Introduction to Applied Digital Control

Second Edition

Gregory P. Starr
Department of Mechanical Engineering
The University of New Mexico

November 2006

Preface

This book is intended to give the senior or beginning graduate student in mechanical engineering an introduction to digital control of mechanical systems with an emphasis on applications. The desire to write this book arose from my frustration with the existing texts on digital control, which—while they were exhaustive—were better suited to reference needs than for tutorial use.

While the appearance of several new texts with a more student-oriented perspective has reduced my frustration a little, I still feel there is a need for an introductory text such as this.

MATLAB has become an almost indispensable tool in the real-world analysis and design of control systems, and this text includes many MATLAB scripts and examples.

My thanks go to my wife Anne, and four boys Paul, Keith, Mark, and Jeff for being patient during the preparation and revision of this manuscript—also my late father Duke Starr who recognized that any kid who liked mathematics and motorcycles can't be all bad.

Albuquerque, New Mexico
November 2006

Contents

List of Figures

Chapter 1

Introduction and Scope of this Book

This book is intended to give the student an introduction to the field of digital control, with an emphasis on applications. Both transform-based and state-variable approaches will be included, with a brief introduction to system identification.

The material requires some understanding of the Laplace transform and assumes that the reader has studied linear feedback control systems.

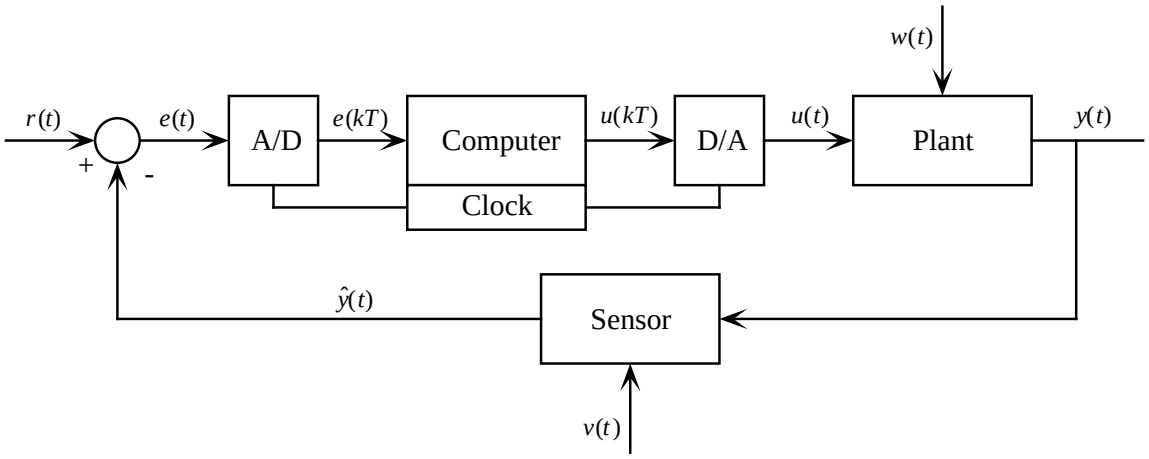
1.1 Continuous and Digital Control

1.1.1 Feedback control

The study of feedback control systems is concerned with using a measurement of the output of a plant (device to be controlled) to modify its input. The *controller* is that part of the system that receives the measurement of the plant output, then generates the plant input, hence closing the loop. Control system design is the task of designing this *controller* such that the *closed-loop system* has satisfactory performance. Broadly speaking, some goals of most closed-loop control systems are:

- *Command Tracking* ... cause the output to track the reference input closely
- *Disturbance Rejection* ... isolate the output from unwanted disturbance inputs
- *Parameter Sensitivity* ... reduce the effect on the output of variations in plant parameters

These are goals of both continuous and digital control systems.



r = reference input or setpoint
u = control force (actuator input)
y = controlled variable or output
$\hat{y}$ = measurement of controlled variable
e = $r - \hat{y}$ = error signal
w = disturbance acting on the plant
v = measurement noise
A/D = analog-to-digital converter
D/A = digital-to-analog converter

Figure 1.1: Basic digital control system.

1.1.2 Digital control

Digital control systems employ a computer as a fundamental component in the controller. The computer typically receives a measurement of the controlled variable, also often receives the reference input, and produces its output using an algorithm. This output is usually converted to an analog signal using a D/A converter, then amplified by a power amplifier to drive the plant. A block diagram of a typical digital control system is shown in Figure 1.1.

When compared to a continuous-time system, there are three new elements in the block diagram of Figure 1.1:

- *A/D converter.* This device acts on a continuous physical variable, typically a voltage, and converts it into an integer number. A/D converters typically have *unipolar* ranges of 0–5 V, 0–10 V, or *bipolar* ranges of ± 5 V, or ± 10 V. These are often jumper-selectable. The A/D conversion causes quantization q, given by the resolution of the converter in bits. Common resolutions are 8 bits (256 levels), and 12 bits (4096 levels). A 12-bit A/D converter of range ±10 volts would have a conversion quantum of $q = 20/4096 = 4.88$ mV. Note that quantization

is a nonlinear operation. The effect of quantization on a continuous signal is often called *quantization noise*.

- *D/A converter*. This device converts an (integer) number to a voltage. The voltage ranges and converter resolutions are the same as for the A/D converter. A D/A converter functions as a *zero-order hold*, holding its output at a constant value until it receives the next discrete input.

- *Sampling*. This is represented by the clock in Figure 1.1. The computer samples the error (or it may sample both the setpoint and the measurement, thus forming the error internally) at particular times. In this book we will assume that sampling is at a constant period T, which is called the *sampling period*. The sampling frequency in Hz is $1/T$. When a continuous signal $e(t)$ has been sampled, it is called a *discrete* signal and is denoted by $e(kT)$ or $e(k)$ or e_k. Discrete signals are only mathematically defined at the sample instants $t_k = kT$.

A system in which there are only discrete signals is called a *discrete-time system*. Systems with both continuous and discrete variables are called *sampled-data systems.* When quantization is added, the system may be called *digital*. Continuous, discrete, and "zero-order hold" (output of D/A) signals are shown in Figure 1.2, where the signal is $sin(t)$.

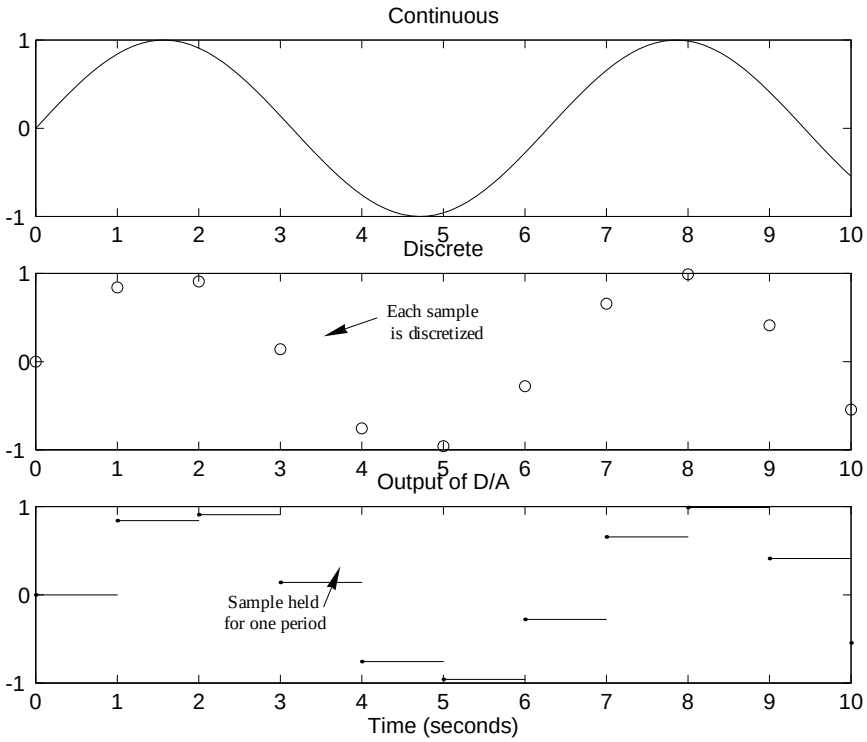


Figure 1.2: Three types of signals.

Much of the task of designing a digital controller is in accounting for the effects of quantization and sampling, especially sampling. If both T and q are small, digital signals approach continuous signals, and continuous methods of analysis and design may be used. However, when this is not the case, continuous methods can lead to erroneous results.

1.2 Philosophy and Text Coverage

The philosophy of this course is to present the basic material necessary for the analysis and design of digital control systems. We assume a background in continuous systems, and relate the digital problem to its continuous counterpart. The emphasis is on understanding the physical reality behind the analysis.

The eight chapters in this book contain the following material:

Chapter 1. This chapter. Describes philosophy and content of book. Some definitions. Rationale for studying digital control.

Chapter 2. Sampled (discrete-time) variables. Introduction of the z-transform for discrete variables, which is analogous to the Laplace transform for continuous variables.

Chapter 3. Discrete simulations of continuous systems. Several methods of simulation are given, specifically numerical integration, pole-zero mapping, and zero-order hold equivalence.

Chapter 4. Frequency spectra of sampled signals. The impulse modulation model of sampling. Aliasing and its effects.

Chapter 5. Transform-based design of digital control systems, primarily using the root locus method. When sampling can be ignored, and when it must be considered.

Chapter 6. State-space modeling and analysis of continuous systems, including nonlinear systems.

Chapter 7. State-space design of digital control systems, primarily using pole placement. Control law design and state estimation. Introduction of the reference input.

Chapter 8. System identification using the least squares method.

It is the author's observation that digital control systems are so widely used that it is rare to see a completely continuous control system. There are several reasons for this:

- Computers are getting faster, cheaper, and more reliable.
- Control systems incorporating computers are inherently more flexible than those without, *e.g.* during the prototyping phase, tuning gains to achieve satisfactory performance is simply a matter of changing numbers in a computer program, rather than changing hardware.
- Advanced control techniques such as *optimal* and *adaptive* control can only be realized digitally.

- Computers are often already present in mechanical systems for communication, visualization, *etc.*, thus their use for control is logical. The reverse is also true—if a computer is used for control, it can *also* address many other functions which may be needed in a system.

These days, for anyone with a desire to design and construct working control systems, at least an introductory course in digital control (like this one) is absolutely necessary.

Homework Problems

1. Go back to your continuous control textbook and review the following concepts:

   - Feedback Control
   - Laplace Transform
   - Transfer Functions
   - Block Diagrams

2. Cite examples of physical variables that are:

   - Continuous
   - Discrete

Chapter 2

Linear Discrete Systems and the Z-Transform

The primary new component of discrete (or digital, we won't treat the effects of quantization) systems is the notion of time discretization. No longer are we dealing with variables which are functions of time, now we have sequences of discrete numbers. These discrete numbers *may* come from sampling a continuous variable, or they may be generated within a computer. In either case, the tools that were used in the analysis of continuous variables will no longer work. We need new methods.

The z-transform bears exactly the same relationship to a discrete variable that the Laplace transform bears to a continuous variable. This is the new tool we need, and the whole of transform-based digital control system design turns on the z-transform.

2.1 Chapter Overview

In Section 2.2 linear difference equations, the discrete counterpart of linear differential equations, will be introduced. Through solutions of difference equations we will get insight into discrete pole locations and stability. Section 2.3 will present the z-transform, which operates on discrete variables like $y(k)$ or y_k to produce functions of z, $Y(z)$. The z-transform will lead to the discrete transfer function, which can be represented with block diagrams composed of sums, gains, and unit delays. The dynamic response of discrete systems will be presented in Section 2.4, where we will examine the step, exponential, and damped sinusoidal functions. The correspondence between discrete signals and the continuous signals from which they were obtained will be investigated in Section 2.5, where we will derive a fundamental mapping linking the s and z planes. A method for obtaining the frequency response of discrete systems will be briefly shown in Section 2.6. Some properties of the z-transform will be discussed in Section 2.7, including several techniques for inverse transforming to obtain $f(kT)$ or $f(k)$ from $F(z)$. A brief table of z-transforms appears in Section 2.9.

2.2 Linear Difference Equations

Physical systems are modeled by (continuous) differential equations. The order of the system is the order of the corresponding differential equation. Discrete systems, of course, cannot be modeled by differential equations, but are instead represented by *difference* equations.

Consider the block diagram of Figure 2.1, where an A/D converter samples a continuous variable $e(t)$ to produce discrete variable $e(kT)$, then a computer processes these $e(kT)$ to produce discrete output $u(kT)$.

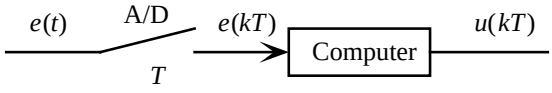


Figure 2.1: A/D converter and computer

To generate the k^{th} output sample $u(kT)$ or u_k, the computer can make use of the following inputs and (past) outputs:

$$\begin{aligned} \text{inputs} &: e_0, e_1, e_2 \ldots, e_k \\ \text{outputs} &: u_0, u_1, u_2 \ldots, u_{k-1} \end{aligned}$$

This discrete relationship can be expressed in the form

$$u(k) = f(e_0, e_1, e_2 \ldots, e_k; u_0, u_1, u_2 \ldots, u_{k-1}) \tag{2.1}$$

If the function f is linear, the relationship becomes a *linear difference equation* given by

$$u_k = a_1 u_{k-1} + a_2 u_{k-2} + \ldots + a_n u_{k-n} + b_0 e_k + b_1 e_{k-1} + b_2 e_{k-2} + \ldots + b_m e_{k-m} \tag{2.2}$$

If the initial conditions and input are known, a difference equation can be simulated by simply evaluating the equation.

Example

Consider the difference equation given by

$$u_k = u_{k-1} + u_{k-2} \tag{2.3}$$

with initial conditions $u_0 = 1$ and $u_1 = 1$. This equation computes the sequence known as Fibonacci[1] numbers. Note that this difference equation has no input... don't worry about that for now. Table 2.1 can then be constructed (assume all variables are zero for $k < 0$). Note that from the response shown in Table 2.1 you would probably say that this system is unstable.

[1] Leonardo Fibonacci of Pisa, who introduced Arabic notation to the Latin world about 1200 A.D.

Sample index k	u_k
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
⋮	⋮

Table 2.1: Behavior of difference equation

2.2.1 Solving Difference Equations

Although any difference equation with a given input can be "solved" in this manner, we need some way to predict the behavior, a way to represent difference equations and discrete systems that is generally useful.

Consider first the solution of a difference equation. With linear differential equations, we often *assume* a solution, then see if it works[2]. For linear differential equations in time we often assume a solution of the form

$$u(t) = Ae^{st} \tag{2.4}$$

where s is a complex variable. Substitution of this into the differential equation will yield values of s for which the solution is valid. The constant A will be determined by initial conditions.

A similar approach can be used with difference equations. Try a solution of the form

$$u(k) = Az^k \tag{2.5}$$

where z is a complex variable and of course k is the sample index. Substitition into 2.3 yields

$$Az^k = Az^{k-1} + Az^{k-2} \tag{2.6}$$

or

$$1 = z^{-1} + z^{-2} \tag{2.7}$$

or

$$z^2 - z - 1 = 0 \tag{2.8}$$

Note that equation 2.8 is the *characteristic equation* for this system, for which the two roots z_1 and z_2 are

$$z_1 = -0.618, z_2 = 1.618^3 \tag{2.9}$$

[2]The guy I'd like to meet is the guy who first figured out what to assume!

[3]This is the value of the *Golden Ratio*; see C. Moler, Numerical Computing with MATLAB.

The general solution is therefore

$$u(k) = A_1 z_1^k + A_2 z_2^k \tag{2.10}$$

where A_1 and A_2 may be found from initial conditions.

Of note here is the behavior of the two "modes" in equation 2.10. The mode associated with z_1 will decay, but the mode associated with z_2 will grow. Clearly if a root z of the characteristic equation has $|z| > 1$ that term will increase.

Since z is a complex variable, we can speak of the z-plane, a complex number plane. An observation on stability that will be of fundamental importance is

> If any roots of the characteristic equation of a discrete system have magnitude > 1, *e.g.* are outside the *unit circle* of the z-plane, that system will be unstable.

This is our first observation on the correlation between z-plane root location and discrete system dynamic response.

Example

Consider the numerical integration of a continuous time variable, as shown graphically in Figure 2.2

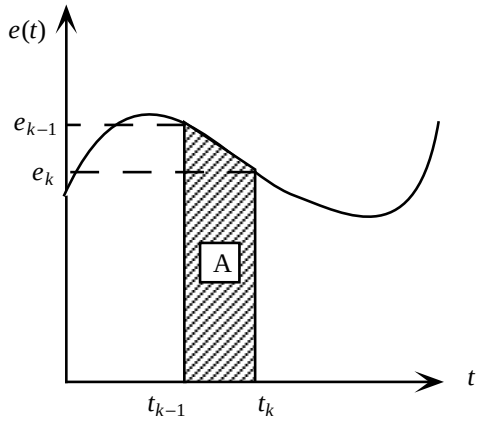


Figure 2.2: Numerical integration.

We want the integral of the function $e(t)$ from $t = 0$ to t as given by

$$\mathcal{I} = \int_0^t e(t)dt \tag{2.11}$$

using only samples $e_0, e_1, \ldots, e_{k-1}, e_k$. We assume that the integral from $t = 0$ to $t = t_{k-1}$ is known, and is u_{k-1}. Thus we just want a procedure to take the "next step."

Although there a numerous methods, here we use trapezoidal integration, in which we approximate the integral by computing the area A of the trapezoid in Figure 2.2. Thus

$$A = \frac{t_k - t_{k-1}}{2}(e_k + e_{k-1}) \tag{2.12}$$

Assume constant stepsize, so $t_k - t_{k-1} = T$, thus

$$u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k-1}) \tag{2.13}$$

Equation 2.13 is a linear difference equation for trapezoidal integration. We will be using this equation in Chapter 3.

2.3 The Z-Transform and the Discrete Transfer Function

First define the z-transform, then use it to find a discrete transfer function.

2.3.1 The z-Transform

Given a discrete variable $e(k)$ or e_k with values $e_0, e_1, \ldots, e_k, \ldots$, the z-transform of this variable is given by

$$E(z) = \mathcal{Z}\{e_k\} = \sum_{k=0}^{\infty} e_k z^{-k} \tag{2.14}$$

The z-transform has the same role in the analysis and design of discrete systems as the Laplace transform has in continuous systems. The transform given in equation 2.14 is the *single-sided* version (summation index from 0 to ∞ rather than from $-\infty$ to ∞, but this causes no loss of generality.

2.3.2 Discrete Transfer Function

Discrete systems can be modeled with transfer functions, just like linear continuous systems. Recall that for continuous systems the transfer function represents the Laplace transform of the output $Y(s)$ over the Laplace transform of the input $U(s)$, and is a ratio of polynomials $b(s)$ and $a(s)$, thus

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\text{polynomial } b(s)}{\text{polynomial } a(s)} \tag{2.15}$$

The order of $b(s)$ must not be greater than the order of $a(s)$ or the system will be non-causal.

Let's apply the z-transform to the difference equation for trapezoidal integration, equation 2.13. We can do this by multiplying equation 2.13 by z^{-k} and summing from 0 to ∞...

$$\sum_{k=0}^{\infty} u_k z^{-k} = \sum_{k=0}^{\infty} u_{k-1} z^{-k} + \frac{T}{2}\left(\sum_{k=0}^{\infty} e_k z^{-k} + \sum_{k=0}^{\infty} e_{k-1} z^{-k}\right) \tag{2.16}$$

For the two terms with the $k-1$ subscript, let $k-1=j$ and write them in the form

$$\sum_{k=0}^{\infty} u_{k-1} z^{-k} = z^{-1} \sum_{j=-1}^{\infty} u_j z^{-j}. \tag{2.17}$$

Since with the single-sided transform all variables are zero for negative sample index, we can change the lower limit from $j=-1$ to $j=0$ and we have

$$\begin{aligned} \sum_{k=0}^{\infty} u_{k-1} z^{-k} &= z^{-1} \sum_{j=0}^{\infty} u_j z^{-j} \\ &= z^{-1} U(z). \end{aligned} \tag{2.18}$$

Now equation 2.16 can be written

$$U(z) = z^{-1} U(z) + \frac{T}{2}\left[E(z) + z^{-1} E(z)\right] \tag{2.19}$$

which may be rearranged to yield discrete transfer function

$$\frac{U(z)}{E(z)} = \frac{T}{2}\frac{z+1}{z-1} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}}. \tag{2.20}$$

Note that the transfer function of equation 2.20 can be expressed using either positive or negative powers of variable z. Each form is preferable for certain uses, as will be seen later.

General Form of Discrete Transfer Function

In general, a discrete transfer function $H(z)$ relating input $E(z)$ and output $U(z)$ will be

$$\begin{aligned} H(z) &= \frac{b_0 z^n + b_1 z^{n-1} + b_2 z^{n-2} + \cdots + b_m z^{n-m}}{z^n - a_1 z^{n-1} - a_2 z^{n-2} - \cdots - a_n} \\ &= \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_n z^{-n}} = \frac{U(z)}{E(z)} \\ &= \frac{b(z)}{a(z)} \quad \text{a ratio of polynomials in } z, \end{aligned} \tag{2.21}$$

where the "1" in the denominator of the negative-power form of $H(z)$ is required (this will be shown later).

Again, in equation 2.21 either the negative or positive power of z form can be used. Given that the transfer function is the ratio of output or input z transforms, we can rearrange to get

$$U(z) = H(z)E(z), \tag{2.22}$$

thus just like for continuous systems, the (z-transformed) output is given by the transfer function times the (z-transformed) input.

Poles and Zeros of Discrete Transfer Function

Since z is a complex variable (like s), so is $H(z)$, and we can define the *poles* and *zeros* of $H(z)$ as

- poles: locations in z-plane where polynomial $a(z) = 0$
- zeros: locations in z-plane where polynomial $b(z) = 0$

For finding poles and zeros it is easier to use the version of $H(z)$ with positive powers of z. The poles and zeros may be real or complex. If complex, they occur in conjugate pairs.

The Unit Delay

If we let $b_1 = 1$ and all other $b_n = 0$; also let all $a_n = 0$, then the transfer function of equation 2.21 degenerates to

$$\frac{U(z)}{E(z)} = z^{-1} = H(z) \tag{2.23}$$

Doing the same thing to the general difference equation of 2.2 yields

$$u_k = e_{k-1} \tag{2.24}$$

which says the present value of the output equals the input delayed by one sample, or the previous input.

Thus a transfer function of z^{-1} is a delay of one sample period, or a *unit delay*. These may be placed in series to effect a delay of multiple samples.

2.3.3 Block Diagrams of Discrete Systems

All linear difference equations are composed of delays, multiplies, and adds, and we can represent these operations in block diagrams. A block diagram will often be helpful in system visualization.

Example

Consider the difference equation for trapezoidal integration,

$$u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k-1}) \tag{2.25}$$

This difference equation is represented by the block diagram shown in Figure 2.3.

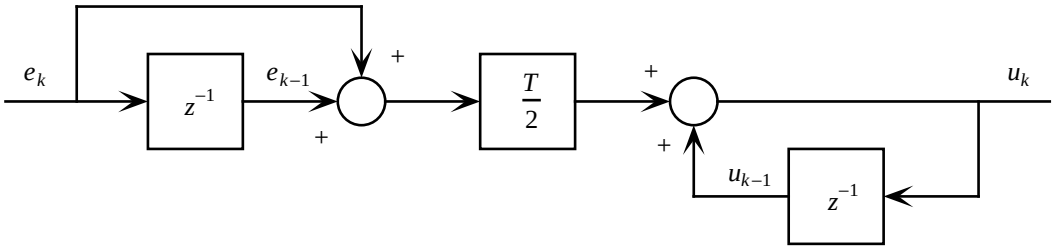


Figure 2.3: Block diagram of trapezoidal integration.

Figure 2.3 can be reduced by standard block diagram reduction techniques and the transfer function of 2.20 will result. This is left as an exercise for the student.

2.3.4 Going From Transfer Function to Difference Equation

It is useful to be able to easily convert a discrete system from its transfer function form to its difference equation. If we take the general discrete transfer function of equation 2.21 given by

$$H(z) = \frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_n z^{-n}} = \frac{b(z)}{a(z)} \text{ polynomials} \tag{2.26}$$

and cross-multiply, we get

$$U(z) - a_1 z^{-1} U(z) - a_2 z^{-2} U(z) - \cdots = b_0 E(z) + b_1 z^{-1} E(z) + b_2 z^{-2} E(z) + \cdots \tag{2.27}$$

We know that $U(z)$ may be inverse-transformed to yield sequence u_k, likewise for $E(z)$. And multiplication by z^{-k} simply delays by k samples. Hence by inspection the difference equation corresponding to 2.27 is given by

$$u_k - a_1 u_{k-1} - a_2 u_{k-2} - \cdots = b_0 e_k + b_1 e_{k-1} + b_2 e_{k-2} + \cdots \tag{2.28}$$

or

$$u_k = a_1 u_{k-1} + a_2 u_{k-2} + \cdots + b_0 e_k + b_1 e_{k-1} + b_2 e_{k-2} + \cdots \tag{2.29}$$

Retracing the development, if the denominator polynomial $a(z)$ in 2.21 did not contain the "1" term, the output term u_k would not be present in equation 2.29, *i.e.* there would be no output!

2.3.5 Relation of the Transfer Function to the Unit Pulse Response

Recall that in continuous systems the transfer function of a thing is also equal to the Laplace transform of the thing's unit impulse response (easily shown... it might be good to verify it). A similar relationship holds for discrete systems. Consider a system with transfer function

$$H(z) = \frac{U(z)}{E(z)}$$

Let input e_k be a unit pulse, *i.e.*

$$\begin{aligned} e_k &= 1 \qquad k = 0 \\ &= 0 \qquad k \neq 0 \\ &= \delta_k \qquad \text{(unit discrete pulse at sample zero)} \end{aligned} \tag{2.30}$$

The z-transform of the unit pulse is given by

$$E(z) = \sum_{k=0}^{\infty} e_k z^{-k} = e_0 z^0 = 1 \tag{2.31}$$

hence $U(z) = H(z)$, and the discrete transfer function $H(z)$ is the z-transform of the system's unit pulse response.

Example

Let's see if this works on the trapezoidal integrator. The difference equation is

$$u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k-1}) \tag{2.32}$$

and the response to a unit pulse is shown in the accompanying table.

k	e_k	u_k
0	1	$T/2$
1	0	T
2	0	T
3	0	T
$\vdots$	$\vdots$	$\vdots$

Now, going on faith that $H(z) = U(z)$ for a unit pulse input, form this term as follows:

$$\begin{aligned} H(z) = U(z) &= \sum_{k=1}^{\infty} Tz^{-k} + \frac{T}{2} \\ &= \sum_{k=0}^{\infty} Tz^{-k} - \frac{T}{2} \end{aligned} \tag{2.33}$$

To reduce the first term on the right of the "=" sign in 2.33 one must use the expression for the sum of a geometric series. This sum is very useful in expressing z-transforms in closed form.

The sum of a geometric series is given by

$$\boxed{\sum_{k=0}^{\infty} ar^k = \frac{a}{1-r}} \tag{2.34}$$

Using equation 2.34 to reduce 2.33 (the "r" term will be z^{-1}) we obtain

$$\begin{aligned} H(z) = U(z) &= \frac{T}{1-z^{-1}} - \frac{T}{2} = \frac{2T - T(1-z^{-1})}{2(1-z^{-1})} \\ &= \frac{T + Tz^{-1}}{2(1-z^{-1})} = \frac{T}{2}\frac{1+z^{-1}}{1-z^{-1}} = \frac{T}{2}\frac{z+1}{z-1} \end{aligned} \tag{2.35}$$

which agrees with the transfer function of equation 2.20.

2.4 Dynamic Response of Discrete Systems

In this section we will examine the dynamic response of discrete systems and begin to associate z-plane pole and zero location with corresponding response. As a designer, it is important to have a good "feel" for the correlation between z-plane pole and zero location and system response. We have already seen the most basic correlation, that of stability. If any poles of the discrete transfer function are outside the unit circle, the system is unstable.

Indeed, one can use the "brute force" technique, given by

1. Obtain the transfer function $H(z)$ of the system.
2. Look up the z-transform of the input $E(z)$ (a table of z transforms will appear at the end of this chapter)
3. Form output $U(z) = H(z)E(z)$
4. Inverse transform $U(z)$ to get u_k.

This can be an involved process, especially inverse transforming. So it is worthwhile to develop a good correlation between pole and zero location and dynamic response. Presumably you have some knowledge of this correlation for continuous systems in the s-plane... if not, it would a good idea to review that now.

Our approach here will be to take three basic discrete functions and examine both their z-transforms (pole locations) and their discrete-domain character. These discrete functions will be

1. Unit step.
2. Exponential.
3. Damped sinusoid.

In the three sections to follow, a numerical subscript will be used to indicate these functions, like $e_1(k), e_2(k)$, and $e_3(k)$.

2.4.1 Unit step

The unit step function is given by $e_1(k) = 1, k \geq 0$, with z-transform

$$E_1(z) = \sum_{k=0}^{\infty} z^{-k} = \frac{1}{1 - z^{-1}} = \frac{z}{z - 1}. \tag{2.36}$$

The unit step has a pole at $z = 1$ and a zero at $z = 0$. A sketch of the pole and zero locations for the unit step and the corresponding discrete behavior are shown in Figure 2.4.

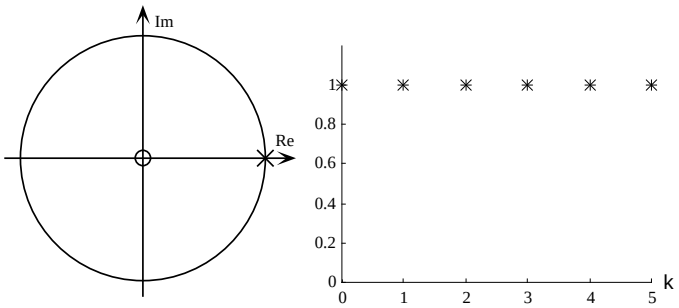


Figure 2.4: Unit step pole/zero locations and response.

2.4.2 Exponential

The discrete exponential function is given by $e_2(k) = r^k, k \geq 0$. The z-transform of this function is

$$E_2(z) = \sum_{k=0}^{\infty} r^k z^{-k} = \sum_{k=0}^{\infty} \left(rz^{-1}\right)^k = \frac{1}{1 - rz^{-1}} = \frac{z}{z - r} \tag{2.37}$$

which has a single real pole at $z = r$, plus a zero at $z = 0$. There are a couple of observations about the behavior of the exponential function.

- Since the exponential function is $e_2(k) = r^k$, if $|r| > 1$ the response will grow. This corresponds to a pole outside the unit circle. If this were the pulse response of a discrete system the system would be unstable. (We saw this in Section 2.2.1).
- If $-1 < r < 0$, the pole is on the negative real axis (but within the unit circle) the response will alternate signs. This form is unique to discrete systems and has no counterpart in the continuous domain, as we will see later.

Three examples of pole and zero locations for the exponential function and the corresponding discrete behavior are shown in Figure 2.5.

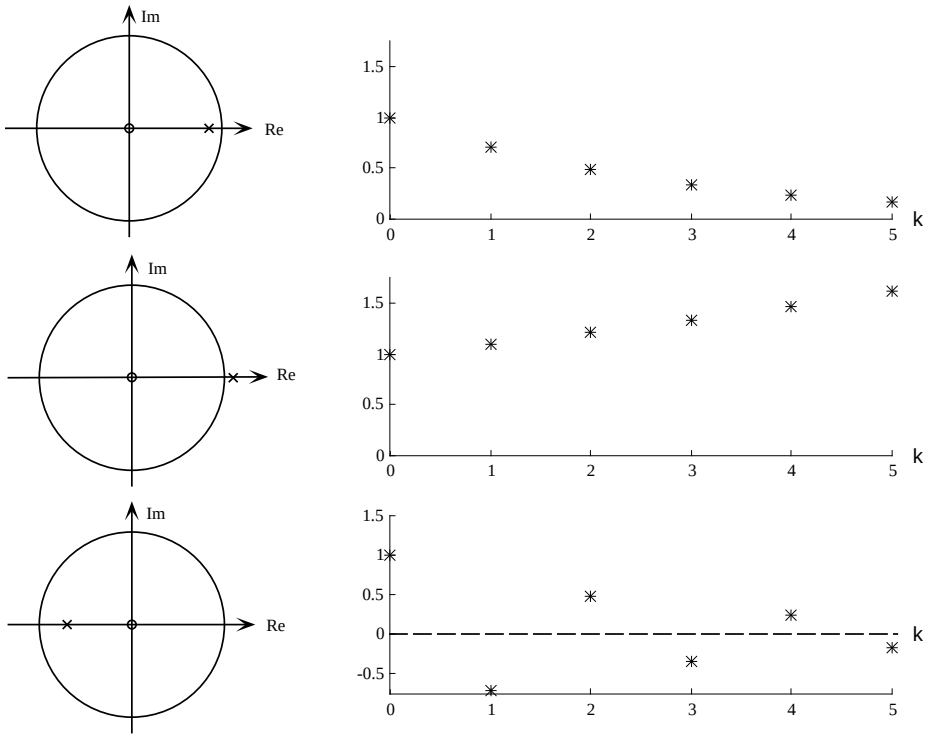


Figure 2.5: Exponential pole/zero locations and response.

2.4.3 Damped Sinusoid

The discrete damped sinusoid is given by $e_3(k) = r^k \cos(k\theta), k \geq 0$. Using an Euler expansion of the cosine, we obtain

$$e_3(k) = r^k \left(\frac{e^{jk\theta} + e^{-jk\theta}}{2} \right) \tag{2.38}$$

Since the z-transform is linear, we can treat each term of equation 2.38 separately, then add the z-transform of both terms at the end. Consider the first term (denominator 2 factored out):

$$e_4(k) = r^k e^{jk\theta}. \tag{2.39}$$

The z-transform of this discrete function is

$$\begin{aligned} E_4(z) = \sum_{k=0}^{\infty} r^k e^{jk\theta} z^{-k} = \sum_{k=0}^{\infty} \left(re^{j\theta} z^{-1}\right)^k &= \frac{1}{1 - re^{j\theta} z^{-1}} \\ &= \frac{z}{z - re^{j\theta}} \end{aligned} \tag{2.40}$$

This function has a single complex pole at $z = re^{j\theta}$. Note that this function will grow if $|r| > 1$ and will decay if $|r| < 1$. Now for the second term in 2.38, it can be shown that

$$\mathcal{Z}\left\{e_5(k)\right\} = \mathcal{Z}\left\{r^k e^{-jk\theta}\right\} = E_5(z) = \frac{z}{z - re^{-j\theta}}, \tag{2.41}$$

and adding both expressions yields

$$E_3(z) = \left(\frac{E_4(z) + E_5(z)}{2}\right) = \frac{z(z - r\cos\theta)}{z^2 - 2r\cos\theta z + r^2} \tag{2.42}$$

The poles of $E_3(z)$ are complex conjugates, and are

$$\begin{aligned} z &= re^{\pm j\theta} \\ &= r\cos\theta \pm jr\sin\theta \end{aligned} \tag{2.43}$$

The zeros are real, at $0, r\cos\theta$ and are in line with the poles.

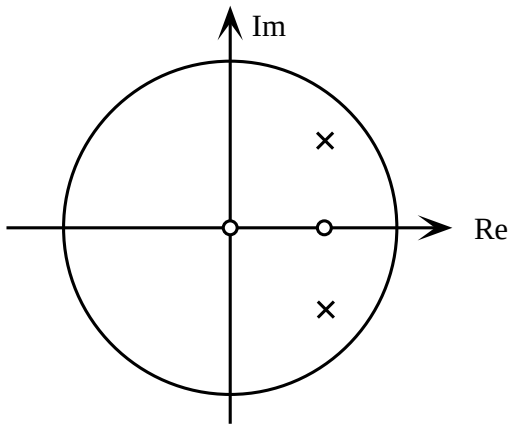
Example

Consider a discrete damped sinusoid with $r = 0.7$ and $\theta = 45° = \pi/4$, thus

$$e_k = e(k) = r^k \cos k\theta = 0.7^k \cos\left(k\frac{\pi}{4}\right) \tag{2.44}$$

With an initial condition of $e_0 = 1$, the behavior of this discrete function is shown in the table below:

k	e_k
0	1.000
1	0.495
2	0.000
3	-0.243
4	-0.240
5	-0.119
6	0.000
7	0.058
8	0.058
9	0.029
10	0.000
⋮	⋮



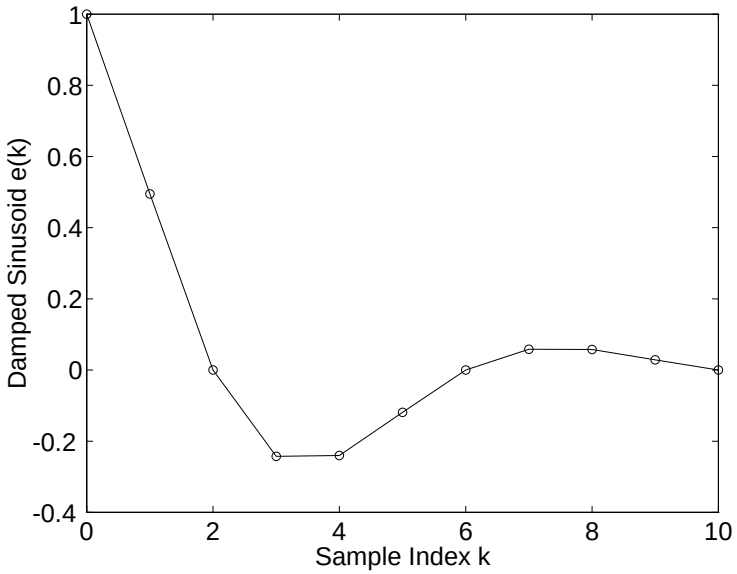


Figure 2.6: Discrete damped sinusoidal response.

The pole/zero locations for this function are shown to the right of the table. A plot of this function is shown in Figure 2.6.

Note that although the samples of Figure 2.6 are connected with straight lines for viewing clarity, in a discrete system nothing truly exists between the samples.

As in continuous systems, higher-order discrete system response is always a combination of first order (exponential) and second order (damped sinusoidal) terms. The notion of *dominant* poles carries over to discrete systems also, with relative proximity to the $z = 1$ point defining dominance. If a system has a single real pole or a complex pole pair much closer to the $z = 1$ point than other poles or zeros, the dynamic response of the system will be dominated by that pole or poles.

2.4.4 Relationship between z-plane poles and transient response

We make the following observations:

1. Rate of decay of exponential response is determined by the radius r of the pole location.
   - If $r > 1$ the sequence increases, larger $r \Rightarrow$ faster growth.
   - If $r = 1$ the sequence doesn't grow, but doesn't decay.
   - If $r < 1$ the sequence decays, the closer r is to zero the faster the decay

   (This is also true for the exponential envelope of damped sinusoids)

2. The oscillation speed (number of samples/cycle) of complex poles is determined by their angle θ. Given

$$e_k = \cos k\theta \tag{2.45}$$

let N be the number of samples per oscillation period, hence

$$\cos k\theta = \cos\left[(k+N)\theta\right] = \cos(k\theta + N\theta). \tag{2.46}$$

Since N is defined to be the period, we must have

$$N\theta = 2\pi \tag{2.47}$$

hence the period N is given by

$$N = \frac{2\pi}{\theta_{rad}} = \frac{360}{\theta_{deg}} \tag{2.48}$$

Example

In the previous example, we had a damped sinusoidal response. The period of this response is given by

$$N = \frac{360}{\theta_{deg}} = \frac{360}{45} = 8 \text{ samples} \tag{2.49}$$

Examination of the response in both the table and figure verifies this period.

2.4.5 Effect of Zeros on Dynamic Response

In continuous transfer functions, zeros represent derivatives of the input which are transferred to the output, thus speeding up response and increasing overshoot.

The same is true of discrete transfer functions, with the effect of a z-plane zero greater when it is near $z = 1$, and its effect decreasing as it is far from $z = 1$. Note that a zero at $z = 0$ represents a unit advance, just as a pole at $z = 0$ is a unit delay.

2.5 Correspondence between Discrete and Continuous Signals

So far in this chapter we have considered discrete systems only. It is time to make a connection between discrete and continuous systems, and in the process to develop a very useful mathematical link between the z domain and the s domain.

Consider discrete variable $y(k)$ (y_k could be used), where we now say that this $y(k)$ was obtained by sampling continuous signal $y(t)$ at sample period T. We shall use a damped sinusoid for this development.

The continuous signal

$$y(t) = e^{-at}\cos bt \tag{2.50}$$

has Laplace transform

$$Y(s) = \frac{s+a}{(s+a)^2+b^2} \tag{2.51}$$

and the poles of $Y(s)$ are at

$$s = -a \pm jb \tag{2.52}$$

Now, if we sample $e^{-at}\cos bt$ at sample period T, at sample times of $t = kT$, we will get discrete signal

$$y(kT) = e^{-akT}\cos bkT \tag{2.53}$$

which is of the form $r^k \cos k\theta$, where r and θ given by

$$r = e^{-aT}, \theta = bT \tag{2.54}$$

We saw from Equation 2.43 that a discrete damped sinusoid has z-plane poles at

$$z = re^{\pm j\theta} \tag{2.55}$$

which here becomes

$$z = re^{\pm j\theta} = e^{-aT}e^{\pm jbT} = e^{(-a\pm jb)T} \tag{2.56}$$

Compare the s-plane pole locations of 2.52 and the z-plane poles of 2.56. The pole locations are related by

$$z = e^{sT} \tag{2.57}$$

This is a general relationship between *poles* of continuous signals and their discrete counterparts, and can be expressed in terms of pole locations:

Poles in the s-plane and z-plane are related by $z = e^{sT}$. Note that this is *not* true of zeros.

This mapping between the s and the z plane is very useful in design.

Example

Let's examine the mapping between some contours in the s plane and z plane, with reference to Figure 2.7.

- s-plane origin. This maps into $z = e^{sT} = 1$, on the unit circle.
- s-plane real axis from $0 \to -\infty$. Using $z = e^{sT}$ this maps into the z real axis from $+1 \to 0$.
- s-plane $j\omega$ axis. This maps into $z = e^{j\omega T}$, which is the unit circle itself.

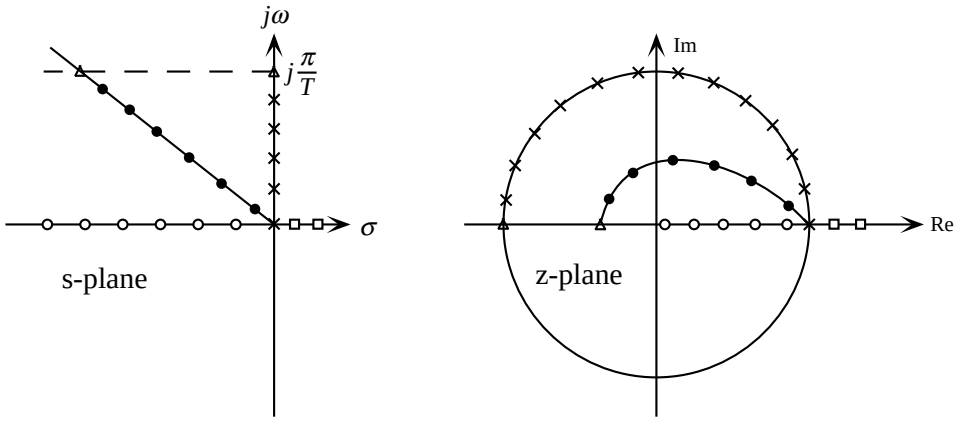


Figure 2.7: Contours in s and z planes.

- Line of constant damping ratio in s plane. This is a radial line, which maps into a logarithmic spiral in the z plane.

Note that the stability boundary in the s-plane ($j\omega$ axis) maps into the stability boundary in the z-plane (unit circle).

However, this mapping is not unique, for consider

$$s_2 = s_1 + j\frac{2\pi}{T} \tag{2.58}$$

where $\frac{2\pi}{T} = \omega_s$, the sampling frequency in rad/sec. Mapping s_2 to the z plane yields

$$z_2 = e^{s_2 T} = e^{\left(s_1 + j\frac{2\pi}{T}\right)T} = e^{s_1 T} e^{j2\pi} = e^{s_1 T} \tag{2.59}$$

thus two different values of s map into the same value of z. These s values correspond to the poles of a Laplace transform of a continuous signal, while the z value corresponds to the poles of a Z transform of the discrete signal obtained by sampling the continuous signal. Hence the samples of two different continuous signals will be the same. This phenomenon is called *aliasing*, and will be examined mathematically in Chapter 4. The s-plane boundary at which aliasing occurs is shown by the dashed line in Figure 2.7.

As a point moves up the s-plane $j\omega$ axis from zero, the corresponding z-plane point travels counter-clockwise around the unit circle, starting at $+1$. When $s = j\frac{\pi}{T}$ then $z = -1$. As s continues up the $j\omega$ axis, the value of z does *not* continue on the unit circle, but "jumps" out to another level. The z-plane is actually a *Riemann space*, but we will not pursue this concept further.

2.6 Frequency Response of Discrete Systems

Recall that for a continuous system the amplitude ratio and phase angle can be found by substituting $s = j\omega$ into the s-domain transfer function, and evaluating the magnitude and angle of the resulting

expression.

A similar technique works for discrete systems, and will be given without proof. Consider a discrete system with input u_k, output y_k, and discrete transfer function

$$H(z) = \frac{Y(z)}{U(z)} \tag{2.60}$$

If the input is a sinusoid at frequency ω_o

$$u_k = U_o \cos(\omega_o kT) \tag{2.61}$$

the steady-state output will also be sinusoidal, of the form

$$y_k = Y_o \cos(\omega_o kT + \phi) \tag{2.62}$$

where the amplitude ratio Y_o/U_o and phase angle ϕ are given by

$$\frac{Y_o}{U_o} = |H(z)|_{z=e^{j\omega_o T}} \tag{2.63}$$

and

$$\phi = \angle\, H(z)|_{z=e^{j\omega_o T}} \tag{2.64}$$

There will be numerous calculations of the frequency response of discrete systems in Chapter 3, so we will defer examples until that point.

2.7 Z-Transform Properties

In this section are a few of the more useful properties of the z-transform.

