1

INTRODUCTION

This chapter introduces the state-space representation for linear time-invariant systems. We begin with a brief overview of the origins of state-space methods to provide a context for the focus of this book. Following that, we define the state equation format and provide examples to show how state equations can be derived from physical system descriptions and from transfer-function representations. In addition, we show how linear state equations arise from the linearization of a nonlinear state equation about a nominal trajectory or equilibrium condition.

This chapter also initiates our use of the MATLAB software package for computer-aided analysis and design of linear state-space control systems. Beginning here and continuing throughout the book, features of MATLAB and the accompanying Control Systems Toolbox that support each chapter's subject matter will be presented and illustrated using a Continuing MATLAB Example. In addition, we introduce two Continuing Examples that we also will revisit in subsequent chapters.

1.1 HISTORICAL PERSPECTIVE AND SCOPE

Any scholarly account of the history of control engineering would have to span several millennia because there are many examples throughout

Linear State-Space Control Systems, by Robert L. Williams II and Douglas A. Lawrence

ancient history, the industrial revolution, and into the early twentieth century of ingeniously designed systems that employed feedback mechanisms in various forms. Ancient water clocks, south-pointing chariots, Watt's flyball governor for steam engine speed regulation, and mechanisms for ship steering, gun pointing, and vacuum tube amplifier stabilization are but a few. Here we are content to survey important developments in the theory and practice of control engineering since the mid-1900s in order to provide some perspective for the material that is the focus of this book in relation to topics covered in most undergraduate controls courses and in more advanced graduate-level courses.

In the so-called classical control era of the 1940s and 1950s, systems were represented in the frequency domain by transfer functions. In addition, performance and robustness specifications were either cast directly in or translated into the frequency domain. For example, transient response specifications were converted into desired closed-loop pole locations or desired open-loop and/or closed-loop frequency-response characteristics. Analysis techniques involving Evans root locus plots, Bode plots, Nyquist plots, and Nichol's charts were limited primarily to single-input, single-output systems, and compensation schemes were fairly simple, e.g., a single feedback loop with cascade compensation. Moreover, the design process was iterative, involving an initial design based on various simplifying assumptions followed by parameter tuning on a trial-and-error basis. Ultimately, the final design was not guaranteed to be optimal in any sense.

The 1960s and 1970s witnessed a fundamental paradigm shift from the frequency domain to the time domain. Systems were represented in the time domain by a type of differential equation called a *state equation*. Performance and robustness specifications also were specified in the time domain, often in the form of a quadratic performance index. Key advantages of the state-space approach were that a time-domain formulation exploited the advances in digital computer technology and the analysis and design methods were well-suited to multiple-input, multiple-output systems. Moreover, feedback control laws were calculated using analytical formulas, often directly optimizing a particular performance index.

The 1980's and 1990's were characterized by a merging of frequency-domain and time-domain viewpoints. Specifically, frequency-domain performance and robustness specifications once again were favored, coupled with important theoretical breakthroughs that yielded tools for handling multiple-input, multiple-output systems in the frequency domain. Further advances yielded state-space time-domain techniques for controller synthesis. In the end, the best features of the preceding decades were merged into a powerful, unified framework.

The chronological development summarized in the preceding paragraphs correlates with traditional controls textbooks and academic curricula as follows. Classical control typically is the focus at the undergraduate level, perhaps along with an introduction to state-space methods. An in-depth exposure to the state-space approach then follows at the advanced undergraduate/first-year graduate level and is the focus of this book. This, in turn, serves as the foundation for more advanced treatments reflecting recent developments in control theory, including those alluded to in the preceding paragraph, as well as extensions to time-varying and nonlinear systems.

We assume that the reader is familiar with the traditional undergraduate treatment of linear systems that introduces basic system properties such as system dimension, causality, linearity, and time invariance. This book is concerned with the analysis, simulation, and control of finite-dimensional, causal, linear, time-invariant, continuous-time dynamic systems using state-space techniques. From now on, we will refer to members of this system class as *linear time-invariant systems*.

The techniques developed in this book are applicable to various types of engineering (even nonengineering) systems, such as aerospace, mechanical, electrical, electromechanical, fluid, thermal, biological, and economic systems. This is so because such systems can be modeled mathematically by the same types of governing equations. We do not formally address the modeling issue in this book, and the point of departure is a linear time-invariant state-equation model of the physical system under study. With mathematics as the unifying language, the fundamental results and methods presented here are amenable to translation into the application domain of interest.

1.2 STATE EQUATIONS

A state-space representation for a linear time-invariant system has the general form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad x(t_0) = x_0 \tag{1.1}$$

in which $x(t)$ is the n-dimensional *state vector*

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

whose n scalar components are called *state variables*. Similarly, the m-dimensional *input vector* and p-dimensional *output vector* are given, respectively, as

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \qquad y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{bmatrix}$$

Since differentiation with respect to time of a time-varying vector quantity is performed component-wise, the time-derivative on the left-hand side of Equation (1.1) represents

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix}$$

Finally, for a specified initial time t_0, the *initial state* $x(t_0) = x_0$ is a specified, constant n-dimensional vector.

The state vector $x(t)$ is composed of a minimum set of system variables that uniquely describes the future response of the system given the current state, the input, and the dynamic equations. The input vector $u(t)$ contains variables used to actuate the system, the output vector $y(t)$ contains the measurable quantities, and the state vector $x(t)$ contains internal system variables.

Using the notational convention $M = [m_{ij}]$ to represent the matrix whose element in the ith row and jth column is m_{ij}, the coefficient matrices in Equation (1.1) can be specified via

$$A = [a_{ij}] \quad B = [b_{ij}] \quad C = [c_{ij}] \quad D = [d_{ij}]$$

having dimensions $n \times n$, $n \times m$, $p \times n$, and $p \times m$, respectively. With these definitions in place, we see that the state equation (1.1) is a compact representation of n scalar first-order ordinary differential equations, that is,

$$\begin{aligned} \dot{x}_i(t) = {} & a_{i1}x_1(t) + a_{i2}x_2(t) + \cdots + a_{in}x_n(t) \\ & + b_{i1}u_1(t) + b_{i2}u_2(t) + \cdots + b_{im}u_m(t) \end{aligned}$$

for $i = 1, 2, \ldots, n$, together with p scalar linear algebraic equations

$$\begin{aligned} y_j(t) = {} & c_{j1}x_1(t) + c_{j2}x_2(t) + \cdots + c_{jn}x_n(t) \\ & + d_{j1}u_1(t) + d_{j2}u_2(t) + \cdots + d_{jm}u_m(t) \end{aligned}$$

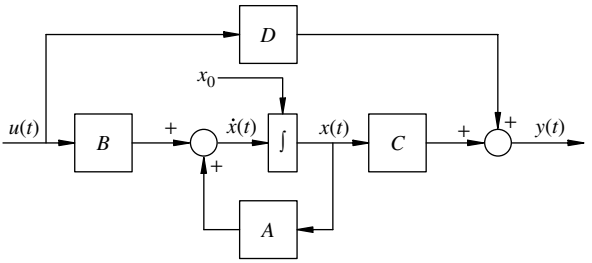

FIGURE 1.1 State-equation block diagram.

for $j = 1, 2, \ldots, p$. From this point on the vector notation (1.1) will be preferred over these scalar decompositions. The state-space description consists of the state differential equation $\dot{x}(t) = Ax(t) + Bu(t)$ and the algebraic output equation $y(t) = Cx(t) + Du(t)$ from Equation (1.1). Figure 1.1 shows the block diagram for the state-space representation of general multiple-input, multiple-output linear time-invariant systems.

One motivation for the state-space formulation is to convert a coupled system of higher-order ordinary differential equations, for example, those representing the dynamics of a mechanical system, to a coupled set of first-order differential equations. In the single-input, single-output case, the state-space representation converts a single nth-order differential equation into a system of n coupled first-order differential equations. In the multiple-input, multiple-output case, in which all equations are of the same order n, one can convert the system of k nth-order differential equations into a system of kn coupled first-order differential equations.

1.3 EXAMPLES

In this section we present a series of examples that illustrate the construction of linear state equations. The first four examples begin with first-principles modeling of physical systems. In each case we adopt the strategy of associating state variables with the energy storage elements in the system. This facilitates derivation of the required differential and algebraic equations in the state-equation format. The last two examples begin with transfer-function descriptions, hence establishing a link between transfer functions and state equations that will be pursued in greater detail in later chapters.

Example 1.1 Given the linear single-input, single-output, mass-spring-damper translational mechanical system of Figure 1.2, we now derive the

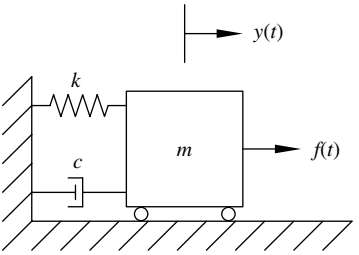

FIGURE 1.2 Translational mechanical system.

