Appendix W4.5 Introduction to Digital Control

As a result of the revolution in the cost-effectiveness of digital computers, there has been an increasing use of digital logic in embedded applications such as controllers in feedback systems. A digital controller gives the designer much more flexibility to make modifications to the control law after the hardware design is fixed, because the formula for calculating the control signal is in the software rather than the hardware. In many instances, this means that the hardware and software designs can proceed almost independently, saving a great deal of time. Also, it is relatively easy to include binary logic and nonlinear operations as part of the function of a digital controller as compared to an analog controller. Special processors designed for real-time signal processing and known as digital signal processors (DSPs) are particularly well suited for use as real-time controllers. Chapter 8 includes a more extensive introduction to the math and concepts associated with the analysis and design of digital controllers and digital control systems. However, in order to be able to compare the analog designs of the next three chapters with reasonable digital equivalents, we give here a brief introduction to the most simple techniques for digital designs.

A digital controller differs from an analog controller in that the signals must be **sampled** and **quantized**.[1] A signal to be used in digital logic needs to be sampled first and then the samples need to be converted by an analog-to-digital converter or A/D[2] into a quantized digital number. Once the digital computer has calculated the proper next control signal value, this value needs to be converted back into a voltage and held constant or otherwise extrapolated by a digital-to-analog converter or D/A[3] in order to be applied to the actuator of the process. The control signal is not changed until the next sampling period. As a result of sampling, there are strict limits on the speed and bandwidth of a digital controller. Discrete design methods that tend to minimize these limitations are described in Chapter 8. A reasonable rule of thumb for selecting the sampling period is that, during the rise-time of the response to a step, the input to the discrete controller should be sampled approximately six times. By adjusting the controller for the effects

[1]A controller that operates on signals that are sampled but *not* quantized is called **discrete**, while one that operates on signals that are both sampled and quantized is called **digital**.

[2]Pronounced “A to D.”

[3]Often spelled DAC and pronounced as one word to rhyme with quack.

of sampling, the sample period can be as large as two to three times per rise time. This corresponds to a sampling frequency that is 10 to 20 times the system's closed-loop bandwidth. The quantization of the controller signals introduces an equivalent extra noise into the system; to keep this interference at an acceptable level, the A/D converter usually has an accuracy of 10 to 12 bits although inexpensive systems have been designed with only 8 bits. For a first analysis, the effects of the quantization are usually ignored, as they will be in this introduction. A simplified block diagram of a system with a digital controller is shown in Fig. W4.3.

For this introduction to digital control, we will describe a simplified technique for finding a discrete (sampled but not quantized) equivalent to a given continuous controller. The method depends on the sampling period, T_s, being short enough that the reconstructed control signal is close to the signal that the original analog controller would have produced. We also assume that the numbers used in the digital logic have enough accurate bits so that the quantization implied in the A/D and D/A processes can be ignored. While there are good analysis tools to determine how well these requirements are met, here we will test our results by simulation, following the well-known advice that "The proof of the pudding is in the eating."

Finding a discrete equivalent to a given analog controller is equivalent to finding a recurrence equation for the samples of the control, which will approximate the differential equation of the controller. The assumption is that we have the transfer function of an analog controller and wish to replace it with a discrete controller that will accept samples of the controller input $e(kT_s)$ from a sampler and, using past values of the control signal $u(kT_s)$ and present and past samples of the input $e(kT_s)$, will compute the next control signal to be sent to the actuator. As an example, consider a PID controller with the transfer function

$$U(s) = (k_P + \frac{k_I}{s} + k_D s)E(s), \tag{W4.18}$$

which is equivalent to the three terms of the time-domain expression

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \dot{e}(t), \tag{W4.19}$$

$$= u_P + u_I + u_D. \tag{W4.20}$$

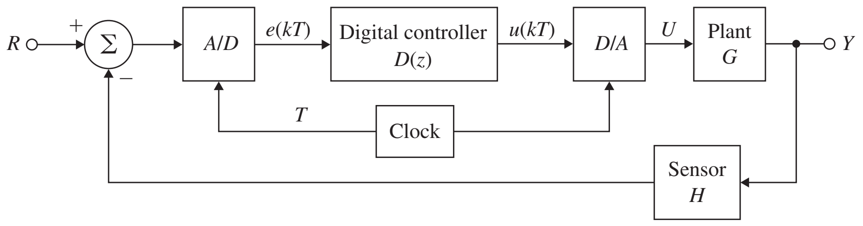


Figure W4.3
Block diagram of a digital controller

Based on these terms and the fact that the system is linear, the next control sample can be computed term-by-term. The proportional term is immediate:

$$u_P(kT_s + T_s) = k_P e(kT_s + T_s). \tag{W4.21}$$

The integral term can be computed by breaking the integral into two parts and approximating the second part, which is the integral over one sample period, as follows.

$$\begin{aligned} u_I(kT_s + T_s) &= k_I \int_0^{kT_s+T_s} e(\tau)d\tau, && \text{(W4.22)} \\ &= k_I \int_0^{kT_s} e(\tau)d\tau + k_I \int_{kT_s}^{kT_s+T_s} e(\tau)d\tau, && \text{(W4.23)} \\ &= u_I(kT_s) + \{\text{area under } e(\tau) \text{ over one period}\}, && \text{(W4.24)} \\ &\cong u_I(kT_s) + k_I \frac{T_s}{2}\{e(kT_s + T_s) + e(kT_s)\}. && \text{(W4.25)} \end{aligned}$$

In Eq. (W4.25), the area in question has been approximated by that of the trapezoid formed by the base T_s and vertices $e(kT_s+T_s)$ and $e(kT_s)$ as shown by the dashed line in Fig. W4.4.