1. *Linearity.* The z-transform is linear, *i.e.*

$$\mathcal{Z}\left\{\alpha f_1(kT) + \beta f_2(kT)\right\} = \alpha F_1(z) + \beta F_2(z)$$

2. *Time Shift.* We have seen this with the unit delay and advance:

$$\mathcal{Z}\left\{f(k+n)\right\} = z^n F(z)$$

3. *Final Value Theorem.* If $(z-1)F(z)$ has all poles inside the unit circle, and has no poles on the unit circle except possibly one pole at $z = 1$ (*i.e.*. not unstable), then the terms in the sequence $f(k)$ due to poles inside the unit circle $\to 0$ as $k \to \infty$. The only value left as $k \to \infty$ will be that due to the pole at $z = 1$, this value is the final value, and may be found as the residue of the $\frac{1}{z-1}$ term in a partial fraction expansion of $F(z)$.

$$\lim_{k\to\infty} f(k) = (z-1)\, F(z)|_{z=1} \tag{2.65}$$

This is called the Final Value Theorem, and is very useful. The final value theorem may be used to find the *DC gain* of a discrete transfer function.

DC gain of transfer function

Given

$$H(z) = \frac{Y(z)}{U(z)} \tag{2.66}$$

let input u_k be a step of magnitude u_∞, with z-transform

$$U(z) = \frac{u_\infty z}{z-1} \tag{2.67}$$

The output is given by

$$Y(z) = H(z)U(z) = H(z)\frac{u_\infty z}{z-1} \tag{2.68}$$

The final value of output sequence y_k can be found using the final value theorem, and is

$$\lim_{k\to\infty} = y_\infty = (z-1)\, Y(z)|_{z=1} = (z-1)\, H(z)\frac{u_\infty z}{z-1}\bigg|_{z=1} = u_\infty H(1) \tag{2.69}$$

Hence the DC gain of the transfer function $H(z)$ is

$$\frac{y_\infty}{u_\infty} = H(1) \tag{2.70}$$

Note that to apply the final value theorem to a discrete variable, that variable *must have a final value*. Responses that are unbounded will yield arithmetic results which are not valid.

When finding the DC gain of a transfer function, *all poles of the transfer function must be inside the unit circle.*

Example

Consider the discrete transfer function given by

$$H(z) = \frac{Y(z)}{U(z)} = \frac{z+1}{z^2 - 0.5z + 0.5} = \frac{z+1}{z - 0.25 \pm j0.66} \tag{2.71}$$

Note that it is necessary to find the pole locations of this transfer function to make sure it is stable (all poles inside unit circle). The DC gain is given by

$$H(1) = \frac{1+1}{1 - 0.5 + 0.5} = 2 \tag{2.72}$$

Thus if this discrete system were given an input that eventually reached a constant value, the output would eventually reach twice that value.

NOTE: If the denominator polynomial above were $z^2 - 0.5z + 2$, the DC gain would evaluate to $H(1) = 0.8$, but that is meaningless since the system is unstable.

2.7.1 Inverse Transforming

Just as in continuous linear dynamic system analysis, given a z-transform $F(z)$ it is of fundamental importance to obtain the corresponding discrete sequence $f(k)$. There are several methods.

Synthetic division

Given the definition of the z-transform for $F(z)$, expressed as a ratio of polynomials in z^{-1}:

$$F(z) = \lim_{k=0}^{\infty} f(k)z^{-k} = \frac{b(z^{-1})}{a(z^{-1})} \tag{2.73}$$

The discrete sequence f_k may be recovered from $F(z)$ by direct synthetic division. The coefficient of z^{-k} in the quotient will be the k^{th} sample.

Example

An s-plane pole at $s = -2$ has time constant $\tau = 0.5$ sec. Let sample period $T = 0.1$ sec, corresponding to sampling frequency $f_s = 10$ Hz. (Note that T is one-fifth $\tau \ldots$ fast enough to "catch" the dynamic response.) The s-plane pole corresponds to a z-plane pole at

$$z = e^{sT} = e^{-0.2} = 0.8187 \tag{2.74}$$

Define a discrete transfer function with this pole and a DC gain of 1, thus

$$H(z) = \frac{0.1813}{z - 0.8187} = \frac{Y(z)}{U(z)} \tag{2.75}$$

Let the input $U(z)$ be a unit step, $z/(z-1)$, thus the response $Y(z)$ is

$$Y(z) = \frac{0.1813z}{(z - 0.8187)(z - 1)} = \frac{0.1813z}{z^2 - 1.8187z + 0.8187} = \frac{0.1813z^{-1}}{1 - 1.8187z^{-1} + 0.8187z^{-2}} \tag{2.76}$$

The sequence $y(k)$ can be obtained by the synthetic division of the polynomials in $Y(z)$ above. As mentioned above, the coefficient of z^0 in the quotient will be the 0^{th} sample, the coefficient of z^{-1} will be the 1^{st} sample, the coefficient of z^{-2} will be the 2^{nd} sample, and so on. This method is tedious for manual computation, but lends itself well to construction of a numerical algorithm for inversion by computer.

Difference equation

In this method we convert the discrete transfer function to a difference equation (see Section 2.3.4), then simply evaluate the equation as presented at the beginning of Section 2.2. We will use this method on the system of the previous example.

Example

Again take discrete transfer function

$$H(z) = \frac{0.1813}{z - 0.8187} = \frac{Y(z)}{U(z)} \tag{2.77}$$

with corresponding transform equation

$$Y(z) - 0.8187z^{-1}Y(z) = 0.1813z^{-1}U(z) \tag{2.78}$$

or

$$y_k - 0.8187y_{k-1} = 0.1813u_{k-1} \tag{2.79}$$

which yields

$$y_k = 0.8187y_{k-1} + 0.1813u_{k-1} \tag{2.80}$$

This difference equation can be evaluated for any input. For a unit step input one can generate the following table:

k	u_k	y_k
0	1	0
1	1	0.1813
2	1	0.3297
3	1	0.4513
4	1	0.5507
5	1	0.6322
6	1	0.6989
7	1	0.7535
⋮	⋮	⋮
∞	1	1.0000

Note that the response y_k reaches 63% of the final value at $k = 5$, or $t = 0.5$ sec, the time constant of the original continuous pole. This method is acceptable for hand calculation for a few samples, but tedious for many samples.

Table lookup

Just as with the Laplace transform, the discrete sequence $f(k)$ corresponding to $F(z)$ may be found by table lookup. It is often complicated by the need to do partial fraction expansion. Again, consider the same example,

$$Y(z) = \frac{0.1813z}{(z - 0.8187)(z - 1)} = \frac{0.1813z}{z^2 - 1.8187z + 0.8187} \tag{2.81}$$

This $Y(z)$ is not in the table of Section 2.9 (admittedly a small table) so we must expand in partial fractions

$$Y(z) = \frac{0.1813z}{(z-0.8187)(z-1)} = \frac{0.1813z}{z^2 - 1.8187z + 0.8187} = \frac{1}{z-1} - \frac{0.8187}{z-0.8187} \tag{2.82}$$

Transform numbers 1 and 3 from Section 2.9 come close to fitting these terms, but they must be expressed as

$$Y(z) = (z^{-1})\frac{z}{z-1} + (0.8187z^{-1})\frac{z}{z-0.8187}. \tag{2.83}$$

Transform pairs 1 and 3 are then applicable, and since z^{-1} is the unit delay, we can write

$$y_k = 1(k-1) + 0.8187^k \tag{2.84}$$

which should yield the same result as the previous two methods.

MATLAB simulation

This is by far the most useful method, *providing that the problem is numerical!*. Consider the previous transfer function,

$$H(z) = \frac{0.1813}{z - 0.8187} \tag{2.85}$$

An easy way to "inverse transform" $Y(z)$ using `MATLAB` is to employ the `dlsim(num,den,u)` function (discrete linear simulation), where `num` is the numerator polynomial of the transfer function (coefficients of decreasing powers of z), `den` is the denominator polynomial, and `u` is the input. The `MATLAB` script below does this for a unit step:

```
>>u = ones(size(0:10));  % Define input to be "1" from samples 0 to 10
>>num = [0 0.1813];      % Numerator coefficients of z
>>den = [1 -0.8187];     % Denominator coefficients of z
>>y = dlsim(num,den,u)   % Do the simulation, displaying resulting samples

y =

         0
    0.1813
    0.3297
    0.4513
    0.5507
    0.6322
    0.6989
    0.7535
    0.7982
    0.8348
    0.8647

>>plot(0:10,y,'o',0:10,y); % Plot samples as "o" connected by lines
```

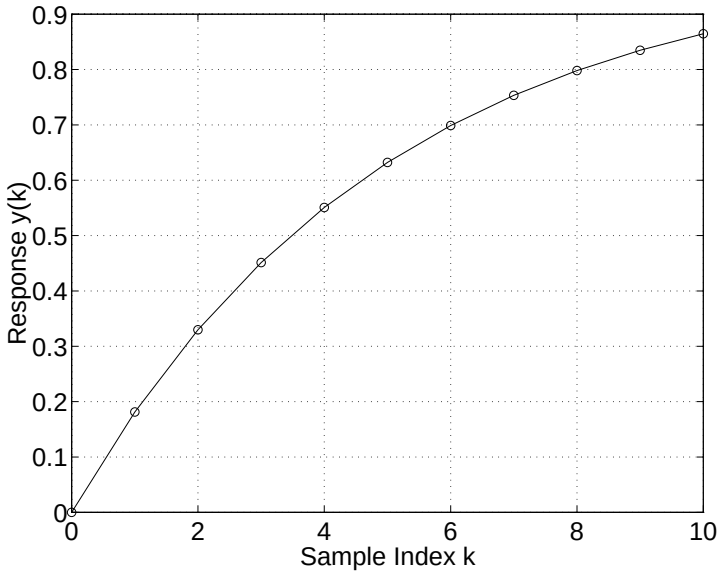


Figure 2.8: Step response of discrete system.

The resulting samples are the same as the previous methods. The plot drawn by the script is shown. Note that it plots an "o" at each sample point, this was determined by the argument in the `plot` function. It also connects the points with lines, this is not true of the discrete output but it does aid in viewing.

Other MATLAB Functions (`DSTEP, DDCGAIN`)

Although `dlsim` is more general the `DSTEP(NUM,DEN)` function produces the unit step response directly. I usually use it with a returned argument as follows:

```
>> y = dstep(num,den);  % Obtain unit step response
>> plot(...);           % Plot as desired...
```

The DC Gain `K` of a discrete transfer function may be found using `ddcgain` if the system is described using explicit `num` and `den`, *or* using `dcgain` if the system is described using an "LTI" model (see Section 2.8 below).

```
>> K = DDCGAIN(NUM,DEN);  % NUM and DEN are vectors w/polynomial coefficients
>> K = DCGAIN(SYS);       % SYS is a discrete transfer function in LTI format
```

2.8 A Word about LTI Systems and MATLAB functions

There has been a fundamental change in MATLAB since I first wrote these notes in 1998. This has do to with the "LTI System" and the notion of "overloaded functions."

2.8.1 LTI Systems

An LTI System is MATLAB's representation of any dynamic system (either continuous- or discrete-time, modeled using either transfer-function or state-variable formulation), and can be created in several ways:

- TF—using numerator and denominator of transfer function (either continuous- or discrete-time)
- ZPK—using zeros, poles, and gain of a transfer function (either...)
- SS—using state-space formulations (Chapters 6-8 of these notes)

Sometimes it is advantageous to cast systems into LTI format (certain MATLAB functions require that)—otherwise it may be useful to have them in numerator/denominator, zero/pole/gain, or explicit state-space format. You will see all representations at various times in these notes. LTI systems are created using the three MATLAB functions listed above (`TF,ZPK,SS`)—type `help tf`, `help zpk`,*etc* in MATLAB to see more discussion.

Note that if a sample period T is used when creating an LTI model, then that model is automatically made a discrete-time model. This is very important.

2.8.2 Overloaded Functions

The concept of *overloading* has its origin in *object-oriented programming*, and means that the same function will operate differently on different objects. For MATLAB, this means that the same function can be used on different LTI systems, *e.g.* **continuous-time** and **discrete-time**.

As an example, consider the MATLAB `step` function, which produces the unit step response of a dynamic system. An excerpt from the MATLAB "help" resource indicates:

```
>> help step
 STEP  Step response of LTI models.

    STEP(SYS) plots the step response of the LTI model SYS (created
    with either TF, ZPK, or SS).  For multi-input models, independent
    step commands are applied to each input channel.  The time range
    and number of points are chosen automatically.
    ...
```

Note that there is no mention of whether "`sys`" is continuous or discrete—the `step` function will handle them both! When you create the LTI object you specify its nature; this is used by `step` to compute the response correctly.

Consider the following two transfer functions $G(s)$ and $G(z)$:

$$G(s) = \frac{1}{s+1} \qquad\qquad G(z) = \frac{0.1813}{z - 0.8187},\ T = 0.2$$

Now create LTI systems `Gc` and `Gd` for each transfer function:

```
>> Gc = tf([0 1],[1 1])    % Enter numerator and denominator polynomials

Transfer function:
  1
-----
s + 1

>> Gd = tf([0 0.1813],[1 -0.8187],0.2)    % Note sample period T = 0.2

Transfer function:
  0.1813
----------
z - 0.8187

Sampling time: 0.2
```

By appending the "`0.2`" to the creation of `Gd` we obtained a discrete-time LTI model.

We can apply the `step` function to *either* model:

```
>> step(Gc);
>> step(Gd);
```

Try it! Both commands will produce plots of the response; the discrete-time system will yield a "stairstep" plot (sometimes appropriate, sometimes not).

Many MATLAB functions are "overloaded" and apply to both types of systems. I'm slowly revising these notes, and I'll try to show examples during class.

2.9 Table of Z-Transforms

The short table below lists some z-transforms. $\mathcal{F}(s)$ is the Laplace transform of $f(t)$ and $F(z)$ is the z-transform of $f(kT)$.

Number	$\mathcal{F}(s)$	$f(kT)$	$F(z)$
1	$\frac{1}{s}$	$1(kT)$	$\frac{z}{z-1}$
2	$\frac{1}{s^2}$	kT	$\frac{Tz}{(z-1)^2}$
3	$\frac{1}{s+a}$	e^{-akT}	$\frac{z}{z-e^{-aT}}$
4	$\frac{1}{(s+a)^2}$	kTe^{-akT}	$\frac{Tze^{-aT}}{(z-e^{-aT})^2}$
5	$\frac{a}{s(s+a)}$	$1-e^{-akT}$	$\frac{z(1-e^{-aT})}{(z-1)(z-e^{-aT})}$
6	$\frac{a}{s^2+a^2}$	$\sin akT$	$\frac{z\sin aT}{z^2-(2\cos aT)z+1}$
7	$\frac{s}{s^2+a^2}$	$\cos akT$	$\frac{z(z-\cos aT)}{z^2-(2\cos aT)z+1}$
8	$\frac{s+a}{(s+a)^2+b^2}$	$e^{-akT}\cos bkT$	$\frac{z(z-e^{-aT}\cos bT)}{z^2-2e^{-aT}(\cos bT)z+e^{-2aT}}$
9	$\frac{b}{(s+a)^2+b^2}$	$e^{-akT}\sin bkT$	$\frac{ze^{-aT}\sin bT}{z^2-2e^{-aT}(\cos bT)z+e^{-2aT}}$
10	$\frac{a^2+b^2}{s((s+a)^2+b^2)}$	$1-e^{-akT}\left(\cos bkT+\frac{a}{b}\sin bkT\right)$	$\frac{z(Az+B)}{(z-1)(z^2-2e^{-aT}(\cos bT)z+e^{-2aT})}$
			$A=1-e^{-aT}\cos bT-\frac{a}{b}e^{-aT}\sin bT$
			$B=e^{-2aT}+\frac{a}{b}e^{-aT}\sin bT-e^{-aT}\cos bT$

Homework Problems

Problem 1.

Given the difference equation

$$y_k = 0.5y_{k-1} + 0.5y_{k-2} + 0.25u_{k-1}$$

(a) Find the discrete transfer function $H(z) = \frac{Y(z)}{U(z)}$.

(b) Draw a block diagram of this discrete system.

(c) Reduce the block diagram to obtain a transfer function and verify with (a).

Problem 2.

(a) Derive the difference equation corresponding to the approximation of integration found by fitting a parabola $\hat{e}(t) = a_0 + a_1 t + a_2 t^2$ to the points e_{k-2}, e_{k-1}, e_k and taking the area under this parabola $\hat{e}(t)$ between $t = kT - T$ and $t = kT$ as the approximation to the integral of $e(t)$ over this range. (Hint: In solving for coefficients a_0, a_1, a_2 pick samples such that $t = 0$ is included, *i.e.* e_{-2}, e_{-1}, e_0, or e_{-1}, e_0, e_1. This will greatly simplify the algebra.

(b) Find the transfer function of the resulting discrete system and plot the poles and zeros in the z-plane.

(c) Test the accuracy of this "parabolic rule" integrator by integrating the sine function $e(t) = \sin 2\pi t$ over the first half cycle. Use six time steps, $k = 0 \ldots 5$; thus $T = 0.1$. Perform this integration using:

i. Parabolic rule

ii. Trapezoidal rule

iii. Exact integration

Compare the three results. Is anything surprising?

Problem 3.

The standard form of the continuous-domain transfer function for an underdamped system is

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

which has pole locations at $s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$. Given corresponding z-plane complex poles at $z = a \pm jb$, find the equivalent damping ratio ζ and natural frequency ω_n of these discrete system poles in terms of a, b, and sampling period T.

Problem 4.

For the discrete transfer function $G(z)$ below:

$$G(z) = \frac{1}{z^2 - 0.5z + 0.5}$$

(a) Find and plot the unit pulse response.

(b) Find and plot the unit step response. Also verify the DC gain using the method of Section 2.7.

(c) Find and plot the response to a sine input of 0.5 Hz. Assume a sample period $T = 0.2$ seconds. Verify the amplitude ratio and phase shift with that computed analytically using the frequency response method of Section 2.6. (Hint: You should use `MATLAB` to do this one, there are too many samples to compute by hand, since you must simulate it until the response reaches steady state).

Chapter 3

Discrete Simulation of Continuous Systems

Finding a discrete system which simulates the behavior of a given continuous system is important in discrete-domain control system design, since the (continuous) plant's response to a discretized input must be known before a controller can be designed for that plant.

However, the study of discrete simulations of continuous systems is important in its own right, because we often have a known $H(s)$ which we wish to realize using a computer, hence we must find a discrete system $H(z)$ which yields the same behavior. All dynamic systems $H(s)$ may be considered filters, hence the subject matter of this chapter is sometimes called *digital filtering*.

3.1 Chapter Overview

Linear filters may be described by their frequency behavior, *i.e.* their amplitude and phase characteristics *vs* frequency. In this chapter we will often judge the accuracy of our discrete simulations by comparing their frequency characteristics to the frequency characteristics of the original continuous system. Some types of filters are

- Radio receiver filter—*bandpass*
- Power line noise filter—*bandstop*
- Audio applications—*equalizer*
- Control system filters—*compensators*

These are all fundamentally similar, they have certain amplitude ratio and phase shift *vs* frequency. Continuous filter design is an established area. We will assume the continuous filter is known, and pose the problem

Given $H(s)$, what $H(z)$ will have approximately the same characteristics?

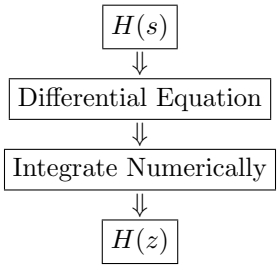
We will look at two approaches...

1. Numerical Integration.
2. Pole-Zero Mapping.

3.2 Discrete Simulation using Numerical Integration

You have probably used numerical integration to solve differential equations. We are doing the same thing here, except we are going to use the result in *real time*. By the way, trying to define what "real-time" means has resulted in endless arguments. For the purposes of this class, "real time" will mean "keeping up with the hardware." Controlling a physical process will require sampling at a desired rate. Our digital controller must be able to generate outputs at this rate. Typically, need for real-time behavior precludes the use of "batch" or multi-user computer operating systems (like Unix), since they do not guarantee any minimum response or sampling time. One of the figures of merit of a real-time operating system (of which there are many, *e.g.* VxWorks, LynxOS (not Linux!), MATLAB Real-Time Workshop) is its *interrupt latency*, which defines how fast the computer can respond to an external event, such as the need of the controlled process for an input. Interrupt latencies of competitive real-time OS's are well under 10 μsec.

In using numerical integration to obtain a real-time discrete equivalent of a continuous system, we do the following:



Given the need for fast response, we typically use simple numerical integration algorithms... you won't see any $4^{th} - 5^{th}$ order Runge-Kutta methods here! The process is best illustrated by an example:

Example

Given the first-order system

$$H(s) = \frac{U(s)}{E(s)} = \frac{a}{s+a} \tag{3.1}$$

we can cross-multiply to get

$$sU(s) + aU(s) = aE(s) \Rightarrow \dot{u}(t) + au(t) = ae(t) \tag{3.2}$$

hence

$$\dot{u}(t) = -au(t) + ae(t) \tag{3.3}$$

We can therefore integrate $\dot{u}$ to obtain u,

$$u(t) = \int_0^t [-au(\tau) + ae(\tau)]d\tau \tag{3.4}$$

where the dummy time variable τ is used as the integrand to avoid confusion with time variable t, the upper limit on the integration.

In discrete time, we can express this integral in two parts... the integral from $t = 0$ to the previous sample time $t = kT - T$, and the integral from $t = kT - T$ to the current sample time $t = kT$, or

$$\begin{aligned} u(kT) &= \int_0^{kT-T} (-au + ae)d\tau + \int_{kT-T}^{kT} (-au + ae)d\tau \\ &= u(kT - T) + [\text{Area of } \ -au + ae \ \text{ over } \ kT - T \leq \tau \leq kT] \end{aligned} \tag{3.5}$$

This area may be approximated with many numerical rules, we will show four:

- Forward rectangular
- Backward rectangular
- Trapezoidal
- Prewarped trapezoidal

3.2.1 Forward rectangular rule (Euler's rule)

The integration to be performed is shown in Figure 3.1, where the function $g(\tau)$ to be integrated is $g(\tau) = -au(\tau) + ae(\tau)$ for the previous example. With the forward rule we project the value of the function g at the previous sample "forward" to the current sample, then take the area A of the resulting rectangle. Thus for our example function

$$\begin{aligned} u(kT) &= u(kT - T) + [-au(kT - T) + ae(kT - T)]T \\ &= (1 - aT)u(kT - T) + aTe(kT - T) \end{aligned} \tag{3.6}$$

From 3.6 we can find a transfer by knowing that discrete variable $u(kT)$ has z-transform $U(z)$, and that a delay of one sample corresponds to multiplication by z^{-1}. Thus

$$U(z) = (1 - aT)z^{-1}U(z) + aTz^{-1}E(z) \tag{3.7}$$

or

$$zU(z) = (1 - aT)U(z) + aTE(z) \tag{3.8}$$

yielding discrete transfer function

$$\frac{U(z)}{E(z)} = H(z) = \frac{aT}{z - (1 - aT)} = \frac{a}{(\frac{z-1}{T}) + a} \tag{3.9}$$

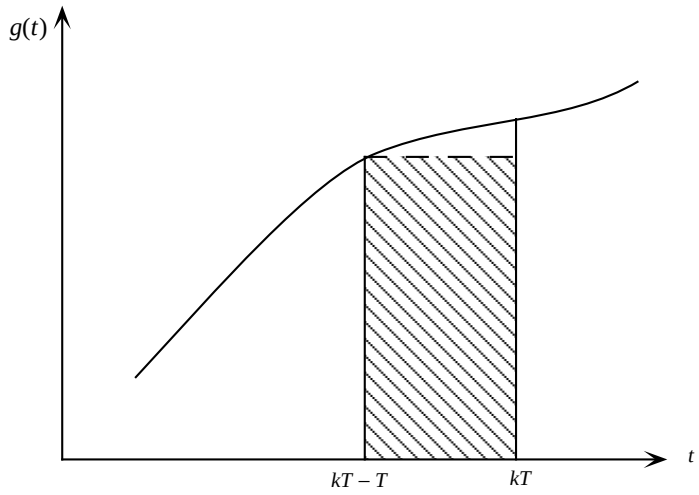


Figure 3.1: Forward integration rule.

3.2.2 Backward rectangular rule

The integration to be performed is shown in Figure 3.2, where the function $g(\tau)$ to be integrated is again $g(\tau) = -au(\tau) + ae(\tau)$. With the backward rule we project the value of the function g at the current sample "back" to the previous sample, then take the area A of the resulting rectangle. Thus for our example function

$$\begin{aligned} u(kT) &= u(kT-T) + [-au(kT) + ae(kT)]T \\ u(kT)(1+aT) &= u(kT-T) + aTe(kT) \end{aligned} \tag{3.10}$$

From (3.10) we can again find a transfer function as in Section 3.2.1. Thus

$$U(z)(1+aT) = z^{-1}U(z) + aTE(z) \tag{3.11}$$

or

$$U(z)(z+aTz) = U(z) + aTzE(z) \tag{3.12}$$

thus

$$U(z)(z+aTz-1) = aTzE(z) \tag{3.13}$$

yielding discrete transfer function

$$\frac{U(z)}{E(z)} = H(z) = \frac{aTz}{z-1+aTz} = \frac{a}{(\frac{z-1}{Tz})+a} \tag{3.14}$$

3.2.3 Trapezoidal rule

This integration rule, sometimes called the *Bilinear Transformation* or *Tustin Approximation*, is shown in Figure 3.3, where the function $g(\tau)$ to be integrated is still

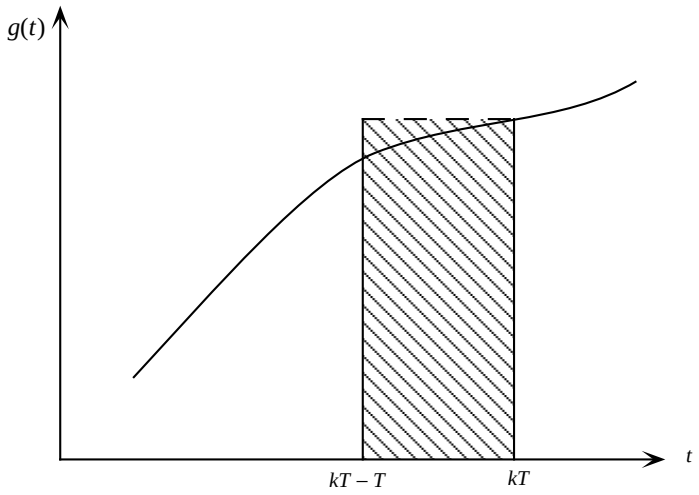


Figure 3.2: Backward integration rule.

$$g(\tau) = -au(\tau) + ae(\tau)$$

With the trapezoidal rule we connect the current sample to the previous sample with a straight line, then take the area A of the resulting trapezoid. This may be thought of as an average of the forward rectangle and the backward rectangle. For our example function

$$u(kT) = u(kT - T) + \frac{T}{2}[-au(kT - T) + ae(kT - T) - au(kT) + ae(kT)] \tag{3.15}$$

From (3.15) we can again find a transfer function,

$$U(z) = z^{-1}U(z) + \frac{T}{2}\left[-az^{-1}U(z) + az^{-1}E(z) - aU(z) + aE(z)\right] \tag{3.16}$$

and after some algebra we obtain discrete transfer function

$$\frac{U(z)}{E(z)} = H(z) = \frac{a\frac{T}{2}(z+1)}{z - 1 + a\frac{T}{2}(z+1)} = \frac{a}{\left(\frac{2}{T}\frac{z-1}{z+1}\right) + a} \tag{3.17}$$

Comparison of the Integration Rules

Comparing the original $H(s) = \dfrac{a}{s+a}$ with the three discrete equivalents $H(z)$, an approximation for s results:

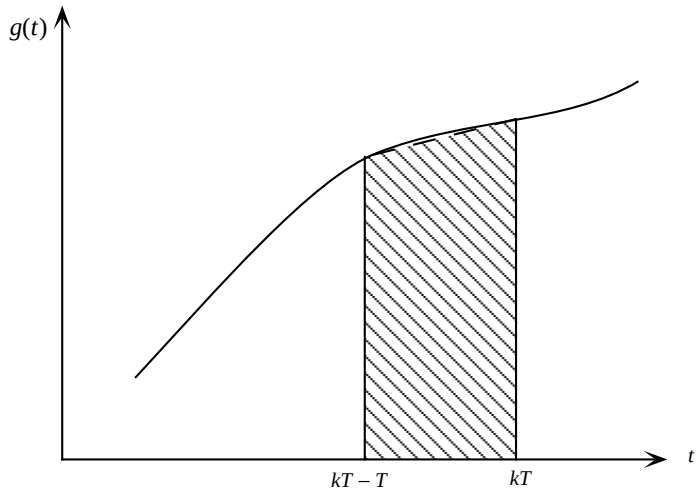


Figure 3.3: Trapezoidal integration rule.

Method	Approximation
Forward Rectangular	$s \approx \dfrac{z-1}{T}$
Backward Rectangular	$s \approx \dfrac{z-1}{Tz}$
Trapezoidal Rule	$s \approx \dfrac{2}{T}\dfrac{z-1}{z+1}$

Each of these approximations may be viewed as a mapping from the s-plane to the z-plane.

It is instructive to see where in the z-plane these approximations transform the left-half s-plane (the region of stable poles). To this end, solve for the inverse mapping, thus

Method	Inverse Mapping
Forward Rectangular	$z = 1 + Ts$
Backward Rectangular	$z = \dfrac{1}{1-Ts}$
Trapezoidal Rule	$z = \dfrac{1+Ts/2}{1-Ts/2}$

then we let $s = j\omega$, which is the boundary at the left-half s-plane. It can be shown that the s-plane $j\omega$ axis maps into the following z-plane contours:

Method	Equivalent $s = j\omega$ contour in z-plane
Forward Rectangular	Vertical line at $z = 1$
Backward Rectangular	Circle of radius $\frac{1}{2}$ centered at $z = \frac{1}{2}$
Trapezoidal Rule	Unit circle (although entire $+j\omega$ axis is stuffed into π of unit circle... much distortion!)

The left-half s-plane mapped into the z-plane with these approximations is shown in Figure 3.4.

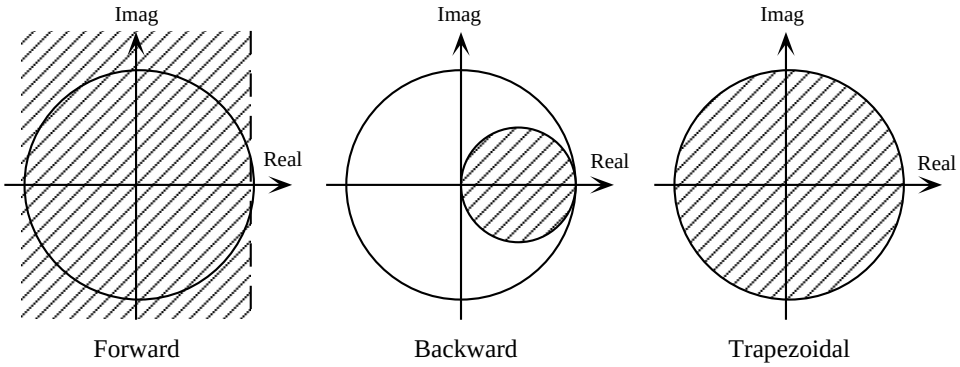


Figure 3.4: Left-half s-plane mapped into z-plane.

Comments

Note that with the forward rectangular rule there is the potential for a stable continuous system (all poles in left-half s-plane) resulting in an unstable discrete approximation.

Even though the trapezoidal rule maps the s-plane stability boundary into the z-plane stability boundary, there is still distortion, as indicated in the table. This distortion inherent in the trapezoidal rule (there is even more distortion in the forward & backward rules) can be corrected exactly at *one and only one frequency*. This lead to the fourth integration rule.

3.2.4 Prewarped trapezoidal rule

This rule, a variation of the trapezoidal rule, corrects the frequency distortion at a given frequency. Start with the trapezoidal approximation,

$$s \approx \frac{2}{T}\frac{z-1}{z+1} \tag{3.18}$$

and consider a particular frequency ω_o, thus $s = j\omega_o$. The exact mapping of this pole is

$$z = e^{Ts} = e^{j\omega_o T}. \tag{3.19}$$

Now take the bilinear transformation

$$s = A\frac{z-1}{z+1}. \tag{3.20}$$

We are going to solve for A to obtain this exact frequency response. We do this by equating the actual s-plane pole location with the mapping of equation (3.20). Thus

$$\begin{aligned} j\omega_o &= A\frac{e^{j\omega_o T}-1}{e^{j\omega_o T}+1} = A\frac{e^{j\omega_o T/2}-e^{-j\omega_o T/2}}{e^{j\omega_o T/2}+e^{-j\omega_o T/2}} \\ &= jA\frac{\sin\frac{\omega_o T}{2}}{\cos\frac{\omega_o T}{2}} = jA\tan\frac{w_o T}{2} \end{aligned} \tag{3.21}$$

or

$$\boxed{A = \frac{\omega_o}{\tan\frac{\omega_o T}{2}}} \tag{3.22}$$

Note that when $\omega_o T$ is small, parameter $A \approx 2/T$, which is the same as the trapezoidal rule. The condition of small $\omega_o T$ means the sampling frequency is fast compared to frequency ω_o, and there is not much distortion in the first place.

Given prewarping frequency w_o, calculate A using equation (3.22), then equation (3.20) yields the substitution to use in finding the discrete simulation.

Example

All four integration rules will be used in discrete simulations of continuous filter

$$H(s) = \frac{10}{s+10} \tag{3.23}$$

which is a first-order low-pass filter with a cutoff frequency of 10 rad/second. Recall that the cutoff frequency is where amplitude ratio or magnitude is -3.01 db down from the low-frequency value, or a factor of 0.7071 below the low-frequency value. The magnitude and angle of $H(s)$ at the cutoff frequency are found by evaluating $H(s)$ at $s = j\omega = j10$:

$$|H(j\omega)| = |H(j10)| = \left|\frac{10}{j10+10}\right| = 0.7071 = -3.01db. \tag{3.24}$$

$$\angle H(j\omega) = \angle H(j10) = \angle\frac{10}{j10+10} = -45^o. \tag{3.25}$$

We will select a sampling frequency of f_s- 20 Hz, thus $T = 0.05$ seconds.

1. *Forward rectangular rule.* Here
$$s = \frac{z-1}{T} = 20(z-1)$$
and
$$H_F(z) = \frac{10}{20z - 20 + 10} = \frac{0.5}{z - 0.5} \tag{3.26}$$
This simulation has no zeros, a pole at $z = 0.5$, and a dc gain of 1. To find the magnitude and angle at 10 rad/second, let
$$z = e^{j\omega T} = e^{j(10)(0.05)} = e^{j0.5} = \cos 0.5 + j \sin 0.5 = 0.878 + j0.479 \tag{3.27}$$
and
$$\begin{aligned} H_F &= \frac{0.5}{0.378 + j0.479} \\ |H_F| &= 0.819 = -1.73db \\ \angle H_F &= -51.721^o \end{aligned}$$

2. *Backward rectangular rule.* Here
$$s = \frac{z-1}{Tz} = 20\left(\frac{z-1}{z}\right)$$
and
$$H_B(z) = \frac{10z}{20z - 20 + 10z} = \frac{0.33z}{z - 0.67} \tag{3.28}$$
This simulation has a zero at $z = 0$, a pole at $z = 0.67$, and a dc gain of 1. Again using $z = 0.878 + j0.479$, we find
$$\begin{aligned} H_B &= \frac{0.290 + j0.158}{0.208 + j0.479} \\ |H_B| &= 0.632 = -3.98db \\ \angle H_B &= -37.945^o \end{aligned}$$

3. *Trapezoidal rule.* Here
$$s = \frac{2}{t}\frac{z-1}{z+1} = 40\frac{z-1}{z+1}$$
and
$$H_T(z) = \frac{10(z+1)}{40z - 40 + 10z + 10} = \frac{0.2(z+1)}{z - 0.6} \tag{3.29}$$
This simulation has a zero at $z = -1$, a pole at $z = 0.6$, and a dc gain of 1. Once again using $z = 0.878 + j0.479$, we find
$$\begin{aligned} H_T &= \frac{0.376 + j0.096}{0.278 + j0.479} \\ |H_T| &= 0.701 = -3.089db \\ \angle H_T &= -45.547^o \end{aligned}$$

4. *Prewarped (at $\omega = 10$) trapezoidal rule.* Here

$$A = \frac{\omega}{\tan \frac{\omega T}{2}} = 39.16$$

and so

$$s = A\frac{z-1}{z+1} = 39.16\frac{z-1}{z+1}$$

therefore

$$H_P(z) = \frac{10(z+1)}{39.16z - 39.16 + 10z + 10} = \frac{0.203(z+1)}{z - 0.5932} \tag{3.30}$$

This simulation has a zero at $z = -1$, a pole at $z = 0.5932$, and a dc gain of 1. Letting $z = 0.878 + j0.479$, we find

$$\begin{aligned} H_P &= \frac{0.381 + j0.097}{0.284 + j0.479} \\ |H_P| &= 0.707 = -3.01db \\ \angle H_P &= -45^o \end{aligned}$$

The frequency response of the prewarped simulation is exact at $\omega = 10$.

Note that the trapezoidal method gives generally better results than either rectangular method, which is to be expected. The prewarped trapezoidal is naturally exact at the prewarping frequency, but may not be as good over other frequencies. Bode plots of magnitude and angle of all these simulations, plus the original continuous system will be given after the last two simulations methods are presented.

3.3 Pole-Zero Mapping

The pole-zero mapping method is probably the easiest method to apply, although most of these simulations can be performed using `MATLAB`, as we will show later in this chapter. Discrete simulation using pole-zero mapping is done by:

1. Map poles of $H(s)$ to poles of $H(z)$ using $z = e^{sT}$.

2. Map finite zeros (if any) of $H(s)$ to zeros of $H(z)$ using $z = e^{sT}$.

3. Map zeros at infinity (recall that $H(s)$ always has an equal number of poles and zeros, with perhaps some zeros at ∞) to $z = -1$.

4. Set the dc gain of $H(z)$ equal to that of $H(s)$.

Example

For the previous $H(s)$ of

$$H(s) = \frac{10}{s+10}$$

with $T = 0.05$ we have

1. The pole maps as $z = e^{(-10)(0.05)} = 0.6065$.
2. No finite zeros.
3. One zero at $z = -1$.
4. Adjust numerator K as necessary.

and we get

$$H_{PZ}(z) = \frac{K(z+1)}{z-0.6065} = \frac{0.1967(z+1)}{z-0.6065} \tag{3.31}$$

This simulation has a zero at $z = -1$, a pole at $z = 0.6065$, and a dc gain (obviously) of 1. Checking the frequency response as before, with $z = 0.878 + j0.479$, we find

$$\begin{aligned} H_{PZ} &= \frac{0.369 + j0.094}{0.271 + j0.479} \\ |H_{PZ}| &= 0.692 = -3.2db \\ \angle H_{PZ} &= -46.21^o \end{aligned}$$

Although one can draw no real conclusions, for this example, at this one frequency, the PZ mapping simulation was more accurate than the rectangular integration rules, but less accurate than the trapezoidal and prewarped rules.

3.4 Comparison of Simulations

All of the simulations will be compared by plotting their amplitude ratios and phase angles *vs* frequency... a Bode plot. As we saw in Chapter 2, the mapping between the s-plane and the z-plane breaks down (actually it becomes non-unique) as $s \to j\pi/T$, or $s \to j\omega_s/2$, one half the sampling frequency. For this reason we would expect the frequency response of these discrete simulations to degrade as we approach one-half the sampling frequency.

Bode plots of the original continuous system, and Forward, Backward, Trapezoidal, Prewarped, and PZ mapping discrete simulations are shown in Figure 3.5.

Although it is somewhat difficult to see in Figure 3.5, the forward and backward methods are definitely the worst, one is too high in magnitude, the other too low. The trapezoidal, prewarped, and pole-zero mapping simulations are all quite similar.

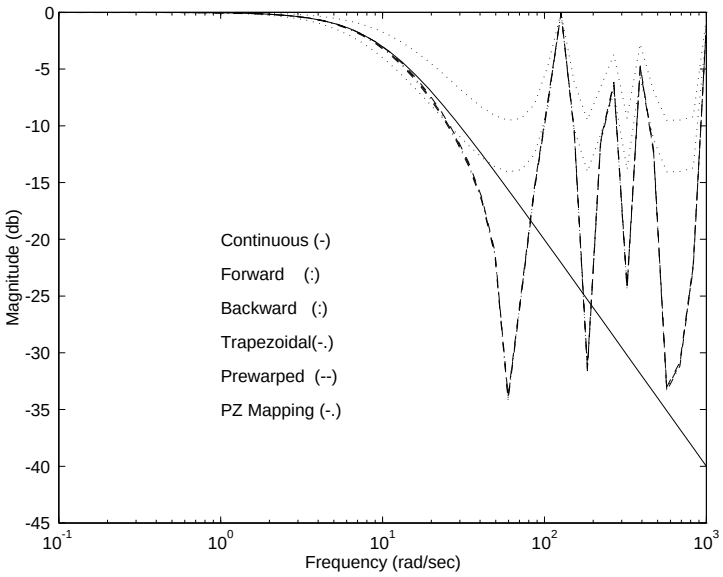


Figure 3.5: Bode magnitude plots.

Note that all methods behave very poorly as the frequency approaches one-half the sampling frequency, or $\omega = 2\pi f \approx 63$ rad/sec. The angle plots are shown in Figure 3.6. The trapezoidal, prewarped, and pole-zero mapping simulations are the best here as well, with all methods breaking down as frequency gets near $\omega_s/2$.

3.5 Using MATLAB in Discrete Simulation

The `MATLAB` numerical analysis package can be used in finding discrete simulations of continuous systems. Following is a description of the `c2d` function:

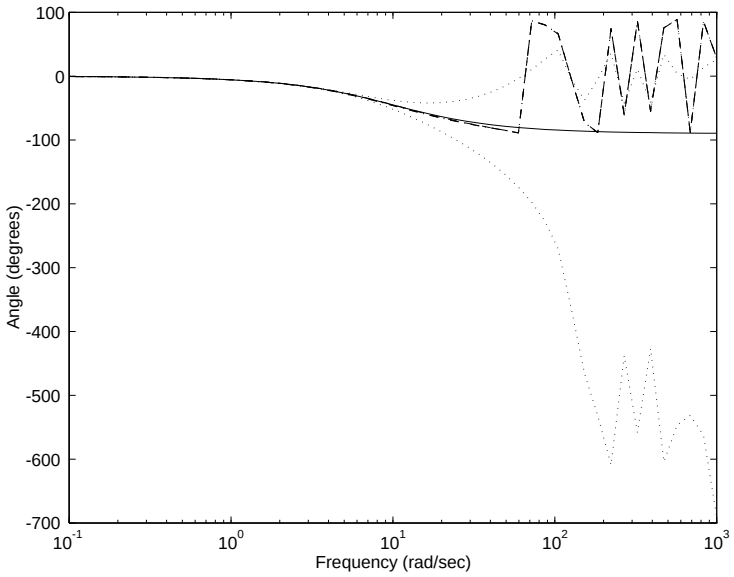


Figure 3.6: Bode angle plots.

```
C2D  Conversion of continuous-time models to discrete time.

   SYSD = C2D(SYSC,Ts,METHOD) converts the continuous-time LTI
   model SYSC to a discrete-time model SYSD with sample time Ts.
   The string METHOD selects the discretization method among the
   following:
      'zoh'       Zero-order hold on the inputs
      'foh'       Linear interpolation of inputs (triangle appx.)
      'imp'       Impulse-invariant discretization
      'tustin'    Bilinear (Tustin) approximation
      'prewarp'   Tustin approximation with frequency prewarping.
                  The critical frequency Wc (in rad/sec) is specified
                  as fourth input by
                     SYSD = C2D(SYSC,Ts,'prewarp',Wc)
      'matched'   Matched pole-zero method (for SISO systems only).
   The default is 'zoh' when METHOD is omitted.
```

The simulations methods that `MATLAB` supports that we have covered in this chapter are:

- `tustin`—trapezoidal (Tustin) integration rule
- `prewarp`—prewarped trapezoidal integration rule
- `matched`—pole-zero mapping

The rectangular integration rules are not supported in MATLAB, they are not good performers anyway. The zoh simulation method is one we will examine in Chapter 4, I have chosen to omit it in this chapter. *NOTE:* MATLAB function c2d works with an "LTI system," not the numerator and denominator. A "transfer function" LTI system can be created using the TF function—this will be done below (function c2dm still works with num and den, but c2d is preferred).

Example

Simulate the $H(s)$ covered in the previous example using MATLAB. Recall that

$$H(s) = \frac{10}{s+10} \tag{3.32}$$

1. *Trapezoidal rule.* The MATLAB script is shown below:

```
>> numc = [0 10];    % Define continuous numerator
>> denc = [1 10];    % Define continuous denominator
>> sysc = tf(numc,denc);    % Create continuous LTI system
>> T = 0.05;    % Define sampling period
>> sysd = c2d(sysc,T,'tustin');    % Discretize using trapezoidal approximation
>> sysd    % Display the resulting discrete LTI system

Transfer function:
0.2 z + 0.2
-----------
  z - 0.6

Sampling time: 0.05
```

2. *Prewarped rule.* The MATLAB script is shown below (the same continuous numerator and denominator are used as before):

```
>>wp = 10;    % Prewarping frequency of 10 rad/sec
>> sysp = c2d(sysc,T,'prewarp',wp);    % Discretize using prewarped
>> sysp    % Display the resulting discrete LTI system

Transfer function:
0.2034 z + 0.2034
-----------------
   z - 0.5932

Sampling time: 0.05
```

3. *Pole-zero mapping*:

```
>> syspz = c2d(sysc,T,'matched');    % Discretize using pole-zero mapping
>> syspz    % Display resulting discrete system

Transfer function:
  0.3935
----------
z - 0.6065

Sampling time: 0.05
```

3.5.1 Finding transfer function from LTI system

Sometimes you may need to extract the numerator and denominator from the LTI description of the system. Using the "pole-zero mapping" result obtained above, here's how to use `tfdata`:

```
>> [numpz,denpz] = tfdata(syspz,'v')

numpz =

         0    0.3935

denpz =

    1.0000   -0.6065
```

NOTE: Don't forget the `'v'` or you'll be in for grief!