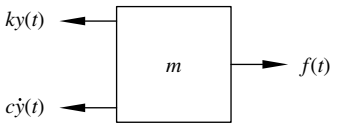

FIGURE 1.3 Free-body diagram.

system model and then convert it to a state-space description. For this system, the input is force $f(t)$ and the output is displacement $y(t)$.

Using Newton's second law, the dynamic force balance for the free-body diagram of Figure 1.3 yields the following second-order ordinary differential equation

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

that models the system behavior. Because this is a single second-order differential equation, we need to select a 2×1 state vector. In general, energy storage is a good criterion for choosing the state variables. The total system energy at any time is composed of potential spring energy $ky(t)^2/2$ plus kinetic energy $m\dot{y}(t)^2/2$ associated with the mass displacement and velocity. We then choose to define the state variables as the mass displacement and velocity:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \quad \begin{aligned} x_1(t) &= y(t) \\ x_2(t) &= \dot{y}(t) = \dot{x}_1(t) \end{aligned}$$

Therefore,

$$\dot{y}(t) = x_2(t)$$

$$\ddot{y}(t) = \dot{x}_2(t)$$

Substituting these two state definitions into the original system equation gives

$$m\dot{x}_2(t) + cx_2(t) + kx_1(t) = f(t)$$

The original single second-order differential equation can be written as a coupled system of two first-order differential equations, that is,

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

The output is the mass displacement

$$y(t) = x_1(t)$$

The generic variable name for input vectors is $u(t)$, so we define:

$$u(t) = f(t)$$

We now write the preceding equations in matrix-vector form to get a valid state-space description. The general state-space description consists of the state differential equation and the algebraic output equation. For Example 1.1, these are

State Differential Equation

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

Algebraic Output Equation

$$y(t) = Cx(t) + Du(t)$$
$$y(t) = [\,1 \quad 0\,] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + [0]u(t)$$

The two-dimensional single-input, single-output system matrices in this example are (with $m = p = 1$ and $n = 2$):

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \qquad C = [\,1 \quad 0\,]$$
$$D = 0$$

In this example, the state vector is composed of the position and velocity of the mass m. Two states are required because we started with one second-order differential equation. Note that $D = 0$ in this example because no part of the input force is directly coupled to the output. □

Example 1.2 Consider the parallel electrical circuit shown in Figure 1.4. We take the input to be the current produced by the independent current source $u(t) = i(t)$ and the output to be the capacitor voltage $y(t) = v(t)$.

It is often convenient to associate state variables with the energy storage elements in the network, namely, the capacitors and inductors. Specifically, capacitor voltages and inductor currents, while not only directly characterizing the energy stored in the associated circuit element, also facilitate the derivation of the required differential equations. In this example, the capacitor voltage coincides with the voltage across each circuit element as a result of the parallel configuration.

This leads to the choice of state variables, that is,

$$x_1(t) = i_L(t)$$
$$x_2(t) = v(t)$$

In terms of these state variables, the inductor's voltage-current relationship is given by

$$x_2(t) = L\dot{x}_1(t)$$

Next, Kirchhoff's current law applied to the top node produces

$$\frac{1}{R}x_2(t) + x_1(t) + C\dot{x}_2(t) = u(t)$$

These relationships can be rearranged so as to isolate state-variable time derivatives as follows:

$$\dot{x}_1(t) = \frac{1}{L}x_2(t)$$
$$\dot{x}_2(t) = -\frac{1}{C}x_1(t) - \frac{1}{RC}x_2(t) + \frac{1}{C}u(t)$$

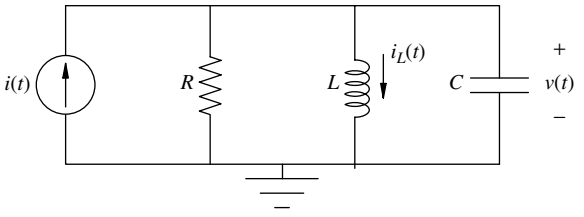

FIGURE 1.4 Parallel electrical circuit.

This pair of coupled first-order differential equations, along with the output definition $y(t) = x_2(t)$, yields the following state-space description for this electrical circuit:

State Differential Equation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{C} \end{bmatrix} u(t)$$

Algebraic Output Equation

$$y(t) = [\,0 \quad 1\,] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + [0]u(t)$$

from which the coefficient matrices A, B, C, and D are found by inspection, that is,

$$A = \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{1}{C} \end{bmatrix} \qquad C = [\,0 \quad 1\,]$$

$$D = 0$$

Note that $D = 0$ in this example because there is no direct coupling between the current source and the capacitor voltage. □

Example 1.3 Consider the translational mechanical system shown in Figure 1.5, in which $y_1(t)$ and $y_2(t)$ denote the displacement of the associated mass from its static equilibrium position, and $f(t)$ represents a force applied to the first mass m_1. The parameters are masses m_1 and m_2, viscous damping coefficient c, and spring stiffnesses k_1 and k_2. The input is the applied force $u(t) = f(t)$, and the outputs are taken as the mass displacements. We now derive a mathematical system model and then determine a valid state-space representation.

Newton's second law applied to each mass yields the coupled second-order differential equations, that is,

$$m_1\ddot{y}_1(t) + k_1 y_1(t) - k_2[y_2(t) - y_1(t)] = f(t)$$

$$m_2\ddot{y}_2(t) + c\dot{y}_2(t) + k_2[y_2(t) - y_1(t)] = 0$$

Here, the energy-storage elements are the two springs and the two masses. Defining state variables in terms of mass displacements and velocities

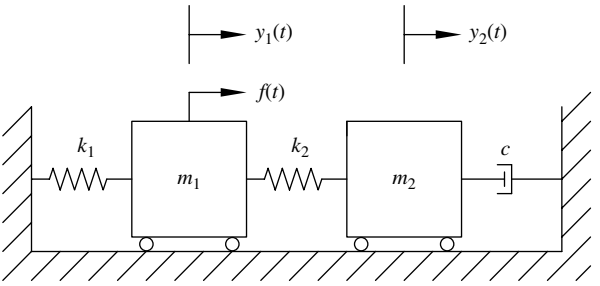

FIGURE 1.5 Translational mechanical system.

yields

$$x_1(t) = y_1(t)$$
$$x_2(t) = y_2(t) - y_1(t)$$
$$x_3(t) = \dot{y}_1(t)$$
$$x_4(t) = \dot{y}_2(t)$$

Straightforward algebra yields the following state equation representation:

State Differential Equation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -\dfrac{k_1}{m_1} & \dfrac{k_2}{m_1} & 0 & 0 \\ 0 & \dfrac{-k_2}{m_2} & 0 & -\dfrac{c}{m_2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{1}{m_1} \\ 0 \end{bmatrix} u(t)$$

Algebraic Output Equation

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(t)$$

from which the coefficient matrices A, B, C, and D can be identified. Note that $D = [\,0 \quad 0\,]^{\mathrm{T}}$ because there is no direct feedthrough from the input to the output.

Now, it was convenient earlier to define the second state variable as the difference in mass displacements, $x_2(t) = y_2(t) - y_1(t)$, because this relative displacement is the amount the second spring is stretched. Instead

we could have defined the second state variable based on the absolute mass displacement, that is $x_2(t) = y_2(t)$, and derived an equally valid state-space representation. Making this one change in our state variable definitions, that is,

$$x_1(t) = y_1(t)$$
$$x_2(t) = y_2(t)$$
$$x_3(t) = \dot{y}_1(t)$$
$$x_4(t) = \dot{y}_2(t)$$

yields the new A and C matrices

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{(k_1 + k_2)}{m_1} & \dfrac{k_2}{m_1} & 0 & 0 \\ \dfrac{k_2}{m_2} & \dfrac{-k_2}{m_2} & 0 & -\dfrac{c}{m_2} \end{bmatrix}$$

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

The B and D matrices are unchanged. □

Example 1.4 Consider the electrical network shown in Figure 1.6. We now derive the mathematical model and then determine a valid state-space representation. The two inputs are the independent voltage and current sources $v_{\text{in}}(t)$ and $i_{\text{in}}(t)$, and the single output is the inductor voltage $v_{\text{L}}(t)$.