The area also can be approximated by the rectangle of amplitude $e(kT_s)$ and width T_s shown by the solid blue in Fig. W4.4 to give $u_I(kT_s + T_s) = u_I(kT_s) + k_I T_s e(kT_s)$. These and other possibilities are considered in Chapter 8.

In the derivative term, the roles of u and e are reversed from integration and the consistent approximation can be written down at once from Eqs. (W4.25) and (W4.19) as

$$\frac{T_s}{2}\{u_D(kT_s + T_s) + u_D(kT_s)\} = k_D\{e(kT_s + T_s) - e(kT_s)\}. \tag{W4.26}$$

As with linear analog transfer functions, these relations are greatly simplified and generalized by the use of transform ideas. At this time, the discrete transform will be introduced simply as a prediction operator z much as if we

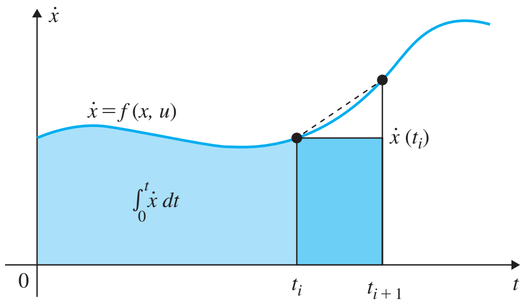


Figure W4.4
Graphical interpretation of numerical integration

described the Laplace transform variable, s, as a differential operator.[4] Here we define the operator z as the forward shift operator in the sense that if $U(z)$ is the transform of $u(kT_s)$ then $zU(z)$ will be the transform of $u(kT_s + T_s)$. With this definition, the integral term can be written as

$$zU_I(z) = U_I(z) + k_I \frac{T_s}{2}\left[zE(z) + E(z)\right], \tag{W4.27}$$

$$U_I(z) = k_I \frac{T_s}{2}\frac{z+1}{z-1}E(z), \tag{W4.28}$$

and from Eq. (W4.26), the derivative term becomes the inverse as

$$U_D(z) = k_D \frac{2}{T_s}\frac{z-1}{z+1}E(z). \tag{W4.29}$$

The complete discrete PID controller is thus described by

$$U(z) = \left(k_P + k_I \frac{T_s}{2}\frac{z+1}{z-1} + k_D \frac{2}{T_s}\frac{z-1}{z+1}\right)E(z). \tag{W4.30}$$

Trapezoid rule

Comparing the two discrete equivalents of integration and differentiation with the corresponding analog terms, it is seen that the effect of the discrete approximation in the z domain is as if everywhere in the analog transfer function the operator s has been replaced by the composite operator $\frac{2}{T_s}\frac{z-1}{z+1}$. This is the trapezoid rule[5] of discrete equivalents.

The discrete equivalent to $D_c(s)$ is

$$D_d(z) = D_c\left(\frac{2}{T_s}\frac{z-1}{z+1}\right). \tag{W4.31}$$

EXAMPLE W4.3 *Discrete Equivalent*

Find the discrete equivalent to the analog controller having transfer function

$$D_c(s) = \frac{U(s)}{E(s)} = \frac{11s+1}{3s+1}, \tag{W4.32}$$

using the sample period $T_s = 1$.

Solution. The discrete operator is $\frac{2(z-1)}{z+1}$ and thus the discrete transfer function is

$$D_d(z) = \frac{U(z)}{E(z)} = D_c(s)\Big|_{s=\frac{2}{T_s}\frac{z-1}{z+1}}, \tag{W4.33}$$

$$= \frac{11\left[\frac{2(z-1)}{z+1}\right]+1}{3\left[\frac{2(z-1)}{z+1}\right]+1}. \tag{W4.34}$$

[4]This is defined as the z-transform in Chapter 8.

[5]The formula is also called Tustin's method after the English engineer who used the technique to study the responses of nonlinear circuits.

Clearing fractions, the discrete transfer function is

$$D_d(z) = \frac{U(z)}{E(z)} = \frac{23z - 21}{7z - 5}. \tag{W4.35}$$

Converting the discrete transfer function to a discrete difference equation using the definition of z as the forward shift operator is done as follows. First we cross-multiply in Eq. (W4.35) to obtain

$$(7z - 5)U(z) = (23z - 21)E(z), \tag{W4.36}$$

and interpreting z as a shift operator, this is equivalent to the difference equation[6]

$$7u(k + 1) - 5u(k) = 23e(k + 1) - 21e(k), \tag{W4.37}$$

where we have replaced $kT_s + T_s$ with $k + 1$ to simplify the notation. To compute the next control at time $kT_s + T_s$, therefore, we solve the difference equation

$$u(k + 1) = \frac{5}{7}u(k) + \frac{23}{7}e(k + 1) - \frac{21}{7}e(k). \tag{W4.38}$$

Now let's apply these results to a control problem. Fortunately, Matlab provides us with the Simulink capability to simulate both continuous and discrete systems allowing us to compare the responses of the systems with continuous and discrete controllers.