3.6 Implementation of Difference Equations in Real Time

It is time to consider how to construct computer programs that will realize the difference equations we obtain. Consider a general second-order discrete transfer function, written in z^{-1}:

$$H(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{3.33}$$

with corresponding difference equation

$$y_k = a_1 y_{k-1} + a_2 y_{k-2} + b_0 u_k + b_1 u_{k-1} + b_2 u_{k-2} \tag{3.34}$$

We assume that in addition to A/D and D/A converters (and subroutines to use them) the computer has a programmable real time clock such that a clock interrupt will occur every T seconds, where T may be set under program control. When a clock interrupt occurs, variable `FLAG` is set to one.

3.6.1 Direct realization

Consider the following program structure:

```
10  read difference equation parameters and initialize variables;
20  set FLAG = 0;
30  select sample period, start clock;
40  if (FLAG == 1) print ("sampling period too short") and exit;
50  while (FLAG == 0) wait;
60  u(0) = adc_in();    % Read input u(k).
70  y(0) = a1*y(1) + a2*y(2) + b0*u(0) + b1*u(1) + b2*u(2); % compute y(k).
80  dac_out(y(0));      % Write output y(k) to D/A converter.
90  u(2) = u(1);        % Propagate variables
100 u(1) = u(0);        %  backwards
110 y(2) = y(1);        %  for next
120 y(1) = y(0);        %  sample.
130 goto 40;            % Execute continuously.
```

In the structure above the difference equation is explicitly calculated in line 70. While this approach is easy to follow, it is inefficient. The problem is the computational delay between the acquisition of the input sample in line 60 and the writing of the output sample in line 80. There are 5 multiplies, 4 adds, and 1 assignment. Note that this computational delay is *not* the sampling period, and it is a delay which we have not modeled. Thus it should be minimized.

Also note the possibility of clock overrun in line 40. If variable `FLAG` has already been set by the time we get back to the top of the loop, we are sampling too fast; the time allotted for the control computation is too short. Either we sample slower, reduce the complexity of the control calculation, or get a faster computer.

3.6.2 Canonical realization

Consider the block diagram shown in Figure 3.7. Input u and output y are present, as are parameters a_k and b_k. *State variables* x_1 and x_2 are new. While the general topic of state variable formulation will be deferred until later in the course, it is necessary to illustrate a more efficient real time implementation. Note that the state variables are the outputs of the delay elements... this will always be the case.

One should first verify that Figure 3.7 implements the difference equation of (3.34), this is left as an exercise for the student. A progam based on this block diagram is

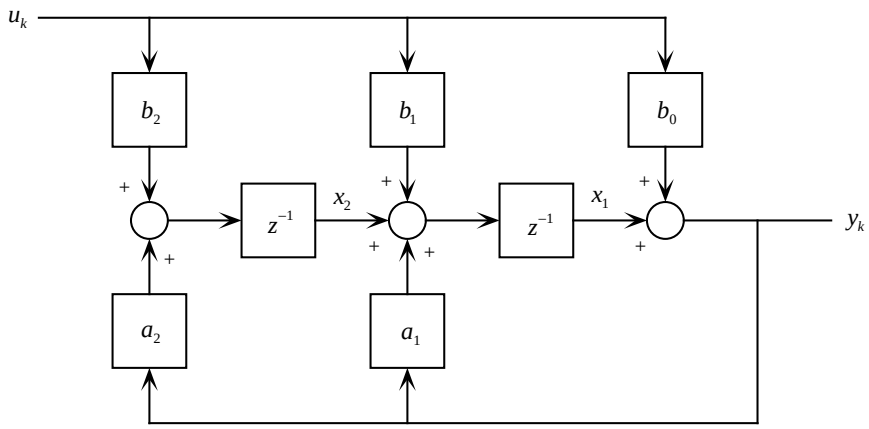


Figure 3.7: Observer canonical block diagram realization.

```
10  read difference equation parameters and initialize variables;
20  set FLAG = 0;
30  select sample period, start clock;
40  if (FLAG == 1) print ("sampling period too short") and exit;
50  while (FLAG == 0) wait;
60  u = adc_in();
70  y = x1 + b0*u;
80  dac_out(y);
90  x1 = x2 + b1*u + a1*y;
100 x2 = b2*u + a2*y;
130 goto 40;
```

Although the program loop is shorter overall, note particularly that the computational delay between line 60 and line 80 is now 1 multiply, 1 add, and 1 assignment. There is more computation *after* writing the output, but it's not as critical there.

Homework Problems

Problem 1.

The following transfer function is a lead network designed to add *about* 60^o phase lead at $\omega_1 = 3$ rad/sec:

$$H(s) = \frac{s+1}{0.1s+1} = 10\frac{s+1}{s+10}$$

(a) For each of the following simulation methods compute and plot in the z-plane the pole and zero locations and compute the amount of phase lead given by the network at $z_1 = e^{j\omega_1 T}$. Let $T = 0.25$ sec.

(i) Forward rectangular rule.

(ii) Backward rectangular rule.

(iii) Trapezoidal rule.

(iv) Prewarped trapezoidal rule (use ω_1 as the prewarping frequency).

(v) Pole-zero mapping.

(b) Plot the amplitude Bode plots of the continuous system and all simulations over the frequency range $0.1 \leq \omega \leq 100$ rad/sec.

(c) Develop a substitution rule based on the "parabolic" integration rule of Chapter 2 Problem 2. Use this rule to obtain a discrete simulation of the $H(s)$ and compare the frequency response with the other methods.

Problem 2.

For this problem, construct the continuous transfer function $G(s)$ of a second-order system with damping ratio $\zeta = 0.2$ and natural frequency $\omega_n = 50$ rad/sec, no zeros, and a dc gain of 1.

(a) Let the sampling frequency be 100 Hz. Find the following discrete simulations of $G(s)$:

(i) Trapezoidal.

(ii) Prewarped (use ω_n as the prewarping frequency).

(iii) Pole-zero mapping.

(b) Compare the Bode plots (both magnitude and phase plots) of the continuous $G(s)$ and all $G(z)$. Use frequency range $0.1 \leq f \leq 100$ Hz.

(c) Compare the unit step response of the $G(s)$ and any $G(z)$.

Problem 3.

A *notch filter* is often useful in reducing the effect of an unwanted signal at a known constant frequency, such as power line noise. The transfer function of a second-order notch filter is given by

$$H(s) = \frac{Y(s)}{U(s)} = \frac{s^2 + \omega_o^2}{s^2 + 2\zeta\omega_o s + w_o^2}$$

where $\zeta = \sqrt{2}/2$ and ω_o = notch frequency.

(a) Design a 60-Hz continuous notch filter $H(s)$.

(b) Using a sampling frequency of 500 Hz, find the prewarped discrete simulation $H(z)$ using the notch frequency as the prewarped frequency.

(c) Draw Bode magnitude plots of $H(s)$ and $H(z)$ over $1 \leq f \leq 1000$ Hz.

(d) Plot the time response of $H(z)$ to an input which is the sum of an 11 Hz. sine wave and a 60 Hz. sine wave, *i.e.*

$$u(t) = \sin 2\pi f_1 t + \sin 2\pi f_2 t = \sin 2\pi(11)t + \sin 2\pi(60)t$$

Show both input $u(t)$ and output $y(t)$ on your plot. Make sure $H(z)$ has reached steady state on your plot. Note the "start-up" transient.

(e) Draw a block diagram for $H(z)$ in the observer canonical format of Figure 3.7. Assign state variables x_i and write a program in "pseudo-code" to realize this filter. What must the excecution time of an arithmetic instruction be to implement this filter at a sampling rate of 500 Hz? Assume equal time for multiply and add, and neglect assignments. Is this within the capabilities of an "average" computer?

Chapter 4

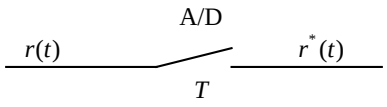
Sampled Data Systems

4.1 Introduction

In Chapter 2 we dealt with discrete systems only, covering difference equations and the z transform. Chapter 3 we did examine finding discrete simulations of continuous systems, but we did so from a discrete perspective. In digital control systems where there are both discrete (the control algorithm) and continuous components (the plant), with a loop closure, it is imperative that we study the interaction between discrete and continuous components with a more rigorous examination of the effect of sampling. In comparison to Chapter 3, now we will work primarily in the continuous time domain and use the Laplace transform.

4.2 The Sampling Process as Impulse Modulation

Consider a continuous signal which is sampled, as shown below, in which continuous signal $r(t)$ is



sampled to produce a discrete signal which we will designate $r^*(t)$. Note that this notation is a significant departure from our previous treatment. Before, we have labeled the sampled signal r_k, a discrete variable. However, now we are claiming that the sampled signal can be called $r^*(t)$, which implies we have a way of modeling this sampled signal in the time domain. We will obtain this time domain model of a sampled signal using *impulse modulation*.

The sampled signal may be represented as the continuous signal convolved with a sequence of unit impulses at the sample instants, which will result in the string of weighted impulses

$$r^*(t) = \delta(t)r(0) + \delta(t-T)r(T) + \delta(t-2T)r(2T) + \dots \tag{4.1}$$

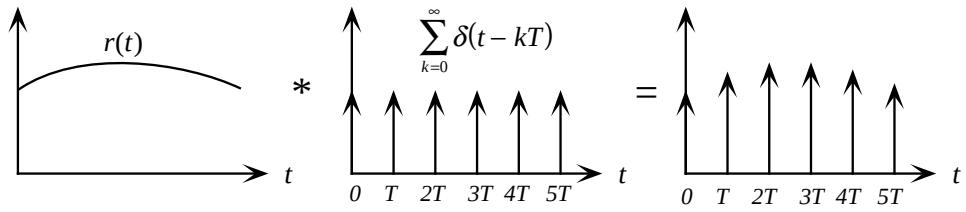


Figure 4.1: Convolution of continuous signal with impulse train.

thus

$$\begin{aligned} r^*(t) &= \sum_{k=0}^{\infty} r(t)\delta(t-kT) \\ &= \sum_{k=0}^{\infty} r(kT)\delta(t-kT) \end{aligned} \tag{4.2}$$

since the impulse train $\delta(t-kT)$ zeros out function $r(t)$ except at the sample instants kT. This convolution of continuous time function with impulse train is shown in Figure 4.1.

Using the definition of $r^*(t)$ in equation (4.2), $r^*(t)$ *is* defined for all time, since the impulse function $\delta(t)$ is defined over all time. We are going to apply the Laplace transform to equation (4.2), and in preparation for this, recall the "sieve" property of the impulse function...

$$\int_0^{\infty} f(t)\delta(t-a)dt = f(a) \tag{4.3}$$

where the $\delta(t-a)$ function results in only the value of $f(a)$ appearing in the integral result. This can be applied to the Laplace transform of $\delta(t)$, with the well-known result

$$\mathcal{L}[\delta(t)] = \int_0^{\infty} \delta(t)e^{-st}dt = 1 \tag{4.4}$$

Now take the Laplace transform of $r^*(t)$ as given by equation (4.2). We have

$$\begin{aligned} R^*(s) &= \mathcal{L}\left\{\sum_{k=0}^{\infty} r(kT)\delta(t-kT)\right\} \\ &= \sum_{k=0}^{\infty} r(kT)\int_0^{\infty} \delta(t-kT)e^{-st}dt \\ &= \sum_{k=0}^{\infty} r(kT)\mathcal{L}\left\{\delta(t-kT)\right\} \\ R^*(s) &= \sum_{k=0}^{\infty} r(kT)e^{-kTs} \end{aligned} \tag{4.5}$$

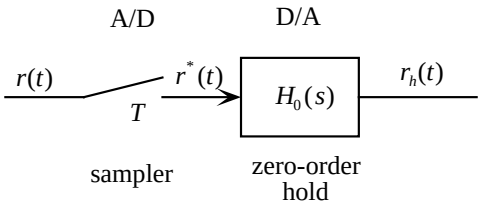


Figure 4.2: Sampler and zero-order hold.

Comparing equation (4.5) to the definition of the z-transform $\mathcal{Z}\{r_k\}$ in Chapter 2, we get the pleasing result

$$R^*(s) = R(z)|_{z=e^{Ts}} \tag{4.6}$$

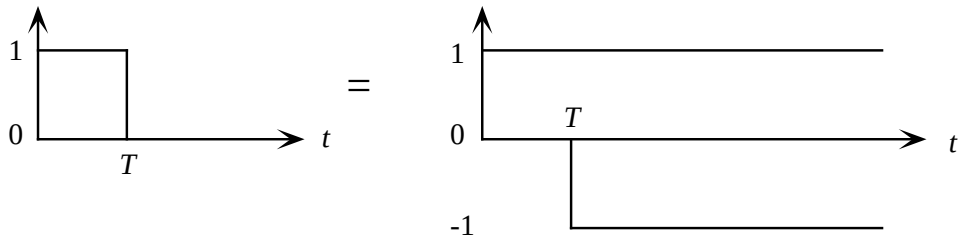
which states that the Laplace transform of the time function obtained by convolving $r(t)$ with impulse train $\delta(t-kT)$ is equal to the z transform of the discrete sequence $r(kT) = r(k) = r_k$, when the substitution $z = e^{Ts}$ is made. This is mathematical justification of the use of the impulse train to obtain a time function from the discrete sequence.

Example

Consider the use of the impulse model of sampling in finding the transfer function of a D/A converter (zero-order hold, or ZOH), shown in Figure 4.2. The D/A behaves like a ZOH, and holds the sample constant until a new sample is received, *i.e.*

1. Sample obtained from A/D
2. Result held fixed for one period by D/A

The response of the ZOH to a unit impulse at $t = 0$ is shown below, The output can be represented



as a unit step at $t = 0$ followed by a step of magnitude -1 occurring at $t = T$, as shown. The Laplace transform of the output can be written

$$Output(s) = \frac{1}{s} - \frac{e^{-Ts}}{s} = \frac{1 - e^{-Ts}}{s}. \tag{4.7}$$

Since the input has Laplace transform of 1 (unit impulse), the ratio of the Laplace transform of the output over that of the input yields the transfer function of the zero-order hold, which is

$$H_o(s) = \frac{1 - e^{-sT}}{s} \tag{4.8}$$

We will use the transfer function of the ZOH, or D/A, given by equation (4.8) extensively in modeling sampled data systems.

4.3 Frequency Spectra of Sampled Signals—Aliasing

We have a method of mathematically representing continuous functions $r(t)$ which have been sampled—this is the impulse sampling model given by equation (4.2). The question posed in this section is—what is the frequency spectrum of sampled signal $r^*(t)$? We will first find a Fourier series representation of the impulse train, use this in the expression for $r^*(t)$, take the Laplace transform of $r^*(t)$, which is $R^*(s)$, then use the fact that $R^*(j\omega) = R^*(s)|_{s=j\omega}$.

4.3.1 Fourier transform and Fourier series

The Fourier transform and Fourier series will be necessary in the derivation to follow, we state them below, first the Fourier transform:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{4.9}$$

then the Fourier series (only applicable if $f(t)$ is periodic):

$$f(t) = \sum_{n=-\infty}^{\infty} C_n e^{j(2\pi n/T)t} \tag{4.10}$$

where the Fourier coefficients are given by

$$C_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)e^{-j(2\pi n/T)t}dt \tag{4.11}$$

In (4.10) and (4.11) T is the period of $f(t)$. The double-sided expressions are given here, since the mathematics is more straightforward than if single-sided were used.

Now, since the complete impulse train

$$\sum_{k=-\infty}^{\infty} \delta(t - kT)$$

is periodic with period T, it can be represented by a **Fourier Series**;

$$\sum_{k=-\infty}^{\infty} \delta(t-kT) = \sum_{n=-\infty}^{\infty} C_n e^{j(2\pi n/T)t} \tag{4.12}$$

where the Fourier coefficients are given by

$$C_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \sum_{k=-\infty}^{\infty} \delta(t-kT) e^{-j(2\pi n/T)t}. \tag{4.13}$$

Although equation (4.13) looks formidable, the "sieving" property of the impulse function (and the only impulse within the integration limits is at $t=0$) yields a very simple result,

$$\begin{aligned} C_n &= \frac{1}{T}\left[e^{-j(2\pi n/T)(0)}\right] \\ &= \frac{1}{T} \end{aligned} \tag{4.14}$$

Substituting the Fourier coefficients C_n from equation (4.14) into the Fourier series of (4.12), so

$$\begin{aligned} \sum_{k=-\infty}^{\infty} \delta(t-kT) &= \sum_{n=-\infty}^{\infty} \frac{1}{T} e^{j(2\pi n/T)t} \\ &= \frac{1}{T}\sum_{n=-\infty}^{\infty} e^{jn\omega_s t} \end{aligned} \tag{4.15}$$

where we have used the relation that sampling frequency $\omega_s = 2\pi/T$. Using the Fourier series expression for the impulse train, we can express sampled signal $r^*(t)$ as

$$r^*(t) = \frac{1}{T}\left\{ r(t) \sum_{n=-\infty}^{\infty} e^{jn\omega_s t} \right\} \tag{4.16}$$

The Laplace transform $R^*(s)$ of sampled signal $r^*(t)$ is

$$\begin{aligned} \mathcal{L}[r^*(t)] &= \int_{-\infty}^{\infty} \frac{1}{T}\left\{ r(t) \sum_{n=-\infty}^{\infty} e^{jn\omega_s t} \right\} e^{-st} dt \\ R^*(s) &= \frac{1}{T}\sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} r(t) e^{jn\omega_s t} e^{-st} dt \\ &= \frac{1}{T}\sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} r(t) e^{-(s-jn\omega_s)t} dt \\ &= \frac{1}{T}\sum_{n=-\infty}^{\infty} R(s-jn\omega_s) \end{aligned} \tag{4.17}$$

If $F(s)$ is the Laplace transform of $f(t)$, the Fourier transform $F(j\omega)$ is given by

$$F(j\omega) = F(s)|_{s=j\omega} \tag{4.18}$$

thus

$$R^*(j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} R(j\omega - jn\omega_s) \tag{4.19}$$

Equation (4.19) is a momentous result, it tells us that two things happen to the frequency spectrum of $r(t)$ when it is sampled to become $r^*(t)$:

1. The magnitude of the sampled spectrum is $\frac{1}{T}$ that of the continuous spectrum, it is *scaled* by $\frac{1}{T}$.
2. The summation in (4.19) indicates that there are an infinite number of repeated spectra in the sampled signal, and they are repeated every $\omega_s = 2\pi/T$ along the frequency axis.

Two Examples of Sampled Spectra

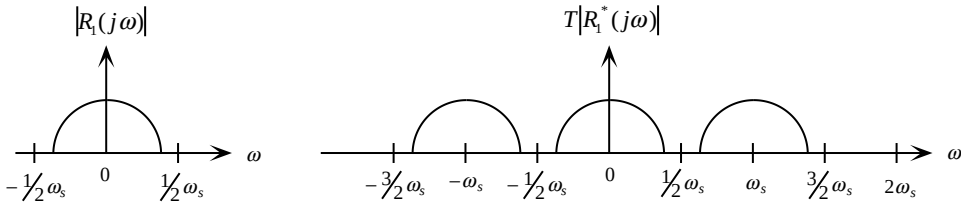
We will examine the effects of sampling on the amplitude spectra of two continuous functions.

Example 1. Consider continuous function $r_1(t)$ which has no frequency content above half the sampling frequency $\omega_s/2$. The original amplitude spectrum $|R_1(j\omega)|$ and the sampled spectrum $|R_1^*(j\omega)|$ are shown in Figure 4.3(a). The spectrum of the sampled $r_1^*(t)$ is scaled by $1/T$ and repeated, but one can see the *possibility* of reconstructing by extracting the original single spectrum from the array of repeated spectra (Remember that all information present in $r(t)$ is also present in $R(j\omega)$, the original spectrum).

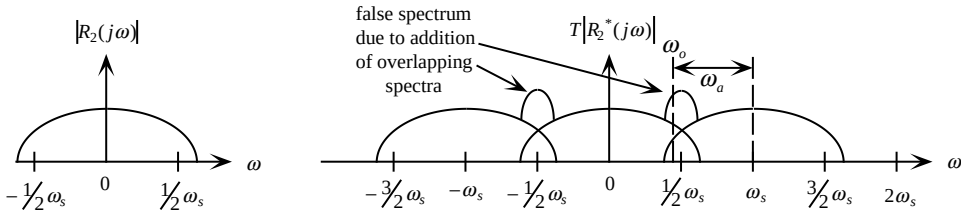
Example 2. Now consider function $r_2(t)$ which *has* frequency content above half the sampling frequency $\omega_s/2$. The same scaling and repetition occurs in the sampled spectrum, but this time due to the addition of the overlapping spectra in Figure 4.3(b) there is *no chance* of recovering the original spectrum after sampling.

Another phenomenon is shown in Figure 4.3(b), that of *aliasing*, illustrated at the frequency marked ω_o. The magnitude of the overall spectrum at ω_o is due to two different spectra, the "original" spectrum $R(j\omega_o)$, and the repeated spectrum centered at $\omega_s = 2\pi/T$. The value of the magnitude from the repeated spectrum is $R(j\omega_a)$, where $\omega_a = \omega_o - \omega_s$. This frequency, ω_a, which shows up at ω_o after sampling, is called an *alias* of ω_o.

Aliasing means that two different sinusoids have identical samples, and we cannot distinguish between them from their samples. When experimental data are to be sampled, it is essential that an "anti-aliasing" analog filter be used *before* sampling to filter out frequencies above one-half the sampling frequency (called the *Nyquist* frequency). Otherwise, frequencies above the Nyquist frequency in



(a) Non-overlapping spectra.



(b) Overlapping spectra showing contamination.

Figure 4.3: Frequency spectra of two signals.

the continuous signal will erroneously appear as lower frequencies after sampling. This is shown in Figure 4.4, where $f_o = \frac{7}{8}$ Hz, and sampling frequency $f_s = 1$ Hz. Then f_o appears at

$$f_a = \frac{7}{8} - 1 = -\frac{1}{8} \tag{4.20}$$

The significance of the negative frequency is that the $\frac{1}{8}$-Hz sinusoid in Figure 4.4 is the negative of the original $\frac{7}{8}$-Hz sinusoid.

Another way of looking at aliasing is that the act of sampling causes a "reflecting" of frequencies around the Nyquist frequency ($f_s/2$); refer to the dashed lines in Figure 4.8 where $f_s = 500$ Hz, and a component at 100 Hz will also show up at 400 Hz, 600 Hz, 900 Hz, 1100 Hz, and so on.

Sampling Theorem

This can be viewed as:

> What is the highest frequency we can allow in $r(t)$ so that aliasing will not occur?

From inspection of Figure 4.3 it is clear that if $R(j\omega)$ has components above $\omega_s/2$ or π/T, then spectral overlap and aliasing will occur. The sampling theorem is

> To recover a signal from its samples, you must sample *at least twice* the highest frequency in the original signal. If $R(j\omega) = 0$ for $\omega \geq \frac{\omega_s}{2}$, there will be no aliasing, and the original signal may be recovered from $R^*(j\omega)$, the Fourier transform of the samples (but not in real time!).

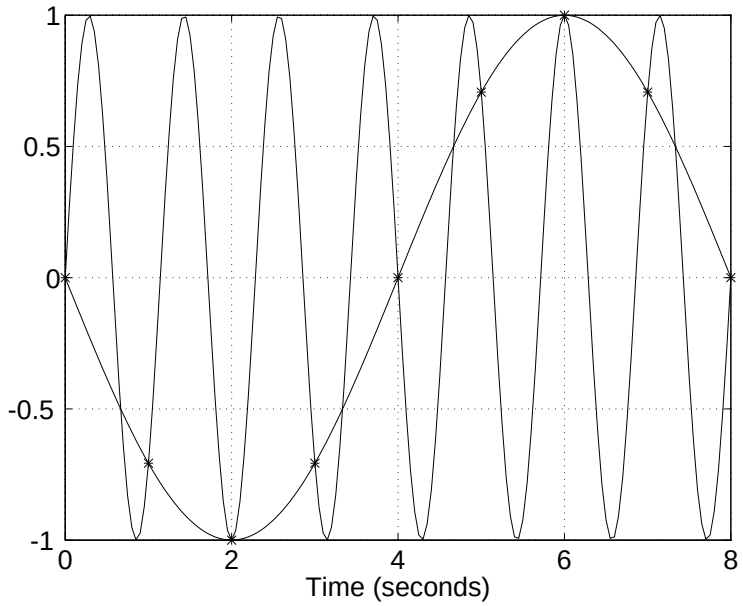


Figure 4.4: Plot of two sinusoids which have identical values at the sampling intervals: an example of aliasing.

4.4 Desampling or Signal Reconstruction

This is the reverse operation of sampling. Given the samples, how can we reconstruct a continuous signal? The samping theorem says that under the right conditions (no aliasing) it is possible to recover the original signal from the samples, although I have indicated this is impossible in real time. This section presents one method for such a reconstruction.

We will use an *ideal filter*. Consider recovering $R(j\omega)$ from $R^*(j\omega)$. This may be done by passing $R^*(j\omega)$ through an ideal low pass filter $L(j\omega)$, with magnitude

$$|L(j\omega)| = T, \quad -\frac{\pi}{T} \leq \omega \leq \frac{\pi}{T} \tag{4.21}$$

$$|L(j\omega)| = 0, \quad |\omega| > \frac{\pi}{T} \tag{4.22}$$

The filter amplitude in equation (4.21) will pass the original spectrum, while that in equation (4.22) will reject the repeated spectra. This is shown in Figure 4.5.

4.4.1 Impulse response of the Ideal Desampling Filter

The ideal sampling filter $L(j\omega)$ can recover $R(j\omega)$ from $R^*(j\omega)$ using

$$R(j\omega) = L(j\omega)R^*(j\omega) \tag{4.23}$$

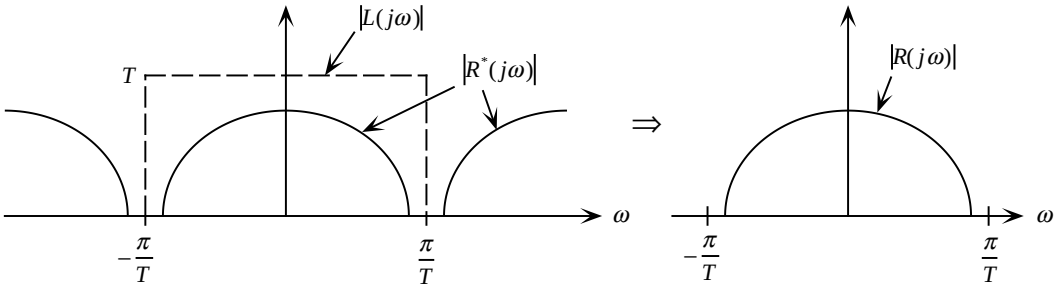


Figure 4.5: Use of ideal desampling filter.

If we take the inverse Fourier transform of $L(j\omega)$ we will obtain the impulse response. The inverse Fourier transform is given by

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega \tag{4.24}$$

and may be used to obtain a time function from a frequency function. Just like the Laplace transform, the inverse Fourier transform of a transfer function yields the impulse response.

Since $L(j\omega)$ is zero outside $\pm\pi/T$, the filter's impulse response is

$$\begin{aligned} l(t) &= \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} Te^{j\omega t}d\omega \\ &= \frac{T}{2\pi}\frac{e^{j\omega t}}{jt}\Big|_{-\pi/T}^{\pi/T} \\ &= \frac{T}{j2\pi t}\left(e^{j\pi t/T} - e^{-j\pi t/T}\right) \\ &= \frac{1}{\pi t/T}\sin\frac{\pi t}{T} \\ &= \operatorname{sinc}\frac{\pi t}{T} \end{aligned} \tag{4.25}$$

A sketch of the $\operatorname{sinc}\frac{\pi t}{T}$ function is shown in Figure 4.6. Since this is the response due to an impulse applied at $t = 0$, the ideal desampling filter is non-causal, because its response begins before it has received the input. This means that we cannot apply the filter in real time.

Disregarding the real time problem at the moment, the ideal desampling filter would be used as given by equation(4.23). Since multiplication in the transform domain is equivalent to convolution in the time domain, the desampled time function $r(t)$ is given by the convolution of $r^*(t)$ with the filter impulse response $l(t)$,

$$r(t) = \int_{-\infty}^{\infty} r^*(\tau)l(t-\tau)d\tau \tag{4.26}$$

Substituting the impulse train model for $r^*(t)$ and $l(t)$ from equation (4.25),

$$r(t) = \int_{-\infty}^{\infty} r(\tau)\sum_{k=-\infty}^{\infty}\delta(t-kT)\operatorname{sinc}\frac{\pi(t-\tau)}{T}d\tau \tag{4.27}$$

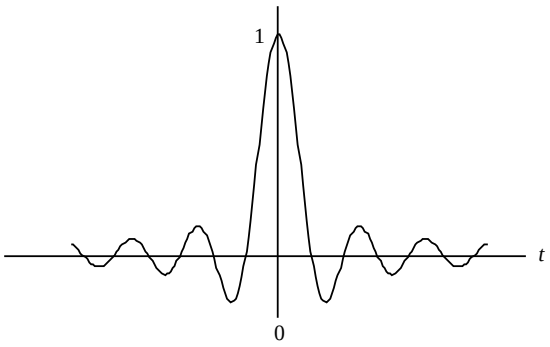


Figure 4.6: Sinc function... impulse response of ideal desampling filter.

or, using the "sieving" property of the $\delta(t-a)$,

$$r(t) = \sum_{k=-\infty}^{\infty} r(kT)\text{sinc}\frac{\pi(t-kT)}{T} \tag{4.28}$$

Equation (4.28) shows how to reconstruct the (band-limited... no aliasing!) function $r(t)$ from its samples $r^*(t)$. The sinc() function fills in the gaps between samples.

4.4.2 The ZOH as a Desampling Filter

The ideal desampling filter, being non-causal, cannot be used in real time, so we will examine the behavior of a zero-order hold, which is exactly the behavior of a D/A converter. Recall the transfer function of a ZOH:

$$H_o(s) = \frac{1-e^{-sT}}{s} \tag{4.29}$$

The frequency behavior of $H_o(s)$ is $H_o(j\omega)$,

$$H_o(j\omega) = \frac{1-e^{-j\omega T}}{j\omega} \tag{4.30}$$

Multiplying numerator and denominator of (4.30) by $e^{j\omega T/2}$, the magnitude of (4.30) is

$$\begin{aligned} |H(j\omega)| &= \left|\frac{e^{j\omega T/2}-e^{-j\omega T/2}}{j\omega e^{j\omega T/2}}\right| \\ &= T\left|\frac{\sin\omega T/2}{\omega T/2}\right| \\ &= T\left|\text{sinc}\frac{\omega T}{2}\right| \end{aligned} \tag{4.31}$$

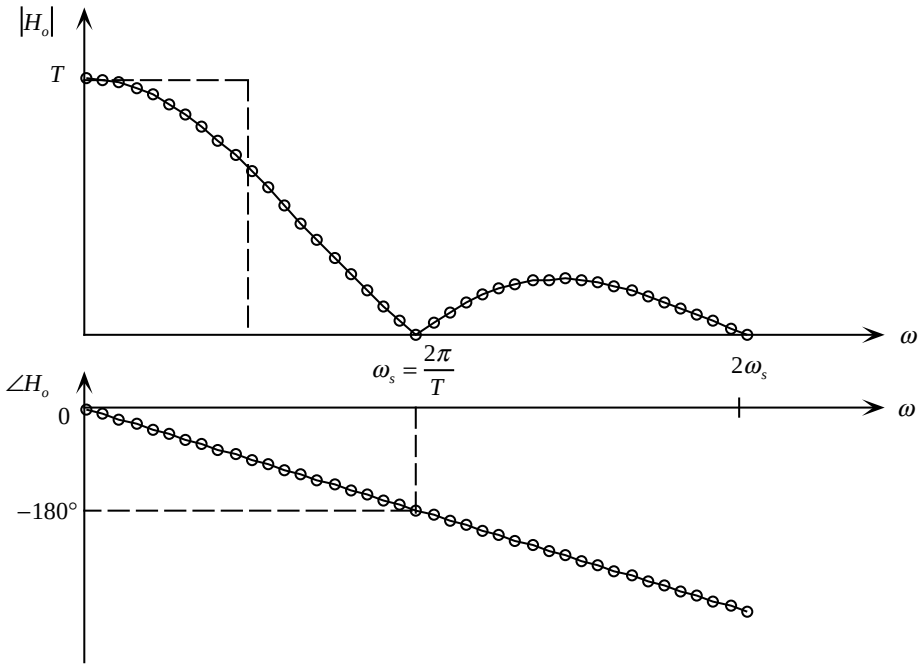


Figure 4.7: Frequency response of zero-order hold.

and the angle of (4.30) is

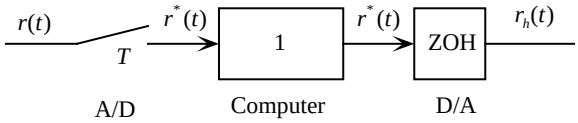
$$\begin{aligned} \angle H_o(j\omega) &= \angle \frac{e^{j\omega T/2} - e^{-j\omega T/2}}{j\omega e^{j\omega T/2}} \\ &= \angle \frac{2\sin \omega T/2}{\omega e^{j\omega T/2}} \\ &= -\frac{\omega T}{2} \end{aligned} \tag{4.32}$$

A plot of the frequency characteristics of the ZOH are shown in Figure 4.7. The ZOH is a low-pass filter, at least an approximation of the ideal desampling filter, and has linear phase lag with frequency. This phase lag can be viewed as the destabilizing effect of information loss at low sampling frequencies. The DC magnitude of T of the ZOH compensates for the frequency scaling of $1/T$ incurred by sampling.

Aliasing Lab Demonstration

The block diagram below shows a laboratory setup with an A/D converter, computer program which passes the numbers "straight through," and D/A converter (or ZOH) at the output. Variables

$r(t)$, $r^*(t)$ represent the continuous and sampled inputs, while $r_h(t)$ is the desampled output. The sampling frequency will be 500 Hz, or $T = 0.002$. A sinusoidal function generator will be used to



provide the input, so

$$r(t) = r_o \sin \omega t$$

and the frequency spectrum $R(j\omega)$ will be a single line at ω. The frequency spectrum of $r_h(t)$ is

$$R_h(j\omega) = R^*(j\omega) H_o(j\omega)$$

with magnitude

$$|R_h(j\omega)| = \frac{1}{T} \sum_{n=-\infty}^{\infty} |R(j\omega - jn\omega_s)| T \left| \text{sinc} \frac{\omega T}{2} \right|$$

The process of sampling will cause the repeated spectra shown in Figure 4.3, but their magnitude will be attenuated by the amplitude ratio of the ZOH shown in Figure 4.7.

The spectral amplitudes of the continuous input and the sampled and reconstructed output will be displayed on a spectrum analyzer. The expected amplitudes at several frequencies are shown in the accompanying table. Signals at these frequencies will be attenuated accordingly.

f (Hz)	ω	$\omega T/2$	sinc$(\omega T/2)$	db
50	314	0.314	0.984	-0.14
100	628	0.628	0.936	-0.58
150	942	0.942	0.858	-1.33
200	1257	1.257	0.757	-2.42
300	1885	1.885	0.505	-5.94

The magnitude of the ZOH frequency response at a sampling frequency of 500 Hz is shown in the plot. If the input is a sinusoid at 100 Hz, there will repeated spectra at 400, 600, 900, 1100... Hz. The discrete spectral lines are shown as solid lines, while their attenuated versions (which are what we actually see in $R_h(j\omega)$ are shown by dashed lines.

4.5 Block Diagram Analysis

In this section methods to reduce block diagrams of sampled data systems will be covered. Unlike continuous systems, sampled data systems contain samplers (A/D converters) and discrete elements (computer programs). In terms of continuous component connections, there are really only two possibilities:

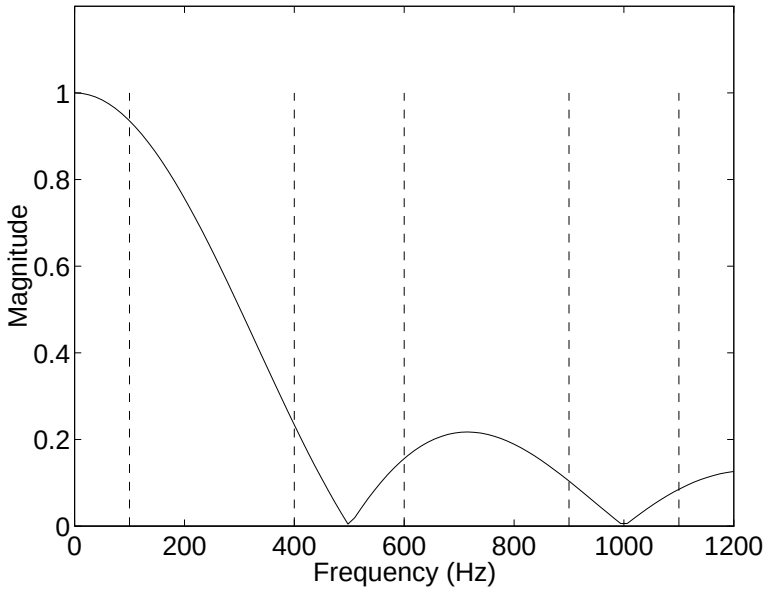
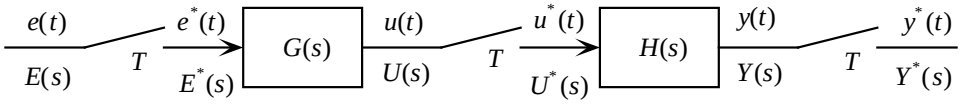


Figure 4.8: Repeated spectra and attenuation by ZOH.

1. Two continuous blocks *with* a sampler between them.
2. Two continuous blocks *without* a sampler between them.

4.5.1 Two blocks with a sampler between them

Consider the block diagram fragment shown below, with blocks $G(s)$ and $H(s)$ connected by a sampler. Variables $e(t)$, $e^*(t)$, $u(t)$, $u^*(t)$, $y(t)$, $y^*(t)$, and their respective Laplace transforms $E(s)$, $E^*(s)$, $U(s)$, $U^*(s)$, $Y(s)$, $Y^*(s)$ are given. The continuous output $Y(s)$ is given by



$$Y(s) = U^*(s)H(s) \tag{4.33}$$

We propose to show that the sampled output $Y^*(s)$ is given by

$$Y^*(s) = (U^*(s)H(s))^* = U^*(s)H^*(s) \tag{4.34}$$

Using the frequency domain expression for the Laplace transform of a sampled signal given by equation (4.19). Recall that in the frequency domain, the act of sampling scales the spectrum by

$1/T$ and causes infinite repeated duplicate spectra at ω_s intervals. Hence $Y^*(s)$ is just $Y(s)$ sampled, or

$$\begin{aligned} Y^*(s) &= \{U^*(s)H(s)\}^* \\ &= \frac{1}{T}\sum_{n=-\infty}^{\infty} U^*(s-jn\omega_s)H(s-jn\omega_s) \end{aligned} \tag{4.35}$$

and

$$U^*(s) = \frac{1}{T}\sum_{k=-\infty}^{\infty} U(s-jk\omega_s) \tag{4.36}$$

so

$$U^*(s-jn\omega_s) = \frac{1}{T}\sum_{k=-\infty}^{\infty} U(s-jk\omega_s-jn\omega_s) \tag{4.37}$$

Let $k = l - n$, or $k + n = l$, then

$$\begin{aligned} U^*(s-jn\omega_s) &= \frac{1}{T}\sum_{l=-\infty}^{\infty} U(s-jl\omega_s) \\ &= U^*(s) \end{aligned} \tag{4.38}$$

The spectrum of sampled variable U^* can be shifted an integral number of periods ω_s because its spectrum is periodic with this "period."

We can make use of this property in equation (4.35), substituting (4.38), so

$$\begin{aligned} Y^*(s) &= \frac{1}{T}\sum_{n=-\infty}^{\infty} U^*(s-jn\omega_s)H(s-jn\omega_s) \\ &= U^*(s)\frac{1}{T}\sum_{n=-\infty}^{\infty} H(s-jn\omega_s) \\ &= U^*(s)H^*(s) \end{aligned} \tag{4.39}$$

The result we were after was $Y^*(s) = U^*(s)H^*(s)$. This could have also been applied to block $G(s)$ and input $E^*(s)$ to yield

$$U^*(s) = E^*(s)G^*(s) \tag{4.40}$$

Relations (4.39) and (4.40) can be combined to yield

$$Y^*(s) = E^*(s)G^*(s)H^*(s) = E^*G^*H^* \tag{4.41}$$

The point in equation (4.41) is that blocks G and H "come apart," we have G^*H^*, not $(GH)^*$. This is due to the sampler between them.

The "*" transform and z-transform It is worthwhile to distinguish the "*" transform and the z-transform. Using equation (4.6) we can find the $Y(z)$ from $Y^*(s)$ by

$$Y(z) = Y^*(s)|_{e^{sT}=z} \tag{4.42}$$

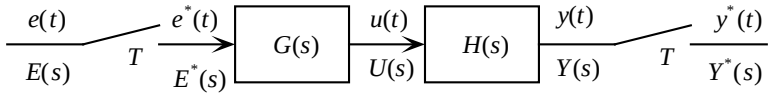
It is important to understand the inverse of both $Y^*(s)$ and $Y(z)$:

$$\begin{aligned} \mathcal{L}^{-1}\left[U^*(s)\right] &\Rightarrow \text{ sequence of impulses with intensities } u(kT) \\ \mathcal{Z}^{-1}\left[U(z)\right] &\Rightarrow \text{ sequence of values } u_k \end{aligned} \tag{4.43}$$

In most of the transform-based design work to follow, we will use $Y(z)$ rather than $Y^*(s)$ because it is more concise.

4.5.2 Two blocks without a sampler between them

Without going into detail, using a similar derivation (but somewhat simpler) as above, it can be shown that for the block connection below that $Y^*(s) \neq U^*(s)H^*(s)$, but rather



$$Y^*(s) = \{UH\}^*(s) \tag{4.44}$$

Likewise,

$$U(s) = E^*(s)G(s) \tag{4.45}$$

Equations (4.44) and (4.45) can be combined to yield

$$Y^*(s) = \{E^*(s)G(s)H(s)\}^* = E^*(s)\{G(s)H(s)\}^* = E^*\{GH\}^* \tag{4.46}$$

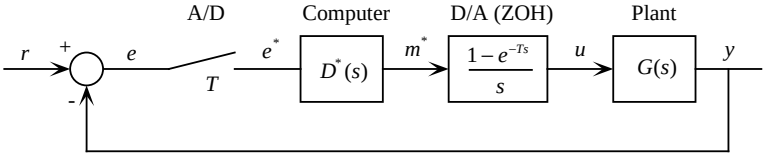
Comparing (4.41) and (4.46) it can be seen that without a sampler between G and H they do not "come apart."

Block Diagram Reduction Procedure

In reducting block diagrams of sampled-data systems, the basic principle is

> Solve for the variables which are sampled!

You will usually get simpler algebraic reduction. In most cases in this course we want transfer functions relating sampled output to sampled input, rather than continuous variables. Now for some examples...



Example 1

In the block diagram shown below, the error signal $e(t)$ is sampled by the A/D converter, there is a computer program modeled as discrete transfer function $D^*(s)$ (note that it could have been, and will soon be modeled as $D(z)$), D/A converter modeled as a ZOH, and finally a continuous plant $G(s)$. In every case the "*" notation signifies a sampled variable with the impulse modulation sampling model.

The error signal is sampled, so start with that...

$$E(s) = R(s) - Y(s) \tag{4.47}$$
$$M^*(s) = E^*(s)D^*(s) \tag{4.48}$$
$$U(s) = M^*(s)\left[\frac{1-e^{-Ts}}{s}\right] \tag{4.49}$$
$$Y(s) = G(s)U(s) \tag{4.50}$$

We'd like to relate the samples of the output $Y*(s)$ to samples of the input $R^*(s)$. We can simply "star" each transform,

$$E^* = R^* - Y^* \tag{4.51}$$
$$M^* = E^*D^* \tag{4.52}$$
$$U^* = M^* \text{ (since samples of } u \text{ and } m \text{ are the same)} \tag{4.53}$$
$$Y^* = [GU]^* \tag{4.54}$$

Equations (4.49) and (4.54) give us

$$Y^* = \left[GM^*\left(\frac{1-e^{-Ts}}{s}\right)\right]^* \tag{4.55}$$

where we should remember that the "star" operation is really the act of impulse-modulating, then Laplace transforming.

Anyway, in equation (4.55) both M^* and $(1-e^{-Ts})$ may be taken outside the "*" since they're already impulse sampled (an easy way to check this is to see that they both have periodic spectra).

Therefore,

$$\begin{aligned} Y^* &= (1-e^{-Ts})M^*\left(\frac{G}{s}\right)^* \\ &= (1-e^{-Ts})E^*D^*\left(\frac{G}{s}\right)^* \\ &= (1-e^{-Ts})D^*\left(\frac{G}{s}\right)^*[R^*-Y^*] \end{aligned} \tag{4.56}$$

Make the definition

$$(1-e^{-Ts})\left(\frac{G}{s}\right)^* = G^* \tag{4.57}$$

then it follows that

$$Y^* = D^*G^*[R^*-Y^*] \tag{4.58}$$

and we get a discrete system transfer function

$$\frac{Y^*}{R^*} = \frac{D^*G^*}{1+D^*G^*} \tag{4.59}$$

which can also be written as

$$\frac{Y(z)}{R(z)} = \frac{D(z)G(z)}{1+D(z)G(z)} \tag{4.60}$$

This block diagram is a common one, and will serve to model many actual systems. Make sure you understand this derivation.

Example 2

Consider the same block diagram as before, and let

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

a first order plant with time constant of 1 second. Let sampling period $T = 0.5$ seconds.

Let the computer program be a backward rectangular integrator,

$$m_k = m_{k-1} + Ke_k$$

and let the D/A be modeled as a ZOH, of course.