In terms of clockwise circulating mesh currents $i_1(t)$, $i_2(t)$, and $i_3(t)$, Kirchhoff's voltage law applied around the leftmost two meshes yields

$$R_1 i_1(t) + v_{\text{C}_1}(t) + L\frac{d}{dt}[i_1(t) - i_2(t)] = v_{\text{in}}(t)$$

$$L\frac{d}{dt}[i_2(t) - i_1(t)] + v_{\text{C}_2}(t) + R_2[i_2(t) - i_3(t)] = 0$$

and Kirchhoff's current law applied to the rightmost mesh yields

$$i_3(t) = -i_{\text{in}}(t)$$

In addition, Kirchhoff's current law applied at the top node of the inductor gives

$$i_{\text{L}}(t) = i_1(t) - i_2(t)$$

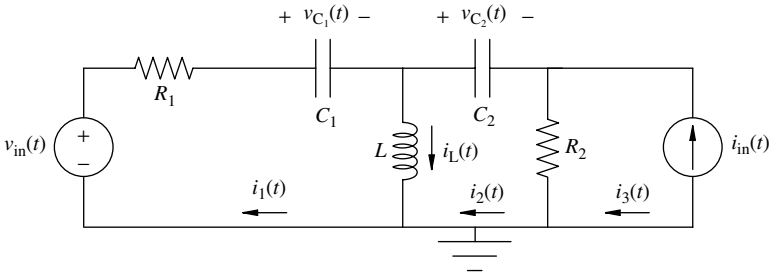

FIGURE 1.6 Electrical circuit.

As in Example 1.2, it is again convenient to associate state variables with the capacitor and inductor energy-storage elements in the network. Here, we select

$$x_1(t) = v_{C_1}(t)$$
$$x_2(t) = v_{C_2}(t)$$
$$x_3(t) = i_L(t)$$

We also associate inputs with the independent sources via

$$u_1(t) = v_{in}(t)$$
$$u_2(t) = i_{in}(t)$$

and designate the inductor voltage $v_L(t)$ as the output so that

$$y(t) = v_L(t) = L\dot{x}_3(t)$$

Using the relationships

$$C_1\dot{x}_1(t) = i_1(t)$$
$$C_2\dot{x}_2(t) = i_2(t)$$
$$x_3(t) = C_1\dot{x}_1(t) - C_2\dot{x}_2(t)$$

the preceding circuit analysis now can be recast as

$$R_1C_1\dot{x}_1(t) + L\dot{x}_3(t) = -x_1(t) + u_1(t)$$
$$R_2C_2\dot{x}_2(t) - L\dot{x}_3(t) = -x_2(t) - R_2u_2(t)$$
$$C_1\dot{x}_1(t) - C_2\dot{x}_2(t) = x_3(t)$$

Packaging these equations in matrix form and isolating the state-variable time derivatives gives

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} R_1C_1 & 0 & L \\ 0 & R_2C_2 & -L \\ C_1 & -C_2 & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -R_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \right)$$

Calculating and multiplying through by the inverse and yields the state differential equation, that is,

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} \dfrac{1}{(R_1+R_2)C_1} & \dfrac{-1}{(R_1+R_2)C_1} & \dfrac{R_2}{(R_1+R_2)C_1} \\ \dfrac{-1}{(R_1+R_2)C_2} & \dfrac{-1}{(R_1+R_2)C_2} & \dfrac{-R_1}{(R_1+R_2)C_2} \\ \dfrac{-R_2}{(R_1+R_2)L} & \dfrac{R_1}{(R_1+R_2)L} & \dfrac{-R_1R_2}{(R_1+R_2)L} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} \dfrac{1}{(R_1+R_2)C_1} & \dfrac{-R_2}{(R_1+R_2)C_1} \\ \dfrac{1}{(R_1+R_2)C_2} & \dfrac{-R_2}{(R_1+R_2)C_2} \\ \dfrac{R_2}{(R_1+R_2)L} & \dfrac{R_1R_2}{(R_1+R_2)L} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

which is in the required format from which coefficient matrices A and B can be identified. In addition, the associated output equation $y(t) = L\dot{x}_3(t)$ can be expanded to the algebraic output equation as follows

$$y(t) = \begin{bmatrix} \dfrac{-R_2}{(R_1+R_2)} & \dfrac{R_1}{(R_1+R_2)} & \dfrac{-R_1R_2}{(R_1+R_2)} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} \dfrac{R_2}{(R_1+R_2)} & \dfrac{R_1R_2}{(R_1+R_2)} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

from which coefficient matrices C and D can be identified.

Note that in this example, there is direct coupling between the independent voltage and current source inputs $v_{\text{in}}(t)$ and $i_{\text{in}}(t)$ and the inductor voltage output $v_{\text{L}}(t)$, and hence the coefficient matrix D is nonzero. □

Example 1.5 This example derives a valid state-space description for a general third-order differential equation of the form

$$\dddot{y}(t) + a_2\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_0 u(t)$$

The associated transfer function definition is

$$H(s) = \frac{b_0}{s^3 + a_2 s^2 + a_1 s + a_0}$$

Define the following state variables:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \quad \begin{aligned} x_1(t) &= y(t) \\ x_2(t) &= \dot{y}(t) = \dot{x}_1(t) \\ x_3(t) &= \ddot{y}(t) = \ddot{x}_1(t) = \dot{x}_2(t) \end{aligned}$$

Substituting these state-variable definitions into the original differential equation yields the following:

$$\dot{x}_3(t) = -a_0 x_1(t) - a_1 x_2(t) - a_2 x_3(t) + b_0 u(t)$$

The state differential and algebraic output equations are then

State Differential Equation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b_0 \end{bmatrix} u(t)$$

Algebraic Output Equation

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + [0]u(t)$$

from which the coefficient matrices A, B, C, and D can be identified. $D = 0$ in this example because there is no direct coupling between the input and output.

This example may be generalized easily to the nth-order ordinary differential equation

$$\frac{d^n y(t)}{dt^n} + a_{n-1}\frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_2\frac{d^2 y(t)}{dt^2} + a_1\frac{dy(t)}{dt} + a_0 y(t) = b_0 u(t) \tag{1.2}$$

For this case, the coefficient matrices A, B, C, and D are

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b_0 \end{bmatrix}$$

$$C = [\,1 \quad 0 \quad 0 \quad \cdots \quad 0\,] \qquad D = [0] \qquad (1.3)\ \square$$

Example 1.6 Consider a single-input, single-output system represented by the third-order transfer function with second-order numerator polynomial

$$H(s) = \frac{b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0}$$

If we attempted to proceed as in the preceding example in defining state variables in terms of the output $y(t)$ and its derivatives, we eventually would arrive at the relationship

$$\dot{x}_3(t) = -a_0 x_1(t) - a_1 x_2(t) - a_2 x_3(t) + b_2 \ddot{u}(t) + b_1 \dot{u}(t) + b_0 u(t)$$

This is not consistent with the state-equation format because of the presence of time derivatives of the input, so we are forced to pursue an alternate state-variable definition. We begin by factoring the transfer function according to $H(s) = H_2(s)H_1(s)$ with

$$H_1(s) = \frac{1}{s^3 + a_2 s^2 + a_1 s + a_0} \qquad H_2(s) = b_2 s^2 + b_1 s + b_0$$

and introducing an intermediate signal $w(t)$ with Laplace transform $W(s)$ so that

$$\begin{aligned} W(s) &= H_1(s)U(s) \\ &= \frac{1}{s^3 + a_2 s^2 + a_1 s + a_0} U(s) \\ Y(s) &= H_2(s)W(s) \\ &= (b_2 s^2 + b_1 s + b_0)W(s) \end{aligned}$$

A block-diagram interpretation of this step is shown in Figure 1.7. In the time domain, this corresponds to

$$\ddot{w}(t) + a_2 \ddot{w}(t) + a_1 \dot{w}(t) + a_0 w(t) = u(t)$$

$$y(t) = b_2 \ddot{w}(t) + b_1 \dot{w}(t) + b_0 w(t)$$

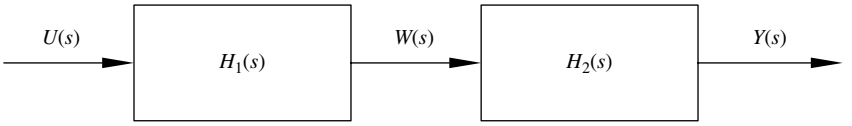

FIGURE 1.7 Cascade block diagram.

