CHAPTER 5

Block Diagrams, State-Variable Models, and Simulation Methods

CHAPTER OUTLINE

CHAPTER OBJECTIVES

When you have finished this chapter, you should be able to

1. Describe a dynamic model as a block diagram.
2. Derive system transfer functions from a block diagram.
3. Convert a differential equation model into state-variable form.
4. Express a linear state-variable model in the standard vector-matrix form.
5. Apply the `ss`, `ssdata`, `tfdata`, `eig`, and `initial` functions to analyze linear models.
6. Use the MATLAB `ode` functions to solve differential equations.
7. Use Simulink to create simulations of dynamic models.

Dynamic models derived from basic physical principles can appear in several forms:

1. As a single equation (which is called the *reduced* form),
2. As a set of coupled first-order equations (which is called the *Cauchy* or *state-variable* form), and
3. As a set of coupled higher-order equations.

In Chapters 2, 3, and 4 we analyzed the response of a single equation, such as $m\ddot{x} + c\dot{x} + kx = f(t)$, and sets of coupled first-order equations, such as $\dot{x} = -5x + 7y$, $\dot{y} = 3x - 9y + f(t)$, by first obtaining the transfer functions and then using the transfer functions to obtain a single, but higher-order equation. We also obtained the response of models that consist of a set of coupled higher-order equations.

Each form has its own advantages and disadvantages. We can convert one form into another, with differing degrees of difficulty. In addition, if the model is *linear,* we can convert any of these forms into the transfer function form or a vector-matrix form, each of which has its own advantages.

GUIDE TO THE CHAPTER

This chapter has three parts. Part I is required to understand Parts II and III, but Parts II and III are independent of each other.

In Part I, which includes Sections 5.1 and 5.2, we introduce block diagrams, which are based on the transfer function concept, and the state-variable model form. The block diagram is a way of representing the dynamics of a system in graphical form. Block diagrams will be used often in the subsequent chapters to describe dynamic systems, and they are also the basis for Simulink programming covered in Part III. An advantage of the state-variable form is that it enables us to express a linear model of any order in a standard and compact way that is useful for analysis and software applications. In Chapter 2 we introduced the `tf`, `step`, and `lsim` functions, which can solve models in transfer function form.

MATLAB has a number of useful functions that are based on the state-variable model form. These functions are covered in Part II, which includes Sections 5.3 and 5.4. Section 5.3 deals with linear models. While the analysis methods of the previous chapters are limited to linear models, the state-variable form is also useful for solving nonlinear equations. It is not always possible or convenient to obtain the closed-form solution of a differential equation, and this is usually true for nonlinear equations. Section 5.4 introduces MATLAB functions that are useful for solving nonlinear differential equations.

Part III includes Sections 5.5 and 5.6 and introduces Simulink, which provides a graphical user interface for solving differential equations. It is especially useful for solving problems containing nonlinear features such as Coulomb friction, saturation, and dead zones, because these features are very difficult to program with traditional programming methods. In addition, its graphical interface might be preferred by some users to the more traditional programming methods offered by the MATLAB solvers covered in Part II. In Section 5.5 we begin with solving linear equations so that we can check the results with the analytical solution. Section 5.6 covers Simulink methods for nonlinear equations. ■

PART I. MODEL FORMS

5.1 TRANSFER FUNCTIONS AND BLOCK DIAGRAM MODELS

We have seen that the complete response of a linear ordinary differential equation (ODE) is the sum of the free and the forced responses. For zero initial conditions, the free response is zero, and the complete response is the same as the forced response.

Thus, we can focus our analysis on the effects of only the input by taking the initial conditions to be zero temporarily. When we have finished analyzing the effects of the input, we can add to the result the free response due to any nonzero initial conditions.

The transfer function is useful for analyzing the effects of the input. Recall from Chapter 2 that the transfer function $T(s)$ is the transform of the forced response $X(s)$ divided by the transform of the input $F(s)$.

$$T(s) = \frac{X(s)}{F(s)}$$

The transfer function can be used as a multiplier to obtain the forced response transform from the input transform; that is, $X(s) = T(s)F(s)$. The transfer function is a property of the system model only. The transfer function is independent of the input function and the initial conditions.

The transfer function is equivalent to the ODE. If we are given the transfer function we can reconstruct the corresponding ODE. For example, the transfer function

$$\frac{X(s)}{F(s)} = \frac{5}{s^2 + 7s + 10}$$

corresponds to the equation $\ddot{x} + 7\dot{x} + 10x = 5f(t)$.

Obtaining a transfer function from a single ODE is straightforward, as we have seen. Sometimes, however, models have more than one input or more than one output. It is important to realize that there is one transfer function for each input-output pair. If a model has more than one input, the transfer function for a particular output variable is the ratio of the transform of the forced response of that variable divided by the input transform, with all the remaining inputs ignored (set to zero temporarily). For example, if the variable x is the output for the equation

$$5\ddot{x} + 30\dot{x} + 40x = 6f(t) - 20g(t)$$

then there are two transfer functions, $X(s)/F(s)$ and $X(s)/G(s)$. These are

$$\frac{X(s)}{F(s)} = \frac{6}{5s^2 + 30s + 40} \qquad \frac{X(s)}{G(s)} = -\frac{20}{5s^2 + 30s + 40}$$

We can obtain transfer functions from systems of equations by first transforming the equations and then algebraically eliminating all variables except for the specified input and output. This technique is especially useful when we want to obtain the response of one or more of the dependent variables in the system of equations. For example, the transfer functions $X(s)/V(s)$ and $Y(s)/V(s)$ of the following system of equations:

$$\dot{x} = -3x + 2y$$
$$\dot{y} = -9y - 4x + 3v(t)$$

are

$$\frac{X(s)}{V(s)} = \frac{6}{s^2 + 12s + 35}$$

and

$$\frac{Y(s)}{V(s)} = \frac{3(s + 3)}{s^2 + 12s + 35}$$

5.1.1 BLOCK DIAGRAMS

We can use the transfer functions of a model to construct a visual representation of the dynamics of the model. Such a representation is a *block diagram*. Block diagrams can be used to describe how system components interact with each other. Unlike a schematic diagram, which shows the physical connections, the block diagram shows the cause and effect relations between the components, and thus helps us to understand the system's dynamics.

Block diagrams can also be used to obtain transfer functions for a given system, for cases where the describing differential equations are not given. In addition, as we will see in Section 5.5, block diagrams can be used to develop *simulation diagrams* for use with computer tools such as Simulink.

5.1.2 BLOCK DIAGRAM SYMBOLS

Block diagrams are constructed from the four basic symbols shown in Figure 5.1.1:

1. The arrow, which is used to represent a variable and the direction of the cause-and-effect relation;
2. The block, which is used to represent the input-output relation of a transfer function;
3. The circle, generically called a *summer,* which represents addition as well as subtraction, depending on the sign associated with the variable's arrow; and
4. The *takeoff point,* which is used to obtain the value of a variable from its arrow, for use in another part of the diagram.

The takeoff point does not modify the value of a variable; a variable has the same value along the entire length of an arrow until it is modified by a circle or a block. You may think of a takeoff point as the tip of a voltmeter probe used to measure a voltage at a point on a wire. If the voltmeter is well-designed, it will not change the value of the voltage it is measuring.

5.1.3 SOME SIMPLE BLOCK DIAGRAMS

The simplest block diagram is shown in Figure 5.1.1b. Inside the block is the system transfer function $T(s)$. The arrow going into the block represents the transform of the input, $F(s)$; the arrow coming out of the block represents the transform of the output, $X(s)$. Thus, the block diagram is a graphical representation of the cause-and-effect relations operating in a particular system. A specific case is shown in Figure 5.1.2a in which the constant transfer function K represents multiplication and the block is called a *multiplier* or a *gain* block. The corresponding equation in the time domain is $x(t) = Kf(t)$. Another simple case is shown in Figure 5.1.2b in which the transfer function $1/s$ represents integration. The corresponding equation in the time domain is $x(t) = \int f(t)\,dt$. Thus, such a block is called an *integrator*. Note that this relation corresponds to the differential equation $\dot{x} = f(t)$.

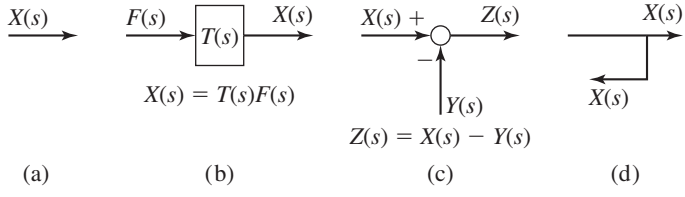


Figure 5.1.1 The four basic symbols used in block diagrams.

Figure 5.1.2 Two types of blocks. (a) Multiplier. (b) Integrator.

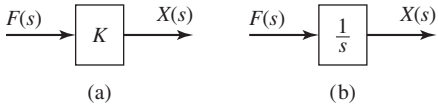
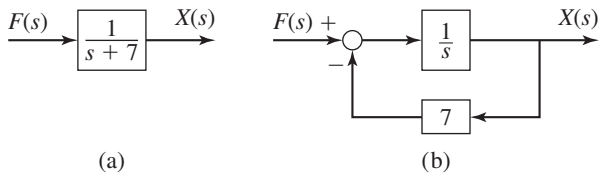


Figure 5.1.3 Diagrams representing the equation $\dot{x} + 7x = f(t)$.



5.1.4 EQUIVALENT BLOCK DIAGRAMS

Figure 5.1.3 shows how more than one diagram can represent the same model, which in this case is $\dot{x} + 7x = f(t)$. The transfer function is $X(s)/F(s) = 1/(s + 7)$, and the corresponding diagram is shown in part (a) of the figure. However, we can rearrange the equation as follows:

$$\dot{x} = f(t) - 7x \qquad \text{or} \qquad x = \int [f(t) - 7x]\, dt$$

which gives

$$X(s) = \frac{1}{s}[F(s) - 7X(s)]$$

In this arrangement the equation corresponds to the diagram shown in part (b) of the figure. Note the use of the takeoff point to feed the variable $X(s)$ to the multiplier. The circle symbol is used to represent addition of the variable $F(s)$ and subtraction of $7X(s)$. The diagram shows how $\dot{x}$, the rate of change of x, is affected by x itself. This is shown by the path from $X(s)$ through the multiplier block to the summer, which changes the sign of $7X(s)$. This path is called a *negative feedback path* or a *negative feedback loop*.

5.1.5 SERIES ELEMENTS AND FEEDBACK LOOPS

Figure 5.1.4 shows two common forms found in block diagrams. In part (a) the two blocks are said to be in *series*. It is equivalent to the diagram in part (b) because we may write

$$B(s) = T_1(s)F(s) \qquad X(s) = T_2(s)B(s)$$

These can be combined algebraically by eliminating $B(s)$ to obtain $X(s) = T_1(s)T_2(s)F(s)$. Note that block diagrams obey the rules of algebra. Therefore, any rearrangement permitted by the rules of algebra is a valid diagram.

Figure 5.1.4 (a) and (b) Simplification of series blocks. (c) and (d) Simplification of a feedback loop.

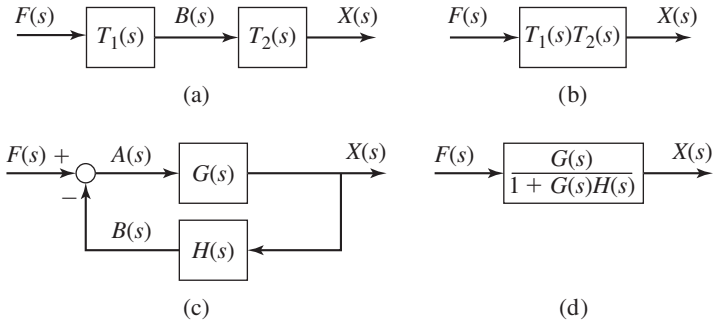


Figure 5.1.4c shows a negative feedback loop. From the diagram, we can obtain the following.

$$A(s) = F(s) - B(s) \qquad B(s) = H(s)X(s) \qquad X(s) = G(s)A(s)$$

We can eliminate $A(s)$ and $B(s)$ to obtain

$$X(s) = \frac{G(s)}{1 + G(s)H(s)}F(s) \tag{5.1.1}$$

This is a useful formula for simplifying a feedback loop to a single block.

5.1.6 REARRANGING BLOCK DIAGRAMS

Now consider the second-order model $\ddot{x} + 7\dot{x} + 10x = f(t)$. The transfer function is $X(s)/F(s) = 1/(s^2 + 7s + 10)$, and the simplest diagram for this model is shown in Figure 5.1.5a. However, to show how x and $\dot{x}$ affect the dynamics of the system, we can construct a diagram that contains the appropriate feedback paths for x and $\dot{x}$. To do this, rearrange the equation by solving for the highest derivative.

$$\ddot{x} = f(t) - 7\dot{x} - 10x$$

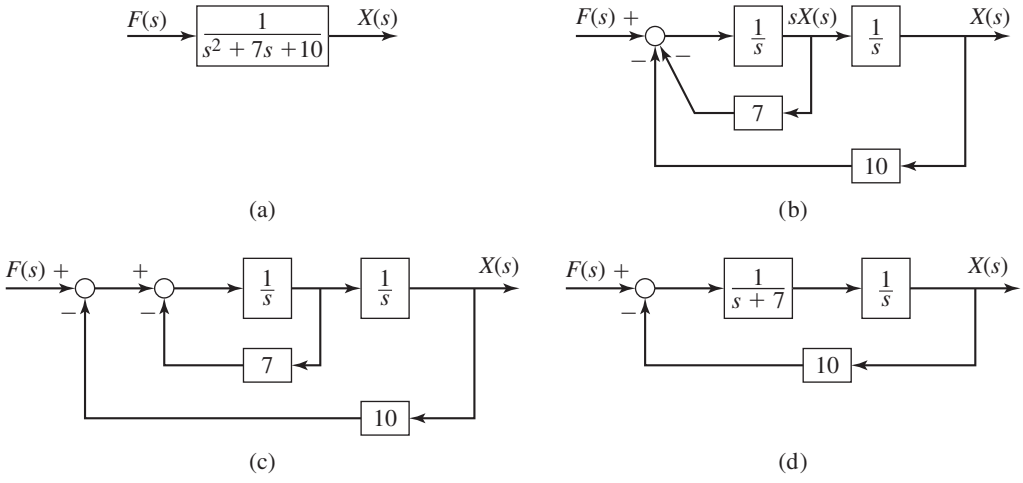
The transformed equation is

$$X(s) = \frac{1}{s}\left(\frac{1}{s}\{F(s) - 7[sX(s)] - 10X(s)\}\right)$$

With this arrangement we can construct the diagram shown in Figure 5.1.5b. Recall that $sX(s)$ represents $\dot{x}$. The term within the pair of curly braces is the output of the summer and the input to the leftmost integrator. The output of this integrator is shown within the outermost pair of parentheses and is the input to the rightmost integrator.

We may use two summers instead of one, and rearrange the diagram as shown in Figure 5.1.5c. This form shows more clearly the negative feedback loop associated with the derivative $\dot{x}$. Referring to Figure 5.1.3, we see that we may replace this inner loop with its equivalent transfer function $1/(s + 7)$. The result is shown in Figure 5.1.5d, which displays only the feedback loop associated with x.

Figure 5.1.5 Diagrams representing the equation $\ddot{x} + 7\dot{x} + 10x = f(t)$.



Two important points can be drawn from these examples.

1. More than one correct diagram can be drawn for a given equation; the desired form of the diagram depends on what information we want to display.
2. The form of the resulting diagram depends on how the equation is arranged. A useful procedure for constructing block diagrams is to first solve for the highest derivative of the dependent variable; the terms on the right side of the resulting equation represent the input to an integrator block.

It is important to understand that the block diagram is a "picture" of the algebraic relations obtained by applying the Laplace transform to the differential equations, assuming that the initial conditions are zero. Therefore, a number of different diagrams can be constructed for a given set of equations and they will all be valid as long as the algebraic relations are correctly represented.

Block diagrams are especially useful when the model consists of more than one differential equation or has more than one input or output. For example, consider the model

$$\dot{x} = -3y + f(t) \qquad \dot{y} = -5y + 4x + g(t)$$

which has two inputs, $f(t)$ and $g(t)$. Suppose we are interested in the variable y as the output. Then the diagram in Figure 5.1.6a is appropriate. Notice that it shows how y affects itself through the feedback loop with the gain of 3, by first affecting x.

Usually we try to place the output variable on the right side of the diagram, with its arrow pointing to the right. We try to place one input on the left side with its arrow point to the right, with a second input, if any, placed at the top of the diagram. The diagram shown in Figure 5.1.6a follows these conventions, which have been established to make it easier for others to interpret your diagrams. Just as in the English language we read from left to right, so the main "flow" of the cause and effect in a diagram (from input to output) should be from left to right if possible.

If instead, we choose the output to be x, then Figure 5.1.6b is more appropriate.

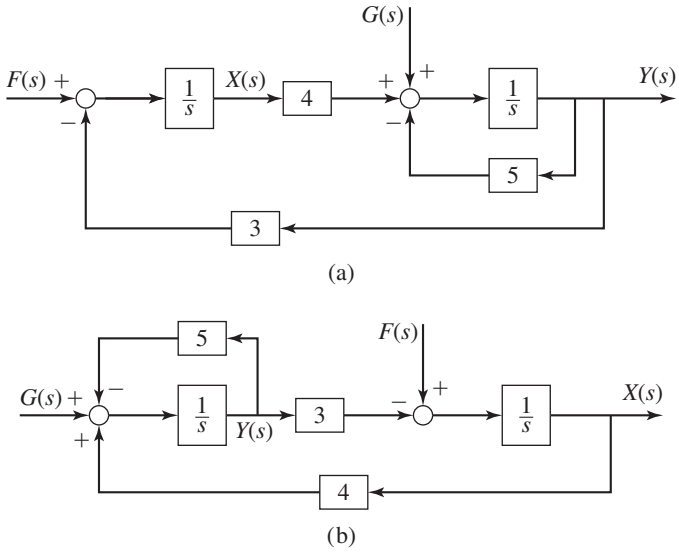


Figure 5.1.6 A diagram with two inputs.

5.1.7 TRANSFER FUNCTIONS FROM BLOCK DIAGRAMS

Sometimes we are given a block diagram and asked to find either the system's transfer function or its differential equation. There are several ways to approach such a problem; the appropriate method depends partly on the form of the diagram and partly on personal preference. The following examples illustrate the process.

Series Blocks and Loop Reduction — EXAMPLE 5.1.1

■ Problem

Determine the transfer function $X(s)/F(s)$ for the system whose diagram is shown in Figure 5.1.7a.

■ Solution

When two blocks are connected by an arrow, they can be combined into a single block that contains the product of their transfer functions. The result is shown in part (b) of the figure. This property, which is called the *series* or *cascade* property, is easily demonstrated. In terms of the variables $X(s)$, $Y(s)$, and $Z(s)$ shown in the diagram, we can write

$$X(s) = \frac{1}{s+12}Y(s) \qquad Y(s) = \frac{1}{s+6}Z(s)$$

Eliminating $Y(s)$ we obtain

$$X(s) = \frac{1}{s+6}\frac{1}{s+12}Z(s)$$

This gives the diagram in part (b) of the figure. So we see that combining two blocks in series is equivalent to eliminating the intermediate variable $Y(s)$ algebraically.

To find the transfer function $X(s)/F(s)$, we can write the following equations based on the diagram in part (b) of the figure:

$$X(s) = \frac{1}{(s+6)(s+12)}Z(s) \qquad Z(s) = F(s) - 8X(s)$$

Eliminating $Z(s)$ from these equations gives the transfer function

$$\frac{X(s)}{F(s)} = \frac{1}{s^2+18s+80}$$

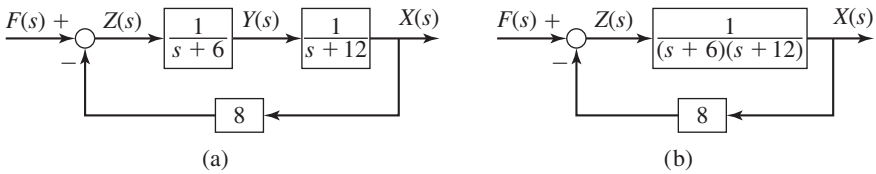


Figure 5.1.7 An example of series combination and loop reduction.

Using Integrator Outputs — EXAMPLE 5.1.2

■ Problem

Determine the model for the output x for the system whose diagram is shown in Figure 5.1.8.

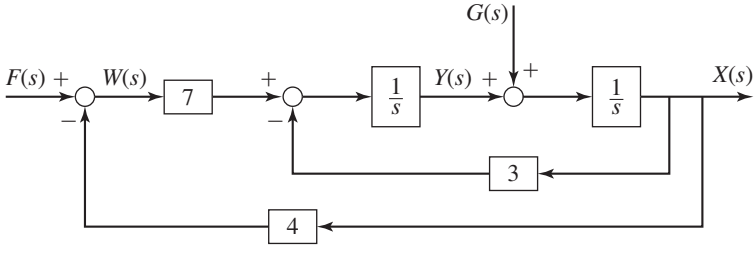


Figure 5.1.8 Diagram for Example 5.1.2.

■ Solution

The input to an integrator block $1/s$ is the time derivative of the output. Thus, by examining the inputs to the two integrators shown in the diagram we can immediately write the time-domain equations as follows.

$$\dot{x} = g(t) + y \qquad \dot{y} = 7w - 3x \qquad w = f(t) - 4x$$

We can eliminate the variable w from the last two equations to obtain $\dot{y} = 7f(t) - 31x$. Thus, the model in differential equation form is

$$\dot{x} = g(t) + y \qquad \dot{y} = 7f(t) - 31x$$

To obtain the model in transfer function form we first transform the equations.

$$sX(s) = G(s) + Y(s) \qquad sY(s) = 7F(s) - 31X(s)$$

Then we eliminate $Y(s)$ algebraically to obtain

$$X(s) = \frac{7}{s^2 + 31}F(s) + \frac{s}{s^2 + 31}G(s)$$

There are two transfer functions, one for each input-output pair. They are

$$\frac{X(s)}{F(s)} = \frac{7}{s^2 + 31} \qquad \frac{X(s)}{G(s)} = \frac{s}{s^2 + 31}$$

Sometimes, we need to obtain the expressions not for just the output variables, but also for some internal variables. The following example illustrates the required method.

EXAMPLE 5.1.3 Deriving Expressions for Internal Variables

■ Problem

Derive the expressions for $C(s)$, $E(s)$, and $M(s)$ in terms of $R(s)$ and $D(s)$ for the diagram in Figure 5.1.9.

■ Solution

Start from the right-hand side of the diagram and work back to the left until *all* blocks and comparators are accounted for. This gives

$$C(s) = \frac{7}{s + 3}[M(s) - D(s)] \tag{1}$$

$$M(s) = \frac{K}{4s + 1}E(s) \tag{2}$$

$$E(s) = R(s) - C(s) \tag{3}$$

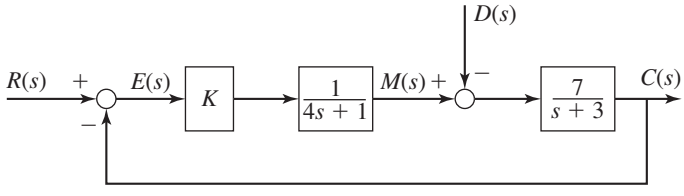


Figure 5.1.9 Block diagram for Example 5.1.3.