Computer. Applying the z-transform to the computer difference equation, we get

$$M(z) = z^{-1}M(z) + KE(z)$$

therefore

$$\frac{M(z)}{E(z)} = D(z) = \frac{K}{1-z^{-1}} = \frac{Kz}{z-1}$$

Plant and ZOH. First consider $(G/s)^*$, where

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

and

$$\left[\frac{G(s)}{s}\right]^* = \mathcal{Z}\left\{\frac{G(s)}{s}\right\} = \frac{z(1-e^{-T})}{(z-1)(z-e^{-T})} = \frac{0.3935z}{(z-1)(z-0.6065)}$$

Using equation (4.57),

$$G^* = (1-e^{-Ts})\left(\frac{G}{s}\right)^* \Rightarrow G(z) = (1-z^{-1})\mathcal{Z}\left\{\frac{G(s)}{s}\right\} = \frac{z-1}{z}\mathcal{Z}\left\{\frac{G(s)}{s}\right\}$$

So

$$G(z) = \frac{z-1}{z}\frac{0.3935z}{(z-1)(z-0.6065)} = \frac{0.3935}{z-0.6065}$$

and

$$D(z)G(z) = \frac{0.3935Kz}{(z-1)(z-0.6065)}$$

Closed-loop system. The closed-loop transfer function is

$$\frac{Y(z)}{R(z)} = \frac{D(z)G(z)}{1+D(z)G(z)} = \frac{0.3935Kz}{(z-1)(z-0.6065)+0.3935Kz} = \frac{0.3935Kz}{z^2+(0.3935K-1.6065)z+0.6065}$$

The roots of the denominator polynomial (closed-loop poles) depend on the value of gain K that is chosen. Is this system conditionally stable?

4.5.3 Response between samples

It is usually not necessary to find the response between samples, but it is possible. Starting from equation (4.50), one can show that

$$Y(s) = R^*\frac{D^*}{1+D^*G^*}\frac{1-e^{-Ts}}{s}G(s) \tag{4.61}$$

The "starred" terms in (4.61) are all composed of impulses, and $1-e^{-Ts}$, which is a delay. The only thing left is $G(s)/s$, which is what the impulses act on. But $G(s)/s$ is the step response of the plant $G(s)$! Thus between samples we will see the (scaled and biased) plant step response.

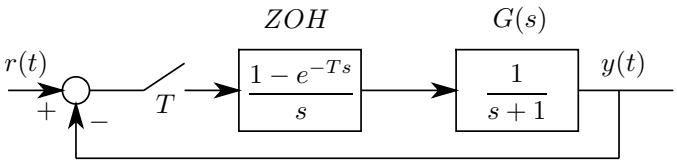
Homework Problems

Problem 1.

Using MATLAB, verify the Fourier series expansion of the impulse train given in equation (4.15) by plotting a few periods of the Fourier series for several numbers of terms, such as 3,10, 100, *etc.*

Problem 2.

Sketch the step response $y(t)$ of the system shown below for the three samples $k = 0, 1, 2$. Use sample period $T = 1$.



Problem 3.

Assume the plant transfer functions below are preceded by a zero-order hold. Compute the equivalent discrete transfer function $G(z)$. Use sampling period $T = 1$. I would like you to find the $G(z)$ by hand, then compare your results to MATLAB. **Note:** This is the "zero-order hold" discrete simulation method that we *didn't* cover in Chapter 3.

(a) $G(s) = \dfrac{1}{s^2}$

(b) $G(s) = \dfrac{1}{s(s+1)}$

(c) $G(s) = \dfrac{1}{s^2 - 1}$

Problem 4.

Using a MATLAB program, verify the ideal reconstruction method given in equation (4.28). For your sampled data, use 6 samples ($k = 0...5$) of a 1 Hz sine wave with $T = 0.2$. Compute a full period of the reconstructed continuous signal at a much finer time step, such as 0.01 second. Compare the reconstructed signal using only 6 samples with the original continuous signal. It really works!

Chapter 5

Design Using Transform Methods

5.1 Introduction

In this chapter we will discuss digital control system design using transform-based methods. As with continuous controllers, the goal of design is to transform the behavior from unacceptable to acceptable. Specific design specifications are, of course, different from application to application. We will take an application of controlling the azimuth angle of a large antenna, and use this application to illustrate some basic performance issues:

1. Steady-state accuracy.
2. Transient response.
3. Disturbance rejection
4. Control effort and gain distribution
5. Sensitivity to parameter changes

A typical control system application will involve all of these issues. We nearly always want adequate steady-state accuracy; transient response is also usually important. Disturbance rejection must be addressed if the plant is subject to disturbance inputs during operation. Control effort and gain distribution must be analyzed to preclude component saturation, and gain distribution specifically takes on greater importance in digital controllers, with A/D and D/A resolution involved. Finally, component parameters will change, a thorough design will examine the system behavior with these changes.

Pole-placement design will be used in this chapter, in which the designer chooses the locations of the system closed-loop poles, then designs a controller to place them there. The *root locus* method is a pole-placement method for transform-based design. The mathematics of the root locus method

are independent of complex domain, thus the technique is the same whether used in the s-plane or z-plane. The reader should review the coverage of the root-locus method in an introductory controls textbook.

Frequency-domain methods (Bode and Nyquist plots), which are frequently used for the design of continuous control systems, require some adaptation when used with digital control systems, and will not be discussed in this chapter.

There are two strategies for pole-placement transform-based design, which are:

- Design in the s-plane by "discrete equivalent"
- Direct Design in the z-plane.

In the first method, we design a continuous controller in the s-plane, using methods from continuous system modeling and analysis, obtaining some compensator $G_c(s)$. To implement this in a digital control system, use one of the methods of Chapter 3 to get a discrete equivalent $G_c(z)$, then use this in the digital control system. Note that this method *never* considers the effect of sampling on the plant. This will cause the resulting system to be in error. The extent of this error depends mainly on how fast you sample. If sampling is "fast" there will be less error; if sampling is slower—more error.

In the second method, "direct" design in the z-plane, we get a discrete domain model of the plant dynamics. Then the controller design is done in the sampled domain, the z-plane, from the outset. In this case the design is "guaranteed" to work, since we have considered the effect of sampling on the plant.

5.2 Example System and Specifications

Consider the control of the azimuth pointing angle of a large antenna, such as a radio telescope. This can be considered angular position control of a large rotary inertia. The actuator will be a DC electric motor. The "plant" will be taken as the combination of actuator and antenna. The equation of motion of the plant is

$$J\ddot{\theta} + B\dot{\theta} = T + T_w \tag{5.1}$$

where

$$\begin{aligned} \theta &= \text{antenna angle (controlled variable} \\ J &= \text{total rotary inertia} \\ B &= \text{total viscous friction (includes back emf)} \\ T &= \text{motor torque} \\ T_w &= \text{wind (disturbance) torque} \end{aligned}$$

A block diagram of the plant is shown in Figure 5.1. Assume that $J/B = 10$ seconds. This is a

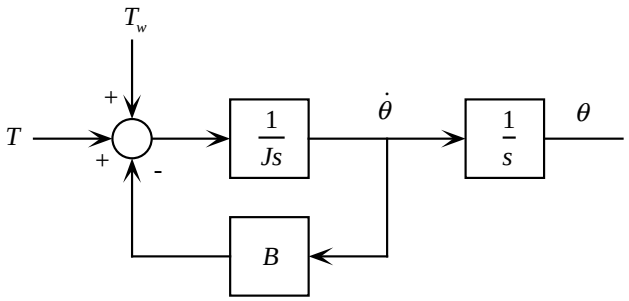


Figure 5.1: Block diagram of antenna plant.

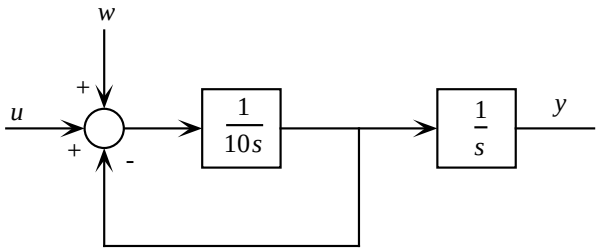


Figure 5.2: Another block diagram of antenna plant.

mechanical time constant, and if the antenna were given an initial velocity, it would decay to 63% of this velocity in one time constant, or 10 seconds. With this assumption, the equation of motion can be written

$$\frac{J}{B}\ddot{\theta} + \dot{\theta} = \frac{T}{B} + \frac{T_w}{B} \tag{5.2}$$

or

$$10\ddot{\theta} + \dot{\theta} = \frac{T}{B} + \frac{T_w}{B} \tag{5.3}$$

Defining new variables as

$$u \triangleq \frac{T}{B} \tag{5.4}$$

$$w \triangleq \frac{T_w}{B} \tag{5.5}$$

$$y \triangleq \theta \tag{5.6}$$

we can redraw the block diagram as shown in Figure 5.2

which has transfer functions

$$\frac{Y(s)}{U(s)} = \frac{1}{s(10s+1)} = \frac{0.1}{s(s+0.1)} \tag{5.7}$$

$$\frac{Y(s)}{W(s)} = \frac{0.1}{s(s+0.1)} \tag{5.8}$$

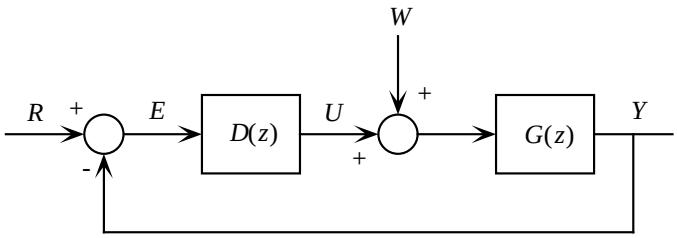


Figure 5.3: Block diagram for ss error analysis.

Our controller for this plant will accept a reference position (set point) θ_r, and will compute control force T such that the tracking error $e = \theta_r - \theta$ satisfies specifications.

5.2.1 Steady-State Accuracy

For this antenna control system, the steady-state accuracy specification is given as follows:

> For reference input $\theta_r(t) = 0.01t$ (this is 34^o/minute) after transients die out, steady-state error $\theta_r - \theta < 0.01$ rad.

In general, tracking error comes from two sources: changes in the reference input and the onset of disturbances. As you will recall from your continuous control course, system **Type** determines steady-state behavior,

Type 0. Finite steady-state error when setpoint is a constant position (step input).

Type 1. Finite steady-state error when setpoint is a constant velocity (ramp input), and zero error when setpoint is constant position.

Type 2. Finite steady-state error when setpoint is a constant acceleration (parabolic input), and zero error when setpoint is constant velocity or position.

Note that "position, velocity, and acceleration" refer to the rates of change of the setpoint variable, not the variable itself (the setpoint could be a temperature, for example).

To compute the steady-state error, consider the system block diagram shown in Figure 5.3. This block diagram is in sampled form; all variables and transfer functions are functions of z. We will let the disturbance w be zero, it is shown in Figure 5.3 for generality. The system equations are

$$E(z) = R(z) - G(z)D(z)E(z) \tag{5.9}$$

so

$$E(z) = \frac{R(z)}{1 + G(z)D(z)} \tag{5.10}$$

Step input, position error doefficient. Let $r(t)$ be a step of magnitude r_o, thus

$$R(z) = \frac{r_o z}{z-1} \tag{5.11}$$

so the error is

$$E(z) = \frac{r_o z}{z-1}\frac{1}{1+G(z)D(z)} \tag{5.12}$$

Now apply the final value theorem to $E(z)$:

$$\begin{aligned} e(\infty) &= \lim_{z\to 1}(z-1)\frac{r_o z}{z-1}\frac{1}{1+G(z)D(z)} \\ &= \frac{r_o}{1+G(1)D(1)} \qquad (5.13) \\ &\triangleq \frac{r_o}{1+K_p} \qquad (5.14) \end{aligned}$$

and the *position error coefficient* is given by

$$K_p \triangleq G(1)D(1) \tag{5.15}$$

and the steady-state error is given by equation (5.14).

The position error coefficient K_p varies with system Type as:

Type 0: $G(z)D(z)$ has no poles at $z = 1$, K_p is finite.

Type 1: $G(z)D(z)$ has one pole at $z = 1$, $K_p = \infty$.

Type 2: $G(z)D(z)$ has two poles at $z = 1$, $K_p = \infty$.

Ramp input, velocity error coefficient. Here consider a ramp input of slope v_o, given by

$$R(z) = \frac{v_o T z}{(z-1)^2} \tag{5.16}$$

then the error is

$$E(z) = \frac{v_o T z}{(z-1)^2}\frac{1}{1+G(z)D(z)} \tag{5.17}$$

The steady-state error is

$$e(\infty) = \lim_{z\to 1}\frac{v_o T z}{(z-1)[1+G(z)D(z)]} \triangleq \frac{v_o}{K_v} \tag{5.18}$$

thus

$$\begin{aligned} K_v &= \lim_{z\to 1}\frac{(z-1)[1+G(z)D(z)]}{Tz} \\ &= \lim_{z\to 1}\frac{(z-1)G(z)D(z)}{Tz} \qquad (5.19) \end{aligned}$$

The velocity error coefficient K_v varies with system Type as:

Type 0: $G(z)D(z)$ has no poles at $z = 1$, K_v is zero.

Type 1: $G(z)D(z)$ has one pole at $z = 1$, K_v is finite.

Type 2: $G(z)D(z)$ has two poles at $z = 1$, $K_v = \infty$.

A similar development can be done for the acceleration error coefficient K_a. Type greater than 2 is extremely difficult to stabilize, therefore rarely seen.

For the antenna servo, we can tolerate a steady-state error of 0.01 rad to a ramp input of slope 0.01. Thus the velocity error coefficient should be used.

Since $\frac{1}{K_v} = \frac{\text{ss error}}{\text{ramp slope}} = \frac{0.01}{0.01}$, then $K_v \geq 1$ is necessary. This requirement must be met when the controller is designed.

Continuous-domain error coefficients.

Recall that in the continuous domain, the error is defined exactly the same way in terms of the three error coefficients, and the error coefficients are given by

$$K_p = \lim_{s\to 0} G(s)D(s) = G(0)D(0) \tag{5.20}$$

$$K_v = \lim_{s\to 0} sG(s)D(s) \tag{5.21}$$

$$K_a = \lim_{s\to 0} s^2G(s)D(s) \tag{5.22}$$

where $G(s)$ is the plant and $D(s)$ is a compensator. As with digital systems, these error coefficients can be 0, finite, or ∞.

5.2.2 Transient Response

The goal here is maintain good dynamic error behavior. The specification on transient response may be given in either the time or frequency domain, but given our preference for the root locus pole placement technique, a time domain specification is appropriate.

Three specifications of the step response will be presented, *percent overshoot*, *rise time*, and *settling time*. We will consider these in the time domain, getting corresponding s plane regions, then map these to the z plane for design purposes. These will all be with reference to a continuous second-order underdamped system with no zeros, which has transfer function

$$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2}. \tag{5.23}$$

An example step response plot for this kind of system is shown in Figure 5.4 (with $\zeta = 0.5$ and $\omega_n = 1$, where the percent overshoot, rise time, and settling time are marked on the plot.

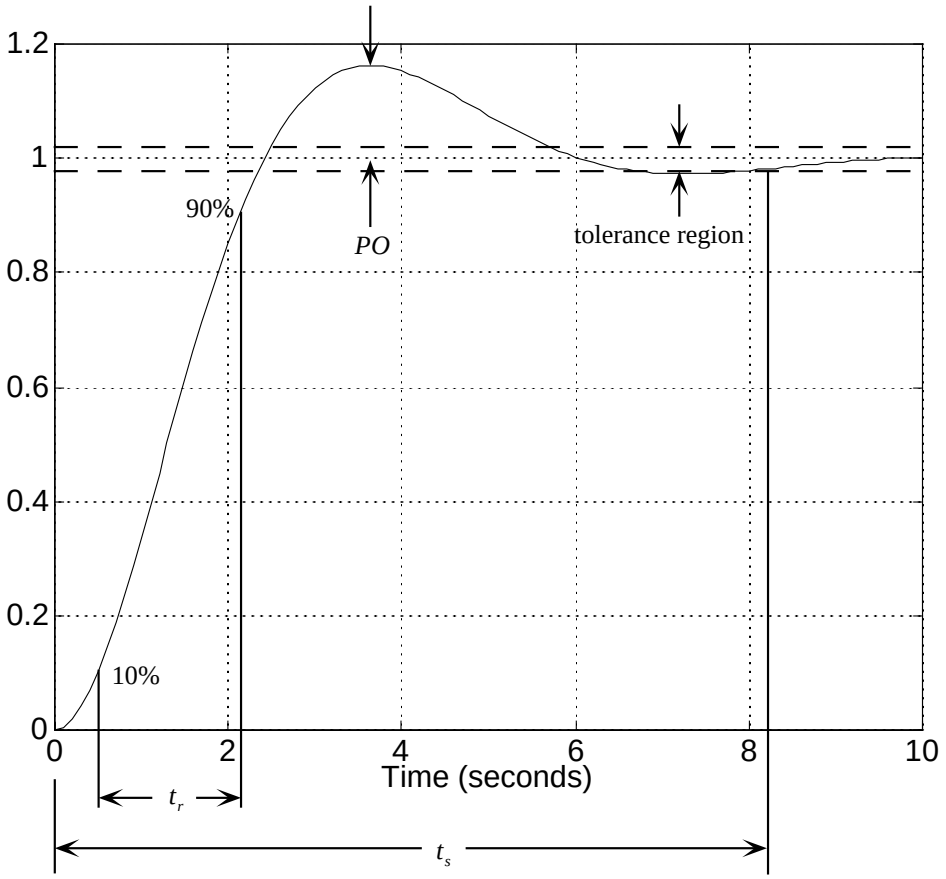


Figure 5.4: Transient response specifications, showing percent overshoot PO, rise time t_r, and settling time t_s

Percent Overshoot. The percent overshoot PO is dependent only on damping ratio, and is given (in percent) by

$$PO = e^{-\zeta\pi/\sqrt{1-\zeta^2}} * 100 \tag{5.24}$$

$$\approx \left(1 - \frac{\zeta}{0.6}\right) * 100 \tag{5.25}$$

Taken as a limitation on damping ratio, if percent overshoot must be less than or equal to a value PO, then an approximate result is

$$\zeta \geq 0.6\left(1 - \frac{PO}{100}\right). \tag{5.26}$$

Rise Time. The rise time t_r is usually taken to be the time from the 10% to the 90% point of the step response, and as such indicates the speed of the response. While there is no exact analytical expression for the rise time, it is strongly dependent on natural frequency ω_n, and a commonly-used approximation is

$$t_r \approx \frac{1.8}{\omega_n} \tag{5.27}$$

which is most accurate when damping ratio $\zeta \approx 0.5$.

Expressed in terms of a desired rise time less than a given amount, the natural frequency must obey

$$\omega_n \geq \frac{1.8}{t_r}. \tag{5.28}$$

Settling Time. The settling time t_s is the time elapsed until the response is guaranteed to stay within a desired tolerance of the steady-state value, expressed as a percentage.

The expression for the unit step response (2nd order underdamped, no zeros) is

$$y(t) = 1 - e^{-\zeta\omega_n t}\cos\left(\omega_d t + \phi\right) \tag{5.29}$$

where damped oscillating frequency ω_d is

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{5.30}$$

The error transient is contained within the envelope $e^{-\zeta\omega_n t}$, and we can require $\zeta\omega_n$ large enough so that the error decreases as fast as necessary.

For example, if we want the error to be less than 1% after t_s seconds, then

$$e^{-\zeta\omega_n t_s} \leq 0.01 \tag{5.31}$$

and taking the natural log of both sides,

$$\zeta\omega_n t_s \geq 4.6 \tag{5.32}$$

thus

$$\zeta\omega_n \geq \frac{4.6}{t_s} \tag{5.33}$$

Allowable regions in the s plane. These three transient response specifications can be used to describe allowable regions in the s plane for the system poles, again considering an underdamped second-order system.

- **Percent overshoot.** PO is dependent only on damping ratio ζ, and lines of constant ζ are radial lines in the s plane.
- **Rise time.** t_r is dependent mainly on natural frequency ω_n, and contours of constant ω_n are circles in the s plane.
- **Settling time.** t_s is dependent on $\zeta\omega_n$, which is the real part of a complex pole location. Lines of constant $\zeta\omega_n$ are vertical lines in the s plane.

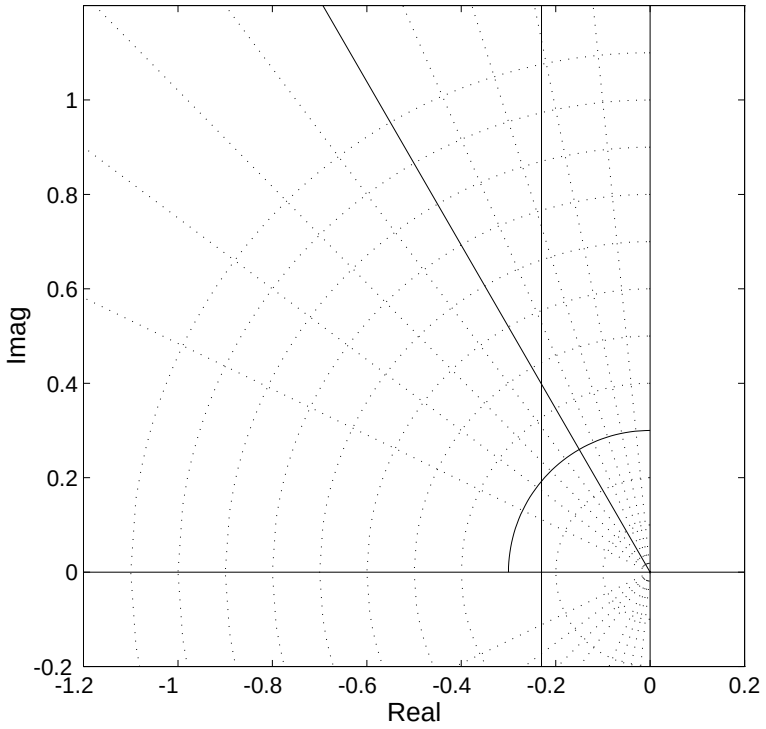


Figure 5.5: Regions in the s-plane corresponding to restrictions on PO, t_r, and t_s.

Example

For the antenna controller, assume that we are given the following transient response specifications:

Percent overshoot: $PO \leq 15\% \Rightarrow \zeta \geq 0.5$.

Rise time: $t_r \leq 6$ sec $\Rightarrow \omega_n \geq \frac{1.8}{6} = 0.3$.

Settling time: $1\% t_s \leq 20$ sec $\Rightarrow \zeta\omega_n \geq 0.23$.

Regions in the s-plane corresponding to these specifications are shown in Figure 5.5. The radial lines shown are lines of constant damping ratio from $\zeta = 0.9, 0.8, 0.7 \ldots 0.1$.

Allowable regions in the z plane. The s plane regions shown in Figure 5.5 can be mapped to the z plane using the pole mapping $z = e^{sT}$. They are shown in the z-plane drawn in Figure 5.6. Recall the discussion in Section 2.5... lines of constant damping ratio ζ are logarithmic spirals; lines of constant natural frequency ω_n are nearly circular near $z = 1$, but are increasingly distorted as ω_n

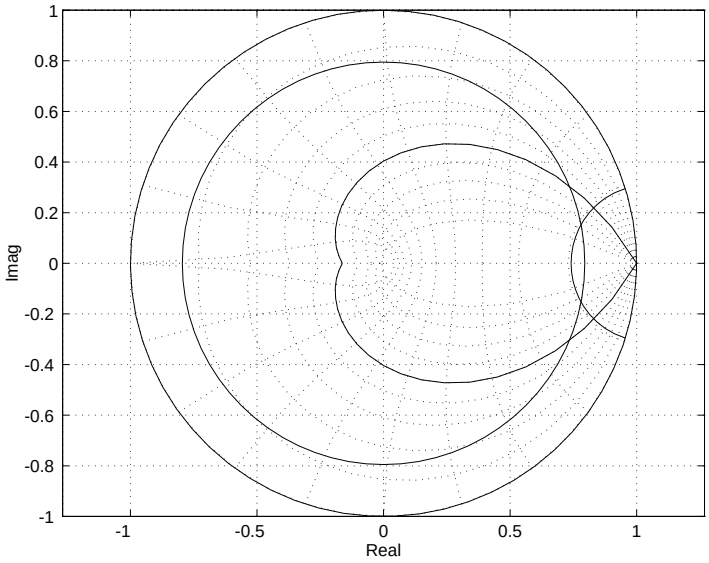


Figure 5.6: Regions in the z-plane corresponding to restrictions on $PO \leq 15\%$, $t_r \leq 6$ sec, and $t_s \leq 20$ sec. (Sampling period $T = 1$ seconds was chosen to obtain the t_r and t_s regions)

increases and the contours move to the left; and lines of constant "real part" or $\zeta\omega_n$ are contours of constant radius. (*NOTE:* to draw the ω_n and t_s contours one must select a sampling period T).

When designing a control system, to achieve the specified transient response we must have the dominant poles within the allowable regions of Figure 5.6. Because clear dominance may not occur, the final design should always be checked with a simulation to verify acceptable response.

5.2.3 Disturbance Rejection

Consider the block diagram of 5.3 with reference input $R(z) = 0$, and $W(z)$ is a disturbance input. Since $R(z) = 0$, then

$$E(z) = -[G(z)D(z)E(z) + G(z)W(z)] \tag{5.34}$$

which can be rearranged

$$E(z) = \frac{G(z)W(z)}{1 + G(z)D(z)} \tag{5.35}$$

Assume that the loop gain $|G(z)D(z)| >> 1$, then

$$E(z) \approx \frac{W(z)}{D(z)} \tag{5.36}$$

From equation(5.36) we conclude that the effect of a disturbance is reduced by having a high gain *preceding* the entry point of the disturbance, in this case in block $D(z)$. A high gain in $G(z)$ would not greatly affect the disturbance rejection.

Note that if $D(z)$ were to contain an integration (pole at $z = 1$) this would cause infinite gain at zero frequency, therefore zero frequency (DC) disturbances would be rejected completely, within the limits of the actuators, of course.

5.2.4 Control Effort and Gain Distribution

In the course of a design, one must take care that components are not saturated. Using a worst-case scenario, commands to amplifiers, actuators, *etc.* must be checked. This is done by redrawing the block diagram to expose the variable of interest as the output, apply the appropriate input (either setpoint or perhaps a disturbance) and obtain the response. This will be shown in future design examples.

The *gain distribution* goes along with component saturation. A given design may call for a forward path gain of K. This gain of K may be "placed" all in one component, such as an amplifier, or it may be distributed among several components, the computer algorithm $D(z)$, an op-amp signal conditioning amplifier, and a power amplifier. Gain should be distributed according to the following criteria:

- Minimize the likelihood of component saturation.
- Make use of the full range of A/D and D/A converters.

The second item is unique to digital control systems. If a 12-bit A/D converter with an input signal range of $\pm$5V receives an input which varies between $\pm$1V, the effective resolution of the A/D converter is reduced to less than 10 bits, since each bit corresponds to a factor of 2.

5.2.5 Parameter Sensitivity

One advantage of closed-loop control is reduced system sensitivity to changes in parameter values within the loop. The larger the loop gain the lower the sensitivity. The principle may be stated without analysis as:

> Large gain around a parameter reduces the effect of changes in its value.

5.3 Design in the s plane by Discrete Equivalent

This is the first design method, and consists of the following steps:

1. Design controller $D(s)$, ignoring the presence of sampling.
2. Convert $D(s)$ to discrete $D(z)$ using one of the simulation methods of Chapter 3.

We will perform a third step,

3. Simulate the resulting discrete system to check performance.

This last step, of course, will require a discrete-domain plant model, and one of the chief reasons one would do design in the continuous domain is because you *don't have* a discrete plant model. However, we'll do it for educational reasons!

Antenna Azimuth Controller. Recall the antenna azimuth control system discussed in Section 5.2, whose transfer function was

$$G(s) = \frac{0.1}{s(s+0.1)} \tag{5.37}$$

The performance specifications for the transient response were

Percent overshoot: $PO \leq 15\% \Rightarrow \zeta \geq 0.5$.

Rise time: $t_r \leq 6$ sec $\Rightarrow \omega_n \geq \frac{1.8}{6} = 0.3$.

Settling time: $1\%t_s \leq 20$ sec $\Rightarrow \zeta\omega_n \geq 0.23$.

A lead compensator $D(s)$ of the form

$$D(s) = \frac{K(s+0.1)}{s+0.5} \tag{5.38}$$

will satisfy these requirements, since the zero of $D(s)$ will cancel the plant pole at $s = -0.1$ and will substitute a pole at $s = -0.5$ which will result in higher natural frequency. Using the block diagram of Figure 5.3, the continuous-domain closed-loop transfer function of the system is

$$\frac{Y(s)}{R(s)} = \frac{G(s)D(s)}{1 + G(s)D(s)} \tag{5.39}$$

The characteristic equation of this system is

$$1 + G(s)D(s) = 0 \tag{5.40}$$

and using the proposed $D(s)$ we have

$$G(s)D(s) = \frac{0.1K}{s(s+0.5)} \tag{5.41}$$

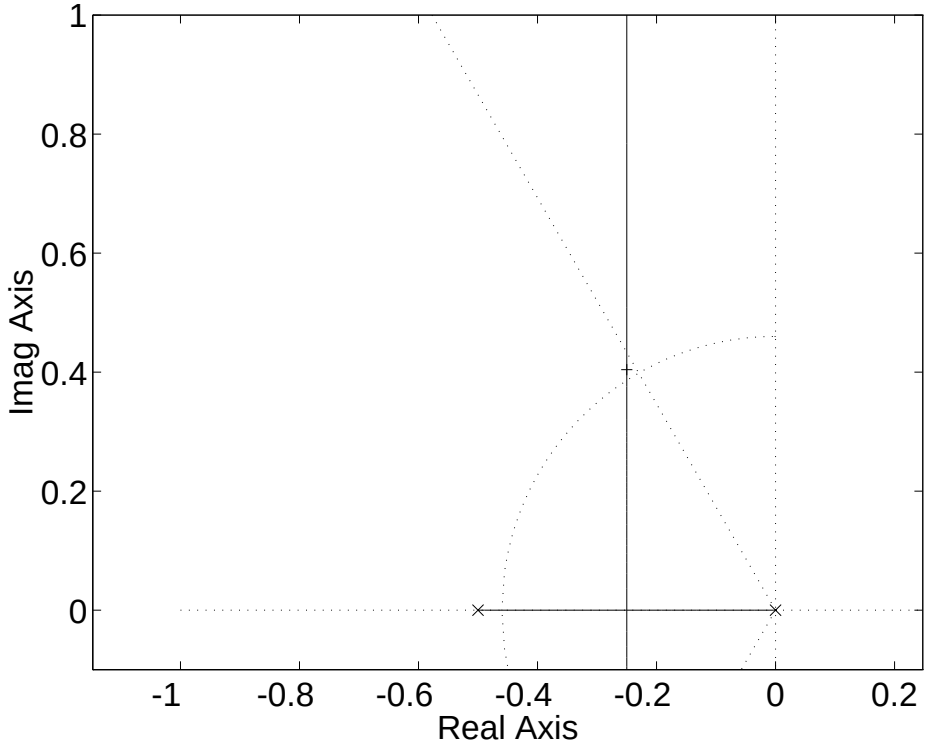


Figure 5.7: Root Locus of the antenna plant with $D(s)$ compensator.

thus the characteristic equation in root locus form is

$$1 + K\frac{0.1}{s(s+0.5)} = 0 \tag{5.42}$$

The root locus diagram *vs* K is shown in Figure 5.7. A radial line at $\zeta = 0.5$ and a circular arc corresponding to $\omega_n = 0.46$ are shown, these are the boundaries defining the acceptable region for transient response. The root locus branch is within the acceptable region, and a value of $K = 2.26$ corresponds to the closed-loop pole shown in Figure 5.7. For this value of K the continuous domain closed-loop transfer function is

$$\frac{Y(s)}{R(s)} = \frac{0.226}{s^2 + 0.5s + 0.226} \tag{5.43}$$

and the damping ratio and natural frequency of the denominator of equation (5.43) are $\zeta = 0.53$ and $\omega_n = 0.48$, which are satisfactory. The unit step response of the antenna servo (completely in the continuous time domain) is shown in Figure 5.8. The transient response of the continuous system satisfies the specifications. In addition, to check the steady-state behavior, compute the velocity

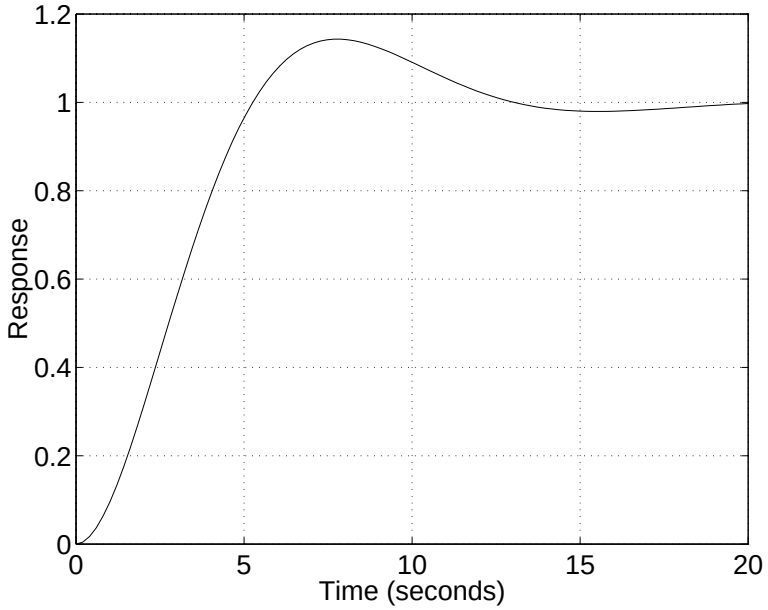


Figure 5.8: Unit Step Response of the Continuous Antenna Controller Design

error coefficient, which for a continuous system is

$$K_v = \lim_{s \to 0} sG(s)D(s) = \frac{0.226}{0.5} = 0.45 \tag{5.44}$$

which is less than the desired values of $K_v = 1$. So this design would not satisfy the steady-state requirements, and a re-design would be necessary. However, neglect that for now.

Performance of the Discrete System

Let's evaluate the transient performance of the sampled data system with the compensator we just obtained. Note that it did satisfy the transient performance requirements.

First, the $D(s)$ is

$$D(s) = \frac{2.26(s + 0.1)}{s + 0.5} \tag{5.45}$$

which is simulated to yield a $D(z)$. A sample period must be chosen, and we shall use $T = 2$ seconds. This corresponds to $f_s = 0.5$ Hz, and is about 6 times the system natural frequency.

Discrete compensator. Using trapezoidal integration, $D(z)$ is

$$D(z) = \frac{1.6562(z - 0.8182)}{z - 0.3333} \tag{5.46}$$

Discrete model of plant. With $T = 2$ seconds, use the method of Chapter 4 to find the discrete transfer function of a plant $G(s)$ preceded by a zero-order hold. Thus

$$\begin{aligned} G(z) &= \left(1 - z^{-1}\right) \mathcal{Z}\left\{\frac{G(s)}{s}\right\} \\ &= \frac{0.1873(z + 0.9355)}{(z - 1)(z - 0.8187)} \end{aligned} \tag{5.47}$$

where the MATLAB `'c2d'` function is useful in finding $G(z)$, with the `'zoh'` method. Note that the pole/zero cancellation is maintained between $G(z)$ and $D(z)$ (the slight variation between pole and zero value is probably because I wrote down a number and thereby lost some significant digits). The closed-loop discrete domain transfer function for this system will be of the form

$$\frac{Y(z)}{R(z)} = \frac{G(z)D(z)}{1 + G(z)D(z)}$$

which yields

$$\frac{Y(z)}{R(z)} = \frac{0.3102(z + 0.9355)}{z^2 - 1.0231z + 0.6235} \tag{5.48}$$

which has poles at $z = 0.5115 \pm j0.6015$, for which $\zeta = 0.2631$ and $\omega_n = 0.4488$. While ω_n is pretty close, the actual damping ratio is significantly less than the desired damping ratio of $\zeta = 0.5$. This was a consequence of designing a discrete-time controller in the continuous domain. For reference, the unit step response of the actual discrete system is shown in Figure 5.9. Compare the two step responses of Figure 5.8 and Figure 5.9. The actual output $y(t)$ in Figure 5.9 would not have straight lines between samples, it would have the shape of the plant's step response, which is a ramp with an exponential start-up transient. The samples, however, are correct.

If we had sampled faster, the continuous-domain design would have performed better.

5.4 Direct Design in the z Plane

Here we will design in the discrete domain from the start. A discrete tranfer function of the antenna plant is given in equation (5.47). Even though we immediately tried a lead compensator in the previous section, we will start here with proportional control.

Proportional Control This is the first control action to try. All controllers contain proportional action, plus perhaps derivative and/or integral action. A discrete-domain block diagram of the system with proportional control is shown below. The closed-loop transfer function of the system

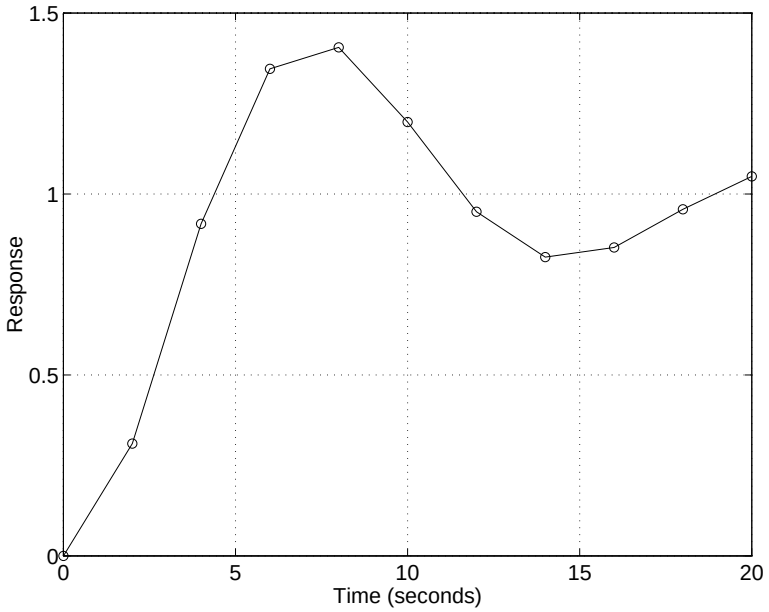


Figure 5.9: Unit Step Response of the Actual Discrete Antenna Controller Design using the $D(z)$ designed in the Continuous Domain.

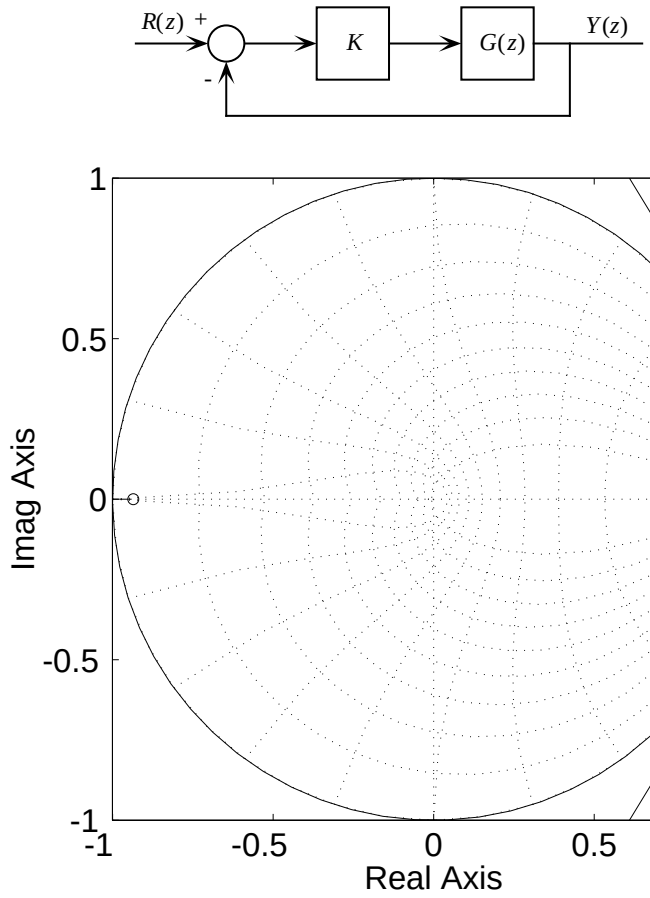


Figure 5.10: Root Locus of antenna system *vs* proportional gain K.

with proportional control is

$$\frac{Y(z)}{R(z)} = \frac{KG(z)}{1 + KG(z)} \tag{5.49}$$

with characteristic equation $1 + KG(z) = 0$, which is already in root locus form (always the case if the root locus parameter is a loop gain). The z plane root locus is shown in Figure 5.10. Checking Figure 5.6 for the acceptable regions for the closed-loop poles, it is apparent that proportional control will not work. We need to reshape the root locus, in particular shifting the branches to the left in the z-plane. This can be done by adding zeros and poles in the form of a compensator. Recall that zeros attract the locus, while poles repel it. Thus a compensator with a zero nearer the $+1$ point and a pole farther to the left will have the desired effect.

The compensator zero may be used to cancel the plant pole, the compensator pole placed far enough to the left to cause the branches to go through the desired region.

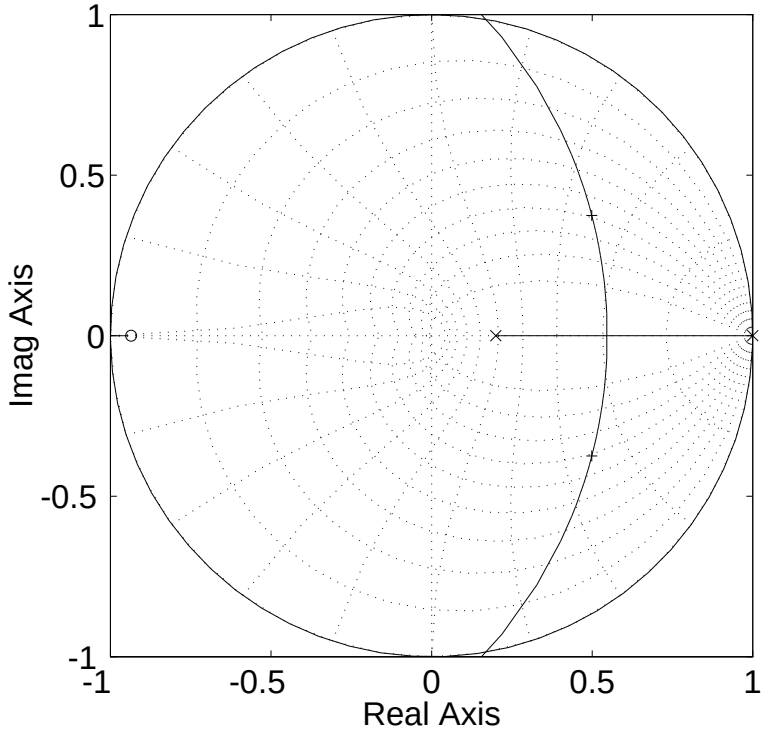


Figure 5.11: Root Locus of antenna system *vs* gain K using compensator $D(z)$.

A point in the z plane at $z = 0.5 \pm j0.37$ has $\zeta = 0.6$, $\omega_n = 0.4$, and $r = 0.62$, which are very close to meeting the overshoot, rise time, and settling time specifications... let's say close enough. Since the breakaway point is about halfway between both poles, and the complex branches curve to the left a little, place the compensator pole at $z = 0.2$, thus the compensator $D(z)$ is

$$D(z) = \frac{K(z - 0.8187)}{z - 0.2} \tag{5.50}$$

The root locus diagram for the system *vs* K with this compensator is shown in Figure 5.11. The point on the complex locus corresponds to $K = 1.08$, and appears to be in or near the satisfactory z plane region. With this compensator the system transfer function is

$$\frac{Y(z)}{R(z)} = \frac{0.2023(z + 0.9355)}{z + 0.4989 \pm j0.3747} \tag{5.51}$$

The unit step response of this system is shown in Figure 5.12. The transient response is what we expected from the design, unlike in the previous section.

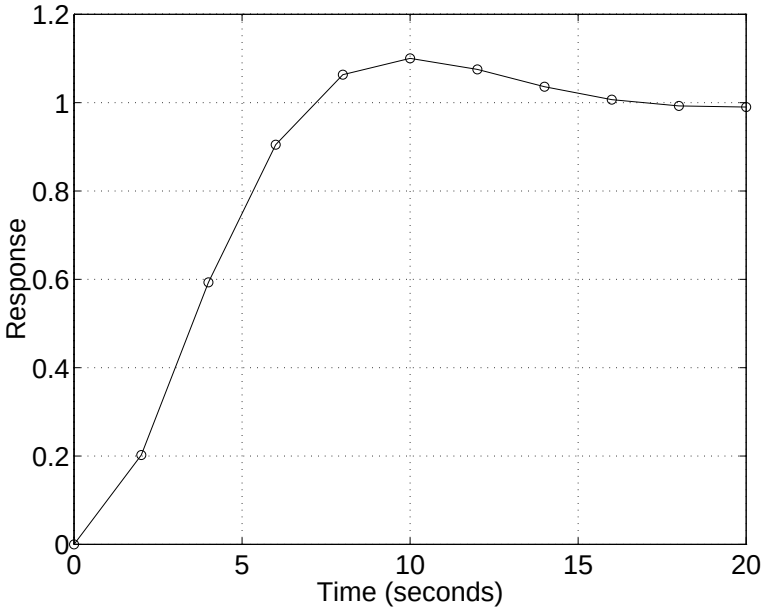


Figure 5.12: Step response of system designed in z plane.

Homework Problem Set 1

Problem 1.

Root-locus review. The following characteristic equations (denominators of closed-loop transfer functions set equal to zero) must first be put into root-locus form using parameter K. You may either sketch these by hand (not likely) or use MATLAB, but be sure you can find the real-axis branches, numbers of asymptotes, asymptote intersection, departure angles, *etc.* manually.

(a) The locus below is typical of the behavior near $s = 0$ of a double integrator with lead compensation or a single integration plus a real pole (DC motor from voltage to displacement, for example) with lag compensation. If done manually, pay attention to the real axis arrival and breakaway points.

$$s^2(s+4) + K(s+1) = 0$$

(b) This transfer function contains complex poles. If done manually, note the departure angles from the complex poles.

$$s(s+1)[(s+2)^2 + 4] + K = 0$$

(c) This locus shows the use of complex zeros to compensate for the presence of complex poles due to vibration modes. Note the departure and arrival angles from/to the complex roots.

$$s(s^2+4) + K[(s+1)^2 + 1] = 0$$

Problem 2.