Now, the key observation is that a state equation describing the relationship between input $u(t)$ and output $w(t)$ can be written down using the approach of the preceding example. That is, in terms of state variables

$$\begin{aligned} x_1(t) &= w(t) \\ x_2(t) &= \dot{w}(t) = \dot{x}_1(t) \\ x_3(t) &= \ddot{w}(t) = \ddot{x}_1(t) = \dot{x}_2(t) \end{aligned}$$

we have

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$w(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + [0]u(t)$$

As the final step, we recognize that an equation relating the true system output $y(t)$ and our chosen state variables follows from

$$\begin{aligned} y(t) &= b_0 w(t) + b_1 \dot{w}(t) + b_2 \ddot{w}(t) \\ &= b_0 x_1(t) + b_1 x_2(t) + b_2 x_3(t) \end{aligned}$$

which gives the desired state equations:

State Differential Equation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

Algebraic Output Equation

$$y(t) = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + [0]u(t)$$

At this point, it should be clear how to extend this approach to systems of arbitrary dimension n beginning with a transfer function of the form

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

Notice that the numerator polynomial in $H(s)$ has degree strictly less than the denominator polynomial degree, so $H(s)$ is referred to as a strictly proper rational function (ratio of polynomials in the complex variable s). The preceding state-equation construction can be extended further to handle proper transfer functions

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

in which the numerator and denominator polynomial degrees are equal. The procedure involves first using polynomial division to write $H(s)$ as a strictly proper part plus a constant

$$H(s) = \frac{\hat{b}_{n-1}s^{n-1} + \cdots + \hat{b}_1 s + \hat{b}_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} + b_n$$

in which the reader may verify that $\hat{b}_i = b_i - b_n a_i$, for $i = 0, 1, \ldots, n-1$. Next, the coefficient matrices A, B, and C are found from the numerator and denominator polynomial coefficients of the strictly proper component and, in addition, $D = b_n$. □

In general, we say that a state equation is a *state-space realization* of a given system's input-output behavior if it corresponds to the relationship $Y(s) = H(s)U(s)$ in the Laplace domain or to the associated differential equation relating $y(t)$ and $u(t)$ in the time domain (for zero initial conditions). The exact meaning of *corresponds to* will be made precise in the next chapter. The preceding example serves to illustrate that a state-space realization of a single-input, single-output system can be written down by inspection simply by plugging the numerator and denominator coefficients into the correct locations in the coefficient matrices C and A, respectively. Owing to its special structure, this state equation is referred to as the *phase-variable canonical form* realization as well as the *controller canonical form* realization.

1.4 LINEARIZATION OF NONLINEAR SYSTEMS

Linear state equations also arise in the course of linearizing nonlinear state equations about nominal trajectories. We begin with a more general

nonlinear, time-varying state equation

$$\begin{aligned} \dot{x}(t) &= f[x(t), u(t), t] \\ y(t) &= h[x(t), u(t), t] \end{aligned} \qquad x(t_0) = x_0 \tag{1.4}$$

where $x(t), u(t)$, and $y(t)$ retain their default vector dimensions and $f(\cdot, \cdot, \cdot)$ and $h(\cdot, \cdot, \cdot)$ are continuously differentiable functions of their $(n + m + 1)$-dimensional arguments. Linearization is performed about a nominal trajectory defined as follows.

Definition 1.1 *For a nominal input signal, $\tilde{u}(t)$, the nominal state trajectory $\tilde{x}(t)$ satisfies*

$$\dot{\tilde{x}}(t) = f[\tilde{x}(t), \tilde{u}(t), t]$$

and the nominal output trajectory $\tilde{y}(t)$ satisfies

$$\tilde{y}(t) = h[\tilde{x}(t), \tilde{u}(t), t]$$

If $\tilde{u}(t) = \tilde{u}$, a constant vector, a special case is an equilibrium state $\tilde{x}$ that satisfies

$$0 = f(\tilde{x}, \tilde{u}, t)$$

for all t. □

Deviations of the state, input, and output from their nominal trajectories are denoted by δ subscripts via

$$\begin{aligned} x_\delta(t) &= x(t) - \tilde{x}(t) \\ u_\delta(t) &= u(t) - \tilde{u}(t) \\ y_\delta(t) &= y(t) - \tilde{y}(t) \end{aligned}$$

Using the compact notation

$$\begin{aligned} \frac{\partial f}{\partial x}(x, u, t) &= \left[\frac{\partial f_i}{\partial x_j}(x, u, t)\right] (n \times n) \\ \frac{\partial f}{\partial u}(x, u, t) &= \left[\frac{\partial f_i}{\partial u_j}(x, u, t)\right] (n \times m) \\ \frac{\partial h}{\partial x}(x, u, t) &= \left[\frac{\partial h_i}{\partial x_j}(x, u, t)\right] (p \times n) \\ \frac{\partial h}{\partial u}(x, u, t) &= \left[\frac{\partial h_i}{\partial u_j}(x, u, t)\right] (p \times m) \end{aligned}$$

and expanding the nonlinear maps in Equation (1.4) in a multivariate Taylor series about $[\tilde{x}(t), \tilde{u}(t), t]$ we obtain

$$\begin{aligned}
\dot{x}(t) &= f[x(t), u(t), t] \\
&= f[\tilde{x}(t), \tilde{u}(t), t] + \frac{\partial f}{\partial x}[\tilde{x}(t), \tilde{u}(t), t][x(t) - \tilde{x}(t)] \\
&\quad + \frac{\partial f}{\partial u}[\tilde{x}(t), \tilde{u}(t), t][u(t) - \tilde{u}(t)] + \text{higher-order terms} \\
y(t) &= h[x(t), u(t), t] \\
&= h[\tilde{x}(t), \tilde{u}(t), t] + \frac{\partial h}{\partial x}[\tilde{x}(t), \tilde{u}(t), t][x(t) - \tilde{x}(t)] \\
&\quad + \frac{\partial h}{\partial u}[\tilde{x}(t), \tilde{u}(t), t][u(t) - \tilde{u}(t)] + \text{higher-order terms}
\end{aligned}$$

On defining coefficient matrices

$$\begin{aligned}
A(t) &= \frac{\partial f}{\partial x}(\tilde{x}(t), \tilde{u}(t), t) \\
B(t) &= \frac{\partial f}{\partial u}(\tilde{x}(t), \tilde{u}(t), t) \\
C(t) &= \frac{\partial h}{\partial x}(\tilde{x}(t), \tilde{u}(t), t) \\
D(t) &= \frac{\partial h}{\partial u}(\tilde{x}(t), \tilde{u}(t), t)
\end{aligned}$$

rearranging slightly, and substituting deviation variables [recognizing that $\dot{x}_\delta(t) = \dot{x}(t) - \dot{\tilde{x}}(t)$] we have

$$\begin{aligned}
\dot{x}_\delta(t) &= A(t)x_\delta(t) + B(t)u_\delta(t) + \text{higher-order terms} \\
y_\delta(t) &= C(t)x_\delta(t) + D(t)u_\delta(t) + \text{higher-order terms}
\end{aligned}$$

Under the assumption that the state, input, and output remain close to their respective nominal trajectories, the high-order terms can be neglected, yielding the linear state equation

$$\begin{aligned}
\dot{x}_\delta(t) &= A(t)x_\delta(t) + B(t)u_\delta(t) \\
y_\delta(t) &= C(t)x_\delta(t) + D(t)u_\delta(t)
\end{aligned} \tag{1.5}$$

which constitutes the *linearization* of the nonlinear state equation (1.4) about the specified nominal trajectory. The linearized state equation

approximates the behavior of the nonlinear state equation provided that the deviation variables remain small in norm so that omitting the higher-order terms is justified.

If the nonlinear maps in Equation (1.4) do not explicitly depend on t, and the nominal trajectory is an equilibrium condition for a constant nominal input, then the coefficient matrices in the linearized state equation are constant; i.e., the linearization yields a time-invariant linear state equation.

Example 1.7 A ball rolling along a slotted rotating beam, depicted in Figure 1.8, is governed by the equations of motion given below. In this example we will linearize this nonlinear model about a given desired trajectory for this system.

$$\left[\frac{J_b}{r^2} + m\right]\ddot{p}(t) + mg\sin\theta(t) - mp(t)\dot{\theta}(t)^2 = 0$$

$$[mp(t)^2 + J + J_b]\ddot{\theta}(t) + 2\,mp(t)\dot{p}(t)\dot{\theta}(t) + mgp(t)\cos\theta(t) = \tau(t)$$

in which $p(t)$ is the ball position, $\theta(t)$ is the beam angle, and $\tau(t)$ is the applied torque. In addition, g is the gravitational acceleration constant, J is the mass moment of inertia of the beam, and m, r, and J_b are the mass, radius, and mass moment of inertia of the ball, respectively. We define state variables according to

$$x_1(t) = p(t)$$
$$x_2(t) = \dot{p}(t)$$
$$x_3(t) = \theta(t)$$
$$x_4(t) = \dot{\theta}(t)$$

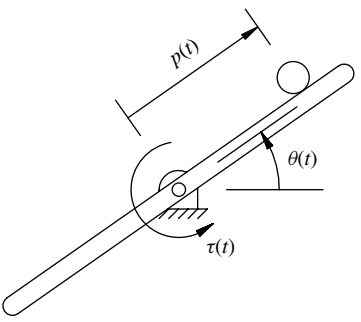

FIGURE 1.8 Ball and beam apparatus.

