CHAPTER 7

DIGITAL CONTROL

7.1 INTRODUCTION

In Chapters 4 through 6 we have shown how to design continuous-time controllers for aircraft. However, with microprocessors so fast, light, and economical, control laws are usually implemented on modern aircraft in digital form. In view of the requirement for gain scheduling of aircraft controllers, digital control schemes are especially useful, for gain scheduling is very easy on a digital computer.

To provide reliability in the event of failures, modern aircraft control schemes are redundant, with two or three control laws for each application. The actual control to be applied is selected by "voting"; that is, there should be good agreement between two out of three controllers. Such schemes are more conveniently implemented on a microprocessor, where the comparison and voting logic reside.

In this chapter we address the design of digital, or discrete-time, controllers, since the design of such controllers involves some extra considerations of which one should be aware. In Section 7.2 we discuss the *simulation* of digital controllers on a digital computer. Then in Sections 7.3 and 7.4 two approaches to digital control *design* are examined. Finally, some aspects of the actual *implementation* are mentioned in Section 7.5.

In the first approach to digital control design, covered in Section 7.3, we show how to convert an already designed continuous-time controller to a discrete-time controller using, for instance, the bilinear transform (BLT). An advantage of this *continuous controller redesign* approach is that the sample period T does not have to be selected until after the continuous controller has been designed.

Unfortunately, controller discretization schemes based on transformations such as the BLT are approximations. Consequently, the sampling period T must be small to ensure that the digital controller performs like the continuous version from which it was designed. Therefore, in Section 7.4 we show how the design of the continuous-time controller may be *modified* to take into account some properties of the sampling process as well as computation delays. Discretization of such a modified continuous controller yields a digital control system with improved performance.

In Section 7.5 we discuss some implementation considerations, such as actuator saturation and controller structure.

There are many excellent references on digital control; some of them are listed at the end of the chapter. We will draw most heavily on the work of Franklin and Powell (1980), Åström and Wittenmark (1984), and Lewis (1992).

7.2 SIMULATION OF DIGITAL CONTROLLERS

A digital control scheme is shown in Figure 7.1-1. The plant $G(s)$ is a continuous-time system, and $K(z)$ is the dynamic digital controller, where s and z are, respectively, the Laplace and Z-transform variables (i.e., $1/s$ represents integration and z^{-1} represents a unit time delay). The digital controller $K(z)$ is implemented using software code in a microprocessor.

The hold device in the figure is a digital-to-analog (D/A) converter that converts the discrete control samples u_k computed by the software controller $K(z)$ into the continuous-time control $u(t)$ required by the plant. It is a *data reconstruction* device. The input u_k and output $u(t)$ for a *zero-order hold (ZOH)* are shown in Figure 7.2-2. Note that $u(kT) = u_k$, so that $u(t)$ is continuous from the right. That is, $u(t)$ is updated at times kT. The sampler with sample period T is an analog-to-digital (A/D) converter that takes the samples $y_k = y(kT)$ of the output $y(t)$ that are required by the software controller $K(z)$.

In this chapter we discuss the design of the digital controller $K(z)$. Once the controller has been designed, it is important to *simulate* it before it is implemented to determine if the closed-loop response is suitable. The simulation should provide the response at all times, including times between the samples.

To simulate a digital controller we may use the scheme shown in Figure 7.2-3. There the continuous dynamics $G(s)$ are contained in the subroutine $F(t, x, \dot{x})$; they are integrated using a Runge-Kutta integrator. Note that two time intervals are involved: the sampling period T and the *Runge-Kutta integration period* $T_R \ll T$. T_R should be selected as an integral divisor of T.

Several numerical integration schemes were discussed in Section 3.5. We have found that the Runge-Kutta routines are very suitable, while Adams-Bashforth

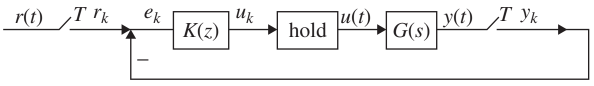


Figure 7.1-1 Digital controller.

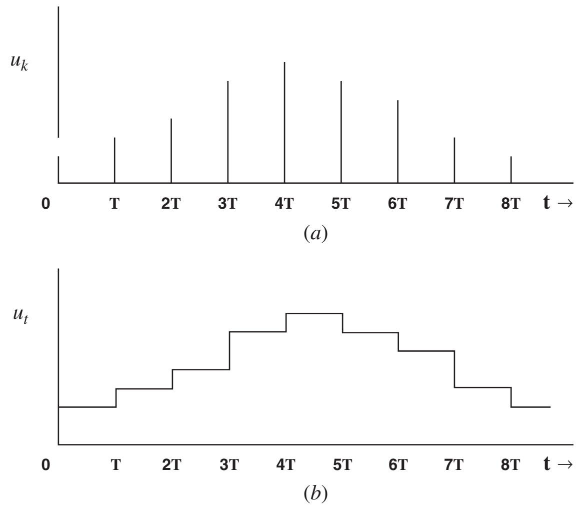
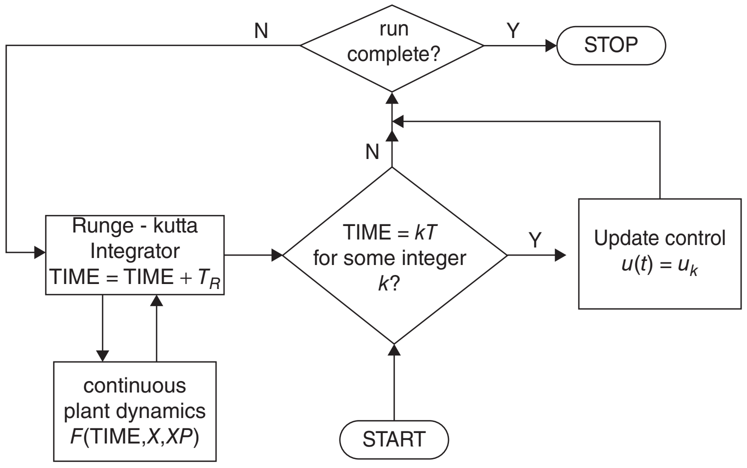


Figure 7.2-2 Data reconstruction using a ZOH: (*a*) discrete control sequence u_k; (*b*) reconstructed continuous signal $u(t)$.

routines do not give enough accuracy for digital control purposes. This is especially true when advanced adaptive and parameter estimation techniques are used. For most purposes, the fixed-step-size Runge-Kutta algorithm in Appendix B is suitable if T_R is selected small enough. In rare instances it may be necessary to use an adaptive step size integrator such as Runge-Kutta-Fehlburg. In all the examples in this book, the fixed-step-size version was used.



N
run complete?
Y
STOP
N
Runge - kutta Integrator TIME = TIME + T_R
TIME = kT for some integer k?
Y
Update control $u(t) = u_k$
continuous plant dynamics F(TIME,X,XP)
START

Figure 7.2-3 Digital control simulation scheme.

A driver program that realizes Figure 7.2-3 is given in Figure 7.2-4. It is written in a modular fashion to apply to a wide variety of situations and calls a Runge-Kutta integration routine such as the one in Appendix B. The Runge-Kutta integrator in turn calls subroutine F(TIME, X, XP) containing the continuous-time dynamics.

The digital controller is contained in subroutine DIG(T, X). Figure 7.2-3 assumes a ZOH; thus, the control input $u(t)$ is updated to u_k at each time kT and then held constant until time $(k + 1)T$. The driver program in Figure 7.2-4 performs this.

It is important to realize that this simulation technique provides $x(t)$ as a continuous function of time, even at values *between* the sampling instants [in fact, it provides $x(t)$ at multiples of T_R]. This is essential in verifying acceptable *intersample behavior* of the closed-loop system prior to implementing the digital controller on the actual

```
C  DRIVER PROGRAM TO COMPUTE AND SIMULATE DIGITAL CONTROL SCHEME
C  REQUIRES SUBROUTINES:
C    DIG(T,X) FOR DIGITAL CONTROL UPDATE AT SAMPLING INSTANTS
C    RUNKUT(TIME,TR,X,NSTATES) TO INTEGRATE CONTINUOUS DYNAMICS
C    F(TIME,X,XP) TO PROVIDE CONTINUOUS PLANT DYNAMICS

         PROGRAM DIGICON
         REAL X(1)
         COMMON/CONTROL/U(1)
         COMMON/OUTPUT/Y(1)

C  SET RUN TIME, SAMPLING PERIOD, RUNGE KUTTA STEP SIZE
         DATA TRUN,T,TR/5.,0.5,0.01/
C  SET INITIAL PLANT STATE
         DATA X(1)/0./

         TIME= 0.
         N= NINT(TRUN/T)
         NT= NINT(T/TR)

*  DIGITAL CONTROL SIMULATION RUN

         DO 10 K= 0,N-1
C    UPDATE DIGITAL CONTROL INPUT

         CALL DIG(T,X)

C    INTEGRATE CONTINUOUS DYNAMICS BETWEEN SAMPLES
         DO 10 I= 1,NT

C    WRITE TO FILE FOR PLOT
         WRITE(7,*) TIME,X(1),U

10        CALL RUNKUT(TIME,TR,X,1)
         WRITE(7,*) TIME,X(1),U
         STOP
         END
```

Figure 7.2-4 Digital control simulation driver program.

plant. Even though the closed-loop behavior is acceptable at the sample points, with improper digital control system design there can be serious problems between the samples. The basic problem is that a badly designed controller can destroy observability, so that poor intersample behavior is not apparent at the sample points (Lewis, 1992). This simulation scheme allows the intersample behavior to be checked prior to actual implementation.

We will soon present several examples that demonstrate the simulation of digital controllers. First, it is necessary to discuss the design of digital controllers.

7.3 DISCRETIZATION OF CONTINUOUS CONTROLLERS

A digital control design approach that could directly use all of the continuous-time techniques of the previous chapters would be extremely appealing. Therefore, in this section we discuss the design of digital controllers by the redesign of existing continuous controllers. In this approach, the continuous controller is first designed using any desired technique. Then the controller is discretized using, for instance, the bilinear transform, to obtain the digital control law, which is finally programmed on the microprocessor.

An alternative approach to digital control design is given by Lewis (1992). In that approach, it is not necessary to design a continuous-time controller first, but a discrete-time controller is designed *directly* using a sampled version of the aircraft dynamics.

We now show how to discretize a continuous controller to obtain a digital controller. The idea is illustrated by designing a digital proportional-integral-derivative (PID) controller in Example 7.3-1 and a digital pitch-rate control system in Example 7.3-2.

Suppose that a continuous-time controller $K^c(s)$ has been designed for the plant $G(s)$ by some means, such as root-locus or LQ design. We will discuss two approximate schemes for converting $K^c(s)$ into a discrete-time controller $K(z)$ that can be implemented on a microprocessor. We discuss first the BLT and then the matched pole-zero (MPZ) technique.