Consider the attitude control of a satellite with rotary inertia $J = 1$. Assume you have proportional actuators (thrusters) which can apply a torque to the satellite, as well as a transducer (inertial gyro or IMU) which measures its attitude.

(a) Given that the continuous transfer function is $G(s) = 1/Js^2$, find the discrete transfer function $G(z)$ when the plant is preceded by a sampler and ZOH. Let sample period $T = 1$ second.

(b) Sketch the root locus of this system as a function of proportional gain K with unity feedback. What is the type of the uncompensated system?

(c) Add a lead network so that the dominant poles are at $\zeta = 0.5$ and $\theta = 45^o$ (this is an angle in the z-plane). Plot the closed-loop step response.

(d) Plot the open- and closed-loop frequency response for the system in part (c). Show the phase and gain margins.

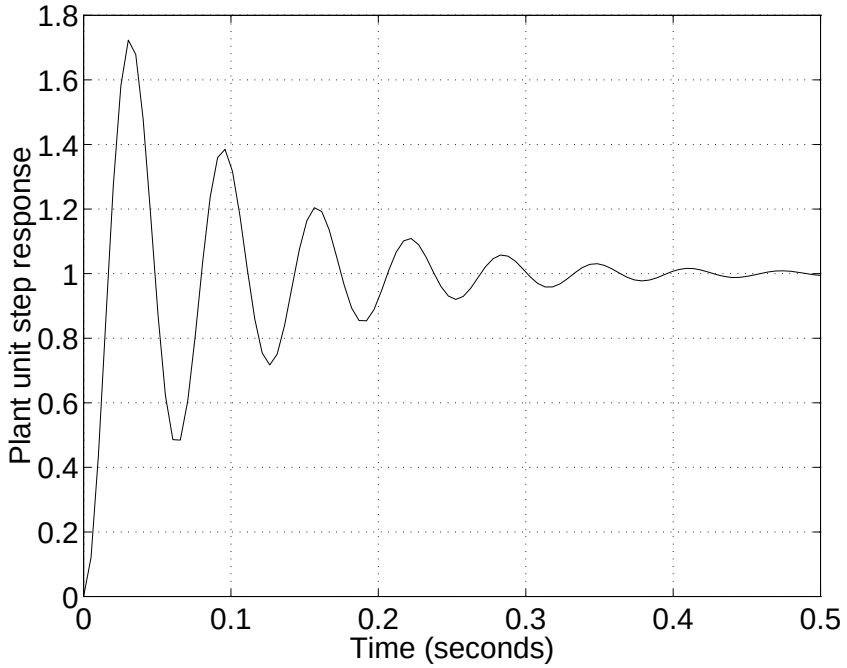


Figure 5.13: Unit step response of plant $G(s)$.

5.5 Another Design Example

Consider the second-order plant with $\zeta = 0.1$ and $\omega_n = 100$ rad/sec ($f_n \approx 16$ Hz):

$$G(s) = \frac{Y(s)}{U(s)} = \frac{100^2}{s^2 + 2(0.1)(100)s + 100^2} = \frac{10000}{s^2 + 20s + 10000} \tag{5.52}$$

This behavior would be encountered when trying to position an inertia through a transmission (shaft, gears, *etc.*) which has flexibility. The unit step response of this plant may be obtained by the following MATLAB script:

```
>>zeta = 0.1;
>>wn = 100;
>>num = [0 0 wn^2];
>>den = [1 2*zeta*wn wn^2];
>>step(num,den);
```

A plot of the response is shown in Figure 5.13. The slow-decaying oscillation, or "ringing," is apparent from the plot. Also, the plant $G(s)$ has no free integrations, and therefore will be Type 0. We propose to use a digital feedback control system to meet three performance criteria—

1. Increase damping so that PO <10%.
2. Achieve rise time of t_r =0.03 seconds (a little slower than the original plant).
3. Increase system Type to 1.

Note that we have not tried to maintain the rise time of the original system. It might be unrealistic to try to ask for too much, we are already attempting to improve both transient response *and* and steady-state error... trying to maintain the original speed of response may well be impossible!

Acceptable z-plane regions for poles The first step will be to find translate the transient response performance specifications into allowable regions in the z plane. The overshoot and rise time give:

- **Overshoot:** PO <10% $\Rightarrow \zeta > 0.54$.
- **Rise time:** $t_r < 0.03$ seconds $\Rightarrow \omega_n > \frac{1.8}{0.03} = 60$ rad/sec.

Sampling rate. To find the acceptable contours in the z plane, we must select a sample period T (needed to map $z = e^{sT}$. A rule of thumb is to pick the sampling frequency 6 to 20 times greater than the highest frequency in the system. The natural frequency of the plant $G(s)$ is about 16 Hz., so select $f_s = 100$ Hz as the sampling frequency, hence $T = 0.01$ seconds. This is on the slow side, especially if high-frequency disturbances are expected, but will yield satisfactory results for setpoint tracking. The z-plane regions corresponding to the limits on ζ and ω_n are shown in Figure 5.14. This figure was generated with the following MATLAB commands:

```
>>T = 0.01;
>>zeta = 0.54;
>>wn = 60;
>>zgrid(zeta,wn*T,'new');
>>grid;
>>axis('square');
>>axis('equal');
```

Block diagram of control system. A block diagram of the feedback control system is shown in Figure 5.14. As before, assume an error-sampled system, with a D/A converter following the digital controller. A unity feedback loop completes the system.

Discrete model of plant. Preceding the plant $G(s)$ by a sampler and following it with a ZOH yields a discrete plant model $G(z)$. The following MATLAB script (entered interactively) will do this:

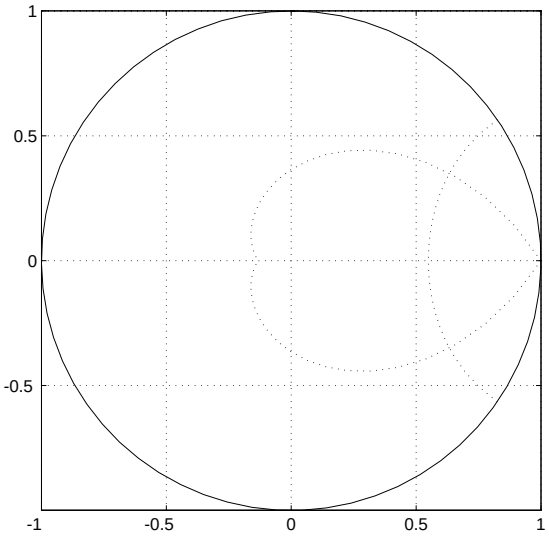
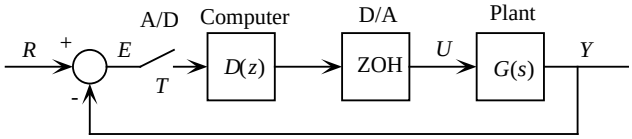


Figure 5.14: Acceptable regions for poles.




```
>> T = 0.01;
>> wn_plant = 100;
>> zeta_plant = 0.1;
>> num_gc = [0 0 wn_plant^2];
>> den_gc = [1 2*zeta_plant*wn_plant wn_plant^2];
>> Gc = tf(num_gc,den_gc);    % Continuous plant G(s)
>> Gd = c2d(Gc,T)    % Discrete plant G(z)

Transfer function:
   0.431 z + 0.4023
-----------------------
z^2 - 0.9854 z + 0.8187

Sampling time: 0.01
```

thus the discrete plant model is

$$G(z) = \frac{Y(z)}{U(z)} = \frac{0.4310z^{-1} + 0.4023z^{-2}}{1 - 0.9854z^{-1} + 0.8187z^{-2}} \tag{5.53}$$

This may expressed in MATLAB "pole/zero" form by using the `zpk` conversion on LTI system `Gd`:

```
>> zpk(Gd)

Zero/pole/gain:
   0.43103 (z+0.9334)
------------------------
(z^2  - 0.9854z + 0.8187)

Sampling time: 0.01
```

hence $G(z)$ may be expressed

$$G(z) = \frac{0.4310(z + 0.9334)}{z - 0.4927 \pm j0.7589}, \tag{5.54}$$

where I took the final step of factoring the denominator to get the poles explicitly in (5.54).

Equation (5.53) is useful in generating the difference equation for $G(z)$, while equation (5.54) explicitly shows the poles and zeros, and from the pole locations the damping ratio and natural frequency (or time constant for real poles) may be found, giving insight into the natural response of the plant.

Control strategy. Pure proportional will clearly not work, since it doesn't increase system Type. The controller $D(z)$ *must* add a pole at $z = 1$ to increase Type to 1. Adding integrations *always* hurts stability, and since this plant is lightly damped already, we will need to do more. Adding zeros in the forward path (or the feedback path) is adding *derivative action*, which stabilizes the system.

The obvious way to add zeros is to cancel undesirable plant dynamics, which is what we did in the antenna controller of the previous section. Here the undesirable plant dynamics are the lightly damped complex poles, hence we will place zeros of $D(z)$ at these locations. There must be a pole of $D(z)$ at $z = 1$. To be causal, compensator $D(z)$ must not have more zeros than poles, hence we must add another pole to $D(z)$. Note that poles tend to repel root locus branches, so this added pole will push the locus to the right. Placing this pole on the negative real axis will lessen its effect, but we must be careful about ending up with a *closed-loop* pole on the negative real axis, because of the "alternating" nature of the corresponding response. Compensator poles on the negative real axis will give the plant command a degree of this alternating response, but as long as it is not excessive it is not a cause for concern. So the form of our $D(z)$ is

$$D(z) = \frac{K(z - 0.4927 \pm j0.7589)}{(z - 1)(z - p)} \tag{5.55}$$

where pole location p and (of course) gain K need to be determined.

It is useful to sketch the root-locus diagram of the system by hand to get an idea of what is likely to happen, especially regarding the placement of the second compensator pole.

The system closed-loop transfer function is

$$\frac{Y(z)}{R(z)} = \frac{G(z)D(z)}{1 + G(z)D(z)} \tag{5.56}$$

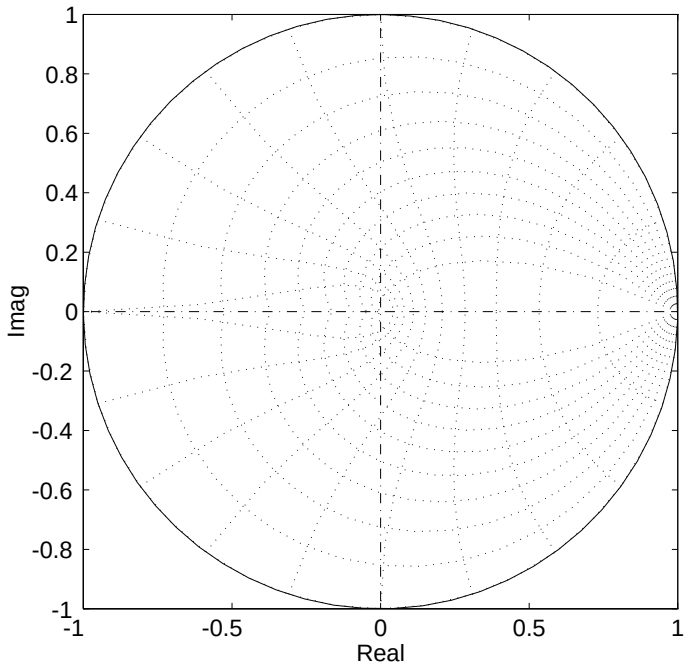


Figure 5.15: Hand sketch of root locus for Example 2.

and the system characteristic equation is

$$1 + G(z)D(z) = 0 \tag{5.57}$$

which upon substitution for $G(z)$ and $D(z)$ becomes

$$1 + \frac{0.4310K(z - 0.4927 \pm j0.7589)(z + 0.9334)}{(z - 0.4927 \pm j0.7589)(z - 1)(z - p)} \tag{5.58}$$

The complex poles and zeros may be algebraically cancelled, leaving

$$1 + \frac{0.4310K(z + 0.9334)}{(z - 1)(z - p)} \tag{5.59}$$

A hand sketch of this locus for $p = 0$ is shown in Figure 5.15. The canceled pole and zero are not shown in Figure 5.15, this simplifies the sketch.

Computer root locus analysis. Comparing the hand-drawn sketch of Figure 5.15 with the region of acceptibility in Figure 5.14, it looks like placing the second pole of $D(z)$ at $z = 0$ may work. A root locus diagram may be generated using MATLAB as follows:

```
>>num_rl = 0.4310*[1 0.9334];
>>den_rl = conv([1 -1],[1 0]);
>>zgrid('new');
>>rlocus(num_rl,den_rl);
```

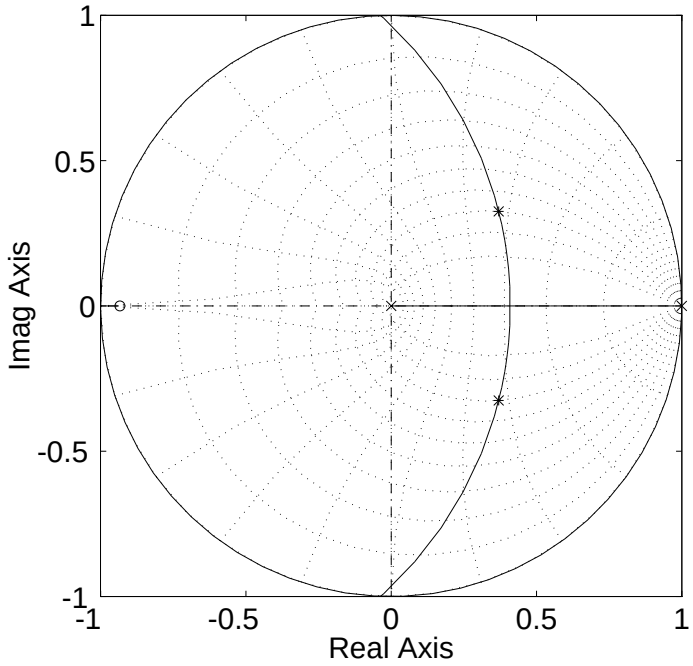


Figure 5.16: MATLAB root locus for Example 2.

The `zgrid('new')` function draws a z-plane, with lines of constant damping ratio ($\zeta = 0.1, 0.2, 0.3, \ldots, 0.8, 0.9$), and lines of constant natural frequency ($\omega_n T = 0.1\pi, 0.2\pi, \ldots, 0.8\pi, 0.9\pi$). These lines aid selection of desired pole locations, which is done using the `rlocfind` function as shown below, where I have selected a damping ratio of $\zeta = 0.7$.

```
>>K = rlocfind(num_rl,den_rl);
Select a point in the graphics window

selected_point =

   0.3707 + 0.3245i

>>K

K =
    0.6022
```

The selected closed-loop pole locations are shown by a "star" on the root locus of Figure 5.16.

Simulation of closed-loop system. We have selected a compensator $D(z)$ that will apparently work; it is given by

$$D(z) = \frac{0.6022(z - 0.4927 \pm j0.7589)}{z(z-1)} \tag{5.60}$$

The MATLAB LTI form of $D(z)$ may be obtained as follows:

```
>>zeros_dz = [0.4927+j*0.7589 0.4927-j*0.7589]'    % Zeros of D(z)

zeros_dz =

   0.4927 - 0.7589i
   0.4927 + 0.7589i

>>poles_dz = [0 1]'    % Poles of D(z)

poles_dz =

     0
     1

>>K_dz = 0.6022;    % Numerator constant for D(z)
>>[num_dz,den_dz] = zp2tf(zeros_dz,poles_dz,K_dz)

num_dz =

    0.6022   -0.5934    0.4930

den_dz =

     1    -1     0
>> Dd = tf(num_dz,den_dz,T)

Transfer function:
0.6022 z^2 - 0.5934 z + 0.493
-----------------------------
           z^2 - z

Sampling time: 0.01
```

so the polynomial form of the compensator $D(z)$ is (negative powers of z):

$$D(z) = \frac{U(z)}{E(z)} = \frac{0.6022 - 0.5934z^{-1} + 0.4930z^{-2}}{1 - z^{-1}}. \tag{5.61}$$

The transfer function of the closed-loop system can be found using MATLAB, in this case (unity feedback) using the `series` (combines two blocks in series) and the `feedback` (computes an LTI

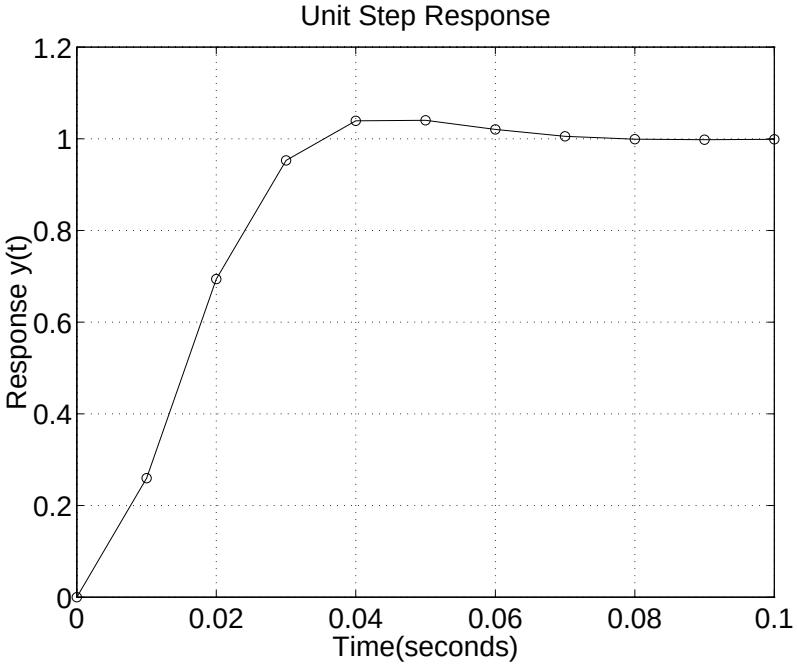


Figure 5.17: Unit step response of closed-loop system for Example 2.

model for a feedback loop) functions. We have both $D(z)$ and $G(z)$ in MATLAB LTI form, so:

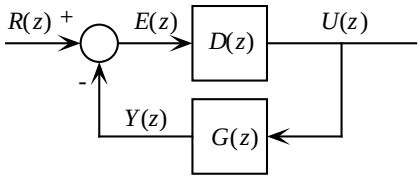
```
>> forward = series(Dd,Gd);    % Forward path is D(z)*G(z)
>> sys_cl = feedback(forward,1);    % Closed-loop, unity feedback
```

Note that MATLAB will *not* do pole/zero cancellation (look at `sys_cl` in MATLAB, for example), but they will cancel numerically during a response simulation.

Step response. The unit step response of the closed-loop system is shown in Figure 5.17. Compare this plot with the uncontrolled plant of Figure 5.13. The MATLAB commands that generated this response are

```
>>k = 0:10;    % Define sample sequence
>>t = k*T;     % Define time vector
>>y = step(sys_cl,t);  % Generate samples corresponding to time vector
>>plot(t,y,'o',t,y,'-');
>>grid;
>>xlabel('Time(seconds)');
>>ylabel('Response y(t)');
>>title('Unit Step Response');
```

Control force. As a final step in this example, let's examine the time history of control force $u(t)$ applied to the plant. One should take a worst-case input, then verify that $u(t)$ will not saturate the plant (the same check can be applied to all input and output signals). The block diagram must be redrawn to expose $U(z)$ as the output, as shown below. Note that pole/zero cancellation will not occur here as it did when $Y(z)$ was the output. The transfer function is then



$$\frac{U(z)}{R(z)} = \frac{D(z)}{1 + G(z)D(z)} \tag{5.62}$$

which may be found using the MATLAB `feedback` function, which finds a closed-loop system given a forward and a feedback transfer function:

```
>> sys_cf = feedback(Dd,Gd);  % D(z) in forward path, G(z) in feedback path
```

Lacking an actual worst-case reference input to apply, we'll just use a unit step input (with the same time vector as before):

```
>> step(sys_cf,t);
>> xlabel('Time (seconds)');
>> ylabel('Control force u(t)');
```

The resulting control force $u(t)$ is shown in Figure 5.18. If we knew the physical components, this plot could be used to check for saturation. Note the "stairstep" appearnace of the plot is appropriate for $u(t)$ since it is the output of the D/A, which is a ZOH, and generates a "stairstep" signal.

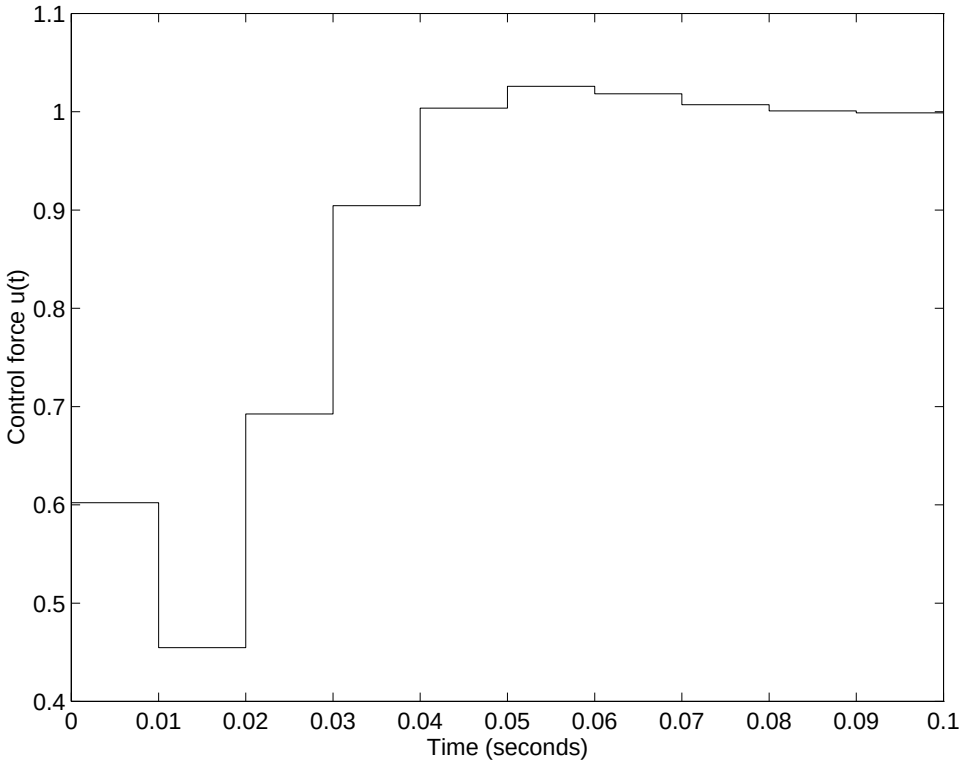


Figure 5.18: Control force $u(t)$ for unit step $r(t)$ in Example 2.

5.6 Modeling using Simulink

The MATLAB "companion" application `Simulink` offers a graphical approach to system simulation, and is frequently more useful than "plain MATLAB" for that function. However, it is the author's opinion that a good grounding in MATLAB *must* precede the use of Simulink. Also,

> "Simulink is *only for simulation*, and is *NOT A DESIGN TOOL!!*"

That being said, we'll use the design example of Section 5.5 as the system to illustrate the use of Simulink. Refer now to the block diagram of Figure 5.14, which is reproduced in Figure 5.19.

Recall that the compensator $D(z)$ transfer function in Figure 5.19 is given by

$$D(z) = \frac{0.6022z^2 - 0.5943z + 0.493}{z(z-1)} = \frac{0.6022z^2 - 0.5943z + 0.493}{z^2 - z} \tag{5.63}$$

The steps in creating a Simulink model of this system will be described below.

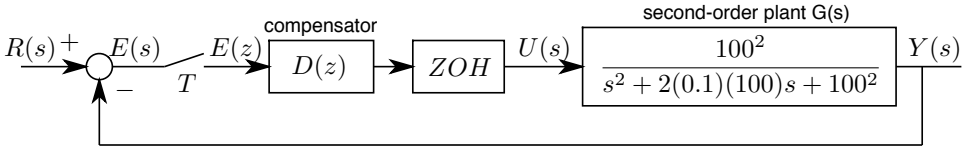


Figure 5.19: Block diagram for Simulink model.

5.6.1 Creating the Simulink Model

Figure 5.20 below shows the finished model. Note that it bears a very strong resemblance to the block diagram of Figure 5.19.

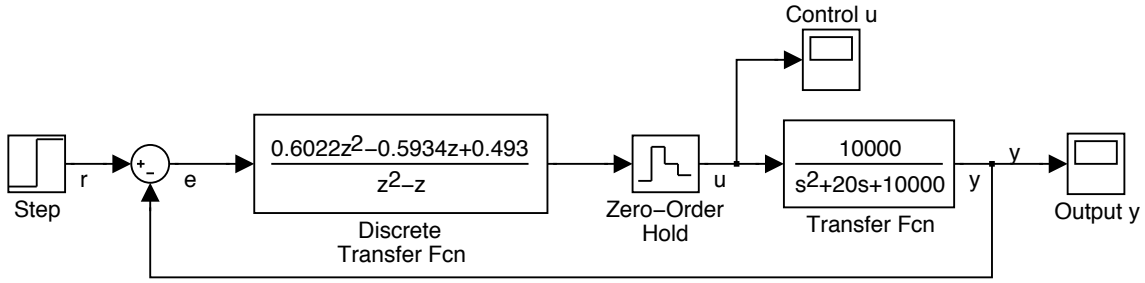


Figure 5.20: Block diagram of the finished Simulink model.

The blocks I used in creating this model are: `Sum`, `Discrete Transfer Fcn`, `Zero-Order Hold`, `Transfer Fcn`, `Step`, `Scope`. I didn't change their names in Figure 5.20 for clarity (except for the scopes). After placing a block you may change its name to reflect its physical significance.

Some items of note are:

- I modified `step` to occur at $t = 0$ instead of the $t = 1$ default.
- In placing the `Discrete Transfer Fcn` a sampler is automatically created. You must specify the sample period ($T = 0.01$ here) while modifying the block.
- The `Zero-Order Hold` block can "inherit" its sample period (-1 for sample time).
- Double-click on a signal path to enter a name (like r, e, u, y).
- The "stop time" under `Simulation/Configuration Parameters` should be changed from its default of 10 seconds to 0.1 seconds.

One of the advantageous of using Simulink is that "intermediate" variables (like control force u) can be easily displaced by just "picking them off" and connecting to a scope. The Simulink plotting facilities are minimal, but a given variable can be connected to a "`to Workspace`" block for subsequent plotting in MATLAB.

Figure 5.21 shows a screenshot of the Simulink model and the plots created by the scopes. The response shown in the plots agrees with our MATLAB simulation of Figures 5.17 and 5.18.

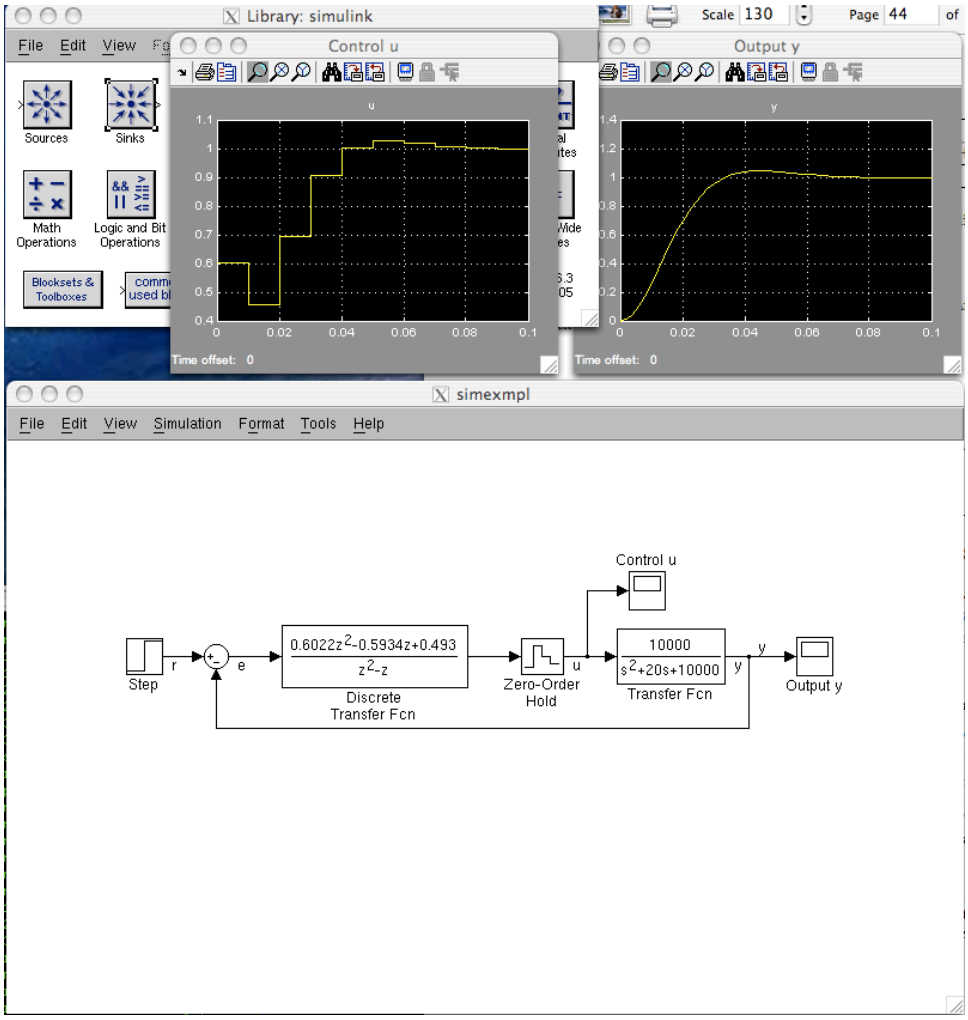


Figure 5.21: Simulink screenshot showing plots.

5.7 PID Control (Mode Controllers)

There are three basic types of control actions:

- Proportional—control force is *proportional* to the error signal.
- Integral—control force is proportional to the *integral* of the error signal
- Derivative—control force is proportional to the *derivative* of the error signal.

Integral and derivative control actions are used in combination with proportional, hence PI, PD, or PID controllers. The control modes are used in parallel, as shown in the block diagram of Figure 5.22. PID control action in the continuous domain is given by

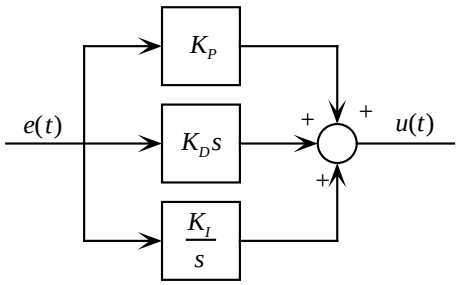


Figure 5.22: PID control action

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \dot{e}(t). \tag{5.64}$$

The three control actions will now be considered in the discrete domain.

5.7.1 Proportional Control

A discrete implementation of proportional control action is identical to continuous, thus

$$u(t) = K_P e(t) \Rightarrow D(s) = K_P,$$

the discrete is

$$u_k = K_P e_k \Rightarrow D(z) = K_P$$

Proportional control is always present—either by itself, or allied with derivative and/or integral control as shown below.

5.7.2 Derivative Action

In continuous systems, derivative action has the form

$$u(t) = K_D \dot{e}(t) \Rightarrow D(s) = K_D s.$$

In the discrete domain, differentiation can be approximated by the first difference, that is,

$$u_k = K_D\{e_k - e_{k-1}\} \Rightarrow D(z) = K_D(1 - z^{-1}) = K_D \frac{z-1}{z}$$

Derivative action is typically used to improve system stability and transient response.

5.7.3 Integral Action

Continuous integral action is given by

$$u(t) = K_I \int_0^t e(t)dt \Rightarrow D(s) = \frac{K_I}{s}$$

Discrete integration is a cumulative sum, represented by

$$u_k = u_{k-1} + K_I e_k \Rightarrow D(z) = K_I \frac{1}{1 - z^{-1}} = K_I \frac{z}{z-1}$$

NOTE: Gains K_I and K_D have different values in the continuous and discrete domains!

Integral action is typically used to improve system steady-state error behavior. It usually worsens transient response and stability.

The I and D control actions are rarely used by themselves, but are coupled with proportional (P) action for PD, PI, or PID controllers.

5.7.4 PD Control

The combination of Proportional and Derivative action is given by

$$\begin{aligned} K_P + K_D \frac{z-1}{z} &= \frac{K_D\left(z - \frac{K_D + K_P}{K_D}\right)}{z} \\ &= \frac{K(z-a)}{z} \end{aligned}$$

which is a lead compensator of the same form we used in Example 2.

5.7.5 PI Control

Proportional plus Integral control is

$$\begin{aligned} K_P + K_I \frac{z}{z-1} &= \frac{(K_P + K_I)\left(z - \frac{K_P}{K_P+K_I}\right)}{z-1} \\ &= \frac{K(z-a)}{z-1} \end{aligned}$$

which has a pole at $z = 1$ corresponding to the integration, and a zero on the positive real axis nearer the origin. It is this pole at $z = 1$ that improves steady-state behavior but hurts stability and transient response. The zero is usually placed fairly near the pole at $z = 1$ to reduce the effect of the "slow" closed-loop pole that will be in this region. Note that it is the *open-loop* pole at $z = 1$ that improves the steady-state error by raising the Type (see Sec. 5.2.1) of the system.

5.7.6 PID Control

Adding integral action to form three-mode proportional plus integral plus derivative action yields

$$\begin{aligned} K_P + K_D \frac{z-1}{z} + K_I \frac{z}{z-1} &= \frac{(K_P + K_D + K_I)\, z^2 - (K_P + 2K_D)\, z + K_D}{z(z-1)} \\ &= \frac{K(z^2 - az + b)}{z(z-1)} \end{aligned}$$

which has a pole at $z = 1$ from the integration, a pole at $z = 0$, and two zeros which may be real or complex, depending on the relative values of the three gains.

One strategy for PID design is as follows:

1. Place one zero and pole as for the PD design.
2. Place the second pole at $z = 1$ (the integration), and place the zero "nearby" for the same reason discussed at the end of Section 5.7.5 above.

Homework Problem Set 2

Problem 1.

It is possible to suspend a mass of magnetic material by means of an electromagnet whose current is controlled by the position of the mass. A sketch of a possible setup is shown in Figure 5.23. The

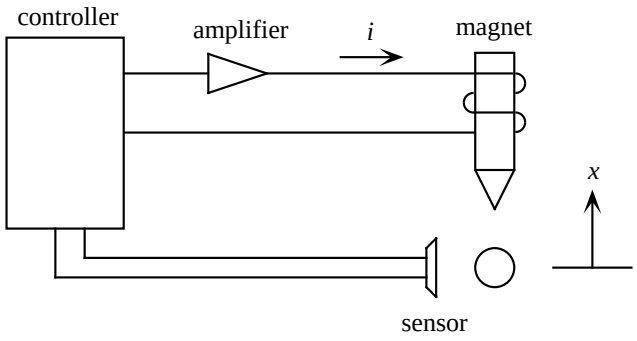


Figure 5.23: A ball suspended by an electromagnet.

equations of motion are

$$m\ddot{x} = -mg + f(x, I),$$

where the force on the ball due to the electromagnet is given by $f(x, I)$. At equilibrium, the magnet force balances the gravity force; call the current at that equilibrium point I_o, this will be the *operating point*. If we write $I = I_o + i$ and expand f about $\{x = 0 \text{ and } I = I_o\}$, neglecting higher-order terms, we obtain

$$m\ddot{x} = k_1 x + k_2 i.$$

Use values of m=0.02 kg, k_1=20 N/m, k_2=0.4 N/A.

(a) Compute the transfer function from i to x and draw the (continuous) root locus for proportional feedback $i = -Kx$.

(b) Let the sample period be T =0.02 sec and compute the plant discrete transfer function when used with a zero-order hold.

(c) Design a digital controller for the magnetic levitator to meet specifications $t_r \leq 0.1$ sec, $t_s \leq 0.4$ sec, and overshoot $\leq 20\%$.

(d) Plot the step response of your design (introduce a reference input x_r) and show both ball displacement x and control current i. If the sensor can measure x over a range of only $\pm\frac{1}{4}$ cm, and if the amplifier can provide a current of only 1A, what is the *maximum* displacement possible for control (neglecting the nonlinear terms in $f(x, I)$)?

Problem 2.

For the automotive cruise-control system shown in Figure 5.24, velocity is given in mph, while grade (a disturbance) is given in "% grade." Use sample period $T = 0.5$ sec and

(a) Design a proportional controller to achieve a t_r of 5 seconds with no overshoot.

(b) Determine the speed error on a 3% grade (*i.e.* $G_r = 3$ in Figure 5.24).

(c) Design a PD, PI, or PID controller (the simplest one necessary) to meet the same specifications as part (a) and that has no error on grades.

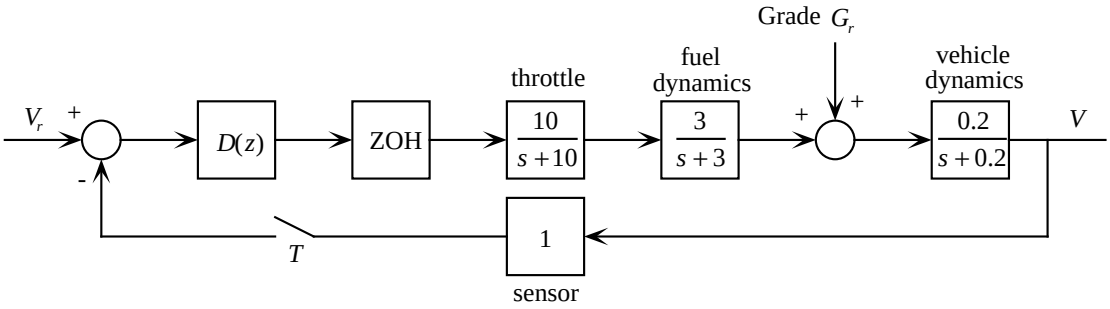


Figure 5.24: An automotive cruise-control system.

Chapter 6

State-Space Analysis of Continuous Systems

6.1 Introduction

As a prelude to the state-space analysis of discrete-time systems, this chapter will introduce state-variable modeling and analysis of continuous-time systems. While many students will have been exposed to this material, we fell that a review is useful before the concepts are applied to discrete-time systems.

The idea of *state space* comes from the state-variable method of describing differential equations. In this method, dynamic systems are described by a set of first-order differential equations in variables called the *states*, and the solution may be thought of as a trajectory in the *state space*. This formulation is particularly well-suited to computer calculation, and is as easily applied to nonlinear, time-varying systems as to linear time-invariant systems. We will examine only linear time-invariant (LTI) systems in this course.

Use of the state-space approach is often referred to as *modern control*, and the use of transform methods termed *classical control*. However, the state-space formulation of differential equations is over 100 years old, being introduced to control design in the late 1950s (by Soviet mathematicians), so it may be misleading to consider it "modern." It is better to refer to the two approaches as *transform* methods and *state-space* methods.

The goal of state-space design is the same as transform-based design, to find compensation $D(s)$ that yields acceptable performance in the system shown in Figure 6.1. The state-space approach is so different that it is easy to lose sight of the end result, which is the same.

State-space methods are especially advantageous when designing controllers for systems with multiple inputs or outputs. The problem formulation is the same, whether there are single or multiple inputs and outputs. We will, however, generally consider single-input single-output (SISO) systems

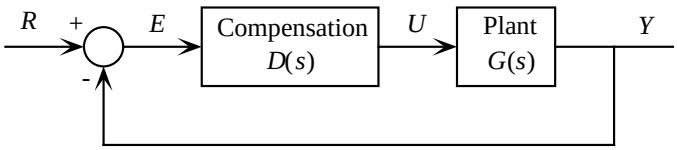


Figure 6.1: Control system design.

in this course, although there will be exceptions. Another advantage of state-space formulation is that the internal behavior is exposed, rather than the "black-box" input-output modeling of transform methods.

6.2 System Description

Any lumped-parameter dynamic system can be described using a set of first-order ordinary differential equations, where the dependent variables are the *state* variables.

State Variables Knowledge of the state variables at some initial time, plus knowledge of the input from that time on, is sufficient to predict the system behavior. Variables with which initial conditions are necessary are by definition state variables. State variables also define the energy stored within the system. In mechanical systems, position and velocity are valid state variables. In electrical systems, voltage and current are state variables. The system output and its derivatives are always valid state variables. Any linear combination of state variables is itself a valid state variable... there are an infinite number of choices. The set of state variables is collectively called the *state vector* or simply the *state*.

6.2.1 State Equation and Output Equation

The general form of each scalar state equation is

$$\dot{x} = f(\mathbf{x}, \mathbf{u}, t)$$

For an LTI system the state equation can be written in matrix form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{6.1}$$

where boldface notation denotes vector/matrix quantities. The $n \times 1$ state vector is $\mathbf{x}$, and $n \times n$ matrix $\mathbf{A}$ is the *system matrix*. Input vector $\mathbf{u}$ is $m \times 1$. *Input matrix* $\mathbf{B}$ is dimension $n \times m$. For a single-input system, u is a scalar, and $\mathbf{B}$ is dimension $n \times 1$.

The output equation is

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{6.2}$$

where $\mathbf{y}$ is the $p \times 1$ vector of outputs, $\mathbf{C}$ is the $p \times n$ output matrix, and $\mathbf{D}$ is the $p \times m$ feedforward matrix which connects the input directly to the output.

Usually in this course we will have a SISO system, and the state equation and output equations can be written

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{6.3}$$

$$y = \mathbf{C}\mathbf{x} + Du \tag{6.4}$$

where feedforward term D is usually zero for physical systems.

Example

Consider the spring-mass-damper system shown in Figure 6.2. The input is applied force $f(t)$ and

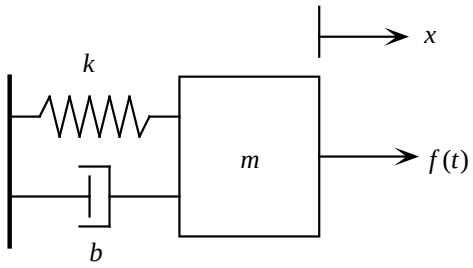


Figure 6.2: Spring-mass-damper system.

the output is position $x(t)$. The equation of motion for this system is

$$f(t) - b\dot{x} - kx = m\ddot{x}$$

Position and velocity are valid state variables, hence

$$\begin{aligned} x_1 &\triangleq x \\ x_2 &\triangleq \dot{x} \end{aligned}$$

The state equations are

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}f(t) \end{aligned}$$

hence the system description is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -k/m & -b/m \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, D = 0$$

6.2.2 State equation from Transfer Function

Given transfer function

$$\frac{Y(s)}{U(s)} = \frac{b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0} \tag{6.5}$$

it can be shown that a state equation is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t) \tag{6.6}$$

and the output equation is

$$y(t) = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{6.7}$$

We will see shortly that equations (6.6–6.7) are in a special (minimal) form called (lower) *control canonical* form.

6.2.3 Transfer Function from State-Variable Description

It is possible to get a transfer function from the state space description of a system. Assuming zero initial conditions on state $\mathbf{x}$, taking the Laplace transform of state equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{6.8}$$

yields

$$s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}U(s) \tag{6.9}$$

which can be rearranged (paying attention to the multiplication order since these are matrices!) as

$$(s\mathbf{I} - \mathbf{A})\mathbf{X}(s) = \mathbf{B}U(s) \tag{6.10}$$

which can be solved for

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s) \tag{6.11}$$

Output equation $y = \mathbf{C}\mathbf{x} + Du$ with $D = 0$ has Laplace transform

$$Y(s) = \mathbf{C}\mathbf{x}(s) \tag{6.12}$$

which can be used to get

$$Y(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s) \tag{6.13}$$

from which we get the transfer function as

$$\frac{Y(s)}{U(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \tag{6.14}$$

Example

The transfer function of the spring-mass-damper system is

$$\frac{X(s)}{F(s)} = \frac{1/m}{s^2 + (b/m)s + (k/m)}$$

and if we let $m = 1, b = 2, k = 4$ we get

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 2s + 4}$$

The corresponding state-variable description is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, D = 0.$$

Then

$$s\mathbf{I} - \mathbf{A} = \begin{bmatrix} s & -1 \\ 4 & s+2 \end{bmatrix}$$

and

$$(s\mathbf{I} - \mathbf{A})^{-1} = \frac{\begin{bmatrix} s+2 & 1 \\ -4 & s \end{bmatrix}}{s(s+2)+4}$$

The transfer function is given by

$$\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \frac{\begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} s+2 & 1 \\ -4 & s \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}}{s(s+2)+4}$$

which yields

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 2s + 4}.$$

Numerical conversion between state-space form to transfer function form can be easily done with MATLAB, as will be shown later.

6.3 Different State-Space Representations

In Section 6.2 we mentioned that there are an infinite number of choices for system state, and corresponding system matrices. This section presents some standard representations. First comes the idea of state equation transformations, followed by several different representations. The usefulness of some of these descriptions may not be immediately apparent. Where we need a specific system, we will consider the third-order transfer function

$$\frac{Y(s)}{U(s)} = \frac{b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0} = \frac{b(s)}{a(s)} \tag{6.15}$$

6.3.1 State Variable Transformation

Given state equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{6.16}$$

and output equation

$$y = \mathbf{C}\mathbf{x} + Du \tag{6.17}$$

let a new state vector $\mathbf{z}$ be defined by

$$\mathbf{z} = \mathbf{T}\mathbf{x} \tag{6.18}$$

where $\mathbf{T}$ is a $n \times n$ non-singular matrix. Note that $\mathbf{T}$ must be invertible to permit mapping between both state vectors, thus

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{z} \tag{6.19}$$

Now, starting with the new state $\mathbf{z}$, we have

$$\dot{\mathbf{z}} = \mathbf{T}\dot{\mathbf{x}} = \mathbf{T}\mathbf{A}\mathbf{x} + \mathbf{T}\mathbf{B}u \tag{6.20}$$

then substituting for $\mathbf{x}$ yields

$$\dot{\mathbf{z}} = \mathbf{T}\mathbf{A}\mathbf{T}^{-1}\mathbf{z} + \mathbf{T}\mathbf{B}u \tag{6.21}$$

and the output equation is

$$y = \mathbf{C}\mathbf{T}^{-1}\mathbf{z} + Du. \tag{6.22}$$

Although *eigenvalues* and *eigenvectors* will described in Section 6.3.3, the eigenvalues of original system matrix $\mathbf{A}$ and "new" matrix $\mathbf{T}\mathbf{A}\mathbf{T}^{-1}$ are *the same!*. For this reason these matrices are called "similar."