Multiply both sides of equation (1) by $s + 3$ to clear fractions, and substitute $M(s)$ and $E(s)$ from equations (2) and (3).

$$\begin{aligned}(s+3)C(s) &= 7M(s) - 7D(s) \\ &= 7\frac{K}{4s+1}E(s) - 7D(s) = \frac{7K}{4s+1}[R(s) - C(s)] - 7D(s)\end{aligned}$$

Multiply both sides by $4s + 1$ to clear fractions, and solve for $C(s)$ to obtain:

$$C(s) = \frac{7K}{4s^2 + 13s + 3 + 7K}R(s) - \frac{7(4s+1)}{4s^2 + 13s + 3 + 7K}D(s) \tag{4}$$

The characteristic polynomial is found from the denominator of either transfer function. It is $4s^2 + 13s + 3 + 7K$.

The equation for $E(s)$ is

$$\begin{aligned}E(s) &= R(s) - C(s) \\ &= R(s) - \frac{7K}{4s^2 + 13s + 3 + 7K}R(s) + \frac{7(4s+1)}{4s^2 + 13s + 3 + 7K}D(s) \\ &= \frac{4s^2 + 13s + 3}{4s^2 + 13s + 3 + 7K}R(s) + \frac{7(4s+1)}{4s^2 + 13s + 3 + 7K}D(s)\end{aligned}$$

Because $4s^2 + 13s + 3$ can be factored as $(4s + 1)(s + 3)$, the equation for $M(s)$ can be expressed as

$$\begin{aligned}M(s) &= \frac{K}{4s+1}E(s) \\ &= \frac{K}{4s+1}\left[\frac{(4s+1)(s+3)}{4s^2 + 13s + 3 + 7K}R(s) + \frac{7(4s+1)}{4s^2 + 13s + 3 + 7K}D(s)\right] \\ &= \frac{K(s+3)}{4s^2 + 13s + 3 + 7K}R(s) + \frac{7K}{4s^2 + 13s + 3 + 7K}D(s)\end{aligned}$$

Note the cancellation of the term $4s + 1$. You should always look for such cancellations. Otherwise, the denominator of the transfer functions can appear to be of higher order than the characteristic polynomial.

5.1.8 BLOCK DIAGRAM ALGEBRA USING MATLAB

MATLAB can be used to perform block diagram algebra if all the gains and transfer function coefficients have numerical values. You can combine blocks in series or in feedback loops using the `series` and `feedback` functions to obtain the transfer function and the state-space model.

If the LTI models `sys1` and `sys2` represent blocks in series, their combined transfer function can be obtained by typing `sys3 = series(sys1,sys2)`. A simple gain need not be converted to a LTI model, and does not require the `series` function. For example, if the first system is a simple gain K, use the multiplication symbol * and enter

```
≫ sys3 = K*sys2
```

If the LTI model `sys2` is in a *negative* feedback loop around the LTI model `sys1`, then enter

```
≫ sys3 = feedback(sys1,sys2)
```

to obtain the LTI model of the closed-loop system.

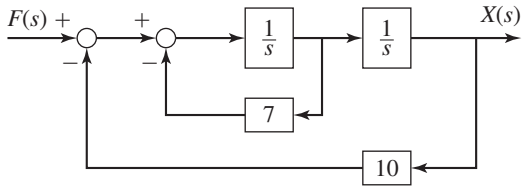


Figure 5.1.10 A typical block diagram.

If the feedback loop is *positive*, use the syntax

```
≫ sys3 = feedback(sys1,sys2,+1)
```

If you need to obtain the numerator and denominator of the closed-loop transfer function, you can use the `tfdata` function and enter

```
≫[num,den] = tfdata(sys3,'v')
```

You can then find the characteristic roots by entering

```
≫ roots(den)
```

For example, to find the transfer function $X(s)/F(s)$ corresponding to the block diagram shown in Figure 5.1.10, you enter

```
≫ sys1=tf(1,[1,0]);
≫ sys2=feedback(sys1,7);
≫ sys3=series(sys1,sys2);
≫ sys4=feedback(sys3,10);
≫ [num,den]=tfdata(sys4,'v')
```

The result is num = [0, 0, 1] and den = [1, 7, 10], which corresponds to

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 7s + 10}$$

5.2 STATE-VARIABLE MODELS

Models that consist of coupled first-order differential equations are said to be in *state-variable* form. This form, which is also called the *Cauchy* form, has an advantage over the reduced form, which consists of a single, higher-order equation, because it allows a linear model to be expressed in a standard and compact way that is useful for analysis and for software applications. This representation makes use of vector and matrix notation. In this section, we will show how to obtain a model in state-variable form and how to express state-variable models in vector-matrix notation. In Section 5.3 we show how to use this notation with MATLAB.

Consider the second-order equation

$$5\ddot{y} + 7\dot{y} + 4y = f(t)$$

Solve it for the highest derivative:

$$\ddot{y} = \frac{1}{5}f(t) - \frac{4}{5}y - \frac{7}{5}\dot{y}$$

Now define two new variables, x_1 and x_2, as follows: $x_1 = y$ and $x_2 = \dot{y}$. This implies that $\dot{x}_1 = x_2$ and

$$\dot{x}_2 = \frac{1}{5}f(t) - \frac{4}{5}x_1 - \frac{7}{5}x_2$$

These two equations, called the *state equations,* are the state-variable form of the model, and the variables x_1 and x_2 are called the *state variables*.

The general mass-spring-damper model has the following form:

$$m\ddot{x} + c\dot{x} + kx = f \tag{5.2.1}$$

If we define new variables x_1 and x_2 such that

$$x_1 = x \qquad x_2 = \dot{x}$$

these imply that

$$\dot{x}_1 = x_2 \tag{5.2.2}$$

Then we can write the model (5.2.1) as: $m\dot{x}_2 + cx_2 + kx_1 = f$. Next solve for $\dot{x}_2$:

$$\dot{x}_2 = \frac{1}{m}(f - kx_1 - cx_2) \tag{5.2.3}$$

Equations (5.2.2) and (5.2.3) constitute a state-variable model corresponding to the reduced model (5.2.1). The variables x_1 and x_2 are the *state variables*.

If (5.2.1) represents a mass-spring-damper system, the state-variable x_1 describes the system's potential energy $kx_1^2/2$, which is due to the spring, and the state-variable x_2 describes the system's kinetic energy $mx_2^2/2$, which is due to the mass. Although here we have derived the state variable model from the reduced form, state-variable models can be derived from basic physical principles. Choosing as state variables those variables that describe the types of energy in the system sometimes helps to derive the model (note that k and m are also needed to describe the energies, but these are parameters, not variables).

The choice of state variables is not unique, but the choice must result in a set of first-order differential equations. For example, we could have chosen the state variables to be $x_1 = x$ and $x_2 = m\dot{x}$, which is the system's momentum. In this case the state-variable model would be

$$\dot{x}_1 = \frac{1}{m}x_2$$

$$\dot{x}_2 = f - \frac{c}{m}x_2 - kx_1$$

State-Variable Model of a Two-Mass System

EXAMPLE 5.2.1

■ Problem

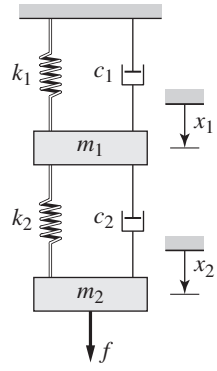
Consider the two-mass system discussed in Chapter 4 (and shown again in Figure 5.2.1). Suppose the parameter values are $m_1 = 5$, $m_2 = 3$, $c_1 = 4$, $c_2 = 8$, $k_1 = 1$, and $k_2 = 4$. The equations of motion are

$$5\ddot{x}_1 + 12\dot{x}_1 + 5x_1 - 8\dot{x}_2 - 4x_2 = 0 \tag{1}$$

$$3\ddot{x}_2 + 8\dot{x}_2 + 4x_2 - 8\dot{x}_1 - 4x_1 = f(t) \tag{2}$$

Put these equations into state-variable form.

Figure 5.2.1 A two-mass system.



■ Solution

Using the system's potential and kinetic energies as a guide, we see that the displacements x_1 and x_2 describe the system's potential energy and that the velocities $\dot{x}_1$ and $\dot{x}_2$ describe the system's kinetic energy. That is

$$\text{PE} = \frac{1}{2}k_1x_1^2 + \frac{1}{2}k_2(x_1 - x_2)^2$$

and

$$\text{KE} = \frac{1}{2}m_1\dot{x}_1^2 + \frac{1}{2}m_2\dot{x}_2^2$$

This indicates that we need four state variables. (Another way to see that we need four variables is to note that the model consists of two coupled second-order equations, and thus is effectively a fourth-order model.) Thus, we can choose the state variables to be

$$x_1 \qquad x_2 \qquad x_3 = \dot{x}_1 \qquad x_4 = \dot{x}_2 \tag{3}$$

Thus, two of the state equations are $\dot{x}_1 = x_3$ and $\dot{x}_2 = x_4$. The remaining two equations can be found by solving equations (1) and (2) for $\ddot{x}_1$ and $\ddot{x}_2$, noting that $\ddot{x}_1 = \dot{x}_3$ and $\ddot{x}_2 = \dot{x}_4$, and using the substitutions given by equation (3).

$$\dot{x}_3 = \frac{1}{5}(-12x_3 - 5x_1 + 8x_4 + 4x_2)$$

$$\dot{x}_4 = \frac{1}{3}[-8x_4 - 4x_2 + 8x_3 + 4x_1 + f(t)]$$

Note that the left-hand sides of the state equations must contain only the first-order derivative of each state variable. This is why we divided by 5 and 3, respectively. Note also that the right-hand sides must not contain any derivatives of the state variables. Failure to observe this restriction is a common mistake.

Now list the four state equations in ascending order according to their left-hand sides, after rearranging the right-hand sides so that the state variables appear in ascending order from left to right.

$$\dot{x}_1 = x_3 \tag{4}$$

$$\dot{x}_2 = x_4 \tag{5}$$

$$\dot{x}_3 = \frac{1}{5}(-5x_1 + 4x_2 - 12x_3 + 8x_4) \tag{6}$$

$$\dot{x}_4 = \frac{1}{3}[4x_1 - 4x_2 + 8x_3 - 8x_4 + f(t)] \tag{7}$$

These are the state equations in standard form.

5.2.1 VECTOR-MATRIX FORM OF STATE-VARIABLE MODELS

Vector-matrix notation enables us to represent multiple equations as a single matrix equation. For example, consider the following set of linear algebraic equations.

$$2x_1 + 9x_2 = 5 \tag{5.2.4}$$

$$3x_1 - 4x_2 = 7 \tag{5.2.5}$$

The term *matrix* refers to an array with more than one column and more than one row. A *column vector* is an array having only one column. A *row vector* has only one row. A matrix is an arrangement of numbers and is not the same as a determinant, which can be reduced to a single number. Multiplication of a matrix having two rows and two columns (a (2×2) matrix) by a column vector having two rows and one column (a (2×1) vector) is defined as follows:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{bmatrix} \tag{5.2.6}$$

This definition is easily extended to matrices having more than two rows or two columns. In general, the result of multiplying an $(n \times n)$ matrix by an $(n \times 1)$ vector is an $(n \times 1)$ vector. This definition of vector-matrix multiplication requires that the matrix have as many columns as the vector has rows. The order of the multiplication cannot be reversed (vector-matrix multiplication does not have the commutative property).

Two vectors are equal if all their respective elements are equal. Thus we can represent the set (5.2.4) and (5.2.5) as follows:

$$\begin{bmatrix} 2 & 9 \\ 3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix} \tag{5.2.7}$$

We usually represent matrices and vectors in boldface type, with matrices usually in upper case letters and vectors in lowercase, but this is not required. Thus we can represent the set (5.2.7) in the following compact form.

$$\mathbf{Ax} = \mathbf{b} \tag{5.2.8}$$

where we have defined the following matrices and vectors:

$$\mathbf{A} = \begin{bmatrix} 2 & 9 \\ 3 & -4 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

The matrix $\mathbf{A}$ corresponds in an ordered fashion to the coefficients of x_1 and x_2 in (5.2.4) and (5.2.5). Note that the first row in $\mathbf{A}$ consists of the coefficients of x_1 and x_2 on the left-hand side of (5.2.4), and the second row contains the coefficients on the left-hand side of (5.2.5). The vector $\mathbf{x}$ contains the variables x_1 and x_2, and the vector $\mathbf{b}$ contains the right-hand sides of (5.2.4) and (5.2.5).

Vector-Matrix Form of a Single-Mass Model

EXAMPLE 5.2.2

■ Problem

Express the mass-spring-damper model (5.2.2) and (5.2.3) as a single vector-matrix equation. These equations are

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{m} f(t) - \frac{k}{m} x_1 - \frac{c}{m} x_2$$

■ Solution

The equations can be written as one equation as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} f(t)$$

In compact form this is

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{B} f(t)$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{c}{m} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

EXAMPLE 5.2.3 Vector-Matrix Form of the Two-Mass Model

■ **Problem**

Express the state-variable model of Example 5.2.1 in vector-matrix form. The model is

$$\dot{x}_1 = x_3$$
$$\dot{x}_2 = x_4$$
$$\dot{x}_3 = \frac{1}{5}(-5x_1 + 4x_2 - 12x_3 + 8x_4)$$
$$\dot{x}_4 = \frac{1}{3}[4x_1 - 4x_2 + 8x_3 - 8x_4 + f(t)]$$

■ **Solution**

In vector-matrix form these equations are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}f(t)$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & \frac{4}{5} & -\frac{12}{5} & \frac{8}{5} \\ \frac{4}{3} & -\frac{4}{3} & \frac{8}{3} & -\frac{8}{3} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{3} \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

5.2.2 STANDARD FORM OF THE STATE EQUATION

We may use any symbols we choose for the state variables and the input function, although the common choice is x_i for the state variables and u_i for the input functions. The standard vector-matrix form of the state equations, where the number of state variables is n and the number of inputs is m, is

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.2.9}$$

where the vectors $\mathbf{x}$ and $\mathbf{u}$ are column vectors containing the state variables and the inputs, if any. The dimensions are as follows:

- The *state vector* $\mathbf{x}$ is a column vector having n rows.
- The *system matrix* $\mathbf{A}$ is a square matrix having n rows and n columns.
- The *input vector* $\mathbf{u}$ is a column vector having m rows.
- The *control* or *input matrix* $\mathbf{B}$ has n rows and m columns.

In our examples thus far there has been only one input, and for such cases the input vector $\mathbf{u}$ reduces to a scalar u. The standard form, however, allows for more than one input function. Such would be the case in the two-mass model if external forces f_1 and f_2 are applied to the masses.

5.2.3 THE OUTPUT EQUATION

Some software packages and some design methods require you to define an *output vector,* usually denoted by **y**. The output vector contains the variables that are of interest for the particular problem at hand. These variables are not necessarily the state variables, but might be some combination of the state variables and the inputs. For example, in the mass-spring model, we might be interested in the total force $f - kx - c\dot{x}$ acting on the mass, and in the momentum $m\dot{x}$. In this case, the output vector has two elements. If the state variables are $x_1 = x$ and $x_2 = \dot{x}$, the output vector is

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} f - kx - c\dot{x} \\ m\dot{x} \end{bmatrix} = \begin{bmatrix} f - kx_1 - cx_2 \\ mx_2 \end{bmatrix}$$

or

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -k & -c \\ 0 & m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} f = \mathbf{C}\mathbf{x} + \mathbf{D}f$$

where

$$\mathbf{C} = \begin{bmatrix} -k & -c \\ 0 & m \end{bmatrix}$$

and

$$\mathbf{D} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

This is an example of the general form: $\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$.

The standard vector-matrix form of the output equation, where the number of outputs is p, the number of state variables is n, and the number of inputs is m, is

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{5.2.10}$$

where the vector **y** contains the output variables. The dimensions are as follows:

- The *output vector* **y** is a column vector having p rows.
- The *state output matrix* **C** has p rows and n columns.
- The *control output matrix* **D** has p rows and m columns.

The matrices **C** and **D** can always be found whenever the chosen output vector **y** is a linear combination of the state variables and the inputs. However, if the output is a nonlinear function, then the standard form (5.2.10) does not apply. This would be the case, for example, if the output is chosen to be the system's kinetic energy: $\text{KE} = mx_2^2/2$.

EXAMPLE 5.2.4 The Output Equation for a Two-Mass Model

■ **Problem**

Consider the two-mass model of Example 5.2.1.

a) Suppose the outputs are x_1 and x_2. Determine the output matrices **C** and **D**. b) Suppose the outputs are $(x_2 - x_1)$, $\dot{x}_2$, and f. Determine the output matrices **C** and **D**.

■ **Solution**

a. In terms of the **z** vector, $z_1 = x_1$ and $z_3 = x_2$. We can express the output vector **y** as follows.

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} f$$

Thus

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

b. Here the outputs are $y_1 = x_2 - x_1$, $y_2 = \dot{x}_2 = x_4$, and $y_3 = f$. Thus we can express the output vector as follows:

$$\mathbf{y} = \begin{bmatrix} x_2 - x_1 \\ x_4 \\ f \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} f$$

Thus

$$\mathbf{C} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

5.2.4 TRANSFER-FUNCTION VERSUS STATE-VARIABLE MODELS

The decision whether to use a reduced-form model (which is equivalent to a transfer-function model) or a state-variable model depends on many factors, including personal preference. In fact, for many applications both models are equally effective and equally easy to use. Application of basic physical principles sometimes directly results in a state-variable model. An example is the following two-inertia fluid-clutch model derived in Chapter 4.

$$I_d\dot{\omega}_d = T_d - c(\omega_d - \omega_1)$$
$$I_1\dot{\omega}_1 = -T_1 + c(\omega_d - \omega_1)$$

The state and input vectors are

$$\mathbf{x} = \begin{bmatrix} \omega_d \\ \omega_1 \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} T_d \\ T_1 \end{bmatrix}$$

The system and input matrices are

$$\mathbf{A} = \begin{bmatrix} -\dfrac{c}{I_d} & \dfrac{c}{I_d} \\ \dfrac{c}{I_1} & -\dfrac{c}{I_1} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \dfrac{1}{I_d} & 0 \\ 0 & -\dfrac{1}{I_1} \end{bmatrix}$$

For example, this form of the model is easier to use if you need to obtain only numerical values or a plot of the step response, because you can directly use the MATLAB function `step(A,B,C,D)`, to be introduced in Section 5.3. However, if you need to obtain the step response as a function, it might be easier to convert the model to transfer function

form and then use the Laplace transform to obtain the desired function. To obtain the transfer function from the state-variable model, you may use the MATLAB function `tf(sys)`, as shown in Section 5.3.

The MATLAB functions cited require that all the model parameters have specified numerical values. If, however, you need to examine the effects of a system parameter, say the damping coefficient c in the clutch model, then it is perhaps preferable to convert the model to transfer function form. In this form, you can examine the effect of c on system response by examining numerator dynamics and the characteristic equation. You can also use the initial- and final-value theorems to investigate the response.

5.2.5 MODEL FORMS HAVING NUMERATOR DYNAMICS

Note that if you only need to obtain the free response, then the presence of input derivatives or numerator dynamics in the model is irrelevant. For example, the free response of the model

$$5\frac{d^3y}{dt^3} + 3\frac{d^2y}{dt^2} + 7\frac{dy}{dt} + 6y = 4\frac{df}{dt} + 9f(t)$$

is identical to the free response of the model

$$5\frac{d^3y}{dt^3} + 3\frac{d^2y}{dt^2} + 7\frac{dy}{dt} + 6y = 0$$

which does not have any inputs. A state-variable model for this equation is easily found to be

$$x_1 = y \qquad x_2 = \dot{y} \qquad x_3 = \ddot{y}$$

$$\dot{x}_1 = x_2 \qquad \dot{x}_2 = x_3 \qquad \dot{x}_3 = -\tfrac{6}{5}x_1 - \tfrac{7}{5}x_2 - \tfrac{3}{5}x_3$$

The free response of this model can be easily found with the MATLAB `initial` function to be introduced in the next section.

For some applications you need to obtain a state-variable model in the standard form. However, in the standard state-variable form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ there is no derivative of the input $\mathbf{u}$. When the model has numerator dynamics or input derivatives, the state variables are not so easy to identify. When there are no numerator dynamics you can always obtain a state-variable model in standard form from a transfer-function or reduced-form model whose dependent variable is x by defining $x_1 = x$, $x_2 = \dot{x}$, $x_3 = \ddot{x}$, and so forth. This was the procedure followed previously. Note that the initial conditions $x_1(0)$, $x_2(0)$, and $x_3(0)$ are easily obtained from the given conditions $x(0)$, $\dot{x}(0)$, and $\ddot{x}(0)$; that is, $x_1(0) = x(0)$, $x_2(0) = \dot{x}(0)$, and $x_3(0) = \ddot{x}(0)$. However, when numerator dynamics are present, a different technique must be used, and the initial conditions are not as easily related to the state variables.

We now give two examples of how to obtain a state-variable model when numerator dynamics exists.

Numerator Dynamics in a First-Order System **EXAMPLE 5.2.5**

■ **Problem**

Consider the transfer function model

$$\frac{Z(s)}{U(s)} = \frac{5s + 3}{s + 2} \tag{1}$$

This corresponds to the equation

$$\dot{z} + 2z = 5\dot{u} + 3u \tag{2}$$

Note that this equation is not in the standard form $\dot{z} = Az + Bu$ because of the input derivative $\dot{u}$. Demonstrate two ways of converting this model to a state-variable model in standard form.

■ Solution

a. One way of obtaining the state-variable model is to divide the numerator and denominator of equation (1) by s.

$$\frac{Z(s)}{U(s)} = \frac{5 + 3/s}{1 + 2/s} \tag{3}$$

The objective is to obtain a 1 in the denominator, which is then used to isolate $Z(s)$ as follows:

$$\begin{aligned} Z(s) &= -\frac{2}{s}Z(s) + 5U(s) + \frac{3}{s}U(s) \\ &= \frac{1}{s}[3U(s) - 2Z(s)] + 5U(s) \end{aligned}$$

The term within square brackets multiplying $1/s$ is the input to an integrator, and the integrator's output can be selected as a state-variable x. Thus,

$$Z(s) = X(s) + 5U(s)$$

where

$$\begin{aligned} X(s) &= \frac{1}{s}[3U(s) - 2Z(s)] = \frac{1}{s}\{3U(s) - 2[X(s) + 5U(s)]\} \\ &= \frac{1}{s}[-2X(s) - 7U(s)] \end{aligned}$$

This gives

$$\dot{x} = -2x - 7u \tag{4}$$

with the output equation

$$z = x + 5u \tag{5}$$

This fits the standard form (5.2.9) and (5.2.10), with $\mathbf{A} = -2$, $\mathbf{B} = -7$, $\mathbf{y} = z$, $\mathbf{C} = 1$, and $\mathbf{D} = 5$.