EXAMPLE W4.4 *Equivalent Discrete Controller for Speed Control*

A motor speed control is found to have the plant transfer function

$$\frac{Y}{U} = \frac{45}{(s + 9)(s + 5)}. \tag{W4.39}$$

A PI controller designed for this system has the transfer function

$$D_c(s) = \frac{U}{E} = 1.4\frac{s + 6}{s}. \tag{W4.40}$$

The closed-loop system has a rise time of about 0.2 sec and an overshoot of about 20%. Design a discrete equivalent to this controller and compare the step responses and control signals of the two systems. (a) Compare the responses if the sample period is 0.07, which is about three samples per rise time. (b) Compare the responses with a sample period of $T_s = 0.035$ sec, which corresponds to about six samples per rise time.

[6]The process is entirely similar to that used in Chapter 3 to find the ordinary differential equation to which a rational Laplace transform corresponds.

Solution. (a) Using the substitution given by Eq. (W4.31), the discrete equivalent for $T_s = 0.07$ sec is given by replacing s by $s \leftarrow \frac{2}{0.07}\frac{z-1}{z+1}$ in $D_c(s)$ as

$$D_d(z) = 1.4\frac{\frac{2}{0.07}\frac{z-1}{z+1}+6}{\frac{2}{0.07}\frac{z-1}{z+1}}, \tag{W4.41}$$

$$= 1.4\frac{2(z-1)+6*0.07(z+1)}{2(z-1)}, \tag{W4.42}$$

$$= 1.4\frac{1.21z-0.79}{(z-1)}. \tag{W4.43}$$

Based on this expression, the equation for the control is (the sample period is suppressed)

$$u(k+1) = u(k) + 1.4*[1.21e(k+1) - 0.79e(k)]. \tag{W4.44}$$

(b) For $T_s = 0.035$ sec, the discrete transfer function is

$$D_d = 1.4\frac{1.105z-0.895}{z-1}, \tag{W4.45}$$

for which the difference equation is

$$u(k+1) = u(k) + 1.4[1.105\,e(k+1) - 0.895\,e(k)].$$

A Simulink block diagram for simulating the two systems is given in Fig. W4.5, and plots of the step responses are given in Fig. W4.6a. The respective control signals are plotted in Fig. W4.6b. Notice that the discrete controller for $T_s = 0.07$ sec results in a substantial increase in the overshoot in the step response, while with $T_s = 0.035$ sec the digital controller matches the performance of the analog controller fairly well.

For controllers with many poles and zeros, making the continuous-to-discrete substitution called for in Eq. (W4.31) can be very tedious.

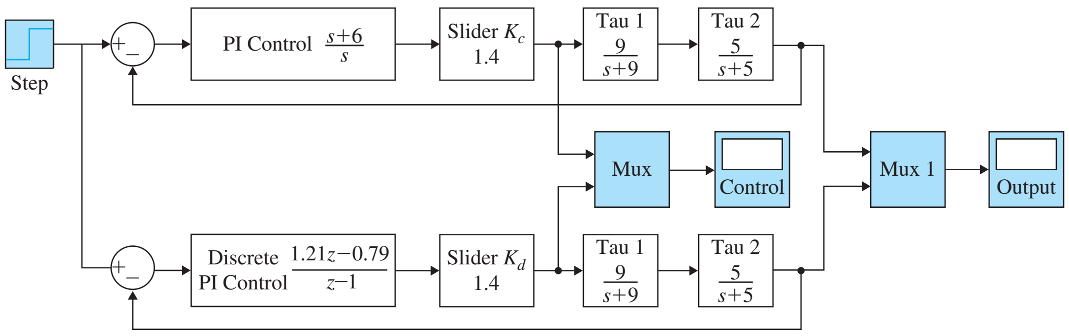


Figure W4.5
Simulink block diagram to compare continuous and discrete controllers

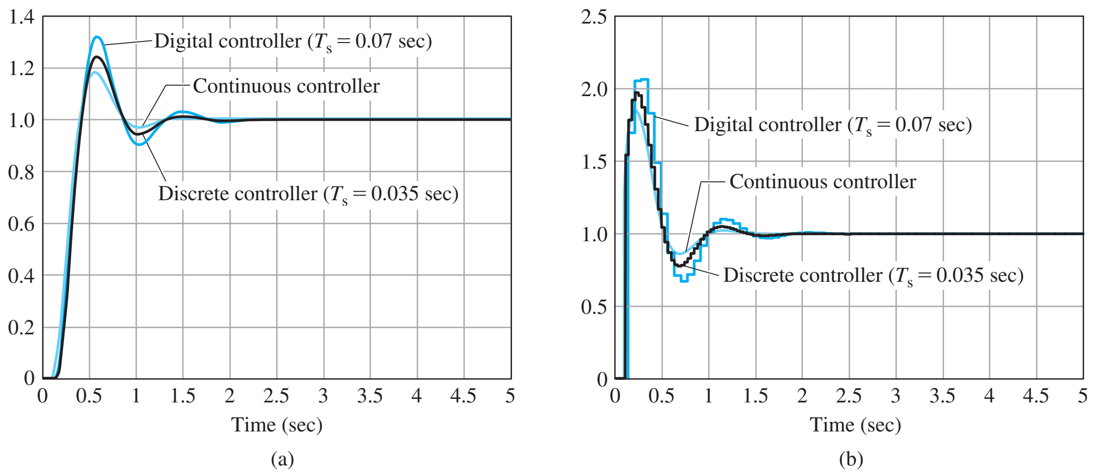


Figure W4.6

Comparison plots of a speed control system with continuous and discrete controllers: (a) output responses and (b) control signals