The sample period is T seconds, so that the *sampling frequency* is

$$f_s = \frac{1}{T}, \omega_s = \frac{2\pi}{T} \tag{7.3-1}$$

Bilinear Transformation

A popular way to convert a continuous transfer function to a discrete one is the *bilinear transformation* or *Tustin's approximation*. On sampling (Franklin and Powell, 1980) the continuous poles are mapped to discrete poles according to $z = e^{sT}$. As may be seen by series expansion

$$z = e^{sT} \approx \frac{1 + sT/2}{1 - sT/2} \tag{7.3-2}$$

Therefore, to obtain an approximate sampling technique for continuous transfer functions, we may propose inverting this transformation and defining

$$s' = \frac{2}{T}\,\frac{z-1}{z+1} \tag{7.3-3}$$

An approximate discrete equivalent of the continuous transfer function is then given by

$$K(z) = K^c(s') \tag{7.3-4}$$

We call (7.3-3) the *bilinear transformation*, or BLT.

The BLT corresponds to approximating integration using the trapezoid rule, since if

$$\frac{Y(z)}{U(z)} = \frac{2}{T}\frac{z-1}{z+1} = \frac{2}{T}\,\frac{1-z^{-1}}{1+z^{-1}}$$

then (recall that z^{-1} is the unit delay in the time domain so that $z^{-1}u_k = u_{k-1}$)

$$u_k = u_{k-1} + \frac{T}{2}(y_k + y_{k-1}) \tag{7.3-5}$$

If the continuous transfer function is

$$K^c(s) = \frac{\Pi_{i=1}^{m}(s+t_i)}{\Pi_{i=1}^{n}(s+s_i)}, \tag{7.3-6}$$

with the relative degree $r = n - m > 0$, then the BLT yields the approximate discrete equivalent transfer function given by

$$\begin{aligned}
K(z) &= \frac{\Pi_{i=1}^{m}\left[\frac{2(z-1)}{T(z+1)} + t_i\right]}{\Pi_{i=1}^{m}\left[\frac{2(z-1)}{T(z+1)} + s_i\right]} \\
K(z) &= \left[\frac{T}{2}(z+1)\right]^r \frac{\Pi_{i=1}^{m}[(z-1)+(z+1)t_iT/2]}{\Pi_{i=1}^{n}[(z-1)+(z+1)s_iT/2]} \\
K(z) &= \left[\frac{T}{2}(z+1)\right]^r \frac{\Pi_{i=1}^{m}[(1+t_iT/2)z-(1-t_iT/2)]}{\Pi_{i=1}^{n}[(1+s_iT/2)z-(1-s_iT/2)]}
\end{aligned} \tag{7.3-7}$$

It can be seen that the poles and finite zeros map to the z-plane according to

$$z = \frac{1+sT/2}{1-sT/2}; \tag{7.3-8}$$

however, the r zeros at infinity in the s-plane map into zeros at $z = -1$. This is sensible since $z = -1$ corresponds to the *Nyquist frequency* ω_N, where $z = e^{j\omega_N T} = -1$, so that $\omega_N T = \pi$ or

$$\omega_N = \frac{\pi}{T} = \frac{\omega_s}{2} \tag{7.3-9}$$

This is the highest frequency before folding of $|K(e^{j\omega T})|$ occurs (see Figure 7.4-1). Since the BLT maps the left-half of the s-plane into the unit circle, it maps stable continuous systems $K^c(s)$ into stable discrete $K(z)$.

According to (7.3-7), the BLT gives discretized transfer functions that have a relative degree of zero; that is, the degrees of the numerator and denominator are the same. If

$$K(z) = \frac{b_0 z^n + b_1 z^{n-1} + \cdots + b_n}{z^n + a_1 z^{n-1} + \cdots + a_n} \tag{7.3-10}$$

and $Y(z) = K(z)U(z)$, then the difference equation relation y_k and u_k is

$$y_k = -a_1 y_{k-1} - \cdots - a_n y_{k-n} + b_0 u_k + b_1 u_{k-1} + \cdots + b_n u_{k-n} \tag{7.3-11}$$

and the current output y_k depends on the current input u_k. This is usually an undesirable state of affairs, since it takes some computation time for the microprocessor to compute y_k. Techniques for including the computation time will be discussed later.

If the continuous-time controller is given in the state-space form

$$\begin{aligned} \dot{x} &= A^c x + B^c u \\ y &= Cx + Du, \end{aligned} \tag{7.3-12}$$

one may use the Laplace transform and (7.3-3) to show that the discretized system using the BLT is given by Hanselmann (1987) as

$$\begin{aligned} x_{k+1} &= A x_k + B_1 u_{k+1} + B_0 u_k \\ y_k &= C x_k + D u_k, \end{aligned} \tag{7.3-13}$$

with

$$\begin{aligned} A &= \left[I - A^c \frac{T}{2}\right]^{-1} \left[I + A^c \frac{T}{2}\right] \\ B_1 = B_0 &= \left[I - A^c \frac{T}{2}\right]^{-1} \frac{T}{2} B^c \end{aligned} \tag{7.3-14}$$

Note that the discretized system is not a traditional state-space system since x_{k+1} depends on u_{k+1}. Aside from computation time delays, this is not a problem in our applications, since all we require of (7.3-13) is to implement it on a microprocessor. Since (7.3-13) is only a set of difference equations, this is easily accomplished. We illustrate how to discretize a continuous-time controller using the BLT in Examples 7.3-1 and 7.3-2, where we design a digital PID controller and a digital pitch-rate controller.

Matched Pole Zero

The second popular approximation technique for converting a continuous transfer function to a discrete one is the MPZ method. Here, both the poles and finite zeros are mapped into the z-plane using the transformation e^{sT} as follows:

1. If $K^c(s)$ has a pole (or finite zero) at $s = s_i$, then $K(z)$ will have a pole (or finite zero) at
$$z_i = e^{s_i T} \tag{7.3-15}$$
2. If the relative degree of $K^c(s)$ is r, so that it has r zeros at infinity, r zeros of $K(z)$ are taken at $z = -1$ by multiplying by the factor $(1 + z)^r$.
3. The gain of $K(z)$ is selected so that the dc gains of $K^c(s)$ and $K(z)$ are the same, that is, so that
$$K(1) = K^c(0) \tag{7.3-16}$$

An alternative to step 2 is to map only $r - 1$ of the infinite s-plane zeros into $z = -1$. This leaves the relative degree of $K(z)$ equal to 1, which allows one sample period for control computation time. We will call this the *modified MPZ* method.

Thus, if
$$K^c(s) = \frac{\Pi_{i=1}^{m}(s + t_i)}{\Pi_{i=1}^{n}(s + s_i)} \tag{7.3-17}$$

and the relative degree is $r = n - m$, the MPZ discretized transfer function is

$$K(z) = k(z + 1)^{r-1} \frac{\Pi_{i=1}^{m}(z - e^{-t_i T})}{\Pi_{i=1}^{n}(z - e^{s_i T})}, \tag{7.3-18}$$

where the gain k is chosen to ensure (7.3-16). Note that if $K^c(s)$ is stable, so is the $K(z)$ obtained by the MPZ, since $z = e^{sT}$ maps the left-half s-plane into the unit circle in the z-plane. Although the MPZ requires simpler algebra than the BLT, the latter is more popular in industry.

Digital Design Examples

Now let us show some examples of digital controller design using the BLT and MPZ to discretize continuous controllers.

Example 7.3-1: Discrete PID Controller Since the continuous PID controller is so useful in aircraft control design, let us demonstrate how to discretize it to obtain a digital PID controller. A standard continuous-time PID controller has the transfer function (Åström and Wittenmark, 1984)

$$K^c(s) = k\left[1 + \frac{1}{T_I s} + \frac{T_D s}{1 + T_D s/N}\right], \tag{1}$$

where k is the proportional gain, T_I is the integration time constant or "reset" time, and T_D is the derivative time constant. Rather than use pure differentiation, a "filtered derivative" is used that has a pole far left in the s-plane at $s = -N/T_D$. A typical value for N is 3 to 10; it is usually fixed by the manufacturer of the controller.

Let us consider a few methods of discretizing (1) with sample period T seconds.

(a) *BLT*. Using the BLT, the discretized version of (1) is found to be

$$K(z) = k\left[1 + \frac{1}{T_1\frac{2(z-1)}{T(z+1)}} + \frac{T_D\frac{2(z-1)}{T(z+1)}}{1 + \frac{T_D}{N}\frac{2(z-1)}{T(z+1)}}\right] \tag{2}$$

or, on simplifying,

$$K(z) = k\left[1 + \frac{T}{T_{Id}}\frac{z+1}{z-1} + \frac{T_{Dd}}{T}\frac{z-1}{z-v}\right] \tag{3}$$

with the discrete integral and derivative time constants

$$T_{Id} = 2T_I \tag{4}$$

$$T_{Dd} = \frac{NT}{1 + NT/2T_D} \tag{5}$$

and the derivative-filtering pole at

$$v = \frac{1 - NT/2T_D}{1 + NT/2T_D} \tag{6}$$

(b) *MPZ*. Using the MPZ approach to discretize the PID controller yields

$$K(z) = k\left[1 + \frac{k_1(z+1)}{T_I(z-1)} + \frac{k_2 N(z-1)}{z - e^{-NT/T_D}}\right], \tag{7}$$

where k_1 and k_2 must be selected to match the dc gains. At dc, the D terms in (1) and (7) are both zero, so we may select $k_2 = 1$. The dc values of the I terms in (1) and (7) are unbounded. Therefore, to select k_1 let us match the low-frequency gains. At low frequencies, $e^{j\omega T} \approx 1 + j\omega T$. Therefore, for small ω, the I terms of (1) and (7) become

$$K^c(j\omega) = \frac{1}{j\omega T_I}$$

$$K(e^{j\omega T}) \approx \frac{2k_1}{T_I(j\omega T)},$$

and to match them, we require that $k_1 = T/2$.

Thus, using the MPZ the discretized PID controller again has the form (3), but now with

$$T_{Id} = 2T_I \tag{8}$$

$$T_{Dd} = NT \tag{9}$$

$$v = e^{-NT/T_D} \tag{10}$$

(c) *Modified MPZ.* If we use the modified MPZ method, then in the I term in (7) the factor $(z + 1)$ does not appear. Then the normalizing gain k_1 is computed to be T. In this case, the discretized PID controller takes on the form

$$K(z) = k\left[1 + \frac{T}{T_{Id}}\frac{1}{z-1} + \frac{T_{Dd}}{T}\frac{z-1}{z-v}\right], \tag{11}$$

with

$$T_{Id} = T_I \tag{12}$$

$$T_{Dd} = NT \tag{13}$$

$$v = e^{-NT/T_D} \tag{14}$$

Now, there is a control delay of one sample period (T s) in the integral term, which could be advantageous if there is a computation delay.

(d) *Difference Equation Implementation.* Let us illustrate how to implement the modified MPZ PID controller (11) using difference equations, which are easily placed into a software computer program. It is best from the point of view of numerical accuracy in the face of computer round-off error to implement digital controllers as several first- or second-order systems in parallel. Such a parallel implementation may be achieved as follows.