In addition, we take the input to be the applied torque $\tau(t)$ and the output to be the ball position $p(t)$, so

$$u(t) = \tau(t)$$
$$y(t) = p(t)$$

The resulting nonlinear state equation plus the output equation then are

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= b[x_1(t)x_4(t)^2 - g\sin x_3(t)] \\
\dot{x}_3(t) &= x_4(t) \\
\dot{x}_4(t) &= \frac{-2mx_1(t)x_2(t)x_4(t) - mgx_1(t)\cos x_3(t) + u(t)}{mx_1(t)^2 + J + J_\mathrm{b}} \\
y(t) &= x_1(t)
\end{aligned}$$

in which $b = m/[(J_\mathrm{b}/r^2) + m]$.

We consider nominal trajectories corresponding to a steady and level beam and constant-velocity ball position responses. In terms of an initial ball position p_0 at the initial time t_0 and a constant ball velocity v_0, we take

$$\begin{aligned}
\tilde{x}_1(t) &= \tilde{p}(t) = v_0(t - t_0) + p_0 \\
\tilde{x}_2(t) &= \dot{\tilde{p}}(t) = v_0 \\
\tilde{x}_3(t) &= \tilde{\theta}(t) = 0 \\
\tilde{x}_4(t) &= \dot{\tilde{\theta}}(t) = 0 \\
\tilde{u}(t) &= \tilde{\tau}(t) = mg\tilde{x}_1(t)
\end{aligned}$$

for which it remains to verify that Definition 1.1 is satisfied. Comparing

$$\begin{aligned}
\dot{\tilde{x}}_1(t) &= v_0 \\
\dot{\tilde{x}}_2(t) &= 0 \\
\dot{\tilde{x}}_3(t) &= 0 \\
\dot{\tilde{x}}_4(t) &= 0
\end{aligned}$$

with

$$\begin{aligned}
\tilde{x}_2(t) &= v_0 \\
b(\tilde{x}_1(t)\tilde{x}_4(t)^2 - g\sin\tilde{x}_3(t)) &= b(0 - g\sin(0)) = 0 \\
\tilde{x}_4(t) &= 0
\end{aligned}$$

$$\frac{\begin{array}{c}-2m\tilde{x}_1(t)\tilde{x}_2(t)\tilde{x}_4(t)- \\ mg\tilde{x}_1(t)\cos\tilde{x}_3(t)+\tilde{u}(t)\end{array}}{m\tilde{x}_1(t)^2+J+J_\mathrm{b}} = \frac{0-mg\tilde{x}_1(t)\cos(0)+mg\tilde{x}_1(t)}{m\tilde{x}_1(t)^2+J+J_\mathrm{b}} = 0$$

we see that $\tilde{x}(t)$ is a valid nominal state trajectory for the nominal input $\tilde{u}(t)$. As an immediate consequence, the nominal output is $\tilde{y}(t) = \tilde{x}_1(t) = \tilde{p}(t)$. It follows directly that deviation variables are specified by

$$x_\delta(t) = \begin{bmatrix} p(t)-\tilde{p}(t) \\ \dot{p}(t)-\dot{\tilde{p}}(t) \\ \theta(t)-0 \\ \dot{\theta}(t)-0 \end{bmatrix}$$

$$u_\delta(t) = \tau(t) - mg\tilde{p}(t)$$

$$y_\delta(t) = p(t) - \tilde{p}(t)$$

With

$$f(x,u) = \begin{bmatrix} f_1(x_1,x_2,x_3,x_4,u) \\ f_2(x_1,x_2,x_3,x_4,u) \\ f_3(x_1,x_2,x_3,x_4,u) \\ f_4(x_1,x_2,x_3,x_4,u) \end{bmatrix} = \begin{bmatrix} x_2 \\ b(x_1{x_4}^2 - g\sin x_3) \\ x_4 \\ \dfrac{-2mx_1x_2x_4 - mgx_1\cos x_3 + u}{m{x_1}^2+J+J_\mathrm{b}} \end{bmatrix}$$

partial differentiation yields

$$\frac{\partial f}{\partial x}(x,u) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ b{x_4}^2 & 0 & -bg\cos x_3 & 2bx_1x_4 \\ 0 & 0 & 0 & 1 \\ \dfrac{\partial f_4}{\partial x_1} & \dfrac{-2mx_1x_4}{m{x_1}^2+J+J_\mathrm{b}} & \dfrac{mgx_1\sin x_3}{m{x_1}^2+J+J_\mathrm{b}} & \dfrac{-2mx_1x_2}{m{x_1}^2+J+J_\mathrm{b}} \end{bmatrix}$$

where

$$\frac{\partial f_4}{\partial x_1} = \frac{\begin{array}{c}[(-2mx_2x_4 - mg\cos x_3)(m{x_1}^2+J+J_\mathrm{b})]- \\ [(-2mx_1x_2x_4 - mgx_1\cos x_3 + u)(2mx_1)]\end{array}}{(m{x_1}^2+J+J_\mathrm{b})^2}$$

$$\frac{\partial f}{\partial u}(x,u) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \hline m{x_1}^2+J+J_\mathrm{b} \end{bmatrix}$$

$$\frac{\partial h}{\partial x}(x,u) = [\,1 \quad 0 \quad 0 \quad 0\,]$$
$$\frac{\partial h}{\partial u}(x,u) = 0$$

Evaluating at the nominal trajectory gives

$$A(t) = \frac{\partial f}{\partial x}[\tilde{x}(t), \tilde{u}(t)]$$
$$= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -bg & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{-mg}{m\tilde{p}(t)^2 + J + J_b} & 0 & 0 & \dfrac{-2m\tilde{p}(t)v_0}{m\tilde{p}(t)^2 + J + J_b} \end{bmatrix}$$

$$B(t) = \frac{\partial f}{\partial u}[\tilde{x}(t), \tilde{u}(t)] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{m\tilde{p}(t)^2 + J + J_b} \end{bmatrix}$$

$$C(t) = \frac{\partial h}{\partial x}[\tilde{x}(t), \tilde{u}(t)] = [\,1 \quad 0 \quad 0 \quad 0\,]$$

$$D(t) = \frac{\partial h}{\partial u}[\tilde{x}(t), \tilde{u}(t)] = 0 \tag{1.6}$$

which, together with the deviation variables defined previously, specifies the linearized time-varying state equation for the ball and beam system.

A special case of the nominal trajectory considered thus far in this example corresponds to zero ball velocity $v_0 = 0$ and, consequently, constant ball position $\tilde{p}(t) = p_0$. The beam must remain steady and level, so the nominal state trajectory and input reduce to

$$\tilde{x}_1(t) = \tilde{p}(t) = p_0$$
$$\tilde{x}_1(t) = \dot{\tilde{p}}(t) = 0$$
$$\tilde{x}_1(t) = \tilde{\theta}(t) = 0$$
$$\tilde{x}_1(t) = \dot{\tilde{\theta}}(t) = 0$$
$$\tilde{u}(t) = \tilde{\tau}(t) = mg\ \ p_0$$

with an accompanying impact on the deviation variables. Given that the nonlinear ball and beam dynamics are time invariant and that now the

nominal state trajectory and input are constant and characterize an equilibrium condition for these dynamics, the linearization process yields a time-invariant linear state equation. The associated coefficient matrices are obtained by making the appropriate substitutions in Equation (1.6) to obtain

$$A = \frac{\partial f}{\partial x}(\tilde{x}, \tilde{u}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -bg & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{-mg}{m\ p_0^2 + J + J_b} & 0 & 0 & 0 \end{bmatrix}$$

$$B = \frac{\partial f}{\partial u}(\tilde{x}, \tilde{u}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{m\ p_0^2 + J + J_b} \end{bmatrix}$$

$$C = \frac{\partial h}{\partial x}(\tilde{x}, \tilde{u}) = [\,1 \quad 0 \quad 0 \quad 0\,]$$

$$D = \frac{\partial h}{\partial u}(\tilde{x}, \tilde{u}) = 0$$

□

1.5 CONTROL SYSTEM ANALYSIS AND DESIGN USING MATLAB

In each chapter we include a section to identify and explain the use of MATLAB software and MATLAB functions for state-space analysis and design methods. We use a continuing example to demonstrate the use of MATLAB throughout; this is a single-input, single-output two–dimensional rotational mechanical system that will allow the student to perform all operations by hand to compare with MATLAB results. We assume that the Control Systems Toolbox is installed with MATLAB.

MATLAB: General, Data Entry, and Display

In this section we present general MATLAB commands, and we start the Continuing MATLAB Example. We highly recommend the use of MATLAB m-files, which are scripts and functions containing MATLAB commands that can be created and modified in the MATLAB Editor and then executed. Throughout the MATLAB examples, **`bold Courier New`** font indicates MATLAB function names, user inputs, and variable names; this is given for emphasis only. Some useful MATLAB statements are listed below to help the novice get started.