6.3.2 Control Canonical Form

Introduce a variable v which relates polynomials $a(s)$ and $b(s)$ as shown in Figure 6.3(a). The transfer function from u to v is

$$\frac{V(s)}{U(s)} = \frac{1}{a(s)} = \frac{1}{s^3 + a_2 s^2 + a_1 s + a_0} \tag{6.23}$$

or

$$\dddot{v} + a_2\ddot{v} + a_1\dot{v} + a_0 v = u \tag{6.24}$$

which may be written

$$\dddot{v} = -a_2\ddot{v} - a_1\dot{v} - a_0 v + u \tag{6.25}$$

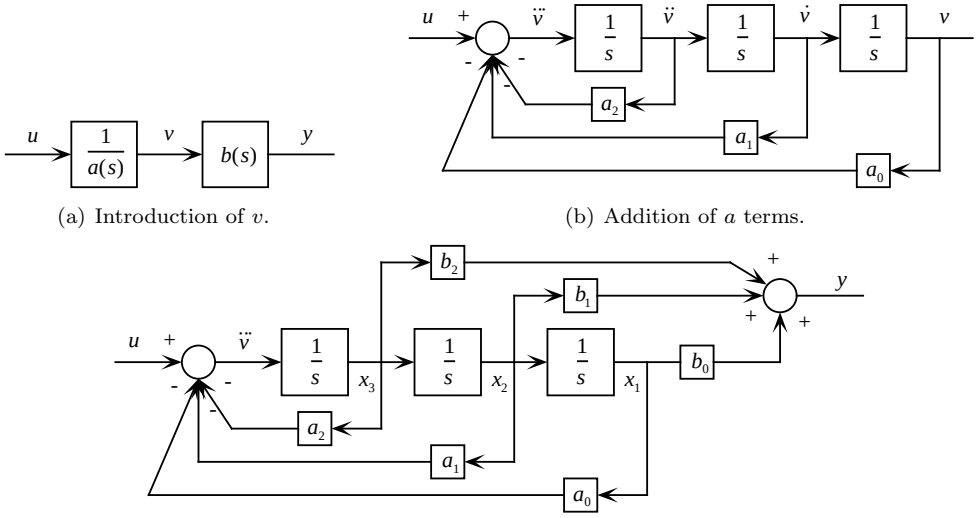
A block diagram illustrating (6.25) is shown in Figure 6.3(b).

The output y is formed by

$$Y(s) = b(s)V(s) \tag{6.26}$$

which means

$$y = b_2\ddot{v} + b_1\dot{v} + b_0 v \tag{6.27}$$



(a) Introduction of v.

(b) Addition of a terms.

(c) Addition of b terms.

Figure 6.3: Control Canonical Form.

The block diagram with the output terms added is shown in Figure 6.3(c).

The outputs of the integrators are the state variables. Taking the state variables as

$$x_1 \triangleq v, x_2 \triangleq \dot{v}, x_3 \triangleq \ddot{v} \tag{6.28}$$

we obtain

$$\dot{x}_1 = x_2 \tag{6.29}$$
$$\dot{x}_2 = x_3 \tag{6.30}$$
$$\dot{x}_3 = -a_0 x_1 - a_1 x_2 - a_2 x_3 + u \tag{6.31}$$

thus the system matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix}, D = 0 \tag{6.32}$$

The form of (6.32) is known as *control canonical* form. It is especially useful in finding state-variable feedback laws (not covered in this chapter). It is a minimal form, and is the same as shown in equation (6.6).

Observer Canonical A similar form, *observer canonical*, is useful when designing an *observer*, or state estimator. This form can be derived in a similar manner to control canonical form, one version

of the result is:

$$\mathbf{A} = \begin{bmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_2 \\ b_1 \\ b_0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, D = 0. \tag{6.33}$$

With the observer canonical form the output $y = x_1$.

6.3.3 Diagonal (modal or decoupled) Form

The simplest representation of a state-space system occurs when the state equation are *decoupled*, that is, when system matrix $\mathbf{A}$ is in *diagonalized* form, in the form

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{6.34}$$

Eigenvalues and Eigenvectors Given matrix $\mathbf{A}$, vector $\mathbf{x}$ is an eigenvector if

$$\mathbf{Ax} = \lambda\mathbf{x} \tag{6.35}$$

where λ is a scalar eigenvalue. These exceptional vectors get shrunk or reversed by $\mathbf{A}$ but *they don't change direction.* The relationship between the state variables stays fixed. Eigenvectors are commonly called *mode shapes*, and they represent the natural response modes of the system.

To find the eigenvalues write the equation as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$. For this equation to have a non-trivial solution for $\mathbf{x}$, matrix $\mathbf{A} - \lambda\mathbf{I}$ must be singular, therefore it must have a zero determinant, and

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \tag{6.36}$$

This is also the *characteristic equation* of the system, and poles and eigenvalues are the same thing. An n^{th} order system will (usually) have n eigenvalues $\lambda_1 \ldots \lambda_n$.

To find the i^{th} eigenvector, solve the system

$$(\mathbf{A} - \lambda_i\mathbf{I})\mathbf{x}_i = \mathbf{0} \tag{6.37}$$

for eigenvector $\mathbf{x}_i$. MATLAB function `eig` is used for finding eigenvalues and eigenvectors.

Diagonalization The matrix of eigenvectors are just what we need to diagonalize matrix $\mathbf{A}$. We need transformation matrix $\mathbf{T}$ such that

$$\mathbf{\Lambda} = \mathbf{TAT}^{-1}$$

Multiply $\mathbf{A}$ times the matrix $\mathbf{S}$ of its eigenvectors,

$$\mathbf{AS} = \mathbf{A}[\mathbf{x}_1 \cdots \mathbf{x}_n] = [\lambda_1 \mathbf{x}_1 \cdots \lambda_n \mathbf{x}_n] \tag{6.38}$$

then write this as

$$[\lambda_1 \mathbf{x}_1 \cdots \lambda_n \mathbf{x}_n] = [\mathbf{x}_1 \cdots \mathbf{x}_n] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} = \mathbf{S}\mathbf{\Lambda} \tag{6.39}$$

So then we have

$$\mathbf{AS} = \mathbf{S}\mathbf{\Lambda} \rightarrow \mathbf{\Lambda} = \mathbf{S}^{-1}\mathbf{AS} \tag{6.40}$$

Note that by the format of equation (6.21) the diagonalizing transform is $\mathbf{T} = \mathbf{S}^{-1}$, where $\mathbf{S}$ is the matrix whose columns are the eigenvectors.

The modal form is especially useful in structural dynamics, where there is concern for structural mode shapes and natural frequencies (square root of eigenvalues). MATLAB function `canon()` with the `modal` option will diagonalize a matrix, and return the transformation $\mathbf{T}$ that it used.

6.4 MATLAB Tools

Many MATLAB functions exist for handling state-space systems, some common ones will be described in this section.

6.4.1 Transform↔State-Space

Changing between transform and state-space domains is done with `tf2ss` and `ss2tf`. These work identically in the continuous and discrete domains.

tf2ss. This function converts a transform-based form to state space representation. The $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ matrices are returned in (upper) control canonical form, slightly different than the (lower) control canonical form of Section 6.3.2. A transformation may be used to convert between these forms. This is illustrated by the following example. Consider transfer function

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

This may be converted to state-space form by

```
>>num = [0 1 5 6];
>>den = [1 2 3 4];
>>[A,B,C,D] = tf2ss(num,den)

A = -2    -3    -4
     1     0     0
     0     1     0

B =  1
     0
     0

C =  1     5     6

D =  0
```

To transform this to lower canonical form one can use the change of state

$$\mathbf{z} = \mathbf{T}\mathbf{x}.$$

with matrix $\mathbf{T}$ given by

```
>>T = flipud(eye(3))

T =  0     0     1
     0     1     0
     1     0     0
```

It is easily seen that $\mathbf{T}$ simply reverses the order of the state variables. To transform the state-space model use the `ss2ss` function:

```
>>[Al,Bl,Cl,Dl] = ss2ss(A,B,C,D,T)

Al = 0     1     0
     0     0     1
    -4    -3    -2

Bl = 0
     0
     1

Cl = 6     5     1

Dl = 0
```

where the state-space is now in the format of equation (6.32).

ss2tf. This just goes the other way. For the previous example, using either canonical form (all state-space forms must yield the same input-output model):

```
>>[num,den] = ss2tf(A,B,C,D)

num =  0    1.0000    5.0000    6.0000

den =  1.0000    2.0000    3.0000    4.0000
```

6.4.2 Eigenvalues, Eigenvectors, Diagonalization

Matlab function `eig` finds the eigenvalues and eigenvectors. The *eigenvectors* will be different depending on the form of the state equations, but the *eigenvalues* will always be the same. Considering the previous example, using the lower canonical form

```
>>[Vectors,Values] = eig(Al)

Vectors = 0.2995            -0.1781 - 0.2754i  -0.1781 + 0.2754i
         -0.4944             0.4572 - 0.2274i   0.4572 + 0.2274i
          0.8160             0.2718 + 0.7469i   0.2718 - 0.7469i

Values =  -1.6506                  0                  0
                0            -0.1747 + 1.5469i        0
                0                  0            -0.1747 - 1.5469i
```

Function `eig` returns a matrix whose columns are the eigenvectors, and a matrix whose diagonal elements are the eigenvalues. For a system matrix with real coefficients the eigenvalues will always be either real or complex conjugate pairs. The eigenvalues are the roots of the denominator of the original $G(s)$.

We showed that the transformation needed to diagonalize the system matrix was the inverse of the eigenvector matrix. To verify this,

```
>>Tdiag = inv(Vectors)

Tdiag =  1.7700 - 0.0000i   0.2552 - 0.0000i   0.7304 - 0.0000i
         0.4032 + 1.1137i   0.9842 + 0.4976i   0.4483 - 0.1073i
         0.4032 - 1.1137i   0.9842 - 0.4976i   0.4483 + 0.1073i
```

Now use transformation **Tdiag** to change state using `ss2ss`:

```
>>[Ad,Bd,Cd,Dd] = ss2ss(Al,Bl,Cl,Dl,Tdiag)

Ad =  -1.6506 - 0.0000i   0.0000 + 0.0000i   0.0000 - 0.0000i
       0.0000 + 0.0000i  -0.1747 + 1.5469i   0.0000 - 0.0000i
       0.0000 - 0.0000i   0.0000 + 0.0000i  -0.1747 - 1.5469i

Bd =  0.7304 - 0.0000i
      0.4483 - 0.1073i
      0.4483 + 0.1073i

Cd =  0.1412 + 0.0000i   1.4891 - 2.0425i   1.4891 + 2.0425i

Dd =  0
```

canon. MATLAB function `canon` transforms the state equations into a modal "canonical" form in which the real eigenvalues appear on the diagonal, and complex eigenvalues appear in a 2×2 block on the diagonal of the $\mathbf{A}$ matrix. For a system with eigenvalues $\lambda_1, \sigma \pm j\omega$, the modal $\mathbf{A}$ matrix is

$$\mathbf{A} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \sigma & \omega \\ 0 & -\omega & \sigma \end{bmatrix}$$

For the original $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ system,

```
>>[Am,Bm,Cm,Dm,Tmodal] = canon(A,B,C,D,'modal')

Am =  -1.6506    0.0000   -0.0000
       0.0000   -0.1747    1.5469
      -0.0000   -1.5469   -0.1747

Bm =  -0.7304
      -0.6526
       0.6511

Cm =  -0.1412    0.9591    2.3387

Dm =   0


Tmodal =  -0.7304   -0.2552   -1.7700
          -0.6526   -0.1840    1.4743
           0.6511    2.1980    1.8541
```

where **Tmodal** is the transformation matrix used to produce the modal form.

6.4.3 Dynamic Response of State-Space Forms

There is a rich set of MATLAB functions to obtain the dynamic response of state-space systems. Two of the three functions shown below require an `LTI`-format model, while the third uses the separate A, B, C, D matrices.

Refer to Section 2.8 for a brief discussion of LTI models. To create an LTI model from a state-space system description, use the `ss` command:

```
>> help ss
 SS  Creates state-space model or converts model to state space.

    SYS = SS(A,B,C,D) creates a SS object SYS representing the
    continuous-time state-space model
          dx/dt = Ax(t) + Bu(t)
           y(t) = Cx(t) + Du(t)
```

`[Y,T,X] = STEP(SYS)` Step response of continuous-time linear systems. The syntax shown returns the output and state time response in the matrices Y and X respectively. No plot is drawn on the screen. The matrix Y has as many columns as there are outputs, and LENGTH(T) rows. The matrix X has as many columns as there are states. If the time vector is not specified, then the automatically determined time vector is returned in T. A desired time vector T can also be specified as the last argument (after SYS) in the argument list of STEP.

`[Y,T,X] = INITIAL(SYS,X0)` Initial condition response of continuous-time linear systems. The syntax shown returns the output and state responses (Y and X), and the time vector (T). No plot is drawn on the screen. The matrix Y has as many columns as outputs and one row for element in T. Similarly, the matrix X has as many columns as states and length(T) rows. Again, a desired time vector T can be given as the last argument, after the initial condition X0.

`[Y,X] = LSIM(A,B,C,D,U,T)` Simulation of continuous-time linear systems to arbitrary inputs. The syntax shown returns the output and state time history in the matrices Y and X. Each row of U corresponds to a new time point, and U must have LENGTH(T) rows. The time vector T must be regularly spaced. LSIM(A,B,C,D,U,T,X0) can be used if initial conditions exist.

For more information on these functions invoke the MATLAB `help` feature.

Homework Problems

Problem 1.

In this problem you are going to write the state equations to simulate the dynamics of a DC motor driving a rotary inertia through a gear train, as shown in the sketch below. The equations that

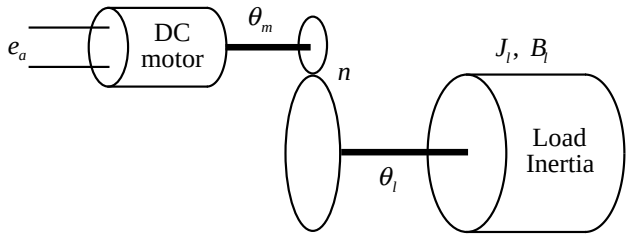


Figure 6.4: DC motor driving inertia.

model the dynamic behavior of a permanent-magnet DC motor are

$$e_a - i_a R_a - L_a \frac{di_a}{dt} - K_b \omega_m = 0$$

$$\tau = K_t i_a$$

$$\tau - B_m \omega_m = J_m \dot{\omega}_m$$

where e_a is applied armature voltage, i_a is armature current, R_a is armature resistance, L_a is armature inductance, K_b is the back emf constant, ω_m is motor speed, τ is generated torque, K_t is motor torque constant, B_m is motor viscous friction coefficient, and J_m is motor armature inertia.

Recall that when a load inertia (and damping) is connected to a motor *via* a gear train, the load inertia the motor "sees" is reduced by a factor of n^2, where n is the gear ratio ($n > 1$ for a speed reducing gear train).

The specifications for the motor (taken from the manufacturer's data sheet) are:

$$\begin{aligned} J_m &= 1.54 * 10^{-6} \text{ kg-m}^2 \\ B_m &= 2.7e-3 \ \frac{\text{mN-m}}{\text{rad/s}} \\ L_a &= 0.69 \text{ mH} \\ R_a &= 1.53 \ \Omega \\ K_t &= 22.3 \ \frac{\text{mN-m}}{\text{A}} \\ K_b &= 0.0223 \ \frac{\text{V}}{\text{rad/s}} \end{aligned}$$

The load inertia is a disk made of aluminum with the following dimensions:

$$\begin{aligned} r &= 100 \text{ mm} \\ \text{thickness} &= 10 \text{ mm} \\ \text{viscous damping} &= 4 \ \frac{\text{mN-m}}{\text{rad/s}} \\ \text{gear ratio } n &= 50 \end{aligned}$$

(a) Choosing the state variables to be $x_1 = i_a$, $x_2 = \theta$, and $x_3 = \dot{\theta} = \omega$, write the state equations for the motor alone. Simulate the motor response to a voltage pulse of magnitude 10V and duration 0.1 second. Simulate for a duration of 0.2 seconds. Plot motor current (A), motor velocity (rpm), and motor displacement (revolutions) *vs* time. What is the maximum armature current? How many revolutions did the motor rotate?

(b) Now add the dynamics of the load, but ignore the inertia of the gears (it was not given anyway!). Perform the same simulation as in (a). Plot motor current (A), load velocity (degrees/sec), and load displacement (degrees) *vs* time.

Problem 2.

Consider the two degree-of-freedom system shown below. Let $m = 1$ and $k = 1$. Write state

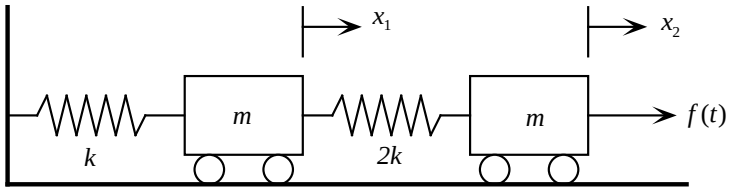


Figure 6.5: Coupled spring-mass system.

equations for this system using

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{bmatrix}$$

By diagonalizing the system, investigate the eigenvalues (natural frequencies) and eigenvectors (mode shapes).

Chapter 7

Digital Controller Design using State Space Methods

7.1 Introduction

The state variable modeling approach presented in the previous chapter can also be applied in the discrete domain. We will first examine state-space discrete models, then look at state feedback controls laws, in which the control force u is a linear combination of the measured state variables. Using state feedback, the system poles can be placed arbitrarily. Since the entire state vector is usually not accessible, a state estimator must be developed. It will be shown that the poles of the overall system are those of the control law plus those of the estimator. Introduction of the reference input will be the final chapter topic.

7.2 Canonical State-Space Forms from Transfer Function

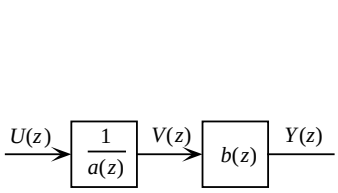
It is easy to get a canonical discrete state-space model if you already have a discrete transfer function. That is not the typical path to follow when modeling, but it is worthwhile to discuss.

Consider the transfer function given by

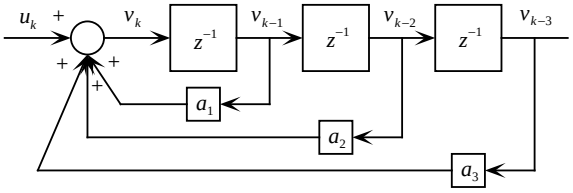
$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3}} = \frac{b(z)}{a(z)} \tag{7.1}$$

where the presence of b_0 indicates a direct "feed-through" term. This was included because it is often present in discrete-domain controllers.

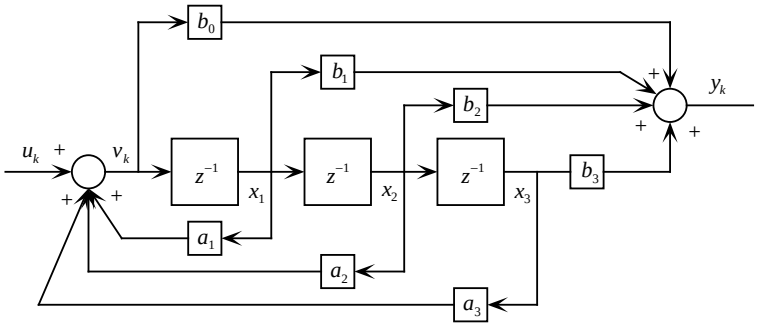
Polynomials $a(z)$ and $b(z)$ can be separated using the block diagram of Figure 7.1(a) and the introduction of intermediate variable v_k with transform $V(z)$. Thus



(a) Introduction of v.



(b) Addition of a terms.



(c) Addition of b terms.

Figure 7.1: Control Canonical Form.

$$V(z) = \frac{U(z)}{a(z)} \text{ or } U(z) = a(z)V(z)$$

and

$$u_k = v_k - a_1 v_{k-1} - a_2 v_{k-2} - a_3 v_{k-3}$$

or

$$v_k = a_1 v_{k-1} + a_2 v_{k-2} + a_3 v_{k-3} + u_k$$

and the block diagram in Figure 7.1(b) results. The delay elements are analogous to integrators for continuous systems.

To form the output y_k use $Y(z) = b(z)V(z)$, which leads to

$$y_k = b_0 v_k + b_1 v_{k-1} + b_2 v_{k-2} + b_3 v_{k-3}.$$

which is shown in the block diagram in Figure 7.1(c)

Discrete State Equations. To avoid confusion with the continuous system and input matrices, we'll use symbols $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ instead of $\mathbf{A}$ and $\mathbf{B}$. The discrete state and output equations are:

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k) \tag{7.2}$$

$$y(k) = \mathbf{C}\mathbf{x}(k) + Du(k) \tag{7.3}$$

Choosing state variables x_i as shown in Figure 7.1(c) leads to state equations

$$\begin{aligned} x_1(k+1) &= a_1x_1(k) + a_2x_2(k) + a_3x_3(k) + u(k) \\ x_2(k+1) &= x_1(k) \\ x_3(k+1) &= x_2(k) \end{aligned}$$

and corresponding system and input matrices

$$\mathbf{\Phi} = \begin{bmatrix} a_1 & a_2 & a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{7.4}$$

This control canonical form is the same form MATLAB returns when doing a `tf2ss` transformation.

The output equation is

$$Y(z) = b(z)V(z)$$

which leads to

$$y_k = b_0v_k + b_1v_{k-1} + b_2v_{k-2} + b_3v_{k-3}$$

where $v_{k-1}, v_{k-2}, v_{k-3}$ are state variables x_1, x_2, x_3. However, the b_0 term complicates things somewhat. The quantity v_k is not a state variable, but is

$$\begin{aligned} v_k &= a_1v_{k-1} + a_2v_{k-2} + a_3v_{k-3} + u_k \\ &= a_1x_1(k) + a_2x_2(k) + a_3x_3(k) + u(k) \end{aligned}$$

where the "()" sampling index notation is used to avoid confusion with the state variable subscripts. Substitition for $v(k)$ yields output equation

$$y(k) = [b_1 + a_1b_0]x_1(k) + [b_2 + a_2b_0]x_2(k) + [b_3 + a_3b_0]x_3(k)$$

thus output and feedforward matrices are

$$\mathbf{C} = \begin{bmatrix} b_1 + a_1b_0 & b_2 + a_2b_0 & b_3 + a_3b_0 \end{bmatrix}, D = b_0 \tag{7.5}$$

The state-space model of a discrete system is then given by equations (7.2) and (7.3) with control canonical matrices given by (7.4) and (7.5). If there is no feedforward term b_0 the output matrix is $\mathbf{C} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$ and of course $D = 0$.

7.3 Solution to the State Equation

If we intend to use state-variable modeling, it is unnecessary to find a transfer function $G(z)$ to get the discrete state-space form. A discrete-time model generates the behavior of the system from sample to sample, and to get this we must solve the continuous state equation. Consider the state equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t). \tag{7.6}$$

The right-hand side of (7.6) consists of two parts, the first of these, $\mathbf{A}\mathbf{x}(t)$, is the *homogeneous* part while the second, $\mathbf{B}u(t)$, is the *input* portion.

7.3.1 Homogeneous solution

The homogeneous (no input) solution to (7.6) is given by

$$\mathbf{x}_h(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}(t_0) \tag{7.7}$$

where the *matrix exponential* is

$$\begin{aligned} e^{\mathbf{A}(t-t_0)} &= \mathbf{I} + \mathbf{A}(t-t_0) + \mathbf{A}^2\frac{(t-t_0)^2}{2!} + \mathbf{A}^3\frac{(t-t_0)^3}{3!} + \cdots & (7.8)\\ &= \sum_{k=0}^{\infty}\mathbf{A}^k\frac{(t-t_0)^k}{k!} & (7.9)\end{aligned}$$

By differentiating (7.7) with respect to t one can show that (7.9) is a solution to the homogeneous problem.

MATLAB has the matrix exponential function available for numerical evaluation as `expm`.

Note that although the initial state has been given at time $t = t_0$, initial conditions are more commonly given at $t = 0$.

Properties of the Matrix Exponential. The matrix exponential function has properties which we will need in the solution of the inhomogeneous problem.

Given times t_1 and t_2, we have

$$\mathbf{x}(t_1) = e^{\mathbf{A}(t_1-t_0)}\mathbf{x}(t_0)$$

and

$$\mathbf{x}(t_2) = e^{\mathbf{A}(t_2-t_0)}\mathbf{x}(t_0).$$

The initial time t_0 is arbitrary, and we can write

$$\mathbf{x}(t_2) = e^{\mathbf{A}(t_2-t_1)}\mathbf{x}(t_1).$$

Substituting for $\mathbf{x}(t_1)$ gives

$$\mathbf{x}(t_2) = e^{\mathbf{A}(t_2-t_1)}e^{\mathbf{A}(t_1-t_0)}\mathbf{x}(t_0)$$

and equating the two solutions for $\mathbf{x}(t_2)$ we get

$$e^{\mathbf{A}(t_2-t_0)} = e^{\mathbf{A}(t_2-t_1)}e^{\mathbf{A}(t_1-t_0)} \tag{7.10}$$

Note especially that if $t_2 = t_0$, then

$$\mathbf{I} = e^{-\mathbf{A}(t_1-t_0)}e^{\mathbf{A}(t_1-t_0)}.$$

Thus we can obtain the inverse of $e^{\mathbf{A}t}$ simply by changing the sign of t.

7.3.2 Particular solution

Assume the solution to be of the form

$$\mathbf{x}_p(t) = e^{\mathbf{A}(t-t_0)}\mathbf{v}(t) \tag{7.11}$$

where $\mathbf{v}(t)$ is a vector of variable parameters to be determined. Substituting (7.11) into (7.6), we obtain

$$\mathbf{A}e^{\mathbf{A}(t-t_0)}\mathbf{v} + e^{\mathbf{A}(t-t_0)}\dot{\mathbf{v}} = \mathbf{A}e^{\mathbf{A}(t-t_0)}\mathbf{v} + \mathbf{B}u,$$

and inverting by changing the sign of the exponent, we can solve for $\mathbf{v}$ as

$$\dot{\mathbf{v}}(t) = e^{-\mathbf{A}(t-t_0)}\mathbf{B}u(t).$$

If the input $u(t)$ is zero for $t < t_0$, we can integrate $\dot{\mathbf{v}}$ from t_0 to t to get

$$\mathbf{v}(t) = \int_{t_0}^{t} e^{-\mathbf{A}(t-t_0)}\mathbf{B}u(\tau)d\tau.$$

Thus, from (7.11), the particular solution is

$$\mathbf{x}_p(t) = e^{\mathbf{A}(t-t_0)}\int_{t_0}^{t} e^{-\mathbf{A}(t-t_0)}\mathbf{B}u(\tau)d\tau.$$

and simplifying using (7.10) we get the particular solution as

$$\mathbf{x}_p(t) = \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}u(\tau)d\tau. \tag{7.12}$$

The complete solution is the sum of (7.7) and (7.12),

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}(t-\tau)}\mathbf{B}u(\tau)d\tau. \tag{7.13}$$

The reason for solving the continuous state equation was to use it from one sample to the next to obtain a discrete-time state-space model. Therefore, let $t_0 = kT$ and $t = kT + T$, so we have

$$\mathbf{x}(kT+T) = e^{\mathbf{A}T}\mathbf{x}(kT) + \int_{kT}^{kT+T} e^{\mathbf{A}(kT+T-\tau)}\mathbf{B}u(\tau)d\tau. \tag{7.14}$$

Note that since $u(\tau)$ is continuous this result is independent of the type of hold used during the sampling period. Assuming a zero-order hold (ZOH), u is constant over a sampling interval,

$$u(\tau) = u(kT), \qquad kT \le kT + T.$$

To simplify the solution, let η be defined as

$$\eta \triangleq kT + T - \tau.$$

Then we have

$$\mathbf{x}(kT+T) = e^{\mathbf{A}T}\mathbf{x}(kT) + \int_0^T e^{\mathbf{A}\eta}d\eta\mathbf{B}u(kT). \tag{7.15}$$

If we define

$$\mathbf{\Phi} \triangleq e^{\mathbf{A}T}, \tag{7.16}$$

$$\mathbf{\Gamma} \triangleq \int_0^T e^{\mathbf{A}\eta}d\eta\mathbf{B}, \tag{7.17}$$

we have from (7.15) the discrete state equation (and accompanying output equation),

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k) \tag{7.18}$$

$$y(k) = \mathbf{C}\mathbf{x}(k) \tag{7.19}$$

which we got in Section 7.2.

7.3.3 Calculating system and output matrices

Both matrices $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ can be found using the matrix exponential series. Let the series for $n \times n$ matrix $\mathbf{\Psi}$ be

$$\mathbf{\Psi} \triangleq \mathbf{I} + \frac{\mathbf{A}T}{2!} + \frac{\mathbf{A}^2T^2}{3!} + \cdots$$

then from the definition of the matrix exponential $\mathbf{\Phi}$ is given by

$$\mathbf{\Phi} = \mathbf{I} + \mathbf{A}T\mathbf{\Psi} \tag{7.20}$$

and the $\mathbf{\Gamma}$ integral in (7.17) can be evaluated term-by-term to give

$$\mathbf{\Gamma} = \mathbf{\Psi}T\mathbf{B} \tag{7.21}$$

The (overloaded) MATLAB function `c2d` works on state-space models, too, so that is an easy way to effect the discretization in the state variable domain. We will use the '`zoh`' option.

```
SYSD = C2D(SYSC,Ts,METHOD) converts the continuous-time LTI
 model SYSC to a discrete-time model SYSD with sample time Ts.
 The string METHOD selects the discretization method among the
 following:
    'zoh'       Zero-order hold on the inputs
    'foh'       Linear interpolation of inputs (triangle appx.)
    'imp'       Impulse-invariant discretization
    'tustin'    Bilinear (Tustin) approximation
    'prewarp'   Tustin approximation with frequency prewarping.
                The critical frequency Wc (in rad/sec) is specified
                as fourth input by
                   SYSD = C2D(SYSC,Ts,'prewarp',Wc)
    'matched'   Matched pole-zero method (for SISO systems only).
 The default is 'zoh' when METHOD is omitted.
```

7.4 Control Law Design

One of the attractive features of state-space design methods is that the procedure consists of two independent steps. The first step *assumes* that we can measure all the states for control purposes, *e.g.* the state is *accessible*. In general this is not true, nevertheless we will formulate a control law with this assumption. The second step is to design a state "estimator" (sometimes referred to as an "observer") which estimates the entire state vector based on a measurement of the (accessible) output given by

$$y = \mathbf{C}\mathbf{x} + Du \qquad \text{(where } D \text{ is usually zero for physical systems)}$$

The final controller will consist of the combination of the control law and the estimator, with the control computed using the estimated states.

The most general linear state control law is simply a linear combination of all state variables, that is:

$$u = -\mathbf{K}\mathbf{x} = -\begin{bmatrix} K_1 & K_2 & \cdots & K_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{7.22}$$

This control law does not provide for a reference input to the system, hence $r = 0$, and such a system is a *regulator*.

Recall that the plant discrete state equation is

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k) \tag{7.23}$$

Substituting (7.22) into (7.23) we get

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) - \mathbf{\Gamma}\mathbf{K}\mathbf{x}(k). \tag{7.24}$$

Taking the z-transform of (7.24) produces

$$(z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K})\mathbf{X}(z) = 0$$

For a non-trivial solution for $\mathbf{X}(z)$ to exist for the above equation, the determinant must be zero, that is,

$$|z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K}| = 0 \tag{7.25}$$

Equation (7.25) is the *characteristic equation* of the plant+control law.

7.4.1 Pole Placement

We will use the method of *pole placement*, since our control law has as many unknown parameters (the K_i), we are able to place the closed-loop poles (eigenvalues) arbitrarily. Note that this

places a burden on the designer to select reasonable closed-loop pole locations. Unlike transform-based design—which often involves iteration—the full state feedback, pole placement approach has guaranteed success.

Let the desired pole locations be

$$z_i = p_1, p_2, \ldots, p_n$$

then the characteristic equation is

$$\alpha_c(z) = (z - p_1)(z - p_2)\cdots(z - p_n) = 0 \tag{7.26}$$

Since (7.25) and (7.26) are both characteristic equations of the controlled system, they must be identical. Since equation (7.26) is known (we pick the pole locations) and equation (7.25) is symbolic, we can equate coefficients of like powers of z in both equations, and n simultaneous equations for the K_i will result.

Example

Consider the plant given by

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

which represents a pure inertia. With two poles at the origin, this plant is unstable. Suppose we wish to design a control law for this plant so the poles have

$$\begin{aligned} \zeta &= 0.707 \\ \omega_n &= 10 \text{ rad/sec} \end{aligned}$$

which corresponds to poles at $s = -7.07 \pm j7.07$.

Pick a sample period of $T = 0.1$ second, this is about six times the desired natural frequency. Letting the state vector be

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

the state equations are

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned}$$

and the output is simply $y = u$, hence the continuous state-space matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, D = 0.$$

Discretizing the system at $T = 0.1$ second, we obtain

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} 0.005 \\ 0.1 \end{bmatrix}$$

Using state feedback law $u(k) = -\mathbf{K}\mathbf{x}(k)$, the controlled system state equation is

$$\mathbf{x}(k+1) = [\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K}]\mathbf{x}(k)$$

where

$$\mathbf{K} = \begin{bmatrix} K_1 & K_2 \end{bmatrix}$$

The characterstic equation of the controlled system is given by the determinant

$$|z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K}| = 0$$

and after some algebra we obtain the symbolic characterstic equation as

$$z^2 + (0.005K_1 + 0.1K_2 - 2)z + (0.005K_1 - 0.1K_2 + 1) = 0$$

The desired pole locations in the s plane can be mapped into the z plane using $z = e^{sT}$, and we obtain

$$z = 0.3749 \pm j0.3203$$

thus the numerical form of the desired characteristic polynomial is

$$\alpha_c(z) = z^2 - 0.7499z + 0.2432$$

Equating coefficients of like powers of z of both characteristic polynomials gives the two simultaneous equations

$$\begin{aligned} 0.005K_1 + 0.1K_2 &= 1.2501 \\ 0.005K_1 - 0.1K_2 &= -0.7568 \end{aligned}$$

from which we obtain

$$\begin{aligned} K_1 &= 49.33 \\ K_2 &= 10.03 \end{aligned}$$

The feedback law is negative position and velocity feedback, since state variables x_1 and x_2 are position and velocity... assuming that the output is displacement, that is!

The controlled system dynamics are

$$\mathbf{x}(k+1) = [\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K}]\mathbf{x}(k)$$

where $\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K}$ is

$$\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K} = \begin{bmatrix} 0.7533 & 0.0498 \\ -4.9331 & -0.0035 \end{bmatrix}$$

Simulation of the controlled system's response from initial condition $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ yields the plot shown in Figure 7.2.

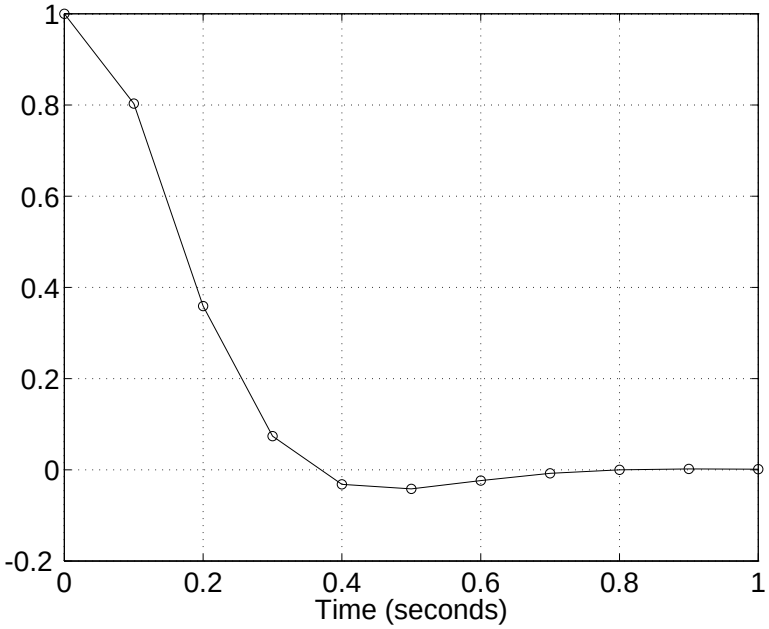


Figure 7.2: Response of controlled inertia plant from initial condition $\mathbf{x}(0) = [1 \quad 1]^T$.

7.4.2 Selecting System Pole Locations

While the selection of the desired system pole locations is highly application-specific, there are some guidelines. Generally, it is unwise to expect the closed-loop system to respond dramatically faster than the slowest component in the loop. Feedback control *can* improve system dynamics (overshoot, stability, *etc.*), but requesting closed-loop response that is much faster than the slowest component response (often the plant) may saturate one or more components. Simulation is always beneficial to explore the limits of system performance.

7.4.3 Controllability and the Control Canonical Form

The algebra for finding the state feedback gain matrix $\mathbf{K}$ is especially simple if the system matrices happen to be in *control canonical* form. Recall for a third order system (with no feed-through term b_0), this form was:

$$\mathbf{\Phi}_c = \begin{bmatrix} a_1 & a_2 & a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{\Gamma}_c = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{C}_c = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}. \tag{7.27}$$

The characteristic polynomial of equation (7.27) is $z^3 - a_1 z^2 - a_2 z - a_3$. The key idea here is that the elements of the first row of $\mathbf{\Phi}_c$ are exactly the coefficients of the characteristic polynomial of the system. If we now form the closed-loop system matrix $\mathbf{\Phi}_c - \mathbf{\Gamma}_c\mathbf{K}$, we find

$$\mathbf{\Phi}_c - \mathbf{\Gamma}_c\mathbf{K} = \begin{bmatrix} a_1 - K_1 & a_2 - K_2 & a_3 - K_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{7.28}$$

By inspection, the characteristic equation of (7.28) is

$$z^3 - (a_1 - K_1)z^2 - (a_2 - K_2)z - (a_3 - K_3) = 0$$

Thus, if the desired pole locations result in the characteristic equation

$$z^3 - \alpha_1 z^2 - \alpha_2 z - \alpha_3 = 0,$$

the necessary values for the control gains are

$$K_1 = a_1 - \alpha_1, \qquad K_2 = a_2 - \alpha_2, \qquad K_3 = a_3 - \alpha_3. \tag{7.29}$$

This leads to a design method: Given an arbitrary $(\mathbf{\Phi}, \mathbf{\Gamma})$, and a desired characteristic equation $\alpha(z) = 0$, convert (by redefinition of the state vector) to control canonical form $(\mathbf{\Phi}_c, \mathbf{\Gamma}_c)$, and solve for the gain by (7.29). Since this gain is for states in the control canonical form, we must express the result back in terms of the original state vector.

Is it always possible to find the control canonical form? Yes, almost always, but there are certain pathological systems called "uncontrollable" for which no control will give arbitrary root locations. These systems have certain modes or subsystems which are unaffected by the control.

If all roots are distinct, the system can be diagonalized using the matrix of eigenvectors as discussed in the previous chapter. The system is then

$$\mathbf{x}(k+1) = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \\ \vdots \\ \Gamma_n \end{bmatrix} u(k) \tag{7.30}$$

and explicitly exhibits the criterion for controllability: when in this form, no element of $\mathbf{\Gamma}$ can be zero. If any $\mathbf{\Gamma}$ element were zero, no control would influence that mode directly and the associated state would remain uncontrolled.

7.4.4 Ackermann's Rule and a Test for Controllability

There is a mathematical test for controllability, although the details will not be presented here. The test is:

The *controllability matrix* $\mathcal{C}$ must be non-singular, where

$$\mathcal{C} = \left[\begin{array}{cccccc} \mathbf{\Gamma} & \mathbf{\Phi\Gamma} & \mathbf{\Phi}^2\mathbf{\Gamma} & \cdots & \mathbf{\Phi}^{n-1}\mathbf{\Gamma} \end{array} \right].$$

If $\mathcal{C}$ is non-singular, there is a convenient formula derived by Ackermann (derivation not shown) which may be used to find the state feedback gain matrix $\mathbf{K}$,

$$\mathbf{K} = \left[\begin{array}{cccccc} 0 & \cdots & 0 & 1 \end{array} \right] \left[\begin{array}{cccccc} \mathbf{\Gamma} & \mathbf{\Phi\Gamma} & \mathbf{\Phi}^2\mathbf{\Gamma} & \cdots & \mathbf{\Phi}^{n-1}\mathbf{\Gamma} \end{array} \right]^{-1} \alpha_c(\mathbf{\Phi}). \tag{7.31}$$

$\alpha_c(z)$ is just the desired system characteristic polynomial obtained from

$$\alpha_c(z) = (z - p_1)(z - p_2)\cdots(z - p_n)$$

Note that in equation (7.31) matrix $\mathbf{\Phi}$ is the argument of the characteristic polynomial α_c.

Example

Consider the dual pendula on a cart shown in Figure 7.3. If the position of the cart u can be

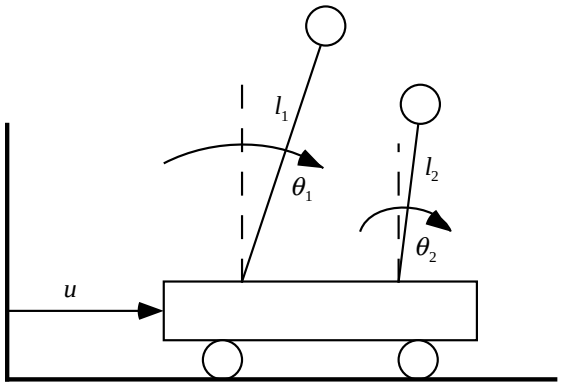


Figure 7.3: Dual pendula on cart.

specified, can both pendula be balanced? That is, is this system controllable? The equations of motion for the pendula are

$$\ddot{\theta}_1 = \frac{g}{l_1}\theta_1 - \frac{1}{l_1}\ddot{u}$$
$$\ddot{\theta}_2 = \frac{g}{l_2}\theta_2 - \frac{1}{l_2}\ddot{u}$$

If we choose state vector

$$\mathbf{x} = \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \right] = \left[\begin{array}{c} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \end{array} \right]$$

and let the input be $\ddot{u}$ for simplicity, we have system matrix $\mathbf{A}$ and input matrix $\mathbf{B}$ as

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ g/l_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & g/l_2 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ -1/l_1 \\ 0 \\ -1/l_2 \end{bmatrix}$$

The controllability matrix $\mathcal{C}$ for continuous systems is exactly the same as for discrete,

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \cdots \end{bmatrix}$$

and $\mathcal{C}$ for the dual pendula system is

$$\mathcal{C} = \begin{bmatrix} 0 & -1/l_1 & 0 & -g/l_1^2 \\ -1/l_1 & 0 & -g/l_1^2 & 0 \\ 0 & -1/l_2 & 0 & -g/l_2^2 \\ -1/l_2 & 0 & -g/l_2^2 & 0 \end{bmatrix}$$

It can be shown that if $l_1 = l_2$ the rank of $\mathcal{C}$ is 3, but if $l_1 \neq l_2$ the rank of $\mathcal{C}$ is 4. Hence if the two pendula are the same length, this system is uncontrollable.

7.4.5 MATLAB Tools

If numerical values are available, here are two MATLAB functions that can assist control law design: `place` and `ctrb`:

place. Ackermann's rule is invoked by function `place`:

```
PLACE  Pole placement technique

    K = PLACE(A,B,P)  computes a state-feedback matrix K such that
    the eigenvalues of  A-B*K  are those specified in vector P.
    No eigenvalue should have a multiplicity greater than the
    number of inputs.

    [K,PREC,MESSAGE] = PLACE(A,B,P)  returns PREC, an estimate of how
    closely the eigenvalues of A-B*K match the specified locations P
    (PREC measures the number of accurate decimal digits in the actual
    closed-loop poles).  If some nonzero closed-loop pole is more than
    10% off from the desired location, MESSAGE contains a warning
    message.
```

ctrb. Form the controllability matrix.

```
CTRB  Compute the controllability matrix.

   CO = CTRB(A,B) returns the controllability matrix [B AB A^2B ...].

   CO = CTRB(SYS) returns the controllability matrix of the
   state-space model SYS with realization (A,B,C,D).  This is
   equivalent to CTRB(sys.a,sys.b).
```

Controllability may be checked by finding the `rank` of the controllability matrix $\mathcal{C}$. If $\mathcal{C}$ has less than full rank, the system is uncontrollable.

7.4.6 Poles and Zeros

It is worthwhile to consider the poles and zeros of a system using the state-space model. Consider the plant state and output equations

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k) \\ y(k) &= \mathbf{C}\mathbf{x}(k), \end{aligned}$$

which may be z-transformed, giving

$$\begin{aligned} [z\mathbf{I} - \mathbf{\Phi}]\mathbf{X}(x) &= \mathbf{\Gamma}U(z) \\ Y(z) &= \mathbf{C}\mathbf{X}(z) \end{aligned}$$

From these two equations we can solve for the transfer function

$$\frac{Y(z)}{U(z)} = \mathbf{C}[z\mathbf{I} - \mathbf{\Phi}]^{-1}\mathbf{\Gamma}.$$

Pole locations are values of z such that the plant can have an output with no input, this is just the natural response. This can only happen if

$$|z\mathbf{I} - \mathbf{\Phi}| = 0,$$

which is the characteristic equation of the plant. Roots of this equation are the poles. We have used this approach in solving for the control gains.

