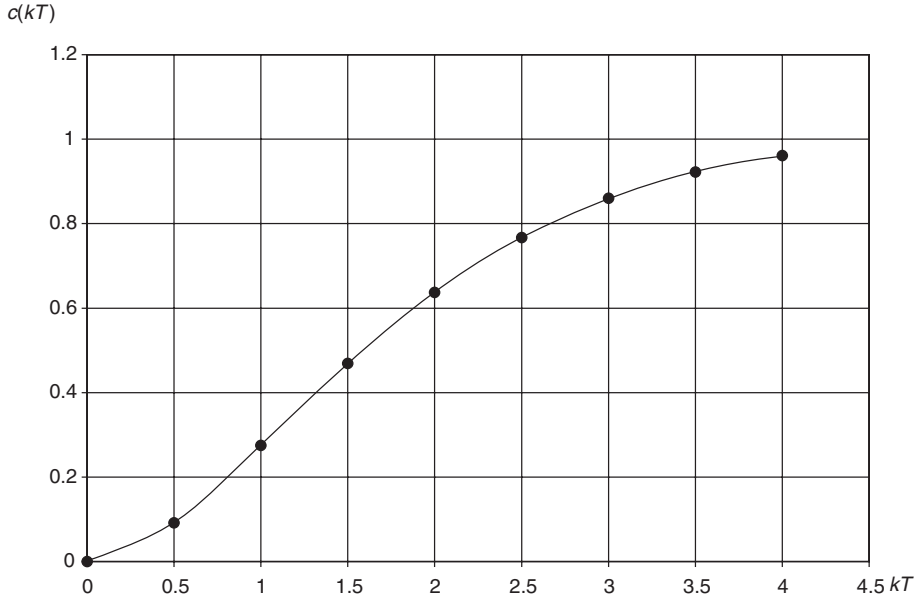


Fig. 7.15 Unit step response for Example 7.4.

$$c(\infty) = \left(\frac{0.092 + 0.066}{1 - 1.276 + 0.434}\right) = 1.0 \tag{7.60}$$

Hence there is no steady-state error.

7.6 Stability in the *z*-plane

7.6.1 Mapping from the *s*-plane into the *z*-plane

Just as transient analysis of continuous systems may be undertaken in the s-plane, stability and transient analysis on discrete systems may be conducted in the z-plane.

It is possible to map from the s to the z-plane using the relationship

$$z = \mathrm{e}^{sT} \tag{7.61}$$

now

$$s = \sigma \pm \mathrm{j}\omega$$

therefore

$$z = \mathrm{e}^{(\sigma \pm \mathrm{j}\omega)T} = \mathrm{e}^{\sigma T}\mathrm{e}^{\mathrm{j}\omega T} \quad \text{(using the positive j}\omega\text{ value)} \tag{7.62}$$

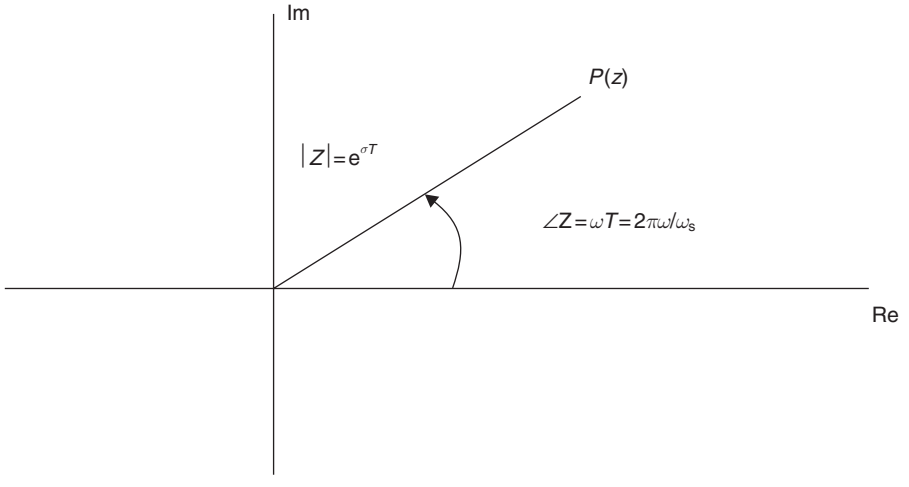


Fig. 7.16 Mapping from the *s* to the *z*-plane.

If $e^{\sigma T} = |z|$ and $T = 2\pi/\omega_s$ equation (7.62) can be written

$$z = |z| e^{j(2\pi\omega/\omega_s)} \tag{7.63}$$

where ω_s is the sampling frequency.

Equation (7.63) results in a polar diagram in the *z*-plane as shown in Figure 7.16. Figure 7.17 shows mapping of lines of constant σ (i.e. constant settling time) from the *s* to the *z*-plane. From Figure 7.17 it can be seen that the left-hand side (stable) of the *s*-plane corresponds to a region within a circle of unity radius (the unit circle) in the *z*-plane.

Figure 7.18 shows mapping of lines of constant ω (i.e. constant transient frequency) from the *s* to the *z*-plane.

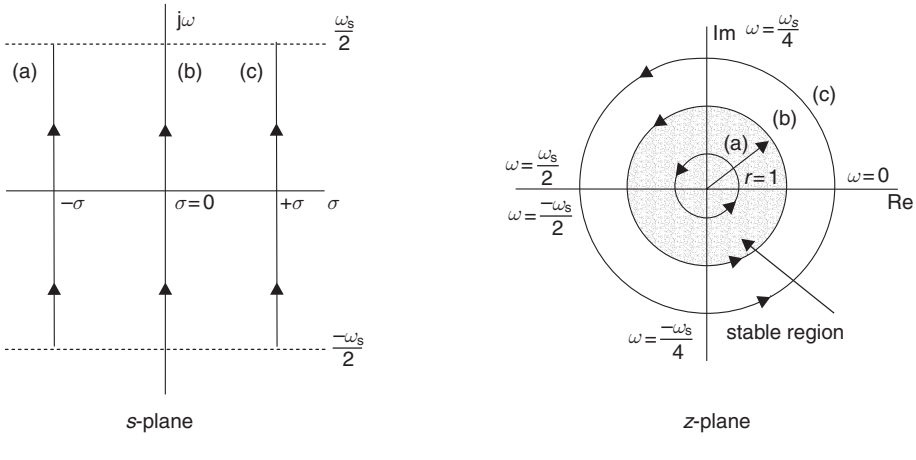


Fig. 7.17 Mapping constant σ from *s* to *z*-plane.

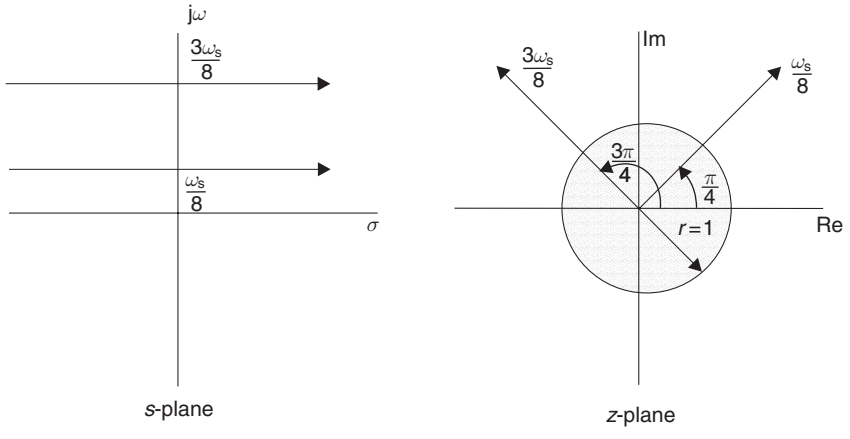


Fig. 7.18 Mapping constant ω from s to z-plane.

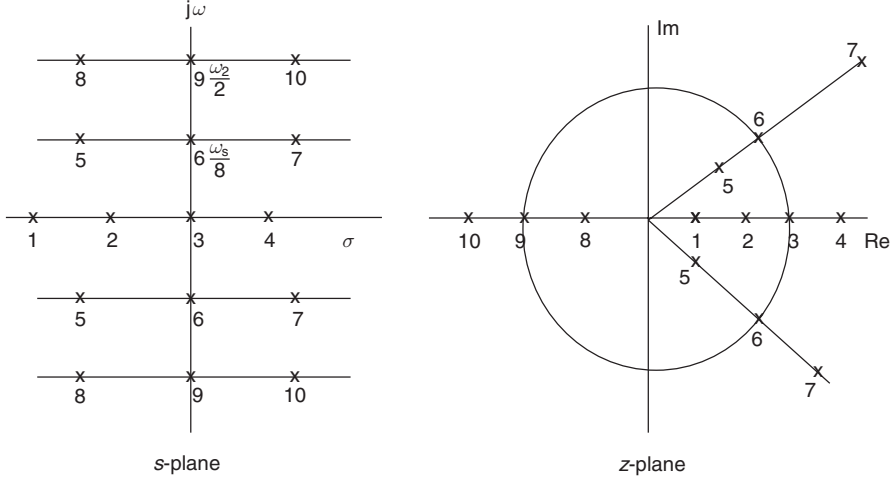


Fig. 7.19 Corresponding pole locations on both s and z-planes.

Figure 7.19 shows corresponding pole locations on both the s-plane and z-plane.

7.6.2 The Jury stability test

In the same way that the Routh–Hurwitz criterion offers a simple method of determining the stability of continuous systems, the Jury (1958) stability test is employed in a similar manner to assess the stability of discrete systems.

Consider the characteristic equation of a sampled-data system

$$Q(z) = a_n z^{\mathrm{n}} + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0 = 0 \tag{7.64}$$

Table 7.4 Jury's array

z^0	z^1	z^2		z^{n-1}	z^n
a_0	a_1	a_2	...	a_{n-1}	a_n
a_n	a_{n-1}	a_{n-2}	...	a_1	a_0
b_0	b_1	b_2	...	b_{n-1}	
b_{n-1}	b_{n-2}	b_{n-3}	...	b_0	
.					
.					
.					
l_0	l_1	l_2	...	l_3	
l_3	l_2	l_1	...	l_0	
m_0	m_1	m_2			
m_2	m_1	m_0			

The array for the Jury stability test is given in Table 7.4 where

$$b_k = \begin{vmatrix} a_0 & a_{n-k} \\ a_n & a_k \end{vmatrix}$$

$$c_k = \begin{vmatrix} b_0 & b_{n-1-k} \\ b_{n-1} & b_k \end{vmatrix} \tag{7.65}$$

$$d_k = \begin{vmatrix} c_0 & c_{n-2-k} \\ c_{n-2} & c_k \end{vmatrix}$$

The necessary and sufficient conditions for the polynomial $Q(z)$ to have no roots outside or on the unit circle are

$$\begin{array}{ll} \text{Condition 1} & Q(1) > 0 \\ \text{Condition 2} & (-1)^n Q(-1) > 0 \\ \text{Condition 3} & |a_0| < a_n \\ \vdots & |b_0| > |b_{n-1}| \\ \vdots & |c_0| > |c_{n-2}| \\ & \vdots \\ \text{Condition } n & |m_0| > |m_2| \end{array} \tag{7.66}$$

Example 7.5 (See also Appendix 1, *examp75.m*)
For the system given in Figure 7.14 (i.e. Example 7.4) find the value of the digital compensator gain K to make the system just unstable. For Example 7.4, the characteristic equation is

$$1 + G(z) = 0 \tag{7.67}$$

In Example 7.4, the solution was found assuming that $K = 1$. Therefore, using equation (7.53), the characteristic equation is

$$1 + \frac{K(0.092z + 0.066)}{(z^2 - 1.368z + 0.368)} = 0 \tag{7.68}$$

or

$$Q(z) = z^2 + (0.092K - 1.368)z + (0.368 + 0.066K) = 0 \tag{7.69}$$

The first row of Jury's array is

z^0	z^1	z^2
$(0.368 + 0.066K)$	$(0.092K - 1.368)$	1

(7.70)

Condition 1: $Q(1) > 0$
From equation (7.69)

$$Q(1) = \{1 + (0.092K - 1.368) + (0.368 + 0.066K)\} > 0 \tag{7.71}$$

From equation (7.71), $Q(1) > 0$ if $K > 0$.

Condition 2 $(-1)^n Q(-1) > 0$
From equation (7.69), when $n = 2$

$$(-1)^2 Q(-1) = \{1 - (0.092K - 1.368) + (0.368 + 0.066K)\} > 0 \tag{7.72}$$

Equation (7.72) simplifies to give

$$2.736 - 0.026K > 0$$

or

$$K < \frac{2.736}{0.026} = 105.23 \tag{7.73}$$

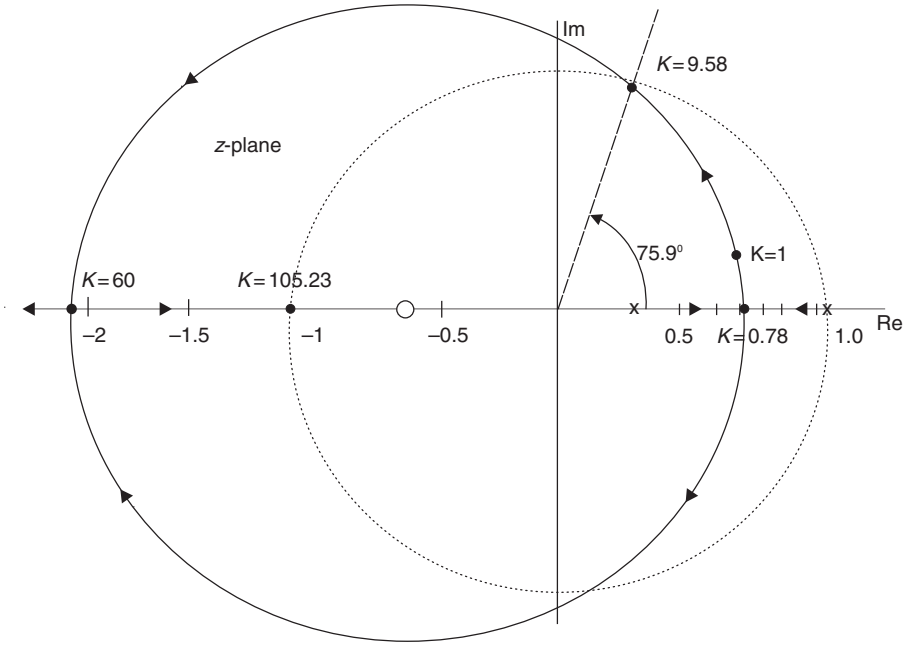


Fig. 7.20 Root locus diagram for Example 7.4.