First, write $K(z)$ in terms of z^{-1}, which is the unit delay in the time domain (i.e., a delay of T s, so that, for instance, $z^{-1}u_k = u_{k-1}$), as

$$K(z^{-1}) = k\left[1 + \frac{T}{T_{Id}}\frac{z^{-1}}{1-z^{-1}} + \frac{T_{Dd}}{T}\frac{1-z^{-1}}{1-vz^{-1}}\right] \tag{15}$$

[*Note:* There is some abuse in notation in denoting (15) as $K(z^{-1})$; this, we will accept.]

Now, suppose that the control input u_k is related to the tracking error as

$$u_k = K(z^{-1})\, e_k \tag{16}$$

Then, u_k may be computed from past and present values of e_k using auxiliary variables as follows:

$$v_k^I = v_{k-1}^I + \frac{T}{T_{Id}} e_{k-1} \tag{17}$$

$$v_k^D = \nu v_{k-1}^D + \frac{T_{Dd}}{T}(e_k - e_{k-1}) \tag{18}$$

$$u_k = k(e_k + v_k^I + v_k^D) \tag{19}$$

The variables v_k^I and v_k^D represent the integral and derivative portions of the PID controller, respectively. For more discussion, see Åström and Wittenmark (1984). ■

Example 7.3-2: Digital Pitch-Rate Controller via BLT In Example 5.5-3 we designed a pitch-rate control system using LQ output feedback techniques. Here we demonstrate how to convert that continuous control system into a digital control system. The BLT is popular in industry; therefore, we will use it here.

The continuous controller is illustrated in Figure 7.3-1, where

$$K_1^c(s) = \frac{k_I}{s} \tag{1}$$

$$K_2^c(s) = \frac{10k_\alpha}{s+10} \tag{2}$$

$$K_3^c(s) = k_q \tag{3}$$

The most suitable feedback gains in Example 5.5-3 were found using derivative weighting design to be

$$k_I = 1.361, \quad k_\alpha = -0.0807, \quad k_q = -0.475 \tag{4}$$

A digital control scheme with the same structure is shown in Figure 7.3-2. We have added samplers with period T to produce the samples of pitch rate, q, and angle

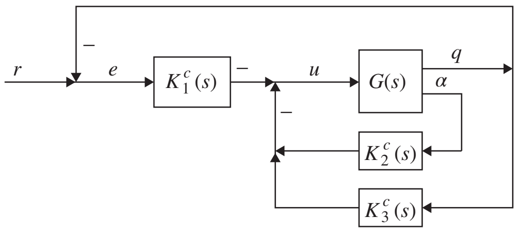


Figure 7.3-1 Continuous pitch-rate controller.

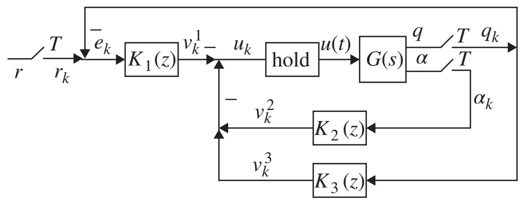


Figure 7.3-2 Digital pitch-rate controller.

of attack, α, as well as a hold device to convert the control samples u_k computed by the digital controller back to a continuous-time control input $u(t)$ for the plant. Note that the reference input $r(t)$ must also be sampled.

Since the integrator and alpha smoothing filter are part of the digital controller, the continuous dynamics $G(s)$ in Figure 7.3-2 are given by $\dot{x} = Ax + Bu$, $y = Cx$, with

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -20.2 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ 20.2 \end{bmatrix}$$

$$C = \begin{bmatrix} 57.2958 & 0 & 0 \\ 0 & 57.2958 & 0 \end{bmatrix}, \tag{5}$$

where

$$x = \begin{bmatrix} \alpha \\ q \\ \delta_e \end{bmatrix}, \qquad y = \begin{bmatrix} \alpha \\ q \end{bmatrix} \tag{6}$$

Using the BLT, the discrete equivalents to (1) to (3) are found to be

$$K_1(z) = k_1 \frac{z+1}{z-1} \quad \text{with } k_1 = \frac{k_I T}{2} \tag{7}$$

$$K_2(z) = k_2 \frac{z+1}{z-\pi} \quad \text{with } k_2 = \frac{10k_\alpha T}{10T+2}, \quad \pi = \frac{1-10T/2}{1+10T/2} \tag{8}$$

$$K_3(z) = k_q \tag{9}$$

Defining the intermediate signals v_k^1, v_k^2, v_k^3 shown in Figure 7.3-2 and denoting the unit delay in the time domain by z^{-1}, we may express (7) to (9) in terms of difference equations as follows:

$$e_k = r_k - q_k, \tag{10}$$

$$v_k^1 = k_1 \frac{1+z^{-1}}{1-z^{-1}} e_k$$

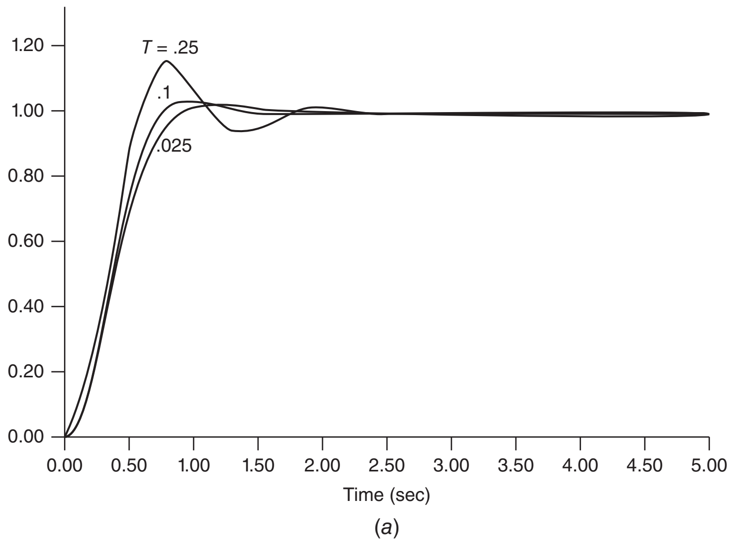
```
C  DIGITAL PITCH RATE CONTROLLER

      SUBROUTINE DIG(IK,T,X)
      REAL X(*), K(2), KI, KA, KQ
      COMMON/CONTROL/U
      COMMON/OUTPUT/AL,Q,UPLOT
      DATA REF, KI,KA,KQ/1., 1.361,-0.0807,-0.475/

      K(1)= KI*T/2
      K(2)= 10*KA*T/(10*T + 2)
      P= (1 - 10*T/2) / (1 + 10*T/2)

      E= REF - Q
      V1= V1   + K(1)*(E + EKM1)
      V2= P*V2 + K(2)*(AL + ALKM1)
      V3= KQ*Q
      U= -(V1 + V2 + V3)
      UPLOT= U

      EKM1= E
      ALKM1= AL

      RETURN
      END

C  CONTINUOUS SHORT PERIOD DYNAMICS

      SUBROUTINE F(TIME,X,XP)
      REAL X(*), XP(1)
      COMMON/CONTROL/U
      COMMON/OUTPUT/AL,Q

      XP(1)=  -1.01887*X(1) + 0.90506*X(2) - 0.00215*X(3)
      XP(2)=   0.82225*X(1) - 1.07741*X(2) - 0.17555*X(3)
      XP(3)=                                - 20.2 *X(3) + 20.2*U

      AL  =  57.2958*X(1)
      Q   =               57.2958*X(2)

      RETURN
      END
```

Figure 7.3-3 Digital simulation software: (*a*) FORTRAN subroutine to simulate digital pitch-rate controller; (*b*) subroutine $F(t, x, \dot{x})$ to simulate continuous plant dynamics.

or

$$v_k^1 = v_{k-1}^1 + k_1(e_k + e_{k-1}), \tag{11}$$

$$v_k^2 = k_2 \frac{1 + z^{-1}}{1 - \pi z^{-1}} \alpha_k$$



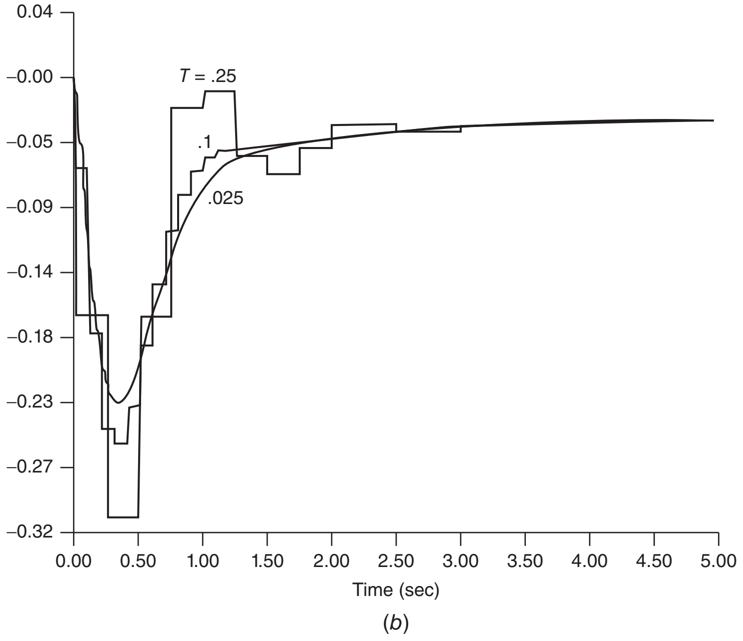


Figure 7.3-4 Effect of sampling period: (*a*) step response $q(t)$; (*b*) control input $u(t)$.

or

$$v_k^2 = \pi v_{k-1}^2 + k_2(\alpha_k + \alpha_{k-1}), \tag{12}$$

$$v_k^3 = k_q q_k \tag{13}$$

The control samples u_k are thus given by

$$u_k = -(v_k^1 + v_k^2 + v_k^3) \tag{14}$$

Note the low-pass filtering effects manifested by the averaging of e_k and α_k that occurs in these equations. This will tend to average out any measurement noise.

These difference equations describe the digital controller and are easily implemented on a microprocessor. First, however, the controller should be simulated. The Fortran subroutine in Figure 7.3-3*a* may be used with the driver program in Figure 7.2-4 to simulate the digital control law. The subroutine $F(t, x, \dot{x})$ required by the Runge-Kutta integrator for the continuous plant dynamics (5) is given in Figure 7.3-3*b*.

The step response using this digital controller was plotted for several sampling periods T in Figure 7.3-4. A zero-order hold was used. Note that the step response improves as T becomes small. Indeed, the response for $T = 0.025$ s is indistinguishable from the response using a continuous controller in Example 5.5-3*c*.

The motivation for selecting $T = 0.025$ s was as follows. The settling time of the continuous controller step response in Example 5.5-3*c* was $t_s = 1$ s. The settling time is about four times the slowest time constant, which is thus 0.25 s. The sampling period should be selected about one-tenth of this for good performance. ■

7.4 MODIFIED CONTINUOUS DESIGN

In Section 7.3 we showed how to convert a continuous-time controller to a digital controller using the BLT and MPZ. However, that technique is only an approximate one that gives worse results as the sample period T increases. In this section we show how to *modify the design of the continuous controller* so that it yields a more suitable digital controller. This allows the use of larger sample periods. To do this we will take into account some properties of the zero-order-hold and sampling processes. Using modified continuous design, we are able to design in Example 7.4-1 a digital pitch-rate control system that works extremely well even for relatively large sample periods.