Zeros are values of z such that the output y is zero with a non-zero input and state vector $\mathbf{x}$. This can be expressed in matrix form,

$$\begin{bmatrix} z\mathbf{I} - \mathbf{\Phi} & -\mathbf{\Gamma} \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{X}(z) \\ U(z) \end{bmatrix} = \begin{bmatrix} 0 \\ Y(z) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

For a non-zero state $\mathbf{X}(z)$ and input $U(z)$, the following determinant must be zero:

$$\begin{vmatrix} z\mathbf{I} - \mathbf{\Phi} & -\mathbf{\Gamma} \\ \mathbf{C} & 0 \end{vmatrix} = 0 \tag{7.32}$$

Then equation (7.32) will yield the zeros. Note that with pole placement we have no control of the eventual locations of the system zeros.

Homework Problem Set 1

Problem 1.

For the example of Section 7.4, where $G(s) = 1/s^2$, compute $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ using the series expansions of equations (7.20) and (7.21). Verify with the example.

Problem 2.

Design a state feedback control law for the DC motor + inertia of Chapter 6, Problem 1. Pick desired system pole locations consistent with the physical limitations of the system. Simulate the system's response to an initial condition.

7.4.7 More on Controllability

We stated in Section 7.4.4 that a system was controllable if the matrix

$$\mathcal{C} \triangleq \left[\begin{array}{cccc} \mathbf{\Gamma} & \mathbf{\Phi\Gamma} & \cdots & \mathbf{\Phi}^{n-1}\mathbf{\Gamma} \end{array} \right]$$

is nonsingular. Let us examine controllability a little more deeply. Consider the statement

> The system $(\mathbf{\Phi}, \mathbf{\Gamma})$ is controllable if for every $\mathbf{x}_i$ and $\mathbf{x}_f$ there is a finite N and a set of controls $u(0), u(1) \ldots u(N)$ such that if the system has state $\mathbf{x}_i$ at $k = 0$, it is forced to state $\mathbf{x}_f$ at $k = N$.

The system state equation is

$$\mathbf{x}(k+1) = \mathbf{\Phi x}(k) + \mathbf{\Gamma}u(k).$$

Solving this equation for a few steps, we find

$$\begin{aligned} \mathbf{x}(1) &= \mathbf{\Phi x}_i + \mathbf{\Gamma}u(0) \\ \mathbf{x}(2) &= \mathbf{\Phi x}(1) + \mathbf{\Gamma}u(1) \\ &= \mathbf{\Phi}^2\mathbf{x}_i + \mathbf{\Phi\Gamma}u(0) + \mathbf{\Gamma}u(1) \\ &\vdots \\ \mathbf{x}(N) &= \mathbf{\Phi}^N\mathbf{x}_i + \left[\begin{array}{cccc} \mathbf{\Gamma} & \mathbf{\Phi\Gamma} & \cdots & \mathbf{\Phi}^{N-1}\mathbf{\Gamma} \end{array} \right] \left[\begin{array}{c} u(N-1) \\ \vdots \\ u(0) \end{array} \right] \end{aligned}$$

If $\mathbf{x}(N)$ is to equal $\mathbf{x}_f$, then we must be able to solve the equations

$$\left[\begin{array}{cccc} \mathbf{\Gamma} & \mathbf{\Phi\Gamma} & \cdots & \mathbf{\Phi}^{N-1}\mathbf{\Gamma} \end{array} \right] \left[\begin{array}{c} u(N-1) \\ u(N-2) \\ \vdots \\ u(0) \end{array} \right] = \mathbf{x}_f - \mathbf{\Phi}^N\mathbf{x}_i$$

for the control sequence $u(k)$.

The "coefficient" matrix is just our controllability matrix $\mathcal{C}$, and it has n rows and N columns. If $N < n$ we cannot find a solution for general $\mathbf{x}_f$. And if $N > n$ we add additional columns, and solution existence is not readily apparent. But note that $\mathbf{\Gamma}$ can be factored out of $\mathcal{C}$ to the right, we are left with increasing powers of $\mathbf{\Phi}$, as

$$\left[\begin{array}{cccc} \mathbf{I} & \mathbf{\Phi} & \mathbf{\Phi}^2 & \cdots \end{array} \right].$$

But the Cayley-Hamilton theorem says (in a nutshell) that a matrix $\mathbf{\Phi}$ satisfies its own characteristic polynomial, this can be rearranged to say that $\mathbf{\Phi}^n$ is a linear combination of lower powers of $\mathbf{\Phi}$, thus the additional columns if $N > n$ do not increase the rank. Therefore we have a solution for the control sequence if and only if the rank of $\mathcal{C}$ is n.

7.5 State Estimator Design

The state feedback control law as used in the previous section presumed that the state was accessible. In fact, this is not likely to be true, and the missing portion of the state (or the entire state) needs to be constructed from the measured output. If the state vector is $\mathbf{x}$, the state estimate will be $\hat{\mathbf{x}}$, and the control law will use the estimate, $u = -\mathbf{K}\hat{\mathbf{x}}$.

NOTE: In this section we make a distinction between output y and *measurable* output y_m. The two are usually the same, but there could be a situation in which they are different. Measurable output y_m has with it a corresponding $\mathbf{C}_m$ output matrix.

7.5.1 Prediction Estimator

Since we have a model of the plant dynamics, one way to estimate the state is just use this model,

$$\hat{\mathbf{x}}(k+1) = \mathbf{\Phi}\hat{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) \tag{7.33}$$

where $\hat{\mathbf{x}}$ denotes the state estimate. The input $u(k)$ is known, as are $\mathbf{\Phi}$ and $\mathbf{\Gamma}$, so if the state estimate can be accurately initialized, $\hat{\mathbf{x}}(0) = \mathbf{x}(0)$, this should work. Let's see...

Figure 7.4 shows this "open-loop" estimator. Define the estimator error by

$$\tilde{\mathbf{x}} \stackrel{\Delta}{=} \mathbf{x} - \hat{\mathbf{x}} \tag{7.34}$$

and it can be shown by subtracting (7.33) from the plant state equation (7.2) that

$$\tilde{\mathbf{x}}(k+1) = \mathbf{\Phi}\tilde{\mathbf{x}}(k). \tag{7.35}$$

This says that the dynamics of the estimator error are identical to those of the uncontrolled plant! If there is any initial error in the state estimate, that error will propagate with dynamics identical to those of the uncontrolled plant. Needless to say, this is undesirable.

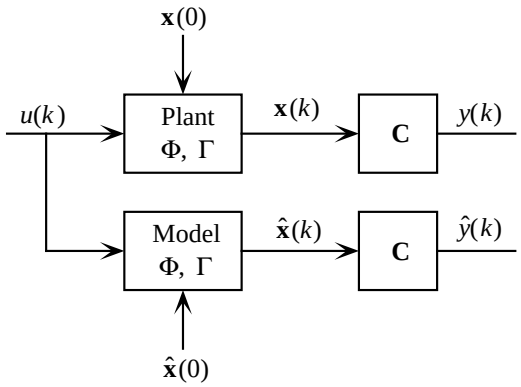


Figure 7.4: Open-loop estimator.

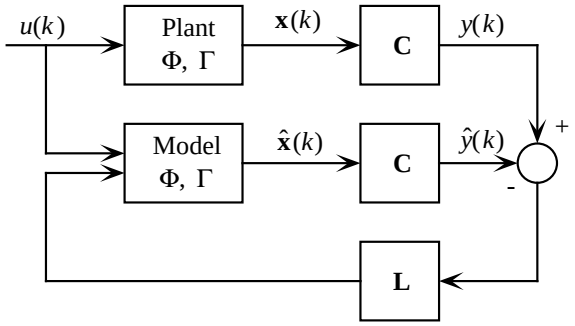


Figure 7.5: Closed-loop estimator.

The solution is to use the difference between the measured output y and the estimated output $\hat{y}$ to correct the estimator. This is shown in Figure 7.5. The estimated output is $\hat{y}_m(k) = \mathbf{C_m}\hat{\mathbf{x}}(k)$, and the difference equation for the closed-loop estimator is

$$\hat{\mathbf{x}}(k+1) = \mathbf{\Phi}\hat{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) + \mathbf{L}[y_m(k) - \mathbf{C_m}\hat{\mathbf{x}}(k)] \tag{7.36}$$

This is called a *prediction estimator* because the state estimate $\hat{\mathbf{x}}(k+1)$ is one sample ahead of the measurement $y_m(k)$. Matrix $\mathbf{L}$ contains the estimator feedback gains which will (hopefully) improve estimator performance.

The difference equation for estimator error can again be found by subtracting equation (7.36) from (7.2), and we get

$$\tilde{\mathbf{x}}(k+1) = [\mathbf{\Phi} - \mathbf{LC_m}]\tilde{\mathbf{x}}(k) \tag{7.37}$$

The dynamics of this system are given by $[\mathbf{\Phi} - \mathbf{LC_m}]$, and if we select $\mathbf{L}$ so this estimator has a fast stable response, any initial state estimate errors will be driven to zero. Thus the state estimate $\hat{\mathbf{x}}(k)$ will converge to the actual state $\mathbf{x}(k)$.

We can find $\mathbf{L}$ the same way we found $\mathbf{K}$, by taking the z-transform of equation (7.37) to get

$$[z\mathbf{I} - \mathbf{\Phi} + \mathbf{L}\mathbf{C_m}]\tilde{\mathbf{X}}(z) = 0$$

then the symbolic estimator error characteristic equation is given by

$$|z\mathbf{I} - \mathbf{\Phi} + \mathbf{L}\mathbf{C_m}| = 0 \tag{7.38}$$

The desired poles for the estimator error dynamics will yield numerical characteristic polynomial $\alpha_e(z)$, then coefficients between the two polynomials can be equated to yield n simultaneous equations in the estimator gains L_i.

Estimator pole locations. Typically, estimator pole locations are selected so that the estimator responds two to six times faster than the controlled plant. If there is noise on the measurement, slower estimation is required.

7.5.2 Observability and Ackermann's Formula

Given a desired set of estimator poles, does gain matrix $\mathbf{L}$ exist? As with controllability, it usually exists, but it may not. The key idea is whether by only a measurement of the output y we can deduce the behavior of the entire state vector $\mathbf{x}$. If all internal modes are "connected" to the output, the system is observable.

Ackermann's formula can be used for estimator gain matrix computation, it is

$$\mathbf{L} = \alpha_e(\mathbf{\Phi}) \begin{bmatrix} \mathbf{C_m} \\ \mathbf{C_m\Phi} \\ \mathbf{C_m\Phi}^2 \\ \vdots \\ \mathbf{C_m\Phi}^{n-1} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{7.39}$$

In equation (7.39) we find the *observability matrix* $\mathcal{O}$ to be

$$\mathcal{O} = \begin{bmatrix} \mathbf{C_m} & \mathbf{C_m\Phi} & \mathbf{C_m\Phi}^2 & \cdots & \mathbf{C_m\Phi}^{n-1} \end{bmatrix}. \tag{7.40}$$

This matrix must be full rank (nonsingular) for the system to be controllable.

7.5.3 MATLAB Tools

If we take the transpose of the estimator error equation system matrix $\mathbf{\Phi} - \mathbf{L}\mathbf{C_m}$, we get $\mathbf{\Phi}^T - \mathbf{C_m}^T\mathbf{L}^T$, which is the same form as system matrix $\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K}$, so if we substitute $\mathbf{\Phi}^T$ for $\mathbf{\Phi}$, $\mathbf{C_m}^T$ for $\mathbf{\Gamma}$, and $\mathbf{L}^T$ for $\mathbf{K}$, we can use the same approach for estimator gain computation as for control gains. Specifically, MATLAB function `place` may be used. Design of a prediction estimator using MATLAB will be shown in the next section.

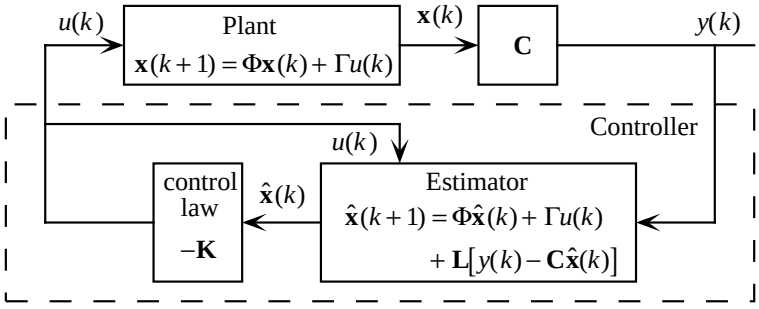


Figure 7.6: Control law plus estimator.

7.6 Regulator: Control Law plus Estimator

The state feedback control law and state estimator taken together comprise a controller. Since there is as yet no reference input, this controller is properly called a *regulator*. A diagram of this system is shown in Figure 7.6. The plant equation is

$$\mathbf{x}(k+1) = \mathbf{\Phi x}(k) + \mathbf{\Gamma}u(k)$$

with control law

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

hence the controlled plant is given by

$$\mathbf{x}(k+1) = \mathbf{\Phi x}(k) - \mathbf{\Gamma K}\hat{\mathbf{x}}(k). \tag{7.41}$$

Since estimator error is $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$, we have $\hat{\mathbf{x}} = \mathbf{x} - \tilde{\mathbf{x}}$, and the controlled plant can be expressed as

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Phi x}(k) - \mathbf{\Gamma K}[\mathbf{x}(k) - \tilde{\mathbf{x}}(k)] && (7.42)\\ &= \mathbf{\Phi x}(k) - \mathbf{\Gamma K x}(k) + \mathbf{\Gamma K}\tilde{\mathbf{x}}(k) && (7.43) \end{aligned}$$

Recall the prediction estimator error equation:

$$\tilde{\mathbf{x}}(k+1) = [\mathbf{\Phi} - \mathbf{LC_m}]\tilde{\mathbf{x}}(k) \tag{7.44}$$

Using both (7.43) and (7.44) we can write a description of the complete system as

$$\begin{bmatrix} \tilde{\mathbf{x}}(k+1) \\ \mathbf{x}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} - \mathbf{LC_m} & 0 \\ \mathbf{\Gamma K} & \mathbf{\Phi} - \mathbf{\Gamma K} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}(k) \\ \mathbf{x}(k) \end{bmatrix}. \tag{7.45}$$

The controlled system state vector is the state estimator error vector $\tilde{\mathbf{x}}$ plus the plant state vector $\mathbf{x}$. (Note that the state vector and estimator error vector are *decoupled*). This is only one possible choice. We could also use $\mathbf{x}$ and $\hat{\mathbf{x}}$, in which case the controlled system model would be different. This is left as a homework problem.

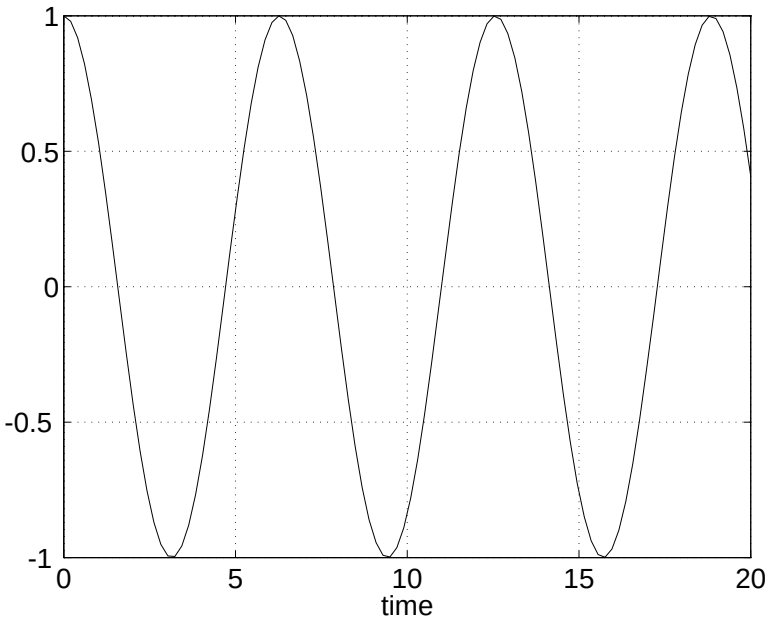


Figure 7.7: Response of undamped harmonic oscillator from initial state.

Example

In this example a control law and estimator will be designed for an undamped 1 rad/sec harmonic oscillator. The plant transfer function is

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

with differential equation $\ddot{y} + y = u$. Using state vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

we have plant matrices

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, D = 0.$$

The response of this plant from initial condition $(1, 0)$ is shown in Figure 7.7.

Specifications. Increase the damping to critical, keeping the same natural frequency of 1 rad/sec. Thus the desired pole locations will be

$$s = -1, -1$$

Sampling period. The controlled system will have natural frequency of 1 rad/sec, or about 1/6 Hz. Pick sampling frequency to be approximately six times this frequency, or 1 Hz. Hence $T = 1$.

Plant Discretization. Using this T, the continuous plant can be defined, then discretized using MATLAB:

```
>> A = [0 1;-1 0];
>> B = [0;1];
>> C = [1 0];
>> D = 0;
>>
>> plant_c = ss(A,B,C,D);
>> T = 1;
>> plant_d = c2d(plant_c,T)

a =
             x1       x2
   x1   0.5403   0.8415
   x2  -0.8415   0.5403

b =
           u1
   x1  0.4597
   x2  0.8415

c =
       x1  x2
   y1   1   0

d =
       u1
   y1   0

Sampling time: 1
Discrete-time model.
```

Note that the state-space matrices can be extracted from LTI model `plant_d` by:

```
>> Phi = plant_d.A

Phi =

    0.5403    0.8415
   -0.8415    0.5403
```

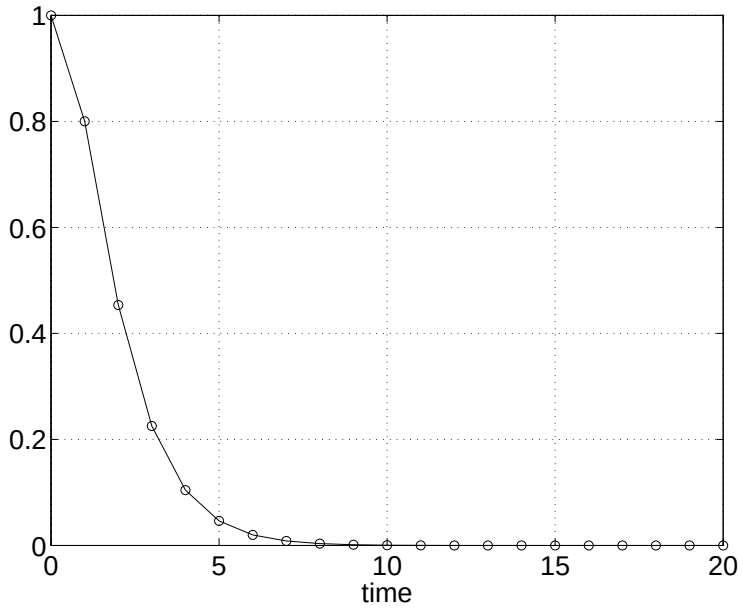


Figure 7.8: Response of controlled plant from initial state.

Control Law. The desired z-plane pole locations are found using the mapping $z = e^{sT}$,

```
>>control_s_poles = [-1 -1]';
>>control_z_poles = exp(control_s_poles*T)

control_z_poles = 0.3679
                  0.3679
```

It is a numerical anomaly of MATLAB function `place` that poles of multiplicity greater than the number of inputs are not allowed. One pole can therefore be offset slightly, and the gain vector $\mathbf{K}$ is

```
>>control_z_poles(1) = control_z_poles(1)+0.0001;
>>K = place(Phi,Gamma,control_z_poles)
place: ndigits= 19

K = -0.5655    0.7186
```

The control law $u = -\mathbf{Kx}$ is seen to be positive position feedback and negative velocity feedback. As a check, simulate the controlled system response from initial condition $(1, 0)$. This is shown in Figure 7.8.

Prediction Estimator. Select the natural frequency of the prediction estimator to be five times the controlled plant, again with critical damping, thus the estimator pole locations are $s = -5, -5$, and

```
>>estimator_s_poles = [-5 -5]';
>>estimator_z_poles = exp(estimator_s_poles*T)

estimator_z_poles = 0.0067
                    0.0067
```

Again, to use `place` we have to offset one pole slightly. In estimator design, remember that $\mathbf{\Phi}^T$ and $\mathbf{C}^T$ must be used (and that $\mathbf{L}^T$ is returned):

```
>>estimator_z_poles(1) = estimator_z_poles(1)+0.0001;
>>L = place(Phi',C',estimator_z_poles);
place: ndigits= 20
>>L = L'

L =  1.0670
    -0.5032
```

It is difficult to attach physical significance to the estimator gains. Remember, reduced physical insight *is* a disadvantage of the state-space design methodology.

Controlled System. One model of the controlled system is given in equation (7.45). The following MATLAB code will build up the "system" matrix for equation (7.45), called `sys`:

```
>>Sys = [Phi-L*C zeros(2);
         Gamma*K Phi-Gamma*K]

Sys = -0.5267    0.8415         0         0
      -0.3383    0.5403         0         0
      -0.2599    0.3303    0.8002    0.5111
      -0.4758    0.6047   -0.3657   -0.0644
```

System Response. The controlled system state used in equation (7.45) is the estimator error state plus the plant state. If we start the complete system with state estimate equal to the actual state, estimator error will always be zero. Thus it is more illustrative to start the system with some estimator error present. Consider initial condition given by

$$\mathbf{x}_0 = \begin{bmatrix} \tilde{\mathbf{x}}_0 \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

which imposes initial estimator errors of 1 unit on both states.

The controlled system (plant + control law + estimator) has no reference input... it is homogeneous. Nevertheless, for MATLAB simulation purposes an input matrix of proper dimension must

be present. Also, an output matrix must be present of the proper dimension must be present. The controlled system is twice the order of the plant. The following will simulate the system's response from the stated initial condition:

```
>> Input = [0 0 0 0]';  % Dummy input matrix
>> Output = [0 0 1 0];  % Output is original y
>> x0 = [1 1 1 0]';     % Apply initial condition
>> sys_ss = ss(Sys,Input,Output,D,T);  % Create SS discrete-time controlled sys
>> [y,t,x] = initial(sys_ss,x0);  % Get response
```

A plot of plant output y *vs* t is shown in Figure 7.9. It is very close to the ideal response of Figure 7.8, although a little different due to the estimator transient. The estimator error $\tilde{x}_1$ and $\tilde{x}_2$ can also be

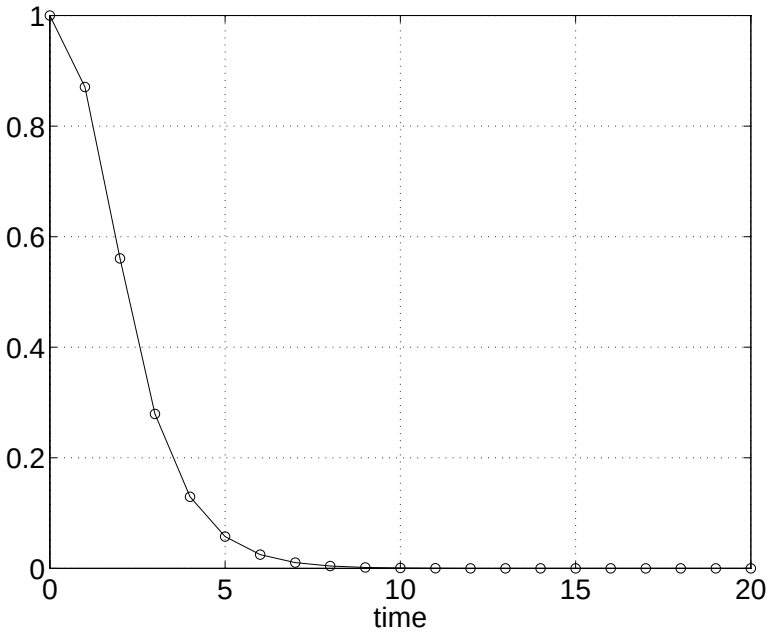


Figure 7.9: y response of plant + control law + estimator from initial state $(1, 1, 1, 0)$.

plotted, and is shown in Figure 7.10. As expected, it converges to zero very quickly.

```
>>x1_error = x(:,1);
>>x2_error = x(:,2);
```

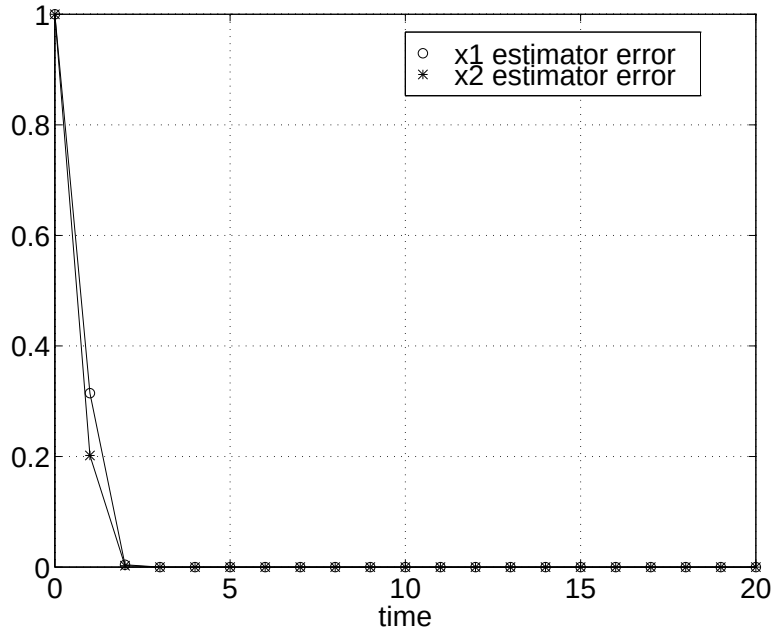


Figure 7.10: Estimator error response.

Homework Problem Set 2

Problem 1.

Derive the symbolic matrix equations like (7.45) but using

$$\mathbf{x}_{system} = \begin{bmatrix} \mathbf{x} \\ \hat{\mathbf{x}} \end{bmatrix}$$

as the controlled system state vector.

Problem 2.

Design a prediction state estimator for the DC motor + inertia of Problem 2 of the previous homework set. Assume the measurement is $y = \theta_l$. Pick desired estimator pole locations consistent with the control law you determined (or the instructor's). Perform appropriate simulations to demonstrate the successful behavior of your system.

7.6.1 Controller Transfer Function

The controller contained within the dashed line of Figure 7.6 accepts a measurement y and generates a control u. It is interesting to find the transfer function of the controller and verify it using transform methods. Note that the controller transfer function need not be determined during a state-space design effort.

We will consider a controller based on the prediction estimator. The prediction estimator update equation (7.36) is

$$\hat{\mathbf{x}}(k+1) = \mathbf{\Phi}\hat{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) + \mathbf{L}[y(k) - \mathbf{C}\hat{\mathbf{x}}(k)] \tag{7.46}$$

and the accompanying control law is

$$u(k) = -\mathbf{K}\hat{\mathbf{x}}(k) \tag{7.47}$$

These two equations define the controller. To find the transfer function, substitute (7.47) in (7.46) and express

$$\hat{\mathbf{x}}(k+1) = [\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K} - \mathbf{L}\mathbf{C}]\hat{\mathbf{x}}(k) + \mathbf{L}y(k). \tag{7.48}$$

Equation (7.48) is the *state equation* for the controller, while equation (7.47) is the *output equation* for the controller. Apply the z-transform to (7.48) to obtain

$$z\hat{\mathbf{X}} = [\mathbf{\Phi} - \mathbf{\Gamma}\mathbf{K} - \mathbf{L}\mathbf{C}]\hat{\mathbf{X}} + \mathbf{L}Y,$$

and solve for $\hat{\mathbf{X}}$ as

$$\hat{\mathbf{X}} = [z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K} + \mathbf{L}\mathbf{C}]^{-1}\mathbf{L}Y.$$

Substitution of $\hat{\mathbf{X}}$ into $U = -\mathbf{K}\hat{\mathbf{X}}$ yields the controller transfer function

$$\frac{U(z)}{Y(z)} = D(z) = -\mathbf{K}[z\mathbf{I} - \mathbf{\Phi} + \mathbf{\Gamma}\mathbf{K} + \mathbf{L}\mathbf{C}]^{-1}\mathbf{L} \tag{7.49}$$

MATLAB Tools. With controller state equation (7.48) and output equation (7.47) the "$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$" matrices can be identified and used in the `ss2tf` function to find the `[num,den]` transfer function form. This will be shown in the following example.

Example

Consider the undamped oscillator of the previous sections. We discretized this system to obtain

$$\mathbf{\Phi} = \begin{bmatrix} 0.5403 & 0.8415 \\ -0.8415 & 0.5403 \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} 0.4597 \\ 0.8415 \end{bmatrix}$$

and found control law and prediction estimator gains

$$\mathbf{K} = \begin{bmatrix} -0.5655 & 0.7186 \end{bmatrix}, \mathbf{L} = \begin{bmatrix} 1.0670 \\ -0.5032 \end{bmatrix}$$

From (7.48) and (7.47) the equivalent state-space matrices for the controller are

$$\begin{aligned} \mathbf{A}_c &\rightarrow \mathbf{\Phi} - \mathbf{\Gamma K} - \mathbf{LC} \\ \mathbf{B}_c &\rightarrow \mathbf{L} \\ \mathbf{C}_c &\rightarrow -\mathbf{K} \\ D_c &\rightarrow 0 \end{aligned}$$

and the MATLAB script below yields the controller transfer function.

```
>>Ac = Phi-Gamma*K-L*C;
>>Bc = L;
>>Cc = K;
>>Dc = 0;
>>[numc,denc] = ss2tf(Ac,Bc,Cc,Dc)

numc =  0   -0.9650   0.1156

denc =  1.0000    0.3311   -0.0531

>>[zeros_c,poles_c,K_c] = tf2zp(numc,denc)

zeros =  0.1198

poles =  -0.4494
          0.1183

K =  0.9650
```

Thus the controller transfer function is

$$D(z) = \frac{-0.965(z - 0.1198)}{(z - 0.1183)(z + 0.4494)}$$

A transfer function for the plant can be found by

```
>>[nump,denp] = ss2tf(Phi,Gamma,C,D)

nump =   0    0.4597    0.4597

denp =  1.0000   -1.0806    1.0000

>>[zeros_p,poles_p,K_p] = tf2zp(nump,denp)

zeros_p =  -1

poles_p =  0.5403 + 0.8415i
           0.5403 - 0.8415i

K_p =  0.4597
```

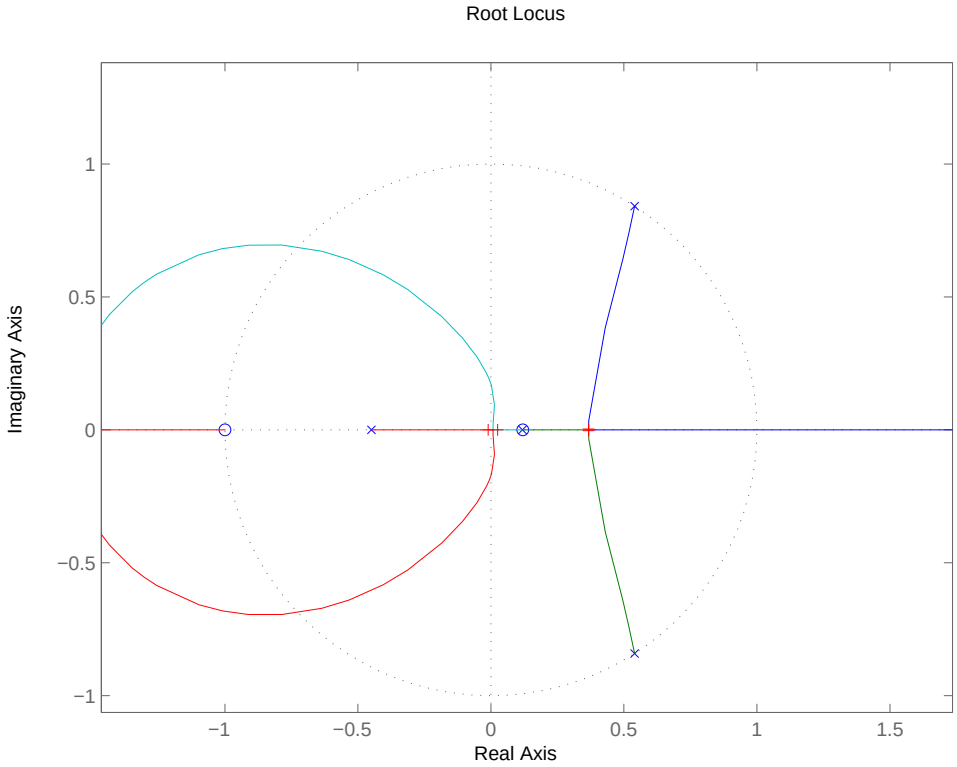


Figure 7.11: Root locus of oscillator plant and controller.

The plant transfer function is

$$G(z) = \frac{0.4597(z+1)}{z - 0.5403 \pm j0.8415}$$

Root Locus. Let the controller gain of 0.965 be replaced by a variable parameter $0.965K$. Now perform a root locus analysis of the plant with K as the parameter. This is shown in Figure 7.11 along with the closed-loop poles when $K = 1$. The poles should be exactly at $z = 0.3679(2)$ and $z = 0.0067$, the desired poles of the control law and estimator error dynamics! They're pretty close...science is wonderful!

7.7 Current and Reduced-Order Estimators

The prediction estimator as presented in Section 7.5.1 is only one possibility for state estimation. Two other approaches will be shown in this Section.

7.7.1 Current Estimator

Inspection of equation (7.36) shows that the state estimate at sample $k+1$, $\hat{\mathbf{x}}(k+1)$, depends on measured system output $y(k)$. Thus there is an inherent one sample delay between the new estimated state and the measurement used to correct it. Common sense tells us that it may improve estimation if we use the *current* value of measurement $y(k+1)$ to correct state estimate $\hat{\mathbf{x}}(k+1)$.

One would only use a current estimator if the control computer were fast enough to perform the estimator calculations very quickly, since this computation time is not modeled, and will therefore have an effect on the system performance. If computational latency is significant, a prediction estimator may be a better choice.

To analyze the current estimator we will separate the estimate *prediction* from the estimate *correction*. The prediction equation is simply the plant model,

$$\check{\mathbf{x}}(k+1) = \mathbf{\Phi}\hat{\mathbf{x}}(k) + \mathbf{\Gamma}u(k) \tag{7.50}$$

where $\check{\mathbf{x}}$ is the predicted state before correction.

The correction equation is

$$\hat{\mathbf{x}}(k+1) = \check{\mathbf{x}}(k+1) + \mathbf{L}[y(k+1) - \mathbf{C}\check{\mathbf{x}}(k+1)] \tag{7.51}$$

where $\hat{\mathbf{x}}(k+1)$ is the new state estimate.

Substitution of (7.50) into (7.51) results in the current estimator equation,

$$\hat{\mathbf{x}}(k+1) = [\mathbf{\Phi} - \mathbf{LC\Phi}]\hat{\mathbf{x}}(k) + [\mathbf{\Gamma} - \mathbf{LC\Gamma}]u(k) + \mathbf{L}y(k+1) \tag{7.52}$$

Note that the coefficient matrices in (7.52) are constant, hence the current estimator of (7.52) is not more computationally complex than the prediction estimator equation of (7.36).

The estimator error $\tilde{\mathbf{x}} \triangleq \mathbf{x} - \hat{\mathbf{x}}$ as before, and subtracting equation (7.52) from plant equation (7.2) yields the current estimator error equation,

$$\tilde{\mathbf{x}}(k+1) = [\mathbf{\Phi} - \mathbf{LC\Phi}]\tilde{\mathbf{x}}(k) \tag{7.53}$$

The poles of the current estimator error equation are given by

$$|z\mathbf{I} - \mathbf{\Phi} + \mathbf{LC\Phi}| = 0,$$

and the symbolic characteristic obtained from this determinant may be equated with the desired $\alpha_e(z) = (z - p_1)(z - p_2)\ldots$ to solve for current estimator gain matrix $\mathbf{L}$.

Ackermann's rule. Equation (7.53) for the current estimator is of the same form as (7.37) for the prediction estimator, except $\mathbf{C}$ is replaced by $\mathbf{C\Phi}$. Thus Ackermann's rule for the current estimator

will be (7.39) with $\mathbf{C}$ replaced by $\mathbf{C\Phi}$. Thus for the current estimator,

$$\mathbf{L} = \alpha_e(\mathbf{\Phi}) \begin{bmatrix} \mathbf{C\Phi} \\ \mathbf{C\Phi}^2 \\ \mathbf{C\Phi}^3 \\ \vdots \\ \mathbf{C\Phi}^n \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{7.54}$$

System Model. Combining the current estimator given by (7.52) and the control law $u(k) = -\mathbf{K}\hat{\mathbf{x}}(k)$, the controlled system model using composite state of $\mathbf{x}$ and $\hat{\mathbf{x}}$ is

$$\begin{bmatrix} \mathbf{x}(k+1) \\ \hat{\mathbf{x}}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} & -\mathbf{\Gamma K} \\ \mathbf{LC\Phi} & \mathbf{\Phi} - \mathbf{\Gamma K} - \mathbf{LC\Phi} \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \hat{\mathbf{x}}(k) \end{bmatrix}. \tag{7.55}$$

MATLAB Tools. The MATLAB `place` function can be used to design a current estimator. In forming a control law, we worked with system matrix $\mathbf{\Phi} - \mathbf{\Gamma K}$, where $\mathbf{K}$ was unknown. For the current estimator, we have matrix $\mathbf{\Phi} - \mathbf{LC\Phi}$, with $\mathbf{L}$ to be determined. Transposing, we have $\mathbf{\Phi}^T - \mathbf{\Phi}^T\mathbf{C}^T\mathbf{L}^T$, which is in the same form as $\mathbf{\Phi} - \mathbf{\Gamma K}$. MATLAB function `place` can be used directly, as shown in the following example.

Example

Design a current estimator for use with the harmonic oscillator plant of the previous example, where a prediction estimator was used. Place the estimator poles at the same locations, which were both at $z = 0.0067$, although one pole was offset by 0.0001 because of MATLAB's numerical anomaly.

The $\mathbf{\Phi}$ and $\mathbf{C}$ matrices were

```
Phi =  0.5403    0.8415
      -0.8415    0.5403

C =  1     0
```

The current estimator gain matrix $\mathbf{L}$ is given by

```
estimator_z_poles = 0.0067
                    0.0067

>>estimator_z_poles(1) = estimator_z_poles(1)+.0001;

>>L = place(Phi',Phi'*C',estimator_z_poles);
place: ndigits= 20

>>L = L'
L =  1.0000
     0.6260
```

so the current estimator gain matrix is

$$\mathbf{L} = \begin{bmatrix} 1 \\ 0.6260 \end{bmatrix}$$

7.7.2 Reduced-Order Estimators

If some state variables can be measured, why estimate them? This is the reason behind reduced-order estimators. It may be necessary to estimate only a subset of the state vector. Although, when the measurement contains significant noise, a full state estimator (being inherently a low-pass filter) will tend to reduce the noise and may give better performance.

Assume the measurement is a scalar y, and denote this by x_a. Let $\mathbf{x}_b$ be the portion to be estimated. The plant state equation of (7.2) may be partitioned as

$$\begin{bmatrix} x_a(k+1) \\ \mathbf{x}_b(k+1) \end{bmatrix} = \begin{bmatrix} \Phi_{aa} & \mathbf{\Phi}_{ab} \\ \mathbf{\Phi}_{ba} & \mathbf{\Phi}_{bb} \end{bmatrix} \begin{bmatrix} x_a(k) \\ \mathbf{x}_b(k) \end{bmatrix} + \begin{bmatrix} \Gamma_a \\ \mathbf{\Gamma}_b \end{bmatrix} u(k) \tag{7.56}$$

with output equation

$$y(k) = \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_a(k) \\ \mathbf{x}_b(k) \end{bmatrix}. \tag{7.57}$$

The portion of (7.56) which describes the dynamics of the unmeasured states is

$$\mathbf{x}_b(k+1) = \mathbf{\Phi}_{bb}\mathbf{x}_b(k) + \underbrace{\mathbf{\Phi}_{ba}x_a(k) + \mathbf{\Gamma}_b u(k)}_{\text{known "input"}} \tag{7.58}$$

where the terms grouped may be considered as a known "input" into the $\mathbf{x}_b$ dynamics. Now express the x_a portion of (7.56) as

$$\underbrace{x_a(k+1) - \Phi_{aa}x_a(k) - \Gamma_a u(k)}_{\text{known "measurement"}} = \mathbf{\Phi}_{ab}\mathbf{x}_b(k). \tag{7.59}$$

The grouped quantity on the left of (7.59) is known (or can be measured). Equations (7.58) and (7.59) are effectively state and output equations for $\mathbf{x}_b$, in the form of (7.2).

Therefore, the prediction estimator equations may be used to estimate $\mathbf{x}_b$ with the following substitutions:

$$\begin{aligned} \mathbf{x} &\leftarrow \mathbf{x}_b, \\ \mathbf{\Phi} &\leftarrow \mathbf{\Phi}_{bb}, \\ \mathbf{\Gamma}u(k) &\leftarrow \mathbf{\Phi}_{ba}x_a(k) + \mathbf{\Gamma}_b u(k), \\ y(k) &\leftarrow x_a(k+1) - \Phi_{aa}x_a(k) - \Gamma_a u(k), \\ \mathbf{C} &\leftarrow \mathbf{\Phi}_{ab}. \end{aligned}$$

Then the reduced-order estimator equation is

$$\begin{aligned}\hat{\mathbf{x}}_b(k+1) &= \mathbf{\Phi}_{bb}\hat{\mathbf{x}}_b(k) + \mathbf{\Phi}_{ba}x_a(k) + \mathbf{\Gamma}_b u(k) \\ &\quad +\mathbf{L}[x_a(k+1) - \Phi_{aa}x_a(k) - \Gamma_a u(k) - \mathbf{\Phi}_{ab}\hat{\mathbf{x}}_b(k)].\end{aligned} \tag{7.60}$$

Reduced-order estimator error may be found by subtracting (7.60) from (7.58) to get

$$\tilde{\mathbf{x}}_b(k+1) = [\mathbf{\Phi}_{bb} - \mathbf{L}\mathbf{\Phi}_{ab}]\tilde{\mathbf{x}}_b(k) \tag{7.61}$$

and gain matrix $\mathbf{L}$ may be selected as before using either the roots of

$$|z\mathbf{I} - \mathbf{\Phi}_{bb} + \mathbf{L}\mathbf{\Phi}_{ab}| = \alpha_e(z) \tag{7.62}$$

to be in desired locations, or by Ackermann's formula

$$\mathbf{L} = \alpha_e(\mathbf{\Phi}_{bb}) \begin{bmatrix} \mathbf{\Phi}_{ab} \\ \mathbf{\Phi}_{ab}\mathbf{\Phi}_{bb} \\ \mathbf{\Phi}_{ab}\mathbf{\Phi}_{bb}^2 \\ \vdots \\ \mathbf{\Phi}_{ab}\mathbf{\Phi}_{bb}^{n-2} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{7.63}$$

Example

Design a reduced-order estimator for use with the harmonic oscillator plant. Assume that position is measured, but velocity must be estimated. Thus

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \text{measured position state } y \\ \text{velocity to be estimated} \end{bmatrix}$$

and the plant partioning is

$$\begin{bmatrix} \Phi_{aa} & \Phi_{ab} \\ \Phi_{ba} & \Phi_{bb} \end{bmatrix} = \begin{bmatrix} 0.5403 & 0.8415 \\ -0.8415 & 0.5403 \end{bmatrix}, \begin{bmatrix} \Gamma_a \\ \Gamma_b \end{bmatrix} = \begin{bmatrix} 0.4597 \\ 0.8415 \end{bmatrix}$$

Estimator gain $\mathbf{L}$ is a scalar, hence there is only one estimator root to pick. We will place this pole at $z = 0.0067$ as with the estimator poles before. From (7.62) the estimator error characteristic equation is

$$z - 0.5403 + 0.8415L = z - 0.0067$$

where we have placed the pole at 0.0067. Solving for L yields

$$L = 0.6341$$

The estimator equation is given by (7.61), thus

$$\begin{aligned}\hat{x}_2(k+1) &= 0.5403\hat{x}_2(k) - 0.8415y(k) + 0.8415u(k) \\ &\quad +0.6341[y(k+1) - 0.5403y(k) - 0.4597u(k) - 0.8415\hat{x}_2(k)]\end{aligned}$$

which may be reduced to

$$\hat{x}_2(k+1) = 0.0067\hat{x}_2(k) + 0.55u(k) - 1.1841y(k) + 0.6341y(k+1)$$

Note that this equation is like a current estimator in that a measurement of y at sample $k+1$ is used to generate the estimate $\hat{x}_2$ at sample $k+1$. To reduce computational delay, this calculation would be done in two steps, one before the sample, and one after sampling. Before sampling,

$$\hat{x}_2 = 0.0067\hat{x}_2(k) + 0.55u(k) - 1.1841y(k)$$

and after sampling this would be updated

$$\hat{x}_2 = \hat{x}_2 + 0.6341y(k+1)$$

The same approach of performing as much computation before sampling should generally be followed, regardless of the type of estimator used.

7.8 Adding a Reference Input

Up to this point, the controller has not included a reference input. Such systems are called regulators. For most applications, a reference input (or set point) is needed. For simplicity, the reference input will first be added to the plant plus full state feedback control law, but no estimator. Then the estimator will be added, and we will examine the effect of the reference input on estimator error.

The approach will be to find the steady-state state vector that corresponds to the steady-state output we want, then find a way to introduce the reference input so it will produce that state.

7.8.1 Reference Input with Full State Feedback

First consider a plant using full state feedback control law $u(k) = -\mathbf{K}\mathbf{x}(k)$. Introduce the reference input *via* matrix $\mathbf{N}_x$ to generate *reference state* $\mathbf{x}_r$, as shown in Figure 7.12. Even though we have dealt primarily with SISO systems in this chapter, in this section we will consider MIMO systems for generality. One portion of the final project will require MIMO analysis and design.

In adding the reference input, we want to find $\mathbf{N}_x$ such that steady-state system output $\mathbf{y}_r$ (which is determined by output matrix $\mathbf{C}_r$) is at the desired reference value. Note that output $\mathbf{y}_r$ might

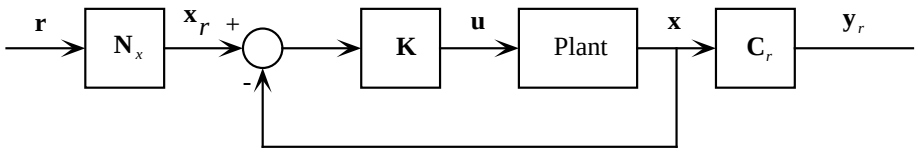


Figure 7.12: Block diagram of reference input added to plant with full state feedback.