Condition 3: $|a_0| < a_2$

$$|0.368 + 0.066K| < 1 \tag{7.74}$$

For marginal stability

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

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

$$1 + GH(z) = 0 \tag{7.76}$$

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

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

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)

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

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

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

Open-loop poles

$$z^2 - 1.368z + 0.368 = 0 \tag{7.79}$$

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

Open-loop zeros

$$0.092z + 0.066 = 0$$

$$z = -0.717 \tag{7.81}$$

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

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

Breakaway points: Using Rule 6

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

$$(z^2 - 1.368z + 0.368)K(0.092) - K(0.092z + 0.066)(2z - 1.368) = 0 \tag{7.83}$$

which gives

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

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)

$$K = 9.58 \text{ and } 105.23 \tag{7.85}$$

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

$$z^2 - 0.487z + 1 = 0 \tag{7.86}$$

The roots of equation (7.86) are

$$z = 0.244 \pm j0.97 \tag{7.87}$$

or

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

Since from equation (7.63) and Figure 7.16

$$z = |z| \angle \omega T \tag{7.89}$$

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

$$\begin{aligned} 0.5\omega &= 1.33 \\ \omega &= 2.66\,\text{rad/s} \end{aligned} \tag{7.90}$$

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

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

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

$$z = 0.638 \pm \text{j}0.164$$

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

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

and the closed-loop pulse transfer function become

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

and hence the characteristic equation is

$$1 + D(z)GH(z) = 0 \tag{7.94}$$

7.7.1 Digital compensator types

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

$$u(t) = \frac{\mathrm{d}e}{\mathrm{d}t}$$

Taking Laplace transforms with zero initial conditions

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

In a discrete system, a differentiation can be approximated to

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

hence

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

Hence, the Laplace operator can be approximated to

$$s = \frac{1 - z^{-1}}{T} = \frac{z - 1}{Tz} \tag{7.97}$$

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

$$\frac{U}{E}(s) = \frac{K_1(T_\mathrm{i}T_\mathrm{d}s^2 + T_\mathrm{i}s + 1)}{T_\mathrm{i}s} \tag{7.98}$$

Inserting equation (7.97) into (7.98) gives

$$\frac{U}{E}(z) = \frac{K_1\left\{T_\mathrm{i}T_\mathrm{d}\left(\frac{z-1}{Tz}\right)^2 + T_\mathrm{i}\left(\frac{z-1}{Tz}\right) + 1\right\}}{T_\mathrm{i}\left(\frac{z-1}{Tz}\right)} \tag{7.99}$$

which can be simplified to give

$$\frac{U}{E}(z) = \frac{K_1(b_2z^2 + b_1z + b_0)}{z(z-1)} \tag{7.100}$$

where

$$\begin{aligned} b_0 &= \frac{T_\mathrm{d}}{T} \\ b_1 &= \left(1 - \frac{2T_\mathrm{d}}{T}\right) \\ b_2 &= \left(\frac{T_\mathrm{d}}{T} + \frac{T}{T_\mathrm{i}} + 1\right) \end{aligned} \tag{7.101}$$

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

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

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

$$\begin{aligned} b_0 &= \frac{T_\mathrm{d}}{T} \\ b_1 &= \left(\frac{T}{2T_\mathrm{i}} - \frac{2T_\mathrm{d}}{T} - 1\right) \\ b_2 &= \left(\frac{T}{2T_\mathrm{i}} + \frac{T_\mathrm{d}}{T} + 1\right) \end{aligned} \tag{7.103}$$

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

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

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

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

(a) Find the z-transform of the missile by selecting a sampling frequency of at least 10 times higher than the system bandwidth.
(b) 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.
(c) 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.
(d) 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
(a) 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

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

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

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

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

(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

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

(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

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

(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)

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

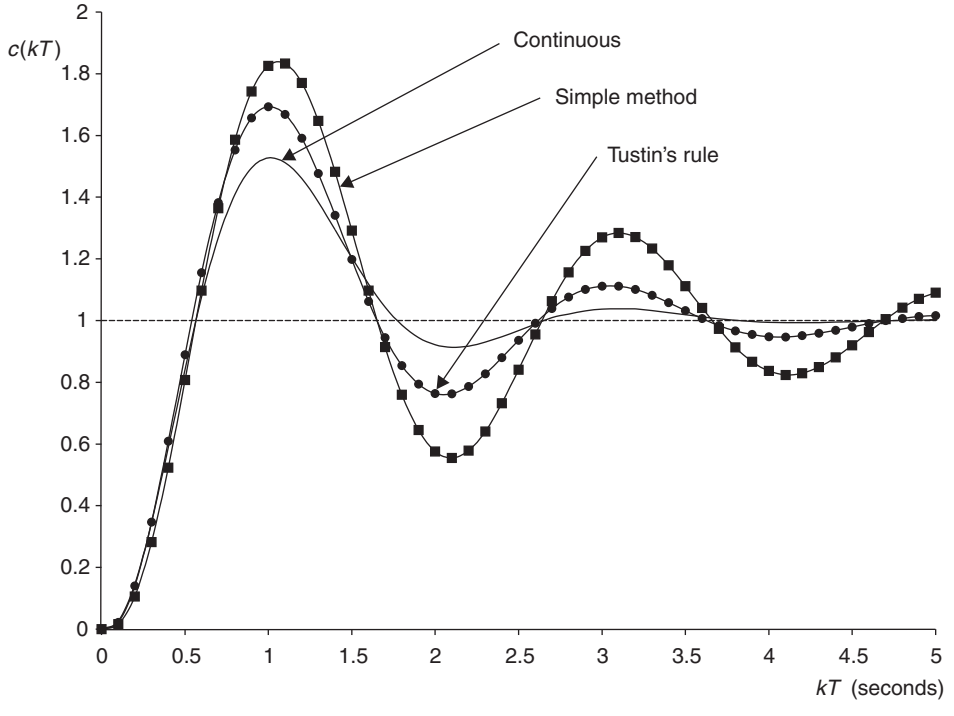


Fig. 7.21 Comparison between discrete and continuous response.

Hence the difference equation for the digital compensator is

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

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).

(a) 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.
(b) 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?
(c) 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.
(d) With $D(z)$ set to the value of K found in (a), compare the continuous and discrete step responses.

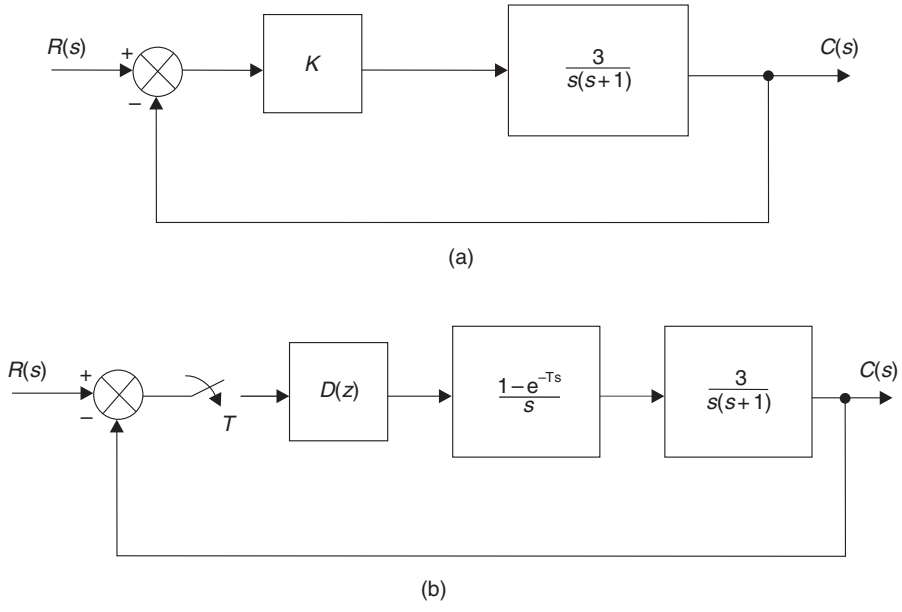


Fig. 7.22 Continuous and digital control systems.

(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 in shown in Figure 7.23. From Figure 7.23 the closed-loop poles are

$$s = -0.5 \pm \mathrm{j}0.866 \tag{7.113}$$

or

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

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

Using transform 7 in Table 7.1

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

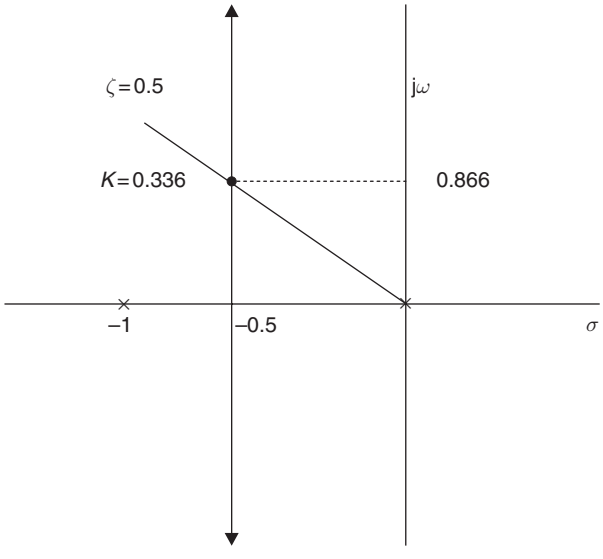


Fig. 7.23 Root locus diagram for continuous system.

Hence

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

(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

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

inserting values

$$|z| = \mathrm{e}^{-0.5 \times 0.5} = 0.779 \tag{7.116}$$

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

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

$$z = 0.707 \pm \mathrm{j}0.327 \tag{7.118}$$

which provides a z-plane characteristic equation

$$z^2 - 1.414z + 0.607 = 0 \tag{7.119}$$

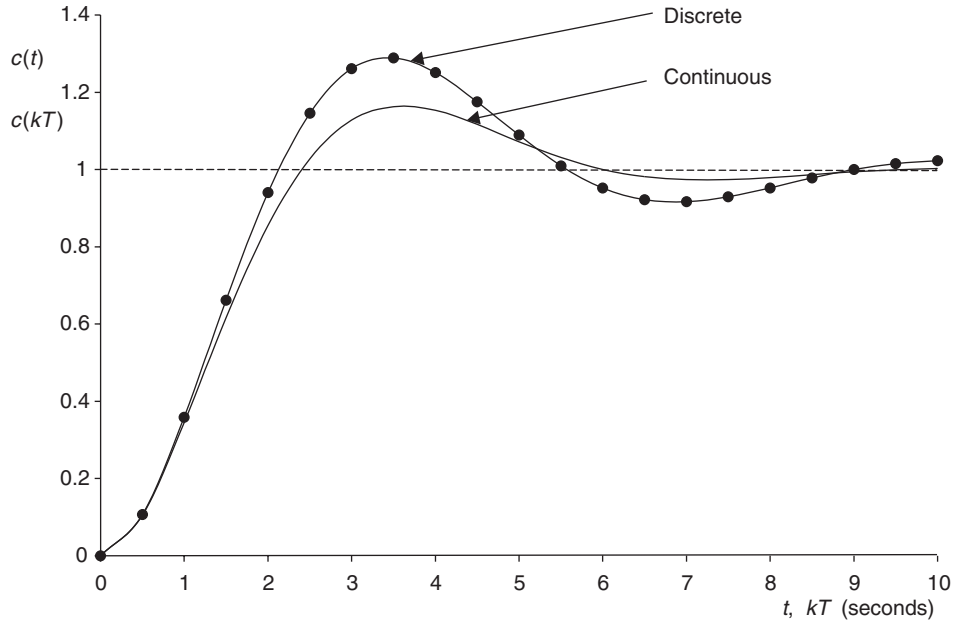


Fig. 7.24 Continuous and digital controllers set to $K = 0.336$.

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

$$1 + D(z)G(z) = 0 \tag{7.120}$$

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

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

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

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

Hence the characteristic equation (7.120) becomes

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

which simplifies to give

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

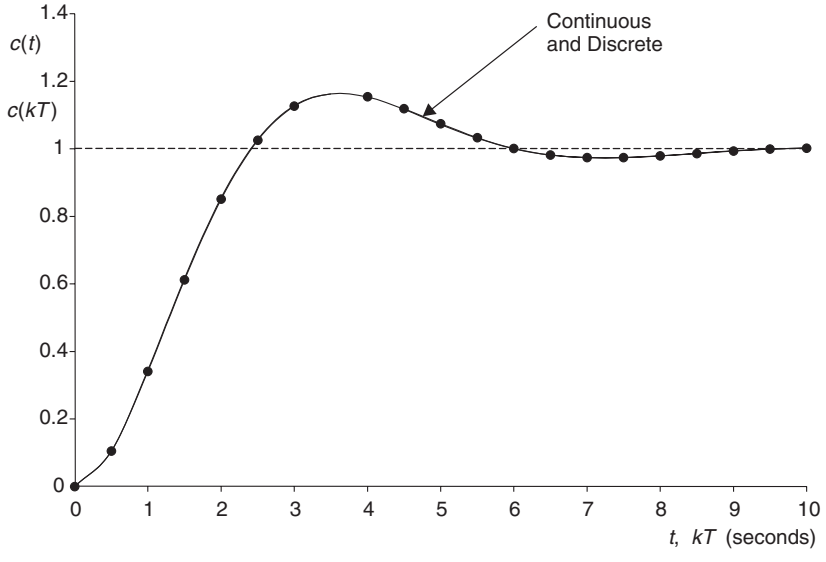


Fig. 7.25 Identical continuous and discrete step responses as a result of pole placement.