Presumably we are given the initial condition $z(0)$, but to solve equation (4) we need $x(0)$. This can be obtained by solving equation (5) for x, $x = z - 5u$, and evaluating it at $t = 0$: $x(0) = z(0) - 5u(0)$. We see that $x(0) = z(0)$ if $u(0) = 0$.

b. Another way is to write equation (1) as

$$Z(s) = (5s + 3)\frac{U(s)}{s + 2} \tag{6}$$

and define the state-variable x as follows:

$$X(s) = \frac{U(s)}{s + 2} \tag{7}$$

Thus,

$$sX(s) = -2X(s) + U(s) \tag{8}$$

and the state equation is

$$\dot{x} = -2x + u \tag{9}$$

To find the output equation, note that

$$Z(s) = (5s + 3)\frac{U(s)}{s + 2} = (5s + 3)X(s) = 5sX(s) + 3X(s)$$

Using equation (8) we have

$$Z(s) = 5[-2X(s) + U(s)] + 3X(s) = -7X(s) + 5U(s)$$

and thus the output equation is

$$z = -7x + 5u \tag{10}$$

The initial condition $x(0)$ is found from equation (10) to be $x(0) = [5u(0) - z(0)]/7 = -z(0)/7$ if $u(0) = 0$.

Although the model consisting of equations (9) and (10) looks different than that given by equations (4) and (5), they are both in the standard form and are equivalent, because they were derived from the same transfer function.

This example points out that there is no unique way to derive a state-variable model from a transfer function. It is important to keep this in mind because the state-variable model obtained from the MATLAB `ssdata(sys)` function, to be introduced in the next section, might not be the one you expect. The state-variable model given by MATLAB is $\dot{x} = -2x + 2u$, $z = -3.5x + 5u$. These values correspond to equation (1) being written as

$$Z(s) = \frac{5s + 3}{s + 2}U(s) = (2.5s + 1.5)\left[\frac{2U(s)}{s + 2}\right]$$

and defining x as the term within the square brackets; that is,

$$X(s) = \frac{2U(s)}{s + 2}$$

The order of the system, and therefore the number of state variables required, can be found by examining the denominator of the transfer function. If the denominator polynomial is of order n, then n state variables are required. Frequently a convenient choice is to select the state variables as the outputs of integrations $(1/s)$, as was done in Example 5.2.5.

Numerator Dynamics in a Second-Order System

EXAMPLE 5.2.6

■ Problem

Obtain a state-variable model for

$$\frac{X(s)}{U(s)} = \frac{4s + 7}{5s^2 + 4s + 7} \tag{1}$$

Relate the initial conditions for the state variables to the given initial conditions $x(0)$ and $\dot{x}(0)$.

■ Solution

Divide by $5s^2$ to obtain a 1 in the denominator.

$$\frac{X(s)}{U(s)} = \frac{\frac{7}{5}s^{-2} + \frac{4}{5}s^{-1}}{1 + \frac{4}{5}s^{-1} + \frac{7}{5}s^{-2}}$$

Use the 1 in the denominator to solve for $X(s)$.

$$X(s) = \left(\frac{7}{5}s^{-2} + \frac{4}{5}s^{-1}\right)U(s) - \left(\frac{4}{5}s^{-1} + \frac{7}{5}s^{-2}\right)X(s)$$

$$= \frac{1}{s}\left\{-\frac{4}{5}X(s) + \frac{4}{5}U(s) + \frac{1}{s}\left[\frac{7}{5}U(s) - \frac{7}{5}X(s)\right]\right\} \tag{2}$$

This equation shows that $X(s)$ is the output of an integration. Thus x can be chosen as a state-variable x_1. Thus,

$$X_1(s) = X(s)$$

The term within square brackets in (2) is the input to an integration, and thus the second state variable can be chosen as

$$X_2(s) = \frac{1}{s}\left[\frac{7}{5}U(s) - \frac{7}{5}X(s)\right] = \frac{1}{s}\left[\frac{7}{5}U(s) - \frac{7}{5}X_1(s)\right] \tag{3}$$

Then from equation (2)

$$X_1(s) = \frac{1}{s}\left[-\frac{4}{5}X_1(s) + \frac{4}{5}U(s) + X_2(s)\right] \tag{4}$$

The state equations are found from (3) and (4).

$$\dot{x}_1 = -\frac{4}{5}x_1 + x_2 + \frac{4}{5}u \tag{5}$$

$$\dot{x}_2 = -\frac{7}{5}x_1 + \frac{7}{5}u \tag{6}$$

and the output equation is $x = x_1$. The matrices of the standard form are

$$\mathbf{A} = \begin{bmatrix} -\frac{4}{5} & 1 \\ -\frac{7}{5} & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \frac{4}{5} \\ \frac{7}{5} \end{bmatrix}$$

$$\mathbf{C} = [1 \quad 0] \qquad \mathbf{D} = [0]$$

Note that the state variables obtained by this technique do not always have straightforward physical interpretations. If the model $m\ddot{x} + c\dot{x} + kx = c\dot{u} + ku$ represents a mass-spring-damper system with a displacement input u with $m = 5$, $c = 4$, $k = 7$, the variable x_2 is the integral of the spring force $k(u - x)$, divided by the mass m. Thus, x_2 is the acceleration of the mass due to this force. Sometimes convenient physical interpretations of the state variables are sacrificed to obtain special forms of the state equations that are useful for analytical purposes.

Using equations (5) and (6), we need to relate the values of $x_1(0)$ and $x_2(0)$ to $x(0)$ and $\dot{x}(0)$. Because x_1 was defined to be $x_1 = x$, we see that $x_1(0) = x(0)$. To find $x_2(0)$, we solve the first state equation, equation (5), for x_2.

$$x_2 = \dot{x}_1 + \frac{4}{5}(x_1 - u)$$

This gives

$$x_2(0) = \dot{x}_1(0) + \frac{4}{5}[x_1(0) - u(0)] = \dot{x}(0) + \frac{4}{5}[x(0) - u(0)]$$

Thus if $u(0) = 0$,

$$x_2(0) = \dot{x}(0) + \frac{4}{5}x(0)$$

Table 5.2.1 A state-variable form for numerator dynamics.

Transfer function model:	$\dfrac{Y(s)}{U(s)} = \dfrac{\beta_n s^n + \beta_{n-1}s^{n-1} + \cdots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0}$
State-variable model:	$\dot{x}_1 = \gamma_{n-1}u - \alpha_{n-1}x_1 + x_2$ $\dot{x}_2 = \gamma_{n-2}u - \alpha_{n-2}x_1 + x_3$ $\vdots$ $\dot{x}_j = \gamma_{n-j}u - \alpha_{n-j}x_1 + x_{j+1}, \qquad j = 1, 2, \ldots, n-1$ $\vdots$ $\dot{x}_n = \gamma_0 u - \alpha_0 x_1$ $y = \beta_n u + x_1$
where	$\gamma_i = \beta_i - \alpha_i \beta_n$
Usual case:	If $u(0) = \dot{u}(0) = \cdots = 0$, then $x_i(0) = y^{(i-1)}(0) + \alpha_{n-1}y^{(i-2)}(0) + \cdots + \alpha_{n-i+1}y(0)$ $i = 1, 2, \ldots, n$
where	$y^{(i)}(0) = \left.\dfrac{d^i y}{dt^i}\right\|_{t=0}$

The method of the previous example can be extended to the general case where the transfer function is

$$\frac{Y(s)}{U(s)} = \frac{\beta_n s^n + \beta_{n-1}s^{n-1} + \cdots + \beta_1 s + \beta_0}{s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_1 s + \alpha_0} \tag{5.2.11}$$

The results are shown in Table 5.2.1. The details of the derivation are given in [Palm, 1986].

PART II. MATLAB METHODS

5.3 STATE-VARIABLE METHODS WITH MATLAB

The MATLAB `step`, `impulse`, and `lsim` functions, treated in Section 2.9, can also be used with state-variable models. However, the `initial` function, which computes the free response, can be used only with a state-variable model. MATLAB also provides functions for converting models between the state-variable and transfer function forms.

Recall that to create an LTI object from the reduced form

$$5\ddot{x} + 7\dot{x} + 4x = f(t) \tag{5.3.1}$$

or the transfer function form

$$\frac{X(s)}{F(s)} = \frac{1}{5s^2 + 7s + 4} \tag{5.3.2}$$

you use the `tf(num,den)` function by typing:

```
≫sys1 = tf(1, [5, 7, 4]);
```

The result, `sys1`, is the LTI object that describes the system in the transfer function form.

The LTI object `sys2` in transfer function form for the equation

$$8\frac{d^3x}{dt^3} - 3\frac{d^2x}{dt^2} + 5\frac{dx}{dt} + 6x = 4\frac{d^2f}{dt^2} + 3\frac{df}{dt} + 5f \tag{5.3.3}$$

is created by typing

```
≫sys2 = tf([4, 3, 5],[8, -3, 5, 6]);
```

5.3.1 LTI OBJECTS AND THE `ss(A,B,C,D)` FUNCTION

To create an LTI object from a state model, you use the `ss(A,B,C,D)` function, where `ss` stands for *state space*. The matrix arguments of the function are the matrices in the following standard form of a state model:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.3.4}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{5.3.5}$$

where **x** is the vector of state variables, **u** is the vector of input functions, and **y** is the vector of output variables. For example, to create an LTI object in state-model form for the system described by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{5}f(t) - \frac{4}{5}x_1 - \frac{7}{5}x_2$$

where x_1 is the desired output, the required matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{4}{5} & -\frac{7}{5} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{5} \end{bmatrix}$$

$$\mathbf{C} = [1 \quad 0] \qquad \mathbf{D} = 0$$

In MATLAB you type

```
≫A = [0, 1; -4/5, -7/5];
≫B = [0; 1/5];
≫C = [1, 0];
≫D = 0;
≫sys3 = ss(A,B,C,D);
```

5.3.2 THE `ss(sys)` AND `ssdata(sys)` FUNCTIONS

An LTI object defined using the `tf` function can be used to obtain an equivalent state model description of the system. To create a state model for the system described by the LTI object `sys1` created previously in transfer function form, you type `ss(sys1)`. You will then see the resulting **A**, **B**, **C**, and **D** matrices on the screen. To extract and save the matrices as `A1`, `B1`, `C1`, and `D1` (to avoid overwriting the matrices from the second example here), use the `ssdata` function as follows.

```
≫[A1, B1, C1, D1] = ssdata(sys1);
```

The results are

$$\mathbf{A1} = \begin{bmatrix} -1.4 & -0.8 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{B1} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

$$\mathbf{C1} = [0 \quad 0.4]$$

$$\mathbf{D1} = [0]$$

which correspond to the state equations:

$$\dot{x}_1 = -1.4x_1 - 0.8x_2 + 0.5f(t)$$
$$\dot{x}_2 = x_1$$

and the output equation $y = 0.4x_2$.

5.3.3 RELATING STATE VARIABLES TO THE ORIGINAL VARIABLES

When using `ssdata` to convert a transfer function form into a state model, note that the output y will be a scalar that is identical to the solution variable of the reduced form; in this case the solution variable of (5.3.1) is the variable x. To interpret the state model, we need to relate its state variables x_1 and x_2 to x. The values of the matrices **C1** and **D1** tell us that the output variable is $y = 0.4x_2$. Because the output y is the same as x, we then see that $x_2 = x/0.4 = 2.5x$. The other state-variable x_1 is related to x_2 by the second state equation $\dot{x}_2 = x_1$. Thus, $x_1 = 2.5\dot{x}$.

5.3.4 THE `tfdata` FUNCTION

To create a transfer function description of the system `sys3`, previously created from the state model, you type `tfsys3 = tf(sys3)`. However, there can be situations where we are given the model `tfsys3` in transfer function form and we need to obtain the numerator and denominator. To extract and save the coefficients of the transfer function, use the `tfdata` function as follows.

```
≫[num, den] = tfdata(tfsys3, 'v');
```

The optional parameter `'v'` tells MATLAB to return the coefficients as vectors if there is only one transfer function; otherwise, they are returned as cell arrays.

For this example, the vectors returned are `num = [0, 0, 0.2]` and `den = [1, 1.4, 0.8]`. This corresponds to the transfer function

$$\frac{X(s)}{F(s)} = \frac{0.2}{s^2 + 1.4s + 0.8} = \frac{1}{5s^2 + 7s + 4}$$

which is the correct transfer function, as seen from (5.2.2).

EXAMPLE 5.3.1 Transfer Functions of a Two-Mass System

■ **Problem**

Obtain the transfer functions $X_1(s)/F(s)$ and $X_2(s)/F(s)$ of the state-variable model obtained in Example 5.2.3. The matrices and state vector of the model are

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & \frac{4}{5} & -\frac{12}{5} & \frac{8}{5} \\ \frac{4}{3} & -\frac{4}{3} & \frac{8}{3} & -\frac{8}{3} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{3} \end{bmatrix}$$

and

$$\mathbf{z} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

■ **Solution**

Because we want the transfer functions for x_1 and x_2, we must define the **C** and **D** matrices to indicate that z_1 and z_3 are the output variables y_1 and y_2. Thus,

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The MATLAB program is as follows.

```
A = [0, 0, 1, 0; 0, 0, 0, 1;...
     -1, 4/5, -12/5, 8/5; 4/3, -4/3, 8/3, -8/3];
B = [0; 0; 0; 1/3];
C = [1, 0, 0, 0; 0, 1, 0, 0]; D = [0; 0]
sys4 = ss(A, B, C, D);
tfsys4 = tf(sys4)
```

The results displayed on the screen are labeled #1 and #2. These correspond to the first and second transfer functions in order. The answers are

$$\frac{X_1(s)}{F(s)} = \frac{0.5333s + 0.2667}{s^4 + 5.067s^3 + 4.467s^2 + 1.6s + 0.2667}$$

$$\frac{X_2(s)}{F(s)} = \frac{0.3333s^2 + 0.8s + 0.3333}{s^4 + 5.067s^3 + 4.467s^2 + 1.6s + 0.2667}$$

Table 5.3.1 summarizes these functions.

Table 5.3.1 LTI object functions.

Command	Description
`sys = ss(A, B, C, D)`	Creates an LTI object in state-space form, where the matrices `A`, `B`, `C`, and `D` correspond to those in the model $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$, $\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$.
`[A, B, C, D] = ssdata(sys)`	Extracts the matrices `A`, `B`, `C`, and `D` of the LTI object `sys`, corresponding to those in the model $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$, $\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$.
`sys = tf(num,den)`	Creates an LTI object in transfer function form, where the vector `num` is the vector of coefficients of the transfer function numerator, arranged in descending order, and `den` is the vector of coefficients of the denominator, also arranged in descending order.
`sys2=tf(sys1)`	Creates the transfer function model `sys2` from the state model `sys1`.
`sys1=ss(sys2)`	Creates the state model `sys1` from the transfer function model `sys2`.
`[num, den] = tfdata(sys, 'v')`	Extracts the coefficients of the numerator and denominator of the transfer function model `sys`. When the optional parameter `'v'` is used, if there is only one transfer function, the coefficients are returned as vectors rather than as cell arrays.

5.3.5 LINEAR ODE SOLVERS

The Control System Toolbox provides several solvers for linear models. These solvers are categorized by the type of input function they can accept: zero input, impulse input, step input, and a general input function.

5.3.6 THE `initial` FUNCTION

The `initial` function computes and plots the free response of a state model. This is sometimes called the *initial condition response* or the *undriven response* in the MATLAB documentation. The basic syntax is `initial(sys,x0)`, where `sys` is the LTI object in state variable form, and `x0` is the initial condition vector. The time span and number of solution points are chosen automatically.

Free Response of the Two-Mass Model **EXAMPLE 5.3.2**

■ **Problem**

Compute the free response $x_1(t)$ and $x_2(t)$ of the state model derived in Example 5.2.3, for $x_1(0) = 5$, $\dot{x}_1(0) = -3$, $x_2(0) = 1$, and $\dot{x}_2(0) = 2$. The model is

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= \frac{1}{5}(-5x_1 + 4x_2 - 12x_3 + 8x_4) \\
\dot{x}_4 &= \frac{1}{3}[4x_1 - 4x_2 + 8x_3 - 8x_4 + f(t)]
\end{aligned}$$

or

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}f(t)$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & \frac{4}{5} & -\frac{12}{5} & \frac{8}{5} \\ \frac{4}{3} & -\frac{4}{3} & \frac{8}{3} & -\frac{8}{3} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{3} \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

■ **Solution**

We must first relate the initial conditions given in terms of the original variables to the state variables. From the definition of the state vector $\mathbf{x}$, we see that $x_1(0) = 5$, $x_2(0) = 1$, $x_3(0) = -3$, $x_4(0) = 2$. Next we must define the model in state-variable form. The system `sys4` created in Example 5.3.1 specified two outputs, x_1 and x_2. Because we want to obtain only one output here

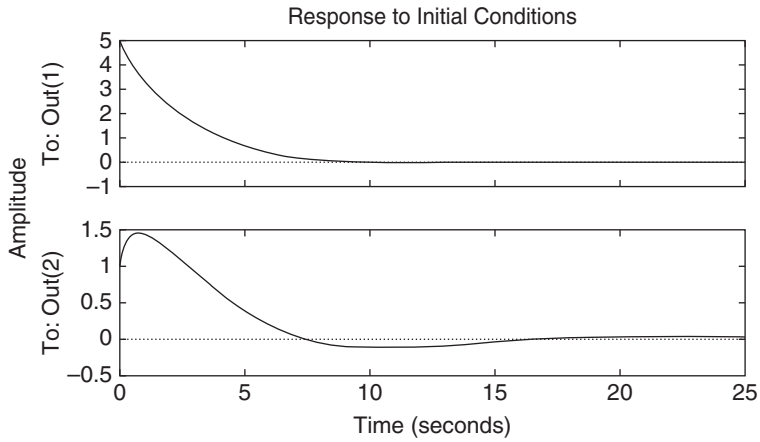


Figure 5.3.1 Response for Example 5.3.2 plotted with the `initial` function.

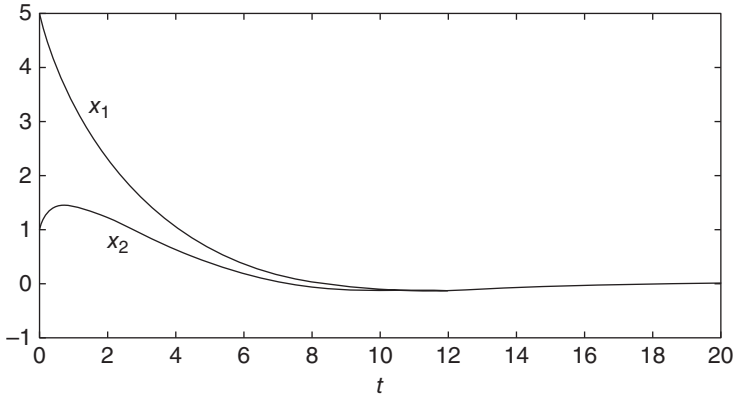


Figure 5.3.2 Response for Example 5.3.2 plotted with the `plot` function.

(x_1), we must create a new state model using the same values for the **A** and **B** matrices, but now using

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The MATLAB program is as follows.

```
A = [0, 0, 1, 0; 0, 0, 0, 1;...
     -1, 4/5, -12/5, 8/5; 4/3, -4/3, 8/3, -8/3];
B = [0; 0; 0; 1/3];
C = [1, 0, 0, 0; 0, 1, 0, 0]; D = [0; 0]
sys5 = ss(A, B, C, D);
initial(sys5, [5, 1, -3, 2])
```

The plot of $x_1(t)$ and $x_2(t)$ will be displayed on the screen (see Figure 5.3.1).

To plot x_1 and x_2 on the same plot you can replace the last line with the following two lines.

```
[y,t] = initial(sys,[5,1,-3,2]);
plot(t,y),gtext('x_1'),gtext('x_2'),xlabel('t')
```

The resulting plot is shown in Figure 5.3.2.

To specify the final time `tfinal`, use the syntax `initial(sys,x0,tfinal)`. To specify a vector of times of the form `t = (0:dt:tfinal)`, at which to obtain the

solution, use the syntax `initial(sys,x0,t)`. When called with left-hand arguments, as `[y, t, x] = initial(sys,x0, ...)`, the function returns the output response `y`, the time vector `t` used for the simulation, and the state vector `x` evaluated at those times. The columns of the matrices `y` and `x` are the outputs and the states, respectively. The number of rows in `y` and `x` equals `length(t)`. No plot is drawn. The syntax `initial(sys1,sys2, ...,x0,t)` plots the free response of multiple LTI systems on a single plot. The time vector `t` is optional. You can specify line color, line style, and marker for each system; for example, `initial(sys1,'r',sys2, 'y--',sys3,'gx',x0)`.

5.3.7 THE `impulse`, `step`, AND `lsim` FUNCTIONS

You may use the `impulse`, `step`, and `lsim` functions with state-variable models the same way they are used with transfer function models. However, when used with state-variable models, there are some additional features available, which we illustrate with the `step` function. When called with left-hand arguments, as `[y, t] = step(sys, ...)`, the function returns the output response `y` and the time vector `t` used for the simulation. No plot is drawn. The array `y` is $(p \times q \times m)$, where p is `length(t)`, q is the number of outputs, and m is the number of inputs. To obtain the state vector solution for state-space models, use the syntax `[y, t, x] = step(sys, ...)`.

To use the `lsim` function for nonzero initial conditions with a state-space model, use the syntax `lsim(sys,u,t,x0)`. The initial condition vector `x0` is needed only if the initial conditions are nonzero.

These functions are summarized in Table 5.3.2.

Table 5.3.2 Basic syntax of linear solvers for state variable models.

`initial(sys,x0,tfinal)`	Generates a plot of the free response of the state variable model `sys`, for the initial conditions specified in the array `x0`. The final time `tfinal` is optional.
`initial(sys,x0,t)`	Generates the free response plot using the user-supplied array of regularly-spaced time values `t`.
`[y,t,x]=initial(sys,x0,...)`	Generates and saves the free response in the array `y` of the output variables, and in the array `x` of the state variables. No plot is produced.
`step(sys)`	Generates a plot of the unit step response of the LTI model `sys`.
`step(sys,t)`	Generates a plot of the unit step response using the user-supplied array of regularly-spaced time values `t`.
`[y,t]= step(sys)`	Generates and saves the unit step response in the arrays `y` and `t`. No plot is produced.
`[y,t,x]=step(sys,...)`	Generates and saves the free response in the array `y` of the output variables, and in the array `x` of the state variables, which is optional. No plot is produced.
`impulse(sys)`	Generates and plots the unit impulse response of the LTI model `sys`. The extended syntax is identical to that of the `step` function.
`lsim(sys,u,t,x0)`	Generates a plot of the total response of the state variable model `sys`. The array `u` contains the values of the forcing function, which must have the same number of values as the regularly-spaced time values in the array `t`. The initial conditions are specified in the array `x0`, which is optional if the initial conditions are zero.
`[y,x]= lsim(sys,u,t,x0)`	Generates and saves the total response in the array `y` of the output variables, and in the array `x` of the state variables, which is optional. No plot is produced.

Recall that the total response of a linear model is the sum of the free response and the forced response. The free response can be obtained with the `initial` function, and the forced response obtained with the `step` function, if the forcing function is a step. Keep in mind that to add the responses, the free and forced solutions must have the same time span and must have the same time spacing. The following example shows how this is done.

EXAMPLE 5.3.3 Total Response of a Two-Mass Model

■ Problem

Obtain the total response $x_1(t)$ and $x_2(t)$ of the two-mass model given in Example 5.3.2, using the same initial conditions but now subjected to a step input of magnitude 3.