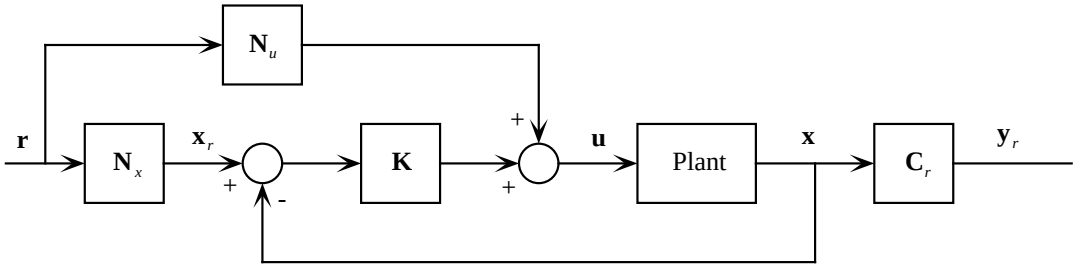


Figure 7.13: Reference input added to plant with full state feedback plus feedforward.

not be the same quantity (I will define $y = \mathbf{C_e x}$) that has been measured and fed to the estimator in previous sections. Now "estimator" output matrix $\mathbf{C_e}$ produces the measurement the estimator uses. Of course, it is quite possible that $\mathbf{C}_r = \mathbf{C_e}$. Therefore, for reference input matrix $\mathbf{N}_x$,

$$\mathbf{N}_x \mathbf{r} = \mathbf{x}_r \tag{7.64}$$

and the control law is

$$\mathbf{u} = -\mathbf{K}(\mathbf{x} - \mathbf{x}_r). \tag{7.65}$$

If the plant is Type 1 or higher, and $\mathbf{r}$ is a position input, the steady-state error will be zero, and the final state will equal the (static) reference state,

$$\mathbf{x}(\infty) \triangleq \mathbf{x}_{ss} = \mathbf{x}_r$$

If the plant is Type 0, state feedback alone will not increase the Type, hence there will be some steady-state error, and $\mathbf{x}_{ss} \neq \mathbf{x}_r$.

This is not all we really want, however. We would like the output to follow the reference input, or $\mathbf{y}_r = \mathbf{r}$. If the plant is not Type 0, since state feedback will not add integral action, a feedforward term which adds a steady-state component of control action $\mathbf{u}$ must be added. This term is

$$\mathbf{u}_{ss} = \mathbf{N}_u \mathbf{r}_{ss} \tag{7.66}$$

and it is shown in Figure 7.13. Feedforward action, although conceptually valid, is rarely used. Instead, integral action is typically used to increase system Type. The reason feedforward action is rarely used is that it is an open-loop correction, and its effectiveness relies on the accuracy of the plant model... and there is always modeling error. Integral control action, in contrast, is closed-loop in nature, and will be more robust to modeling errors. While this is a heuristic argument, it is valid. If time permits, a method for adding integral action to state feedback control will be covered in this chapter.

Our steady-state requirements are

$$\mathbf{N}_x \mathbf{r}_{ss} = \mathbf{x}_{r_{ss}} = \mathbf{x}_{ss}$$

and

$$\mathbf{C}_r \mathbf{x}_{ss} = \mathbf{y}_{r_{ss}} = \mathbf{r_{ss}} \tag{7.67}$$

which may be combined to yield

$$\mathbf{C}_r\mathbf{N}_x\mathbf{r_{ss}} = \mathbf{r_{ss}}$$

thus

$$\boxed{\mathbf{C}_r\mathbf{N}_x = \mathbf{I}} \tag{7.68}$$

Now we are assuming steady-state, hence

$$\mathbf{x}(k+1) = \mathbf{\Phi x}(k) + \mathbf{\Gamma u}(k) \longrightarrow \mathbf{x}_{ss} = \mathbf{\Phi x}_{ss} + \mathbf{\Gamma u}_{ss}$$

or

$$(\mathbf{\Phi} - \mathbf{I})\mathbf{x}_{ss} + \mathbf{\Gamma u}_{ss} = \mathbf{0}$$

and from (7.67) and (7.66),

$$(\mathbf{\Phi} - \mathbf{I})\mathbf{N}_x\mathbf{r_{ss}} + \mathbf{\Gamma N}_u\mathbf{r_{ss}} = \mathbf{0}$$

which reduces to

$$(\mathbf{\Phi} - \mathbf{I})\mathbf{N}_x + \mathbf{\Gamma N}_u = \mathbf{0} \tag{7.69}$$

Equations (7.68) and (7.69) may be combined into a single matrix equation,

$$\begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{\Gamma} \\ \mathbf{C}_r & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{N}_x \\ \mathbf{N}_u \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{7.70}$$

If the number of plant inputs in $\mathbf{u}$ and desired outputs in $\mathbf{y}_r$ are the same, then the coefficient matrix in equation (7.70) is square, and the solution is

$$\begin{bmatrix} \mathbf{N}_x \\ \mathbf{N}_u \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{\Gamma} \\ \mathbf{C}_r & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{7.71}$$

Example

The undamped oscillator plant considered in previous examples is of Type 0. The relevant system matrices for that system are

$$\mathbf{\Phi} = \begin{bmatrix} 0.5403 & 0.8415 \\ -0.8415 & 0.5403 \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} 0.4597 \\ 0.8415 \end{bmatrix}, \mathbf{C}_r = \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

and the reference input matrices are found to be

```
>>inv([Phi-eye(2) Gamma ; C 0]) * [0;0;1]

ans =  1.0000
      -0.0000
       1.0000
```

hence

$$\mathbf{N}_x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, N_u = 1.$$

As expected, feedforward term N_u is nonzero for this Type 0 plant.

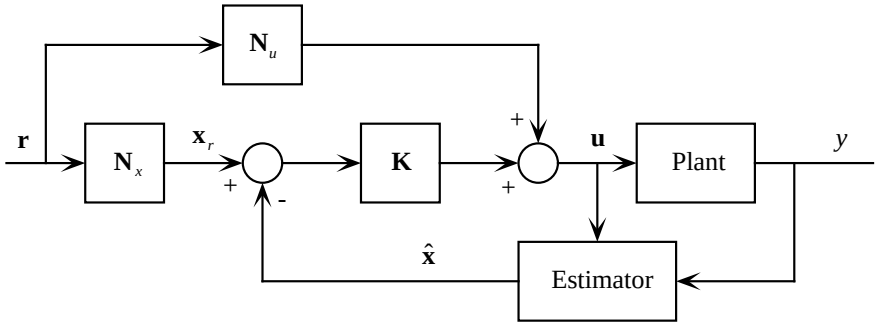


Figure 7.14: Reference input added to plant with estimator.

7.8.2 Reference Input with an Estimator

The same procedure can be applied to the case where a state estimator is used instead of a full state measurement. One must make sure to apply the same control $\mathbf{u}$ to the estimator as well as the plant, as shown in Figure 7.14. Thus equation (7.48) should *not* be used directly, since it was based on a control $\mathbf{u}$ which did not include a reference input. Look at Figure 7.5, where we first formulated the estimator, and verify that the estimator *must* get the same control force $\mathbf{u}$ as the plant. Rather, equation (7.36) should be used, with $\mathbf{u}(k)$ given by

$$\mathbf{u}(k) = -\mathbf{K}[\hat{\mathbf{x}}(k) - \mathbf{x}_r(k)] + \mathbf{N}_u\mathbf{r}(k). \tag{7.72}$$

It can be shown that in the absence of modeling errors the addition of the reference input will not excite any estimator error.

Homework Problem Set 3

Problem 1.

Derive the system equations for a system with a prediction estimator and a reference input. Assume a SISO system, and use state vector

$$\mathbf{x}_{sys} = \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \end{bmatrix}$$

and show that the reference input does not excite any estimator error $\tilde{\mathbf{x}}$.

Problem 2.

Introduce a motor position reference input to the DC motor + inertia of Chapter 7, Problem 2. Use control law and prediction estimator from the homework solutions. Simulate the system's response

to a rest-to-rest *load* maneuver of 10 degrees in 0.5 sec, where $\theta_l(t)$ is given by a cubic polynomial $\theta_l(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$. This is a more realistic reference input to track than a step, for example.

7.9 Uniqueness of Solution—The MIMO Case

The previous approach to the introduction of a reference input is general with respect to the number of inputs and outputs. However, the solution may not be unique. Consider a system of the following dimension:

- m plant inputs—$\mathbf{u}$ is $m \times 1$
- n states—$\mathbf{x}$ is $n \times 1$
- p outputs—$\mathbf{y}_r$ and $\mathbf{r}$ are $p \times 1$

The dimension of equation (7.70) is

$$\underbrace{\begin{bmatrix} \overbrace{\mathbf{\Phi} - \mathbf{I}}^{n\times n} & \overbrace{\mathbf{\Gamma}}^{n\times m} \\ \underbrace{\mathbf{C}_r}_{p\times n} & \underbrace{\mathbf{0}}_{p\times m} \end{bmatrix}}_{(n+p)\times(n+m)} \underbrace{\begin{bmatrix} \overbrace{\mathbf{N}_x}^{n\times p} \\ \underbrace{\mathbf{N}_u}_{m\times p} \end{bmatrix}}_{(n+m)\times p} = \underbrace{\begin{bmatrix} \overbrace{\mathbf{0}}^{n\times p} \\ \underbrace{\mathbf{I}}_{p\times p} \end{bmatrix}}_{(n+p)\times p}. \tag{7.73}$$

From equation (7.73) it is clear that if $m = p$ the coefficient matrix is square and may be directly inverted. However, if $m \neq p$ this is not true. But there is still a solution, there is just not a *unique* solution. The problem is analogous to the classic linear algebra problem $\mathbf{Ax} = \mathbf{b}$.

The mathematics of why the following approach works will be treated somewhat in Chapter 8 on System Identification. Suffice it to say that we achieve a best least squares fit. When $\mathbf{A}$ is not square, multiply by $\mathbf{A}^T$ to get $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$. Now $\mathbf{A}^T\mathbf{A}$ may be inverted, and the solution is $\mathbf{x} = [\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{x}$. The reference input problem is identical, and if we let

$$\begin{bmatrix} \mathbf{\Phi} - \mathbf{I} & \mathbf{\Gamma} \\ \mathbf{C}_r & \mathbf{0} \end{bmatrix} \triangleq \mathbf{\Psi}$$

the solution when there are differing numbers of plant inputs and controlled outputs is

$$\begin{bmatrix} \mathbf{N}_x \\ \mathbf{N}_u \end{bmatrix} = \left[\mathbf{\Psi}^T\mathbf{\Psi}\right]^{-1}\mathbf{\Psi}^T \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \tag{7.74}$$

The resulting $\mathbf{N}_x$ and $\mathbf{N}_u$ can then be used with the methods of Section 7.8.

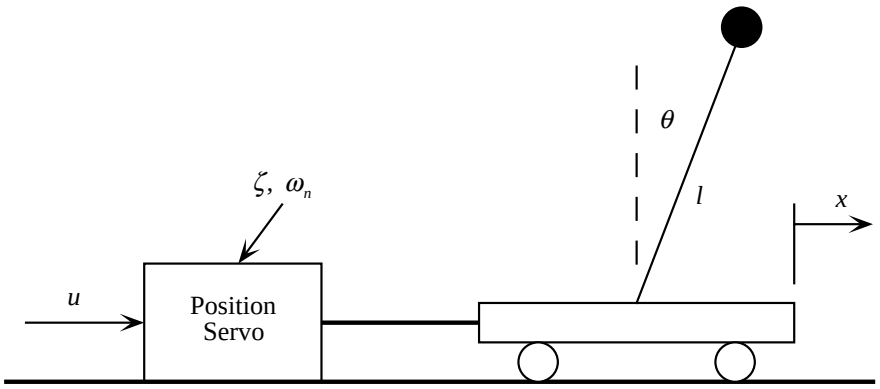


Figure 7.15: Pendulum and cart driven by position servo.

7.10 Another Example—Stick Balancer

Consider the inverted pendulum on a cart driven by a position servo as shown in Figure 7.15. The equation of motion of the pendulum is

$$\ddot{x} + l\ddot{\theta} - g\theta = 0$$

and the cart is

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = \omega_n^2 u$$

where u is the input to the position servo.

Eliminate $\ddot{x}$ using both equations of motion, and one gets

$$\ddot{\theta} = \frac{g}{l}\theta + \frac{\omega_n^2}{l}x + \frac{2\zeta\omega_n}{l}\dot{x} - \frac{\omega_n^2}{l}u$$

Pick the state vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix}$$

then the following system and input matrices obtain:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ g/l & 0 & \omega_n^2/l & 2\zeta\omega_n/l \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\omega_n^2 & -2\zeta\omega_n \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ -\omega_n^2/l \\ 0 \\ \omega_n^2 \end{bmatrix}$$

Numerical Values. To go any farther we need some numerical values. For the pendulum,

$$\begin{aligned} g &= 9.8 \text{ m/sec}^2 \\ l &= 0.3 \text{ m} \end{aligned}$$

and for the position servo

$$\begin{aligned} \zeta &= 0.707 \\ \omega_n &= 20 \text{ rad/sec} \end{aligned}$$

and we get plant matrices

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 32.7 & 0 & 1333 & 94.3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -400 & -28.3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ -1333 \\ 0 \\ 400 \end{bmatrix}$$

Since the position servo natural frequency is about 3 Hz, we won't require any higher frequency response from the controlled system. Thus a sampling frequency of 25 Hz ($T = 0.04$) should be adequate. Discretizing the plant with this T yields

$$\mathbf{\Phi} = \begin{bmatrix} 1.0262 & 0.0403 & 0.7240 & 0.0619 \\ 1.3181 & 1.0262 & 29.0507 & 2.7782 \\ 0 & 0 & 0.7839 & 0.0215 \\ 0 & 0 & -8.6106 & 0.1751 \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} -0.7240 \\ -29.0507 \\ 0.2161 \\ 8.6106 \end{bmatrix}$$

Step Response of Plant. As a "reality check," find the response of the pendulum to a step position input of 1 cm (0.01 m). Selecting $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$ to yield θ as the output, we obtain the pendulum angular response shown in Figure 7.16. The pendulum angle of Figure 7.16 looks believable, it is in the negative direction which is consistent with our intuition for a positive base displacement.

Plant Controllability. Before continuing on with the controller development, check that the plant is controllable with the given input. By adjusting the base position, it seems obvious that it is possible to control both base position and velocity, and pendulum position and velocity. Nevertheless, intuition can be in error, and it is best to check.

```
>>Ctr = ctrb(Phi,Gamma);
>>rank(Ctr)

ans =  4
```

As expected, the plant is controllable.

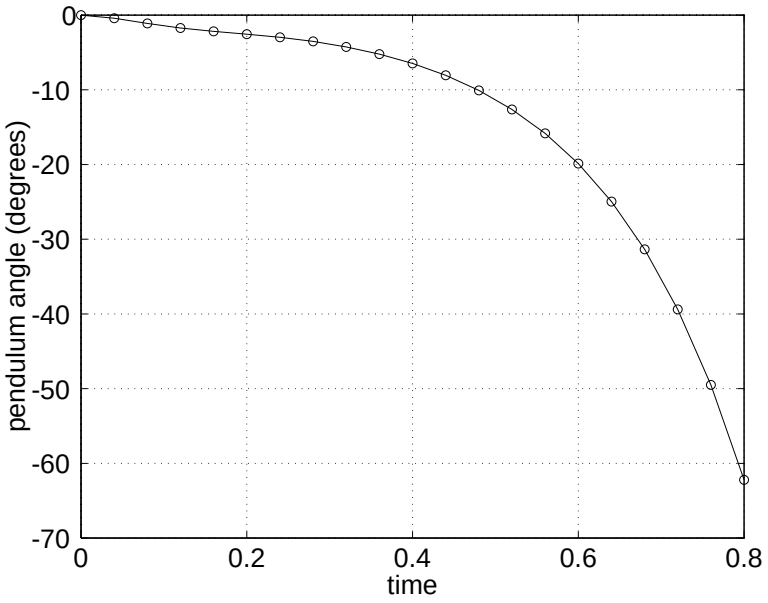


Figure 7.16: Response of pendulum to 1 cm step servo displacement.

Control Law. In keeping with Butterworth filter guidelines, equally space four poles at radius ω_n around the negative real axis in the s-plane, hence at angles of $\pm 22.5^o, \pm 67.5^o$ from the negative real axis. Thus the four poles are at $s = -7.6537 \pm j18.4776$, $s = -18.4776 \pm j7.6537$. In the z plane these four poles are $z = 0.5442 \pm j0.4960$, $z = 0.4553 \pm j0.1439$.

Using Matlab `place`, the state feedback gain is

$$\mathbf{K} = \begin{bmatrix} -3.1715 & -0.3895 & -8.5452 & -1.2489 \end{bmatrix}$$

Plant Observability Assume that only a measurement of pendulum angle is available (measurement of cart position is probably available, since its position is controlled), can we estimate the full state vector from only a measurement of pendulum angle? The $\mathbf{C}$ matrix will be

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

and a check of the observability matrix shows

```
>>C = [1 0 0 0];
>>rank(ctrb(Phi',C'))

ans =  4
```

Since the observability matrix $\mathcal{O}$ has full rank, the system is observable, and we can proceed with a prediction estimator design.

Prediction Estimator Place the estimator poles like the previous control law poles, but use a higher natural frequency of $\omega_n = 50$ rad/sec. These poles are at $s = -19.1342 \pm j46.1940$ and $s = -46.1940 \pm j19.1342$. The corresponding z-plane locations are $z = -0.1272 \pm j0.4474$ and $z = 0.1136 \pm j0.1092$. Again using the `place` function, we get

$$\mathbf{L} = \begin{bmatrix} 3.0386 \\ 64.3273 \\ -0.0320 \\ 5.6954 \end{bmatrix}$$

Reference Input We wish to specify the position of the cart using the reference input, or $y_r = x$. This is a different variable than the measurement θ, so $\mathbf{C}_r \neq \mathbf{C}$, and

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

It doesn't really make sense to provide a reference input for pendulum angle θ, except for trimming, since $\theta = 0$ is the only pendulum angle that can exist in steady-state. So the number of plant inputs u and desired outputs x are the same ($m = p$), hence the coefficient matrix in equation (7.70) is square, and terms $\mathbf{N}_x$ and N_u are given by equation (7.71),

```
>>Cr = [0 0 1 0];
>>inv([Phi-eye(4) Gamma ; Cr 0])*([0;0;0;0;1])

ans = -0.0000
       0.0000
       1.0000
       0.0000
       1.0000
```

Thus

$$\mathbf{N}_x = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, N_u = 1$$

System Modeling. The complete system is modeled by the five following equations (integrated in the proper manner, of course!):

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k) \\ y(k) &= \mathbf{C}\mathbf{x}(k) \\ \hat{\mathbf{x}}(k+1) &= \mathbf{\Phi}\hat{\mathbf{x}}(k) + \mathbf{\Gamma}\mathbf{u}(k) + \mathbf{L}[y(k) - \mathbf{C}\hat{\mathbf{x}}(k)] \\ u(k) &= -\mathbf{K}[\hat{\mathbf{x}}(k) - \mathbf{x}_r(k)] + N_u r(k) \\ \mathbf{x}_r(k) &= \mathbf{N}_x r(k) \end{aligned}$$

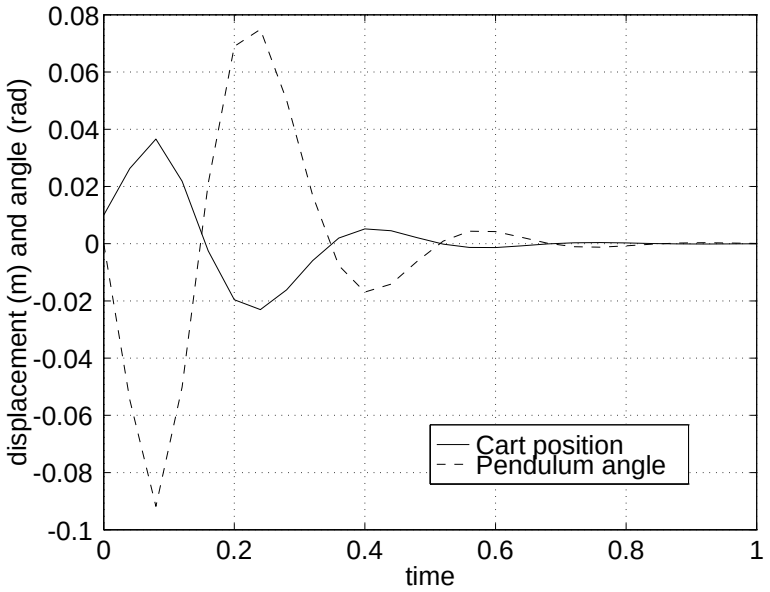


Figure 7.17: Response of cart and pendulum to 1 cm initial condition of cart (note initial motion of cart is away from equilibrium).

Initial Condition Response. First impose an initial condition on the plant and observe its return to equilibrium. Note that this controller will return all four states to equilibrium... it will balance the stick and also center the cart. Let's pick an initial offset of 0.01 m (1 cm) for the cart, with the pendulum balanced, hence

$$\mathbf{x}(0) = \begin{bmatrix} \theta_0 \\ \dot{\theta}_0 \\ x_0 \\ \dot{x}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.01 \\ 0 \end{bmatrix}$$

The response of the cart position and pendulum angle as the system returns to equilibrium is shown in Figure 7.17. Note that the initial motion of the cart is to the right, *away* from the desired center position! However, this is intuitive, since the stick was initially balanced, and to move the cart back to the left one must first move to the right to get the stick leaning to the left, then the cart can move left.

Reference Input Response. Our reference input acted on cart position, thus we can specify reference cart position and the other variables will be regulated. Choose cart motion of 0.2 m (20 cm) in 1 second (25 samples) at constant velocity. The system behavior to this reference trajectory appears in Figure 7.18, where again cart position and pendulum angle are plotted. As before, the cart moves the "wrong way" first, then moves back in the desired direction. The pendulum angle in Figure 7.18 is pretty large, approaching 0.4 radians, which is getting close to violating the "small angle" approximation in the pendulum equation of motion. Use of the small angle approximation is

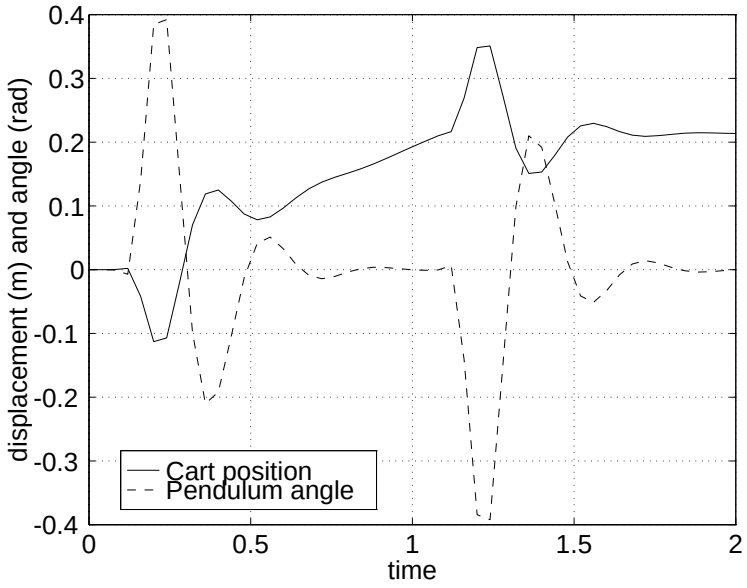


Figure 7.18: Response of cart and pendulum to reference trajectory of 0.2 m in 1 second at constant velocity.

an example of a modeling error, or unmodeled dynamics.

Chapter 8

System Identification

8.1 Introduction

To design a controller for a dynamic system it is necessary to have a model that describes the dynamic behavior of that system. There are fundamentally two approaches to finding this model:

Analytical Modeling Using basic principles... the laws of motion, electrical laws, and other physical principles, the designer obtains differential equations of motion describing the response of mechanical, electrical, thermal, fluid, and other systems.

Experimental Identification Excite the dynamic system with a given input, and record its output. Construct a model from this observed data.

While analytical modeling is almost always of some use, if for nothing more than to give insight into the expected model structure, actual physical phenomena may be too complex to permit satisfactory description using physical principles. Under these circumstances, the designer looks to experimental data, and experimental *system identification*.

This chapter will contain a very brief treatment of system identification using the least squares method. We will make the assumption that the system to be identified is linear and time-invariant, in keeping with the prior chapters.

8.2 Models and Data Organization

Let's use a second-order transform-based discrete system to illustrate the procedure. Consider

$$\frac{Y(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{8.1}$$

This transfer function and associated a_i and b_i could have been expressed in state-space form with $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ in one of the canonical forms of Chapter 7 (either control canonical or observer canonical). A canonical state-space form is a "minimal" realization... it has the minimum number of parameters, like the transfer function of equation 8.1. Note that a feedforward term b_0 was *not* included in equation 8.1 because it is virtually never seen in a physical system (unlike a controller).

The difference equation that corresponds to equation 8.1 is

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) \tag{8.2}$$

Now consider a sequence of input samples from $k = 0 \ldots N$, given by $\{u(0), u(1), \ldots, u(N)\}$. Assuming zero initial conditions on output y, the output samples are given by

$$\begin{aligned}
y(2) &= a_1 y(1) + a_2 y(0) + b_1 u(1) + b_2 u(0) \\
y(3) &= a_1 y(2) + a_2 y(1) + b_1 u(2) + b_2 u(1) \\
&\vdots \\
y(k) &= a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2) \\
&\vdots \\
y(N) &= a_1 y(N-1) + a_2 y(N-2) + b_1 u(N-1) + b_2 u(N-2)
\end{aligned} \tag{8.3}$$

Equation 8.3 can be arranged in matrix form as

$$\begin{bmatrix} y(2) \\ y(3) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} y(1) & y(0) & u(1) & u(0) \\ y(2) & y(1) & u(2) & u(1) \\ \vdots & \vdots & \vdots & \vdots \\ y(N-1) & y(N-2) & u(N-1) & u(N-2) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \tag{8.4}$$

In linear algebra form, equation (8.4) is the form $\mathbf{b} = \mathbf{A}\mathbf{x}$ or the more familiar

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{8.5}$$

where we have defined

$$\mathbf{A} \triangleq \begin{bmatrix} y(1) & y(0) & u(1) & u(0) \\ y(2) & y(1) & u(2) & u(1) \\ \vdots & \vdots & \vdots & \vdots \\ y(N-1) & y(N-2) & u(N-1) & u(N-2) \end{bmatrix}, \mathbf{x} \triangleq \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix}, \mathbf{b} \triangleq \begin{bmatrix} y(2) \\ y(3) \\ \vdots \\ y(N) \end{bmatrix}.$$

Just as in equation (8.5), in the system identification problem matrix $\mathbf{A}$ is known (inputs u and outputs y are known), vector $\mathbf{b}$ is known (outputs y are known), and vector $\mathbf{x}$ is unknown (system parameters a_i and b_i are unknown). Thus, as constructed, this is a linear algebra problem.

8.3 Least Squares Approximations

In the exact sense, the system of equations $\mathbf{Ax} = \mathbf{b}$ either has a unique solution or not. There are several reasons why there may not be a unique solution. One reason, and this is the reason that is our case, is that the dimension of $\mathbf{b}$ is larger than the dimension of $\mathbf{x}$, and there are many solutions for $\mathbf{x}$. In terms of system identification, this means that number of data samples taken is greater than the number of model parameters. This is the normal case.

So if there are many solutions, which is the best? One approach is to choose the $\mathbf{x}$ that minimizes the average error over the entire set of data. If we let $\hat{\mathbf{x}}$ be our estimate of the actual $\mathbf{x}$, we can form the error vector for the entire data set as ϵ, where

$$\epsilon \triangleq \mathbf{A}\hat{\mathbf{x}} - \mathbf{b} \tag{8.6}$$

There are many ways to define the average error over the entire data set, but the most convenient is the *sum of squares*:

$$\epsilon^2 = \epsilon^T \epsilon = (\mathbf{A}\hat{\mathbf{x}} - \mathbf{b})^T (\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}) \tag{8.7}$$

The sum of squares of the error ϵ^2 is a scalar, and is what we want to minimize. We will present two approaches to minimize ϵ^2, which yield the same result.

8.3.1 Minimization of ϵ^2 by Calculus

Multiplying out equation (8.7) yields

$$\epsilon^2 = \hat{\mathbf{x}}^T \mathbf{A}^T \mathbf{A}\hat{\mathbf{x}} - 2\mathbf{b}^T \mathbf{A}\hat{\mathbf{x}} + \mathbf{b}^T \mathbf{b}.$$

To minimize ϵ^2 take the derivative with respect to $\hat{\mathbf{x}}$ and equate to zero. Using the rules of vector calculus,

$$\frac{\partial \epsilon^2}{\partial \hat{\mathbf{x}}} = 2\hat{\mathbf{x}}^T \mathbf{A}^T \mathbf{A} - 2\mathbf{b}^T \mathbf{A} = \mathbf{0},$$

thus

$$\hat{\mathbf{x}}^T \mathbf{A}^T \mathbf{A} = \mathbf{b}^T \mathbf{A},$$

and

$$\mathbf{A}^T \mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T \mathbf{b}.$$

The solution for $\hat{\mathbf{x}}$ is given by

$$\hat{\mathbf{x}} = [\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{b} \tag{8.8}$$

The result for $\hat{\mathbf{x}}$ in equation (8.8) yields the lowest sum of squared errors.

8.3.2 Minimization of ϵ^2 by Linear Algebra

Since this is not a course in linear algebra, this section will employ heuristics *in lieu* of rigor. Again, the problem is to choose $\hat{\mathbf{x}}$ to minimize the error in the least squares sense.

In a geometrical sense, $\hat{\mathbf{x}}$ is *projecting* the columns of $\mathbf{A}$ onto the elements of $\mathbf{b}$. The distances from the elements of $\mathbf{b}$ to this projection are given by $||\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}||$, and the least squares best fit corresponds to minimizing these distances. This will occur when error vector $\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}$ is perpendicular to the column space of $\mathbf{A}$. If two vectors $\mathbf{p}$ and $\mathbf{q}$ are perpendicular, their dot product must be zero, hence $\mathbf{p}^T\mathbf{q} = 0$.

Applying this to our case, we find that the columns of $\mathbf{A}$ must be perpendicular to error vector $\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}$, hence

$$\mathbf{A}^T(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}) = \mathbf{0},$$

or

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} - \mathbf{A}^T\mathbf{b} = \mathbf{0}$$

or

$$\mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{b}$$

which leads to the same result as by calculus.

8.4 Application to System Identification

The results of the least squares approximation of Section 8.3 may be directly applied to system identification. Consider the case represented by equation (8.4), but generalized for an n^{th} order system. Again consider a data set consisting of $N+1$ measurements from $k = 0 \ldots N$. Define the following matrices, after equation (8.4):

$$\mathbf{Y} \triangleq \underbrace{\begin{bmatrix} y(n) \\ y(n+1) \\ \vdots \\ y(N) \end{bmatrix}}_{(N-n+1)\times 1} \tag{8.9}$$

$$\boldsymbol{\Psi} \triangleq \underbrace{\begin{bmatrix} y(n-1) & y(n-2) & \cdots & y(0) & u(n-1) & u(n-2) & \cdots & u(0) \\ y(n) & y(n-1) & \cdots & y(1) & u(n) & u(n-1) & \cdots & u(1) \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ y(N-1) & y(N-2) & \cdots & y(N-n) & u(N-1) & u(N-2) & \cdots & u(N-n) \end{bmatrix}}_{(N-n+1)\times 2n} \tag{8.10}$$

$$\boldsymbol{\Theta} \triangleq \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \\ b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}}_{2n \times 1} \tag{8.11}$$

With the definitions of (8.9), (8.10), and (8.11), the equation describing the input-output data and model parameters may be written as

$$\mathbf{Y} = \boldsymbol{\Psi}\boldsymbol{\Theta} \tag{8.12}$$

which is exactly in the form of $\mathbf{b} = \mathbf{A}\mathbf{x}$. Thus the development of Section 8.3 is directly applicable, and the solution for the model parameter vector $\boldsymbol{\Theta}$ is

$$\boldsymbol{\Theta} = [\boldsymbol{\Psi}^T\boldsymbol{\Psi}]^{-1}\boldsymbol{\Psi}^T\mathbf{Y}. \tag{8.13}$$

Equation (8.13) yields the best fit in the least squares sense.

Note that the order of the model is up to the designer. Selection of the order should be based on physical reasoning. Of course, the designer may perform multiple fits with different orders and select the lowest order fit that yields acceptable agreement. Selecting a model of too-high order means that you use the "extra" parameters to fit noise in the data. And remember that there is *always* quantization noise.

8.5 Practical Issues—How Well Does it Work?

Least squares is the simplest and most straightforward parametric identification method. If the system to be identified exhibits linear behavior and the input/output data is relatively noise-free, least squares will yield acceptable results. There are a host of more sophisticated methods that work better than least squares under more difficult conditions, but they are beyond the scope of this book.

8.5.1 Selection of Input

The choice of input u used in identification is important. Examine equations (8.3) for a moment, in particular the input terms involving $u(k)$ and b_i. Consider the (absurd) case where the input $u(k)$ was a constant. Then in each equation in (8.3) the b_i terms would be multiplied by the same number, and would be combined into a single quantity. We could not ever separate the b_i terms from the observed data. The constant u fails to "excite" all the dynamics of the plant. This problem has been studied extensively, and an input sequence $\{u(k)\}$ that fluctuates enough to avoid the possibility of linear combinations of the elements of model parameter vector $\boldsymbol{\Theta}$ showing up in the error is called *persistently exciting.*

Two types of inputs will be shown in this section:

- Random Input
- Random Binary Input

Random Input. This is just random noise, which may be easily generated using MATLAB. If random the harmonic content will be rich enough to be persistently exciting. Either a normal or uniform distribution is available. A normal distribution with mean of 0.0 and variance 1.0 may be created with the `randn` function. A plot of a such a normal distribution for $k = 0 \ldots 100$ is shown in Figure 8.1. Naturally the random input may have to be biased and scaled before application to a given physical system.

Random Binary Input This is a random input that fluctuates between two distinct values. It is a series of pulses with random durations. Since a step signal has infinite harmonic content, this type of input will also persistently excite the system. A random binary input is shown in Figure 8.2

Both types of random inputs may be easily generated using the MATLAB `randn` (normally-distributed random sequence) function, along with the `sign` function for producing the binary signal.

8.5.2 Quantization Noise

As briefly discussed in Section 1.2, quantization effects are always present whenever there is a conversion between the analog and digitial domains. The size of the `quantum` is dependent on the range and resolution of the conversion device. For example, an 8-bit A/D converter digitizes to 1 part in 256. If the A/D has a range of ± 5V, the voltage quantum is $q = 10/256 = 0.0391$ V. Depending on how close the analog voltage is to one of the levels of quantization, there will be more or less quantization error present. Note that in identification, quantization only affects output y, since the input u is already quantized when it is acquired by the computer.

Quantization can be either *rounding* or *truncation*. A plot of the effect of rounding is shown in Figure 8.3. In Figure 8.3, if the input is between ± 0.5, the output is zero. If the input is between 0.5 and 1.5, the output is 1, and so on. The *quantization error* fluctuates between $\pm q$ in a "sawtooth" variation.

The amount of quantization noise is also dependent on how well we use the full range of the conversion device. If our input is only ± 1 V, and the A/D has a range of ± 5 V, the effective resolution will be less than the possible resolution. If possible, the full range of conversion devices should be used.

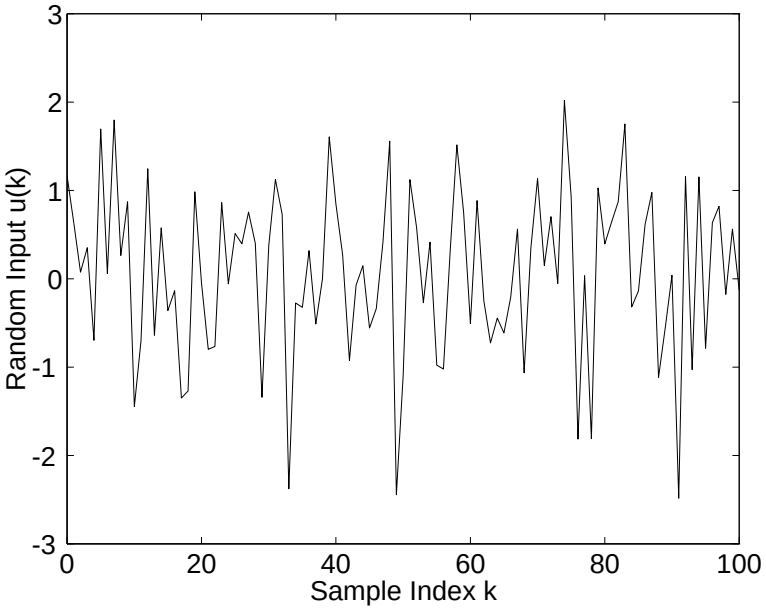


Figure 8.1: Normally-distributed random noise with mean 0.0 and variance 1.0.

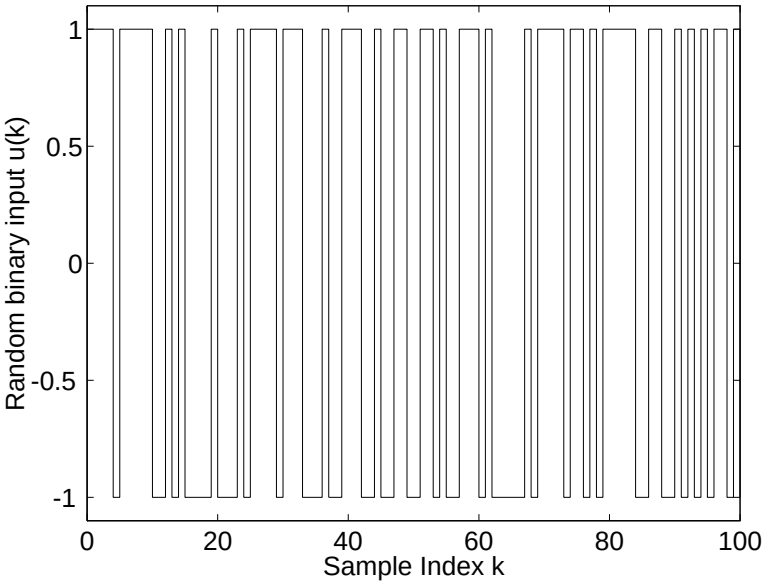


Figure 8.2: Random binary signal.

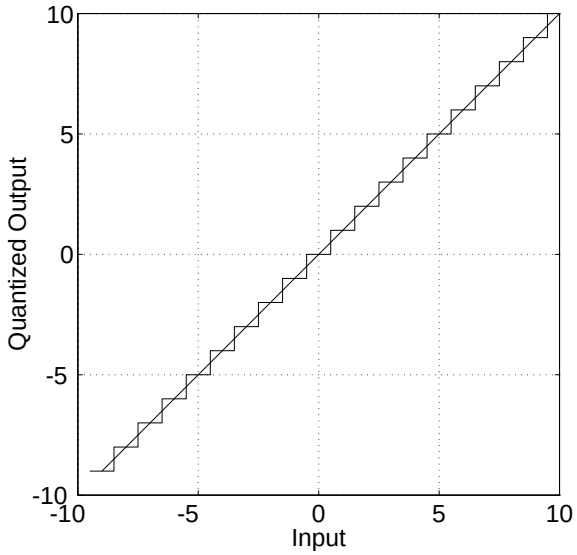


Figure 8.3: Effect of rounding with $q = 1$.

8.5.3 Identification Example

As an example, consider the third order continuous system with transfer function

$$G(s) = \frac{2000}{(s+10)(s^2+20s+200)} = \frac{2000}{(s+10)(s+10 \pm j10)}$$

The complex poles have natural frequency $\omega_n \approx 14$ rad/sec, or about 2.25 Hz, so pick sampling frequency of 20 Hz, or $T = 0.05$ sec. Preceding this $G(s)$ by a sampler and following it by a ZOH (our standard discretization procedure) yields its discrete form,

$$G(z) = \frac{0.0284z^{-1} + 0.0775z^{-2} + 0.0134z^{-3}}{1 - 1.6711z^{-1} + 1.0136z^{-2} - 0.2231z^{-3}} = \frac{0.0284(z+0.1857)(z+2.54)}{(z-0.6065)(z-0.5323 \pm j0.2908)}$$

Excitation. Excite this $G(z)$ with 101 samples of a random binary signal of amplitude ± 1 as shown in Figure 8.2. Assume that the output is quantized by an A/D with 8-bit resolution and ± 5V range.

The quantized response of this system is shown in Figure 8.4.

Least-Squares Fit. The input data $u(k)$ and output data $y(k)$ must be arranged according to equations (8.9) and (8.10) to form the $\mathbf{Y}$ and $\mathbf{\Psi}$ matrices, then the least-squares solution for model parameter vector $\mathbf{\Theta}$ is given by equation (8.13). Here $N = 100$ and $n = 3$, and $\mathbf{Y}$ and $\mathbf{\Psi}$ are constructed accordingly.

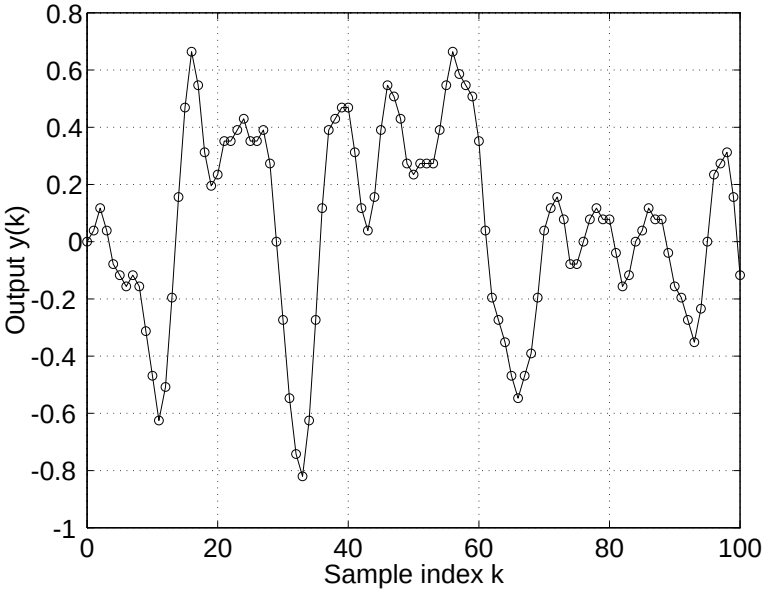


Figure 8.4: Quantized output of example $G(z)$ driven by random binary signal.

Executing this yields model parameter vector

$$\boldsymbol{\Theta} = \begin{bmatrix} 1.4067 & -0.5852 & 0.0192 & 0.0302 & 0.0865 & 0.0284 \end{bmatrix}^T$$

and corresponding estimated transfer function

$$\widehat{G}(z) = \frac{0.0302z^{-1} + 0.0865z^{-2} + 0.0284z^{-3}}{1 - 1.4067z^{-1} + 0.5852z^{-2} - 0.0192z^{-3}} = \frac{0.0302(z + 0.3784)(z + 2.4894)}{(z - 0.0358)(z - 0.6855 \pm j0.2574)}$$

Step Response. How does the step response of the model compare with the actual system? Both unit step responses are shown in Figure 8.5. While the steady-state value is somewhat inaccurate, the transient portion of the response exhibits excellent agreement.

Comments. In this example, we did *not* make good use of the range of the A/D converter, using only about ±1V range of the ±5V converter. This contributed to the inaccuracy. Even so, the $\widehat{G}(z)$ we obtained would serve pretty well for controller design.

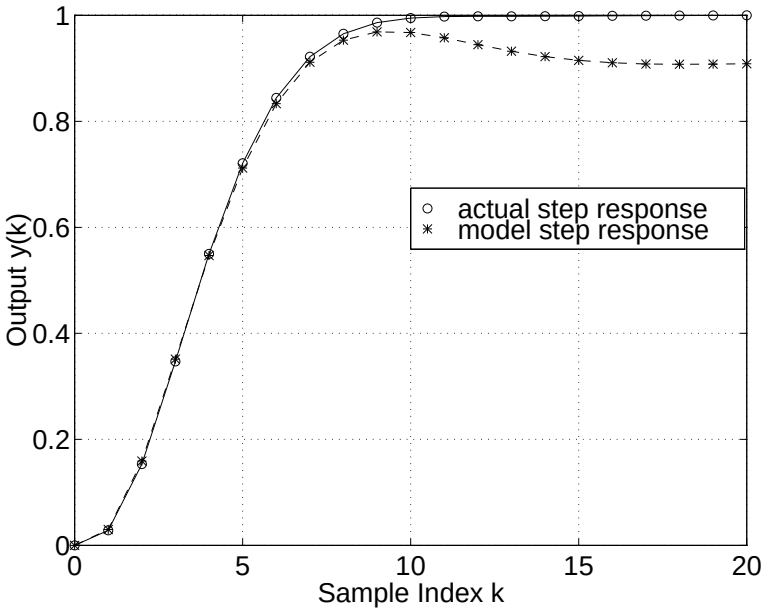


Figure 8.5: Comparison of actual and model step responses.

Homework Problems

Problem 1.

Data from a mass moving in a plane (x, y) without force are given below. Plot the points and use

t	x	y
0	-0.426	1.501
1	0.884	1.777
2	2.414	2.530
3	2.964	3.441
4	3.550	3.762
5	5.270	4.328
6	5.625	4.016
7	7.188	5.368
8	8.129	5.736
9	9.225	6.499
10	10.388	6.233

least squares to estimate the initial value of x, the initial value of y, and the velocity of the mass in x and y. Draw the curve (line) that corresponds to your estimates.

Problem 2.

Select a discrete transfer function (at least second order) and investigate identifying its parameters.

(a) Drive the system with both random and PRBS inputs.

(b) Perform least-square fits both using a model structure that is the correct order, and an order *greater* than the "actual" order.

(c) What happens when the model order is too high?