CHAPTER

State-Variable Models 3

在第 2 章中，我们介绍了两种描述线性时不变模拟系统的模型：常系数线性微分方程模型以及传递函数模型。传递函数模型可以通过拉普拉斯变换从微分方程模型中获得。相应的，微分方程模型可以通过拉普拉斯反变换从传递函数模型中获得。本章将介绍一种新的模型：状态空间模型。这是一种拥有特殊形态微分方程的模型。状态空间模型由 n 个一阶耦合微分方程组成。使用状态空间模型有以下优点：

（1）对于高阶系统，使用状态空间模型更易于进行计算机辅助分析和设计。相反由于数值问题，传递函数的方法总会失效。

（2）由于在状态空间控制器使用了更多的系统反馈信息，因此，相比于传递函数控制方法，基于状态空间模型的系统的控制将变得更加彻底和精准。

（3）最优控制理论基本都是基于状态空间控制模型的。

（4）即便我们不能够按照最优原则进行状态空间变量的设计，我们依然能够得到近似“最优”的系统响应。因为我们可以尝试用传统的设计方法近似得到这个“最优”的响应。

（5）状态空间模型更适合数字模拟仿真。

In Chapter 2, two models of linear time-invariant (LTI) analog systems were presented: linear differential equations with constant coefficients and transfer functions. By use of the Laplace transform, the transfer function can be derived from the differential equations, and a differential equation model can be derived from the transfer function using the inverse Laplace transform. In this chapter, we consider a third type of model: the *state-variable model* [1,2]. This model is a differential equation model, but the equations are always written in a specific format. The state-variable model, or state-space model, is expressed as n first-order coupled differential equations. These equations preserve the system's input–output relationship (that of the transfer function); in addition, an internal model of the system is given. Some additional reasons for developing the state model are as follows:

1. Computer-aided analysis and design of state models are performed more easily on digital computers for higher-order systems, while the transfer–function approach tends to fail for these systems because of numerical problems.
2. In state-variable design procedures, we feedback more information (internal variables)

about the plant; hence, we can achieve a more complete control of the system than is possible with the transfer–function approach.

3. Design procedures that result in the "best" control system are almost based on state-variable models. By "best" we mean that the system has been designed in such a way as to minimize (or maximize) a mathematical function that expresses the design criteria.
4. Even if we do not implement the state-variable design (some are not practical), we can produce a "best" system response. Then we attempt to approximate this "best" system response using classical design procedures (covered in Chapters 7 and 9).
5. State-variable models are generally required for digital simulation (digital computer solution of differential equations).

3.1 STATE-VARIABLE MODELING

We begin this section by giving an example to illustrate state-variable modeling. The system model used to illustrate state variables, a linear mechanical translational system, is given in Figure 3.1. This same model was utilized in Section 2.5, except here the displacement is denoted as $y(t)$. The differential equation describing this system is given by

$$M\frac{d^2y(t)}{dt^2} = f(t) - B\frac{dy(t)}{dt} - Ky(t) \tag{3-1}$$

and the transfer function is given by

$$G(s) = \frac{Y(s)}{F(s)} = \frac{1}{Ms^2 + Bs + K} \tag{3-2}$$

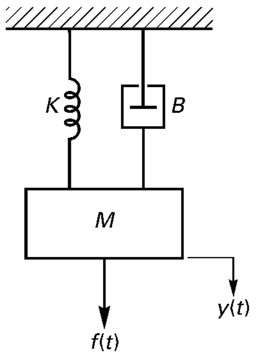


FIGURE 3.1
Mechanical translational system.

This equation gives a description of the position $y(t)$ as a function of the force $f(t)$. Suppose that we also want information about the velocity. Using the state-variable approach, we define the two state variables $x_1(t)$ and $x_2(t)$ as

$$x_1(t) = y(t) \tag{3-3}$$

and

$$x_2(t) = \frac{dy(t)}{dt} = \frac{dx_1(t)}{dt} = \dot{x}_1(t) \tag{3-4}$$

Thus $x_1(t)$ is the position of the mass and $x_2(t)$ is its velocity. Then, from (3-1), (3-3), and (3-4), we may write

$$\frac{d^2y(t)}{dt^2} = \frac{dx_2(t)}{dt} = \dot{x}_2(t) = -\left(\frac{B}{M}\right)x_2(t) - \left(\frac{K}{M}\right)x_1(t) + \left(\frac{1}{M}\right)f(t) \tag{3-5}$$

where the overdot denotes the derivative with respect to time. The state-variable model is contained in (3-3), (3-4), and (3-5). However, the model is usually written in a specific format, which is given by rearranging the preceding equations as

$$\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= -\left(\frac{K}{M}\right)x_1(t) - \left(\frac{B}{M}\right)x_2(t) + \left(\frac{1}{M}\right)f(t)\\ y(t) &= x_1(t)\end{aligned}$$

Usually, state equations are written in a vector-matrix format, since this allows the equations to be manipulated much more easily (see Appendix A for a review of matrices). In this format the preceding equations become

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

$$y(t) = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

We will now define the state of a system.

Definition The state of a system at any time t_0 is the amount of information at t_0 that, together with all inputs for $t \geq t_0$, uniquely determines the behavior of the system for all $t \geq t_0$.

It will be shown that the state vector of the standard form of the state-variable equations, to be developed next, satisfies this definition.

The standard form of the state equations of a LTI analog system is given by

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)\end{aligned} \tag{3-6}$$

where the vector $\dot{\mathbf{x}}(t)$ is the time derivative of the vector $\mathbf{x}(t)$. In these equations,

$\mathbf{x}(t)$ = state vector = $(n \times 1)$ vector of the states of an nth-order system
$\mathbf{A}$ = $(n \times n)$ system matrix
$\mathbf{B}$ = $(n \times r)$ input matrix
$\mathbf{u}(t)$ = input vector = $(r \times 1)$ vector composed of the system input functions
$\mathbf{y}(t)$ = output vector = $(p \times 1)$ vector composed of the defined outputs
$\mathbf{C}$ = $(p \times n)$ output matrix
$\mathbf{D}$ = $(p \times r)$ matrix to represent direct coupling between input and output

The signals in expanded form are given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \vdots\\ \dot{x}_n(t)\end{bmatrix}\quad \mathbf{x}(t) = \begin{bmatrix}x_1(t)\\ x_2(t)\\ \vdots\\ x_n(t)\end{bmatrix}\quad \mathbf{u}(t) = \begin{bmatrix}u_1(t)\\ u_2(t)\\ \vdots\\ u_r(t)\end{bmatrix}\quad \mathbf{y}(t) = \begin{bmatrix}y_1(t)\\ y_2(t)\\ \vdots\\ y_p(t)\end{bmatrix}$$

We refer to the two matrix equations of (3-6) as the *state-variable equations* of the system. The first equation is called the *state equation,* and the second one is called the *output equation.* The first equation, the state equation, is a first-order matrix differential equation, and the state vector, $\mathbf{x}(t)$, is its solution. Given knowledge of $\mathbf{x}(t)$ and the input vector $\mathbf{u}(t)$, the output equation yields the output $\mathbf{y}(t)$. Usually the matrix $\mathbf{D}$ is zero, since in physical systems, dynamics appear in all paths between the inputs and the outputs. A nonzero value of $\mathbf{D}$ indicates at least one direct path between the inputs and the outputs, in which the path transfer function can be modeled as a pure gain.

In the equation for the state variables $\mathbf{x}(t)$ in (3-6), only the first derivatives of the state variables may appear on the left side of the equation, and no derivatives may appear on the right side. No derivatives may appear in the output equation. Valid equations that model a system may be written without following these rules; however, those equations will not be in the standard format.

The general form of the state equations just given allows for more than one input and output; these systems are called *multivariable systems.* For the case of one input, the matrix $\mathbf{B}$ is a column vector, and the vector $\mathbf{u}(t)$ is a scalar. For the case of one output, the vector $\mathbf{y}(t)$ is a scalar, and the matrix $\mathbf{C}$ is a row vector. In this book, we do not differentiate notationally between a column vector and a row vector. The use of the vector implies the type. An example is now given to illustrate a multivariable system.

Example 3.1

Consider the system described by the coupled differential equations

$$\ddot{y}_1 + k_1\dot{y}_1 + k_2 y_1 = u_1 + k_3 u_2$$
$$\dot{y}_2 + k_4 y_2 + k_5\dot{y}_1 = k_6 u_1$$

where u_1 and u_2 are inputs, y_1 and y_2 are outputs, and $k_i, i = 1, \ldots, 6$ are system parameters. The notational dependence of the variables on time has been omitted for convenience. We may define the states as the outputs and, where necessary, the derivatives of the outputs.

$$x_1 = y_1 \qquad x_2 = \dot{y}_1 = \dot{x}_1 \qquad x_3 = y_2$$

From the system differential equations, we write

$$\dot{x}_2 = -k_2 x_1 - k_1 x_2 + u_1 + k_3 u_2$$
$$\dot{x}_3 = -k_5 x_2 - k_4 x_3 + k_6 u_1$$

We rewrite the differential equations in the following order:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -k_2 x_1 - k_1 x_2 + u_1 + k_3 u_2$$
$$\dot{x}_3 = -k_5 x_2 - k_4 x_3 + k_6 u_1$$

with the output equations

$$y_1 = x_1$$
$$y_2 = x_3$$

These equations may be written in matrix form as

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ -k_2 & -k_1 & 0 \\ 0 & -k_5 & -k_4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1 & k_3 \\ k_6 & 0 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}$$

In the preceding example, suppose that the equations model a mechanical translational system. Suppose further that both y_1 and y_2 are displacements in the system. Then x_2, which is $\dot{y}_1$, is a velocity in the system. Hence, all the state variables represent physical variables within the system; x_1 and x_3 are displacements and x_2 is a velocity. Generally, we prefer that the state variables be physically identifiable variables, but this is not necessary; this topic is developed further in Section 3.5. Next, we consider a method for obtaining the state model directly from the system transfer function. In general, this method does not result in the state variables being physical variables.

3.2 SIMULATION DIAGRAMS

In Section 3.1, we presented an example of finding the state model of a system directly from the system differential equations. The procedure in that example is very useful and is employed in many practical situations. However, if a system-identification technique (introduced in Section 2.13) is used to obtain the system model, only a transfer function may be available to describe the system. For this and other reasons, it is advantageous to have a method available to obtain a state model directly from a transfer function.

The method considered here is based on the use of simulation diagrams. A *simulation diagram* is a certain type of either a block diagram or a flow graph that is constructed to have a specified transfer function or to model a specified set of differential equations. Given the transfer function, the differential equations, or the state equations of a system, we can construct a simulation diagram of the system. The simulation diagram is aptly named, since it is useful in constructing either digital computer or analog computer simulations of a system.

The basic element of the simulation diagram is the integrator. Figure 3.2 shows the block diagram of an integrating device. In this figure

$$y(t) = \int x(t)\,dt$$

and the Laplace transform of this equation yields

$$Y(s) = \frac{1}{s} X(s) \tag{3-7}$$

Hence, the transfer function of a device that integrates a signal is $1/s$, as shown in Figure 3.2. We are interested only in the transfer function; thus we have ignored the initial condition on $y(t)$.

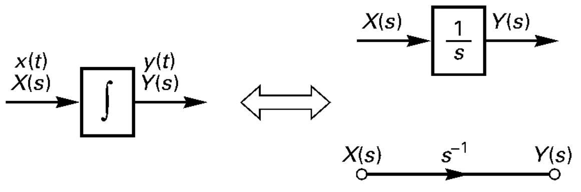


FIGURE 3.2
Integrating device.

If the output of an integrator is labeled as $y(t)$, the input to the integrator must be dy/dt. We use this characteristic of the integrator to aid us in constructing simulation diagrams. For example, two integrators are cascaded in Figure 3.3(a). If the output of the second integrator is $y(t)$, the input to this integrator must be $\dot{y}(t)$. In a like manner, the input to the first integrator must be $\ddot{y}(t)$, where $\ddot{y}(t) = d^2y(t)/dt^2$.

We can use these two integrators to construct a simulation diagram of the mechanical system of Figure 3.1. The input to the cascaded integrators in Figure 3.3(a) is $\ddot{y}(t)$, and the equation that $\ddot{y}(t)$ must satisfy for the mechanical system is obtained from (3-1).

$$\ddot{y}(t) = -\frac{B}{M}\dot{y}(t) - \frac{K}{M}y(t) + \frac{1}{M}f(t) \tag{3-8}$$

Hence a summing junction and appropriate gains can be added to the block diagram of Figure 3.3(a) to satisfy this equation, as shown in Figure 3.3(b). Figure 3.3(b) is called a *simulation diagram* of the mechanical system.

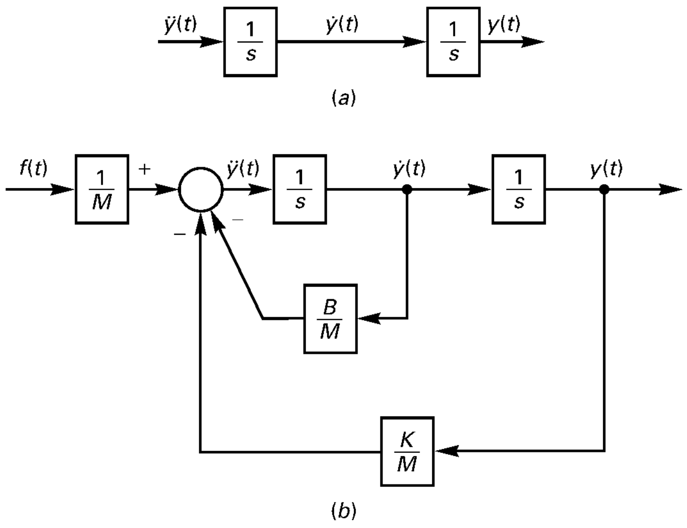


FIGURE 3.3
Simulation diagrams.

In the approach just described, a simulation diagram of integrators, gains, and summing junctions was constructed that satisfied a given differential equation. This is the general approach for the construction of simulation diagrams. Note that, by Mason's gain formula, the transfer function of the simulation diagram in Figure 3.3(b) is given by

$$G(s) = \frac{(1/M)s^{-2}}{1 + (B/M)s^{-1} + (K/M)s^{-2}} = \frac{1/M}{s^2 + (B/M)s + (K/M)} \tag{3-9}$$

This transfer function checks that derived in (3-2).