Fortunately, Matlab provides a command that does all the work. If one has a continuous transfer function given by $D_c(s) = \frac{\text{numD}}{\text{denD}}$ represented in Matlab as sysDc = tf(numD,denD), then the discrete equivalent with sampling period T_s is given by

$$\text{sysDd} = \text{c2d(sysDc,}T_s\text{,'t')}. \tag{W4.46}$$

In this expression, of course, the polynomials are represented in Matlab form. The last parameter in the c2d function given by 't' calls for the conversion to be done using the trapezoid (or Tustin) method. The alternatives can be found by asking Matlab for help c2d. For example, to compute the polynomials for $T_s = 0.07$ sec for Example W4.4, the commands would be

```
numD = [1 6];
denD = [1 - 0];
sysDc = tf(numD,denD)
sysDd = c2d( sysDc,0.07,'t')
```

Appendix W6.7.2 Digital Implementation of Example 6.15

EXAMPLE W6.1 *Lead Compensation for a DC Motor*

As an example of designing a lead compensator, let us repeat the design of compensation for the DC motor with the transfer function

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

that was carried out in Section 5.4.1. This also represents the model of a satellite-tracking antenna (see Fig. 3.60). This time we wish to obtain a steady-state error of less than 0.1 for a unit-ramp input. Furthermore, we desire an overshoot $M_p < 25\%$.

1. Determine the lead compensation satisfying the specifications.
2. Determine the digital version of the compensation with $T_s = 0.05$ sec.
3. Compare the step and ramp responses of both implementations.

Solution.

1. The steady-state error is given by

$$e_{ss} = \lim_{s\to 0} s\left[\frac{D_c}{1+KD_c(s)G(s)}\right]R(s), \tag{W6.1}$$

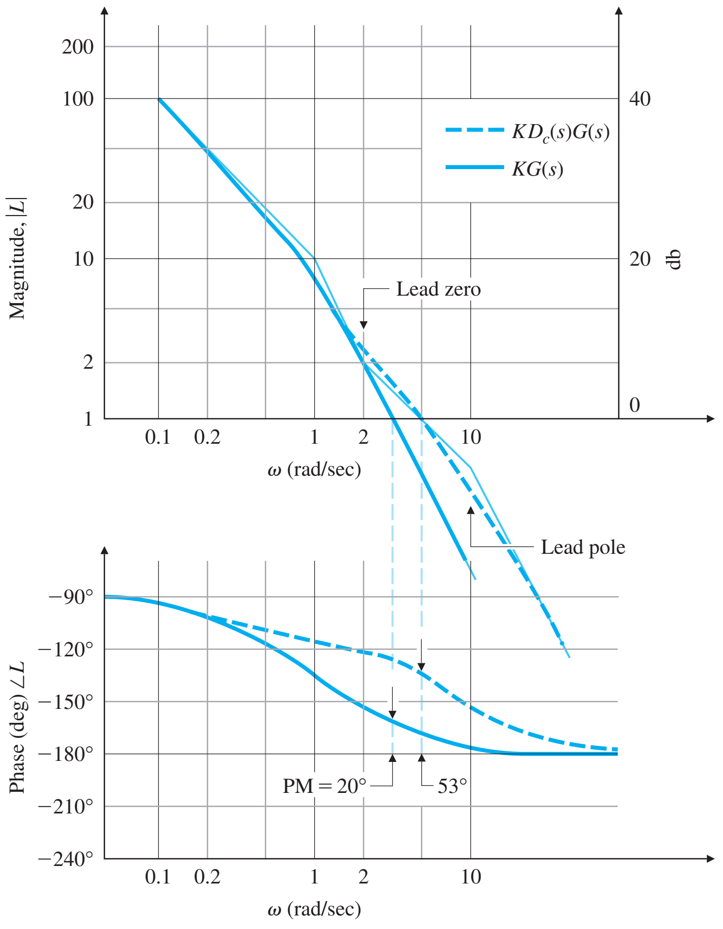
where $R(s) = 1/s^2$ for a unit ramp, so Eq. (W6.1) reduces to

$$e_{ss} = \lim_{s\to 0}\left\{\frac{1}{s+KD_c(s)[1/(s+1)]}\right\} = \frac{1}{KD_c(0)}.$$