■ Solution

We first define the **A**, **B**, **C**, and **D** matrices and then create the LTI model `sys`. Then we compute the step response, saving it in the arrays `ystep` and `t`. Note that the `step` function automatically selects a time span and a time spacing for the array `t`. We then use this array to compute the free response `yfree`. Finally we add the two arrays `ystep` and `yfree` to obtain the total response. Note that we could not add these two arrays if they did not have the same number of points. Note also, that if they had different time increments, we could add them, but the sum would be meaningless. The following script file shows the procedure. The resulting plot is shown Figure 5.3.3.

```
% InitialPlusStep.m
A = [0,0,1,0;0,0,0,1;-1,4/5,-12/5,8/5;4/3,-4/3,8/3,-8/3];
B = [0;0;0;1/3];
C = [1,0,0,0;0,1,0,0];
D = [0;0];
sys = ss(A,B,C,D);
[ystep,t] = step(3*sys);
yfree = initial(sys,[5,1,-3,2],t);
y = yfree + ystep;
plot(t,y),xlabel('t'),gtext('x_1'),gtext('x_2')
```

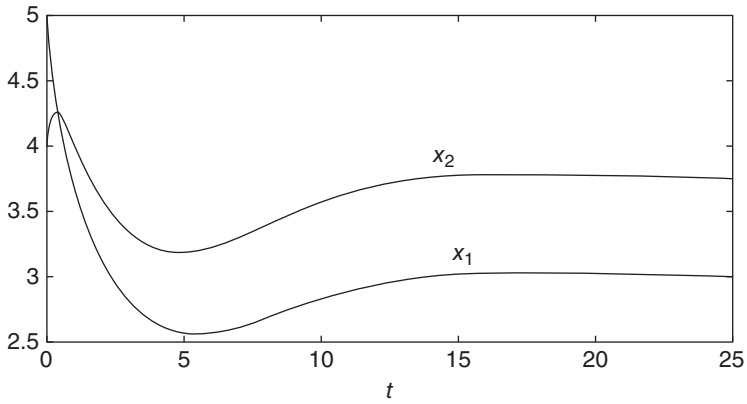


Figure 5.3.3 Step plus free response for Example 5.3.3.

5.3.8 OBTAINING THE CHARACTERISTIC POLYNOMIAL

The MATLAB command `poly` can find the characteristic polynomial that corresponds to a specified state matrix **A**. For example, the matrix **A** given in Example 5.2.2 is, for $m = 1$, $c = 5$, $k = 6$,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix} \tag{5.3.6}$$

This matrix corresponds to the equation $\ddot{x} + 5\dot{x} + 6x = 0$, whose characteristic polynomial is $s^2 + 5s + 6 = 0$.

The coefficients of its characteristic polynomial are found by typing

```
≫A = [0, 1; -6, -5];
≫poly(A)
```

MATLAB returns the answer `[1, 5, 6]`, which corresponds to the polynomial $s^2 + 5s + 6$. The `roots` function can be used to compute the characteristic roots; for example, `roots(poly(A))`, which gives the roots `[-2, -3]`.

MATLAB provides the `eig` function to compute the characteristic roots without obtaining the characteristic polynomial, when the model is given in the state-variable form. Its syntax is `eig(A)`. (The function's name is an abbreviation of *eigenvalue,* which is another name for characteristic root.) For example, typing `eig([0, 1; -6, -5])` returns the roots `[-2, -3]`.

5.4 THE MATLAB ode FUNCTIONS

Recall that we can categorize differential equations as *linear* or *nonlinear*. Linear differential equations are recognized by the fact that they contain only linear functions of the dependent variable and its derivatives. Nonlinear functions of the *independent* variable do *not* make a differential equation nonlinear. For example, the following equations are linear.

$$\dot{y} + 3y = 5 + t^2 \qquad \dot{y} + 3t^2 y = 5 \qquad \ddot{y} + 7\dot{y} + t^2 x = \sin t$$

whereas the following equations are nonlinear:

$$\dot{y} = -y^2 \sin t \qquad \text{because of } y^2$$

$$\ddot{y} + 5\dot{y}^2 + 6y = 4 \qquad \text{because of } \dot{y}^2$$

$$\ddot{y} + 4y\dot{y} + 3y = 10 \qquad \text{because of } y\dot{y}$$

The equation $\ddot{y} + 7\dot{y} + t^2 x = \sin t$ is a *variable-coefficient* differential equation, so named because one of its coefficients (t^2) is a function of the independent variable t.

5.4.1 SELECTING A SOLUTION METHOD

Recall that the Laplace transform method and the state variable MATLAB methods presented in Section 5.3.1 cannot be used to solve variable coefficient equations and nonlinear equations.

It is possible sometimes, mainly for first order equations, to obtain the closed-form solution of a nonlinear differential equation. If not, then we must solve the equation numerically. In this section, we introduce numerical methods for solving differential equations. First we will treat first-order equations, and then we will show how to extend the techniques to higher order equations.

The essence of a numerical method is to convert the differential equation into a *difference equation* that can be programmed on a digital computer. Numerical algorithms differ partly as a result of the specific procedure used to obtain the difference equations. In general, as the accuracy of the approximation is increased, so is the complexity of the programming involved. It is important to understand the concept of *step size* and its effects on solution accuracy. In order to provide a simple introduction to these issues, we begin with the simplest numerical method, the *Euler method*.

5.4.2 THE EULER METHOD

The Euler method is the simplest algorithm for numerical solution of a differential equation. It usually gives the least accurate results, but provides a basis for understanding more sophisticated methods. Consider the equation

$$\frac{dy}{dt} = r(t)y(t) \tag{5.4.1}$$

where $r(t)$ is a known function. From the definition of the derivative,

$$\frac{dy}{dt} = \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t}$$

If the time increment Δt is chosen small enough, the derivative can be replaced by the approximate expression

$$\frac{dy}{dt} \approx \frac{y(t + \Delta t) - y(t)}{\Delta t}$$

Assume that the right-hand side of (5.4.1) remains constant over the time interval $[t, t + \Delta t]$, and replace (5.4.1) by the following approximation:

$$\frac{y(t + \Delta t) - y(t)}{\Delta t} = r(t)y(t)$$

or

$$y(t + \Delta t) = y(t) + r(t)y(t)\Delta t \tag{5.4.2}$$

The smaller Δt is, the more accurate are our two assumptions leading to (5.4.2). This technique for replacing a differential equation with a difference equation is the Euler method. The increment Δt is called the *step size*.

Equation (5.4.2) can be written in more convenient form by replacing t with t_k as follows.

$$y(t_k + \Delta t) = y(t_{k+1}) = y(t_k) + r(t_k)y(t_k)\Delta t \tag{5.4.3}$$

where $t_{k+1} = t_k + \Delta t$.

The Euler method for the general first order equation $\dot{y} = f(t, y)$ is

$$y(t_{k+1}) = y(t_k) + \Delta t f[t_k, y(t_k)] \tag{5.4.4}$$

This equation can be applied successively at the times t_k by putting it in a `for` loop. (See Table 5.4.1 for definitions of MATLAB functions used in this section.)

For example, the following script file solves the differential equation $\dot{y} = ry$ and plots the solution over the range $0 \le t \le 0.5$ for the case where $r = -10$ and the initial condition is $y(0) = 2$. The time constant is $\tau = -1/r = 0.1$, and the true solution is $y(t) = 2e^{-10t}$. To illustrate the effect of the step size on the solution's accuracy, we use a step size $\Delta t = 0.02$, which is 20% of the time constant.

Table 5.4.1 MATLAB functions used in this section.

Functions Introduced in This Section	
`axis[x1 x2 y1 y2]`	Sets the minimum and maximum values for the x- and y-axes.
`cos(x)`	Computes $\cos x$.
`end`	Terminates a loop, such as a `for` loop.
`exp(x)`	Computes e^x
`for`	Denotes the beginning of a `for` loop.
`function`	Indicates the beginning of a function file.
`gtext('text')`	Enables placement of text on a plot using the cursor.
`ode45`	Implements the 4th–5th order Runge-Kutta algorithm for solving differential equations.
`sin(x)`	Computes $\sin x$.
`sqrt(x)`	Computes $\sqrt{x}$.
`\theta`	Puts the Greek letter θ in a plot label.
Functions Introduced in Earlier Sections	
`eig(A)`	Computes the eigenvalues of a matrix **A**. (Section 5.3)
`plot(x,y)`	Creates a two-dimensional plot on rectilinear axes (Section 2.9).
`xlabel('text')`	Puts a label on the abscissa of a plot (Section 2.9).
`ylabel('text')`	Puts a label on the ordinate of a plot (Section 2.9).

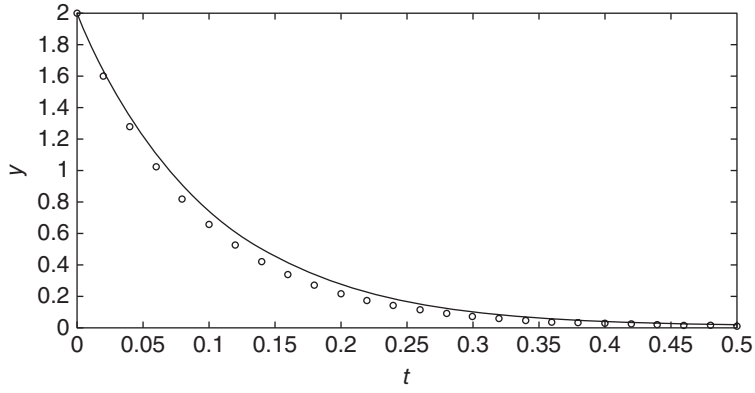


Figure 5.4.1 Euler-method solution for the free response of $\dot{y} = -10y$, $y(0) = 2$.

```
r = -10; delta = 0.02; y(1) = 2;
k = 0;
for time = [delta:delta:.5]
    k = k + 1;
    y(k+1) = y(k) + r*y(k)*delta;
end
t = (0:delta:0.5);
y_exact=2*exp(-10*t);
plot(t,y, 'o',t,y_exact),xlabel('t'),ylabel('y')
```

Figure 5.4.1 shows the results. The numerical solution is shown by the small circles. The exact solution is shown by the solid line. There is some noticeable error. If we had used a step size equal to 5% of the time constant, the error would not be noticeable on the plot.

MATLAB provides functions called *solvers,* that implement several numerical solution methods. The `ode45` solver is sufficient to solve the problems encountered in this text.[1]

[1]The `ode45` solver is based on the 4th and 5th order Runge-Kutta algorithms. For an introduction to Runge-Kutta algorithms, see Appendix E on the text website.

Table 5.4.2 Basic Syntax of the `ode45` solver.

`[t, y] = ode45(@ydot, tspan, y0)` Solves the vector differential equation $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ specified in the function file `ydot`, whose inputs must be t and $\mathbf{y}$, and whose output must be a *column* vector representing $d\mathbf{y}/dt$; that is, $\mathbf{f}(t, \mathbf{y})$. The number of rows in this column vector must equal the order of the equation. The vector `tspan` contains the starting and ending values of the independent variable t, and optionally, any intermediate values of t where the solution is desired. The vector `y0` contains the initial values $\mathbf{y}(t_0)$. The function file must have the two input arguments, `t` and `y`, even for equations where $\mathbf{f}(t, \mathbf{y})$ is not an explicit function of t.

5.4.3 SOLVER SYNTAX

We begin our coverage with several examples of solving first-order equations. Solution of higher-order equations is covered later in this section. When used to solve the equation $\dot{y} = f(t, y)$, the basic syntax is:

```
[t,y] = ode45(@ydot, tspan, y0)
```

where `ydot` is the name of the function file whose inputs must be t and y, and whose output must be a *column* vector representing dy/dt; that is, $f(t, y)$. The number of rows in this column vector must equal the order of the equation. The syntax for the other solvers is identical. Use the MATLAB Editor to create and save the file `ydot`. Note that we use the `'at'` sign, `@`, to specify the function.

The vector `tspan` contains the starting and ending values of the independent variable t, and optionally, any intermediate values of t where the solution is desired. For example, if no intermediate values are specified, `tspan` is `[t0, tfinal]`, where `t0` and `tfinal` are the desired starting and ending values of the independent parameter t. As another example, using `tspan = [0, 5, 10]` forces MATLAB to find the solution at $t = 5$. You can solve equations backward in time by specifying `t0` to the greater than `tfinal`.

The parameter `y0` is the initial value $y(t_0)$. The function file must have two input arguments, `t` and `y`, even for equations where $f(t, y)$ is *not* an explicit function of t. You need not use array operations in the function file because the ODE solvers call the file with scalar values for the arguments.

Table 5.4.1 summarizes the MATLAB functions used in this section. Table 5.4.2 summarizes the basic syntax of the ODE solvers.

As a first example of using a solver, let us solve an equation whose solution is known in closed form, so that we can make sure we are using the method correctly. We can also assess the performance of the `ode45` solver when applied to find an oscillating solution, which can be difficult to obtain numerically.

EXAMPLE 5.4.1 MATLAB Solution of $\dot{y} = -y^2 \sin t$

■ **Problem**

Equations with oscillating solutions provide a good test of the accuracy of a numerical solver. To check our results, we choose an equation whose closed-form solution can be found. We can use separation of variables to solve the following equation, which is nonlinear because of the y^2 term.

$$\dot{y} = -y^2 \sin t \qquad y(0) = 1 \tag{1}$$

$$\int_{y(0)}^{y(t)} \frac{dy}{y^2} = -\int_0^t \sin t \, dt$$

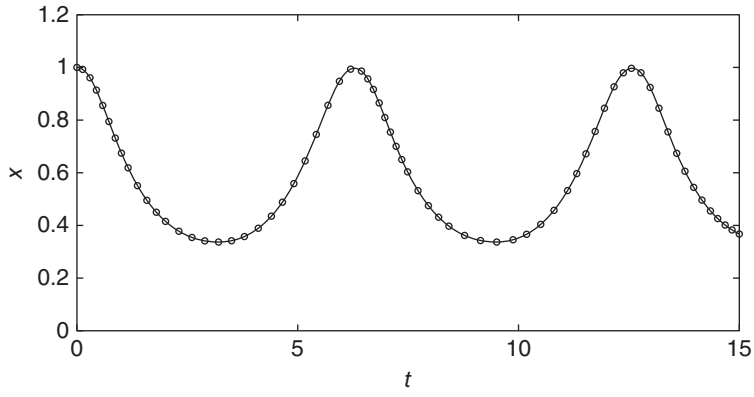


Figure 5.4.2 MATLAB (ode45) and exact solutions of $\dot{y} = -y^2 \sin t$, $y(0) = 1$.

$$-\frac{1}{y(t)} + \frac{1}{y(0)} = \cos t - 1$$

or, since $y(0) = 1$,

$$y(t) = \frac{1}{2 - \cos t}$$

Use the ode45 solver to solve equation (1) for $0 \le t \le 15$.

■ Solution

Create and save the following function file. Give it a unique name of your choice. Here we will name it after the example number 5.4.2, abbreviated as Ex5p4p2.

```
function ydot=Ex5p4p2(t,y)
ydot=-sin(t)*y^2;
```

Note that we need not use array arithmetic (.* and .^) in the expression sin(t)*y^2, because the solver does not call the function ydot with array inputs. Type the following lines in the MATLAB Command window. The solution from ode45 is a discrete set of points stored in the arrays y and t, which we show on the plot as small circles.

```
≫ [t,y] = ode45(@Ex5p4p2,[0,15],1);
≫ y_exact = 1./(2-cos(t));
≫ plot(t,y,'o',t,y_exact),xlabel('t'),ylabel('y'),axis([0 15 0 1.2])
```

Note that we use the t values generated by the ode45 function to compute the exact solution at the same values of t. Note also that in practice we do not know the range of the solution $y(t)$ ahead of time, so we must first run the program using guessed values for the range of $y(t)$. The final choice of 0 and 1.2 gives the plot shown in Figure 5.4.2. The solid line is the exact solution and the circular data markers are the numerical solution. This plot shows that the numerical solution is accurate.

The main application of numerical methods is to solve equations for which a closed-form solution cannot be obtained. The next example shows such an application.

A Rocket-Propelled Sled — EXAMPLE 5.4.2

■ Problem

A rocket-propelled sled on a track is represented in Figure 5.4.3 as a mass m with an applied force f that represents the rocket thrust. The rocket thrust initially is horizontal, but the engine

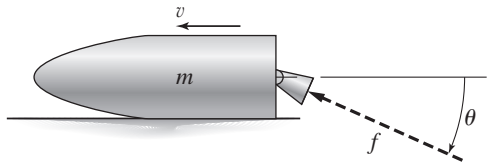


Figure 5.4.3 A rocket-propelled sled.

accidentally pivots during firing and rotates with an angular acceleration of $\ddot{\theta} = \pi/50$ rad/s. Compute the sled's velocity v for $0 \le t \le 6$ if $v(0) = 0$. The rocket thrust is 4000 N and the sled mass is 450 kg.

■ Solution

The sled's equation of motion is $450\dot{v} = 4000 \cos\theta(t)$. To obtain $\theta(t)$, note that

$$\dot{\theta} = \int_0^t \ddot{\theta}\, dt = \frac{\pi}{50} t$$

and

$$\theta = \int_0^t \dot{\theta}\, dt = \int_0^t \frac{\pi}{50} t\, dt = \frac{\pi}{100} t^2$$

Thus the equation of motion becomes

$$\dot{v} = \frac{80}{9} \cos\left(\frac{\pi}{100} t^2\right) \tag{1}$$

The solution is formally given by

$$v(t) = \frac{80}{9} \int_0^t \cos\left(\frac{\pi}{100} t^2\right) dt$$

Unfortunately, no closed-form solution is available for the integral, which is called *Fresnel's cosine integral*. The value of the integral has been tabulated numerically, but we will use a MATLAB ODE solver to obtain the solution.

First create the following user-defined function file, which is based on equation (1).

```
function vdot = sled(t,v)
vdot = 80*cos(pi*t^2/100)/9;
```

As a check on our results, we will use the solution for $\theta = 0$ as a comparison. The equation of motion for this case is $\dot{v} = 80/9$, which gives $v(t) = 80t/9$. The following session solves the equation and plots the two solutions, which are shown in Figure 5.4.4.

```
[t,v] = ode45(@sled,[0 6],0);
plot(t,v,t,(80*t/9)),xlabel('t (s)'),...
   ylabel('v (m/s)'),gtext('\theta = 0'),gtext('\theta \neq 0')
```

We can make two observations that help us determine whether or not our numerical solution is correct. From the plot we see that the solution for $\theta \neq 0$ is almost identical to the solution for $\theta = 0$, for small values of θ. This is correct because $\cos\theta \approx 1$ for small values of θ. As θ increases, we would expect the velocity to be smaller than the velocity for $\theta = 0$ because the horizontal component of the thrust is smaller. The plot confirms this.

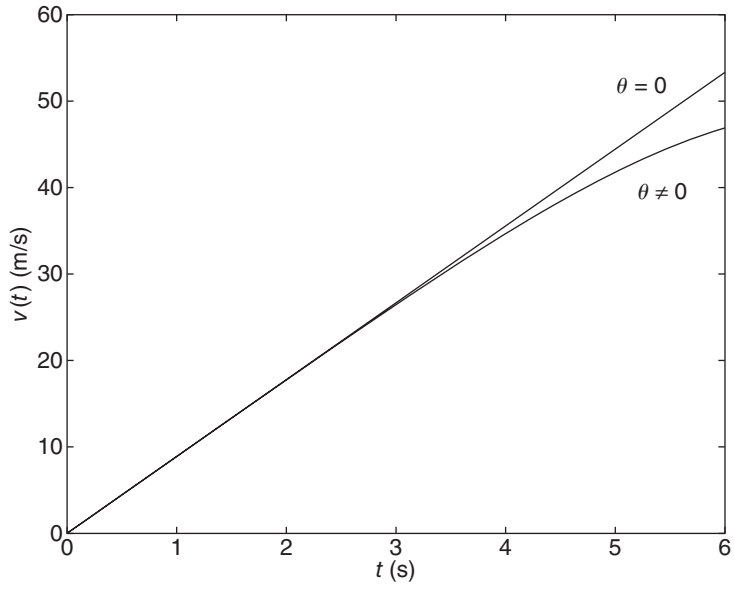


Figure 5.4.4 Speed response of the sled for $\theta = 0$ and $\theta \neq 0$.

5.4.4 EXTENSION TO HIGHER ORDER EQUATIONS

To use the ODE solvers to solve an equation of order two or greater, you must first write the equation in state-variable form, as a set of first-order equations, and then create a function file that computes the derivatives of the state variables. One advantage of using the `ode45` solver, even for *linear* equations, is that it can handle nonzero initial conditions and variable coefficients, whereas the `step` and `impulse` functions cannot.

Sinusoidal response of a mass-spring-damper system

EXAMPLE 5.4.3

■ Problem

Consider the second-order equation $5\ddot{y} + 7\dot{y} + 4y = f(t)$. Define the variables, $x_1 = y$ and $x_2 = \dot{y}$. The state-variable form is

$$\dot{x}_1 = x_2 \tag{1}$$

$$\dot{x}_2 = \frac{1}{5}f(t) - \frac{4}{5}x_1 - \frac{7}{5}x_2 \tag{2}$$

Now write a function file that computes the values of $\dot{x}_1$ and $\dot{x}_2$ and stores them in a *column* vector. To do this, we must first have a function specified for $f(t)$. Suppose that $f(t) = \sin t$. Solve the equation over the interval $0 \leq t \leq 6$ using the initial conditions $y(0) = 3$ and $\dot{y}(0) = 9$.

■ Solution

The required file is

```
function xdot = Ex5p4p3(t,x)
% Computes derivatives of two equations
xdot(1) = x(2);
xdot(2) = (1/5)*(sin(t)-4*x(1)-7*x(2));
xdot = [xdot(1); xdot(2)];
```

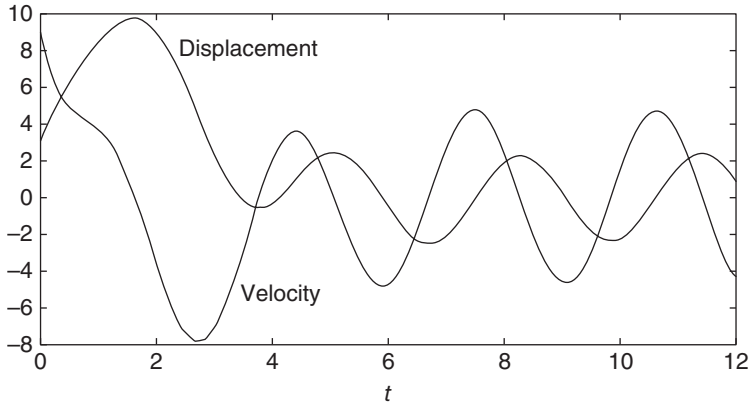


Figure 5.4.5 Sinusoidal response of a mass-spring-damper system.