Note that we can reverse the last procedure. We can begin with the given transfer function and, through the reverse of the given steps, construct the simulation diagram. If the simulation diagram is constructed from the system differential equations, the simulation diagram will usually be unique. However, if the transfer function is used to construct the simulation diagram, the simulation diagram can be of many different forms, that is, the simulation diagram is not unique. Two common, useful forms of the simulation diagram are now presented.

The two different simulation diagrams to be given realize the general transfer function

$$\begin{aligned} G(s) &= \frac{b_{n-1}s^{-1} + b_{n-2}s^{-2} + \cdots + b_0 s^{-n}}{1 + a_{n-1}s^{-1} + a_{n-2}s^{-2} + \cdots + a_0 s^{-n}} \\ &= \frac{b_{n-1}s^{n-1} + b_{n-2}s^{n-2} + \cdots + b_0}{s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \cdots + a_0} \end{aligned} \tag{3-10}$$

The first simulation diagram, called the *control canonical form,* is given in Figure 3.4 for the case $n = 3$, that is, for

$$G(s) = \frac{b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0} \tag{3-11}$$

The second simulation diagram, called the *observer canonical form,* is given in Figure 3.5 for (3-11). The reasons for these particular names are from modern control theory and will become evident later. It is seen from Mason's gain formula that the two simulation diagrams have the transfer function of (3-11). It is noted that the state vector $\mathbf{x}(t)$ in Figure 3.4 is *not* equal to state vector $\mathbf{x}(t)$ in Figure 3.5.

Note that in the transfer function of (3-10), the order of the numerator must be at least one less than the order of the denominator. Of course, any coefficient a_i or b_i may be zero. The general transfer function of a physical system is always assumed to be of this form; the reason for this assumption will become evident when the frequency response of systems is covered. However, simulation diagrams can be constructed for transfer functions for which the order of the numerator is equal to that of the denominator.

Once a simulation diagram of a transfer function is constructed, a state model of the system is easily obtained. The procedure to do this has two steps:

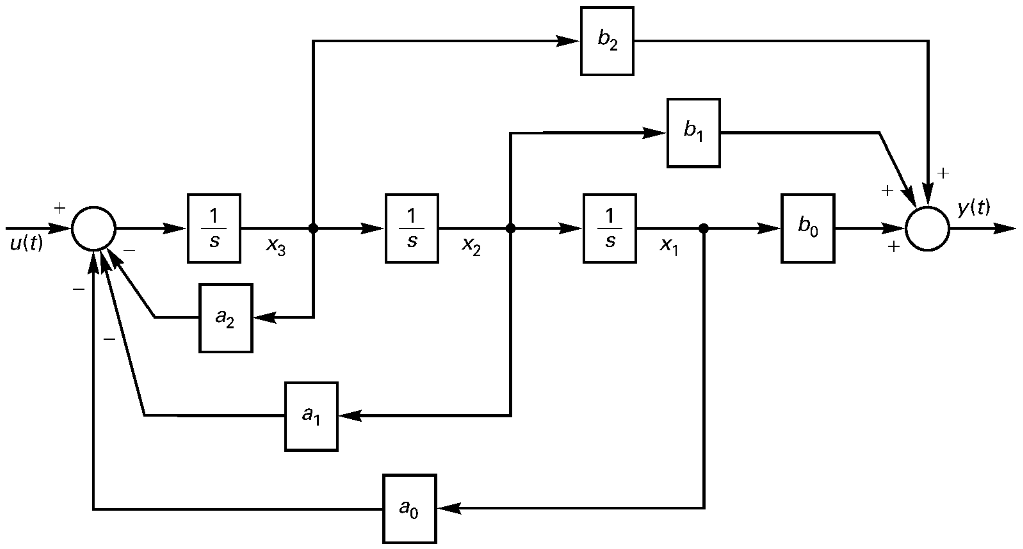


FIGURE 3.4
Control canonical form.

1. Assign a state variable to the output of each integrator.
2. Write an equation for the input of each integrator and an equation for each system output. These equations are written as functions of the integrator outputs and the system inputs.

This procedure yields the following state equations for the control canonical form of Figure 3.4, where the states are identified in the figure. The state equations are, from Figure 3.4,

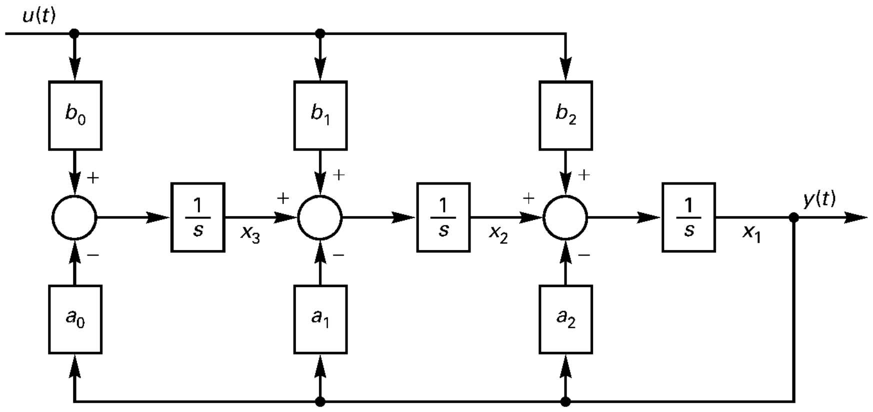


FIGURE 3.5
Observer canonical form.

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{3-12}$$

$$y = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix} \mathbf{x}$$

These equations are easily extended to the nth-order system (see Problem 3.5). Suppose, in (3.12), $y(t)$ is position for a position control system. Then, in general, $x_1(t)$ is not position, $x_2(t)$ is not velocity, and so on. For the observer canonical form of Figure 3.5, the state equations are written as

$$\dot{\mathbf{x}} = \begin{bmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} b_2 \\ b_1 \\ b_0 \end{bmatrix} u \tag{3-13}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}$$

Suppose again that $y(t)$ is position for a position control system. Then $x_1(t)$ is also position, but $x_2(t)$ in general is not velocity, and so on. Note that the coefficients that appear in the transfer function also appear directly in the matrices of the state equations. In fact, it is evident that the state equations for these two standard forms can be written directly from the transfer function, without drawing simulation diagrams.

To illustrate further the procedure for writing state equations from the simulation diagram, consider the following example.

Example 3.2

Consider again the mechanical system of Figures 3.1 and 3.3(b). In the simulation diagram of Figure 3.3(b), which is repeated in Figure 3.6, a state has been assigned to each integrator output. With this assignment, the input to the rightmost integrator is $\dot{x}_1$; thus the equation for this integrator input is

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

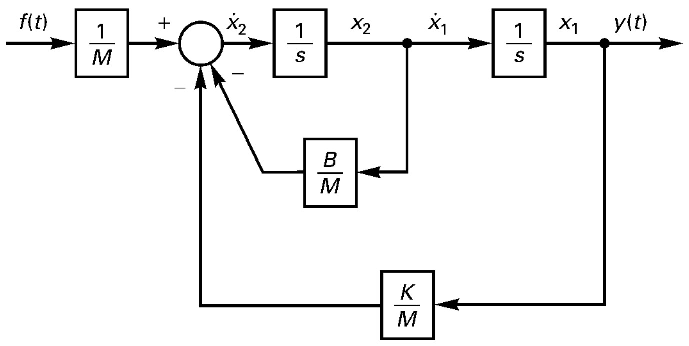


FIGURE 3.6
Simulation diagram.

For the other integrator the input is $\dot{x}_2$. Thus the equation for this integrator input is

$$\dot{x}_2 = -\frac{K}{M}x_1 - \frac{B}{M}x_2 + \frac{1}{M}f$$

and the system output equation is

$$y = x_1$$

These equations may be written in the standard state-variable matrix format as

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{B}{M} \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} f(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t)$$

These equations are the same as those developed in the first section of this chapter, and the state model is seen to be in the control canonical form.

Example 3.3

In this example, we develop a state-variable model of a dc motor used in rolling steel. Figure 3.7(a) illustrates steel rolling. The purpose of the rolling is to reduce the thickness of the steel, and $w_3 < w_2 < w_1$. Two sets of rollers are shown in Figure 3.7(a); in general, several more sets will appear in a steel-rolling mill. As the steel is rolled, the thickness decreases, with a corresponding increase in the length of the steel sheet. In Figure 3.7(a) the second set of rollers must rotate at a greater speed than the first set. The speed of the second set of rollers is determined by the speed of the first set and the reduction in the thickness of the steel in the first set. Hence, the speed of the second set must be controlled accurately. A block diagram of a motor speed-control system is given in Figure 3.7(b), and a dc motor is depicted in Figure 3.7(c). From Section 2.7, the equations of the motor are

[eq. 2–41] $$e_m(t) = K_m \frac{d\theta(t)}{dt}$$

[eq. 2–43] $$e_a(t) = L_m \frac{di_a(t)}{dt} + R_m i_a(t) + e_m(t)$$

[eq. 2–45] $$\tau(t) = K_\tau i_a(t)$$

[eq. 2–47] $$J\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta(t)}{dt} = \tau(t)$$

In these equations the variables are

$e_a(t)$ = armature voltage (the input)
$e_m(t)$ = back-EMF
$i_a(t)$ = armature current
$\tau(t)$ = developed torque
$\theta(t)$ = motor shaft angle
$d\theta(t)/dt = \omega(t)$ = shaft speed (the output)

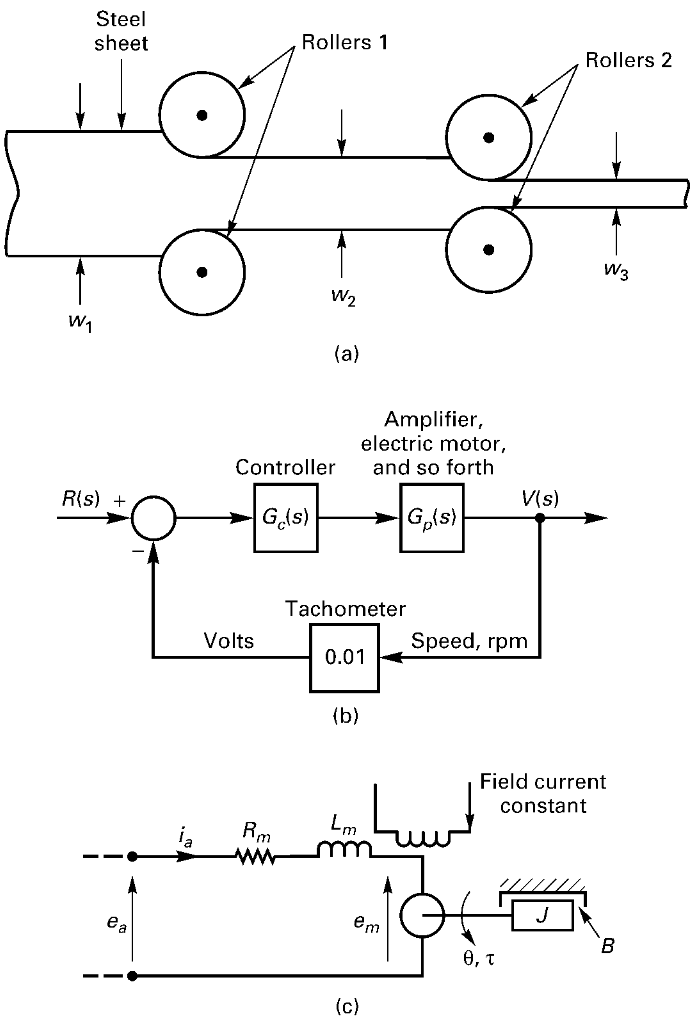


FIGURE 3.7
(a) Steel rolling; (b) dc motor speed control for one set of rollers; (c) model for the motor.

In a speed-control system, the motor speed $\omega(t)$ is controlled by varying the armature voltage $e_a(t)$. Hence $e_a(t)$ is the input variable and $\omega(t)$ is the output variable.

We choose as the state variables $x_1(t) = \omega(t) = d\theta(t)/dt$ and $x_2(t) = i_a(t)$. Note that both state variables can be measured easily, and this is one reason for the choice. The measurements of the states of a system are shown to be very important in the state-variable design procedures of Chapter 10.

The state equations will now be derived. From (2-45) and (2-47),

$$\frac{d^2\theta(t)}{dt^2} = -\frac{B}{J}\frac{d\theta(t)}{dt} + \frac{K_\tau}{J} i_a(t)$$

or

$$\dot{x}_1(t) = -\frac{B}{J}x_1(t) + \frac{K_\tau}{J}x_2(t).$$

From (2-41) and (2-43),

$$\frac{di_a(t)}{dt} = \dot{x}_2(t) = -\frac{K_m}{L_m}x_1(t) - \frac{R_m}{L_m}x_2(t) + \frac{1}{L_m}e_a(t)$$

The output equation is $y(t) = d\theta(t)/\mathrm{dt} = x_1(t)$. Hence the state equations are

$$\dot{x}(t) = \begin{bmatrix} -\frac{B}{J} & \frac{K_\tau}{J} \\ -\frac{K_m}{L_m} & -\frac{R_m}{L_m} \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{L_m} \end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t)$$

where the input $u(t) = e_a(t)$. The simulation diagram for these equations is shown in Figure 3.8, where $\Omega(s) = \mathrm{L}[d\theta(t)/dt]$. This example develops a state model that is used in practical speed-control design; the model is neither of the canonical forms presented earlier.

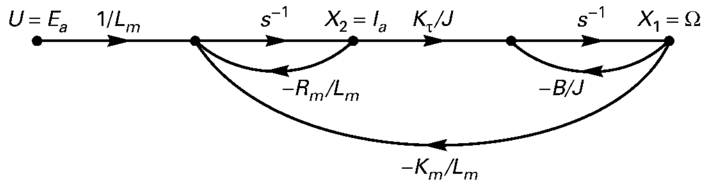


FIGURE 3.8
Simulation diagram for Example 3.3.