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

$$0.3196K + b - 1 = -1.414 \tag{7.124}$$

$$0.2706K - b = 0.607 \tag{7.125}$$

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

or

$$0.5902K = 0.193$$

$$K = 0.327 \tag{7.126}$$

Inserting equation (7.126) into (7.125)

$$(0.2706 \times 0.327) - 0.607 = b$$

$$b = -0.519 \tag{7.127}$$

Thus the required compensator is

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

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

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

hence

$$u(kT) = 0.327\mathrm{e}(kT) - 0.1983\mathrm{e}(k-1)T + 0.519u(k-1)T \tag{7.130}$$

Introduction to Real-time System

The earliest proposal to use a computer operating in "real time" as part of a control system was made at 1950 by Brown and Campbell. They are assumed that analog-computing element.

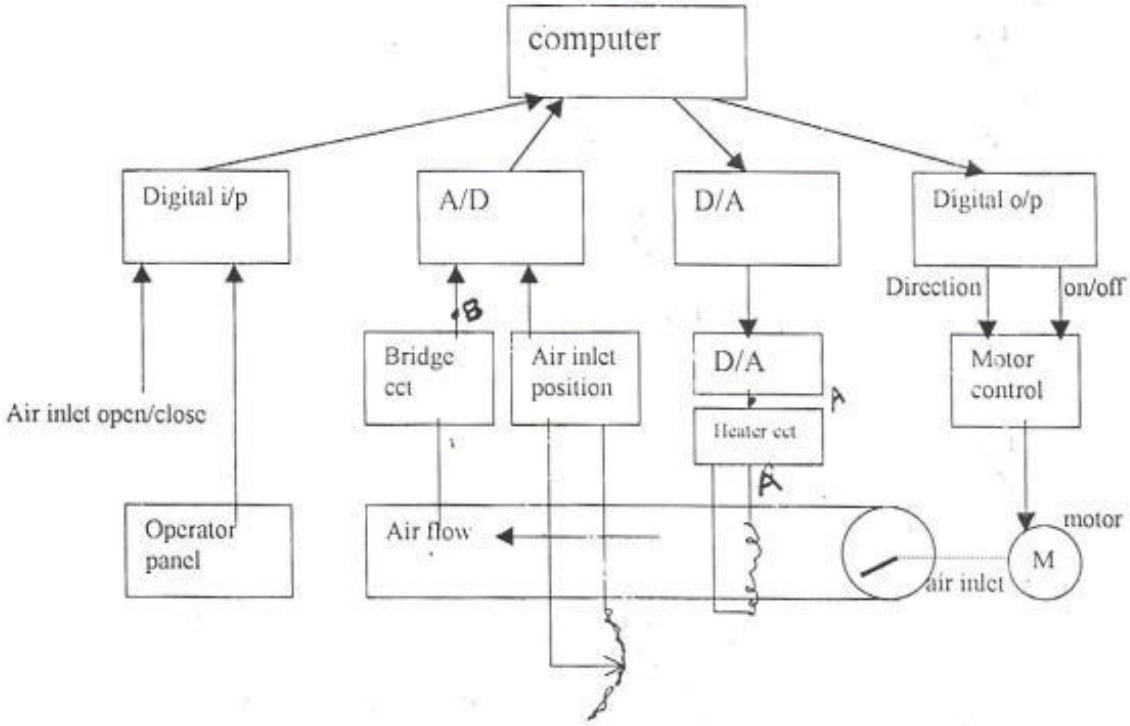
The first digital computers developed specifically for real –time control were for airborne operation, and in 1954 digital computer was successfully used to provide an automatic flight and weapons control system.

Element of a Computer Control System: -

As an example we shall consider a simple plant a "hot-air blower " as shown in figure bellow.

A Fan blows air over a heating element and into a tube. A thermistor bead (sensor) is placed at the outlet of the tube and forms one arm at a bridge circuit. The amplified output of the bridge circuit is available at B and provides a voltage, in the range (0 – 10) volt, proportional to temperature. The current supplied to the heating element can be varied by supplying a dc voltage in the range (0 - 10) volt to point A.

The position of the air-infect cover to the fan is adjusted by means of a reversible motor. The motor operates at constant speed and it turned ON/OFF by a logic signal applied to its controller ; a second logic signal determines the direction of rotation. A potentiometer wiper is attached to the air-inlet cover and the voltage output is proportional to the position of cover. Micro switches are used to detected when the cover is fully open and fully closed.



computer control of hot-air blower

The operation is provided with a panel from which the control system can be switched from auto to manual. Panel lights indicate " fan on ", "heater on", "cover fully open ", " cover fully closed " and "auto/manual" status.

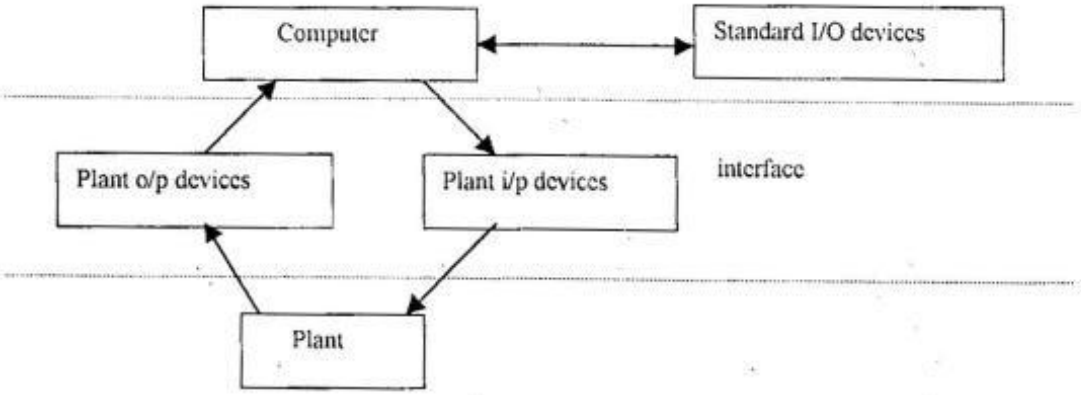
The operation of this simple plant by a computer requires *monitoring, control calculation and actuation.*

Monitoring involves obtaining information about the current state of the plant. In the above example the information is available from the plant instruments in the form of analog signals for air temperature and fan-inlet cover position; in the form of digital (logic) signals for extremes of fan-inlet cover position (fully open, fully closed); and the various other status signals; auto/manual, fan motor on, heater on.

The control calculation involve the digital equivalent of continuous feedback control for the control of temperature (Direct Digital Control – DDC). There is feedback-position control for fan-inlet cover position and there is sequence and interlock control: the heater should not be on if the fan is not running; automatic change from tracking to controlling when the operator changes from manual to auto.

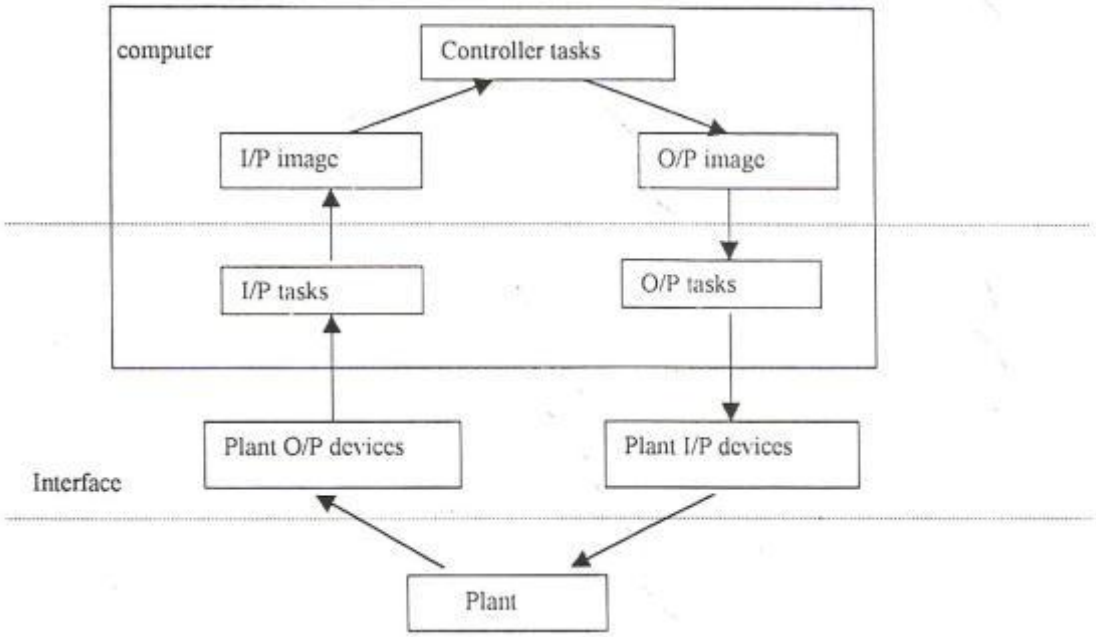
The actuation requires the provision of a voltage proportional to the demand heat output for the heater control, logic signals indicating ON/OFF and direction of the fan-inlet cover and logic signals for the operation display.

The monitoring and actuation tasks thus involve a range of interface device including A/D , D/A ,digital input and output lines and pulse generators. We will represent them simply as *input and output devices.* The generalized computer control as shown in figure bellow.



Generalized computer control system

Each of the various types of device will required software to operate it; again we will represent this software as *input and output tasks.* The generalized picture of a computer control system is shown bellow

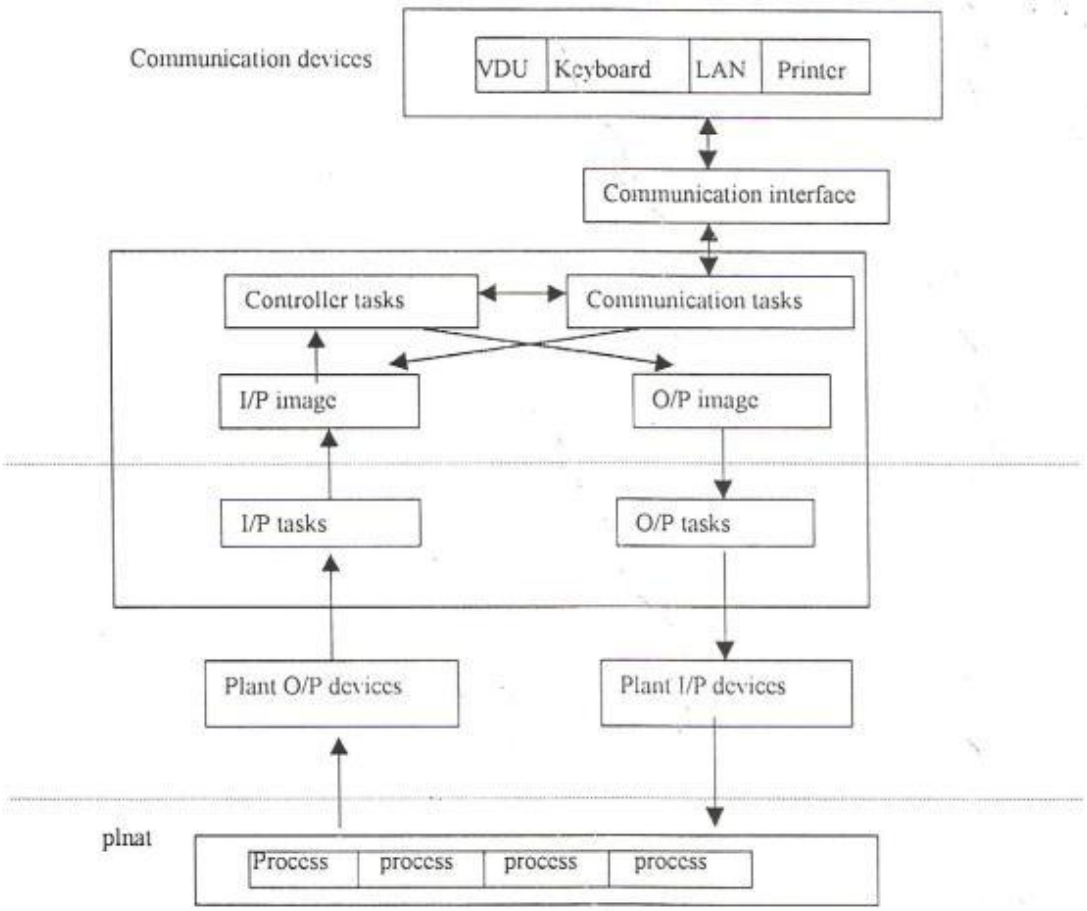


Generalized computer control system (hardware & software)

The input devices plus the input software provide the information to create "*input image* " of the plant. The input image is a snapshot of the status of the plant and this snapshot is renewed at specified interval.

The output image represents the current set of outputs generated by the control calculations. The output image will be updated periodically by control tasks. It is the job of the output task to convey the output image to the plant. The control tasks can thus be considered as operating on an internal image (or model) of the plant.

The simple model of computer control described above divides the tasks to be performed into three major areas: plant input tasks, plant output tasks and control tasks; and it is assumed that communication with the operator is treated as part of the plant input and plant output tasks. However, in many applications communication will extend beyond simple indication and switches. Control of the system may be shared between several computers and hence information may have to be transmitted between computers. The model must therefore be extended to included communication tasks as shown in figure.



computer control system showing communication tasks

Classification of Real-time system

The real time system will be used to refer to systems in which:

1- the order of computation is determined by the passage of the time or by events external to the computer; and

2- the results of the particular calculation may depend upon the value of the variable ' time ' at the instance of execution of the caculations or the time taken to execute the computation.

clock- based system: -

A process plant operates in real time and thus we talk a bout the plant time constants, these may be measured in hours (chemical process) or millisecond (aircraft system). For feedback control system the required sampling rate will be dependent on the time constant of the process to be controlled. The shorter the time constant of the process, the faster the required sampling rate.

The computer which is used to control the plant must therefore be synchronized to real time or natural time and must be able to carry out all the required operations measurement, control and actuation within each sampling interval.

The completion of the operation within the specified time is dependent on the number of operation to be performed and the speed of the computer. Synchronization is usually obtained by adding to the computer system a clock – normally referred to a "real – time " clock and using a signal from this clock to interrupt the operations of the computer at some predetermined fixed time interval.

2/ **Sensor-based systems:-**

The actions of the system will be not at particular times or time interval, but in response to some event. (Ex. turn off a pump when the level in a liquid tank reaches a predetermined value).

Sensor _based systems are used extensively to indicate alarm conditions and initiate alarm actions. (Ex. too high temperature or too great pressure).