Sampling, Hold Devices, and Computation Delays

We will examine some of the properties of the discretization and implementation processes to see how the continuous controller may be designed in a fashion that will yield an improved digital controller. Specifically, in the design of the continuous controller it is desirable to include the effects of sampling, hold devices, and computation delays.

Sampling and Aliasing We would like to gain some additional insight on the sampling process (Oppenheim and Schafer, 1975; Franklin and Powell, 1980; Åström and Wittenmark, 1984). To do so, define the Nyquist frequency $\omega_N = \omega_s/2 = \pi/T$

and the sampling frequency $\omega_s = 2\pi / T$ and picture the output $y^*(t)$ of the sampler with input $y(t)$ as the string of impulses

$$y^*(t) = \sum_{k=-\infty}^{\infty} y(t)\delta(t - kT), \tag{7.4-1}$$

where $\delta(t)$ is the unit impulse. Since the impulse train is periodic, it has a Fourier series that may be computed to be

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{jn\omega_s t} \tag{7.4-2}$$

Using this in (7.4-1) and taking the Laplace transform yield

$$Y^*(s) = \frac{1}{T} \int_{-\infty}^{\infty} y(t) \left[\sum_{n=-\infty}^{\infty} e^{jn\omega_s t} \right] e^{-st} dt$$

$$Y^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} y(t) e^{-(s-jn\omega_s)t} dt$$

$$Y^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} Y(s - jn\omega_s), \tag{7.4-3}$$

where $Y(s)$ is the Laplace transform of $y(t)$ and $Y^*(s)$ is the Laplace transform of the sampled signal $y^*(t)$. Due to the factor $1/T$ appearing in (7.4-3), the sampler is said to have a *gain of* $1/T$.

Sketches of a typical $Y(j\omega)$ and $Y^*(j\omega)$ are shown in Figure 7.4-1, where ω_H is the highest frequency contained in $y(t)$. Notice that the digital frequency response is symmetric with respect to ω_N and periodic with respect to ω_s. At frequencies less than ω_N,

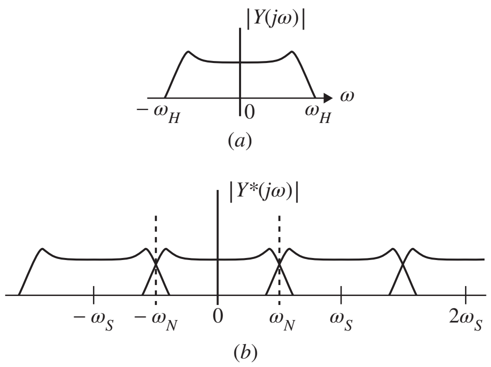


Figure 7.4-1 Sampling in the frequency domain: (*a*) spectrum of $y(t)$; (*b*) spectrum of sampled signal $y^*(t)$.

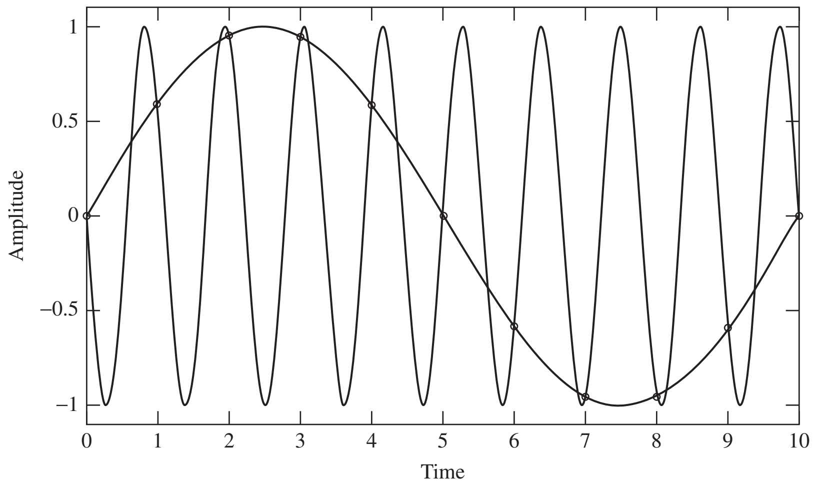


Figure 7.4-2 Example of aliasing in the time domain.

the spectrum of $Y^*(j\omega)$ has two parts: one part comes from $Y(j\omega)$ and is the portion that should appear. However, there is an additional portion from $Y(j(\omega - \omega_s))$; the "tail" of $Y(j(\omega - \omega_s))$, which contains high-frequency information about $y(t)$, is "folded" back or *aliased* into the lower frequencies of $y^*(j\omega)$. Thus, the high-frequency content of $y(t)$ appears at low frequencies and can lead to problems in reconstructing $y(t)$ from its samples.

If $\omega_H < \omega_N$, the tail of $Y^*(j(\omega - \omega_s))$ does not appear to the right of $\omega - \omega_N$ and $y(t)$ can be uniquely reconstructed from its samples by low-pass filtering. This condition is equivalent to

$$\omega_s > 2\omega_H, \tag{7.4-4}$$

which is the sampling theorem of Shannon that guarantees aliasing does not occur.

It is interesting to see what the sampling theorem means in the time domain. In Figure 7.4-2, we show two continuous signals that have the same samples. If the original signal was the higher-frequency signal, the D/A reconstruction process will produce the lower-frequency signal from the samples of the higher-frequency signal. Thus, aliasing can result in *high-frequency signals being misinterpreted as low-frequency signals*. If the sampling frequency ω_s is greater than twice the highest frequency ω_H appearing in the continuous signal, the problem depicted in the figure does not occur and the signal can be accurately reconstructed from its samples.

Selecting the Sampling Period For control design, the sampling frequency ω_s must generally be significantly greater than twice the highest frequency of any signal appearing in the system. That is, in control applications the sampling theorem does not usually provide much insight in selecting ω_s. Some guides for selecting the sampling period T are now discussed.

If the continuous-time system has a single dominant complex pole pair with natural frequency of ω, the rise time is given approximately by

$$t_r = \frac{1.8}{\omega} \tag{7.4-5}$$

It is reasonable to have at least two to four samples per rise time so that the error induced by ZOH reconstruction is not too great during the fastest variations of the continuous-time signal (Åström and Wittenmark, 1984). Then we have $t_r = 1.8/\omega \geq 4T$, or approximately

$$T \leq \frac{1}{2\omega} \tag{7.4-6}$$

However, if high-frequency components are present up to a frequency of ω_H radians and it is desired to retain them in the sampled system, a rule of thumb is to select

$$T \leq \frac{1}{4\omega_H} \tag{7.4-7}$$

These formulas should be used with care, and to select a suitable T it may be necessary to perform digital control designs for several values of T for each case carrying out a computer simulation of the behavior of the plant under the influence of the proposed controller. Note particularly that using continuous redesign of digital controllers with the BLT or MPZ, even smaller sample periods may be required since the controller discretization technique is only an approximate one.

Zero-Order Hold The D/A hold device in Figure 7.1-1 is required to reconstruct the plant control input $u(t)$ from the samples u_k provided by the digital control scheme. The ZOH is usually used. There, we take

$$u(t) = u(kT) = u_k, \qquad kT \leq t < (k+1)T, \tag{7.4-8}$$

with u_k the kth sample of $u(t)$. The ZOH yields the sort of behavior in Figure 7.2-2 and has the impulse response shown in Figure 7.4-3. This impulse response may be written as

$$h(t) = u_{-1}(t) - u_{-1}(t-T),$$

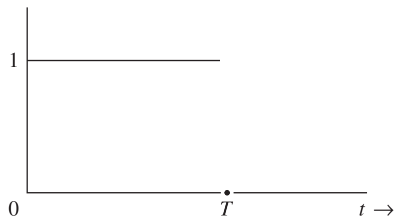


Figure 7.4-3 ZOH impulse response.

with $u_{-1}(t)$ the unit step. Thus, the transfer function of the ZOH is

$$G_0(s) = \frac{1 - e^{-sT}}{s} \tag{7.4-9}$$

To determine the Bode magnitude and phase of $G_0(s)$, write

$$G_0(j\omega) = \frac{1 - e^{-j\omega T}}{j\omega} = e^{-j\omega T/2}\frac{e^{j\omega T/2} - e^{-j\omega T/2}}{j\omega}$$

$$G_0(j\omega) = Te^{-j\omega T/2}\frac{\sin(\omega T/2)}{\omega T/2} = Te^{-j\omega T/2}\operatorname{sinc}\frac{\omega}{\omega_s}, \tag{7.4-10}$$

where $\operatorname{sinc} x \equiv (\sin \ \pi x)/\pi x$. The magnitude and phase of the ZOH are shown in Figure 7.4-4. Note that the ZOH is a low-pass filter of magnitude $T|\operatorname{sinc}(\omega/\omega_s)|$ with a phase of

$$\angle\text{ZOH} = -\frac{\omega T}{2} + \theta = -\frac{\pi\omega}{\omega_s} + \theta, \qquad \theta = \begin{cases} 0, & \sin\frac{\omega T}{2} > 0 \\ \pi, & \sin\frac{\omega T}{2} < 0 \end{cases} \tag{7.4-11}$$

According to (7.4-10), for frequencies ω much smaller than ω_s, the ZOH may be approximated by

$$G_0(s) \approx Te^{-sT/2}, \tag{7.4-12}$$

that is, by a pure delay of half the sampling period and a scale factor of T.

As we saw in the digital pitch-rate controller in Example 7.3-1, the performance of the digital controller deteriorates with increasing T, so that sample periods are required which may be too small. (We note that smaller values of T require faster computation to compute u_k; thus a faster, and more expensive, microprocessor may be required for small T.) This deterioration is partly due to the delay introduced by the hold device. We will soon see how to take this delay into account *while designing the continuous controller*, so that discretization yields a digital controller that gives suitable performance for larger values of T.

Computation Delay If the microprocessor is fast so that the time Δ required to compute the digital control law is negligible, Δ will have little effect when a digital controller is implemented. However, if Δ is appreciable, it can have a deleterious effect on the closed-loop response. Then it may be necessary to account for it.

If $\Delta \leq T$, the computation delay may be accounted for by ensuring that u_k depends only on *previous* values of the outputs. This may often be achieved by using the modified MPZ approach for digital controller design. However, the BLT is more popular and it always yields a u_k that depends on *current* values of the outputs [see (7.3-11)]. Moreover, the discrete PID controller (see Example 7.3-1) always has a dependence on the current outputs through the derivative term, even if the modified MPZ is used.