Therefore, we find that $KD(0)$, which is the steady-state gain of the compensation, cannot be less than 10 ($K_v \geq 10$) if it is to meet the error criterion, so we pick $K = 10$. To relate the overshoot requirement to PM, Fig. 6.37 shows that a PM of 45° should suffice. The frequency response of $KG(s)$ in Fig. W6.1 shows that the PM = 20° if no phase lead is added by compensation. If it were possible to simply add phase without affecting the magnitude, we would need an additional phase of only 25° at the $KG(s)$ crossover frequency of $\omega = 3$ rad/sec. However, maintaining the same low-frequency gain and adding a compensator zero would increase the crossover frequency; hence, more than a 25° phase contribution will be required from the lead compensation. To be

Figure W6.1

Frequency response for lead-compensation design



safe, we will design the lead compensator so that it supplies a maximum phase lead of 40°. Fig. 6.53 shows that $1/\alpha = 5$ will accomplish that goal. We will derive the greatest benefit from the compensation if the maximum phase lead from the compensator occurs at the crossover frequency. With some trial and error, we determine that placing the zero at $\omega = 2$ rad/sec and the pole at $\omega = 10$ rad/sec causes the maximum phase lead to be at the crossover frequency. The compensation, therefore, is

$$KD_c(s) = 10\frac{s/2+1}{s/10+1}.$$

The frequency-response characteristics of $L(s) = KD_c(s)G(s)$ in Fig. W6.1 can be seen to yield a PM of 53°, which satisfies the design goals.

The root locus for this design, originally given as Fig. 5.24, is repeated here as Fig. W6.2, with the root locations marked for $K = 10$. The locus is not needed for the frequency-response design procedure; it

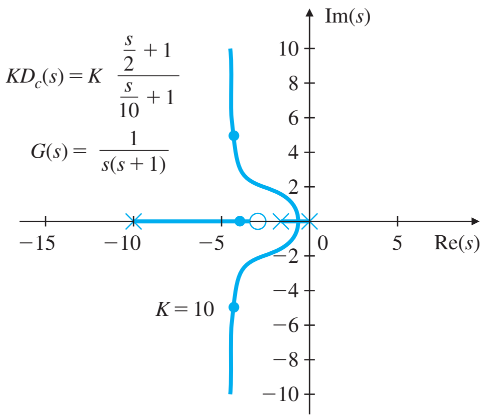


Figure W6.2
Root locus for lead-compensation design

is presented here only for comparison with the root locus design method presented in Chapter 5. The entire process can be expedited by the use of Matlab's sisotool routine, which simultaneously provides the root locus and the Bode plot through an interactive GUI interface. For this example, the Matlab statements

```
G=tf(1,[1 1 0]);
Dc=tf(10*[1/2 1],[1/10 1]);
sisotool(G,Dc)
```

will provide the plots as shown in Fig. W6.3. It also can be used to generate the Nyquist and time-response plots if desired.

2. To find the discrete equivalent of $D_c(s)$, we use the trapezoidal rule given by Eq. (W4.31). That is,

$$D_d(z) = \frac{\frac{2}{T_s}\frac{z-1}{z+1}/2+1}{\frac{2}{T_s}\frac{z-1}{z+1}/10+1}, \tag{W6.2}$$

which, with $T_s = 0.05$ sec, reduces to

$$D_d(z) = \frac{4.2z - 3.8}{z - 0.6}. \tag{W6.3}$$

This same result can be obtained by the Matlab statement

```
sysDc = tf([0.5 1],[0.1 1]);
sysDd = c2d(sysDc, 0.05, 'tustin').
```

Because

$$\frac{U(z)}{E(z)} = KD_d(z), \tag{W6.4}$$

the discrete control equation that results is

$$u(k+1) = 0.6u(k) + 10(4.2e(k+1) - 3.8e(k)). \tag{W6.5}$$

3. The Simulink block diagram of the continuous and discrete versions of $D_c(s)$ controlling the DC motor is shown in Fig. W6.4. The step