In this and the preceding section, we examined some specific ways to represent a system with state variables. These methods are by no means exhaustive, since the selection of the state-variables is not a unique process. As will be shown in Section 3.5, there are actually an unbounded number of different correct state-variable representations for any system.

The number of state variables required to model a system will always be equal to the order of the system. While this number is apparent for single-input, single-output systems described by a single nth-order differential equation or by a transfer function, such is sometimes not the case for multivariable systems such as the one in Example 3.1. For that example, the number of state variables is equal to the number of dependent variables plus the number of their derivatives up to $(r - 1)$, where r is the order of the highest derivative of a given dependent variable.

3.3 SOLUTION OF STATE EQUATIONS

We have developed procedures for writing the state equations of a system, given either the system differential equations or the system transfer function. In this section, we present two methods for finding the solution of the state equations.

3.3.1 Laplace Transform Solution

The standard form of the state equation is given by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{3-14}$$

This equation will now be solved using the Laplace transform. Consider the first equation in the set (3-14):

$$\dot{x}_1 = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + b_{11}u_1 + \cdots + b_{1r}u_r \tag{3-15}$$

where a_{ij} and b_{ij} are the appropriate elements of the matrices **A** and **B**, and the dependence on time is omitted for convenience. The Laplace transform of this equation yields

$$\begin{aligned} sX_1(s) - x_1(0) = a_{11}X_1(s) + a_{12}X_2(s) + \cdots + a_{1n}X_n(s) \\ + b_{11}U_1(s) + \cdots + b_{1r}U_r(s) \end{aligned} \tag{3-16}$$

where the initial condition is included, since we will find the complete solution. The Laplace transform of the second equation in (3-14) yields

$$\begin{aligned} sX_2(s) - x_2(0) = a_{21}X_1(s) + a_{22}X_2(s) + \cdots + a_{2n}X_n(s) \\ + b_{21}U_1(s) + \cdots + b_{2r}U_r(s) \end{aligned} \tag{3-17}$$

The Laplace transform of the remaining equations in (3-14) yields equations of the same form. These equations may be written in matrix form as

$$s\mathbf{X}(s) - \mathbf{x}(0) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s)$$

where

$$\mathbf{x}(0) = [x_1(0)\ x_2(0) \cdots x_n(0)]^T$$

We would like to solve this equation for $\mathbf{X}(s)$; to do this we collect all terms containing $\mathbf{X}(s)$ on the left side of the equation:

$$s\mathbf{X}(s) - \mathbf{A}\mathbf{X}(s) = \mathbf{x}(0) + \mathbf{B}\mathbf{U}(s) \tag{3-18}$$

It is necessary to factor $\mathbf{X}(s)$ in the left side to solve this equation. First the term $s\mathbf{X}(s)$ must be written as $s\mathbf{I}\mathbf{X}(s)$, where $\mathbf{I}$ is the identity matrix. Then

$$s\mathbf{I}\mathbf{X}(s) - \mathbf{A}\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})\mathbf{X}(s) = \mathbf{x}(0) + \mathbf{B}\mathbf{U}(s) \tag{3-19}$$

This additional step is necessary since the subtraction of the matrix **A** from the scalar s is not defined; we cannot factor $\mathbf{X}(s)$ directly in (3-18). Equation (3-19) may now be solved for $\mathbf{X}(s)$:

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) \tag{3-20}$$

and the state vector $\mathbf{x}(t)$ is the inverse Laplace transform of this equation.

To obtain a general relationship for the solution, we define the *state transition matrix* $\Phi(t)$ as

$$\Phi(t) = \mathcal{L}^{-1}[(s\mathbf{I} - \mathbf{A})^{-1}] \tag{3-21}$$

This matrix is also called the *fundamental matrix*. The matrix $(s\mathbf{I} - \mathbf{A})^{-1}$ is called the *resolvant* of $\mathbf{A}$[2]. Note that for an nth-order system, the state transition matrix is an $(n \times n)$ matrix. The inverse Laplace transform of a matrix is defined as the inverse Laplace transform of the elements of the matrix. Finding the inverse Laplace transform indicated in (3-21) in general is difficult, time consuming, and prone to errors. A more practical procedure for calculating the state vector $\mathbf{x}(t)$ is by way of a computer simulation. Both digital computer simulations and analog computer simulations are discussed later in this chapter.

As a final point, note from (3-20) that the state transition matrix is the solution to the differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \Rightarrow \mathbf{x}(t) = \Phi(t)\mathbf{x}(0) \tag{3-22}$$

Now an example is presented to illustrate the calculation of a state transition matrix using (3-21).

Example 3.4

As an example, consider the system described by the transfer function

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2 + 3s + 2} = \frac{s^{-2}}{1 + 3s^{-1} + 2s^{-2}}$$

The observer canonical form is used to develop a state model for this example, and the flow graph form of the simulation diagram is given in Figure 3.9. The state equations are then

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = [1 \quad 0]\,\mathbf{x}(t)$$

It is seen that Mason's gain formula yields the correct transfer function. In addition, a MATLAB program that calculates the transfer function is

```
e3p4=ss([-3 1;-2 0],[0; 1],[1 0],0);
tf(e3p4)
```

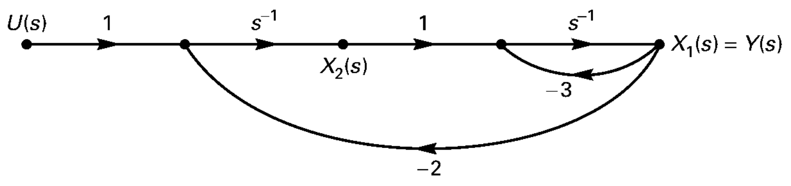


FIGURE 3.9
System for Example 3.4.

To find the state transition matrix, we first calculate the matrix ($s\mathbf{I}-\mathbf{A}$).

$$s\mathbf{I}-\mathbf{A} = s\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix} = \begin{bmatrix} s+3 & -1 \\ 2 & s \end{bmatrix}$$

To find the inverse of this matrix, we calculate its adjoint matrix.

$$\text{Adj}\,(s\mathbf{I}-\mathbf{A}) = \begin{bmatrix} s & 1 \\ -2 & s+3 \end{bmatrix}$$

The determinant of the matrix is

$$\det\,(s\mathbf{I}-\mathbf{A}) = s^2 + 3s + 2 = (s+1)(s+2)$$

and the inverse matrix is then the adjoint matrix divided by the determinant.

$$(s\mathbf{I}-\mathbf{A})^{-1} = \begin{bmatrix} \dfrac{s}{(s+1)(s+2)} & \dfrac{1}{(s+1)(s+2)} \\ \dfrac{-2}{(s+1)(s+2)} & \dfrac{s+3}{(s+1)(s+2)} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{-1}{s+1} + \dfrac{2}{s+2} & \dfrac{1}{s+1} + \dfrac{-1}{s+2} \\ \dfrac{-2}{s+1} + \dfrac{2}{s+2} & \dfrac{2}{s+1} + \dfrac{-1}{s+2} \end{bmatrix}$$

The state transition matrix is the inverse Laplace transform of this matrix.

$$\Phi(t) = \begin{bmatrix} -e^{-t}+2e^{-2t} & e^{-t}-e^{-2t} \\ -2e^{-t}+2e^{-2t} & 2e^{-t}-e^{-2t} \end{bmatrix}$$

Hence, we see that the state transition matrix for a second-order system is a (2×2) matrix. In a like manner, the state transition matrix for an nth-order system is $(n \times n)$. A MATLAB program that solves directly for the elements of $\Phi(t)$ using (3-22) is given by

```
>> Syms s
>> A = [-3 1;-2 0], B = [0;1]
>> SImA = s*eye(2)-A
>> Phis = inv(SImA)
>> phit = ilaplace(Phis)
```

MATLAB can also be used to find the inverse Laplace transforms of each of the element of the state transition matrix, from (3-21).

With the definition of the state transition matrix in (3-21), the equation for the complete solution of the state equations (3-20), which is repeated next, can be found.

[eq. (3-20)] $$\mathbf{X}(s) = (s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$$

We illustrate the complete solution of this equation using an example before giving the general form of the solution.

Example 3.5

Consider the same system as described in Example 3.4. State equations were derived to be

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(t)$$

with the Laplace transform of the transition matrix given by

$$\Phi(s) = \mathcal{L}\left[\phi(t)\right] = \left(s\mathbf{I}-\mathbf{A}\right)^{-1} = \begin{bmatrix} \dfrac{s}{(s+1)(s+2)} & \dfrac{-1}{(s+1)(s+2)} \\ \dfrac{-2}{(s+1)(s+2)} & \dfrac{s+3}{(s+1)(s+2)} \end{bmatrix}$$

Suppose that a unit step is applied as an input. Then $U(s) = 1/s$, and the second term in (3-20) becomes

$$(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s) = \begin{bmatrix} \dfrac{s}{(s+1)(s+2)} & \dfrac{1}{(s+1)(s+2)} \\ \dfrac{-2}{(s+1)(s+2)} & \dfrac{s+3}{(s+1)(s+2)} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\frac{1}{s}$$

$$= \begin{bmatrix} \dfrac{1}{s(s+1)(s+2)} \\ \dfrac{s+3}{s(s+1)(s+2)} \end{bmatrix} = \begin{bmatrix} \dfrac{\frac{1}{2}}{s} + \dfrac{-1}{s+1} + \dfrac{\frac{1}{2}}{s+2} \\ \dfrac{\frac{3}{2}}{s} + \dfrac{-2}{s+1} + \dfrac{\frac{1}{2}}{s+2} \end{bmatrix}$$

The inverse Laplace transform of this term is

$$\mathcal{L}^{-1}((s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s)) = \begin{bmatrix} \left(\dfrac{1}{2}\right) - e^{-t} + \left(\dfrac{1}{2}\right)e^{-2t} \\ \left(\dfrac{3}{2}\right) - 2e^{-t} + \left(\dfrac{1}{2}\right)e^{-2t} \end{bmatrix}$$

A MATLAB program that solves for this term is given by

```
>> A = [-3 1;-2 0], B=[0;1]
>> SImA = s*eye(2)-A
>> Phis = inv(SImA)
>> Steps = Phis*B*(1/s)
>> Steptime = ilaplace(Steps)
```

The state transition matrix was derived in Example 3.4. Hence the complete solution of the state equations is given by

$$\mathbf{x}(t) = \mathcal{L}^{-1}((s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s))$$

$$= \begin{bmatrix} -e^{-t}+2e^{-2t} & e^{-t}-e^{-2t} \\ -2e^{-t}+2e^{-2t} & 2e^{-t}-e^{-2t} \end{bmatrix}\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} + \begin{bmatrix} \left(\dfrac{1}{2}\right) - e^{-t} + \left(\dfrac{1}{2}\right)e^{-2t} \\ \left(\dfrac{3}{2}\right) - 2e^{-t} + \left(\dfrac{1}{2}\right)e^{-2t} \end{bmatrix}$$

and the state variables are given by

$$x_1(t) = (-e^{-t}+2e^{-2t})x_1(0) + (e^{-t}-e^{-2t})x_2(0) + \left(\frac{1}{2}\right) - e^{-t} + \left(\frac{1}{2}\right) - e^{-2t}$$

and

$$x_2(t) = (-2e^{-t}+2e^{-2t})x_1(0) + (2e^{-t}-e^{-2t})x_2(0) + \left(\frac{3}{2}\right) - 2e^{-t} + \left(\frac{1}{2}\right)e^{-2t}$$

Finding the complete solution of the state equations is long and involved, even for a second-order system. The necessity for reliable machine solutions, such as a digital computer simulation, is evident.

A general form of the solution of the state equations of (3-20) is developed next. The second term in the right member of this equation is a product of two terms in the Laplace variable s. Thus the inverse Laplace transform of this term can be expressed as a convolution integral (see Appendix **B**). The inverse Laplace transform of (3-20) is then

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) + \int_0^t \Phi(t - \tau)\mathbf{B}\mathbf{u}(\tau)\, d\tau \tag{3-23}$$

By the convolution theorem, this solution can also be expressed as

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) + \int_0^t \Phi(\tau)\, \mathbf{B}\mathbf{u}(t - \tau)\, d\tau \tag{3-24}$$

Note that the solution is composed of two terms. The first term is often referred to as either the *zero-input part* or the *initial-condition part* of the solution, and the second term is called either the *zero-state part* or the *forced part.* Equations (3-23) and (3-24) are sometimes called the *convolution solution.*

We see that the state transition matrix is central to the solution of the state equations. The solution in this form is quite difficult to calculate except for the simplest of systems. The preceding example is used to illustrate this form of the solution.

Example 3.6

In the preceding example, the input was a unit step. Thus, in (3-23),

$$\int_0^t \Phi(t - \tau)\mathbf{B}\mathbf{u}(\tau)\, d\tau = \int_0^t \begin{bmatrix} -e^{-(t-\tau)}+2e^{-2(t-\tau)} & e^{-(t-\tau)}-e^{-2(t-\tau)} \\ -2e^{-(t-\tau)}+2e^{-2(t-\tau)} & 2e^{-(t-\tau)}-e^{-2(t-\tau)} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\tau$$

$$= \begin{bmatrix} \displaystyle\int_0^t (e^{-(t-\tau)}-e^{-2(t-\tau)})\, d\tau \\ \displaystyle\int_0^t (2e^{-(t-\tau)}-e^{-2(t-\tau)})\, d\tau \end{bmatrix}$$

$$= \begin{bmatrix} \left(e^{-t}e^{\tau} - \left(\frac{1}{2}\right)e^{-2t}e^{2\tau}\right)\Big|_0^t \\ \left(2e^{-t}e^{\tau} - \left(\frac{1}{2}\right)e^{-2t}e^{2\tau}\right)\Big|_0^t \end{bmatrix} = \begin{bmatrix} (1 - e^{-t}) - \left(\frac{1}{2}\right)(1 - e^{-2t}) \\ 2(1 - e^{-t}) - \left(\frac{1}{2}\right)(1 - e^{-2t}) \end{bmatrix}$$

$$= \begin{bmatrix} \left(\frac{1}{2}\right) - e^{-t} + \left(\frac{1}{2}\right)e^{-2t} \\ \left(\frac{3}{2}\right) - 2e^{-t} + \left(\frac{1}{2}\right)e^{-2t} \end{bmatrix}$$

This result checks that of Example 3.5. The result derived here is only the forced part of the solution. The initial-condition part of the solution is obtained from the term $\mathbf{\Phi}(t)\mathbf{x}(0)$ in (3-23). This term was evaluated in Example 3.5 and is not repeated here.