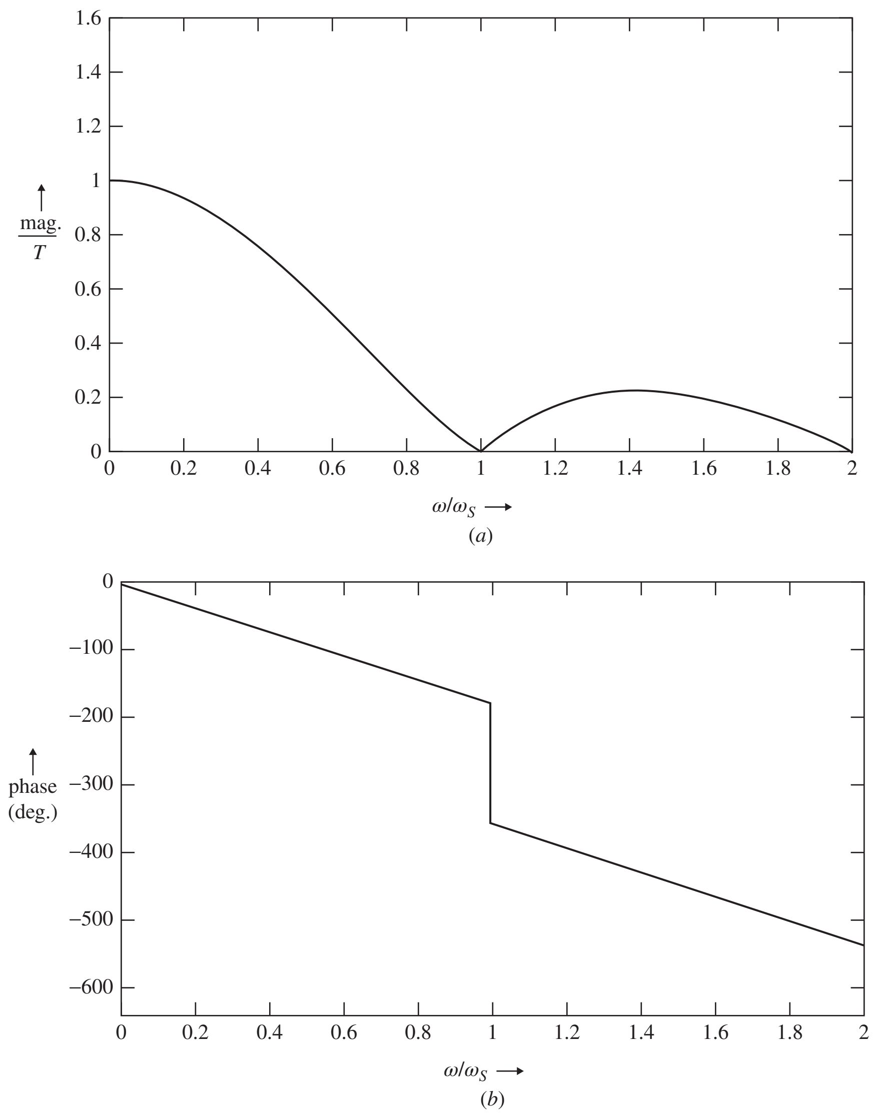


Figure 7.4-4 ZOH Bode plots: (*a*) magnitude; (*b*) phase.

If the computation delay is not negligible but is only a fraction of T, it seems inefficient to allow it to cause a delay of a full T seconds in applying the control to the plant. If there is noise present in the system, then using outputs delayed by an entire sample period to compute u_k can, for large sample periods, lead to significant deterioration over using more recent outputs to compute u_k.

Modified Continuous Design Procedures

We will now show how to account for the hold delay and computation delay *while designing the continuous controller* $K^c(s)$ for discretization. Then, when the BLT or MPZ is used to discretize $K^c(s)$, a digital controller $K(z)$ with improved performance will be obtained. We call this approach *modified continuous controller design for discretization*.

A disadvantage of modified continuous design techniques is that the sample period T must be selected prior to the continuous controller design. However, good software makes it easy to redesign the continuous controller with a different value of T. The advantage of the approach is that the effects of the sampling and hold operations, computation delay, and aliasing are apparent *while the continuous design is being performed*. Thus, they may be to some extent compensated for.

Modified continuous design can often allow significantly larger sample periods than direct application of the BLT or MPZ to a continuous controller designed with no consideration that the next step will be conversion to a digital control law. This will be illustrated in Example 7.4-1, where we design a pitch-rate control system by modified continuous design.

Let us discuss aliasing, computation delays, and then the ZOH.

Aliasing The plant $G(s)$ is generally a low-pass filter. We have seen in Figure 7.4-1 that as long as the sampling frequency ω_s is selected at least twice as large as the plant cutoff frequency ω_H, the effects of aliasing will be small.

However, one type of signal appearing in the closed-loop system that may not be bandlimited is *measurement noise*. High-frequency measurement noise may be aliased down to lower frequencies that are within the plant bandwidth and thus have a detrimental effect on system performance. To avoid this, low-pass *anti-aliasing filters* of the form

$$H_a(s) = \frac{a}{s+a} \tag{7.4-13}$$

may be inserted after the measuring devices and before the samplers. The cutoff frequency a should be selected less than $\omega_N = \omega_s/2$, so that there is good attenuation beyond ω_n rad/s.

If the cutoff frequency of the anti-aliasing filter is not much higher than the plant cutoff frequency, the filter will affect the closed-loop performance, and it should be appended to the plant *at the design stage* so that the continuous controller is designed taking it into account. See Figure 7.4-5, which represents the actual plant $G(s)$ augmented by various filters, some still to be discussed, that should be taken into account in the design stage.

Computation Delay The delay associated with a computation time of Δ has a transfer function of

$$G_{\text{comp}}(s) = e^{-s\Delta}, \tag{7.4-14}$$

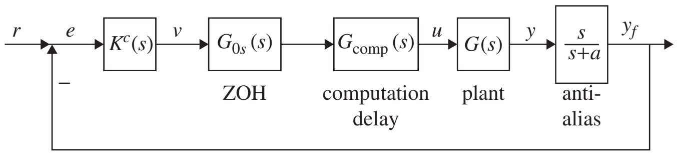


Figure 7.4-5 Modified continuous plant with anti-aliasing filter and compensation to model hold device and computation delays.

which has a magnitude of 1 and a phase of $-\omega\Delta$ radians. To account for this delay, we may perform the continuous controller design not on the plant $G(s)$ but on $G(s)e^{-s\Delta}$. However, it is awkward to design a controller for a plant whose transfer function is not rational (Franklin et al., 1980). It is more convenient to approximate the delay with a rational transfer function.

For this purpose, we may use Padé approximants to $e^{-s\Delta}$, which match the first few terms of the Taylor series expansion (Su, 1971; Franklin et al., 1980). Table 7.4-1 gives several Padé approximants to $e^{-s\Delta}$. These approximants match the first $n + m + 1$ terms of the Taylor series expansion, where n is the denominator degree and m the numerator degree.

To perform a modified continuous design that takes into account the computation delay Δ, it is only necessary to incorporate a Padé approximant $G_{\mathrm{comp}}(s)$ to $e^{-s\Delta}$ of suitable order into the plant as shown in Figure 7.4-5. The continuous controller $K^c(s)$ designed for this modified plant is then discretized using the BLT or MPZ to produce a digital controller $K(z)$.

Notice that the Padé approximants in Table 7.4-1 having finite zeros are *non-minimum-phase*. This is a property of a pure time delay. The advantage of the modified continuous design approach is that the non-minimum-phase nature of the delayed plant manifests itself at the continuous controller design stage, so that the digital controller that results after using the BLT compensates for this problem automatically.

Zero-Order Hold Finally, let us discuss modified continuous design taking into account the ZOH. Since the sampler has a gain of $1/T$, the sampler plus ZOH has a transfer function of

$$G_{0s}(s) = \frac{1 - e^{-sT}}{sT} \tag{7.4-15}$$

Some useful approximants to $G_{0s}(s)$ are given in Table 7.4-2. These have been computed using Padé approximants of e^{-sT}, so they are not strictly speaking Padé approximants, since they only match the first $n + m$ terms of the Taylor series. They are, however, sufficiently accurate for our purposes. Note that the approximants to $G_{0s}(s)$ have unstable zeros. Modified continuous controller design taking into account $G_{0s}(s)$ involves designing a controller for $G(s)G_{0s}(s)$ (see Figure 7.4-5).

TABLE 7.4-1 Padé Approximants to $e^{-s\Delta}$ for Approximation of Computation Delay

$\frac{1}{1 + s\Delta}$	$\frac{1 - s\Delta/2}{1 + s\Delta/2}$	
$\frac{1}{1 + s\Delta + (s\Delta)^2/2}$	$\frac{1 - s\Delta/3}{1 + 2s\Delta/3 + (s\Delta)^2/6}$	$\frac{1 - s\Delta/2 + (s\Delta)^2/12}{1 + s\Delta/2 + (s\Delta)^2/12}$
$\frac{1}{1 + s\Delta + (s\Delta)^2/2 + (s\Delta)^3/6}$	$\frac{1 - s\Delta/4}{1 + 3s\Delta/4 + (s\Delta)^2/4 + (s\Delta)^3/24}$	$\frac{1 - 2s\Delta/5 + (s\Delta)^2/20}{1 + 3s\Delta/5 + 3(s\Delta)^2/20 + (s\Delta)^3/60}$

TABLE 7.4-2 Approximants to $(1 - e^{-sT})\, sT$ for Approximation of Hold Delay

$\dfrac{1}{1 + sT/2}$
$\dfrac{1 - sT/6}{1 + sT/3}$
$\dfrac{1 - sT/10 + (sT)^2/60}{1 + 2sT/5 + (sT)^2/20}$
$\dfrac{1 - sT/14 + 23(sT)^2/840 - (sT)^3/840}{1 + 3sT/7 + (sT)^2/14 + (sT)^3/120}$

Implementation It is important to realize that the anti-aliasing filter should be implemented using analog circuitry as part of the plant $G(s)$. It should immediately precede the sampler. On the other hand, $G_{comp}(s)$ is not implemented since it is a model of the computation delay; $G_{0s}(s)$ is implemented by the ZOH and the sampler, and $K^c(s)$ is discretized using the BLT or MPZ and becomes the digital controller $K(z)$.

The next example illustrates modified continuous design for discretization.

Example 7.4-1: Digital Pitch-Rate Controller via Modified Continuous Design In Example 5.5-3 we designed a continuous-time pitch-rate controller. In Example 7.3-2 we showed how to use the BLT to convert that controller into digital form. It was seen that the response was good for $T = 0.025$ s, slightly worse for $T = 0.1$ s, and unacceptable for $T = 0.25$ s.

In this example let us design a modified continuous controller which, on discretization, will yield a better digital controller using larger sample periods than the one of Example 7.3-2. We will select the sampling period in this example to be $T = 0.25$ s.