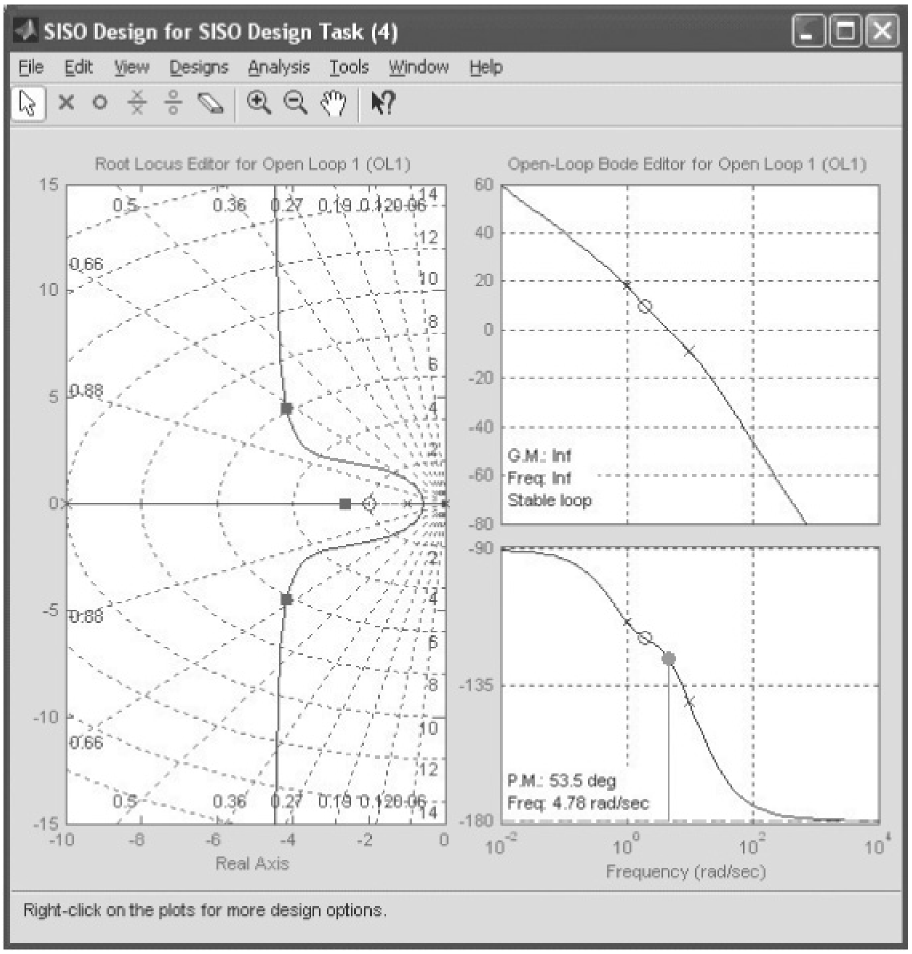

SISO Design for SISO Design Task (4)
File Edit View Designs Analysis Tools Window Help
Root Locus Editor for Open Loop 1 (OL1)
Real Axis
Open-Loop Bode Editor for Open Loop 1 (OL1)
G.M.: Inf
Freq: Inf
Stable loop
P.M.: 53.5 deg
Freq: 4.78 rad/sec
Frequency (rad/sec)
Right-click on the plots for more design options.

Figure W6.3
SISOTOOL graphical interface for Example W6.1

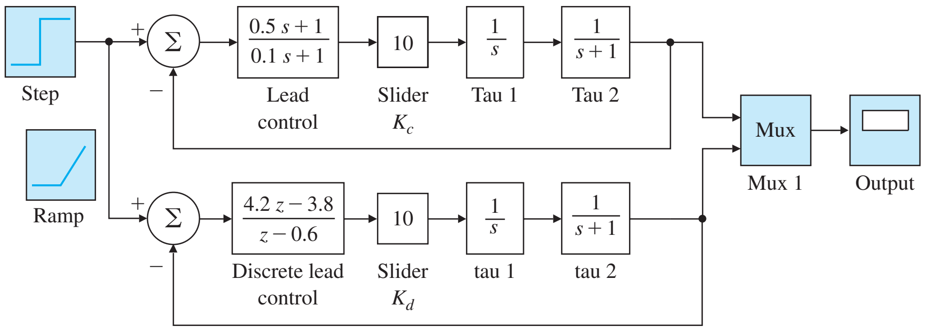


Figure W6.4
Simulink block diagram for transient response of lead-compensation design

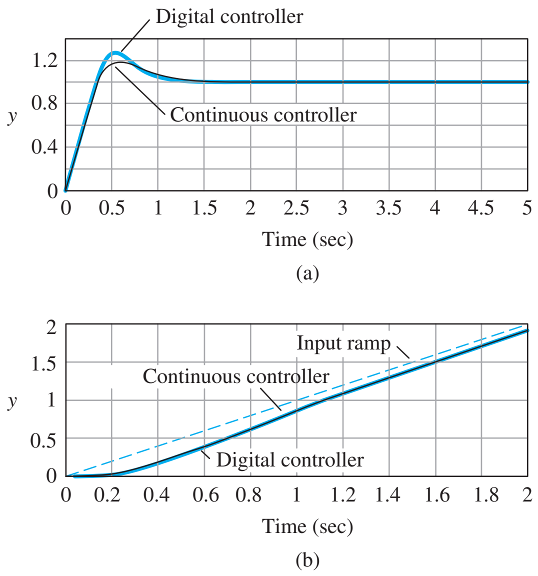


Figure W6.5
Lead-compensation design: (a) step response; (b) ramp response

responses of the two controllers are plotted together in Fig. W6.5a and are reasonably close to one another; however, the discrete controller does exhibit slightly increased overshoot, as is often the case. Both overshoots are less than 25%, and thus meet the specifications. The ramp responses of the two controllers, shown in Fig. W6.5b, are essentially identical, and both meet the 0.1 specified error.

Appendix W7.8
Digital Implementation of Example 7.31

EXAMPLE W7.1 *Redesign of the DC Servo Compensator*

For Example 7.31, derive an equivalent discrete controller with a sampling period of $T_s = 0.1$ sec (10 times the fastest pole), and compare the continuous and digital control outputs and control efforts. Verify the design by plotting the step response and commenting on the comparison of the continuous and discrete responses.