Note that `xdot(1)` represents $\dot{x}_1$, `xdot(2)` represents $\dot{x}_2$, `x(1)` represents x_1, and `x(2)` represents x_2. Once you become familiar with the notation for the state-variable form, you will see that the preceding code could be replaced with the following shorter form.

```
function xdot = example1(t,x)
% Computes derivatives of two equations
xdot = [x(2); (1/5)*(sin(t)-4*x(1)-7*x(2))];
```

Suppose we want to solve (5.4.5) and (5.4.6) for $0 \leq t \leq 6$ with the initial conditions $y(0) = x_1(0) = 3$ and $\dot{y}(0) = x_2(0) = 9$. Then the initial condition for the *vector* `x` is `[3, 9]`. To use `ode45`, you type

```
[t, x] = ode45(@Ex5p4p3, [0, 6], [3, 9]);
```

Each row in the vector `x` corresponds to a time returned in the column vector `t`. If you type `plot(t,x)`, you will obtain a plot of both x_1 and x_2 versus t. Note that `x` is a matrix with two columns; the first column contains the values of x_1 at the various times generated by the solver. The second column contains the values of x_2. Thus, to plot only x_1, type `plot(t,x(:,1))`. Figure 5.4.5 shows the results after labeling with the MATLAB plot Editor.

Most variable coefficient equations do not have closed form solutions, and so we must usually solve these numerically.

EXAMPLE 5.4.4 Response Due to an Aging Spring

■ Problem

Consider a mass-spring-damper system in which the spring element gets weaker with time due to metal fatigue. Suppose the spring constant varies with time as follows:

$$k = 20(1 + e^{-t/10})$$

The equation of motion is

$$m\ddot{x} + c\dot{x} + 20(1 + e^{-t/10})x = f(t)$$

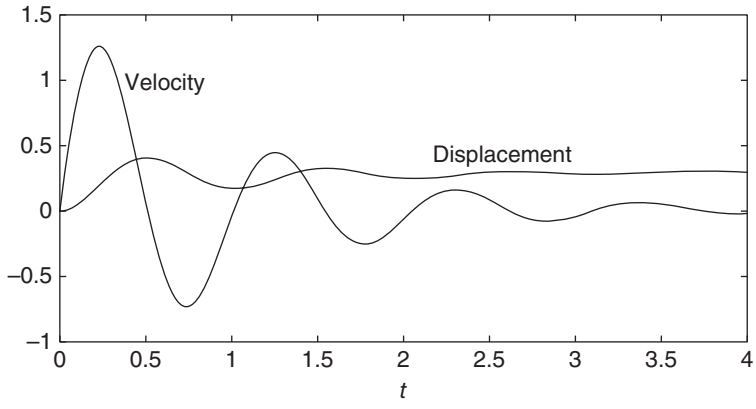


Figure 5.4.6 Displacement and velocity of a mass with a damper and aging spring.

Use the values $m = 1$, $c = 2$, and $f = 10$, and solve the equations for zero initial conditions over the interval $0 \leq t \leq 4$.

■ Solution

Defining the state variables to be $x_1 = y$ and $x_2 = \dot{y}$, the state variable equations are

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{1}{m} f(t) - \frac{20(1 + e^{-t/10})}{m} x_1 - \frac{c}{m} x_2$$

If $m = 1$, $c = 2$, and $f = 10$, these equations become

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = 10 - 20(1 + e^{-t/10})x_1 - 2x_2$$

Create a function file that computes the values of $\dot{x}_1$ and $\dot{x}_2$ and stores them in a column vector. This file is shown below.

```
function xdot = aging_spring(t,x)
% Mass-spring-damper with aging spring
k = 20*(1+exp(-t/10));
xdot = [x(2);10-k*x(1)-2*x(2)];
```

To solve these equations for zero initial conditions over the interval $0 \leq t \leq 4$, enter the following in the Command window.

```
≫ [t,x] = ode45(@aging_spring,[0,4],[0,0]);
≫ plot(t,x),xlabel('t'),gtext('Velocity'),gtext('Displacement')
```

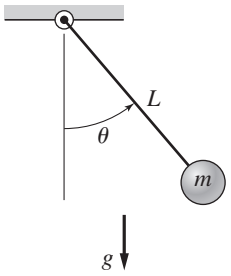
The result is shown in Figure 5.4.6.

When solving nonlinear equations, sometimes it is possible to check the numerical results by using an approximation that reduces the equation to a linear one. Example 5.4.5 illustrates such an approach with a second-order equation.

EXAMPLE 5.4.5

A Nonlinear Pendulum Model

Figure 5.4.7 A pendulum.



■ Problem

By studying the dynamics of a pendulum like that shown in Figure 5.4.7, we can better understand the dynamics of machines such as a robot arm. The pendulum shown consists of a concentrated mass m attached to a rod whose mass is small compared to m. The rod's length is L. The equation of motion for this pendulum is

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0 \tag{1}$$

Suppose that $L = 1$ m and $g = 9.81$ m/s^2. Use MATLAB to solve this equation for $\theta(t)$ for two cases: $\theta(0) = 0.5$ rad, and $\theta(0) = 0.8\pi$ rad. In both cases $\dot{\theta}(0) = 0$. Discuss how to check the accuracy of the results.

■ Solution

If we use the small angle approximation $\sin \approx \theta$, the equation becomes

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \tag{2}$$

which is linear and has the solution:

$$\theta(t) = \theta(0)\cos\sqrt{\frac{g}{L}}t \tag{3}$$

Thus the amplitude of oscillation is $\theta(0)$ and the period is $P = 2\pi\sqrt{L/g} = 2$ s. We can use this information to select a final time, and to check our numerical results.

First rewrite the pendulum equation (1) as two first order equations. To do this, let $x_1 = \theta$ and $x_2 = \dot{\theta}$. Thus

$$\dot{x}_1 = \dot{\theta} = x_2$$
$$\dot{x}_2 = \ddot{\theta} = -\frac{g}{L}\sin x_1 = -9.81\sin x_1$$

The following function file is based on the last two equations. Remember that the output `xdot` must be a *column* vector.

```
function xdot = pendulum(t,x)
xdot = [x(2); -9.81*sin(x(1))];
```

The function file is called as follows. The vectors `ta` and `xa` contain the results for the case where $\theta(0) = 0.5$. The vectors `tb` and `xb` contain the results for $\theta(0) = 0.8\pi$.

```
[ta, xa] = ode45(@pendulum, [0, 5], [0.5, 0]);
[tb, xb] = ode45(@pendulum, [0, 5], [0.8*pi, 0]);
plot(ta,xa(:,1),tb,xb(:,1)),xlabel('Time (s)'),...
   ylabel('Angle (rad)'),gtext('Case 1'),gtext('Case 2')
```

The results are shown in Figure 5.4.8. The amplitude remains constant, as predicted by the small angle analysis, and the period for the case where $\theta(0) = 0.5$ is a little larger than 2 s, the value predicted by the small angle analysis. So we can place some confidence in the numerical procedure. For the case where $\theta(0) = 0.8\pi$, the period of the numerical solution is about 3.3 s. This illustrates an important property of nonlinear differential equations. The free response of a linear equation has the same period for any initial conditions; however, the form

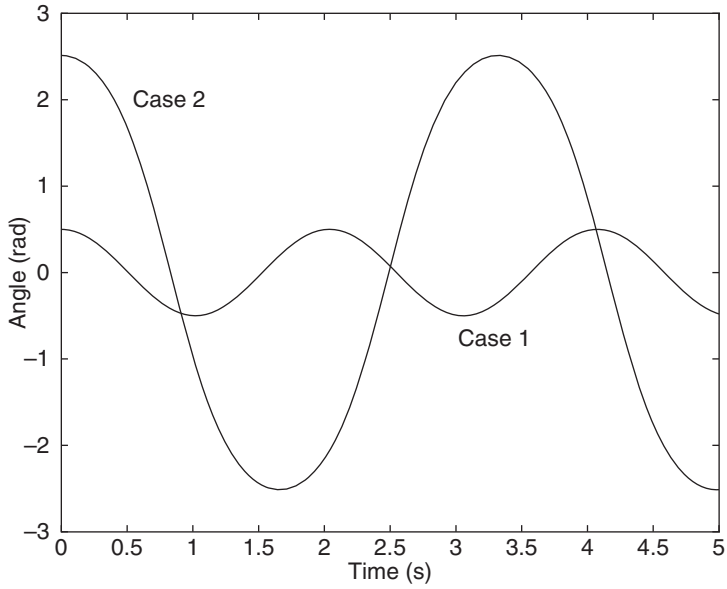


Figure 5.4.8 The pendulum angle as a function of time for two starting positions.

of the free response of a nonlinear equation often depends on the particular values of the initial conditions.

In the previous example, the values of g and L did not appear in the function `pendulum(t,x)`. Now suppose you want to obtain the pendulum response for different lengths L or different gravitational accelerations g. Starting with MATLAB 7, the preferred method is to use a nested function. Nested functions are discussed in [Palm, 2008] and [Palm, 2011]. The following program shows how this is done.

```
function pendula
g = 9.81; L = 0.75; % First case.
tfinal = 6*pi*sqrt(L/g); % Approximately 3 periods.
[t1, x1] = ode45(@pendulum, [0,tfinal], [0.4, 0]);
%
g = 1.63; L = 2.5; % Second case.
tfinal = 6*pi*sqrt(L/g); % Approximately 3 periods.
[t2, x2] = ode45(@pendulum, [0,tfinal], [0.2, 0]);
plot(t1, x1(:,1), t2, x2(:,1)), ...
    xlabel ('time (s)'), ylabel ('\theta (rad)')
% Nested function.
    function xdot = pendulum(t,x)
    xdot = [x(2);-(g/L)*sin(x(1))];
    end
end
```

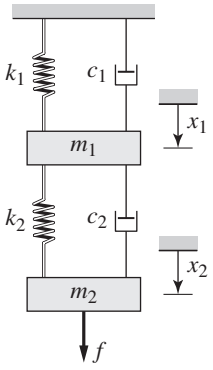
5.4.5 MATRIX METHODS

We can use matrix operations to reduce the number of lines to be typed in the derivative function file.

EXAMPLE 5.4.6

A Fourth Order System

Figure 5.4.9 A two-mass system.



■ Problem

Consider the two-mass system considered in Section 5.2 and shown again in Figure 5.4.9. The state variable model $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$ developed in Example 5.2.3 is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 4/5 & -12/5 & 8/5 \\ 4/3 & -4/3 & 8/3 & -8/3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/3 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} \qquad \mathbf{u} = [f(t)]$$

Solve this model over the interval $0 \leq t \leq 30$ for the initial conditions $x_1(0) = 5$, $x_2(0) = 4$, $\dot{x}_1(0) = -3$, and $\dot{x}_2(0) = 2$. The forcing function is a modified step function: $f(t) = 1 - e^{-t/8}$.

■ Solution

The following function file shows how to use matrix operations with the `ode45` function.

```
function xdot = Ex5p4p6(t,x)
%Two-mass system with non-zero initial conditions and time
%varying input.
f = 1-exp(-t/8);
A = [0,0,1,0;0,0,0,1;-1,4/5,-12/5,8/5;4/3,-4/3,8/3,-8/3];
B = [0;0;0;1/3];
xdot = A*x+B*f;
```

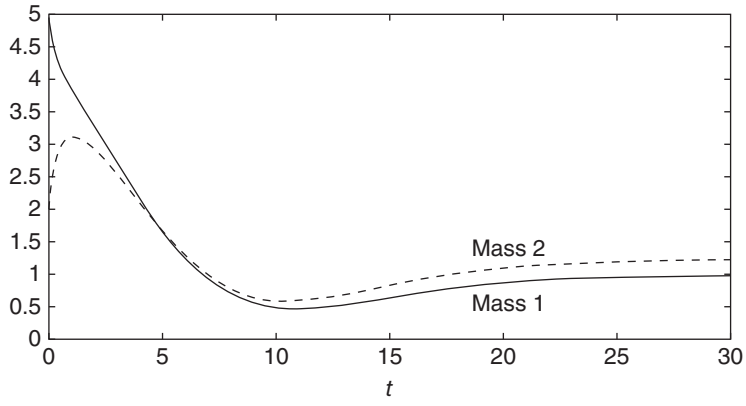
Note that `xdot` will be a column vector because of the definition of matrix-vector multiplication.

The solution for x_1 and x_2 can be plotted by typing the following in the Command window.

```
≫plot(t,x(:,1),t,x(:,2),'--'),xlabel('t'),...
    gtext('Mass 1'),gtext('Mass 2')
```

The result is shown in Figure 5.4.10.

Figure 5.4.10 Response of a two-mass system to modified step input.



The characteristic roots and the time constants may be found by typing

```
≫A = [0,0,1,0;0,0,0,1;-1,4/5,-12/5,8/5;4/3,-4/3,8/3,-8/3];
≫r = eig(A);
≫tau = -1./real(r)
```

The results for the eigenvalues are -4.0595, -0.5261, $-0.2405 \pm 0.2588j$. The time constants are $\tau = 0.2463, 1.9006, 4.1575, 4.1575$. The dominant time constant is 4.1575, and four times that value is 16.63, but steady state will be reached later than this time because the time constant of the input function $f(t)$ is 8. Thus steady state should be reached after about $t = 4(8) = 32$. This is verified by the plot on Figure 5.4.10.

Table 5.4.2 summarizes the basic syntax of the `ode45` solver.

PART III. SIMULINK METHODS

5.5 SIMULINK AND LINEAR MODELS

Simulink is built on top of MATLAB, so you must have MATLAB to use Simulink. It is included in the Student Edition of MATLAB and is also available separately from The MathWorks, Inc. It provides a graphical user interface that uses various types of elements called *blocks* to create a simulation of a dynamic system; that is, a system that can be modeled with differential or difference equations whose independent variable is time. For example, one block type is a multiplier, another performs a sum, and another is an integrator. Its graphical interface enables you to position the blocks, resize them, label them, specify block parameters, and interconnect the blocks to describe complicated systems for simulation.

Type `simulink` in the MATLAB Command window to start Simulink. The Simulink Library Browser window opens. See Figure 5.5.1. To create a new model, select **File** > **New** > **Model** in the Browser. A new untitled window opens for you to create the model. To select a block from the Library Browser, double-click on the appropriate library category and a list of blocks within that category then appears, as shown in Figure 5.5.1. Figure 5.5.1 shows the result of double-clicking on the Continuous library, then clicking on the Derivative block. Click on the block name or icon, hold the mouse button down, drag the block to the new model window, and release the button. You can access help for that block by right-clicking on its name or icon, and selecting **Help** from the drop-down menu.

Simulink model files have the default extension `.slx` as of Release 2012b. The `.slx` format is a new compressed format that provides smaller file sizes. The `.mdl` format is still available as a save option. Use the **File** menu in the model window to Open, Close, and Save model files. To print the block diagram of the model, select **Print** on the **File** menu. Use the **Edit** menu to copy, cut and paste blocks. You can also use the mouse for these operations. For example, to delete a block, click on it and press the **Delete** key.

Getting started with Simulink is best done through examples, which we now present.

5.5.1 SIMULATION DIAGRAMS

You construct Simulink models by constructing a diagram that shows the elements of the problem to be solved. Such diagrams are called *simulation diagrams*. Consider the

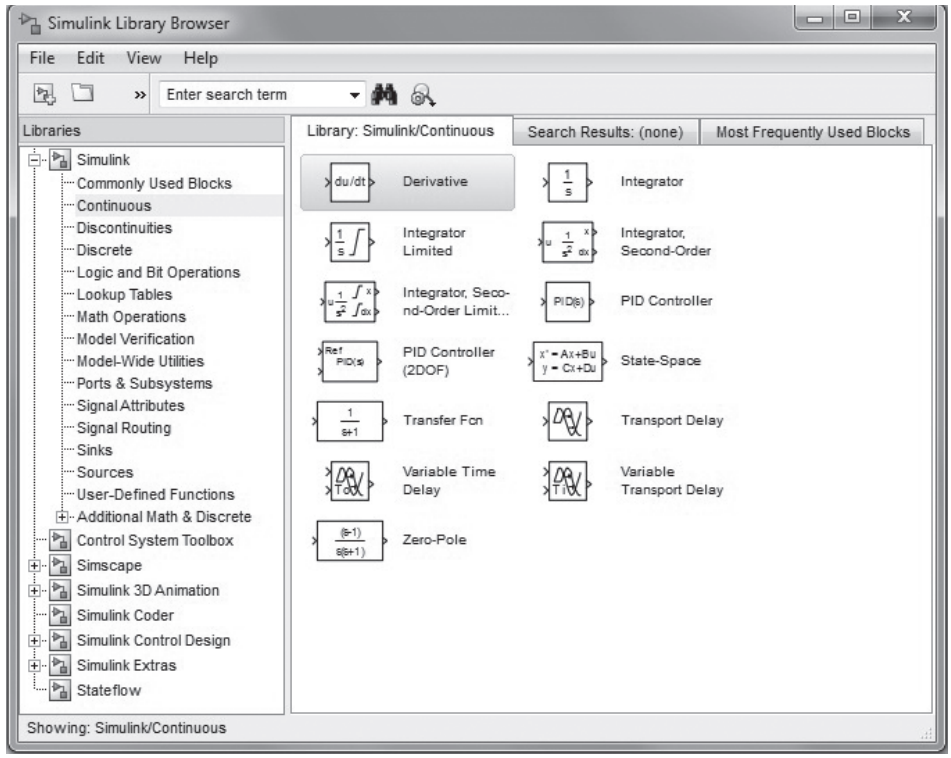


Figure 5.5.1 Simulink Library Browser showing the CONTINUOUS LIBRARY being selected.

equation $\dot{y} = 10f(t)$. Its solution can be represented symbolically as

$$y(t) = \int 10f(t)\,dt$$

which can be thought of as two steps, using an intermediate variable x:

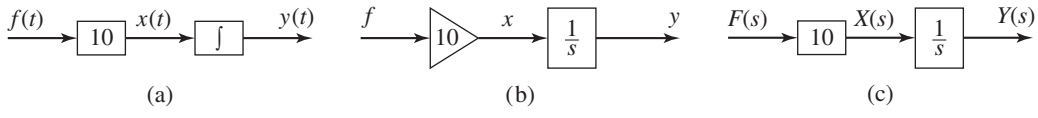
$$x(t) = 10f(t) \qquad \text{and} \qquad y(t) = \int x(t)\,dt$$

This solution can be represented graphically by the simulation diagram shown in Figure 5.5.2a. The arrows represent the variables y, x, and f. The blocks represent cause-and-effect processes. Thus, the block containing the number 10 represents the process $x(t) = 10f(t)$, where $f(t)$ is the cause (the *input*) and $x(t)$ represents the effect (the *output*). This type of block is called a *multiplier* or *gain* block.

The block containing the integral sign $\int$ represents the integration process $y(t) = \int x(t)\,dt$, where $x(t)$ is the cause (the *input*) and $y(t)$ represents the effect (the *output*). This type of block is called an *integrator* block.

There is some variation in the notation and symbols used in simulation diagrams. Part (b) of Figure 5.5.2 shows one variation. Instead of being represented by a box, the

Figure 5.5.2 Simulation diagrams for $\dot{y} = 10f(t)$.



(a) (b) (c)

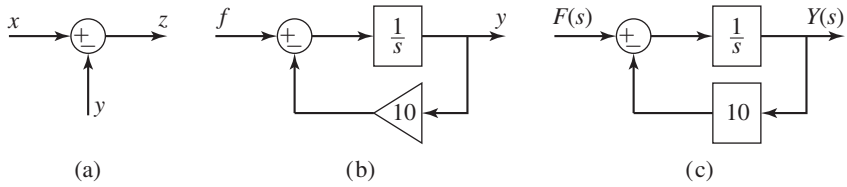


Figure 5.5.3 (a) The summer element. (b) Simulation diagram for $\dot{y} = f(t) - 10y$. (c) Block diagram.

multiplication process is now represented by a triangle like that used to represent an electrical amplifier, hence the name *gain* block.

Part (c) of Figure 5.5.2 shows the block diagram representation. If you are familiar with block diagrams, you can immediately create a Simulink model from the block diagram.

In addition, the integration symbol in the integrator block has been replaced by the symbol $1/s$, which represents integration in Laplace transform notation. Thus the equation $\dot{y} = 10f(t)$ is represented by $sy = 10f$, and the solution is represented as

$$y = \frac{10f}{s}$$

or as the two equations

$$x = 10f \qquad \text{and} \qquad y = \frac{1}{s}x$$

Another element used in simulation diagrams is the *summer* that, despite its name, is used to subtract as well as to sum variables. Its symbol is shown in Figure 5.5.3a. The symbol represents the equation $z = x - y$. Note that a plus or minus sign is required for each input arrow.

The summer symbol can be used to represent the equation $\dot{y} = f(t) - 10y$, which can be expressed as

$$y(t) = \int [f(t) - 10y]\, dt$$

or as

$$y = \frac{1}{s}(f - 10y)$$

You should study the simulation diagram shown in part (b) of Figure 5.5.3 to confirm that it represents this equation.

Part (c) of Figure 5.5.3 shows the corresponding block diagram. Once again, note the similarity with the Simulation diagram.

Figure 5.5.2b forms the basis for developing a Simulink model to solve the equation $\dot{y} = f(t)$.

Simulink Solution of $\dot{y} = 10 \sin t$ — EXAMPLE 5.5.1

■ Problem

Let us use Simulink to solve the following problem for $0 \le t \le 13$.

$$\frac{dy}{dt} = 10 \sin t \qquad y(0) = 0$$

The exact solution is $y(t) = 10(1 - \cos t)$.

■ Solution

To construct the simulation, do the following steps. Refer to Figure 5.5.4.

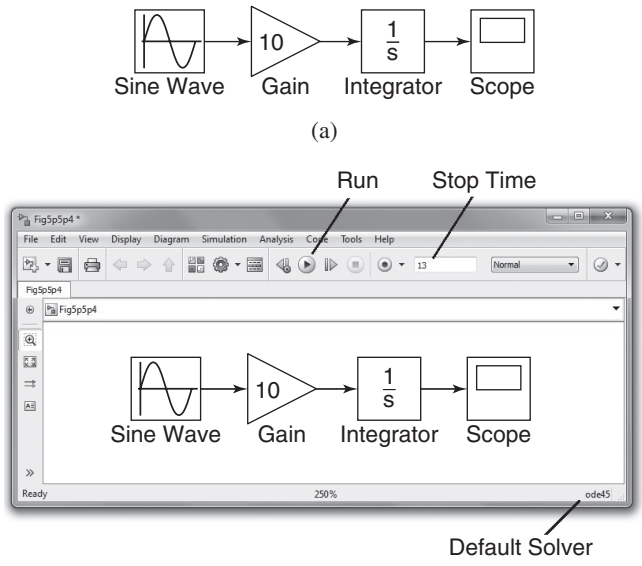


Figure 5.5.4 (a) Simulink model for $\dot{y} = 10 \sin t$. (b) Simulink model window.