(a) *Modified Continuous-Time Plant.* To account for the effects of the hold delay we will incorporate a model of the sampling and hold processes into the continuous-time dynamical model of the aircraft as shown in Figure 7.4-5. Let us use a Padé approximant to (7.4-15). Specifically, examining Table 7.4-2, select

$$G_{0s}(s) = \frac{1 - sT/6}{1 + sT/3} = -\frac{1}{2} + \frac{9/2T}{s + 3/T} \tag{1}$$

According to Figure 7.4-5, the ZOH/sampler approximant should act as a filter on the plant control input $u(t)$. Thus, a state-variable representation of $G_{0s}(s)$ is given by

$$\begin{aligned} \dot{x}_z &= -\frac{3}{T}x_z + \frac{9}{2T}v \\ u &= x_z - \frac{1}{2}v, \end{aligned} \tag{2}$$

where $v(t)$ is the new input shown in Figure 7.4-5. With $T = 0.25$ s this becomes

$$\begin{aligned} \dot{x}_z &= -12x_z + 18v \\ u &= x_z - 0.5v \end{aligned} \tag{3}$$

We should like to propose the same control structure used in Example 5.5-3. There, an angle-of-attack filter and an integrator in the feedforward channel were used. The ZOH/sampler dynamics (3) may be augmented into the system-plus-compensator state equations by defining the augmented state

$$x = \begin{bmatrix} \alpha & q & \delta_e & \alpha_F & \varepsilon & x_z \end{bmatrix}^T \tag{4}$$

Then

$$\dot{x} = Ax + Bv + Er \tag{5}$$

$$y = Cx + Fr \tag{6}$$

$$z = Hx \tag{7}$$

with

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 & 0 & 0 & 0 \\ 0.82225 & -1.07741 & -0.17555 & 0 & 0 & 0 \\ 0 & 0 & -20.2 & 0 & 0 & 20.2 \\ 10.0 & 0 & 0 & -10 & 0 & 0 \\ 0 & -57.2958 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -12 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ 0 \\ -10.1 \\ 0 \\ 0 \\ 18 \end{bmatrix}, \qquad E = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 57.2958 & 0 & 0 \\ 0 & 57.2958 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 57.2958 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then, according to Example 5.5-3, the control input $v(t)$ is given by

$$v = -Ky = -\begin{bmatrix} k_\alpha & k_q & k_I \end{bmatrix} y = -k_\alpha \alpha_F - k_q q - k_1 \varepsilon \tag{8}$$

We are now in a position to perform the control design to select the control gains.

(b) *PI and Continuous Controls Design.* To design the continuous-time controller, let us select the performance index (PI)

$$J = \tfrac{1}{2}\int_0^{\infty} \left(q_5 t^2 e^2 + \dot{\delta}_e^2\right)\, dt \tag{9}$$

that weights elevator rate of change, since this is closely related to actuator energy. Since $e(t) = \dot{\varepsilon}(t)$, this may be written

$$J = \tfrac{1}{2}\int_0^{\infty} \left(q_5 t^2 \dot{\varepsilon}^2 + \dot{\delta}_e^2\right)\, dt, \tag{10}$$

with $\varepsilon(t)$ and $\delta_e(t)$ the deviations in the integrator output and elevator deflection. Thus, this is the PI with derivative weighting discussed in Section 5.5.

Using $q_5 = 5$ and the software described in Appendix B, we computed the optimal gain matrix

$$K = \begin{bmatrix} -0.04238 & -0.4098 & 0.8426 \end{bmatrix}, \tag{11}$$

which gave the closed-loop poles

$$\begin{aligned} s = &-2.40 \pm j4.71 \\ &-1.08, -2.76 \\ &-9.86, -25.80 \end{aligned} \tag{12}$$

The closed-loop step response of the continuous-time controller is shown in Figure 7.4-6. Note that it is comparable to the responses shown in Example 5.5-3.

Let us note that the transfer function from $v(t)$ to $q(t)$ contains the approximate ZOH/sampler dynamics described by (1) and (3). These include a pole at $s = -12$ which has no significant effect. However, they also include a non-minimum-phase zero at $s = 24$. This zero significantly changes the root locus, and the control gains (11) selected automatically by the LQ approach take this non-minimum-phase zero into account. Indeed, note the delay of approximately $T = 0.25$ s in Figure 7.4-6.

It should also be realized that, in contrast to the situation in Example 7.3-2, which relied on the continuous design from Example 5.5-3, the sampling period is now needed to write the continuous dynamics (5) and hence to design the continuous-time controller.

(c) *Digital Controller.* The modified continuous controller just designed is described by exactly the same equations as in Example 7.3-2, with, however, the modified gain K given in (11). Thus, the new digital controller is exactly the same as the one described in that example, though using the modified gains.

To examine the performance of the modified digital controller, we may use the driver program described in Section 7.2, along with the continuous-time aircraft dynamics and the subroutine DIG(IK, T, X) from Example 7.3-2 with the gains in (11). The response for $T = 0.25$ s is shown in Figure 7.4-7. Note that at this design sample period of $T = 0.25$ s, the digital control response is much like the response

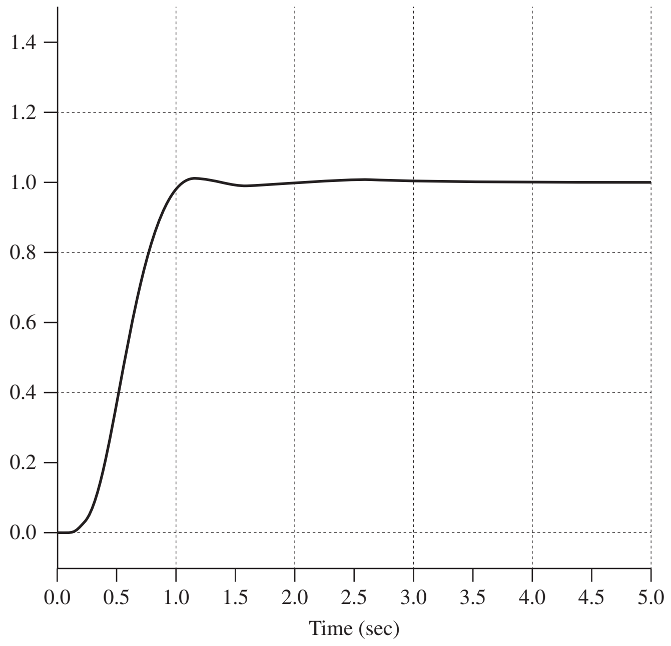


Figure 7.4-6 Step response $\boldsymbol{q}(\boldsymbol{t})$ using modified continuous-time pitch-rate controller.

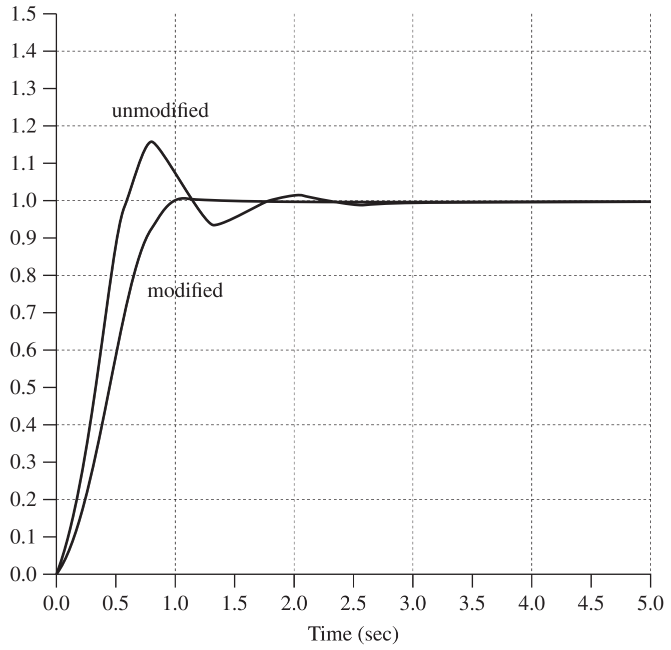


Figure 7.4-7 Response of digital controller using modified and unmodified continuous-time design.

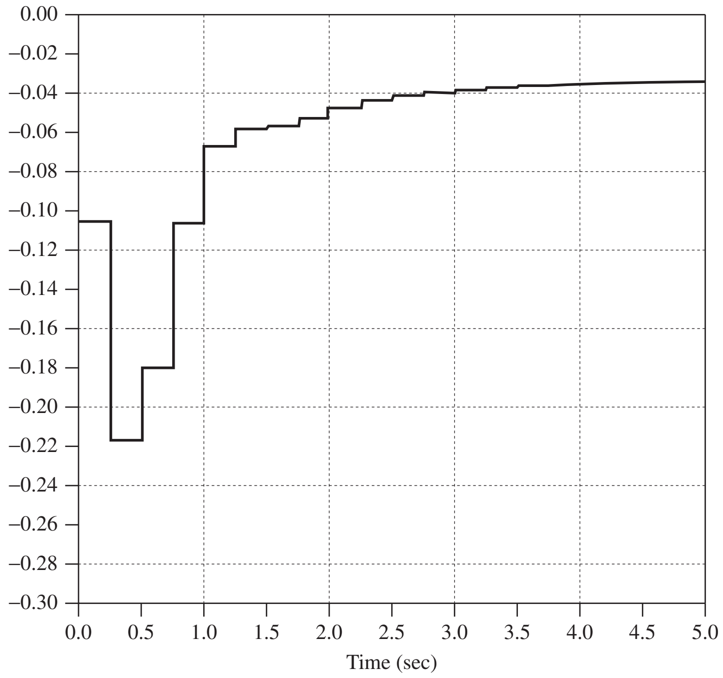


Figure 7.4-8 Control input ***u***(***t***) required for modified digital pitch-rate controller.

using the continuous-time controller shown in Figure 7.4-6. It is important to note, however, that, using the digital controller, the delay noted in Figure 7.4-6 does not appear.

For comparison we have also shown in Figure 7.4-7 the unacceptable response from Example 7.3-2 for $T = 0.25$ s. This was the result of using a digital controller, obtained simply by applying the BLT to the unmodified continuous-time controller, which did not take into account the effects of the hold delay.

The control input $u(t)$ required in the modified digital controller with $T = 0.25$ s is shown in Figure 7.4-8. It may be compared to the control signals in Example 7.3-2.

Clearly, the response shown in Figure 7.4-7 obtained using modified continuous design is excellent. It far surpasses the digital control response in Example 7.3-2 for $T = 0.25$ s. Thus, we have demonstrated that a sensible technique for taking into account some of the properties of the sample-and-hold process in the design stage of the continuous controller results in improved digital controllers that may be used with larger sample periods T. ■

7.5 IMPLEMENTATION CONSIDERATIONS

In this chapter we have discussed a design approach for digital controllers that is based on discretizing a continuous-time controller using the BLT or MPZ. It now behooves us to look at some practical considerations involved with implementing the digital controller. Our discussion will necessarily be brief, giving only an indication of some of the issues. More detail may be discovered in the work of Åström and Wittenmark (1984), Franklin and Powell (1980), Franklin et al. (1980), Phillips and Nagle (1984), Hanselmann (1987), Lewis (1992), and Slivinsky and Borninski (1987). We will mention actuator saturation and windup and controller realization structures.

Actuator Saturation and Windup

Actuator saturation is a problem that occurs in both continuous-time and digital control systems. Since it is easy to protect against by using a digital controller, we have placed it in this section.

A digital controller may be represented in the dynamic state-space form

$$x_{k+1} = Fx_k + Gw_k \tag{7.5-1}$$

$$u_k = Cx_k + Dw_{k,} \tag{7.5-2}$$

where $x_k \in \mathbf{R}^n$ is the controller state and w_k the controller input, composed generally of the tracking error and the plant measured output.