General MATLAB Commands:

help	Provides a list of topics for which you can get online help.
help fname	Provides online help for MATLAB function **fname** .
%	The **%** symbol at any point in the code indicates a comment; text beyond the **%** is ignored by MATLAB and is highlighted in **green** .
;	The semicolon used at the end of a line suppresses display of the line's result to the MATLAB workspace.
clear	This command clears the MATLAB workspace, i.e., erases any previous user-defined variables.
clc	Clears the cursor.
figure(n)	Creates an empty figure window (numbered **n**) for graphics.
who	Displays a list of all user-created variable names.
whos	Same as **who** but additionally gives the dimension of each variable.
size(name)	Responds with the dimension of the matrix **name** .
length(name)	Responds with the length of the vector **name** .
eye(n)	Creates an $n \times n$ identity matrix I_n.
zeros(m,n)	Creates a $m \times n$ array of zeros.
ones(m,n)	Creates a $m \times n$ array of ones.
t = t0:dt:tf	Creates an evenly spaced time array starting from initial time **t0** and ending at final time **tf** , with steps of **dt** .
disp('string')	Print the text **string** to the screen.
name = input('string')	The **input** command displays a text **string** to the user, prompting for input; the entered data then are written to the variable **name** .

In the MATLAB Editor (not in this book), comments appear in **green**, text strings appear in **red**, and logical operators and other reserved programming words appear in **blue**.

MATLAB for State-Space Description

MATLAB uses a data-structure format to describe linear time-invariant systems. There are three primary ways to describe a linear time-invariant system in MATLAB: (1) state-space realizations specified by coefficient matrices **A**, **B**, **C**, and **D** (**ss**); (2) transfer functions with (**num**, **den**), where **num** is the array of polynomial coefficients for the transfer-function numerator and **den** is the array of polynomial coefficients for the transfer-function denominator (**tf**); and (3) transfer functions with (**z**, **p**, **k**), where **z** is the array of numerator polynomial roots (the zeros), **p** is the array of denominator polynomial roots (the poles), and **k** is the system gain. There is a fourth method, frequency response data (**frd**), which will not be considered in this book. The three methods to define a continuous-time linear time-invariant system in MATLAB are summarized below:

```
SysName = ss(A,B,C,D);
SysName = tf(num,den);
SysName = zpk(z,p,k);
```

In the first statement (**ss**), a scalar 0 in the **D** argument position will be interpreted as a zero matrix **D** of appropriate dimensions. Each of these three statements (**ss, tf, zpk**) may be used to define a system as above or to convert between state-space, transfer-function, and zero-pole-gain descriptions of an existing system. Alternatively, once the linear time-invariant system **SysName** is defined, the parameters for each system description may be extracted using the following statements:

```
[num,den]          = tfdata(SysName);
[z,p,k]            = zpkdata(SysName);
[A,B,C,D]          = ssdata(SysName);
```

In the first two statements above, if we have a single-input, single-output system, we can use the switch **'v'**: **tfdata(SysName,'v')** and **zpkdata(SysName,'v')**. There are three methods to access data from the defined linear time-invariant **SysName**: **set**/**get** commands, direct structure referencing, and data-retrieval commands. The latter approach is given above; the first two are:

```
set(SysName,PropName,PropValue);
PropValue = get(SysName,PropName);
SysName.PropName = PropValue;
% equivalent to 'set' command
```

```
PropValue  = SysName.PropName;
% equivalent to 'get' command
```

In the preceding, **SysName** is set by the user as the desired name for the defined linear time-invariant system. **PropName** (property name) represents the valid properties the user can modify, which include **A, B, C, D** for **ss**, **num, den, variable** (the default is '**s** ' for a continuous system Laplace variable) for **tf**, and **z, p, k, variable** (again, the default is '**s** ') for **zpk**. The command **set(SysName)** displays the list of properties for each data type. The command **get(SysName)** displays the value currently stored for each property. **PropValue** (property value) indicates the value that the user assigns to the property at hand. In previous MATLAB versions, many functions required the linear time-invariant system input data (**A, B, C, D** for state space, **num, den** for transfer function, and **z, p, k** for zero-pole-gain notation); although these still should work, MATLAB's preferred mode of operation is to pass functions the **SysName** linear time-invariant data structure. For more information, type **help ltimodels** and **help ltiprops** at the MATLAB command prompt.

Continuing MATLAB Example

Modeling A single-input, single-output rotational mechanical system is shown in Figure 1.9. The single input is an externally applied torque $\tau(t)$, and the output is the angular displacement $\theta(t)$. The constant parameters are motor shaft polar inertia J, rotational viscous damping coefficient b, and torsional spring constant k_R (provided by the flexible shaft). This example will be used in every chapter to demonstrate the current topics via MATLAB for a model that will become familiar. To derive the system model, MATLAB does not help (unless the Symbolic Math Toolbox capabilities of MATLAB are used).

In the free-body diagram of Figure 1.10, the torque resulting from the rotational viscous damping opposes the instantaneous direction of the angular velocity and the torque produced by the restoring spring

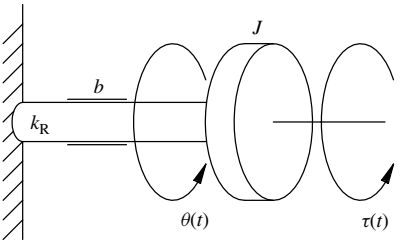

FIGURE 1.9 Continuing MATLAB Example system.

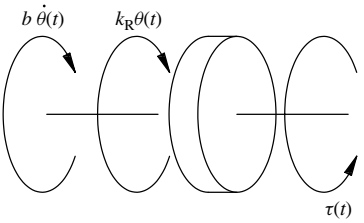

FIGURE 1.10 Continuing MATLAB Example free-body diagram.

opposes the instantaneous direction of the angular displacement. We apply Euler's rotational law (the rotational equivalent of Newton's Second Law), to derive the system model. Euler's rotational law may be stated as $\sum M = J\alpha$, where $\sum M$ is the sum of moments, J is the polar moment of inertia, and α is the shaft angular acceleration.

$$\sum M = J\ddot{\theta}(t) = \tau(t) - b\dot{\theta}(t) - k_R\theta(t)$$

This system can be represented by the single second-order linear time-invariant ordinary differential equation

$$J\ddot{\theta}(t) + b\dot{\theta}(t) + k_R\theta(t) = \tau(t)$$

This equation is the rotational equivalent of a translational mechanical mass-spring-damper system with torque $\tau(t)$ as the input and angular displacement $\theta(t)$ as the output.

State-Space Description Now we derive a valid state-space description for the Continuing MATLAB Example. That is, we specify the state variables and derive the coefficient matrices A, B, C, and D. We start with the second-order differential equation above for which we must define two state variables $x_i(t), i = 1, 2$. Again, energy-storage elements guide our choice of states:

$$x_1(t) = \theta(t)$$

$$x_2(t) = \dot{\theta}(t) = \dot{x}_1(t)$$

We will have two first-order differential equations, derived from the original second-order differential equation, and $\dot{x}_1(t) = x_2(t)$ from above. The state differential equation is

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{-k_R}{J} & \dfrac{-b}{J} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{J} \end{bmatrix} \tau(t)$$

TABLE 1.1 Numerical Parameters for the Continuing MATLAB Example

Parameter	Value	Units	Name
J	1	kg-m^2	motor shaft polar inertia
b	4	N-m-s	motor shaft damping constant
k_R	40	N-m/rad	torsional spring constant

The algebraic output equation is:

$$y(t) = [\,1 \quad 0\,] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + [0]\tau(t)$$

The coefficient matrices A, B, C, D for this Continuing MATLAB Example are thus:

$$A = \begin{bmatrix} 0 & 1 \\ -\dfrac{k_R}{J} & -\dfrac{b}{J} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \dfrac{1}{J} \end{bmatrix} \qquad C = [\,1 \quad 0\,]$$
$$D = 0$$

This specifies a two–dimensional single-input, single-output system with $m = 1$ input, $p = 1$ output, and $n = 2$ states. Let us assume the constant parameters listed in Table 1.1 for the continuing MATLAB example.