1. Start Simulink and open a new model window as described previously.
2. Select and place in the new window the Sine Wave block from the Sources category. Double-click on it to open the Block Parameters window, and make sure the Amplitude is set to 1, the Bias to 0, the Frequency to 1, the Phase to 0, and the Sample time to 0. Then click **OK.**
3. Select and place the Gain block from the Math category, double-click on it, and set the Gain Value to 10 in the Block Parameters window. Then click **OK.** Note that the value 10 then appears in the triangle. To make the number more visible, click on the block, and drag one of the corners to expand the block so that all the text is visible.
4. Select and place the Integrator block from the Continuous category, double-click on it to obtain the Block Parameters window, and set the Initial Condition to 0 [this is because $y(0) = 0$]. Then click **OK.**
5. Select and place the Scope block from the Sinks category.
6. Once the blocks have been placed as shown in Figure 5.5.4(a), connect the input port on each block to the outport port on the preceding block. To do this, click on the preceding block, hold down the **Ctrl** key, and then click on the destination block. Simulink will connect the blocks with an arrow pointing at the input port. Your model should now look like that shown in Figure 5.5.4(b).
7. Enter 13 for the Stop time.
8. Run the simulation by clicking on the **Run** icon on the toolbar (this is the black triangle).
9. You will hear a bell sound when the simulation is finished. Then click on the Autoscale button in the Scope display to enable autoscaling. You should see an oscillating curve with an amplitude of 10 and a period of 2π. The independent variable in the Scope block is time t; the input to the block is the dependent variable y. This completes the simulation.

Note that the default solver is `ode45`, the same solver discussed in Section 5.4.

Note that blocks have a Block Parameters window that opens when you double-click on the block. This window contains several items, the number and nature of which depend on the specific type of block. In general, you can use the default values of these

parameters, except where we have explicitly indicated that they should be changed. You can always click on Help within the Block Parameters window to obtain more information.

When you click Apply in the Block Parameters window, any parameter changes immediately take effect and the window remains open. If you click Close, the changes take effect and the window closes.

Note that most blocks have default labels. You can edit text associated with a block by clicking on the text and making the changes.

The Scope block is useful for examining the solution, but if you want to obtain a labeled and printed plot you should use the To Workspace block, which is described in the next example.

Exporting to the MATLAB Workspace

EXAMPLE 5.5.2

■ Problem

We now demonstrate how to export the results of the simulation to the MATLAB workspace, where they can be plotted or analyzed with any of the MATLAB functions.

■ Solution

Modify the Simulink model constructed in Example 5.5.1 as follows. Refer to Figure 5.5.5.

1. Delete the arrow connecting the Scope block by clicking on it and pressing the Delete key. Delete the Scope block in the same way.
2. Select and place the To Workspace block from the Sinks category. Your model should now look like that shown in Figure 5.4.5.
3. Double-click on the To Workspace block. You can specify any variable name you want as the output; the default is `simout`. Change its name to `y`. The output variable `y` will have as many rows as there are simulation time steps, and as many columns as there are inputs to the block. Specify the Save Format as Array. Use the default values for the other parameters (these should be `inf`, 1, and `-1` for Limit data points to last: Decimation, and Sample Time, respectively). Click on **OK.**
4. After running the simulation, you can use the MATLAB plotting commands from the Command window to plot the columns of `y` (or `simout` in general). Simulink puts the time variable `tout` into the MATLAB workspace automatically when using the To Workspace block. To plot $y(t)$, type in the MATLAB Command window:

```
≫plot(tout,y),xlabel('t'),ylabel('y')
```

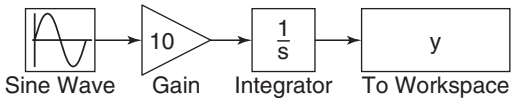


Figure 5.5.5 Simulink model for $\dot{y} = 10 \sin t$ using the To Workspace block.

Simulink Model for $\dot{y} = -10y + f(t)$

EXAMPLE 5.5.3

■ Problem

Construct a Simulink model to solve

$$\dot{y} = -10y + f(t) \qquad y(0) = 1$$

where $f(t) = 2 \sin 4t$, for $0 \leq t \leq 3$.

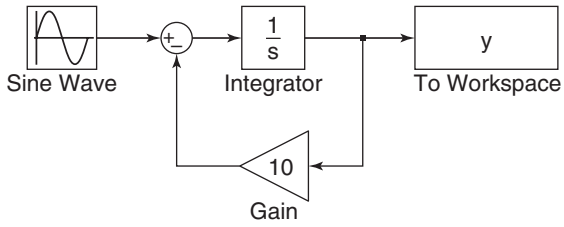


Figure 5.5.6 Simulink model for $\dot{y} = -10y + f(t)$.

■ Solution

To construct the simulation, do the following steps.

1. You can use the model shown in Figure 5.5.4 by rearranging the blocks as shown in Figure 5.5.6. You will need to add a Sum block.
2. Select the Sum block from the Math Operations library and place it as shown in the simulation diagram. Its default setting adds two input signals. To change this, double-click on the block, and in the List of Signs window, type $|+-$. The signs are ordered counterclockwise from the top. The symbol | is a spacer indicating here that the top port is to be empty.
3. To reverse the direction of the gain block, right-click on the block, select **Rotate/Flip** from the pop-up menu, and select **Flip Block.**
4. When you connect the negative input port of the Sum block to the output port of the Gain block, Simulink will attempt to draw the shortest line. To obtain the more standard appearance shown in Figure 5.5.6, first extend the line vertically down from the Sum input port. Release the mouse button and then click on the end of the line and attach it to the Gain block. The result will be a line with a right angle. Do the same to connect the input of the Gain to the arrow connecting the Integrator and the To Workspace block. A small dot appears to indicate that the lines have been successfully connected.
5. Remember to specify Save Format as Array, and to change the name to `y`. Set the Stop time to 3.
6. Run the simulation as before, you can plot the results by typing the following

```
≫plot(tout,y)
```

5.5.2 SIMULATING STATE VARIABLE MODELS

State variable models, unlike transfer function models, can have more than one input and more than one output. Simulink has the State-Space block that represents the linear state variable model $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$, $\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$. The vector $\mathbf{u}$ represents the inputs, and the vector $\mathbf{y}$ represents the outputs. Thus when connecting inputs to the State-Space block, care must be taken to connect them in the proper order. Similar care must be taken when connecting the block's outputs to another block. The following example illustrates how this is done.

EXAMPLE 5.5.4 Simulink Model of a Two-Mass System

■ Problem

The state-variable model of the two-mass system discussed in Example 5.2.3 is

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{B}f(t)$$

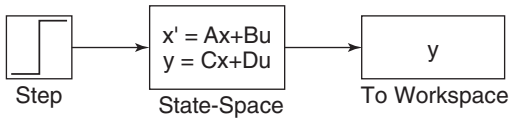


Figure 5.5.7 Simulink model containing the State-Space block and the Step block.

where

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & \frac{4}{5} & -\frac{12}{5} & \frac{8}{5} \\ \frac{4}{3} & -\frac{4}{3} & \frac{8}{3} & -\frac{8}{3} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{3} \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$

Develop a Simulink model to plot the unit-step response of the variables x_1 and x_2 with the initial conditions $x_1(0) = 5$, $\dot{x}_1(0) = -3$, $x_2(0) = 1$, and $\dot{x}_2(0) = 2$.

■ Solution

First select appropriate values for the matrices in the output equation $\mathbf{y} = \mathbf{C}\mathbf{z} + \mathbf{D}f(t)$. Since we want to plot x_1 and x_2, which are z_1 and z_3, we choose **C** and **D** as follows.

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

To create this simulation, obtain a new model window. Then do the following to create the model shown in Figure 5.5.7.

1. Select and place in the new window the Step block from the Sources category. Double-click on it to obtain the Block Parameters window, and set the Step time to 0, the Initial and Final values to 0 and 1, and the Sample time to 0. Click **OK.**
2. Select and place the State-Space block from the CONTINUOUS LIBRARY. Double-click on it, and enter `[0, 0, 1, 0; 0, 0, 0, 1; -1, 4/5, -12/5, 8/5; 4/3, -4/3, 8/3, -8/3]` for **A**, `[0; 0; 0; 1/3]` for **B**, `[1, 0, 0, 0; 0, 1, 0, 0]` for **C**, and `[0; 0]` for **D**. Then enter `[5, 1, -3, 2]` for the initial conditions. Click **OK.** Note that the dimension of the matrix **B** tells Simulink that there is one input. The dimensions of the matrices **C** and **D** tell Simulink that there are two outputs.
3. Select and place the To Workspace block. Change the name to `y` and the Save As format to Array.
4. Once the blocks have been placed, connect the input port on each block to the outport port on the preceding block as shown in the figure.
5. For this application, a Stop time of 20 is satisfactory.

5.6 SIMULINK AND NONLINEAR MODELS

Unlike linear models, closed-form solutions are not available for most nonlinear differential equations, and we must therefore solve such equations numerically. *Piecewise-linear* models are actually nonlinear, although they may appear to be linear. They are composed of linear models that take effect when certain conditions are satisfied. The effect of switching back and forth between these linear models makes the overall model

nonlinear. An example of such a model is a mass attached to a spring and sliding on a horizontal surface with Coulomb friction. The model is

$$m\ddot{x} + kx = f(t) - \mu mg \quad \text{if } \dot{x} \geq 0$$
$$m\ddot{x} + kx = f(t) + \mu mg \quad \text{if } \dot{x} < 0$$

These two linear equations can be expressed as the single, nonlinear equation

$$m\ddot{x} + kx = f(t) - \mu mg \,\text{sign}(\dot{x}) \qquad \text{where} \qquad \text{sign}(\dot{x}) = \begin{cases} +1 & \text{if } \dot{x} \geq 0 \\ -1 & \text{if } \dot{x} < 0 \end{cases}$$

Solution of linear or nonlinear models that contain piecewise-linear functions is very tedious to program. However, Simulink has built-in blocks that represent many of the commonly-found functions such as Coulomb friction. Therefore Simulink is especially useful for such applications.

EXAMPLE 5.6.1 Simulink Model of a Rocket-Propelled Sled

■ Problem

A rocket-propelled sled on a track is represented in Figure 5.6.1 as a mass m with an applied force f that represents the rocket thrust. The rocket thrust initially is horizontal, but the engine accidentally pivots during firing and rotates with an angular acceleration of $\ddot{\theta} = \pi/50$ rad/s. Compute the sled's velocity v for $0 \leq t \leq 10$ if $v(0) = 0$. The rocket thrust is 4000 N and the sled mass is 450 kg.

The sled's equation of motion was derived in Example 5.4.2 and is

$$\dot{v} = \frac{80}{9} \cos\left(\frac{\pi}{100} t^2\right) \tag{1}$$

The solution is formally given by

$$v(t) = \frac{80}{9} \int_0^t \cos\left(\frac{\pi}{100} t^2\right) dt$$

Unfortunately, no closed-form solution is available for the integral, which is called *Fresnel's cosine integral*. The value of the integral has been tabulated numerically, but we will use Simulink to obtain the solution.

a. Create a Simulink model to solve this problem for $0 \leq t \leq 10$ s.
b. Now suppose that the engine angle is limited by a mechanical stop to 60°, which is $60\pi/180$ rad. Create a Simulink model to solve the problem.

■ Solution

a. There are several ways to create the input function $\theta = (\pi/100)t^2$. Here we note that $\ddot{\theta} = \pi/50$ rad/s and that

$$\dot{\theta} = \int_0^t \ddot{\theta}\, dt = \frac{\pi}{50} t$$

and

$$\theta = \int_0^t \dot{\theta}\, dt = \frac{\pi}{100} t^2$$

Thus we can create $\theta(t)$ by integrating the constant $\ddot{\theta} = \pi/50$ twice. The simulation diagram is shown in Figure 5.6.2. This diagram is used to create the corresponding Simulink model shown in Figure 5.6.3.

Figure 5.6.1 A rocket-propelled sled.

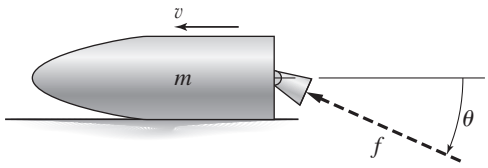


Figure 5.6.2 Simulation model for $v = (80/9)\cos(\pi t^2/100)$.

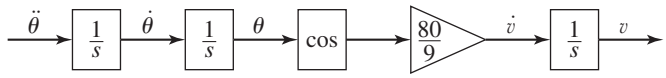


Figure 5.6.3 Simulink model for $\dot{v} = (80/9)\cos(\pi t^2/100)$.

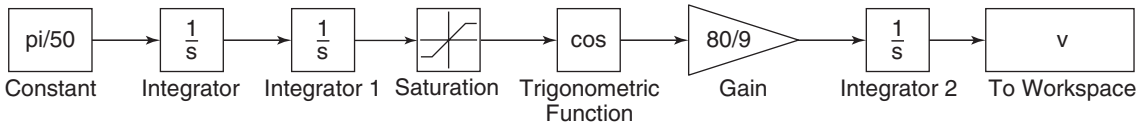
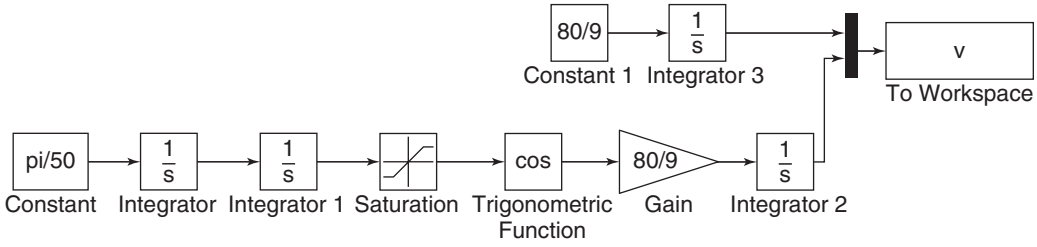


Figure 5.6.4 Simulink model for $\dot{v} = (80/9)\cos(\pi t^2/100)$ with a Saturation block.



There are two new blocks in this model. The Constant block is in the Sources library. After placing it, double click on it and type `pi/50` in its Constant Value window.

The Trigonometric function block is in the Math Operations library. After placing it, double click on it and select `cos` in its Function window.

Set the Stop Time to 10, run the simulation, and examine the results in the Scope.

b. Modify the model in Figure 5.6.3 as follows to obtain the model shown in Figure 5.6.4. We use the Saturation block in the Discontinuities library to limit the range of θ to $60\pi/180$ rad. After placing the block as shown in Figure 5.6.4, double-click on it and type `60*pi/180` in its Upper Limit window. Then type 0 or any negative value in its Lower Limit window.

Select and place the Mux block from the Signal Routing category, double-click on it, and set the Number of inputs to 2. Click **OK.** (The name "Mux" is an abbreviation for "multiplexer," which is an electrical device for transmitting several signals.)

Enter and connect the remaining elements as shown, and run the simulation. The upper Constant block and Integrator block are used to generate the solution when the engine angle is $\theta = 0$, as a check on our results. [The equation of motion for $\theta = 0$ is $\dot{v} = 80/9$, which gives $v(t) = 80t/9$.]

You can plot the results in MATLAB by typing the following:

```
≫ plot(tout,v)
```

The array `v(:,1)` appears at the topmost (first) connection on the Mux; array `v(:,2)` appears at the second (bottom) connection. The resulting plot is shown in Figure 5.6.5.

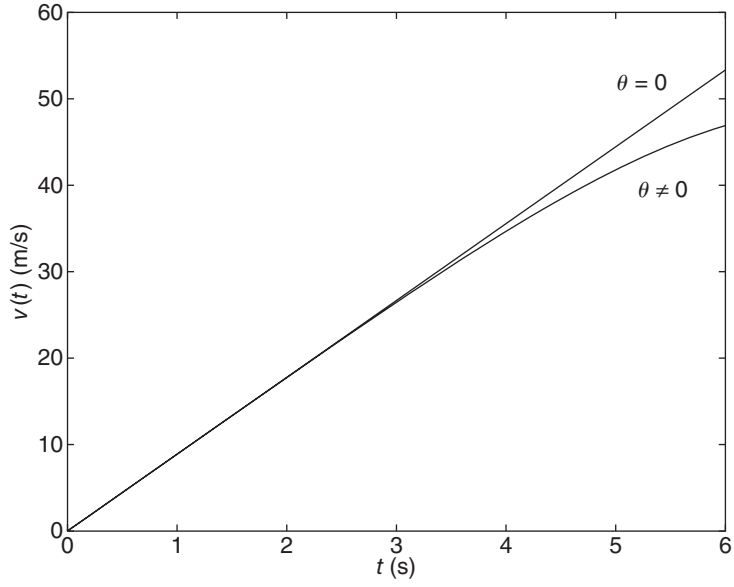


Figure 5.6.5 Speed response of the sled for $\theta = 0$ and $\theta \neq 0$.

5.6.1 SIMULATING TRANSFER FUNCTION MODELS

The equation of motion of a mass-spring-damper system is

$$m\ddot{y} + c\dot{y} + ky = f(t) \tag{5.6.1}$$

Simulink can accept a system description in transfer function form and in state-variable form. If the mass-spring system is subjected to a sinusoidal forcing function $f(t)$, it is easy to use the MATLAB commands presented thus far to solve and plot the response $y(t)$. However, suppose that the force $f(t)$ is created by applying a sinusoidal input voltage to a hydraulic piston that has a *dead-zone* nonlinearity. This means that the piston does not generate a force until the input voltage exceeds a certain magnitude, and thus the system model is piecewise linear. A graph of a particular dead-zone nonlinearity is shown in Figure 5.6.6. When the input (the independent variable on the graph) is

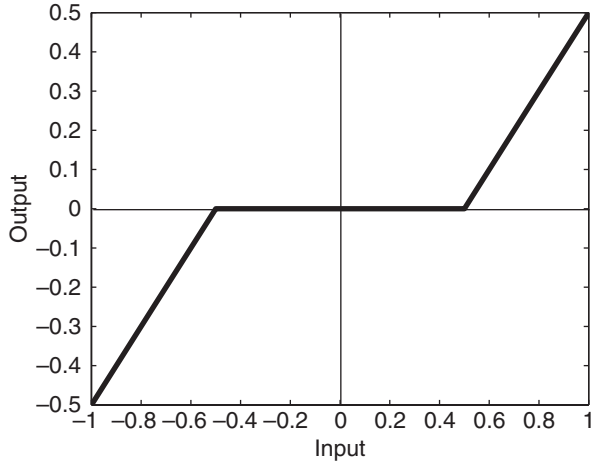


Figure 5.6.6 A dead-zone nonlinearity.

between -0.5 and 0.5, the output is zero. When the input is greater than or equal to the upper limit of 0.5, the output is the input minus the upper limit. When the input is less than or equal to the lower limit of -0.5, the output is the input minus the lower limit. In this example, the dead zone is symmetric about 0, but it need not be in general.

Simulations with dead zone nonlinearities are somewhat tedious to program in MATLAB, but are easily done in Simulink. Example 5.6.2 illustrates how it is done.

A Simulink Model of Response with a Dead Zone

EXAMPLE 5.6.2

■ Problem

Create and run a Simulink simulation of a mass-spring-damper system (5.6.1) using the parameter values $m = 1$, $c = 2$, and $k = 4$. The forcing function is the function $f(t) = \sin 1.4t$. The system has the dead-zone nonlinearity shown in Figure 5.6.6.

■ Solution

To construct the simulation, do the following steps.

1. Start Simulink and open a new model window as described previously.
2. Select and place in the new window the Sine Wave block from the Sources category. Double-click on it, and set the Amplitude to 1, the Bias to 0, the Frequency to 1.4, the Phase to 0, and the Sample time to 0. Click **OK.**
3. Select and place the Dead Zone block from the Discontinuities category, double-click on it, and set the Start of dead zone to -0.5 and the End of dead zone to 0.5. Click **OK.**
4. Select and place the Transfer Fcn block from the Continuous category, double-click on it, and set the Numerator to `[1]` and the Denominator to `[1, 2, 4]`. Click **OK.**
5. Select and place the To Workspace block from the Sinks category. Change the name to `y` and the Save As Format to Array.
6. Once the blocks have been placed, connect the input port on each block to the outport port on the preceding block. Your model should now look like that shown in Figure 5.6.7.
7. Enter 10 for the Stop time.
8. Run the simulation.
9. You should see an oscillating curve if you plot the output.

It is informative to plot both the input and the output of the Transfer Fcn block versus time on the same graph. To do this,

1. Delete the arrow connecting the To Workspace block to the Transfer Fcn block. Do this by clicking on the arrow line and then pressing the **Delete** key.
2. Select and place the Mux block from the Signal Routing category, double-click on it, and set the Number of inputs to 2. Click **Apply,** then **OK.**
3. Connect the top input port of the Mux block to the output port of the Transfer Fcn block. Then use the same technique to connect the bottom input port of the Mux block to the arrow from the outport port of the Dead Zone block. Just remember to start with the input port. Simulink will sense the arrow automatically and make the connection. Your model should now look like that shown in Figure 5.6.8.
4. Set the Stop time to 10, run the simulation as before. Plot the results by typing:

```
≫ plot(tout,y(:,1),tout,y(:,2))
```

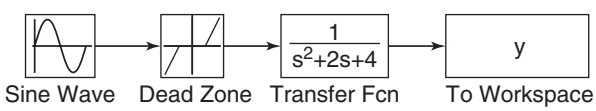


Figure 5.6.7 The Simulink model of dead-zone response.

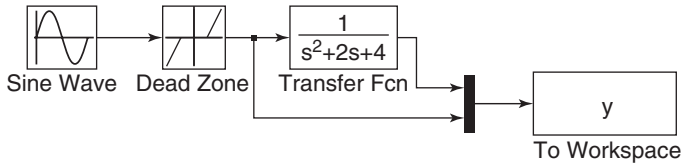


Figure 5.6.8 Modification of the dead-zone model to include a Mux block.

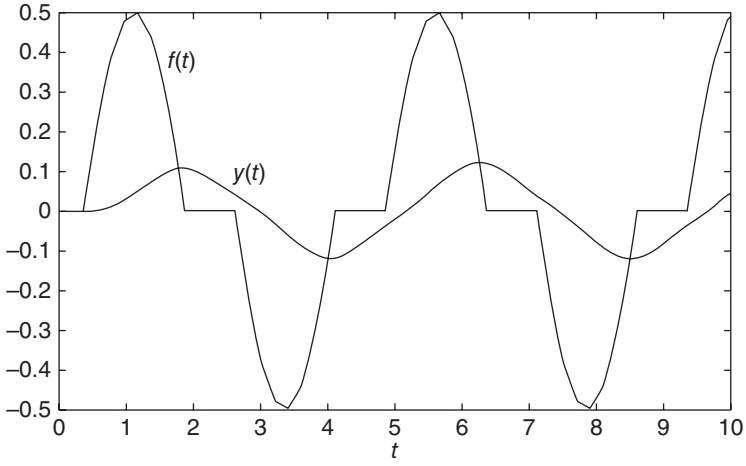


Figure 5.6.9 Plot of the response of the dead-zone model shown in Figure 5.6.8.

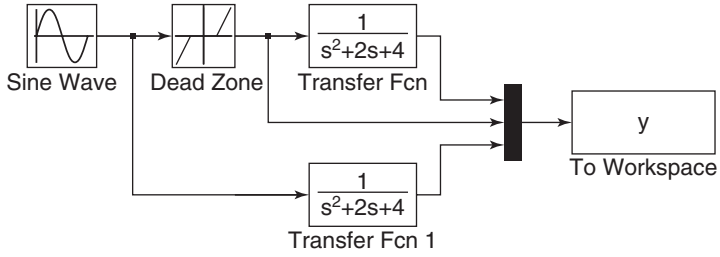


Figure 5.6.10 Modification of the dead-zone model to export variables to the MATLAB workspace.