Sensor based systems normally employ interrupts to inform the computer system that action is required. Some small, simple systems may use polling: that is, the computer periodically asks (polls) the various sensors to see if action is required.

3/ **Interactive system**

Interactive system probably represent the largest class of real time system and cover such systems as automatic bank tellers, reservation systems for hotels, etc. The real time requirement is usually expressed in terms of the average response time not exceeding a specified value. For example, an automatic bank teller system might require an averge response time not exceeding 20 sec. Although this type of system superficially seems similar to the sensor based system – that is, it apparently responds to a signal from the plant (in this case usually a person) – it is different in that it responds at a time determined by the internal state of the computer.

Classification of Programs

1) **Sequential "single- tasking"**

In classical sequential programming action are strictly ordered as a time sequence the behavior of the program depends only on the effects of the individual actions and their order, the time taken to perform the action of consequence. Therefore, requires two kinds of argument:

1- that a particular statement defines a stated action

2- that the various program structures produce a stated sequence of events.

2) **Multi –Tasking**

The design and programming of large real-time (and large interactive) systems can be considerably eased if the background portioning can be extended into multiple partitions to allow the concept of many active task.

The implication of this approach are that each task may be carried in parallel and there is no assumption made at the design stage as to how many processors will be used in the system.

The system must be :-

1-create separate tasks.
2-schedule running at the tasks.(on a priority basis)
3-share data between tasks.
4-synchronize tasks, with each other
5-prevent tasks corrupting each other
6-control the starting and stopping of tasks.

Concepts of computer control

أنشطة

There are several different computer activities carried out :-

1- Sequence control.
2- Loop control.
3- Supervisory control.
4- Data acquisition.
5- Data analysis.
6- Human interfacing (MMI).

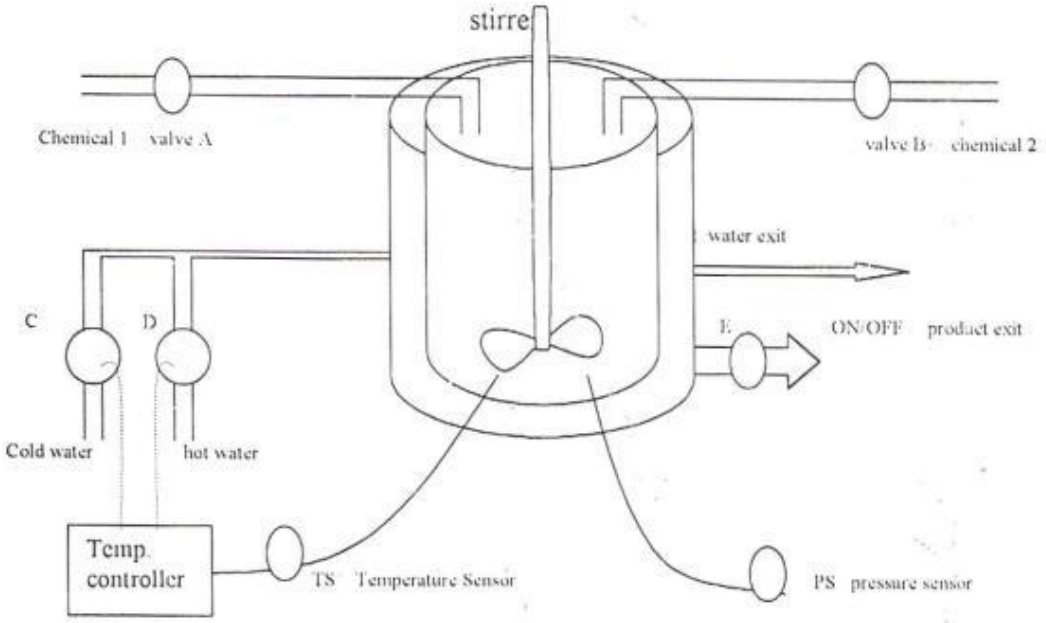
The overcall objectives of the control of these processes can by summarized as :-

- Safety,
- Product specification,
- Environment البيئة المحيطة
- Ease of operating and
- Economics.

Sequence Control

Although Sequence control will occur in same part most systems it often predominates in batch systems and hence a batch system is used to illustrate it , batch system are widely used in the load-processing and chemical industries where the operations carried out frequently involve mixing raw materials , carrying out some process and then discharging the product .

Typical example of simple complete processing plant is shown in figure bellow:



In this plant a chemical is produced by the reaction of two other chemicals at a specified temperature. The chemicals are mixed together in a sealed vessel (the reactor) and the temperature of the reaction is controlled by feeding hot or cold water through the water jacket which surrounds the vessel. The water flow is controlled by adjusting valves C and D. The flow if material into and out of the vessel is regulated by the valve A , B and E . The temperature of the contents of the vessel and the pressure in the vessel are monitored

The procedure for the operation of the system as follows:-

1. open valve "A" .(To charge the Vessel with chemical 1).
2. check the level of the chemical , (by monitoring the pressure in the vessel), when correct amount of chemical has been admitted , Close valve "A".
3. start the stirrer.
4. repeat stage "1 & 2" with value "B" (to admitted the second chemical).
5. switch ON the three-term controller and supply a set point so that the chemical mixed is heated up to the required reaction temperature.
6. Monitor the reaction temperature; when it reaches the set point, start a timer to time the duration of the reaction
7. when the timer indicates that the reaction is complete switch off the controller and open value "C" cool down the reactor contents. Switch off the stirrer.
8. monitor the temperature; when the contents have cooled ,open value "E" to remove the product from reactor.

When implemented by computer all of the above actions and timings would be based upon software.

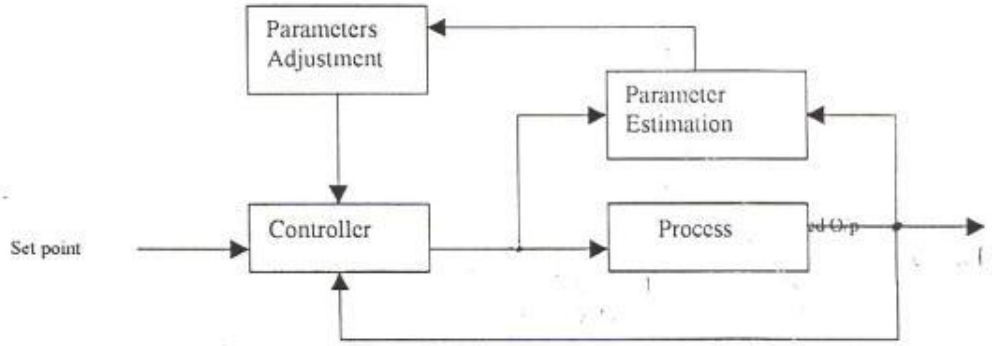
Loop control (Direct digital control (DDC)

In DDC the computer is in the feedback loop. A consequence of this is that the computer becomes a critical component and great care is needed to ensure that ,in the event of the failure or multifunctioning of the computer ,the plant remains in a safe condition.
The advantages claimed for DDC over analog control are :-

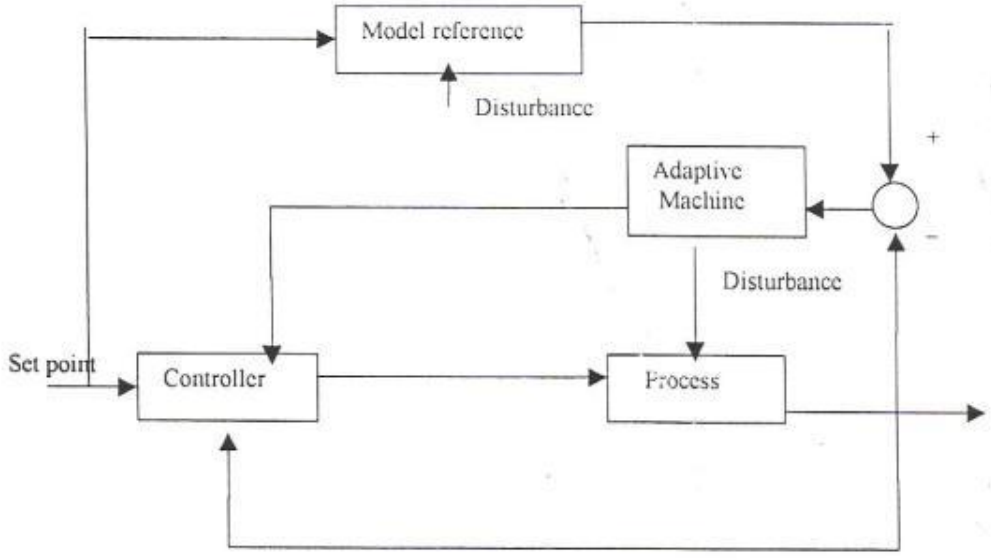
- Cost , in the early days the break even point was between 50-100 loops ; with the introduction of Microprocessors a single loop DDC unit can be cheaper then an the analog unit.
- Performance. Digital control makes it possible to use improves the accuracy of the controller and provides a wider range of control settings.

DDC control may be applied either to a single loop system implemented on a small microprocessor or to a large system involving several hundred loops . the loops may be cascaded that is , with the output or actuation signal of one loop acting as the set point for another loop. Signals added together (rate loop) and conditional switches may be used to alter signal connections

The latter method is used in aircraft control, for example, when control parameters are changed with alterations in altitude and aircraft speed. Adaptive control using self-tuning and uses identification techniques to achieve control determination of the parameters of the process being controlled; changes in the process parameters are then used to adjust the actual controller.



The model reference technique is illustrated in fig below; it is relies on the ability to construct an accurate model of the process and to measure the disturbances, which affect the process.



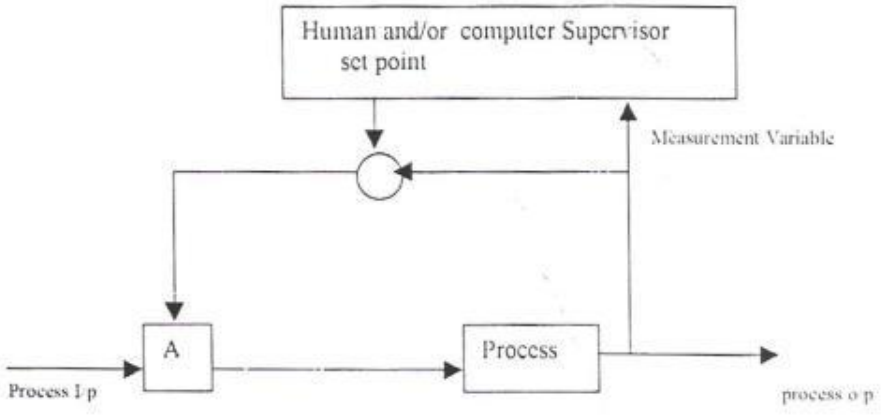
Supervisory control

The adoption of computers for process control has increased the range of activities that can be performed, for not only can the computer system directly control the operation of the plant, it can also provide managers and engineers with a comprehensive picture of the status of the plant operation.

It is in their supervisory role and in the presentation of information to the plant operation-large rooms full of dials and switches have been replaced by VDUs and keyboards-that the major changes have been made.

The main reasons for this were that computers in the early days were not always very reliable and caution dictated that the plant should still be able to run in the event of a computer failure, and that computers were very expensive and it was not economically able to use a computer to replace the analog control equipment in current use.

A computer system used to adjust the set points of the existing analog control system in an optimum manner (to minimize energy or to maximize production) could perhaps be economically justified.



Human or Man-Machine Interface (MMI)

The key to the successful adoption of a computer control scheme can frequently be the facilities provided for the plant operator. It is important that he is provided with a simple and clear system for the day-to-day operation of the plant.

All the information relevant to the current state of its operation should be readily available and facilities to enable interaction with the plant-to change set points, to manually adjust actuators, to acknowledge alarm condition, etc-should be provided.

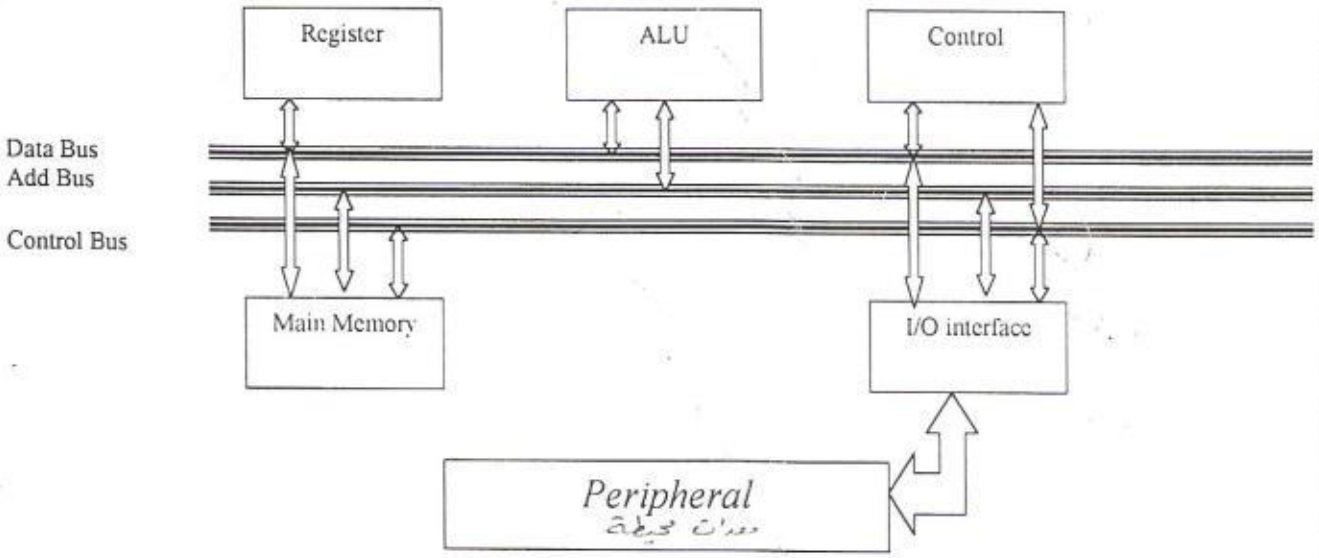
The exact nature of the displays is usually determined by the engineer responsible for the plant or part of the plant.