Then the numerical coefficient matrices are

$$A = \begin{bmatrix} 0 & 1 \\ -40 & -4 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad C = [\,1 \quad 0\,] \qquad D = 0$$

Chapter by chapter we will present MATLAB code and results dealing with the topics at hand for the Continuing MATLAB Example. These code segments will be complete only if taken together over all chapters (i.e., ensuing code portions may require previously defined variables in earlier chapters to execute properly). Appendix C presents this complete program for all chapters. To get started, we need to define the coefficient matrices A, B, C, and D in MATLAB. Then we can find the system transfer function and zero-pole-gain descriptions.

```
%-------------------------------------------------
%   Chapter 1. State-Space Description
%-------------------------------------------------

J  = 1;
b  = 4;
kR = 40;
```

```
A = [0 1;-kR/J -b/J];                  % Define the
                                       % state-space
                                       % realization
B = [0;1/J];
C = [1 0];
D = [0];

JbkR = ss(A,B,C,D);                    % Define model from
                                       % state-space

JbkRtf  = tf(JbkR);                    % Convert to
                                       % transfer function
JbkRzpk = zpk(JbkR);                   % Convert to
                                       % zero-pole
                                       % description

[num,den]   = tfdata(JbkR,'v');        % Extract transfer
                                       % function
                                       % description
[z,p,k]     = zpkdata(JbkR,'v');       % Extract zero-pole
                                       % description

JbkRss  = ss(JbkRtf)                   % Convert to
                                       % state-space
                                       % description
```

The **ss** command yields

```
a =
                      x1          x2
           x1          0           1
           x2        -40          -4

b =
                      u1
           x1          0
           x2          1

c =
                      x1          x2
           y1          1           0
```

```
d =
                        u1
           y1            0
Continuous-time model.
```

The **tf** and **zpk** commands yield

```
Transfer function:
      1
--------------
s^2 + 4 s + 40


Zero/pole/gain:
       1
---------------
(s^2 + 4s + 40)
```

The **tfdata** and **zpkdata** commands yield

```
num =
     0     0     1

den =
    1.0000    4.0000   40.0000

z =
   Empty matrix: 0-by-1

p =
  -2.0000 + 6.0000i
  -2.0000 - 6.0000i

k =
     1
```

Finally, the second **ss** command yields

```
a =
                        x1          x2
           x1           -4          -5
           x2            8           0
```

```
b =
                    u1
          x1      0.25
          x2         0

c =
                    x1          x2
          y1         0         0.5

d =
                    u1
          y1         0
```

Note that when MATLAB converted from the **tf** to the **ss** description above, it returned a different state-space realization than the one originally defined. The validity of this outcome will be explained in Chapter 2.

1.6 CONTINUING EXAMPLES

Continuing Example 1: Two-Mass Translational Mechanical System

This multiple-input, multiple-output example will continue throughout each chapter of this book, building chapter by chapter to demonstrate the important topics at hand.

Modeling A mechanical system is represented by the two degree-of-freedom linear time-invariant system shown in Figure 1.11. There are two force inputs $u_i(t)$ and two displacement outputs $y_i(t), i = 1, 2$. The constant parameters are masses m_i, damping coefficients c_i, and spring coefficients $k_i, i = 1, 2$. We now derive the mathematical model for this system; i.e., we draw the free-body diagrams and then write the correct number of independent ordinary differential equations. All motion is constrained to be horizontal, as shown in Figure 1.11. Outputs $y_i(t)$ are each measured from the neutral spring equilibrium location of each mass m_i. Figure 1.12 shows the two free-body diagrams.

Now we apply Newton's second law twice, once for each mass, to derive the two second-order dynamic equations of motion:

$$\sum F_1 = m_1\ddot{y}_1(t) = k_2[y_2(t) - y_1(t)] + c_2[\dot{y}_2(t) - \dot{y}_1(t)] - k_1 y_1(t) - c_1\dot{y}_1(t) + u_1(t)$$

$$\sum F_2 = m_2\ddot{y}_2(t) = -k_2[y_2(t) - y_1(t)] - c_2[\dot{y}_2(t) - \dot{y}_1(t)] + u_2(t)$$

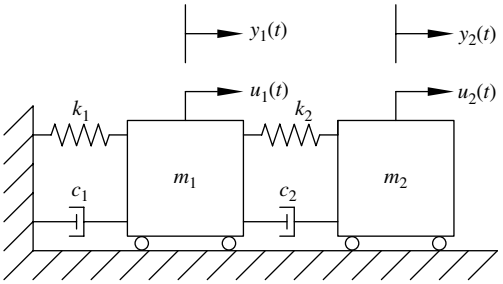

FIGURE 1.11 Continuing Example 1 system.

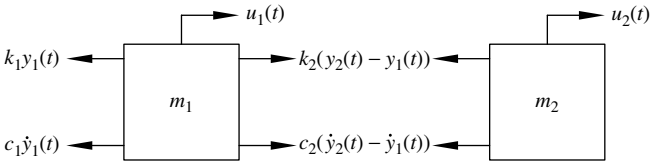

FIGURE 1.12 Continuing Example 1 free-body diagrams.

We rewrite these equations so that the output-related terms $y_i(t)$ appear on the left side along with their derivatives and the input forces $u_i(t)$ appear on the right. Also, $y_i(t)$ terms are combined.

$$m_1\ddot{y}_1(t) + (c_1 + c_2)\dot{y}_1(t) + (k_1 + k_2)y_1(t) - c_2\dot{y}_2(t) - k_2 y_2(t) = u_1(t)$$

$$m_2\ddot{y}_2(t) + c_2\dot{y}_2(t) + k_2 y_2(t) - c_2\dot{y}_1(t) - k_2 y_1(t) = u_2(t)$$

These equations are two linear, coupled, second-order ordinary differential equations. In this type of vibrational system, it is always possible to structure the equations such that the coefficients of $\ddot{y}_i(t)$, $\dot{y}_i(t)$, and $y_i(t)$ are positive in the ith equation, and the coefficients of any $\dot{y}_j(t)$ and $y_j(t)$ terms that appear in the ith equation are negative for $j \neq i$.

Example 1 is a multiple-input, multiple output system with two inputs $u_i(t)$ and two outputs $y_i(t)$. We can express the two preceding second-order differential equations in standard second-order matrix-vector form, $M\ddot{y}(t) + C\dot{y}(t) + Ky(t) = u(t)$, that is,

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{bmatrix} \ddot{y}_1(t) \\ \ddot{y}_2(t) \end{bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}\begin{bmatrix} \dot{y}_1(t) \\ \dot{y}_2(t) \end{bmatrix}$$

$$+ \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

State-Space Description Next, we must derive a valid state-space description for this system. That is, we specify the state variables and then

derive the coefficient matrices A, B, C, and D. We present two distinct cases:

a. Multiple-input, multiple-output: Both inputs and both outputs
b. Single-input, single-output: One input $u_2(t)$ and one output $y_1(t)$

We start with the form of the two coupled second-order differential equations above in which the highest-order derivatives $\ddot{y}_i(t), i = 1, 2$, are isolated. For both cases, the choice of state variables and the resulting system dynamics matrix A will be identical. This always will be true, i.e., A is fundamental to the system dynamics and does not change with different choices of inputs and outputs. For case a, we use both inputs $u_i(t)$; for case b, we must set $u_1(t) = 0$.

Case a: Multiple-Input, Multiple-Output Since we have two second-order differential equations, the state-space dimension is $n = 4$, and thus we need to define four state variables $x_i(t), i = 1, 2, 3, 4$. Again, energy-storage elements guide our choice of states:

$$
\begin{aligned}
x_1(t) &= y_1(t) \\
x_2(t) &= \dot{y}_1(t) = \dot{x}_1(t) \\
x_3(t) &= y_2(t) \\
x_4(t) &= \dot{y}_2(t) = \dot{x}_3(t)
\end{aligned}
$$

We will have four first-order ordinary differential equations derived from the original two second-order differential equations. Two are $\dot{x}_i(t) = x_{i+1}(t)$ from the state variable definitions above, for $i = 1, 3$. The remaining two come from the original second-order differential equations, rewritten by isolating accelerations and substituting the state variable definitions in place of the outputs and their derivatives. Also, we must divide by m_i to normalize each equation.

$$
\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{-(k_1 + k_2)x_1(t) - (c_1 + c_2)x_2(t) + k_2x_3 + c_2x_4(t) + u_1(t)}{m_1} \\
\dot{x}_3(t) &= x_4(t) \\
\dot{x}_4(t) &= \frac{k_2x_1(t) + c_2x_2(t) - k_2x_3(t) - c_2x_4(t) + u_2(t)}{m_2}
\end{aligned}
$$

The state differential equation is

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{-(k_1+k_2)}{m_1} & \dfrac{-(c_1+c_2)}{m_1} & \dfrac{k_2}{m_1} & \dfrac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_2}{m_2} & \dfrac{c_2}{m_2} & \dfrac{-k_2}{m_2} & \dfrac{-c_2}{m_2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \dfrac{1}{m_1} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{m_2} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

from which we identify coefficient matrices A and B:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{-(k_1+k_2)}{m_1} & \dfrac{-(c_1+c_2)}{m_1} & \dfrac{k_2}{m_1} & \dfrac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_2}{m_2} & \dfrac{c_2}{m_2} & \dfrac{-k_2}{m_2} & \dfrac{-c_2}{m_2} \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ \dfrac{1}{m_1} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{m_2} \end{bmatrix}$$

The algebraic output equation is

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

from which we identify coefficient matrices C and D:

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

This is a four-dimensional multiple-input, multiple-output system with $m = 2$ inputs, $p = 2$ outputs, and $n = 4$ states.