You should see what is shown in Figure 5.6.9. This plot shows the effect of the dead zone on the sine wave.

Suppose we want to examine the effects of the dead zone by comparing the response of the system with and without a dead zone. We can do this with the model shown in Figure 5.6.10. To create this model,

1. Copy the Transfer Fcn block by right-clicking on it, holding down the mouse button, and dragging the block copy to a new location. Then release the button.
2. Double-click on the Mux block and change the number of its inputs to 3.
3. The output variable `y` will have as many rows as there are simulation time steps, and as many columns as there are inputs to the Mux block.
4. Connect the blocks as shown, and run the simulation.
5. You can use the MATLAB plotting commands from the Command window to plot the columns of `y`; for example, to plot the response of the two systems and the output of the Dead Zone block versus time, type

```
≫plot(tout,y)
```

Nonlinear models cannot be put into transfer function form or the state-variable form $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$. However, they can be solved in Simulink. Example 5.6.3 shows how this can be done.

Simulink Model of a Nonlinear Pendulum

EXAMPLE 5.6.3

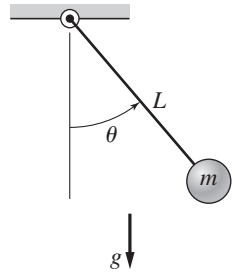
■ Problem

The pendulum shown in Figure 5.6.11 has the following nonlinear equation of motion, if there is viscous friction in the pivot and if there is an applied moment $M(t)$ about the pivot.

$$I\ddot{\theta} + c\dot{\theta} + mgL\sin\theta = M(t) \tag{1}$$

where I is the mass moment of inertia about the pivot. Create a Simulink model for this system for the case where $I = 4$, $mgL = 10$, $c = 0.8$, and $M(t)$ is a square wave with an amplitude of 3 and a frequency of 0.5 Hz. Assume that the initial conditions are $\theta(0) = \pi/4$ rad and $\dot{\theta}(0) = 0$.

Figure 5.6.11 A pendulum.



■ Solution

To simulate this model in Simulink, define a set of variables that lets you rewrite the equation as two first-order equations. Thus let $\omega = \dot{\theta}$. Then the model can be written as

$$\dot{\theta} = \omega$$

$$\dot{\omega} = \frac{1}{I}\left[-c\omega - mgL\sin\theta + M(t)\right] = 0.25\left[-0.8\omega - 10\sin\theta + M(t)\right]$$

Integrate both sides of each equation over time to obtain

$$\theta = \int \omega\, dt$$

$$\omega = 0.25\int \left[-0.8\omega - 10\sin\theta + M(t)\right] dt$$

We will introduce four new blocks to create this simulation. Obtain a new model window and do the following.

1. Select and place in the new window the Integrator block from the Continuous category, and change its label to Integrator 1 as shown in Figure 5.6.12. You can edit text associated with a block by clicking on the text and making the changes. Double-click on

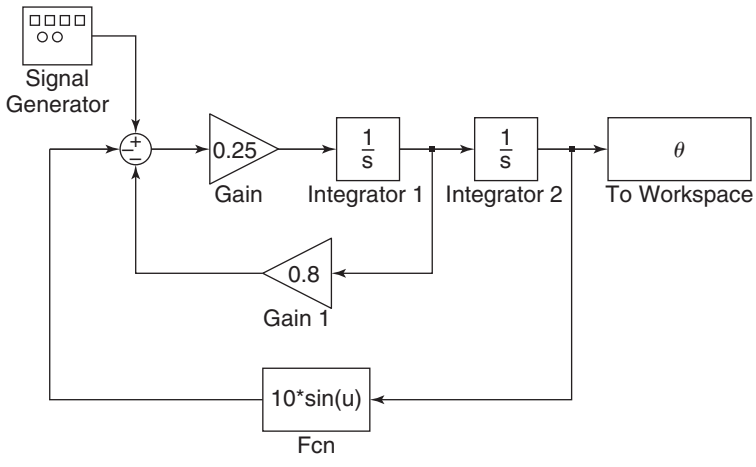


Figure 5.6.12 Simulink model of a nonlinear pendulum.

the block to obtain the Block Parameters window, and set the Initial condition to 0 [this is the initial condition $\dot{\theta}(0) = 0$]. Click **OK.**

2. Copy the Integrator block to the location shown and change its label to Integrator 2. Set its initial condition to $\pi/4$ by typing `pi/4` in the Block Parameters window. This is the initial condition $\theta(0) = \pi/4$.
3. Select and place a Gain block from the Math Operations category, double-click on it, and set the Gain value to 0.25. Click **OK.** Change its label to `1/I`. Then click on the block, and drag one of the corners to expand the box so that all the text is visible.
4. Copy the Gain box, change its label to `c`, and place it as shown in Figure 5.6.12. Double-click on it, and set the Gain value to 0.8. Click **OK.** To flip the box left to right, right-click on it, select **Format,** and select **Flip Block.**
5. Select and place the To Workspace block from the Sinks category. Edit the name and Save As format.
6. For the term $10 \sin\theta$, we cannot use the Trig function block in the Math Operations category without using a separate gain block to multiply the $\sin\theta$ by 10. Instead we will use the Fcn block under the User-Defined Functions category (Fcn stands for function). Select and place this block as shown. Double-click on it, and type `10*sin(u)` in the expression window. This block uses the variable `u` to represent the input to the block. Click **OK.** Then flip the block.
7. Select and place the Sum block from the Math Operations category. Double-click on it, and select round for the Icon shape. In the List of signs window, type $+--$. Click **OK.**
8. Select and place the Signal Generator block from the Sources category. Double-click on it, select square wave for the Wave form, 3 for the Amplitude, and 0.5 for the Frequency, and Hertz for the Units. Click **OK.**
9. Once the blocks have been placed, connect arrows as shown in the figure.
10. Set the Stop time to 10, run the simulation, and plot the results by typing

```
≫ plot(tout,y)
```

5.6.2 VEHICLE SUSPENSION RESPONSE

In this section, we introduce four additional Simulink elements that enable us to model a wide range of nonlinearities and input functions.

As our example, we will use the single-mass suspension model shown in Figure 5.6.13, where the spring and damper elements have the nonlinear models shown in Figures 5.6.14 and 5.6.15. These models represent a hardening spring and a degressive damper. In addition, the damper model is asymmetric. It represents a damper

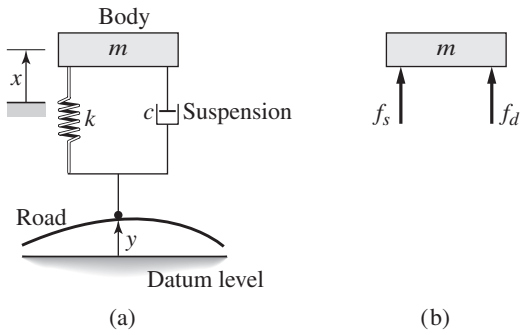


Figure 5.6.13 Single-mass suspension model.

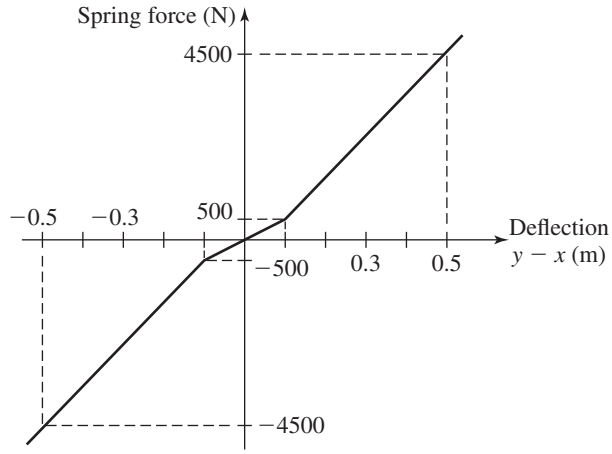


Figure 5.6.14 Hardening spring model.

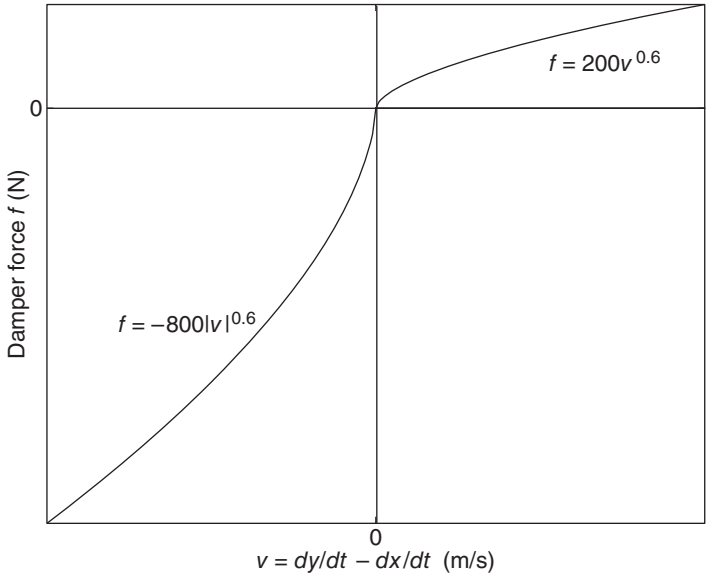


Figure 5.6.15 Degressive damper model.

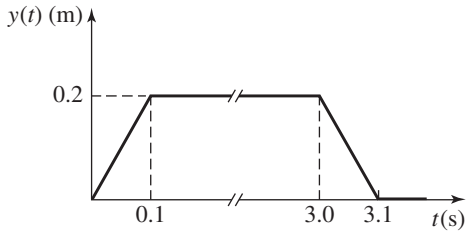


Figure 5.6.16 Bump profile.

whose damping during rebound is higher than during jounce (to minimize the force transmitted to the passenger compartment when the vehicle strikes a bump). The bump is represented by the trapezoidal function $y(t)$ shown in Figure 5.6.16. This function corresponds approximately to a vehicle traveling at 30 mi/h over a 0.2-m-high road surface elevation 48 m long.

The system model from Newton's law is

$$m\ddot{x} = f_s + f_d \tag{5.6.2}$$

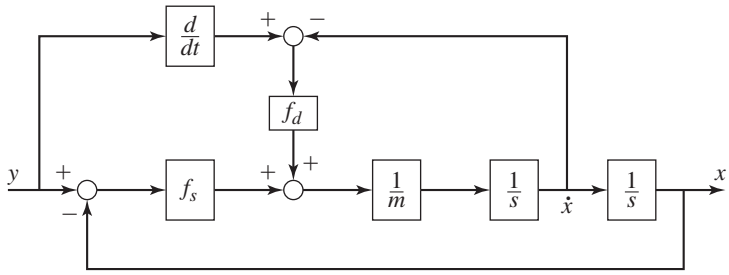
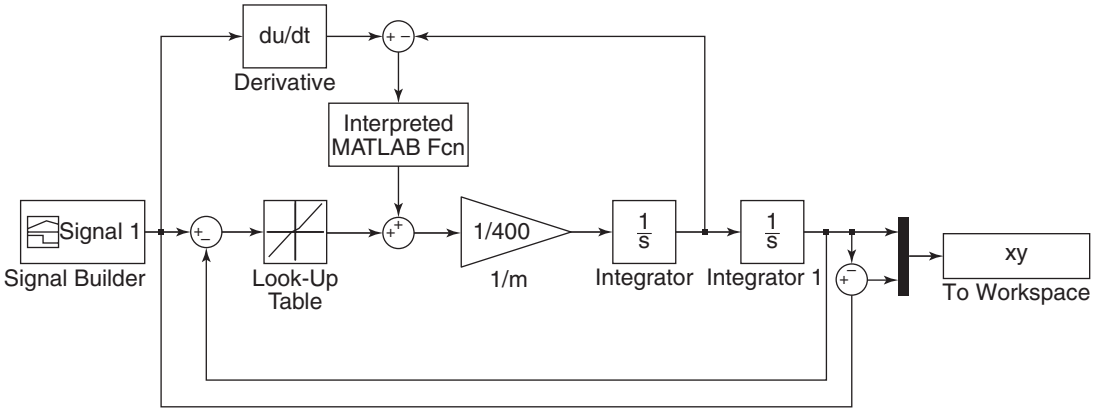


Figure 5.6.17 Simulation diagram for the suspension model.

Figure 5.6.18 Simulink diagram for the suspension model.



where $m = 400$ kg, f_s is the nonlinear spring function shown in Figure 5.6.14, and f_d is the nonlinear damper function shown in Figure 5.6.15. The corresponding simulation diagram is shown in Figure 5.6.17.

This diagram shows that we need to compute $\dot{y}$. Because Simulink uses numerical and not analytical methods, it computes derivatives only approximately, using the Derivative block, which is in the Continuous library. We must keep this in mind when using rapidly changing or discontinuous inputs. The Derivative block has no settings, so merely place it in the Simulink diagram as shown in Figure 5.6.18.

Next place the Signal Builder block, which is in the Sources library, then double-click on it. A plot window appears that enables you to place points to define the input function. Follow the directions in the window to create the function shown in Figure 5.6.16. The spring function f_s is created with the Lookup Table block, which is in the Lookup Tables library. After placing it, double-click on it and enter `[-0.5, -0.1, 0, 0.1, 0.5]` for the `Vector of input values` and `[-4500, -500, 0, 500, 4500]` for the `Vector of output values`. Use the default settings for the remaining parameters.

Place the two integrators as shown, and make sure the initial values are set to 0. Then place the Gain block and set its gain to 1/400. The To Workspace block will enable us to plot $x(t)$ and $y(t) - x(t)$ versus t in the MATLAB Command window.

5.6.3 CREATING FUNCTIONS

For the damper function shown in Figure 5.6.15 we write a user-defined function to describe it. This function is as follows.

```
function f = damper(v)
if v <= 0
   f = -800*(abs(v)).^(0.6);
else
   f = 200*v.^(0.6);
end
```

Create and save this function file. After placing the MATLAB Interpolated Function block, double-click on it and enter its name `damper`. Make sure `Output dimensions` is set to `-1` and the `Output signal type` is set to `auto`.

The Fcn, MATLAB Interpolated Function, Math Function, and S-Function blocks can be used to implement functions, but each has its advantages and limitations. The Fcn block can contain an expression but its output must be a scalar, and it cannot call a function file. The MATLAB Interpolated Function block is slower than the Fcn block, but its output can be an array and it can call a function file. The Math Function block can produce an array output but it is limited to a single MATLAB function and cannot use an expression or call a file. The S-Function block provides more advanced features, such as the ability to use C language code.

The Simulink model when completed should look like Figure 5.6.18. After running it, you can plot the response $x(t)$ in the Command window as follows:

```
≫plot(tout,xy(:,1)),grid,xlabel('t (s)'),ylabel('x (m)')
```

The result is shown in Figure 5.6.19. The maximum overshoot is seen to be $(0.26 - 0.2) = 0.06$ m, but the maximum *under*shoot is seen to be much greater, -0.168 m.

You can plot the difference $y - x$ by typing `plot(tout,xy(:,2))`.

Problems involving piecewise-linear functions such as the saturation block are much easier to solve with Simulink. In later chapters we will discover other blocks and other advantages to using Simulink. There are menu items in the model window we have not discussed. However, the ones we have discussed are the most important ones for getting started. We have introduced just a few of the blocks available within

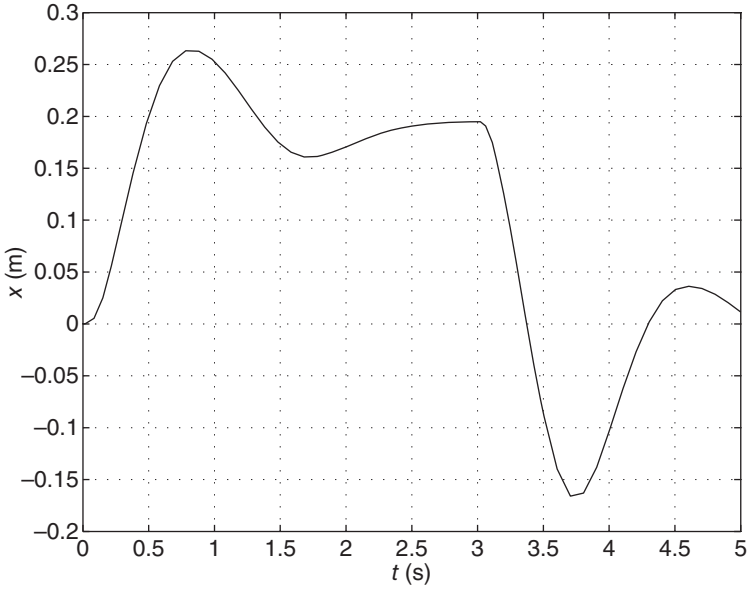


Figure 5.6.19 Response plot for the suspension model.

Simulink and we will introduce more in later chapters. In addition, some blocks have additional properties that we have not mentioned. However, the examples given here will help you get started in exploring the other features of Simulink. Consult the online help for information about these items.

5.7 CHAPTER REVIEW

Part I of the chapter covers block diagrams and state-variable models. The Laplace transform enables an algebraic description of a dynamic system model to be developed. This model form, the transfer function, is the basis of a graphical description of the system, called the block diagram (Section 5.1). Thus a block diagram is a "picture" of the algebraic description of the system that shows how the subsystem elements interact with the other subsystems. Block diagrams can be developed either from the differential equation model or from a schematic diagram that shows how the system components are connected.

The state-variable model form, which can be expressed as a vector-matrix equation, is a concise representation that is useful for analytical purposes and for writing general-purpose computer programs. Section 5.2 shows how to convert models into state-variable form.

In theory it is possible to use the Laplace transform to obtain the closed-form solution of a linear constant-coefficient differential equation if the input function is not too complicated. However, the Laplace transform method cannot be used when the Laplace transform or inverse transform either does not exist or cannot be found easily, as is the case with higher-order models. The reasons include the large amount of algebra required and the need to solve the characteristic equation numerically (closed-form solutions for polynomial roots do not exist for polynomials of order five and higher). Therefore, in Part II of this chapter we introduced several types of numerical methods for solving differential equations.

Section 5.3 focuses on MATLAB functions for solving linear state-variable models. These are the `ss`, `ssdata`, `tfdata`, `step`, `impulse`, `lsim`, `initial`, and `eig` functions. Section 5.4 treats the MATLAB `ode` functions, which are useful for solving both linear and nonlinear equations.

Part III covers Simulink, which provides a graphical user interface for solving differential equations. It includes many program blocks and features that enable you to create simulations that are otherwise difficult to program in MATLAB. Section 5.5 treats applications to linear systems, while Section 5.6 treats nonlinear system applications.

Now that you have finished this chapter, you should be able to

1. Draw a block diagram either from the differential equation model or from a schematic diagram.
2. Obtain the differential equation model from a block diagram.
3. Convert a differential equation model into state-variable form.
4. Express a linear state-variable model in the standard vector-matrix form.
5. Apply the `ss`, `ssdata`, `tfdata`, `eig`, and `initial` functions to analyze linear models.
6. Use the MATLAB `ode` functions to solve linear and nonlinear differential equations.
7. Use Simulink to create simulations of linear and nonlinear models expressed either as differential equations or, if linear, as transfer functions.

REFERENCES

[Palm, 1986] Palm, W. J. III, *Control Systems Engineering,* John Wiley & Sons, New York, 1986.

[Palm, 2009] Palm, W. J. III, *A Concise Introduction to MATLAB,* McGraw-Hill, New York, 2009.

[Palm, 2011] Palm, W. J. III, *Introduction to MATLAB for Engineers,* 3rd ed., McGraw-Hill, New York, 2011.

PROBLEMS

Section 5.1 Transfer Functions and Block Diagram Models

5.1 Obtain the transfer function $X(s)/F(s)$ from the block diagram shown in Figure P5.1.

Figure P5.1

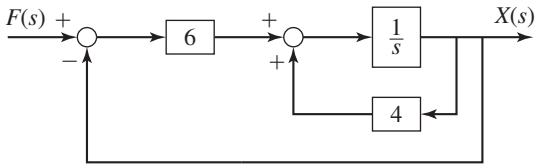
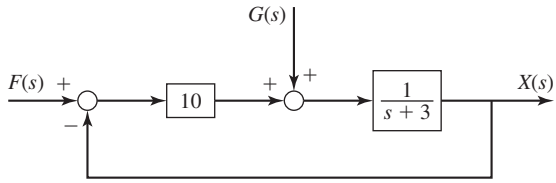


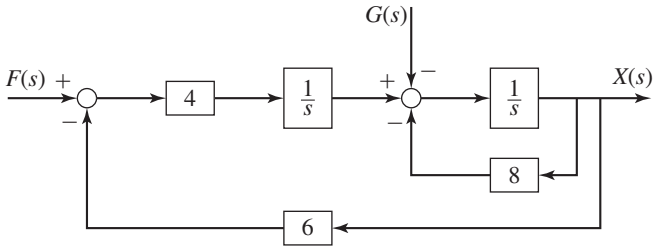
Figure P5.2



5.2 Obtain the transfer function $X(s)/F(s)$ from the block diagram shown in Figure P5.2.

5.3 Obtain the transfer function $X(s)/F(s)$ from the block diagram shown in Figure P5.3.

Figure P5.3



5.4 Draw a block diagram for the following equation. The output is $X(s)$; the inputs are $F(s)$ and $G(s)$.

$$5\ddot{x} + 3\dot{x} + 7x = 10f(t) - 4g(t)$$

5.5 Draw a block diagram for the following model. The output is $X(s)$; the inputs are $F(s)$ and $G(s)$. Indicate the location of $Y(s)$ on the diagram.

$$\dot{x} = y - 5x + g(t) \qquad \dot{y} = 10f(t) - 30x$$

5.6 Referring to Figure P5.6, derive the expressions for the variables $C(s)$, $E(s)$, and $M(s)$ in terms of $R(s)$ and $D(s)$.

5.7 Referring to Figure P5.7, derive the expressions for the variables $C(s)$, $E(s)$, and $M(s)$ in terms of $R(s)$ and $D(s)$.

Figure P5.6

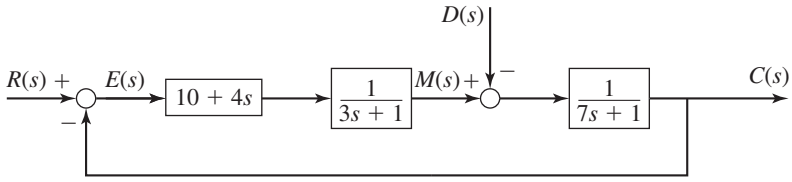
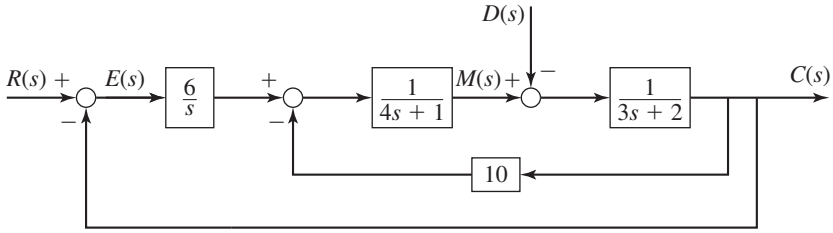


Figure P5.7



5.8 Use the MATLAB `series` and `feedback` functions to obtain the transfer functions $X(s)/F(s)$ and $X(s)/G(s)$ for the block diagram shown in Figure P5.3.