The complete solution to the state equations was derived in this section. This solution may be evaluated either by the Laplace transform or by a combination of the Laplace transform and the convolution integral. Either procedure is long, time consuming, and prone to errors. The practical method for evaluating the time response of a system is through simulation.

3.3.2 Infinite Series Solution

As just shown, the state transition matrix can be evaluated using the Laplace transform. An alternative procedure for this evaluation is now developed.

One method of solution of differential equations is to assume as a solution an infinite series with unknown coefficients. The infinite series is then substituted into the differential equation to evaluate the unknown coefficients. This method is now used to find the state transition matrix. For the case that all inputs to a system are zero, the state equations may be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \tag{3-25}$$

with the solution, from (3-23),

$$\mathbf{x}(t) = \Phi(t)\mathbf{x}(0) \tag{3-26}$$

Since we are solving for a vector $\mathbf{x}(t)$, the solution must be assumed to be of the form

$$\begin{aligned} \mathbf{x}(t) &= (\mathbf{K}_0 + \mathbf{K}_1 t + \mathbf{K}_2 t^2 + \mathbf{K}_3 t^3 + \cdots)\mathbf{x}(0) = \sum_{i=0}^{\infty} \mathbf{K}_i t^i \mathbf{x}(0) \\ &= \mathbf{\Phi}(t)\mathbf{x}(0) \end{aligned} \tag{3-27}$$

where the $(n \times n)$ matrices $\mathbf{K}_i$ are unknown and t is the scalar time. Differentiating this expression yields

$$\dot{\mathbf{x}}(t) = (\mathbf{K}_1 + 2\mathbf{K}_2 t + 3\mathbf{K}_3 t^2 + \cdots)\,\mathbf{x}(0) \tag{3-28}$$

Substituting (3-27) and (3-28) into (3-25) yields

$$(\mathbf{K}_1 + 2\mathbf{K}_2 t + 3\mathbf{K}_3 t^2 + \cdots)\mathbf{x}(0) = \mathbf{A}(\mathbf{K}_0 + \mathbf{K}_1 t + \mathbf{K}_2 t^2 + \cdots)\mathbf{x}(0) \tag{3-29}$$

We next perform the following operations. First evaluate (3-29) at $t = 0$. Then differentiate (3-29) and evaluate the result at $t = 0$. Differentiate again and evaluate at $t = 0$. Repeat this operation, and each evaluation results in an equation in the unknown matrices $\mathbf{K}_i$. The effect of this procedure is to equate the coefficients of t^i in (3-29). The resulting equations are

$$\begin{aligned} \mathbf{K}_1 &= \mathbf{AK}_0 \\ 2\mathbf{K}_2 &= \mathbf{AK}_1 \\ 3\mathbf{K}_3 &= \mathbf{AK}_2 \\ &\vdots \end{aligned} \tag{3-30}$$

Evaluating (3-27) at $t = 0$ shows that $\mathbf{K}_0 = \mathbf{I}$. Then the other matrices are evaluated from (3-30) as

$$\begin{aligned} \mathbf{K}_1 &= \mathbf{A} \\ \mathbf{K}_2 &= \frac{\mathbf{A}^2}{2!} \\ \mathbf{K}_3 &= \frac{\mathbf{A}^3}{3!} \\ &\vdots \end{aligned} \tag{3-31}$$

Hence, from (3-27), the state transition matrix may be expressed as

$$\Phi(t) = \mathbf{I} + \mathbf{A}t + \mathbf{A}^2\frac{t^2}{2!} + \mathbf{A}^3\frac{t^3}{3!} + \cdots \tag{3-32}$$

Because of the similarity of (3-32) and the Taylor's series expansion of the scalar exponential,

$$e^{kt} = 1 + kt + k^2\frac{t^2}{2!} + k^3\frac{t^3}{3!} + \cdots \tag{3-33}$$

the state transition matrix is often written, for notational purposes only, as the matrix exponential

$$\Phi(t) = e^{\mathbf{A}t} \tag{3-34}$$

The *matrix exponential* is defined by (3-32) and (3-34). An example illustrating this technique is presented next.

Example 3.7

To give an example for which the series in (3-32) has a finite number of nonzero terms, the satellite model of Section 2.6 is used. From Example 2.13, the transfer function of the satellite is

$$G(s) = \frac{\Theta(s)}{T(s)} = \frac{1}{Js^2}$$

where $T(s) = \mathcal{L}[\tau(t)]$ is the torque applied to the satellite and $\theta(s)$ is the attitude angle of the satellite. A block diagram for a control system for the satellite is given in Figure 3.10(a). In this control system, both position $\theta(t)$ and rate $\theta(t)$ are measured and fed back to control the attitude of the satellite. The simulation diagram for the satellite is given in Figure 3.10(b). We write the state equations for the satellite directly from this simulation diagram:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ \dfrac{1}{J} \end{bmatrix} \mathrm{u}(t)$$

Thus in (3-32),

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{A}^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{A}^3 = \mathbf{A}\mathbf{A}^2 = \mathbf{0}$$

In a like manner,

$$\mathbf{A}^n = \mathbf{A}^2\mathbf{A}^{n-2} = \mathbf{0}; \qquad n \geq 3$$

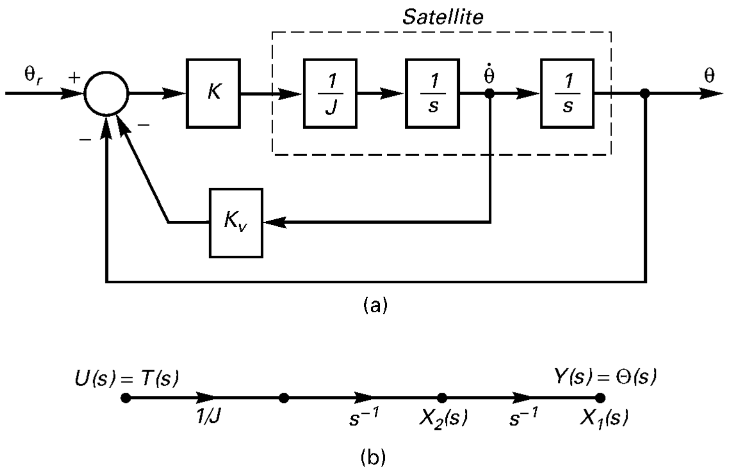


FIGURE 3.10
Attitude-control system for satellite.

The state transition matrix is then, from (3-32),

$$\Phi(t) = \mathbf{I} + \mathbf{A}t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} t = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

For this example, the state transition matrix is easily calculated using the infinite series expansion. In general this will not be the case.

The series expansion of $\mathbf{\Phi}(t)$ is well suited to evaluation on a digital computer if $\mathbf{\Phi}(t)$ is to be evaluated at only a few instants of time. The series expansion is also very useful in the analysis of digital control systems [3,4]. However, as a practical matter, the time response of a system should be evaluated by simulation.

3.4 TRANSFER FUNCTIONS

A procedure was given in Section 3.2 for writing the state equations of a system if the transfer function is known. In this section, we are interested in the inverse of this procedure: Given the state equations, find the transfer function. One approach is the use of the state equations to construct a simulation diagram and the use of Mason's gain formula to find the transfer function of the simulation diagram. This approach is direct but tends to be long and tedious except for the simplest of systems. In addition, finding all the loops and all the forward paths in a simulation diagram is an almost impossible task for high-order systems. In this section, we present a matrix procedure for finding the transfer function from the state equations; this procedure can be implemented as a digital computer program [2]. This procedure will now be developed. The standard form of the state equations is given by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) \\ y(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{3-35}$$

for a single-input, single-output system. The matrix $\mathbf{D}$ of (3-6) has been omitted, since it is normally the null matrix. The Laplace transform of the differential equation in (3-35) yields [see (3-18)]

$$s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}U(s) \tag{3-36}$$

ignoring initial conditions. This equation may be written as

$$(s\mathbf{I} - \mathbf{A})\mathbf{X}(s) = \mathbf{B}U(s) \tag{3-37}$$

and is easily solved for $\mathbf{X}(s)$:

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s) \tag{3-38}$$

The Laplace transform of the output equation in (3-35) yields

$$Y(s) = \mathbf{C}\mathbf{X}(s) \tag{3-39}$$

Substitution of (3-38) into (3-39) gives the desired transfer function,

$$Y(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}U(s) = G(s)U(s) \tag{3-40}$$

by the definition of the transfer function. Hence, the transfer function of the system is

$$G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \mathbf{C}\Phi(s)\mathbf{B} \tag{3-41}$$

For the case that D is not zero, the transfer function is found to be

$$G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + D \tag{3-42}$$

The interested reader may derive this relationship (see Problem 3.25). An example is now given to illustrate this procedure.

Example 3.8

For this example the system of Example 3.4 is used. The transfer function for that example was given as

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2 + 3s + 2}$$

and the state equations were found to be

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(t)$$
$$y(t) = [1 \;\; 0]\,\mathbf{x}(t)$$

The matrix $(s\mathbf{I} - \mathbf{A})^{-1}$ was calculated in Example 3.4. Then, from (3-41) and Example 3.4, the transfer function is given by

$$G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s+3 & -1 \\ 2 & s \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{s}{(s+1)(s+2)} & \frac{1}{(s+1)(s+2)} \\ \frac{-2}{(s+1)(s+2)} & \frac{s+3}{(s+1)(s+2)} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{(s+1)(s+2)} \\ \frac{s+3}{(s+1)(s+2)} \end{bmatrix} = \frac{1}{(s+1)(s+2)}$$

This transfer function checks the one given. A MATLAB program that verifies the transfer function calculation is given by

```
A = [-3 1;-2 0],B = [0;1],C = [1 0],D = [0]
[num,den]=ss2tf(A,B,C,D)
Gs = tf(num,den)
```

where n gives the numerator coefficients for $G(s)$ and d the denominator coefficients.

For the multivariable case (more than one input and/or more than one output), $\mathbf{U}(s)$ is an $(r \times 1)$ vector and $\mathbf{Y}(s)$ is a $(p \times 1)$ vector. The preceding derivation remains valid, with (3-40) becoming

$$\mathbf{Y}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) = \mathbf{G}(s)\mathbf{U}(s) \tag{3-43}$$

where $\mathbf{G}(s)$ is now a $(p \times r)$ matrix. The elements of $\mathbf{G}(s)$, $G_{ij}(s)$, are each transfer functions, where, by superposition,

$$G_{ij}(s) = \frac{Y_i(s)}{U_j(s)} \tag{3-44}$$

with all other inputs equal to zero, that is, with each $U_k(s)$, $k \neq j$, equal to zero. An example of a multivariable system is the temperature-control system in Section 2.9.

3.5 SIMILARITY TRANSFORMATIONS

In this chapter, procedures for finding a state-variable model from either the system differential equations or the system transfer function have been presented. It has been shown that a unique state model does not exist. Two general state models, the control canonical form and the observer canonical form, can always be found. A single-input single-output system has only one input–output model (transfer function), but the number of internal models (state models) is unbounded, as we now show.

The state model of a single-input, single-output model is

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) \tag{3-45}$$

$$y(t) = \mathbf{C}\mathbf{x}(t) + Du(t) \tag{3-46}$$

and the transfer function is given by

[eq. (3–42)] $$\frac{Y(s)}{U(s)} = G(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + D$$

There are many combinations of the matrices **A, B, C**, and D that will satisfy (3-42) for a given $G(s)$.

Suppose that we are given a state model of a system as in (3-45) and (3-46). Now, define a different state vector $\mathbf{v}(t)$ that is of the same order as $\mathbf{x}(t)$, such that the elements of $\mathbf{v}(t)$ are linear combinations of the elements of $\mathbf{x}(t)$, that is,

$$\begin{aligned} v_1(t) &= q_{11}x_1(t) + q_{12}x_2(t) + \cdots + q_{1n}x_n(t) \\ v_2(t) &= q_{21}x_1(t) + q_{22}x_2(t) + \cdots + q_{2n}x_n(t) \\ &\;\;\vdots \\ v_n(t) &= q_{n1}x_1(t) + q_{n2}x_2(t) + \cdots + q_{nn}x_n(t) \end{aligned} \tag{3-47}$$

This equation can be written in matrix form:

$$\mathbf{v}(t) = \mathbf{Q}\mathbf{x}(t) = \mathbf{P}^{-1}\mathbf{x}(t) \tag{3-48}$$

where the matrix **Q** has been defined as the inverse of the matrix **P**, to satisfy common notation. Hence, the state vector $\mathbf{x}(t)$ can be expressed as

$$\mathbf{x}(t) = \mathbf{P}\mathbf{v}(t) \tag{3-49}$$

where the matrix **P** is called a transformation matrix, or simply a transformation. An example is now given.

Example 3.9

Consider the system of Example 3.4, which has the transfer function

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2 + 3s + 2}$$

and the state equations

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t)$$

For example, suppose that the elements of $\mathbf{v}(t)$ are arbitrarily defined as

$$v_1(t) = x_1(t) + x_2(t)$$
$$v_2(t) = x_1(t) + 2x_2(t)$$

as shown in Figure 3.11. Thus

$$\mathbf{v}(t) = \mathbf{Q}\mathbf{x}(t) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\mathbf{x}(t)$$

and

$$\mathbf{P}^{-1} = \mathbf{Q} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \therefore \mathbf{P} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

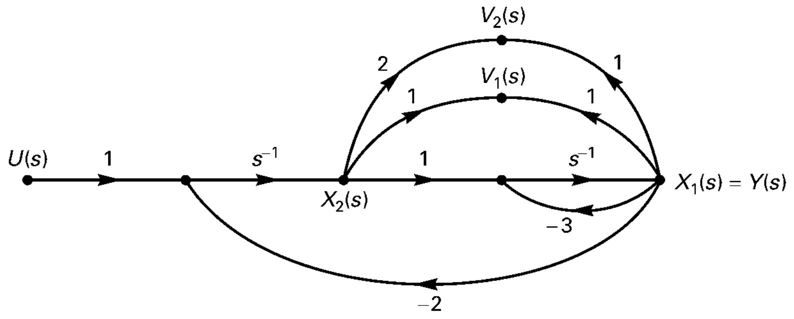


FIGURE 3.11
System for Example 3.9.