Solution. The discrete equivalent for the controller is obtained from Matlab with the c2d command, as in

```
nc=94.5*conv([1 7.98],[1 2.52]); % form controller numerator
dc=conv([1 8.56 59.5348],[1 10.6]); % form controller denominator
sysDc=tf(nc,dc); % form controller system description
ts=0.1;% sampling time of 0.1 sec
sysDd=c2d(sysDc,ts,'zoh'); % convert controller to discrete time
```

Discrete controller

The resulting controller has the discrete transfer function

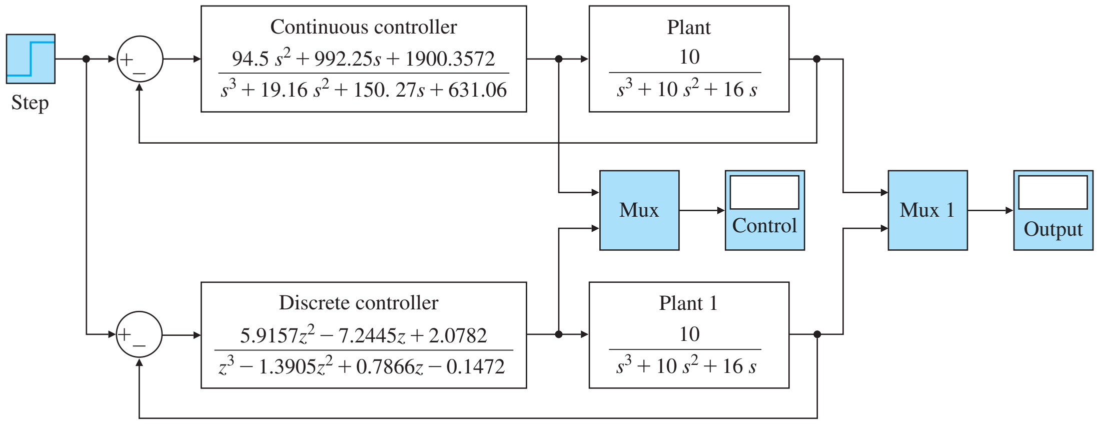
$$D_d(z) = \frac{5.9157(z+0.766)(z+0.4586)}{(z-0.522 \pm 0.3903j)(z+0.3465)}.$$

The equation for the control law (with the sample period suppressed for clarity) is

$$u(k+1) = 1.3905u(k) - 0.7866u(k-1) + 0.1472u(k-2)$$
$$+ e(k) - 7.2445e(k-2) + 2.0782e(k-2).$$

Simulink simulation

A Simulink diagram for simulating both the continuous and discrete systems is shown in Fig. W7.1. A comparison of the continuous and discrete step responses and control signals is shown in Fig. W7.2. Better agreement between the two responses can be obtained if the sampling period is reduced.



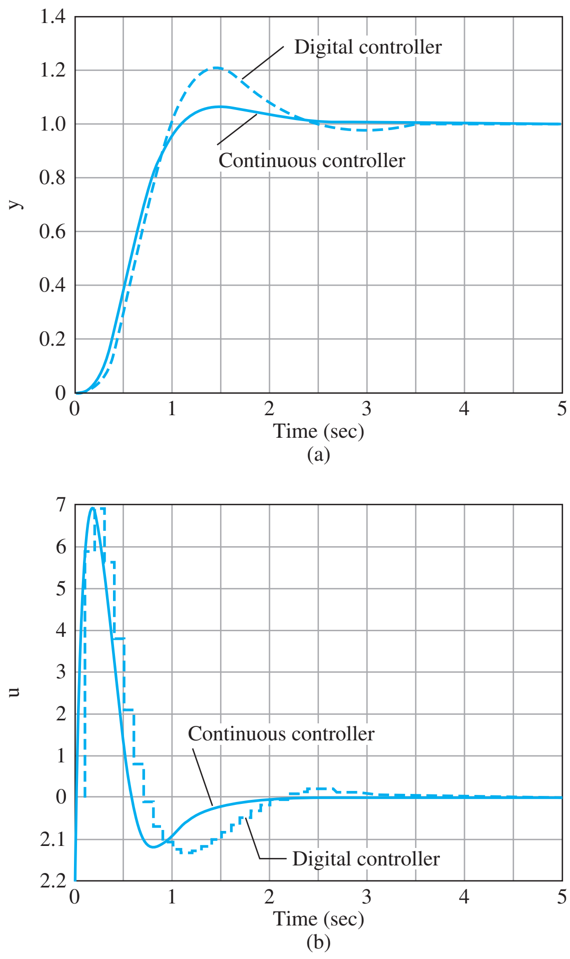
Step
Continuous controller
$\frac{94.5 s^2 + 992.25s + 1900.3572}{s^3 + 19.16 s^2 + 150.27s + 631.06}$
Plant
$\frac{10}{s^3 + 10 s^2 + 16 s}$
Mux
Control
Mux 1
Output
Discrete controller
$\frac{5.9157z^2 - 7.2445z + 2.0782}{z^3 - 1.3905z^2 + 0.7866z - 0.1472}$
Plant 1
$\frac{10}{s^3 + 10 s^2 + 16 s}$

Figure W7.1

Simulink block diagram to compare continuous and discrete controllers

Figure W7.2

Comparison of step responses and control signals for continuous and discrete controllers: (a) step responses; (b)control signals



Appendix W7.9 Digital Implementation of Example 7.33

EXAMPLE W7.2 *Servomechanism*

For Example 7.33, derive an equivalent discrete controller with a sampling period of $T_s = 0.1$ sec ($20 \times \omega_n = 20 \times 0.05 = 0.1$ sec), and compare the continuous and digital control outputs, as well as the control efforts. Verify the design by plotting the step response and commenting on the comparison of the continuous and discrete responses.