Addition displays, including trends and summaries of past operations, are frequently available to the engineer, often in the form of hard copy, in addition the plant engineer (maintenance engineer) will require information on which to base decisions about maintenance schedules and instrument actuator and plant component replacements.

Computer hardware requirement for R.T application

Although almost any digital computer can be used for real time computer control and other real-time operations, they are not all equally easily a adapted for such work. A process control computer has to communicate both with plant and personal: this communication must be efficient and effective and processor must be capable of rapid execution to provide for real time action.

A characteristic of computer used in control systems is that they are modular; they provide the means of adding extra units, in particular, specialized input and output devices, to a basic unit. The capability of the basic unit (in term of its processing power , storage capacity , I/O bandwidth and interrupt structure) , determine the overall performance of the system . the arithmetic and logic (ALU) ,control , register , memory and I/O units represents a general purpose digital computer. A simplified block diagram of the basic unit is shown in figure bellow .



General purpose computer

1- Central processing units (CPU)

The ALU together with the control unit and the general purpose register make up the central processing unit. The ALU contains the circuits necessary to carry out arithmetic and logic operations.(adding, subtracting, comparing, multiplication and division, etc).

The control unit continually supervises the operations with in the CPU fetches program instructions from main memory, decodes the instructions and up the necessary data paths and timing cycles for execution of the instruction. The important features of the CPU which determine the processing power available and hence influence the choice of computer for process control include:-

1. word length
2. instruction set
3. addressing methods
4. number of registers
5. information transfer rates
6. interrupt structures

The word length used by the computer is important both in ensuring adequate precision in calculations and in allowing direct access to a large area of main storage with in one instruction word.

Word length	integer range	memory size
8	-128 + 127	256
16	-32768 + 32767	64 K
32	-4294967296 + 4294967295	8 M

formula is integer range (-2^n) → (2^n-1) n:-no. of bits in a word

The basic instruction set of the CPU is also important in determining its overall performance, features which are desirable are:-

1. flexible addressing modes for direct and immediate addressing
2. relative addressing modes
3. Address modification by using of index register
4. instructions to transfer variable length blocks of data between storage unit or locations with in memory.
5. single commands to carry out multiple operations.

These features reduce the number of instructions required to perform 'housekeeping' operations and hence both reduce storage requirements and improve overall speed of operation by reducing the number of access to main memory required to carry out the operations.

Another area which must be considered carefully when selecting a computer for process control is information transfer, both within the CPU, between backing store and the CPU, and with the I/O devices.

2- Storage

The storage used on computer control systems divided in to two categories fast access storage and auxiliary storage.

The fast access memory is that part of the system which contains data, programs and results which are currently being operated on.

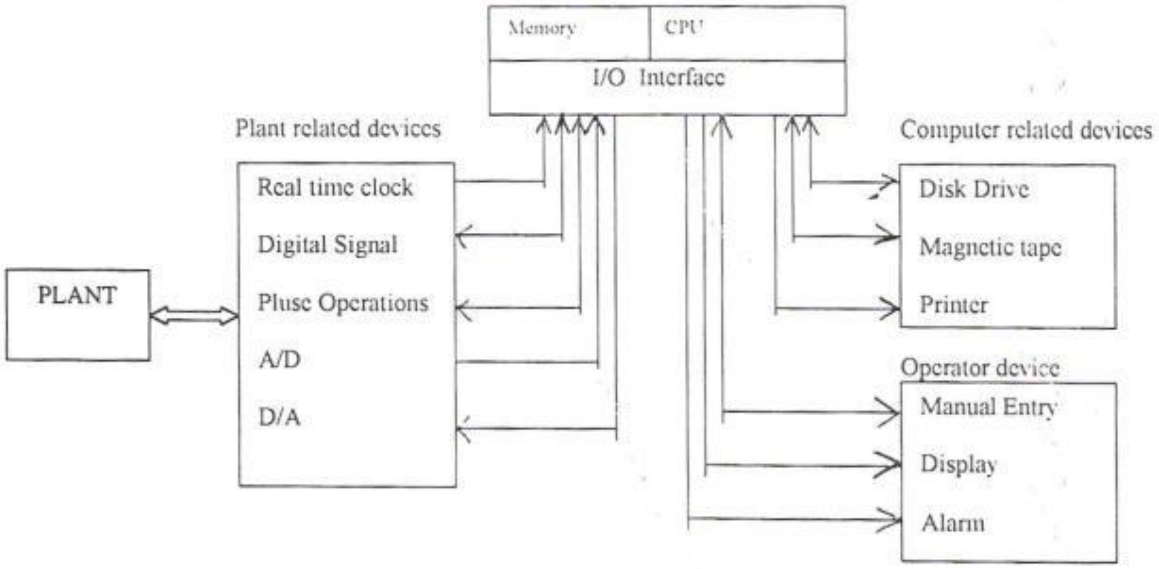
The auxiliary storage medium is typically disk or magnetic tape. These devices provide bulk storage for programs or data which are required infrequently at a much lower cost than fast access memory. The pen ally is a much larger access time and the need for interface boards and software to connect them to the CPU.

3- Input and Output

The I/O interface is one of the most complex areas of a computer system part of the complication arises because of the wide variety of devises which have to be connected and the wide variation in the rates of data transfer.

A printer may operate at 300-band rate where as a disk may require a rate of 500 k bands. The devises may require parallel or serial data transfer may require A/D or D/A conversion or conversion to pulse rate.

The I/O system of most control computers can be divided in to three section: process I/O, operator I/O and computer I/O it is modern practice that all these device (a typical range is shown in figure) share the same bus system and hence the CPU treats all devises in the same way and all the devises have to conform to the bus standard.



Process –~~Rated~~ Related Interface

Instruments and actuators connected to the process or plant can track a wide variety of form: they may use for measuring temperatures and hence could use thermo couples, resistance thermometer, etc, they could be measuring flow rates and use impulse turbines, they could be used to open valves or to control thrusters –operated heaters. In all these operation there is a requirement to convert a digital quantity, In the form of a bit pattern in a computer word, to a physical quantity or to convert a physical quantity to a bit pattern.

Digital Signal Interface

The digital quantities can be either binary i.e, a valve operator close,a switch is on or of,a relay should be open or closed etc.

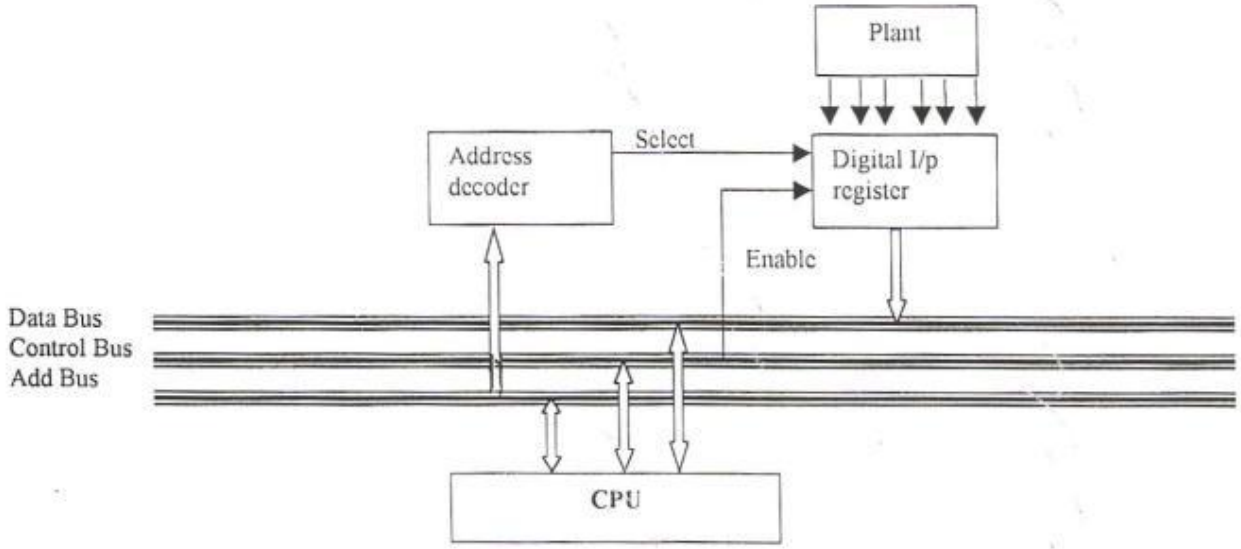
1- Read data

A simple digital input interface is shown below it is a assumed that the plant outputs are logic signal which appear on lines connected to the digital input register

In order to read the line connected to the digital input register the computer has to place on the address bus .the address bus of the register while come decoding circuitry is required in the interface (address decoder) to select the digital input register

In addition to the 'select' signal an 'enable' signal may be also required there
Could be provided by the 'read' signal from the computer control bus

In response to both the 'select' and 'enable' signals the digital input register would enable its output gates to put data on to the computer data bus.



Simple digital I/p interface

The timing of the transfer of information will be governed by the CPU timing .it is assumed for this system that the transfer requires three cycles of the system clock. Labeled T1, T2 &T3.

The address lines begin to change of the beginning of the cycle T1and they are guaranteed to be valid by the start of cycle T2;also at the start of cycle T2 the read line become active .for the correct read operation the digital input register has to provide stable data at the negative going edge (or earlier) of the clock during the T3 cycle and data must remain of the data lines until the negative going edge of the following clock cycle.

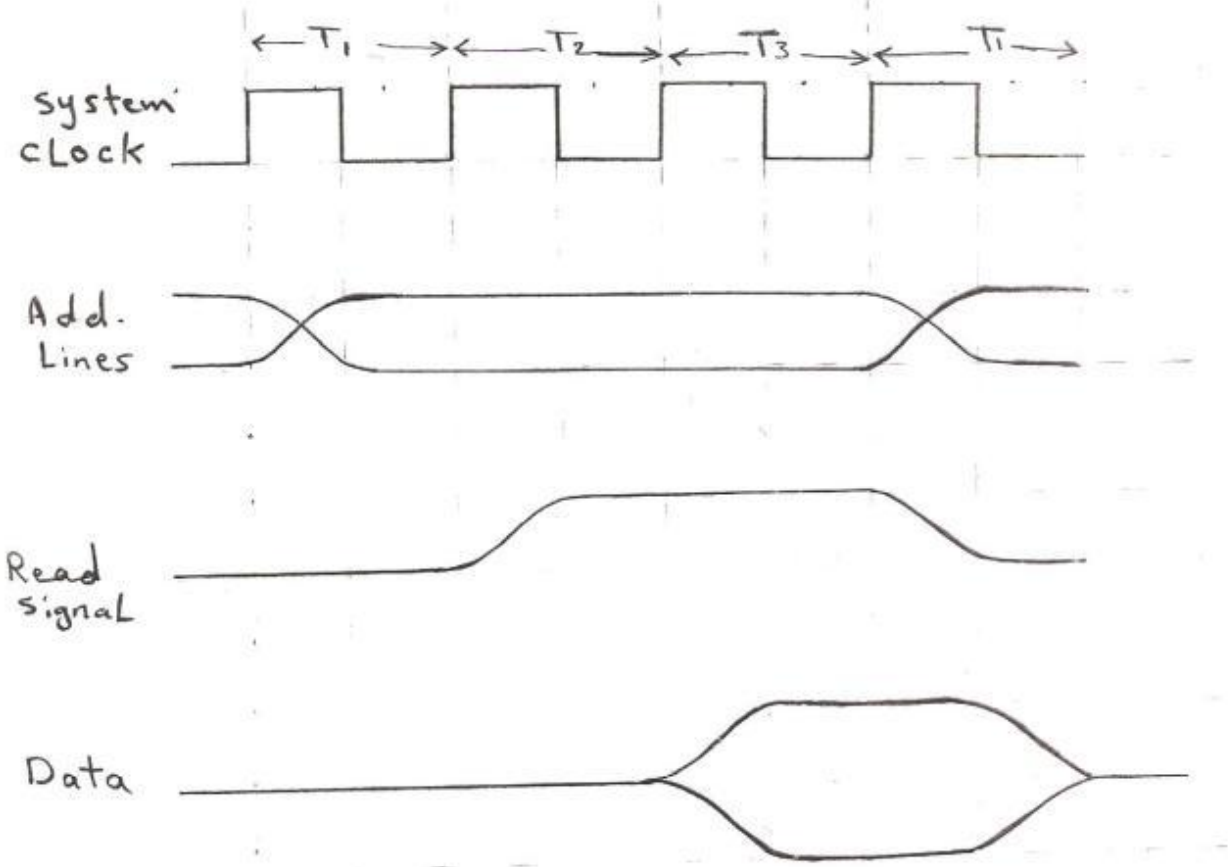
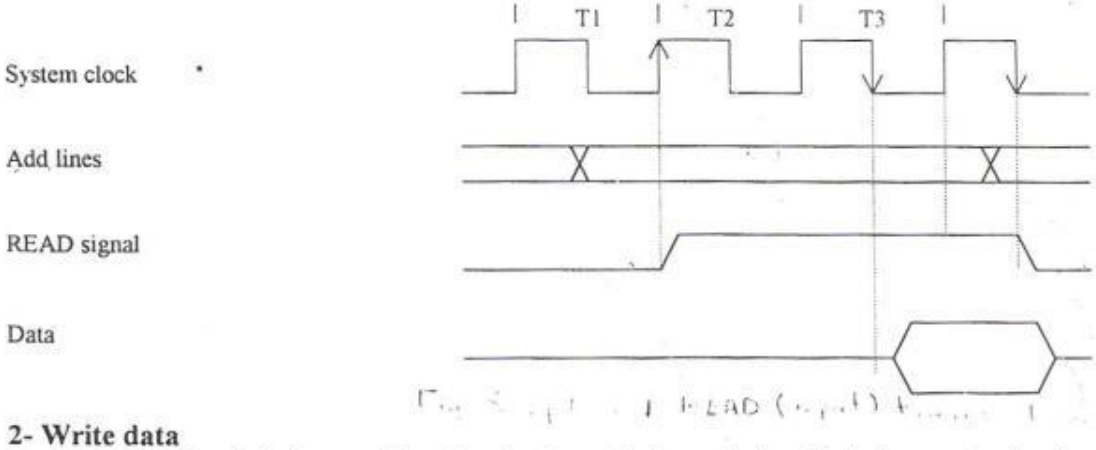