We have assumed thus far that the plant control input $u_k \in \mathbf{R}^m$ which is computed by the controller can actually be applied to the plant. However, in flight controls the plant inputs (such as elevator deflection δ_e, throttle, and so on) are limited by *maximum* and *minimum* allowable values. Thus, the relation between the *desired plant input* v_k and the *actual plant input* u_k is given by the sort of behavior shown in Figure 7.5-1, where u_H and u_L represent, respectively, the maximum and minimum control effort allowed by the mechanical actuator. Thus, to describe the actual case in an aircraft flight control system, we are forced to include *nonlinear saturation functions* in the control channels as shown in Figure 7.5-2.

Consider the simple case where the controller is an integrator with input w_k and output v_k. Then all is well as long as v_k is between u_L and u_H, for in this region the aircraft input u_k equals v_k. However, if v_k exceeds u_H, then u_k is limited to its maximum value u_H. This in itself may not be a problem. The problem arises if w_k

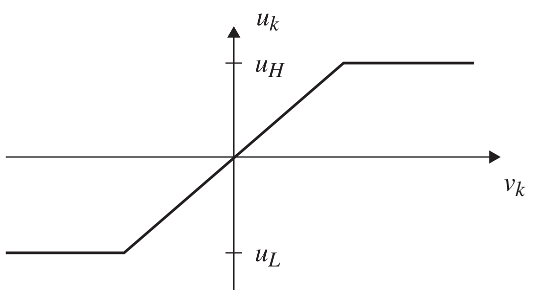


Figure 7.5-1 Actuator saturation function.

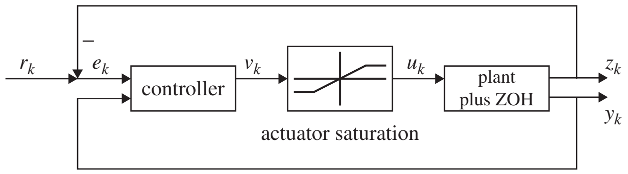


Figure 7.5-2 Flight control system including actuator saturation.

remains positive, for then the integrator continues to integrate and v_k may increase well beyond u_H. Then, when w_k becomes negative, it may take considerable time for v_k to decrease below u_H. In the meantime, u_k is held at u_H, giving an incorrect control input to the aircraft. This effect of integrator saturation is called *windup*. It arises because the controllers we design are generally dynamical in nature, which means that they store information or energy.

To correct integrator windup, it is necessary to *limit the state of the controller* so that it is consistent with the saturation effects being experienced by the plant input u_k. This is not difficult to achieve (Åström and Wittenmark, 1984). Indeed, write (7.5-2) in the form

$$0 = u_k - Cx_k - Dw_k,$$

multiply it by L, which will soon be selected, and add it to (7.5-1) to obtain

$$x_{k+1} = (F - LC)x_k + (G - LD)w_k + Lu_k \tag{7.5-3}$$

This is the form in which the digital controller should be implemented to avoid windup, as we now argue. A little thought shows that actuator windup occurs in the form (7.5-1) when F is not asymptotically stable. For then, as long as w_k in (7.5-1) is nonzero, x_k will continue to increase. However, by selecting L so that

$$F_0 = F - LC \tag{7.5-4}$$

is asymptotically stable, this problem is averted.

A special case occurs when L is selected so that F_0 has all poles at the origin. Then x_k displays *deadbeat behavior*; after n time steps it remains limited to an easily computed value dependent on the values of w_k and u_k (see the problems at the end of the chapter).

The *anti-windup gain L* may be selected to place the poles of F_0 arbitrarily if (C, F) is observable. However, as long as (C, F) is *detectable* (i.e., has all its unstable poles observable), windup may be eliminated using this technique.

To complete the design for anti-windup protection, the digital controller should be implemented in the form (7.5-3) and the aircraft control input then selected according to

$$u_k = \text{sat}\ (Cx_k + Dw_k), \tag{7.5-5}$$

where the *saturation function* (shown in Figure 7.5-1) is defined for scalars as

$$\text{sat}(v) = \begin{cases} u_H, & v \geq u_H \\ v, & u_L < v < u_H \\ u_L, & v \leq u_L, \end{cases} \tag{7.5-6}$$

with u_H and u_L the maximum and minimum allowable values, respectively. For vectors, the saturation function is defined as

$$\text{sat}(v) = \begin{bmatrix} \text{sat}(v_1) \\ \text{sat}(v_2) \\ \vdots \\ \text{sat}(v_m) \end{bmatrix} \tag{7.5-7}$$

The values of u_H and u_L for each component v_i should be selected to correspond to the actual limits on the components of the plant input u_k.

Note that the limited signal u_k is used in (7.5-3), providing a feedback arrangement in the controller with anti-windup protection. What we have in effect done is include an *observer* with dynamics F_0 in the digital controller. Since F_0 is asymptotically stable, the observer will provide reasonable "estimates" even in the event of saturation.

Where u_k is not saturated, the controller with anti-windup compensation (7.5-3), (7.5-5) is identical to (7.5-1), (7.5-2).

If the controller is given in transfer function form

$$R(z^{-1})u_k = T(z^{-1})r_k - S(z^{-1})\ w_k, \tag{7.5-8}$$

where r_k is the reference command and z^{-1} is interpreted in the time domain as a unit delay of T seconds, anti-windup compensation may be incorporated as follows.

Select a desired stable observer polynomial $A_0(z^{-1})$ and add $A_0(z^{-1})u_K$ to both sides to obtain

$$A_0 u_k = Tr_k - Sw_K + (A_0 - R)u_k \tag{7.5-9}$$

A regulator with anti-windup compensation is then given by

$$A_0 v_k = Tr_k - Sw_K + (A_0 - R)u_k \tag{7.5-10}$$

$$u_k = \text{sat}(v_k) \tag{7.5-11}$$

Example 7.5-1: Anti-Windup Compensation for Digital Proportional-Integral Controller From Example 7.3-1 a general digital proportional-integral controller is given by

$$u_k = k\left[1 + \frac{T}{T_I}\frac{1}{z-1}\right] W_k, \tag{1}$$

where we have used design by the modified MPZ to obtain a delay of T seconds in the integrator to allow for computation time. The proportional gain is k and the reset time is T_I; both are fixed in the design stage.

Multiply z^{-1} and write

$$(1 - z^{-1})u_k = k \left[\left(1 - z^{-1}\right) + \frac{Tz^{-1}}{T_I} \right] W_k, \tag{2}$$

which is in the transfer function form (7.5-8). The corresponding difference equation form for implementation is

$$u_k = u_{k-1} + kw_k + k \left(-1 + \frac{T}{T_I} \right) w_{k-1} \tag{3}$$

This controller will experience windup problems since the autoregressive polynomial $R = 1 - z^{-1}$ has a root at $z = 1$, making it marginally stable. Thus, when u_k is limited, the integrator will continue to integrate, "winding up" beyond the saturation level.

To correct this problem, select an observer polynomial of

$$A_0(z^{-1}) = 1 - \alpha z^{-1}, \tag{4}$$

which has a pole at some desirable location $|\alpha| < 1$. The design parameter α may be selected by simulation studies. Then the controller with anti-windup protection (7.5-10)/(7.5-11) is given by

$$(1 - \alpha z^{-1})v_k = k \left[1 + \left(-1 + T/T_I\right) z^{-1}\right] w_k + (1 - \alpha)z^{-1}u_k \tag{5}$$

$$u_k = \text{sat}(v_k) \tag{6}$$

The corresponding difference equations for implementation are

$$v_k = kw_k + \alpha v_{k-1} + k \left(-1 + \frac{T}{T_I} \right) w_{k-1} + (1 - \alpha)u_{k-1} \tag{7}$$

$$u_k = \text{sat}(v_k) \tag{8}$$

A few lines of Fortran code implementing this digital controller are given in Figure 7.5-3. This subroutine may be used as the control update routine DIG with the digital simulation driver program in Section 7.2.

If $\alpha = 1$, we obtain the special case (2), which is called the *position form* and has no anti-windup compensation.

If $\alpha = 0$, we obtain the *deadbeat anti-windup compensation*

$$v_k = k \left[1 + \left(-1 + T/T_I\right) z^{-1}\right] w_k + u_{k-1}, \tag{9}$$

```
C  DIGITAL PI CONTROLLER WITH ANTIWINDUP COMPENSATION

      SUBROUTINE CONUP(T)
      REAL K
      COMMON/CONTROL/U
      COMMON/OUTPUT/Z
      COMMON/REF/R
      DATA K,AL,TI,ULOW,UHIGH/ 0.5, 0.2, 5., −0.5, 0.5/

      E=  R−Z
      V=  K*E + V
      U=  AMAX1(ULOW,V)
      U=  AMIN1(UHIGH,U)
      V=  AL*V + K*(−1 + T/TI)*E + (1−AL)*U

      RETURN
      END
```

Figure 7.5-3 Fortran code implementing proportional-integral controller with anti-windup compensation.

with corresponding difference equation implementation

$$v_k = u_{k-1} + kw_k + \left(-1 + \frac{T}{T_I}\right) w_{k-1} \tag{10}$$

If u_k is not in saturation, this amounts to updating the plant control by adding the second and third terms in (10) to u_{k-1}. These terms are, therefore, nothing but $u_k - u_{k-1}$. The compensator with $\alpha = 0$ is thus called the *velocity form* of the proportional-integral controller. ■

Controller Realization Structures

Round-off errors can occur every time an arithmetic operation is performed. Moreover, since all the stable behavior of a discrete system is described by the location of the poles within the unit circle, great accuracy is required in the filter coefficients to obtain desired closed-loop pole locations.

A direct implementation of the digital filter would involve simply writing n difference equations describing (7.5-1)/(7.5-2) and would be virtually guaranteed to have severe numerical problems if n is larger. Specifically, the controller and observer canonical forms (Kailath, 1980) are notoriously unstable numerically. That is, their poles are very sensitive to small variations in their coefficients. It can be shown that the sensitivity to coefficient variations of the impulse response and frequency response is also high in direct implementations (Hanselmann, 1987). For good numerical performance with fixed-point arithmetic, digital filters should be implemented as *cascade or parallel combinations of first- and second-order filters*.

A state-space transformation may be used to place the digital filter into an appropriate form for implementation. To obtain real coefficients, the *real Jordan form* is suitable (Phillips and Nagle, 1984; Hanselmann, 1987; Lewis, 1992). This is a block-diagonal form for (7.5-1), (7.5-2) having first- and second-order blocks in cascade and parallel. Corresponding to each real pole there will be first-order blocks, and corresponding to each complex pole there will be second-order blocks.