Thus, the components of $\mathbf{x}(t)$ can be expressed as functions of the components of $\mathbf{v}(t)$.

$$x_1(t) = 2v_1(t) - v_2(t)$$
$$x_2(t) = -v_1(t) + v_2(t)$$

It is seen from this exercise that, given the vector $\mathbf{x}(t)$, we can solve for the vector $\mathbf{v}(t)$. Or, given the vector $\mathbf{v}(t)$, we can solve for the vector $\mathbf{x}(t)$. The matrix inversion can be verified by the MATLAB program

```
Q = [1 1; 1 2];
P = inv(Q)
```

The preceding example illustrates transforming from one state vector to a different state vector. This transformation alters the internal model of the system (the state model) but does not alter the input–output model of the system (the transfer function). This type of transformation is called a *similarity transformation*. The details of similarity transformations are developed next.

Assume that we are given a state model of a *multivariable* system of the form of (3-45) and a similarity transformation of the form of (3-49).

[eq.(3-45)] $$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

[eq.(3-49)] $$\mathbf{x}(t) = \mathbf{P}\mathbf{v}(t)$$

Substituting (3-49) into (3-45) yields

$$\mathbf{P}\dot{\mathbf{v}}(t) = \mathbf{A}\mathbf{P}\mathbf{v}(t) + \mathbf{B}u(t) \tag{3-50}$$

Solving this equation for $\dot{\mathbf{v}}(t)$ results in the state model for the variables $\mathbf{v}(t)$:

$$\dot{\mathbf{v}}(t) = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}\mathbf{v}(t) + \mathbf{P}^{-1}\mathbf{B}u(t) \tag{3-51}$$

The output equation for a multivariable system

[eq.(3-46)] $$y(t) = \mathbf{C}\mathbf{x}(t) + Du(t)$$

becomes, using (3-49),

$$y(t) = \mathbf{C}\mathbf{P}\mathbf{v}(t) + Du(t) \tag{3-52}$$

Therefore, we have the state equations expressed as a function of the state vector $\mathbf{x}(t)$ in (3-45) and (3-46) and as a function of the transformed state vector $\mathbf{v}(t)$ in (3-51) and (3-52).

The state equations as a function of $\mathbf{v}(t)$ can be expressed in the standard format for a multivariable system as

$$\dot{\mathbf{v}}(t) = \mathbf{A}_v\mathbf{v}(t) + \mathbf{B}_v\mathbf{u}(t) \tag{3-53}$$
$$\mathbf{y}(t) = \mathbf{C}_v\mathbf{v}(t) + D_v u(t) \tag{3-54}$$

where the subscripts indicates the transformed matrices. The matrices for the vector $\mathbf{x}(t)$ will not be subscripted. Comparing (3-53) with (3-51) and (3-54) with (3-52), we see that the equations for the transformed matrices are

$$\mathbf{A}_v = \mathbf{P}^{-1}\mathbf{A}\mathbf{P} \qquad \mathbf{B}_v = \mathbf{P}^{-1}\mathbf{B}$$
$$\mathbf{C}_v = \mathbf{C}\mathbf{P} \qquad \mathbf{D}_v = \mathbf{D} \tag{3-55}$$

An example will be presented now to illustrate similarity transformations.

Example 3.10

As an example, consider the system of Example 3.9,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(t)$$
$$y(t) = \mathbf{C}\mathbf{x}(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t)$$

with the transformation

$$\mathbf{P}^{-1} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

From (3-55), the system matrices for the transformed system become

$$\mathbf{A}_v = \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} -5 & 1 \\ -7 & 1 \end{bmatrix}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -11 & 6 \\ -15 & 8 \end{bmatrix}$$
$$\mathbf{B}_v = \mathbf{P}^{-1}\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$
$$\mathbf{C}_v = \mathbf{C}\mathbf{P} = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -1 \end{bmatrix}$$

The transformed state equations are then

$$\dot{\mathbf{v}}(t) = \mathbf{A}_v\mathbf{v}(t) + \mathbf{B}_v u(t) = \begin{bmatrix} -11 & 6 \\ -15 & 8 \end{bmatrix}\mathbf{v}(t) + \begin{bmatrix} 1 \\ 2 \end{bmatrix}u(t)$$
$$y(t) = \mathbf{C}_v\mathbf{v}(t) = \begin{bmatrix} 2 & -1 \end{bmatrix}\mathbf{v}(t)$$

The calculations in this example can be verified by the MATLAB program

```
  A = [-3 1; -2 0], B = [0; 1], C = [1 0], D = [0]
  T = [1 1; 1 2];
sys = ss(A,B,C,D)
sysv = ss2ss(sys, T)
```

The preceding example gives two state models of the same system. If a different transformation matrix **P** had been chosen, a third model would result. In fact, for each different transformation **P** that has an inverse, a different state model results. Thus, there are an unlimited number of state models for a given system transfer function. The choice of the state model for a given system is usually based on using the natural state variables (position, velocity, etc.), on ease of design, and so forth.

To check the state model developed in the preceding example, we derive the transfer function of this model by two different procedures.

Example 3.11

To obtain the transfer function for the transformed system equations of Example 3.10 using Mason's gain formula, the simulation diagram is drawn from the state equations,

$$\dot{\mathbf{v}}(t) = \begin{bmatrix} -11 & 6 \\ -15 & 8 \end{bmatrix} \mathbf{v}(t) + \begin{bmatrix} 1 \\ 2 \end{bmatrix} u(t)$$

$$\dot{y}(t) = \begin{bmatrix} 2 & -1 \end{bmatrix} \mathbf{v}(t)$$

It is shown in Figure 3.12. Note that the transfer function of this simulation diagram must be

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

To verify this transfer function, Mason's gain formula is used. Note in Figure 3.12 that the simulation diagram has three loops and that two of these loops do not touch. Thus for Mason's gain formula,

$$\begin{aligned}\Delta &= 1 - [-11s^{-1} + 8s^{-1} + (-15)\,s^{-1}(6)\,s^{-1}] + [-11s^{-1}]\,[8s^{-1}] \\ &= 1 + 3s^{-1} + 2s^{-2}\end{aligned}$$

There are four forward paths; as an exercise the reader might try to find these before reading further. The forward path gains and the corresponding Δ_i are

$$\begin{aligned}
M_1 &= 2s^{-1} & \Delta_1 &= 1 - 8s^{-1} \\
M_2 &= 15s^{-2} & \Delta_2 &= 1 \\
M_3 &= -2s^{-1} & \Delta_3 &= 1 + 11s^{-1} \\
M_4 &= (2)\,(6)\,(2)s^{-2} & \Delta_4 &= 1
\end{aligned}$$

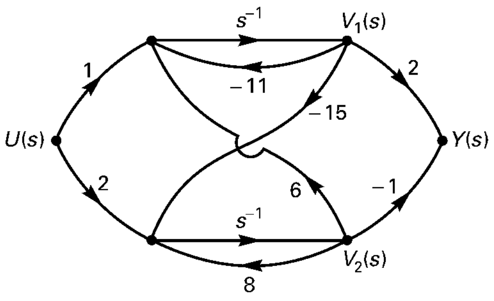


FIGURE 3.12
System for Example 3.11.

The transfer function is then

$$G(s) = \frac{M_1\Delta_1 + M_2\Delta_2 + M_3\Delta_3 + M_4\Delta_4}{\Delta}$$

$$= \frac{(2s^{-1})\,(1 - 8s^{-1}) + (15s^{-2})\,(1) + (-2s^{-1})\,(1 + 11s^{-1}) + (24s^{-2})\,(1)}{1 + 3s^{-1} + 2s^{-2}}$$

$$= \frac{s^{-2}}{1 + 3s^{-1} + 2s^{-2}} = \frac{1}{s^2 + 3s + 2}$$

This verifies the transfer function. Note the difficulty in identifying all loops and forward paths.

This transfer function can also be checked using the matrix approach

[eq.(3-42)] $$G(s) = \mathbf{C}_v(s\mathbf{I} - \mathbf{A}_v)^{-1}\mathbf{B}_v$$

Now

$$s\mathbf{I} - \mathbf{A}_v = \begin{bmatrix} s + 11 & -6 \\ 15 & s - 8 \end{bmatrix}$$

and thus

$$\det(s\mathbf{I} - \mathbf{A}_v) = s^2 + 3s - 88 + 90 = s^2 + 3s + 2$$

The adjoint of $s\mathbf{I} - \mathbf{A}_v$ is given by

$$\operatorname{Adj}(s\mathbf{I} - \mathbf{A}_v) = \begin{bmatrix} s - 8 & 6 \\ -15 & s + 11 \end{bmatrix}$$

Thus the inverse of this matrix is

$$(s\mathbf{I} - \mathbf{A}_v)^{-1} = \frac{\operatorname{Adj}(s\mathbf{I} - \mathbf{A}_v)}{\det(s\mathbf{I} - \mathbf{A}_v)} = \begin{bmatrix} \dfrac{s - 8}{s^2 + 3s + 2} & \dfrac{6}{s^2 + 3s + 2} \\ \dfrac{-15}{s^2 + 3s + 2} & \dfrac{s + 11}{s^2 + 3s + 2} \end{bmatrix}$$

From (3-42), the system transfer function is given by

$$G(s) = \mathbf{C}_v(s\mathbf{I} - \mathbf{A}_v)^{-1}\mathbf{B}_v = \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} \dfrac{s - 8}{s^2 + 3s + 2} & \dfrac{6}{s^2 + 3s + 2} \\ \dfrac{-15}{s^2 + 3s + 2} & \dfrac{s + 11}{s^2 + 3s + 2} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{2s - 1}{s^2 + 3s + 2} & \dfrac{-s + 1}{s^2 + 3s + 2} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \frac{1}{s^2 + 3s + 2}$$

The transfer function is checked, with much less probability of error when compared to using Mason's gain formula. Recall also that the matrix procedure can be implemented in a digital computer program, as illustrated in the MATLAB program of Example 3.8.

Similarity transformations have been demonstrated through examples. Certain important properties of these transformations are derived next. Consider first the determinant of $s\mathbf{I} - \mathbf{A}_v$. From (3-55),

$$\det(s\mathbf{I} - \mathbf{A}_v) = \det(s\mathbf{I} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}) = \det(s\mathbf{P}^{-1}\mathbf{I}\mathbf{P} - \mathbf{P}^{-1}\mathbf{A}\mathbf{P}) \tag{3-56}$$

where det $(\cdot)$ denotes the determinant. For two square matrices

$$\det \mathbf{R}_1\mathbf{R}_2 = \det \mathbf{R}_1 \det \mathbf{R}_2$$

Then (3-56) becomes

$$\det(s\mathbf{I} - \mathbf{A}_v) = \det\mathbf{P}^{-1} \det(s\mathbf{I} - \mathbf{A}) \det\mathbf{P} \tag{3-57}$$

Since, for a matrix square $\mathbf{R}$,

$$\mathbf{R}^{-1}\mathbf{R} = \mathbf{I}$$

then

$$\det \mathbf{R}^{-1}\det \mathbf{R} = \det \mathbf{R}^{-1}\mathbf{R} = \det \mathbf{I} = 1 \tag{3-58}$$

Thus (3-57) becomes

$$\det(s\mathbf{I} - \mathbf{A}_v) = \det(s\mathbf{I} - \mathbf{A}) \tag{3-59}$$

The zeros of $\det(s\mathbf{I} - \mathbf{A})$ are the characteristic values, or eigenvalues, of $\mathbf{A}$ (see Appendix A). Thus the characteristic values of $\mathbf{A}_v$ are equal to the characteristic values of $\mathbf{A}$. This is the first property of similarity transformations.

A second property can be derived as follows. From (3-55),

$$\det \mathbf{A}_v = \det \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \det \mathbf{P}^{-1} \det \mathbf{A} \det \mathbf{P} = \det \mathbf{A} \tag{3-60}$$

Then the determinant of $\mathbf{A}_v$ is equal to the determinant of $\mathbf{A}$. This property can also be seen from the fact that the determinant of a matrix is equal to the product of its characteristic values (see Appendix A). As shown earlier, the characteristic values of $\mathbf{A}_v$ are equal to those of $\mathbf{A}$; thus the determinants must be equal.

The third property of a similarity transformation can also be seen from the fact that the eigenvalues of $\mathbf{A}_v$ are equal to those of $\mathbf{A}$. Since the trace of a matrix is equal to the sum of the eigenvalues,

$$\mathrm{tr}\mathbf{A}_v = \mathrm{tr}\mathbf{A}$$

To summarize the properties of similarity transformations, let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of the matrix $\mathbf{A}$. Then for the similarity transformation

$$\mathbf{A}_v = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$$

1. The eigenvalues of $\mathbf{A}$ are equal to those of $\mathbf{A}_v$; or

$$\det(s\mathbf{I} - \mathbf{A}) = \det(s\mathbf{I} - \mathbf{A}_v) = (s - \lambda_1)(s - \lambda_2)\cdots(s - \lambda_n) \tag{3-61}$$

2. The determinant of $\mathbf{A}$ is equal to the determinant of $\mathbf{A}_v$:

$$\det\mathbf{A} = \det\mathbf{A}_v = \lambda_1\lambda_2\cdots\lambda_n \tag{3-62}$$

3. The trace of $\mathbf{A}$ is equal to the trace of $\mathbf{A}_v$:

$$\text{tr}\mathbf{A} = \text{tr}\mathbf{A}_v = \lambda_1 + \lambda_2 + \cdots + \lambda_n \tag{3-63}$$

4. The following transfer functions are equal:

$$\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \mathbf{C}_v(s\mathbf{I} - \mathbf{A}_v)^{-1}\mathbf{B}_v$$

The proof of the fourth property is left as an exercise (see Problem 3.30); this property is illustrated in Example 3.11. The first three properties are illustrated by the following example.