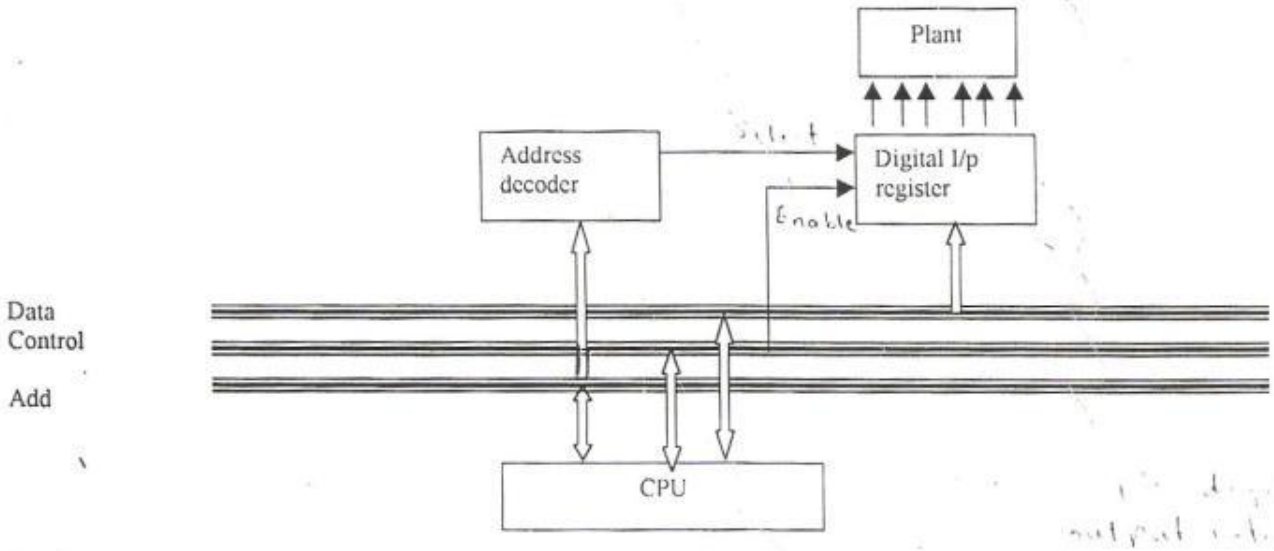
Fig. Simplified READ (input) timing diagram.



2- Write data

A simple digital output interface is shown in figure below. Digital output is the simplest form of the output; all that required is a register or latch which can hold the data output from the computer. The 'enable' signal is used to indicate to the device that the data is stable on the data bus and can be read. The latch must be capable of accepting the data in Avery short length of time (<1 micro second).

The output from the latch is a set of logic levels (0-5) volt. If these levels are not adequate to operate the actuators on the plant, some signal conversion is necessary using the low level signals to operate relays, which carry to higher voltage signals.



② Pulses Interfaces

In its simplest form a pulse input interface consists of a counter connected to a line from the plant. The counter is reset under program control and after a fixed length of time the contents are read by the computer. A typical arrangement is shown in figure, which also shows a simple pulse output interface. The transfer of data from the counter to the computer uses techniques similar to those for the digital input.

If the timing is done by the computer then the 'enable' signal must inhibit further counting of

pulses.

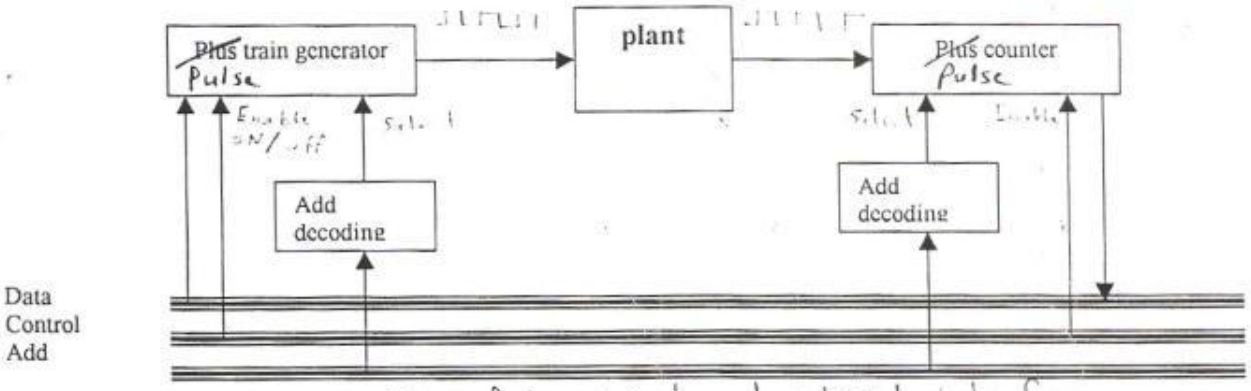


Fig Pulse input and output interface

Pulse generators can be of two types. They can either send a series of pulses of fixed duration or single pulse of variable length. For the former, the computer can be used either to turn a pulse generator ON or OFF or to lead a register with number of pluses to be transmitted.

③ Analog Interface

The analog signal come from example thermocouples, strain gauges, tachogenerator or any analog sensors.

The normal method of operation of an analog input interface is that the computer issues '*start*' or '*sample*' signal typically a short pulse (1 nsec.) . In response to which the A\D switches the 'sample-hold' into SAMPLE for a short period after which the quantization process commence : quantization may take from few (μ sec.) to several (m sec.) on completion of the conversion the A\D raise a 'ready' or 'complete' line which is either polled by computer or is used to generate an interrupt .

We can used multiplexer to switch the inputs from several input lines to a single A\D . the sequence of event is then:- select the channel ,send the start conversion command and wait for the conversion complete signal .

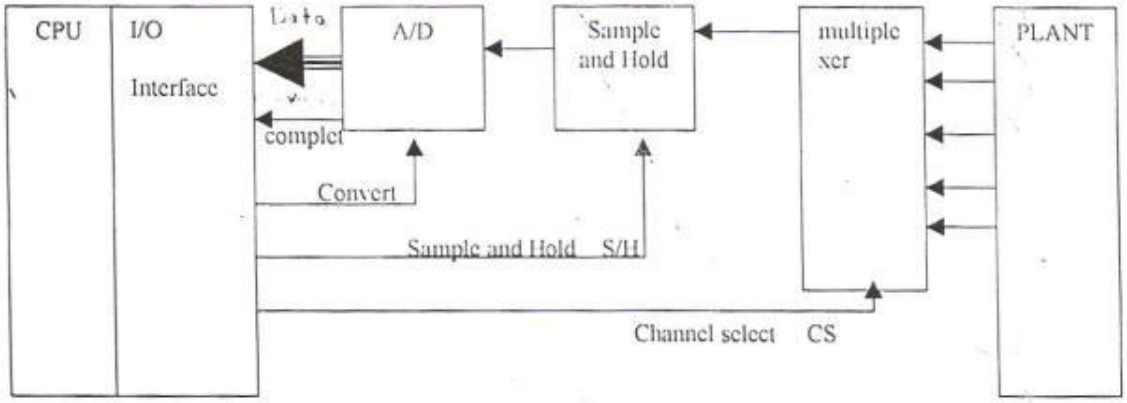


Fig. Analog input system

The action of D/A is simpler (and hence cheaper) than A/D and as a consequence it is normal to provide on converter for each output

Atypical arrangement is shown in figure below.

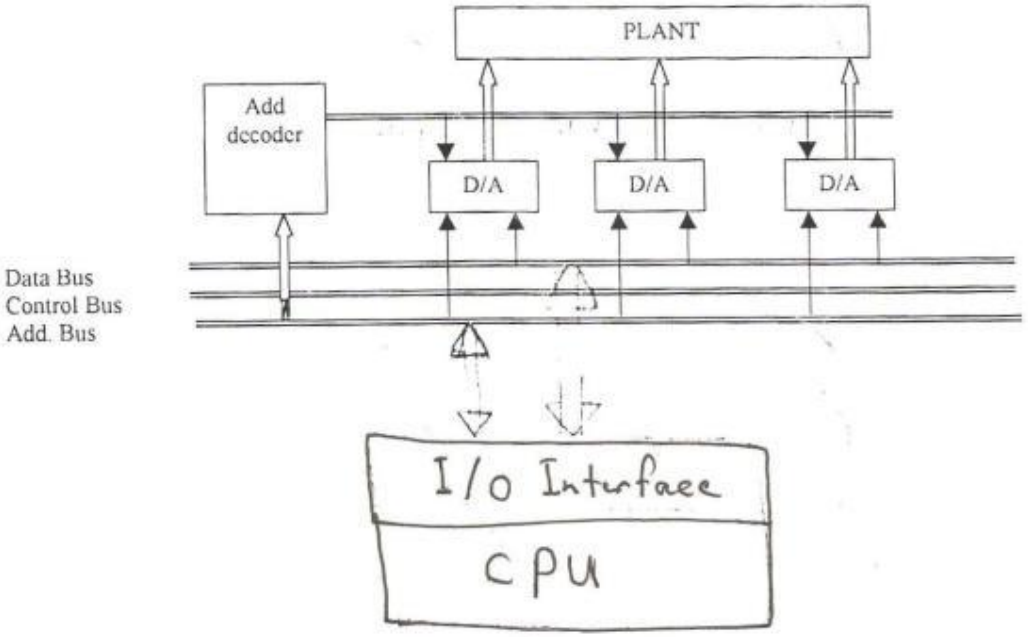


Fig. Analog output system

Data Transfer Technique :

1- Polling

Although the meaning of the data transmitted by the various process , operator and computer peripherals differs , there are many common features which relate to transfer of data from the interface to the computer .

A characteristic of most interface devices is that they operate synchronously with respect to the computer and that they operate at much lower speeds. This difference in speed would severally limit the speed of operation of the computer if it directly controlled the device; however, for maximum flexibility of operation program control is desirable . Operation in this way is known as ' programmed transfer ' and involves the use of the CPU. The alternative is direct memory access (DMA).

A major problem in data transfer is timing. It may be thought that under programmed transfer, the computer can read or write at any time to a device, i.e. can make an *unconditional transfer*. For some process output devices,(e.g. switches & indication lights) these would be connected to a digital output interface or for D\A convector , unconditional transfer is possible : they are always ready to receive data .

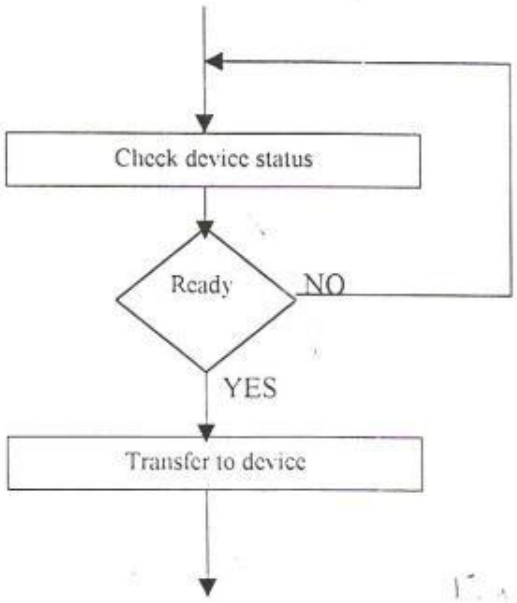
For other output devices; (e.g. printer &communications channels). Which are not fast enough to keep up with the computer but must accept a sequence of data items without missing any item, unconditional transfer can not be used.

The computer must always be sure that the device is ready to accept the next item of data, hence either a timing loop to synchronize the computer to the external device or ' *conditional transfer* ' has to be used .

1) Conditional Wait

A simple example of conditional transfer is shown bellow. Assuming that the data is being transferred to printer which operates at 40 characters\second the computer will find that the device is ready ones every 25 ms . The three instruction involved in performing the test will take approximately 5 μ sec. (the actual time will depend on the speed of the processor) ; thus the condition test will be carried out about 5000 times for each character transmitted . The computer will spend 99.98% of its time in checking to see if the devices is ready and only 0.02% of that time doing useful work.

```
TEST : IN   A , (STATUS)
       BIT  0 , A
       JR   Z , TEST      repeat until A is zero
       OUT  DATA , A
```



2) Conditional delay loop:

As an alternative to providing , on the interface , a status line which can be tested by the computer , a timing loop generated in the computer by loading a register and then decrementing it a specified number of times can be used , e.g.

```
        LD      B,25        ; Load register B with time delay
Loop :  DEC     B           ; decrement B
        JR      NZ,Loop     ; repeat until B is Zero
```

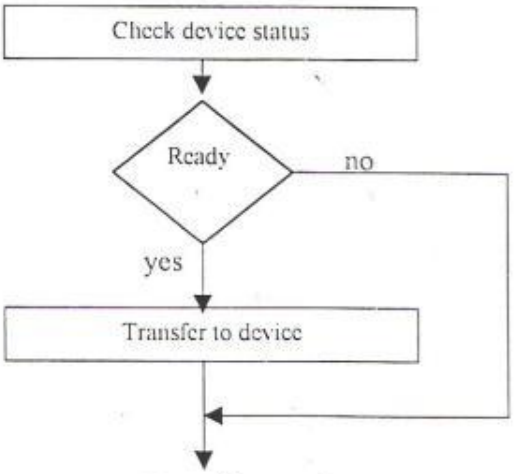
To ensure that no transfer is made before the peripheral is ready the time delay must be slightly greater than maximum delay expected in the peripheral ; thus in term of use the CPU this method is even more inefficient than the use of the conditional wait . It does slightly simplify and reduce the cost of the interface .

3) Continues method (Periodic checks):

An alternative arrangement for conditional transfer , which allows the computer to continue doing useful work if the device is busy .