A form suitable for implementation may also be found by performing a partial fraction expansion (PFE) on the transfer function. A technique for doing this in terms of the eigenstructure is given in Section 5.2. However, a *real PFE* should be found which will have the form (in the case of a simple matrix F)

$$H(z) = D + \sum_{i=1}^{r} H_i(z), \tag{7.5-12}$$

where $H_i(z)$ is first order for real poles and second order for complex poles. We are therefore concerned with implementing first-order filters of the form

$$H_1(z^{-1}) = \frac{b_1 z^{-1}}{1 + a_1 z^{-1}} \tag{7.5-13}$$

and second-order filters of the form

$$H_2(z^{-1}) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{7.5-14}$$

To implement $H_1(z^{-1})$, we may write

$$\begin{aligned} y_k &= H_1(z^{-1})\, u_k \\ (1 + a_1 z^{-1}) y_k &= b_1 z^{-1} u_k \\ y_k &= -a_1 y_{k-1} + b_1 u_{k-1}, \end{aligned} \tag{7.5-15}$$

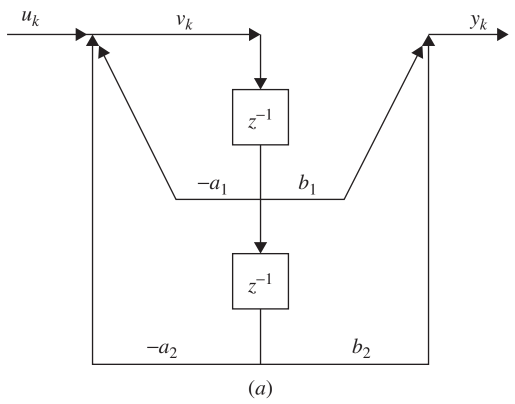
which is a difference equation that may easily be programmed.

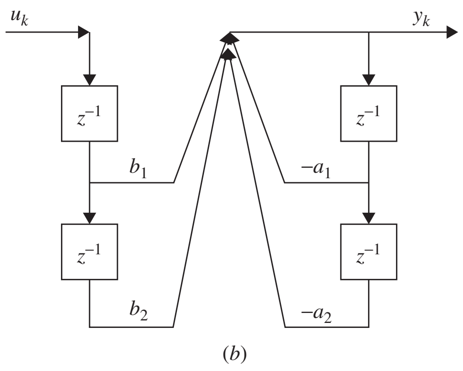
There are many ways to implement the second-order transfer function (Phillips and Nagle, 1984). Among these are *direct forms 1 through 4* (denoted D1, D2, D3, D4) and *cross-coupled forms 1 and 2* (denoted X1, X2).

In Figure 7.5-4 we show D1, D3, and X1. Forms D2, D4, and X2, respectively, are their duals (i.e., all arrows are reversed and the roles of the input and the output are interchanged). In Table 7.5-1 we give a comparison of the number of time delay elements, multipliers, and summing junctions for each form. Note that D1 and X1 conserve time delay elements, while D3 conserves summing junctions.

The difference equation implementations of these second-order modules are given in Table 7.5-2, with $y_k = H_2(z^{-1})u_k$. It is interesting that the difference equations for the X1 module may be written from the complex PFE (and hence, with a little manipulation, from the usual complex Jordan form).

When implementing these modules on a fixed-point microprocessor, it is important to incorporate overflow protection (Slivinsky and Borninski, 1987). When interconnecting them to produce $H(z)$, scaling may be introduced between the modules (Phillips and Nagle, 1984).





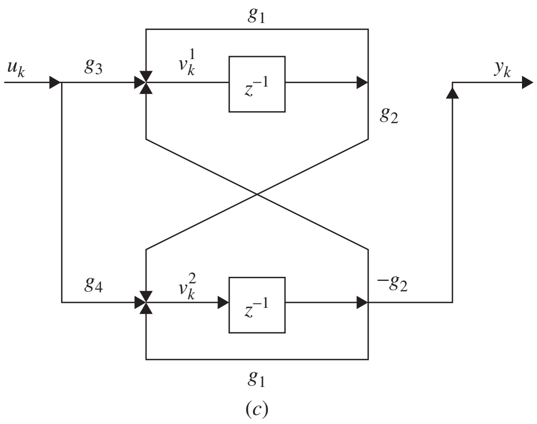


Figure 7.5-4 Implementations of second-order digital filters: (*a*) direct form 1, D1; (*b*) direct form 3, D3; (*c*) cross-coupled form 1, X1.

TABLE 7.5-1 Elements of Second-Order Modules

	Structure		
	D1	D3	X1
Time delay elements	2	4	2
Multipliers	4	4	6
Summing junctions	2	1	2

TABLE 7.5-2 Difference Equation Implementation of Second-Order Modules

D1:

$$v_k = -a_1 v_{k-1} - a_2 v_{k-2} + u_k$$
$$y_k = b_1 v_{k-1} + b_2 v_{k-2}$$

D3:

$$y_k = -a_1 y_{k-1} - a_2 y_{k-2} + b_1 u_{k-1} + b_2 u_{k-2}$$

X1:

$$v_k^1 = g_1 v_{k-1}^1 - g_2 v_{k-1}^2 + g_3 u_k$$
$$v_k^2 = g_1 v_{k-1}^2 + g_2 v_{k-1}^1 + g_4 u_k$$
$$y_k = v_{k-1}^2$$

where g_i are defined by

$$H_2(z^{-1}) = \frac{Nz^{-1}}{1+pz^{-1}} + \frac{N*z^{-1}}{1+p*z^{-1}}$$
$$g_1 = -\mathrm{Re}(p)$$
$$g_2 = -\mathrm{Im}(p)$$
$$g_3 = 2\,\mathrm{Im}(N)$$
$$g_4 = 2\,\mathrm{Re}(N)$$

7.6 SUMMARY

Since most aircraft control systems are implemented using digital signal processors, in this chapter we have outlined the basics of digital control. In Section 7.2 we discussed how to simulate digital control schemes using a Runge-Kutta integrator on the continuous aircraft dynamics. This approach yields the time responses not only at the sample points but also between the samples. It is important to check the intersample performance of the closed-loop system before implementing a digital controller on an aircraft, since it can be unsatisfactory even though all is well at the sample points.

In Section 7.3 we gave a design technique for digital controllers that is based on *redesign of an existing continuous-time controller* by discretizing it using approximation techniques like the BLT and MPZ. This results in digital controllers that require small sample periods to work properly. To overcome the requirement for unreasonably small sample periods, in Section 7.4 we showed how to modify the continuous-time controller so that, after discretization, a better digital controller is obtained that works for larger sample periods. This modification allowed the delay properties of the sample-and-hold process to be taken into account.

Finally, in Section 7.5 we mentioned some digital controller implementation considerations. We showed how to design controllers with anti-windup protection to overcome the problems of saturation of the control signals due to control limitations such as elevator deflection stops and throttle maximum limits. We gave some low-order controller structures that allow the implementation of digital controllers with maximum accuracy and efficiency.

REFERENCES

Åström, K. J., and B. Wittenmark. *Computer Controlled Systems*. Englewood Cliffs, N.J.: Prentice Hall, 1984.

Franklin, G. F., and J. D. Powell. *Digital Control*. Reading, Mass.: Addison-Wesley, 1980.

Franklin, G. F., J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Reading, Mass.: Addison-Wesley, 1980.

Hanselmann, H. "Implementation of Digital Controllers: A Survey." *Automatica* 23, no. 1 (1987): 7–32.

Kailath, T. *Linear Systems*. Englewood Cliffs, N.J.: Prentice Hall, 1980.

Lewis, F. L. *Applied Optimal Control and Estimation*. Englewood Cliffs, N.J.: Prentice Hall, 1992.

Oppenheim, A. V., and R. W. Schafer. *Digital Signal Processing*. Englewood Cliffs, N.J.: Prentice Hall, 1975.

Phillips, C. L., and H. T. Nagle, Jr. *Digital Control System Analysis and Design*. Englewood Cliffs, N.J.: Prentice Hall, 1984.

Slivinsky, C., and J. Borninski. "Control System Compensation and Implementation with the TMS32010." In *Digital Signal Processing Applications*, ed. K.-S. Lin. Englewood Cliffs, N.J.: Prentice Hall, 1987.

Su, K. L. *Time-Domain Synthesis of Linear Networks*. Englewood Cliffs, N.J.: Prentice Hall, 1971.

PROBLEMS

Section 7.3

7.3-1 Prove (7.3.13)/(7.3.14) by taking the Laplace transform of $\dot{x} = Ax + Bu$ and then using the BLT.

7.3-2 **Digital Pitch-Rate Controller.** Design a digital pitch-rate controller (see Example 7.3-2) using the MPZ technique. Simulate the step response for a few sample periods T and compare to the digital controller designed using the BLT.

7.3-3 Digital Normal Acceleration CAS. A normal acceleration CAS was designed in Example 5.4-1.

(a) Using the BLT, design a digital normal acceleration CAS controller. Simulate the time response for various sampling periods.

(b) Repeat using the modified MPZ.

7.3-4 Digital Wing Leveler. A wing leveler was designed in Example 5.5-4.

(a) Using the BLT, design a digital wing leveler. Simulate the time response for various sampling periods.

(b) Repeat using the modified MPZ.

Section 7.4

7.4-1 A Padé approximant for e^{-sT} is

$$G(s) = \frac{1 - 2sT/3 + (sT)^2/6}{1 + sT/3}$$

(a) Use long division to determine how many terms of the Taylor series of e^{-sT} are matched by $G(s)$.

(b) Use $G(s)$ to derive one of the approximants for the ZOH plus sampler shown in Table 7.4-2. How many terms of the Taylor series are matched by this approximant?

7.4-2 Digital Normal Acceleration CAS. A normal acceleration CAS was designed in Example 5.4-1. Using the BLT, design a digital normal acceleration CAS controller. Use modified continuous design, including the hold delay. Use $T = 0.1$ s. Simulate the time response and compare to the results of Problem 7.3-3.

7.4-3 Digital Wing Leveler. A wing leveler was designed in Example 5.5-4. Using the modified MPZ, design a digital wing leveler. Use modified continuous design, including the hold delay. Use $T = 0.1$ s. Simulate the time response and compare to the results of Problem 7.3-4.

Section 7.5

7.5-1 Anti-Windup Compensator

(a) Write down the value of the state x_k in the anti-windup controller (7.5-3) for the deadbeat case where F_0 has all poles at the origin. Assume that w_k and u_k are constant and that $k > n$. Note that if F_0 has all poles at the origin, then $F_0^n = 0$, where n is the dimension of F_0.

(b) Repeat for the case where the controller is just an integrator so that $x_{k+1} = x_k + (T/T_I)w_k, u_k = \text{sat}(x_k)$. Simplify as far as possible.

7.5-2 Show how to determine the X1 difference equations in Table 7.5-2 directly from the complex Jordan form blocks corresponding to a complex pair of poles.

7.5-3 **Anti-Windup Protection for Normal Acceleration CAS.** In Example 5.4-1 a normal acceleration CAS was designed; it had a PI controller in the feedforward loop. In the problems for Section 7.3 this design was digitized.

(a) Modify the digital normal acceleration CAS to add anti-windup protection.

(b) Now, set limits into the elevator actuator in your simulation program. Obtain time responses with and without the anti-windup protection.

7.5-4 **Anti-Windup Protection for Pitch-Rate CAS.** Repeat the previous problem for the pitch-rate controller in Example 5.5-3, which was digitized in the problems for Section 7.3.