Example 3.12

The similarity transformation of Example 3.10 is used to illustrate the three properties just developed. From Example 3.10, the matrices $\mathbf{A}$ and the $\mathbf{A}_v$ are given by

$$\mathbf{A} = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix} \quad \mathbf{A}_v = \begin{bmatrix} -11 & 6 \\ -15 & 8 \end{bmatrix}$$

Then

$$\det(s\mathbf{I} - \mathbf{A}) = \begin{vmatrix} s+3 & -1 \\ 2 & s \end{vmatrix} = s^2 + 3s + 2$$

and

$$\det(s\mathbf{I} - \mathbf{A}_v) = \begin{vmatrix} s+11 & -6 \\ 15 & s-8 \end{vmatrix} = s^2 + 3s - 88 + 90 = s^2 + 3s + 2$$

Thus the two determinants are equal.

Next, the eigenvalues are found from

$$\det(s\mathbf{I} - \mathbf{A}) = s^2 + 3s + 2 = (s+1)(s+2)$$

and $\lambda_1 = -1$, $\lambda_2 = -2$. The determinants of the two matrices are

$$\det\mathbf{A} = \begin{vmatrix} -3 & 1 \\ -2 & 0 \end{vmatrix} = 2 \qquad \det\mathbf{A}_v = \begin{vmatrix} -11 & 6 \\ -15 & 8 \end{vmatrix} = -88 + 90 = 2$$

and both determinants are equal to the products of the eigenvalues. The traces of the two matrices are both equal to −3, which is the sum of the eigenvalues. The results in this example can be verified with the MATLAB program

```
A = [-3 1; -2 0]; Av = [-11 6; -15 8];
detsImA = poly(A),  detsImAv = poly(Av)
evA = eig(A),       evAv = eig(Av)
detA = det(A),      detAv = det(Av)
trA = trace(A),     trAv = trace(Av)
```

3.6 DIGITAL SIMULATION

It has been stated previously that the practical procedure for finding the time response of a system is through simulation, rather than by directly solving the differential equations or by using the Laplace transform. The Laplace transform is used throughout this book because of the ease in analyzing low-order systems. However, for higher-order systems, simulations are necessary.

In this section, we consider the numerical solution of differential equations by *integration algorithms.* Many algorithms are available for numerical integration [5].

Integration algorithms differ in execution speed, accuracy, programming complexity, and so forth. In this section, we consider a very simple algorithm—*Euler's method.* Euler's method is seldom used in practical situations because more efficient and more accurate algorithms are available. It is presented here because of the simplicity and its ease of programming.

The process of numerically integrating a time function using Euler' s method is illustrated in Figure 3.13. We wish to integrate $z(t)$ numerically; that is, we wish to find $x(t)$, where

$$x(t) = \int_0^t z(\tau)\, d\tau + x(0) \tag{3-64}$$

Suppose that we know $x(\mathrm{t})$ at $t = (k-1)H$; that is, we know $x[(k-1)H]$ and we want to calculate $x(kH)$, where H is called the numerical integration increment and is the step size of the algorithm. Euler's algorithm is obtained by assuming $z(t)$ constant at the value $z[(k-1)H]$ for $(k-1)H \le t < kH$. Then

$$x(kH) = x[(k-1)H] + Hz[(k-1)H] \tag{3-65}$$

In (3-65), $x(kH)$ is only an approximation for $x(t)$ in (3-64) evaluated at $t = kH$. Note in Figure 3.13 that we are approximating the area under the $z(t)$ curve for $(k-1)H \le t < kH$ with the area of the rectangle shown. For this reason, this method is also called the *rectangular rule.*

In simulation, we are interested in the integration of differential equations [that is, in (3-64), $z(t)$ is the derivative *of* $x(t)$]. We illustrate this case with a simple example. Suppose that we wish to solve the differential equation

$$\dot{x}(t) + x(t) = 0 \qquad x(0) = 1 \tag{3-66}$$

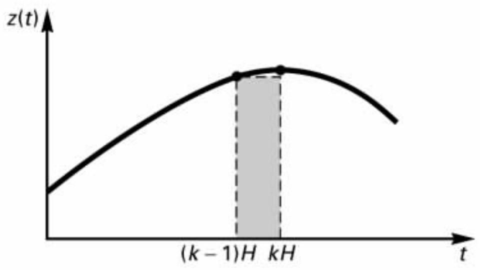


FIGURE 3.13
Euler rule for numerical integration.

The solution is obviously

$$x(t) = e^{-t}, \qquad t \geq 0$$

However, by numerical integration in (3-65),

$$x(kH) = x[(k-1)H] + H\{-x[(k-1)H]\} \tag{3-67}$$

since, from (3-66), for this differential equation,

$$z(t) = \dot{x}(t) = -x(t)$$

Suppose that we choose $H = 0.1$ s. Then, solving (3-67) iteratively starting with $k = 1$ [we know $x(0)$],

$$\begin{aligned} x(0.1) &= x(0) - Hx(0) = 1.0 - (0.1)(1.0) = 0.9 \\ x(0.2) &= x(0.1) - Hx(0.1) = 0.9 - 0.09 = 0.81 \\ &\vdots \\ x(1.0) &= x(0.9) - Hx(0.9) = 0.3487 \end{aligned}$$

Since, for this example, we know the solution, we calculate $x(t)$ at $t = 1.0$ s as

$$x(1.0) = e^{-1.0} = 0.3679$$

and we can see the error due to the numerical integration. If we choose $H = 0.01$ s, the value of $x(1.0)$ is calculated to be 0.3660, and the error is much less.

In the preceding example, if we choose H larger, the error is larger. If we decrease H, the error decreases, as shown. In fact, as H decreases, the error will decrease to a minimum value. Then a further decrease in H results in an increase in the error, due to round-off in the computations. With H equal to 0.1 s, 10 iterations are required to calculate $x(1)$. With H equal to 0.001 s, 1000 iterations are required to calculate $x(1)$. The round-off errors in the computations are larger for the latter case, since more calculations are made. If H is made sufficiently smaller, this round-off error becomes appreciable. Thus, if H is chosen to be too large, the errors are large due to the algorithm being a poor approximation for the integral. If H is chosen to be too small, the errors are also large because of the round-off in the computations.

An acceptable value of H is usually found by experimenting with the algorithm, that is, by varying H and noting the effect on the solution. This can be done by choosing a value of H and calculating the solution. Then H is reduced and the solution is calculated again. If the solution has changed very little, we can conclude that the first value of H is small enough. If not, H is reduced again and the same test applied. The value of H can also be increased, with the same test used.

Next, we consider the simulation of a system described by the state equations

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{3-68}$$

To develop Euler's rule for this matrix equation, consider the first element of the state vector, $x_1(t)$. This element can be expressed as

$$x_1(t) = \int_0^t \dot{x}_1(\tau)\, d\tau + x_1(0) \tag{3-69}$$

Comparing this equation with (3-64), from (3-65) we write

$$x_1(kH) = x_1[(k-1)H] + H\dot{x}_1[(k-1)H] \tag{3-70}$$

The same development applies for any element of the state vector, $x_i(t)$. Hence we can write

$$x_i(kH) = x_i[(k-1)H] + H\dot{x}_i[(k-1)H] \qquad 1 \le i \le n \tag{3-71}$$

This equation may be written in terms of vectors:

$$\mathbf{x}(kH) = \mathbf{x}[(k-1)H] + H\dot{\mathbf{x}}[(k-1)H] \tag{3-72}$$

where, from (3-68), $\dot{\mathbf{x}}[(k-1)H]$ is given by

$$\dot{\mathbf{x}}[(k-1)H] = \mathbf{A}\mathbf{x}[(k-1)H] + \mathbf{B}\mathbf{u}[(k-1)H] \tag{3-73}$$

The numerical integration algorithm is then

1. Let $k = 1$.
2. Evaluate $\dot{\mathbf{x}}[(k\text{-}1)H]$ in (3-73).
3. Evaluate $\mathbf{x}(kH)$ in (3-72).
4. Let $k = k + 1$.
5. Go to step 2.

This algorithm is particularly easy to program but in general requires a much smaller step size than does some of the more complex algorithms. The interested reader may consult Ref. 5 for additional discussion of numerical integration. An example of the integration of state equations is given next.

Example 3.13

In this example, we illustrate the numerical integration of state equations by Euler's rule. For the example we use the state equations of Example 3.5, since the complete analytical solution of the equations was derived in Example 3.5. From that example, the state equations are given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t)$$

and the solution is, for a unit step input,

$$x_1(t) = (-e^{-t}+2e^{-2t})x_1(0) + (e^{-t}-e^{-2t})x_2(0) + \left(\tfrac{1}{2}\right)-e^{-t}+\left(\tfrac{1}{2}\right)e^{-2t}$$

$$x_2(t) = (-2e^{-t}+2e^{-2t})x_1(0) + (2e^{-t}-e^{-2t})x_2(0) + \left(\tfrac{2}{3}\right)-2e^{-t}+\left(\tfrac{1}{2}\right)e^{-2t}$$

Suppose we choose $H = 0.01$ s. In addition, let $\mathbf{x}(0) = \mathbf{0}$ to simplify the calculations somewhat. From step 2 of the algorithm and (3-73),

$$\dot{\mathbf{x}}(0) = \mathbf{A}\mathbf{x}(0) + \mathbf{B}u(0) = \mathbf{B}u(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}(1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Then, from step 3 and (3-72),

$$\mathbf{x}(0.01) = \mathbf{x}(0) + H\dot{\mathbf{x}}(0) = H\dot{\mathbf{x}}(0) = (0.01)\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix}$$

For the second iteration of the algorithm,

$$\dot{\mathbf{x}}(0.01) = \mathbf{A}\mathbf{x}(0.01) + \mathbf{B}u(0.01) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0.01 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.01 \\ 1 \end{bmatrix}$$

and from (3-72),

$$\mathbf{x}(0.02) = \mathbf{x}(0.01) + H\dot{\mathbf{x}}(0.01) = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix} + (0.01)\begin{bmatrix} 0.01 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.0001 \\ 0.02 \end{bmatrix}$$

An additional iteration yields

$$\dot{\mathbf{x}}(0.02) = \begin{bmatrix} -3 & 1 \\ -2 & 0 \end{bmatrix}\begin{bmatrix} 0.0001 \\ 0.02 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.0197 \\ 0.9998 \end{bmatrix}$$

$$x(0.03) = \begin{bmatrix} 0.0001 \\ 0.02 \end{bmatrix} + (0.01)\begin{bmatrix} 0.0197 \\ 0.9998 \end{bmatrix} = \begin{bmatrix} 0.00030 \\ 0.03000 \end{bmatrix}$$

The exact values of the state vector can be calculated from the analytical solution just given and are computed with the simulation values in Table 3.1. Although giving values of the state vector over this short period of time does not give a good indication of the accuracy of the algorithm for the given value of H, this example does indicate the nature of numerical integration.

TABLE 3.1 Numerical Integration Results

	Simulation		Exact	
t	$x_1(t)$	$x_2(t)$	$x_1(t)$	$x_2(t)$
0	0	0	0	0
0.01	0	0.01	0.00005	0.00999
0.02	0.0001	0.02	0.0002	0.01999
0.03	0.0003	0.03	0.0004	0.02999

The iterative nature of numerical integration makes it well suited for digital computation. A MATLAB program that implements the Euler rule for this example is given by.

```
A = [−3 1; −2 0]; B = [0; 1]; C = [1 0];
H = 0.01;
xk = [0;0 ];
for k=1:4
   t = (k-1)*H;
   t
   xk
   xdotkp1 = A*xk + B;
   xkp1 = xk + H*xdotkp1;
   xk = xkp1;
end
```

The MATLAB control system toolbox offers a simple and convenient procedure for numerically solving linear constant-coefficient differential equations. One procedure is illustrated in the following example.

Example 3.14

We consider the system of Example 3.13. The state equations and the system output $y(t)= x_1(t)$ are given in that example. One MATLAB program that simulates this system is given by

```
A = [−3 1; −2 0]; B = [0; 1]; C = [1 0]; D = 0;
e3p14=ss(A, B, C, D)
step(e3p14)
```

The first statement enters the systems matrices, the second one identifies the system e3p14 as a system in state space, and the third one calculates the system step response $y(t)$. This response is plotted, with the result shown in Figure 3.14. Note that the final value of $y(t)$ is 0.5, which is checked by the response given in Example 3.13.

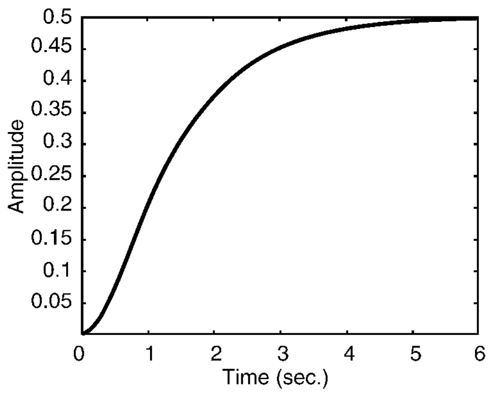


FIGURE 3.14
Simulation step response for Example 3.14.

Almost all numerical-integration routines in MATLAB choose the step size automatically. Several routines use a variable-step-size routine, in which the step size varies as the calculated signals vary.

As a second example, the total response of the system of Example 3.13 to a step input and nonzero initial conditions is now calculated by simulation.