Matlab c2d

Solution. The discrete equivalent for the controller is obtained from Matlab by using the c2d command, as in

```
nc=conv([1 1],[8.32 0.8]); % controller numerator
dc=conv([1 4.08],[1 0.0196]); % controller denominator
sysDc=tf(nc,dc); % form controller system description
ts=0.1; % sampling time of 0.1 sec
sysDd=c2d(sysDc,ts,'zoh'); % convert to discrete time controller
```

The discrete controller has the discrete transfer function

$$D_d(z) = \frac{8.32z^2 - 15.8855z + 7.5721}{z^2 - 1.6630z + 0.6637} = \frac{8.32(z - 0.9903)(z - 0.9191)}{(z - 0.998)(z - 0.6665)}.$$

The equation for the control law (with sample period suppressed for clarity) is

$$\begin{aligned} u(k+1) = {} & 1.6630u(k) + 0.6637u(k-1) + 8.32e(k+1) \\ & - 15.8855e(k) + 7.5721e(k-1). \end{aligned}$$

Simulink simulation

A Simulink diagram for simulating both the continuous and discrete systems is shown in Fig. W7.3. A comparison of the continuous and discrete step responses and control signals is shown in Fig. W7.4. Better agreement between the two responses can be achieved if the sampling period is reduced.

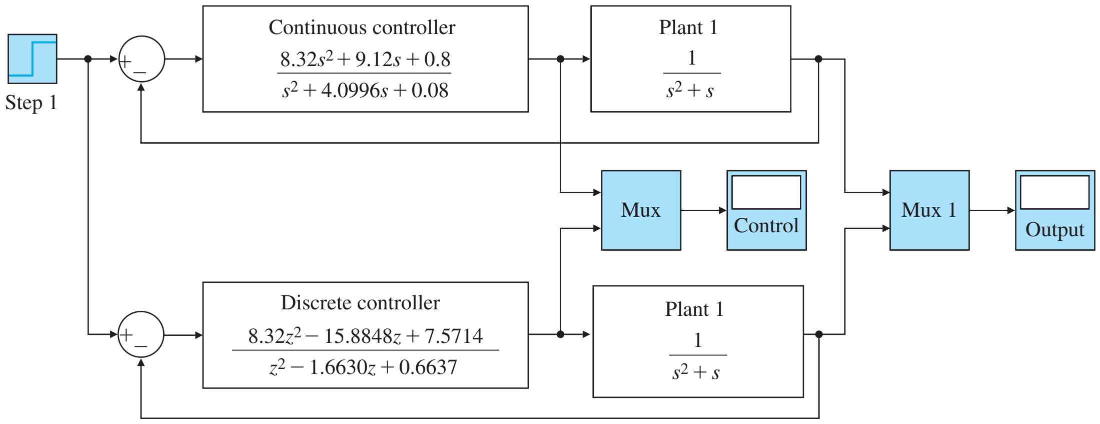
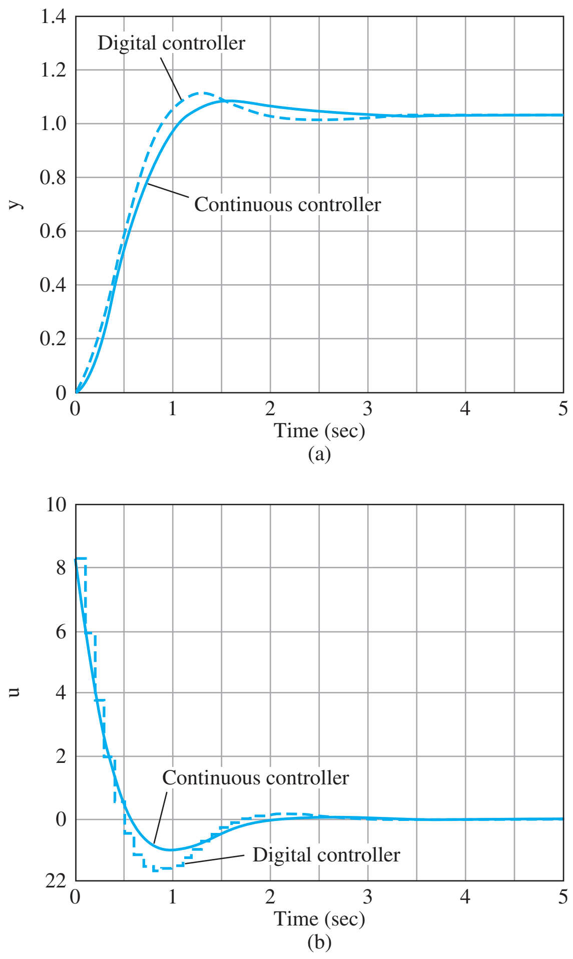


Figure W7.3
Simulink block diagram to compare continuous and discrete controllers

Figure W7.4
Comparison of step responses and control signals for continuous and discrete controllers: (a) step responses; (b) control signals


Digital controller
Continuous controller
y
Time (sec)
(a)
Continuous controller
Digital controller
u
Time (sec)
(b)