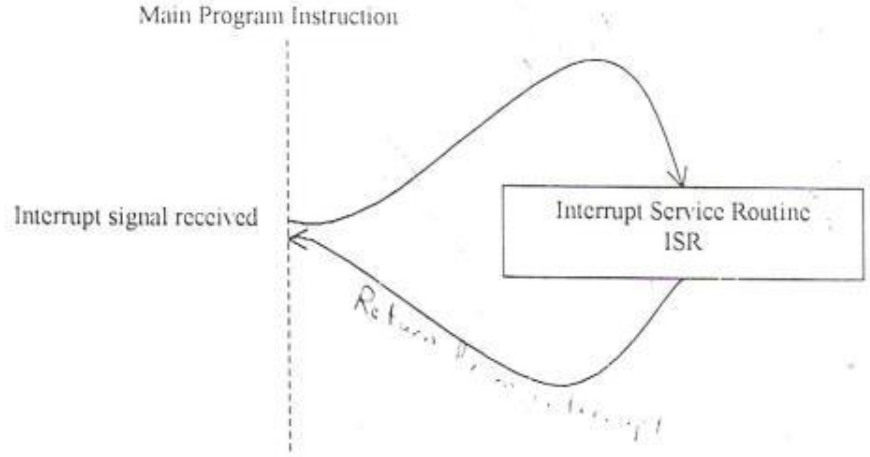
In this method a check is made to see if the device is ready : if it is ready then the transfer is made ; otherwise the computer continues with other work and returns at later time to check if the device is ready .

The technique avoids the inefficiency of waiting in a loop for a device to become ready , but presents the programmer with the difficult task of arranging the software such that all device are checked at frequent intervals .



2- Interrupts :

An interrupt is a mechanism by which the flow of the program can be temporarily stopped to allow a special piece of software (an Interrupt Service Routine, or Interrupt Handler) to run. When this routines has finished, the program was temporarily suspended is resumed. The process is illustrated bellow: -



- operating system : Interrupts are used to force entry to the operating system before the end of a time slice.

Interrupt are essential for the correct operation of most real –time computer systems, in addition to providing a solution to the conditional wait problem they used for: -

- Real-time clock

The external hardware provides a signal at regulated spaced intervals of time, the interrupt service routine (ISR) count the signal and keeps a clock.

- Alarm input

Various sensors can be used to provide a change in a logic level in the event of an alarm.

- Hardware failure indication.
- Debugging aids

Interrupt are frequently used to insert breakpoints or traces in the program during program testing.

- Power failure warning.

It is simple to include in the computer system a circuit that detects very quickly the loss of power in the system and provides a few milliseconds warning before the loss is such that the system stops working.

(1) Saving and restoring registers

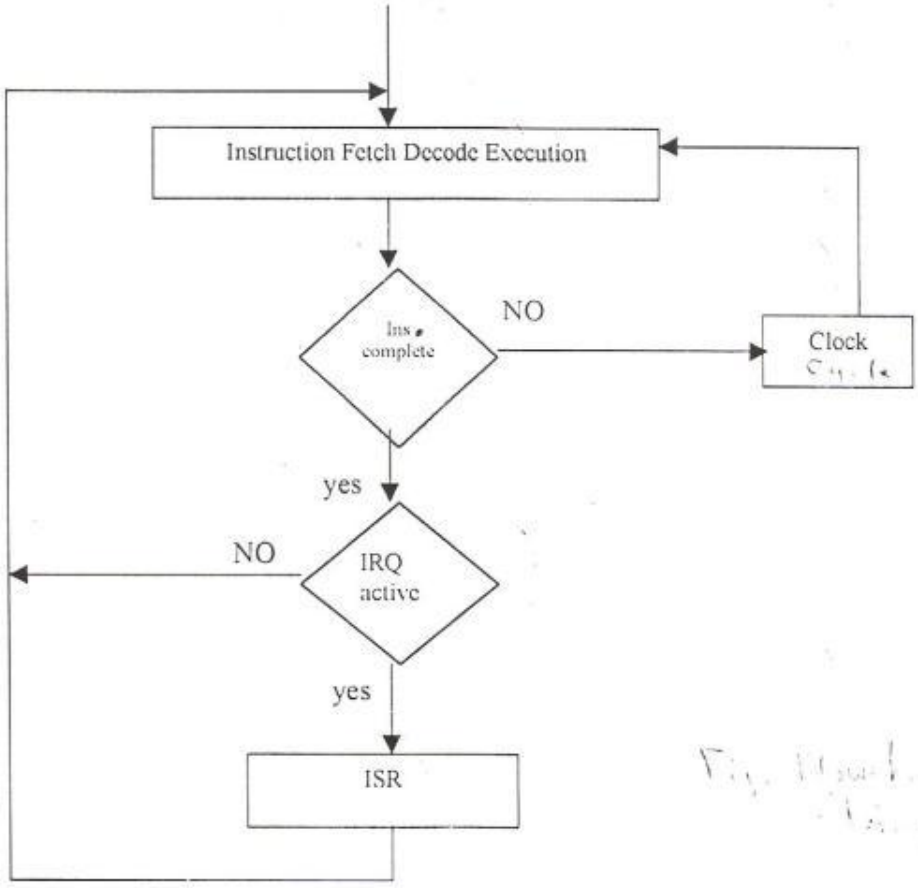
Since an interrupt can occur at any point in a program precautions have to be taken to prevent information, which is being held temporarily in the CPU registers from the being, overwritten.

All CPU s automatically save the contents of the program counter. This is vital:if the contents are not saved then a return to the point in the program at which the interrupt occurred could not be made .

(2) Interrupt input mechanisms

A simple form of interrupt input is shown bellow. In between each instruction the CPU checks the IRQ line. if it is active, an interrupt is present and ISR is entered;if it is not active the next instruction is fetched and cycle repeats.

Note that an instruction involves more then one CPU clock cycle and that the interrupt line is checked only between instructions. Because several clock cycles may elapse between successive checks of the interrupt line, the interrupt signal must be latched and only cleared when the interrupt is acknowledged.



The use of polling with either busy-wait or periodic checks on device status provides the simplest method in term of the programming requirements and in the testing of programs. Interrupt driven systems are much more difficult to debug since many of error may be time dependent. At high data transfer rates the use of interrupts is inefficient because of the overheads involved in the interrupt service routine (saving and restoring the environment) hence polling is often used

Direct memory access (DMA)

There are three modes for used DMA

1- Burst mode

In burst mode the DMA controller takes over the data high ways of the computer and locks out the CPU for the period of time necessary to transfer. The use of burst mode can seriously affect the response time of a real time system to an external Event and because of their mode not be acceptable.

2- Distributed mode

In this mode the DMA controller takes occasional machine cycle from the CPU 's control uses each cycle to transfer a byte of information to or from fast memory to the backing memory. In real-time system, if software-timing loops are used then the loss of machine cycle will affect the time taken to complete the loop.

3- cycle-stealing method

In this mode transfer data only during cycles when the CPU is not using the data bus. Therefore The program proceeds at the normal rate completely unaffected by DMA data transfer. This is, however, the slowest method of transfer to backing store.

Communications

The use of distributed computer systems implies the need for communication. As the distance between the source and receiver increase it becomes more difficult, when using analog techniques, to obtain a high signal to noise ratio; this is particularly so in an industrial environment where there are therefor generally limited to short distance.

The use of parallel digital transmission provides high data transfer rates but is expensive in term of cabling and interface circuitry and again is normally only used over short distance (or when very high rates of transfer are required)

Serial communication techniques can be characterized in several way

1- mode
   a) Asynchronous
   b) Synchronous

2-quantity
   a) Character – by –character
   b) Block

3- distance
   a) Local
   b) Wide area i.e remote

4- code
   a) ASCII
   b) Other

Asynchronous and synchronous transmission techniques

Asynchronous transmission implies that the transmitter and receiver circuits use their own local clock signals to gate data on and off the data transmission line.

In order that the data can be interpreted unambiguously there must be some agreement between the transmitter and receiver clock signal, this agreement is forced by the transmitter periodically sending synchronization information down the transmission Line.

The most common form of synchronous transmission is the character–by–character System which is frequently used for connecting terminals to computer equipment and was introduced for transmission of information over telegraph lines.

It is some times called the stop – start system. In this system each character which is transmitted is preceded by 'start' bit and followed by one or two 'stop' bits (see the figure below)

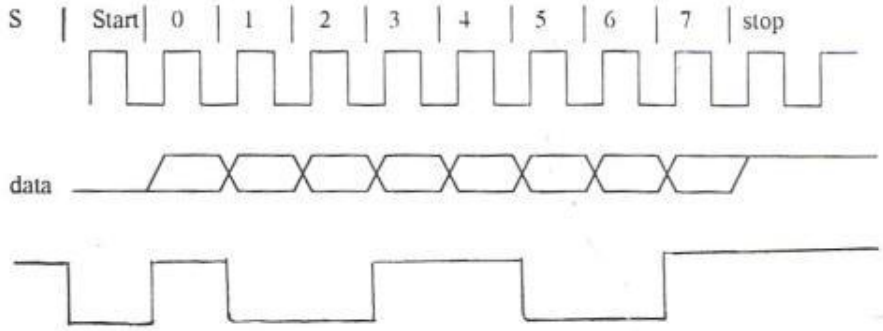


Fig. Asynchronous transmission

The start bit is used by the receiver to synchronize its clock with the incoming data, the signal must remain synchronized for the time taken to receive the following eight data bits and two stop bits.

The advantage of the stop-start system is that, particularly at the lower transmission rates, the frequencies of the clock signal generators do not have to be closely matched.

The disadvantage of the system is that for each character transmitted (8bits) three or four extra bits of information have also to be transmitted, i.e. the overall information ratio is not very high. The range of transmission speeds used for this system from (75)bits/sec to(9600)bit/sec. the standard speed are (75,110,300,600,1200,1800,2400,4800,9600)b/s .

To overcome the problem of transmitting redundant bits, synchronous systems designed to transmit large volumes of data over short periods of time, such as computer-to-computer systems, use block synchronous transmission techniques. Here, the characters are grouped into records, e.g., blocks of 80 characters, and each record is preceded by a synchronization signal and terminated with a stop sequence.

The synchronization sequences is used to enable the receiver to synchronize with the transmitter clock. In order to establish effective communication it is necessary to transmit more than just a synchronization signal–the additional information is called the protocol.

A simple protocol is:-

1-At the start of a transmission, bit synchronization is achieved by the transmitter sending out a sequence of 0s and 1s.

2-Followed by the ASCII code 'SYN'. The transmitter will continue to send the 'SYN' code until receiver responds by sending back the code 'ACK' or a preset time elapses (device time out).

3-If time out occurs; the transmitter sends the bit pattern of 1s again.

4-Once contact has been established the transmitter will sends one out 'SYN' characters during any idle period and the receiver will respond by sending back 'ACK'.

5-The line will only be completely idle when the transmitter has sent 'EOT' (end of transmission) character.

The text is broken up in to blocks and each block is preceded by an 'STX' (start of text) character and ended by an 'ETX' (end of text).

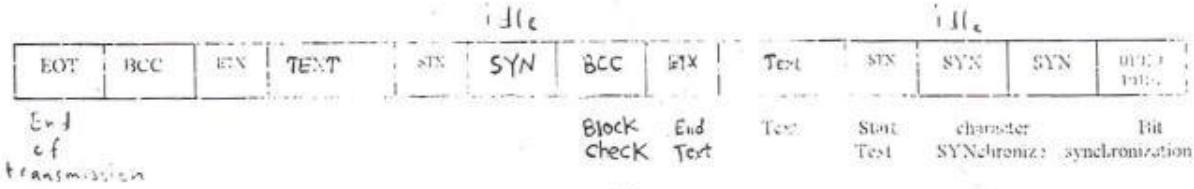


Fig. Synchronous transmission

Following the ETX will be an integrity check on the data, typically this will take the form of a parity check.

Lecture 5

Real-Time Languages

The requirements of real time software place heavy demands on programming languages. It should by now be obvious that it is essential that real-time software is reliable: the failure of a real-time system can be expensive both in terms of lost production, or in some cases, in the loss of human life (an aircraft control system).

User Requirements

The user requirement is divided into six general areas.

① security ② readability ③ Flexibility
④ simplicity ⑤ portability ⑥ efficiency

① Security

security can be considered to be a measure of ~~extent~~ extent to which a language to detect errors automatically either at compile time or through the run time.

Economically it is important to detect errors at the compilations stages than at run-time since the earlier the error is detected the less it costs to correct.

In real-time system development the compilation is often perform on a computer than the one used in actual system, where as run-time testing has done on the actual hardware and, in the later stages, on the hardware connected plant.

Connect plant.

Readability :-

The readability of a program is a measure of the ~~code~~ ease with which its operation can be understood ~~what~~ without resort to supplementary documentation such as flowcharts or natural language descriptions. The emphasis is on ease of reading because a particular segment of code will only be written once but will be read many time. /

The benefits of good readability are

1) reduction in documentation costs.

2) easy error detection.

3) easy maintenance.

③ Flexibility

For a language to be described as a general purpose language there is a requirement that the programmer should be able to express all the operations required in a program without the need to use assembly coding. The flexibility of language is a measure of this facility.

It is particularly important in real-time systems, in that frequently non-standard I/O devices will have to be controlled. The achievement of high flexibility can conflict with achieving high security.

④ Simplicity

In language design as in other areas of design, the simple is to be preferred to the complex. Simplicity contributes to security. It reduces the cost of training. it reduces the probability of programming errors arising from misinterpretation of the language features. it reduces compiler size and leads to more efficient object code.

تابلية النقل

5) Portability

The achievement of portability, while very desirable as a means of speeding up developments, reducing costs and increasing securityes difficult to achieve in practice. Surface portability has improved with the standardization agreements on many languages. i.e. It is now often possible to transfer a program from one Computer to another and find that it will compile and run on the Computer to which it has been ~~trans~~ transferred.

However
There are, still problems when the word-lengths of the two machines differ; there may also be problems with precision with which numbers are represented even on computers with the same word-length.

② Efficiency

In the early Computer Control systems great emphasis was placed on efficiency of the coding - both in terms of the size of the object code and the speed of operation - as Computers were both expensive and very slow (by today standards). As a consequence programming was carried out using assembly languages and frequently 'tricks' were used to keep the code small and fast. The desire for the generation of efficient object code was carried over into the designs of the early real-time languages and in these languages the emphasis was on efficiency rather than security and readability.

Language Requirements & features

The major features which must be considered are listed below

1) Declarations
2) Constants
3) Initialization
4) Types
5) Control structures
6) Scope and visibility
7) Modularity
8) Exception handling
9) Independent / seperate Compilation
10) Multi-tasking
11) Low-Level constructs

1) Declarations :-

The purpose of declaring an object used in a program is to provide the compiler with information on the storage requirements and to inform the system explicitly of the names being used.

Languages such as pascal require all objects to be specifically declared used for a type, to be associated with the object at declaration.