Example 3.15

We now modify the program given in the last example to calculate the system step response with the initial conditions $\mathbf{x}(0) = [1\ 1]^T$. The solution to this set of equations is given in Example 3.5, with $y(t)$ given by

$$y(t) = x_1(t) = 0.5 - e^{-t} + 1.5e^{-2t} \qquad t \geq 0$$

The program required is

```
A = [-3 1;-2 0]; B = [0; 1]; C = [1 0]; D = 0;
e3p14 = ss(A, B, C, D)
t = 0:0.1:6;
u = stepfun(t,0);
x0 = [1; 1];
lsim(e3p14, u, t, x0)
```

The third statement sets the initial time to zero, the numerical integration increment to 0.1 s, and the final time to 6 s. The fourth statement identifies the input as a unit step function, and the fifth statement enters the initial conditions. The sixth statement performs the simulation. The response is plotted in Figure 3.15. If the last statement is replaced with

$$[y, t] = \text{lsim}(\text{e3p14}, u, t, x0)y', t'$$

the values of $y(t)$ and t are listed. For example,

$$y(t)|_{t=1} = (0.5 - e^{-t} + 1.5e^{-2t})|_{t=1} = 0.3351$$

Running the simulation gives the same value, which verifies the solution found in Example 3.5.

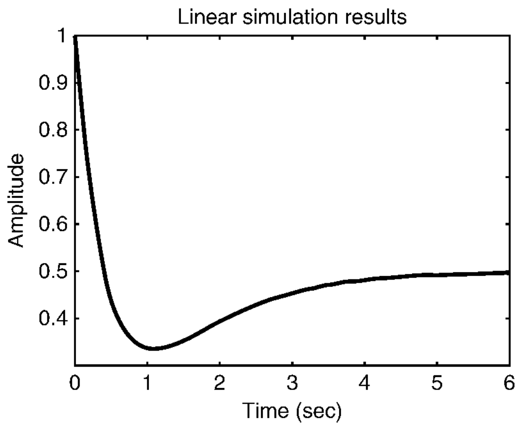


FIGURE 3.15
Simulation response for Example 3.15.

Example 3.16

In this example, we illustrate MATLAB simulation using a transfer function. It is shown in Example 3.4 that the transfer function for the system of Example 3.14 is

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

A MATLAB program that finds the system unit step response is as follows:

```
num = [0 0 1]; den = [1 3 2];
e3p16 = tf(num, den);
step(e3p16)
```

This simulation results in a plot that is identical to that of Figure 3.14.

Simulation using MATLAB offers many options not covered here. Some of these options will be covered as required. In addition, for more complex systems, SIMULINK®, a simulation program based on MATLAB, is available, and will be introduced at appropriate points in this book.

3.7 CONTROLS SOFTWARE

Many commercial software packages are available for the analysis, design, and simulation of feedback control systems. In general, we use the student versions of MATLAB and SIMULINK in this book. These packages contain all the required control-system analysis and simulations programs. Some of the design techniques introduced in this book are not included in MATLAB. These techniques are implemented as MATLAB m-files in the sections on design, and can be run with the MATLAB package. See the Preface for instructions for downloading all MATLAB programs given in this book.

3.8 SUMMARY

In this chapter, we presented the state-variable method of modeling physical systems. This method of modeling is required for the simulation of a system. Simple digital and analog simulation was also covered in this chapter. State-variable models are also required in the area called modern control analysis and design, which is introduced in Chapter 10. The advantage of the state model of a system is that knowledge of the internal structure as well as the input–output characteristics of the system are available, whereas the transfer function model gives us only input–output information. Generally, we make use of both model forms in analysis and design.

本章介绍了如何应用状态空间模型来描述现实物理系统。这些建模方法都需要系统仿真。因此,我们简单介绍了数值模拟方法。状态空间模型在现代控制的分析与设计中占据着十分重要的位置。状态空间模型的优点是能够反映系统的内部耦合结构以及系统的输入-输出特性。相反,传递函数模型只能给出系统的输入-输出信息。

REFERENCES

1. P. M. De Russo, R. J. Roy, and C. M. Close. *State Variables for Engineers.* New York: Wiley, 1965.
2. B. Friedland. *Control System Design.* New York: McGraw-Hill, 1986.
3. G. F. Franklin, J. D. Powell, and M. Workman. *Digital Control of Dynamic Systems,* 3rd ed. Reading, MA: Addison-Wesley, 1998.
4. C. L. Phillips and H. T. Nagle. *Digital Control System Analysis and Design,* 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1995.
5. S. D. Conte and C. deBoor. *Elementary Numerical Analysis: An Algorithmic Approach.* New York: McGraw-Hill, 1982.

PROBLEMS

Section 3.2 Problems

3.1. ***(a)** Letting $x_1(t) = y(t)$ and $x_2(t) = \dot{y}(t)$, write the state equations for the system described in the differential equation. Express these equations in matrix form.

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

***(b)** Draw a simulation diagram for the system in (a).

(c) Write the state equations for the system described by the differential equation

$$\dot{y}(t) + 3y(t) = 2u(t)$$

(d) Draw both the control-canonical and the observer-canonical simulation diagrams.

3.2. **(a)** Letting $x_1(t) = y(t)$, $x_2(t) = \dot{y}(t)$, and $x_3(t) = \ddot{y}(t)$, write the state equations for the system described in the differential equation. Express these equations in matrix form.

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

(b) Draw a simulation diagram for the system in (a).

(c) Draw both the control-canonical and the observer-canonical simulation diagrams.

3.3. ***(a)** Draw a simulation diagram of the system described by the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{7}{s^2 + 9s + 8}$$

(b) From the simulation diagram, write a set of state equations for the system of (a).

3.4. **(a)** Draw a simulation diagram of the system described by the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{5}{s^2 + 7s + 10}$$

(b) Write state equations for the system in (a).

(c) Repeat (a) and (b) for the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{5}{s - 4}$$

(d) Repeat (a) and (b) for the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{2s}{s^3 + 1}$$

3.5. For the general nth-order system with the transfer function

$$G(s) = \frac{b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}$$

(a) Write the state equations for this system in control canonical form.
(b) Write the state equations for this system in observer canonical form.

Section 3.3 Problems.

***3.6.** **(a)** Consider the closed-loop control system in Figure P3.6(a), and the simulation diagram in part (b) of the figure. Write the state equations for the control system, from the simulation diagram.
(b) Using the state equations found in (a), derive the resolvant of the matrix A.
(c) Using the state equations found in (a), derive the state transition matrix of the system.
(d) Modify the MATLAB program of Example 3.4 to verify results of (b) and (c).

3.7. **(a)** Consider the closed-loop control system in Figure P3.6(a), and the simulation diagram in part (c) of the figure. Write the state equations for the control system, from the simulation diagram.

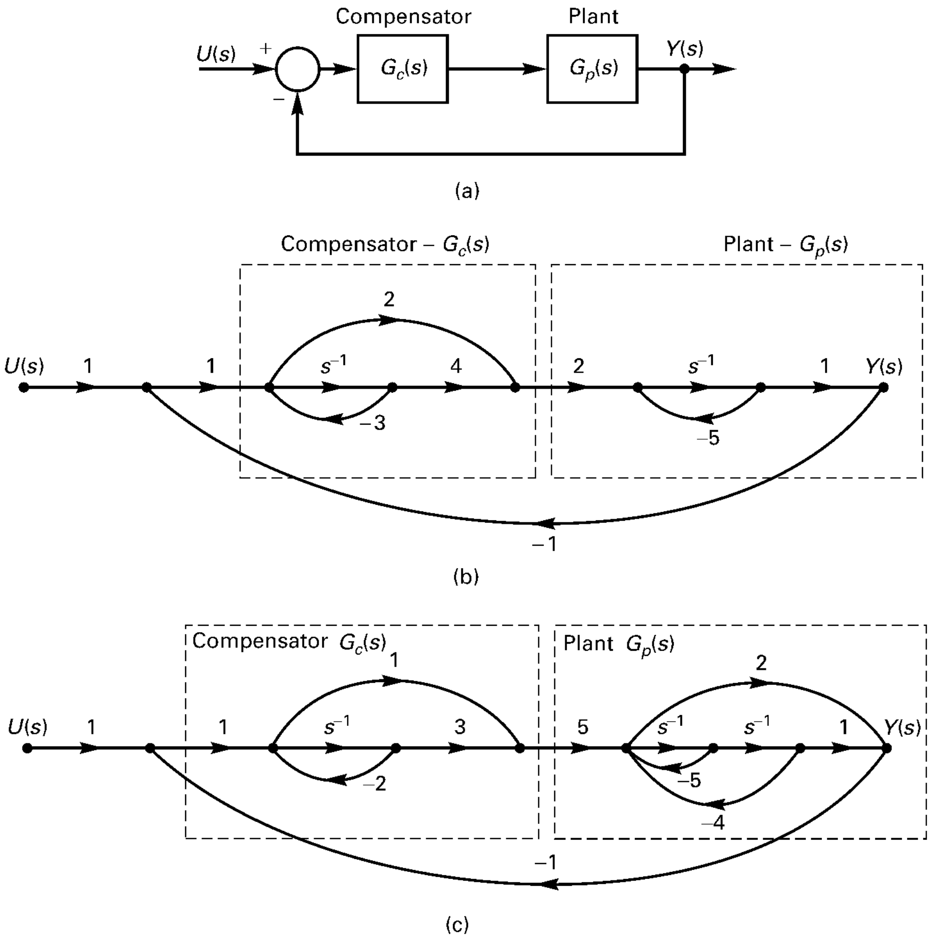


FIGURE P3.6

(b) Using the state equations found in (a), derive the resolvant of the matrix A.
(c) Using the state equations found in (a), derive the state transition matrix of the system.
(d) Modify the MATLAB program of Example 3.4 to verify results of (b) and (c).

3.8. A first-order system is modeled by the state equations

$$\dot{x}(t) = -3x(t) + 4u(t)$$
$$y(t) = x(t)$$

(a) Find the Laplace transform of the state transition matrix.
(b) Find the state transition matrix.
(c) If the input $u(t)$ is a unit step function, with $x(0) = 0$, find $y(t), t > 0$, using (3-24).
(d) If the input $u(t)$ is a unit step function, with $x(0) = -1$, find $y(t), t > 0$, using (3-24). The results of (b) and (c) are useful.
(e) Verify the results of (c), using the transfer function approach.
(f) Verify the results in (d), using the Laplace transform of the state equation.
(g) Use MATLAB to verify the results in (d).

3.9. Repeat all parts of Problem 3.8 for the first-order system

$$\dot{x}(t) = -x(t) + u(t)$$
$$y(t) = x(t) + u(t)$$

3.10. Given the state equations

$$\dot{x}(t) = \begin{bmatrix} 0 & 2 \\ -2 & -5 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

***(a)** Find the Laplace transform of the state transition matrix.
***(b)** Find the state transition matrix.
(c) If the input $u(t)$ is a unit step function, with, $x_1(0) = x_2(0) = 0$, find, $y(t), t \geq 0$ using (3-24).
(d) If the input $u(t)$ is a unit step function, with $x_1(0) = 1, x_2(0) = 0$, find, $x(t), t \geq 0$ the results of (c) can be useful.
(e) Verify the calculations of $x_1(t)$ in (c), using a transfer function approach.
(f) Modify the MATLAB program of Example 3.5 to verify the results in (d).

3.11. Given the state equations

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -3 & -6 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 0 & 9 \end{bmatrix} x(t)$$

(a) Find the resolvant matrix.
(b) Find the state transition matrix.
(c) If the input $u(t)$ is a unit step function, with $x_1(0) = 0$ and $x_2(0) = 0$, find $x(t), t \geq 0$, using (3-24).
(d) If the input $u(t)$ is a unit step function, with $x_1(0) = 1, x_2(0) = -1$, find $y(t), t \geq 0$. The results of (c) are useful.
(e) Verify the results of (c), using a transfer function approach.
(f) Modify the MATLAB program of Example 3.5 to verify the results in (d).

Section 3.4 Problems

3.12. ***(a)** Consider the closed-loop control system in Figure P3.6(a), and its simulation diagram in part (b) of the figure. Write the state equations for the control system, from the simulation diagram.

***(b)** Find $G_c(s)$, the compensator transfer function, and $G_p(s)$, the plant transfer function, directly from the simulation diagram.

***(c)** Find the closed-loop transfer function $Y(s)/U(s)$.

(d) Show that the denominator of the closed-loop transfer function is equal to det(sI-A), using A from (a).

(e) Verify the results of (d) by finding the Δ of Mason's gain formula. Note that in this problem the denominator of the closed-loop transfer function has been calculated by three different procedures.

(f) Modify the MATLAB program of Example 3.8 to verify results of (d).

3.13. (a) Consider the closed-loop control system in Figure P3.6(a), and its simulation diagram in part (c) of the figure. Write the state equations for the control system, from the simulation diagram.

(b) Find $G_c(s)$, the compensator transfer function, and $G_p(s)$, the plant transfer function, directly from the simulation diagram.

(c) Find the closed-loop transfer function $Y(s)/U(s)$.

(d) Show that the denominator of the closed-loop transfer function is equal to det(sI-A), using A from (a).

(e) Verify the results of (d) by finding the Δ of Mason's gain formula. Note that in this problem the denominator of the closed-loop transfer function has been calculated by three different procedures.

(f) Modify the MATLAB program of Example 3.8 to verify results of (d).

(g) Repeat (a)–(d) for the case that the compensator is replaced by a constant gain K.

3.14. ***(a)** Draw a simulation diagram of the system described by the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{10s + 5}{s^2 + 5s + 6}$$

***(b)** From the simulation diagram, write a set of state equations.

(c) Use Mason's gain formula to verify that the simulation diagram matches the transfer function.

3.15. (a) Draw a simulation diagram of the system described by the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{10s}{(s + 2)(s + 5)}$$

(b) From the simulation diagram, write a set of state equations.

(c) Use Mason's gain formula to verify that the simulation diagram matches the transfer function.

(d) Repeat (a)–(c) for the transfer function

$$\frac{Y(s)}{U(s)} = G(s) = \frac{s^2 + 5s + 4}{s^2 + 7s + 10}$$

(e) For (a), (c), and (d), modify the MATLAB program of Example 3.8 to verify that the state equations match the transfer functions.

3.16. The dc motor described in Example 2.14 is to be used in a position control system as shown in Figure 2.27.

(a) Choosing the state variables to be $x_1 = i_{a,} x_2 = \theta, x_3 = \frac{d\theta}{dt}$, and $y(t) = \theta$ write the state equations for the motor.

(b) Draw a simulation diagram for (a). Note that because the states are defined using the laws of physics, the state equations are in neither the control canonical or observer canonical form.