5.9 Use the MATLAB `series` and `feedback` functions to obtain the transfer functions $C(s)/R(s)$ and $C(s)/D(s)$ for the block diagram shown in Figure P5.6.

5.10 Use the MATLAB `series` and `feedback` functions to obtain the transfer functions $C(s)/R(s)$ and $C(s)/D(s)$ for the block diagram shown in Figure P5.7

Section 5.2 State-Variable Models

5.11 Obtain the state model for the reduced-form model $5\ddot{x} + 7\dot{x} + 4x = f(t)$.

5.12 Obtain the state model for the reduced-form model

$$2\frac{d^3y}{dt^3} + 5\frac{d^2y}{dt^2} + 4\frac{dy}{dt} + 7y = f(t)$$

5.13 Obtain the state model for the reduced-form model $2\ddot{x} + 5\dot{x} + 4x = 4y(t)$.

5.14 Obtain the state model for the transfer-function model

$$\frac{Y(s)}{F(s)} = \frac{6}{3s^2 + 6s + 10}$$

5.15 Obtain the state model for the two-mass system whose equations of motion are

$$m_1\ddot{x}_1 + k_1(x_1 - x_2) = f(t)$$

$$m_2\ddot{x}_2 - k_1(x_1 - x_2) + k_2x_2 = 0$$

5.16 Obtain the state model for the two-mass system whose equations of motion for specific values of the spring and damping constants are

$$10\ddot{x}_1 + 8\dot{x}_1 - 5\dot{x}_2 + 40x_1 - 25x_2 = 0$$

$$5\ddot{x}_2 - 25x_1 + 25x_2 - 5\dot{x}_1 + 5\dot{x}_2 = f(t)$$

5.17 Put the following model in standard state-variable form and obtain the expressions for the matrices **A**, **B**, **C**, and **D**. The output is x.

$$2\ddot{x} + 5\dot{x} + 4x = 4y(t)$$

5.18 Given the state-variable model

$$\dot{x}_1 = -5x_1 + 3x_2 + 2u_1$$
$$\dot{x}_2 = -4x_2 + 6u_2$$

and the output equations

$$y_1 = x_1 + 3x_2 + 2u_1$$
$$y_2 = x_2$$

obtain the expressions for the matrices **A**, **B**, **C**, and **D**.

5.19 Given the following state-variable models, obtain the expressions for the matrices **A**, **B**, **C**, and **D** for the given inputs and outputs.

a. The outputs are x_1 and x_2; the input is u.

$$\dot{x}_1 = -5x_1 + 3x_2$$
$$\dot{x}_2 = x_1 - 4x_2 + 5u$$

b. The output is x_1; the inputs are u_1 and u_2.

$$\dot{x}_1 = -5x_1 + 3x_2 + 4u_1$$
$$\dot{x}_2 = x_1 - 4x_2 + 5u_2$$

5.20 Obtain the expressions for the matrices **A**, **B**, **C**, and **D** for the state-variable model you obtained in Problem 5.16. The outputs are x_1 and x_2.

5.21 The transfer function of a certain system is

$$\frac{Y(s)}{F(s)} = \frac{6s + 7}{s + 3}$$

Use two methods to obtain a state-variable model in standard form. For each model, relate the initial value of the state-variable to the given initial value $y(0)$.

5.22 The transfer function of a certain system is

$$\frac{Y(s)}{F(s)} = \frac{s + 2}{s^2 + 4s + 3}$$

Use two methods to obtain a state-variable model in standard form. For each model, relate the initial values of the state variables to the given initial values $y(0)$ and $\dot{y}(0)$.

Section 5.3 State-Variable Methods with MATLAB

5.23 Use MATLAB to create a state-variable model; obtain the expressions for the matrices **A**, **B**, **C**, and **D**, and then find the transfer functions of the following models, for the given inputs and outputs.

a. The outputs are x_1 and x_2; the input is u.

$$\dot{x}_1 = -5x_1 + 3x_2$$
$$\dot{x}_2 = x_1 - 4x_2 + 5u$$

b. The output is x_1; the inputs are u_1 and u_2.

$$\dot{x}_1 = -5x_1 + 3x_2 + 4u_1$$
$$\dot{x}_2 = x_1 - 4x_2 + 5u_2$$

5.24 Use MATLAB to obtain a state model for the following equations; obtain the expressions for the matrices **A**, **B**, **C**, and **D**. In both cases, the input is $f(t)$; the output is y.

a.
$$2\frac{d^3y}{dt^3} + 5\frac{d^2y}{dt^2} + 4\frac{dy}{dt} + 7y = f(t)$$

b.
$$\frac{Y(s)}{F(s)} = \frac{6}{3s^2 + 6s + 10}$$

5.25 Use MATLAB to obtain a state-variable model for the following transfer functions.

a.
$$\frac{Y(s)}{F(s)} = \frac{6s + 7}{s + 3}$$

b.
$$\frac{Y(s)}{F(s)} = \frac{s + 2}{s^2 + 4s + 3}$$

5.26 For the following model the output is x_1 and the input is $f(t)$.

$$\dot{x}_1 = -5x_1 + 3x_2$$
$$\dot{x}_2 = x_1 - 4x_2 + 5f(t)$$

a. Use MATLAB to compute and plot the free response for $x_1(0) = 3$, and $x_2(0) = 5$.

b. Use MATLAB to compute and plot the unit-step response for zero initial conditions.

c. Use MATLAB to compute and plot the response for zero initial conditions with the input $f(t) = 3 \sin 10\pi t$, for $0 \le t \le 2$.

d. Use MATLAB to compute and plot the total response using the initial conditions given in Part (a) and the forcing function given in part (c).

5.27 Given the state-variable model

$$\dot{x}_1 = -5x_1 + 3x_2 + 2u_1$$
$$\dot{x}_2 = -4x_2 + 6u_2$$

and the output equations

$$y_1 = x_1 + 3x_2 + 2u_1$$
$$y_2 = x_2$$

Use MATLAB to find the characteristic polynomial and the characteristic roots.

5.28 The equations of motion for a two-mass, quarter-car model of a suspension system are

$$m_1\ddot{x}_1 = c_1(\dot{x}_2 - \dot{x}_1) + k_1(x_2 - x_1)$$
$$m_2\ddot{x}_2 = -c_1(\dot{x}_2 - \dot{x}_1) - k_1(x_2 - x_1) + k_2(y - x_2)$$

Suppose the coefficient values are: $m_1 = 240$ kg, $m_2 = 36$ kg, $k_1 = 1.6 \times 10^4$ N/m, $k_2 = 1.6 \times 10^5$ N/m, $c_1 = 98$ N · s/m.

a. Use MATLAB to create a state model. The input is $y(t)$; the outputs are x_1 and x_2.

b. Use MATLAB to compute and plot the response of x_1 and x_2 if the input $y(t)$ is a unit impulse and the initial conditions are zero.

c. Use MATLAB to find the characteristic polynomial and the characteristic roots.

d. Use MATLAB to obtain the transfer functions $X_1(s)/Y(s)$ and $X_2(s)/Y(s)$.

5.29 A representation of a car's suspension suitable for modeling the bounce and pitch motions is shown in Figure P5.29, which is a side view of the vehicle's body showing the front and rear suspensions. Assume that the car's motion is constrained to a vertical translation x of the mass center and rotation θ about a single axis which is perpendicular to the page. The body's mass is m and its moment of inertia about the mass center is I_G. As usual, x and θ are the displacements from the equilibrium position corresponding to $y_1 = y_2 = 0$. The displacements $y_1(t)$ and $y_2(t)$ can be found knowing the vehicle's speed and the road surface profile.

a. Assume that x and θ are small, and derive the equations of motion for the bounce motion x and pitch motion θ.

b. For the values $k_1 = 1100$ lb/ft, $k_2 = 1525$ lb/ft, $c_1 = c_2 = 4$ lb-sec/ft, $L_1 = 4.8$ ft, $L_2 = 3.6$ ft, $m = 50$ slugs, and $I_G = 1000$ slug-ft^2, use MATLAB to obtain a state-variable model in standard form.

c. Use MATLAB to obtain and plot the solution for $x(t)$ and $\theta(t)$ when $y_1 = 0$ and y_2 is a unit impulse. The initial conditions are zero.

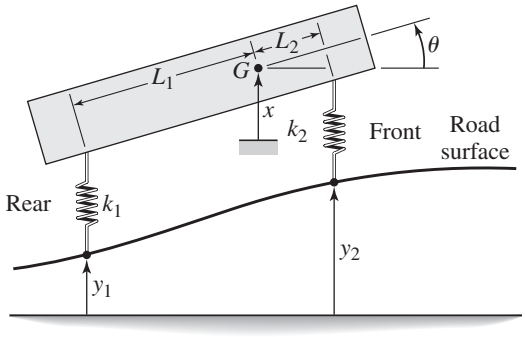


Figure P5.29

Section 5.4 The MATLAB `ode` Functions

5.30 a. Use a MATLAB `ode` function to solve the following equation for $0 \le t \le 12$. Plot the solution.

$$\dot{y} = \cos t \qquad y(0) = 6$$

b. Use the closed-form solution to check the accuracy of the numerical method.

5.31 a. Use a MATLAB `ode` function to solve the following equation for $0 \le t \le 1$. Plot the solution.

$$\dot{y} = 5e^{-4t} \qquad y(0) = 2$$

b. Use the closed-form solution to check the accuracy of the numerical method.

5.32 a. Use a MATLAB `ode` function to solve the following equation for $0 \le t \le 1$. Plot the solution.

$$\dot{y} + 3y = 5e^{4t} \qquad y(0) = 10$$

b. Use the closed-form solution to check the accuracy of the numerical method.

5.33 a. Use a MATLAB `ode` function to solve the following nonlinear equation for $0 \le t \le 4$. Plot the solution.

$$\dot{y} + \sin y = 0 \qquad y(0) = 0.1 \tag{1}$$

b. For small angles, $\sin y \approx y$. Use this fact to obtain a linear equation that approximates equation (1). Use the closed-form solution of this linear equation to check the output of your program.

5.34 Sometimes it is tedious to obtain a solution of a linear equation, especially if all we need is a plot of the solution. In such cases, a numerical method might be preferred. Use a MATLAB `ode` function to solve the following equation for $0 \le t \le 7$. Plot the solution.

$$\dot{y} + 2y = f(t) \qquad y(0) = 2$$

where

$$f(t) = \begin{cases} 3t & \text{for } 0 \le t \le 2 \\ 6 & \text{for } 2 \le t \le 5 \\ -3(t-5)+6 & \text{for } 5 \le t \le 7 \end{cases}$$

5.35 A certain jet-powered ground vehicle is subjected to a nonlinear drag force. Its equation of motion, in British units, is

$$50\dot{v} = f - (20v + 0.05v^2)$$

Use a numerical method to solve for and plot the vehicle's speed as a function of time if the jet's force is constant at 8000 lb and the vehicle starts from rest.

5.36 The following model describes a mass supported by a nonlinear, hardening spring. The units are SI. Use $g = 9.81$ m/s^2.

$$5\ddot{y} = 5g - (900y + 1700y^3)$$

Suppose that $\dot{y}(0) = 0$. Use a numerical method to solve for and plot the solution for two different initial conditions: (1) $y(0) = 0.06$ and (2) $y(0) = 0.1$.

5.37 Van der Pol's equation is a nonlinear model for some oscillatory processes. It is

$$\ddot{y} - b(1 - y^2)\dot{y} + y = 0$$

Use a numerical method to solve for and plot the solution for the following cases:

1. $b = 0.1$, $y(0) = \dot{y}(0) = 1$, $0 \le t \le 25$

2. $b = 0.1$, $y(0) = \dot{y}(0) = 3$, $0 \le t \le 25$

3. $b = 3$, $y(0) = \dot{y}(0) = 1$, $0 \le t \le 25$

5.38 Van der Pol's equation is

$$\ddot{y} - b(1 - y^2)\dot{y} + y = 0$$

This equation can be difficult to solve for large values of the parameter b. Use $b = 1000$ and $0 \le t \le 3000$, with the initial conditions $y(0) = 2$ and $\dot{y}(0) = 0$. Use `ode45` to plot the response.

5.39 The equation of motion for a pendulum whose base is accelerating horizontally with an acceleration $a(t)$ is

$$L\ddot{\theta} + g \sin\theta = a(t) \cos\theta$$

Suppose that $g = 9.81$ m/s^2, $L = 1$ m, and $\dot{\theta}(0) = 0$. Solve for and plot $\theta(t)$ for $0 \le t \le 10$ s for the following three cases.

a. The acceleration is constant: $a = 5$ m/s^2, and $\theta(0) = 0.5$ rad.

b. The acceleration is constant: $a = 5$ m/s^2, and $\theta(0) = 3$ rad.

c. The acceleration is linear with time: $a = 0.5t$ m/s^2, and $\theta(0) = 3$ rad.

5.40 Suppose the spring in Figure P5.40 is nonlinear and is described by the cubic force-displacement relation. The equation of motion is

$$m\ddot{x} = c(\dot{y} - \dot{x}) + k_1(y - x) + k_2(y - x)^3$$

where $m = 100$, $c = 600$, $k_1 = 8000$, and $k_2 = 24000$. Approximate the unit-step input $y(t)$ with $y(t) = 1 - e^{-t/\tau}$, where τ is chosen to be small compared to the period and time constant of the model when the cubic term is neglected. Use MATLAB to plot the forced response $x(t)$.

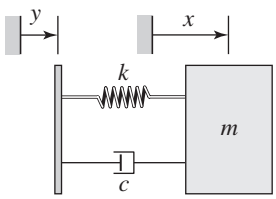


Figure P5.40

Section 5.5 Simulink and Linear Models

5.41 Create a Simulink model to plot the solution of the following equation for $0 \le t \le 6$.

$$10\ddot{y} = 7 \sin 4t + 5 \cos 3t \qquad y(0) = 4 \qquad \dot{y}(0) = 1$$

5.42 A projectile is launched with a velocity of 100 m/s at an angle of 30° above the horizontal. Create a Simulink model to solve the projectile's equations of motion, where x and y are the horizontal and vertical displacements of the projectile.

$$\ddot{x} = 0 \qquad x(0) = 0 \qquad \dot{x}(0) = 100 \cos 30^\circ$$

$$\ddot{y} = -g \qquad y(0) = 0 \qquad \dot{y}(0) = 100 \sin 30^\circ$$

Use the model to plot the projectile's trajectory y versus x for $0 \le t \le 10$ s.

5.43 In Chapter 2 we obtained an approximate solution of the following problem, which has no analytical solution even though it is linear.

$$\dot{x} + x = \tan t \qquad x(0) = 0$$

The approximate solution, which is less accurate for large values of t, is

$$x(t) = \frac{1}{3}t^3 - t^2 + 3t - 3 + 3e^{-t}$$

Create a Simulink model to solve this problem and compare its solution with the approximate solution over the range $0 \le t \le 1$.

5.44 Construct a Simulink model to plot the solution of the following equation for $0 \le t \le 10$.

$$15\dot{x} + 5x = 4u_s(t) - 4u_s(t-2) \qquad x(0) = 2$$

5.45 Use Simulink to solve Problem 5.18 for zero initial conditions, u_1 a unit-step input, and $u_2 = 0$.

5.46 Use Simulink to solve Problem 5.18 for the initial conditions $x_1(0) = 4$, $x_2(0) = 3$, and $u_1 = u_2 = 0$.

5.47 Use Simulink to solve Problem 5.19a for zero initial conditions and $u = 3\sin 2t$.

5.48 Use Simulink to solve Problem 5.26c.

Section 5.6 Simulink and Nonlinear Models

5.49 Use the Transfer Function block to construct a Simulink model to plot the solution of the following equation for $0 \le t \le 4$.

$$2\ddot{x} + 12\dot{x} + 10x^2 = 5u_s(t) - 5u_s(t-2) \qquad x(0) = \dot{x}(0) = 0$$

5.50 Construct a Simulink model to plot the solution of the following equation for $0 \le t \le 4$.

$$2\ddot{x} + 12\dot{x} + 10x^2 = 5\sin 0.8t \qquad x(0) = \dot{x}(0) = 0$$

5.51 Use the Saturation block to create a Simulink model to plot the solution of the following equation for $0 \le t \le 6$.

$$3\dot{y} + y = f(t) \qquad y(0) = 2$$

where

$$f(t) = \begin{cases} 8 & \text{if } 10\sin 3t > 8 \\ -8 & \text{if } 10\sin 3t < -8 \\ 10\sin 3t & \text{otherwise} \end{cases}$$

5.52 Construct a Simulink model of the following problem.

$$5\dot{x} + \sin x = f(t) \qquad x(0) = 0$$

The forcing function is

$$f(t) = \begin{cases} -5 & \text{if } g(t) \le -5 \\ g(t) & \text{if } -5 \le g(t) \le 5 \\ 5 & \text{if } g(t) \ge 5 \end{cases}$$

where $g(t) = 10\sin 4t$.

5.53 Create a Simulink model to plot the solution of the following equation for $0 \le t \le 3$.

$$\dot{x} + 10x^2 = 2\sin 4t \qquad x(0) = 1$$

5.54 Construct a Simulink model of the following problem.

$$10\dot{x} + \sin x = f(t) \qquad x(0) = 0$$

The forcing function is $f(t) = \sin 2t$. The system has the dead-zone nonlinearity shown in Figure 5.6.6.

5.55 The following model describes a mass supported by a nonlinear, hardening spring. The units are SI. Use $g = 9.81$ m/s^2.

$$5\ddot{y} = 5g - (900y + 1700y^3) \qquad y(0) = 0.5 \qquad \dot{y}(0) = 0.$$

Create a Simulink model to plot the solution for $0 \le t \le 2$.

5.56 Consider the system for lifting a mast, discussed in Chapter 3 and shown again in Figure P5.56. The 70-ft-long mast weighs 500 lb. The winch applies a force $f = 380$ lb to the cable. The mast is supported initially at an angle of $\theta = 30^\circ$, and the cable at A is initially horizontal. The equation of motion of the mast is

$$25{,}400\,\ddot{\theta} = -17{,}500\,\cos\theta + \frac{626{,}000}{Q}\sin(1.33 + \theta)$$

where

$$Q = \sqrt{2020 + 1650\,\cos(1.33 + \theta)}$$

Create and run a Simulink model to solve for and plot $\theta(t)$ for $\theta(t) \le \pi/2$ rad.

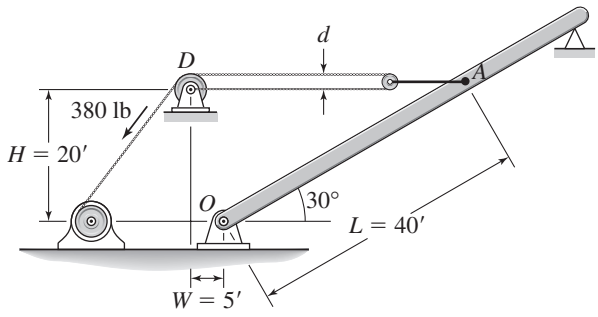


Figure P5.56

5.57 A certain mass, $m = 2$ kg, moves on a surface inclined at an angle $\phi = 30^\circ$ above the horizontal. Its initial velocity is $v(0) = 3$ m/s up the incline. An external force of $f_1 = 5$ N acts on it parallel to and up the incline. The coefficient of dynamic friction is $\mu_d = 0.5$. Use the Coulomb Friction block or the Sign block and create a Simulink model to solve for the velocity of the mass until the mass comes to rest. Use the model to determine the time at which the mass comes to rest.

5.58 If a mass-spring system has Coulomb friction on the horizontal surface rather than viscous friction, its equation of motion is

$$m\ddot{y} = -ky + f(t) - \mu mg \quad \text{if } \dot{y} \ge 0$$

$$m\ddot{y} = -ky + f(t) + \mu mg \quad \text{if } \dot{y} < 0$$

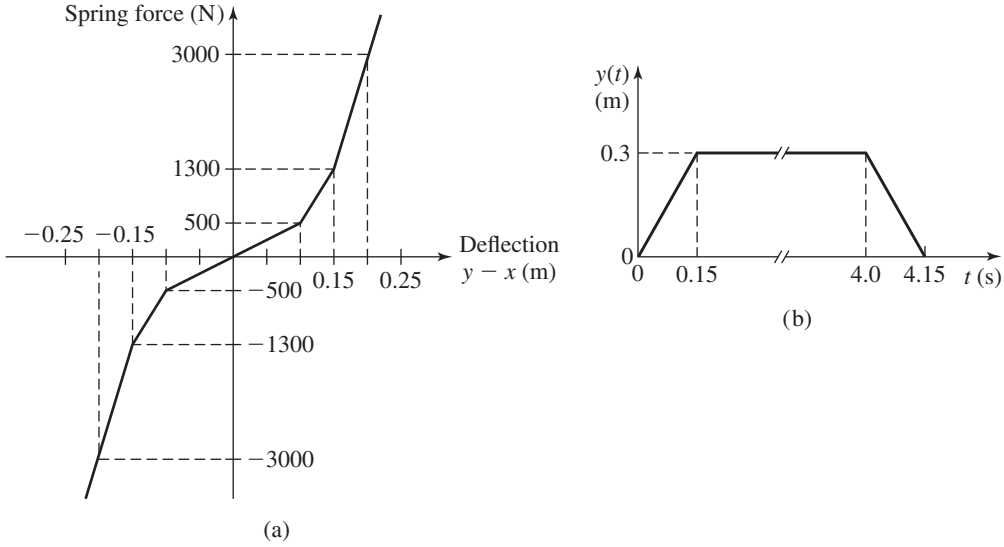
where μ is the coefficient of friction. Develop a Simulink model for the case where $m = 1$ kg, $k = 5$ N/m, $\mu = 0.4$, and $g = 9.8$ m/s^2. Run the simulation for two cases: (a) the applied force $f(t)$ is a step function with a magnitude of 10 N, and (b) the applied force is sinusoidal: $f(t) = 10 \sin 2.5t$.

5.59 Redo the Simulink suspension model developed in subsection 5.6.2, using the spring relation and input function shown in Figure P5.59, and the following damper relation.

$$f_d(v) = \begin{cases} -500|v|^{1.2} & v \le 0 \\ 50v^{1.2} & v > 0 \end{cases}$$

Use the simulation to plot the response. Evaluate the overshoot and undershoot.

Figure P5.59



5.60 Consider the system shown in Figure P5.60. The equations of motion are

$$m_1\ddot{x}_1 + (c_1 + c_2)\dot{x}_1 + (k_1 + k_2)x_1 - c_2\dot{x}_2 - k_2x_2 = 0$$

$$m_2\ddot{x}_2 + c_2\dot{x}_2 + k_2x_2 - c_2\dot{x}_1 - k_2x_1 = f(t)$$

Suppose that $m_1 = m_2 = 1$, $c_1 = 3$, $c_2 = 1$, $k_1 = 1$, and $k_2 = 4$.

a. Develop a Simulink simulation of this system. In doing this, consider whether to use a state-variable representation or a transfer function representation of the model.

b. Use the Simulink program to plot the response $x_1(t)$ for the following input. The initial conditions are zero.

$$f(t) = \begin{cases} t & 0 \le t \le 1 \\ 2 - t & 1 < t < 2 \\ 0 & t \ge 2 \end{cases}$$

Figure P5.60