The provision of type information allows the compiler to check that the objects is used only in operations associated with that type. If for example an object is declared as being of type real and then is used as an operand in a logic operations the compiler should detect the type incompatibility and flag the statement as being incorrect.

② Type

As we have seen above, the allocation of type is closely associated with the declaration of objects. The allocation of a type defines the set of values that can be taken by an object of that type and the set of operations that can be performed on the objects.

The richness of types supported by a language and the degree of rigor with which type compatibility is enforced by the languages are important influences on the security of programs written in the language.

Languages which rigorously enforce type compatibility are said to be strongly typed. Languages which do not enforce type compatibility are said to be weakly typed.

③ :- Initialization :-

It is useful if, at the time of declaration of a variable, it can be given an initial value. This is not, of course, strictly necessary as a value can always be assigned to a variable.

In terms of the security of a language it is important that the compiler checks that a variable is not used before it has had a value assigned to it. It is bad practice to rely on the compiler to initialize variables to some zero or 'null' value.

The security of languages such as Pascal is enhanced by the compiler checking that all variables have been given an initial value.

4) Constants

Some of the objects referenced in a program will have constant values, either because they are physical or mathematical entities such as the speed of light or pi (π) or because they are a parameter which is fixed for that particular implementation of the program.

It is always possible to provide constants by means of initializing a variable to the appropriate quantity.

5) Control structures

There has been extensive argument over the past few year about the use of both Conditional and unConditional GOTO statements in high-level language.

It is argued that the use of GoTo's makes a program difficult to read and it has been shown that any program can be expressed without the use of GOTO, as long as the language supports the WHILE statement 'IF ... THEN ... ELSE' Conditional

⑥ Scope and Visibility

The scope of a variable is defined as the region of a program in which the variable potentially accessible or modifiable. The regions in which it may actually be accessed or modified are the regions in which it is said to be visible.

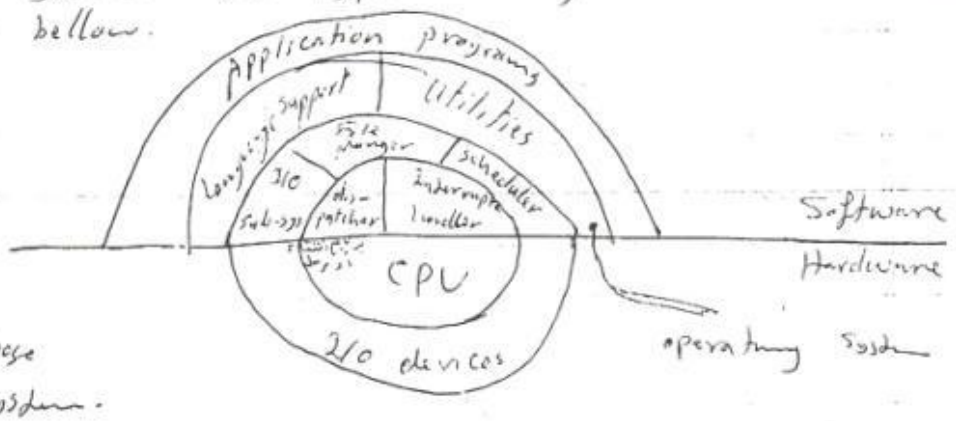
The scope of a variable declared in the main program extends over the whole program, but such a variable, unless named in a COMMON statement, will visible in any sub-programs. Scope and visibility are closely related to where in a program a variable is declared.

Choice of Programming Language

1. editor
2. library manager
3. linker / loader
4. debugger
5. version control
6. database manager
7. pretty printer
8. cross-reference generator.

Operating System :-

The traditional approach to providing a system to meet a wide range of requirements has been to incorporate all the requirements inside a general purpose operating System. The typical arrangement is illustrated in figure bellow.



General purpose operating system.

Access to the hardware of the system and to the I/O devices is through the operating system. In any real-time and multi-programming systems restriction of access is enforced by hardware and software traps.

In the single-job operating systems access through operating system is not usually enforced.

In addition to supporting and controlling the basic activities, operating system particularly provide various utility programs. e.g. Loaders, linkers, assemblers and debuggers, as well as run-time support for high-level languages.

1. Single-user, single-job Operating System

As an example of single-user, single-task, disk-based operating system, the CP/M 80 system of Digital Research will be described. This system is available for 8080, Z80 based Computer systems.

CP/M Consists of three major sections

① Console Command Processor (CCP)
② Basic Input/Output System (BIOS)
③ Basic Disk Operating System (BDOS)

The relationship between the various sections of the operating system, the Computer hardware and the user is illustrated in figure bellow.

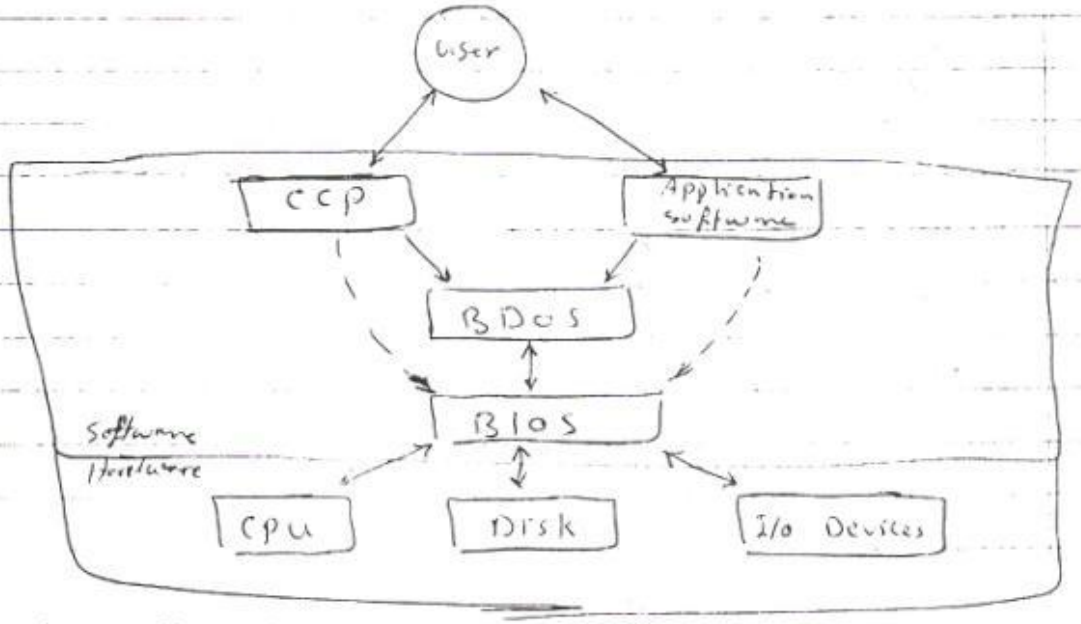


Fig. General structure of CP/M

The Console Command Processor provides a means by which the user communicate with the operating system from the Computer Console device. It is use to issue Commands to the operating System and to provide the user with information about the actions being performed by the operating system.

The actual processing is the Commands issued by the user is done by the BDos which also handles the input and output and the file operations on the disks.

The BDos makes the act management of the file and input/output operations transparent to the use. Application programs will normally Communicate with the hardware of the system through 'system Calls' which are processed by the BDOS.

The BIOS Contains the various device drivers which manipulate the physical devices, this section of the operating system may vary from implementation to implementation as it has to operate directly with the underlying hardware of the Computer. For example, depending on the manufacturer the physical addresses of the peripherals may vary, as may the type of peripheral, and the type of Control used for the disk drive. All these differences will be accommodated in the Cordinet the BIOS.

On starting the Computer the three sub-systems which reside on the disk are loaded into memory.

The Computer must, therefore, have some means of booting the operating system from a system's disk. The normal arrangement is to provide a small bootstrap loader in a ROM chip

The Combination of BDOS and BIOS frequently referred to as the FDOS (Functional Disk Operating System) and these units have to remain in memory.

CCP direct Commands

There are many direct Commands available to the user; the routines to support these Commands are held in the CCP area of memory.

1- DIR : Display a list of all filenames

2- ERA : Erased a list of files

3- REN <new>=<old> : rename the file <old> as <new>.

4- Type < > : listed the contents of file

Basic Disk Operating System BDOS

The CCP facilities are useful in developing user programs, but one of the purposes of the operating system is to support user programs while they are running.

The facilities provided by the BDOS are used for this purpose and these are essentially concerned with operating the input and output devices attached to the system.

Input / Output Devices

Two types of I/O devices are supported

1 - Sequential. Data is transferred one byte at a time

2 - Blocked. Data is transferred in fixed length blocks or records of 128 bytes these are assumed to be disk drives in CP/M.

In dealing with I/O devices CP/M considers devices to be 'logical' 'physical'. Logical devices are software constructs used to simplify the user inter user programs perform input and output to logical devices, the BDOS connects the logical device to the physical device. The actual operation of the physical device performed by software in the BIOS.

2) Real-time Multi-Tasking Operating System

The natural way to structure a typical Computer Control system is in the form of a number of different tasks which all apparently run in parallel.

The implementation of a design based on this approach is made easier if an operating system which supports multi-tasking can be used.

The traditional real-time operating systems are based on the assumption that all the tasks in the system will be executed on a single CPU or processor. In this section the same assumption will be made.

Confusion can arise between multi-user or multi programming operating systems and multi-tasking operating systems.

The function of a multi-user operating system is illustrated in figure

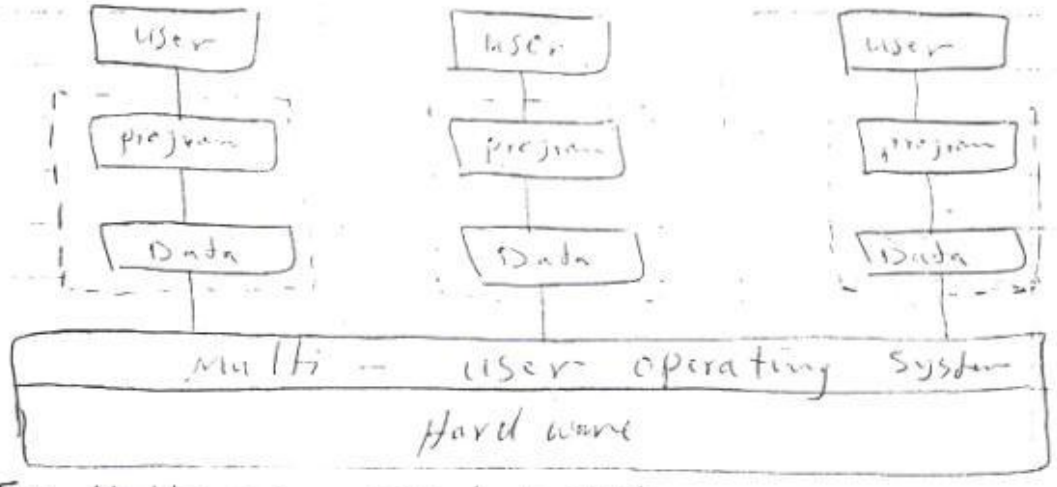


Fig. Multi-user operating system

The operating system ensures that each user can run a single program as if they had the whole of the Computer system for their program.

Although at any given instance it is not possible to predict which user will have the use of the CPU or even if the user's code is in the memory.

The operating system ensures that one user program cannot interfere with the operation of another user program.

Each user program runs in its own protected enviroment: a primary concern of the operating system is to prevent one program corrupting another, either deliberately or through error. In a multi-tasking operating system it is assumed that there is a single user and that the various tasks are to co-operate to serve the requirements of the users.

Co-operation will require that the tasks communicate with each other and share common data. this is illustrated in figure.

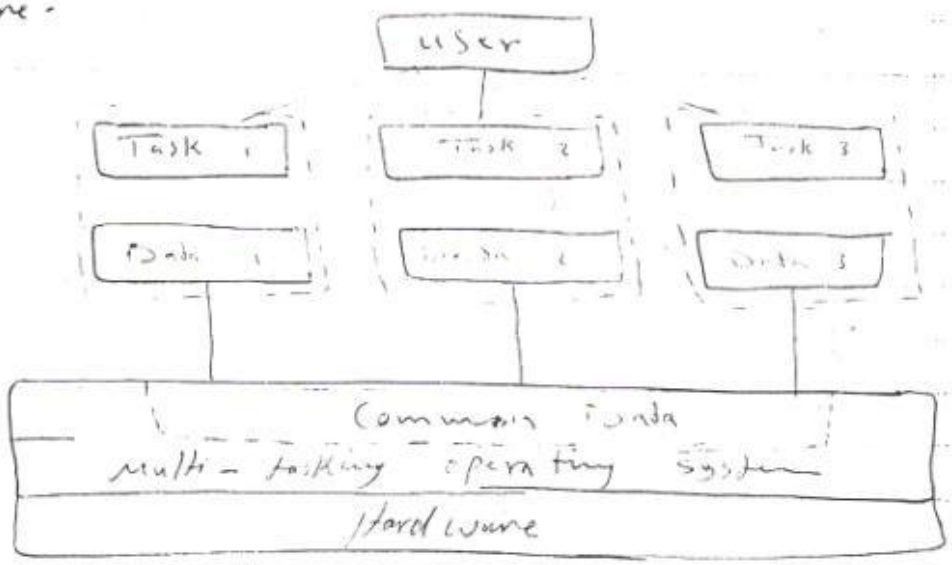


Fig. Multi-tasking operating system

In a good multi-tasking operating system the way in which tasks communicate and share data will be regulated such that the operating system is able to prevent inadvertent communication or data access (arising through an error in the coding of one task) and hence protect data which is private to a task.

A fundamental requirement of an operating system is to allocate the resources of the computer to the various activities which have to be performed: in a real-time operating system this allocation procedure is complicated by the fact that some of the activities are time-critical and hence have a higher priority than others. There must therefore be some means of allocating priorities to tasks and of scheduling allocation of CPU time to the tasks according to some priority scheme.

A task may use another task, i.e. it may require certain activities which are contained in another task to be performed and it may itself be used by another task. Thus tasks may need to communicate with each other. The operating system therefore has to have some means of enabling tasks either to share memory for the exchange of data or to provide a mechanism by ~~tasks can~~ which tasks can send messages to each other. In addition, tasks may need to be invoked by external events; hence the operating system must support the use of interrupts.