(c) Using Mason's gain formula and the simulation diagram from (b), verify the transfer function $\frac{\Theta(s)}{E_a(s)}$.

(d) Modify the MATLAB program of Example 3.8 to verify that the state equations match the transfer function.

3.17. The equations for the circuit of Figure P3.17 are developed in Example 2.2.

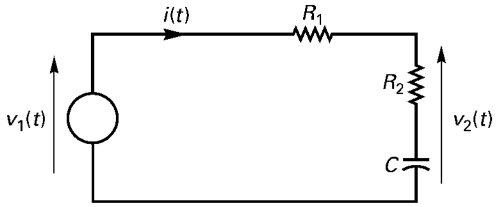


FIGURE P3.17

(a) Express these equations in state-variable format using the variables

$$x(t) = q(t) = \int i(t)dt, \text{ and } y(t) = v_2(t).$$

(b) Draw a simulation diagram for these state equations.

(c) Verify the transfer function calculated in Example 2.3, using (3-42).

(d) Verify the transfer function in (c) using Mason's gain formula.

3.18. Consider the rigid satellite of Figure P3.18, which was discussed in Example 2.13 of Section 2.6. The thrusters develop a torque τ, and the resultant rotation is modeled by

$$\tau(t) = J\frac{d^2\theta(t)}{dt^2}$$

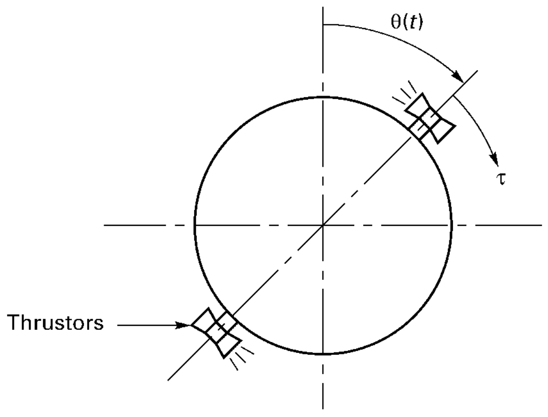


FIGURE P3.18

***(a)** Develop a state model for this system, with the torque τ as the input and the angle $\theta(t)$ as the output. Use $\theta(t)$ and $\theta(t)$ as the states.

***(b)** Use (3-42) to derive the transfer function of the system.

***(c)** Letting $J = 1$, use MATLAB to verify the transfer function in (b).

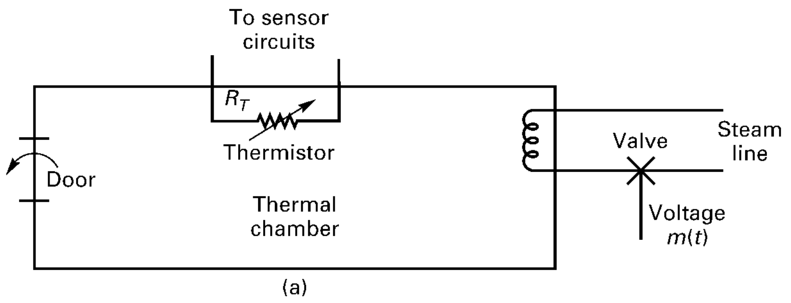
3.19. A thermal test chamber is illustrated in Figure P3.19(a). This chamber, which is a large room, is used to test large devices under various thermal stresses. The chamber is heated with steam, which is controlled by an electrically activated valve. The temperature of the chamber is measured by a sensor based on a thermistor, which is a semiconductor resistor whose resistance varies with temperature. Opening the door of the chamber affects the temperature of the chamber and hence must be considered as a disturbance.

A model of the thermal chamber is shown in Figure P3.19(b). The control input is the voltage $m(t)$, which controls the valve in the steam line, as shown in the figure. A step function $d(t) = 4u(t)$ is used to model the opening of the chamber door. With the door closed, $d(t) = 0$.

(a) Find a second-order state model of the system with two inputs, $m(t)$ and $d(t)$, and one output, $c(t)$.

(b) Find a first-order state model of the system with the same inputs and outputs as in (a). This model is possible because the characteristic equations are identical for both blocks in Figure P3.19(b).

(c) Find the transfer function matrix $\boldsymbol{G}(s)$, where $C(s) = \boldsymbol{G}(s)\boldsymbol{U}(s)$ and where $\boldsymbol{G}(s)$ is of order 1×2 and $\boldsymbol{U}(s) = [M(s)\ D(s)]^T$.



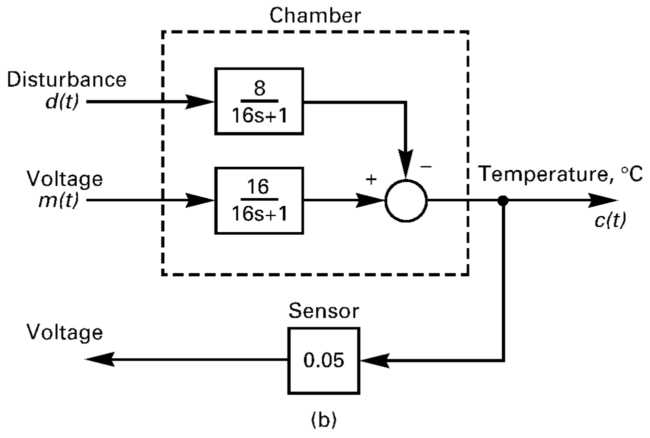
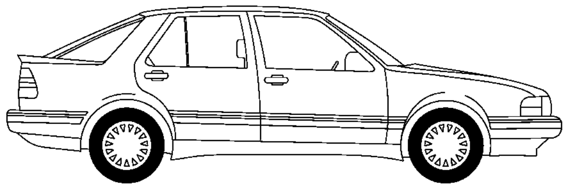


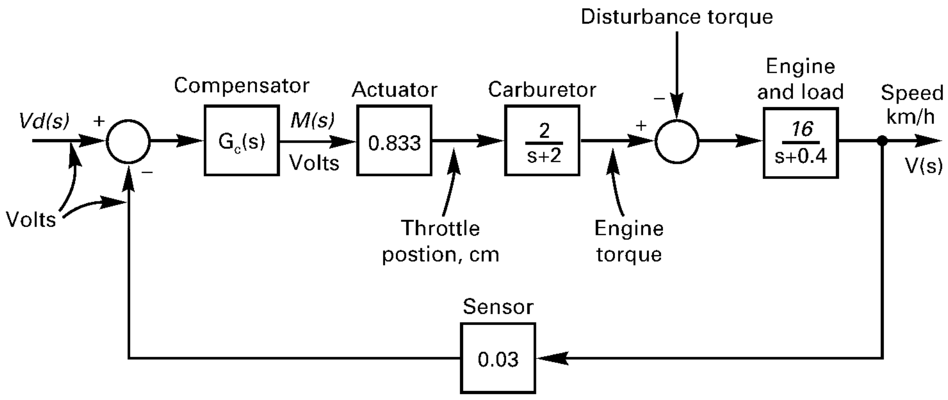
FIGURE P3.19

3.20. Figure P3.20 is the block diagram of a simplified cruise-control system for an automobile. The actuator controls the throttle position, the carburetor is modeled as a first-order lag with a 0.5-second time constant, and the engine and load is also modeled as a first-order lag with a 2.5-second time constant. For this problem, assume that the disturbance is zero.

(a) Draw a simulation diagram for the plant using $M(s)$ as the input and $V(s)$ as the output.

(b) Find a state model for the plant in (a), with one state associated with the engine torque and the another state associated with speed.


(a)



(b)

FIGURE P3.20

(c) Let $G_c(s) = 1$. Find the state model of the closed-loop system such that the states are the same as those in (b).

(d) Use Mason's gain formula to find the closed-loop transfer function $\dfrac{V(s)}{V_d(s)}$

(e) Verify the results in (d) using MATLAB and the state model.

3.21. Consider the mechanical translational system shown in Figure P3.21(a). All parameters are in consistent units.

***(a)** Write a set of state equations for this system.

***(b)** From these equations, construct a simulation diagram for the system.

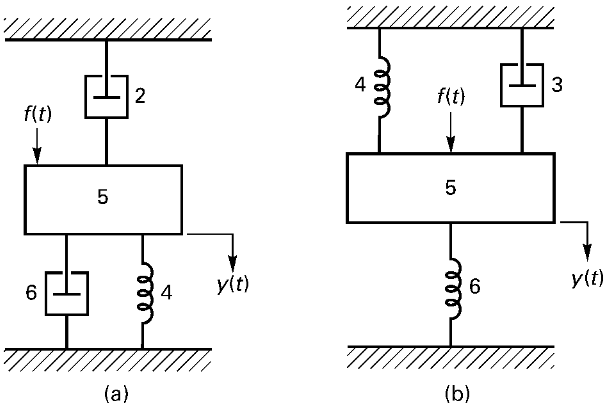


FIGURE P3.21

***(c)** Using Mason's gain formula and the simulation diagram of (b), find the transfer function $Y(s)/F(s)$.

(d) Verify the results of (c) using MATLAB and the state equations of (a).

3.22. Consider the mechanical translational system shown in Figure P3.21(b). All parameters are in consistent units.

(a) Write a set of state equations for this system.

(b) From these equations, construct a simulation diagram for the system.

(c) Using Mason's gain formula and the simulation diagram of (b), find the transfer function $Y(s)/F(s)$.

(d) Verify the results of (c) using MATLAB and the state equations of (a).

3.23. Use the state equations and (3-42), verify the transfer functions found in

(a) Problem 3.13

(b) Problem 3.14

3.24. For the satellite of Problem 3.18 with J = 0.1, a state model is given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 10 \end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t)$$

With $u(t) = \tau(t)$ and $y(t) = \theta(t)$.

(a) Find the resolvant matrix.

(b) Find the state transition matrix.

(c) If the input $u(t)$ is a unit step function, with $\boldsymbol{x}(0) = 0$, find $\boldsymbol{x}(t)$, $t > 0$, using (3-24).

(d) Verify the results of (c), using the transfer function approach.

(e) Modify the MATLAB program of Example 3.8 to verify the results in (d).

3.25. Given the state equations $\dot{\mathbf{x}}(t) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t)$ and $y(t) = \mathbf{C}\boldsymbol{x}(t) + \mathbf{D}\boldsymbol{u}(t)$, derive the transfer function $G(s) = Y(s)/U(s)$

Section 3.5 Problems

3.26. Consider the rigid satellite of Figure P3.18, which was discussed in Example 2.13 of Section 2.6. The thrusters develop a torque, $\tau(t)$, and the resultant rotation is modeled by $\tau(t) = Jd^2\theta(t)/dt^2$

***(a)** Develop a state model for this system, with the torque τ as the input and the angle θ as the output. Use $\theta(t)$ and $\dot{\theta}(t)$ as the states.

***(b)** Draw a simulation diagram for the system.

***(c)** A similarity transformation is defined as [see (3-49)]

$$\dot{\mathbf{x}}(t) = \mathbf{P}v(t) = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix}\mathbf{v}(t)$$

Express the state model in terms of the states $\mathbf{v}(t)$.

(d) Draw a simulation diagram for the state equations in (c), and use Mason's gain formula to verify the transfer function.

(e) Modify the simulation diagram of (b) to show the states $\mathbf{v}(t)$, as in Figure 3.11. Compare the transfer functions from the input $\tau(t)$ to the states, $v_1(t)$ and $v_2(t)$, for the two simulation diagrams of (d) and (e).

(f) Simulate the two state models of (a) and (c) with unit step inputs, with $J = 1$. The two responses will be equal, indicating that the two models have the same input–output characteristics.

(g) With $J= 1$, verify the results in (c) using MATLAB.

3.27. For the two state models of Problem 3.26, show that:

(a) $|s\mathbf{I} - \mathbf{A}| = |s\mathbf{I} - \mathbf{A}_v|$

(b) $|\mathbf{A}| = |\mathbf{A}_v|$

(c) $\operatorname{tr}\mathbf{A} = \operatorname{tr}\mathbf{A}_v$.

(d) $\mathbf{C}[s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B} = \mathbf{C}_v[s\mathbf{I} - \mathbf{A}_v]^{-1}\mathbf{B}_v$

3.28. Consider the system

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ -5 & -7 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix} 2 & 1 \end{bmatrix}\mathbf{x}(t)$$

A similarity transformation is defined as [see (3-49)]

$$\mathbf{x}(t) = \mathbf{P}\mathbf{v}(t) = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}\mathbf{v}(t)$$

(a) Express the state model in terms of the states $\mathbf{v}(t)$.

(b) Draw a simulation diagram for the state model in $\mathbf{x}(t)$ and one for the state model in $\mathbf{v}(t)$.

(c) Show by Mason's gain formula that the transfer functions of the two simulation diagrams in (b) are equal.

(d) Simulate the two state models with unit step inputs using MATLAB. The two responses will be equal, indicating that the two models have the same input–output characteristics.

3.29. For the two state models of Problem 3.28, show that:

(a) $|s\mathbf{I} - \mathbf{A}| = |s\mathbf{I} - \mathbf{A}_v|$

(b) $|\mathbf{A}| = |\mathbf{A}_v|$

(c) $\operatorname{tr}\mathbf{A} = \operatorname{tr}\mathbf{A}_v$

(d) $\mathbf{C}[s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B} = \mathbf{C}_v[s\mathbf{I} - \mathbf{A}_v]^{-1}\mathbf{B}_v$

3.30. Show that $\mathbf{C}[s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B} = \mathbf{C}_v[s\mathbf{I} - \mathbf{A}_v]^{-1}\mathbf{B}_v$, where the various matrices are related by $\mathbf{A}_v = \mathbf{P}^{-1}\mathbf{A}\mathbf{P}$, $\mathbf{B}_v = \mathbf{P}^{-1}\mathbf{B}$ and $\mathbf{C}_v = \mathbf{C}\mathbf{P}$

Section 3.6 Problems

3.31. Use a MATLAB simulation to display the unit step responses of the systems of

(a) Problem 3.8

(b) Problem 3.9

(c) Problem 3.10

(d) Problem 3.11

(e) Problem 3.14

(f) Problem 3.15