Case b: Single-Input, Single-Output: One Input u_2, One Output y_1. Remember, system dynamics matrix A does not change when considering different system inputs and outputs. For the single-input, single-output

case b, only coefficient matrices B, C, and D change. The state differential equation now is:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{-(k_1+k_2)}{m_1} & \dfrac{-(c_1+c_2)}{m_1} & \dfrac{k_2}{m_1} & \dfrac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_2}{m_2} & \dfrac{c_2}{m_2} & \dfrac{-k_2}{m_2} & \dfrac{-c_2}{m_2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{m_2} \end{bmatrix} u_2(t)$$

A is the same as that given previously, and the new input matrix is

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{m_2} \end{bmatrix}$$

The algebraic output equation now is:

$$y_1(t) = [\,1 \quad 0 \quad 0 \quad 0\,] \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + [0]u_2(t)$$

so that

$$C = [\,1 \quad 0 \quad 0 \quad 0\,] \qquad D = 0$$

This is still a four-dimensional system, now with $m = 1$ input and $p = 1$ output.

Continuing Example 2: Rotational Electromechanical System

This example also will continue throughout each chapter of this book, building chapter by chapter to demonstrate the important topics.

Modeling A simplified dc servomotor model is shown in Figure 1.13. The input is the armature voltage $v(t)$ and the output is the motor shaft

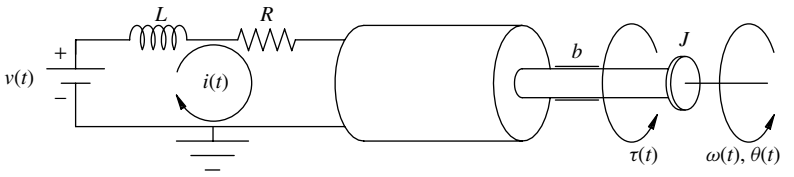

FIGURE 1.13 Continuing Example 2 system.

angular displacement $\theta(t)$. The constant parameters are armature circuit inductance and resistance L and R, respectively, and motor shaft polar inertia and rotational viscous damping coefficient J and b, respectively. The intermediate variables are armature current $i(t)$, motor torque $\tau(t)$, and motor shaft angular velocity $\omega(t) = \dot{\theta}(t)$. In this Continuing Example 2, we have simplified the model; we ignore back emf voltage, and there is no gear ratio or load inertia included. For improvements on each of these issues, see Continuing Exercise 3.

We can derive the dynamic model of this system in three steps: circuit model, electromechanical coupling, and rotational mechanical model. For the circuit model, Kirchhoff's voltage law yields a first-order differential equation relating the armature current to the armature voltage, that is,

$$L\frac{di(t)}{dt} + Ri(t) = v(t)$$

Motor torque is modeled as being proportional to the armature current, so the electromechanical coupling equation is

$$\tau(t) = k_{\mathrm{T}} i(t)$$

where k_{T} is the motor torque constant. For the rotational mechanical model, Euler's rotational law results in the following second-order differential equation relating the motor shaft angle $\theta(t)$ to the input torque $\tau(t)$:

$$J\ddot{\theta}(t) + b\dot{\theta}(t) = \tau(t)$$

To derive the overall system model, we need to relate the designated system output $\theta(t)$ to the designated system input $v(t)$. The intermediate variables $i(t)$ and $\tau(t)$ must be eliminated. It is convenient to use Laplace transforms and transfer functions for this purpose rather than manipulating the differential equations. Here, we are applying a method similar to Examples 1.5 and 1.6, wherein we use a transfer-function description to derive the state equations. We have

$$\frac{I(s)}{V(s)} = \frac{1}{Ls + R} \qquad \frac{T(s)}{I(s)} = k_{\mathrm{T}} \qquad \frac{\Theta(s)}{T(s)} = \frac{1}{Js^2 + bs}$$

Multiplying these transfer functions together, we eliminate the intermediate variables to generate the overall transfer function:

$$\frac{\Theta(s)}{V(s)} = \frac{k_{\mathrm{T}}}{(Ls+R)(Js^2+bs)}$$

Simplifying, cross-multiplying, and taking the inverse Laplace transform yields the following third-order linear time-invariant ordinary differential equation:

$$LJ\dddot{\theta}(t) + (Lb+RJ)\ddot{\theta}(t) + Rb\dot{\theta}(t) = k_{\mathrm{T}}v(t)$$

This equation is the mathematical model for the system of Figure 1.13. Note that there is no rotational mechanical spring term in this equation, i.e., the coefficient of the $\theta(t)$ term is zero.

State-Space Description Now we derive a valid state-space description for Continuing Example 2. That is, we specify the state variables and derive the coefficient matrices A, B, C, and D. The results then are written in matrix-vector form. Since we have a third-order differential equation, the state-space dimension is $n = 3$, and thus we need to define three state variables $x_i(t), i = 1, 2, 3$. We choose

$$\begin{aligned} x_1(t) &= \theta(t) \\ x_2(t) &= \dot{\theta}(t) = \dot{x}_1(t) \\ x_3(t) &= \ddot{\theta}(t) = \dot{x}_2(t) \end{aligned}$$

We will have three first-order differential equations, derived from the original third-order differential equation. Two are $\dot{x}_i(t) = x_{i+1}(t)$ from the state variable definitions above, for $i = 1, 2$. The remaining first-order differential equation comes from the original third-order differential equation, rewritten by isolating the highest derivative $\dddot{\theta}(t)$ and substituting the state-variable definitions in place of output $\theta(t)$ and its derivatives. Also, we divide the third equation by LJ:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= x_3(t) \\ \dot{x}_3(t) &= \frac{-(Lb+RJ)}{LJ}x_3(t) - \frac{Rb}{LJ}x_2(t) + \frac{k_{\mathrm{T}}}{LJ}v(t) \end{aligned}$$

The state differential equation is

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \dfrac{-Rb}{LJ} & \dfrac{-(Lb+RJ)}{LJ} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{k_T}{LJ} \end{bmatrix} v(t)$$

from which we identify coefficient matrices A and B:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \dfrac{-Rb}{LJ} & \dfrac{-(Lb+RJ)}{LJ} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ \dfrac{k_T}{LJ} \end{bmatrix}$$

The algebraic output equation is

$$y(t) = [\,1 \quad 0 \quad 0\,] \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + [0]v(t)$$

from which we identify coefficient matrices C and D:

$$C = [\,1 \quad 0 \quad 0\,] \qquad D = 0$$

This is a three-dimensional single-input, single-output system with $m = 1$ input, $p = 1$ output, and $n = 3$ states.

1.7 HOMEWORK EXERCISES

We refer the reader to the Preface for a description of the four classes of exercises that will conclude each chapter: Numerical Exercises, Analytical Exercises, Continuing MATLAB Exercises, and Continuing Exercises.

Numerical Exercises

NE1.1 For the following systems described by the given transfer functions, derive valid state-space realizations (define the state variables and derive the coefficient matrices A, B, C, and D).

a. $G(s) = \dfrac{Y(s)}{U(s)} = \dfrac{1}{s^2 + 2s + 6}$

b. $G(s) = \dfrac{Y(s)}{U(s)} = \dfrac{s + 3}{s^2 + 2s + 6}$

c. $G(s) = \dfrac{Y(s)}{U(s)} = \dfrac{10}{s^3 + 4s^2 + 8s + 6}$

d. $G(s) = \dfrac{Y(s)}{U(s)} = \dfrac{s^2 + 4s + 6}{s^4 + 10s^3 + 11s^2 + 44s + 66}$

NE1.2 Given the following differential equations (or systems of differential equations), derive valid state-space realizations (define the state variables and derive the coefficient matrices A, B, C, and D).

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

b. $\ddot{y}(t) + 3\dot{y}(t) + 10y(t) = u(t)$

c. $\dddot{y}(t) + 2\ddot{y}(t) + 3\dot{y}(t) + 5y(t) = u(t)$

d. $\ddot{y}_1(t) + 5y_1(t) - 10[y_2(t) - y_1(t)] = u_1(t)$

$2\ddot{y}_2(t) + \dot{y}_2(t) + 10[y_2(t) - y_1(t)] = u_2(t)$

Analytical Exercises

AE1.1 Suppose that A is $n \times m$ and H is $p \times q$. Specify dimensions for the remaining matrices so that the following expression is valid.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

AE1.2 Suppose that A and B are square matrices, not necessarily of the same dimension. Show that

$$\begin{vmatrix} A & 0 \\ 0 & B \end{vmatrix} = |A| \cdot |B|$$

AE1.3 Continuing AE1.2, show that

$$\begin{vmatrix} A & 0 \\ C & B \end{vmatrix} = |A| \cdot |B|$$

AE1.4 Continuing AE1.3, show that if A is nonsingular,

$$\begin{vmatrix} A & D \\ C & B \end{vmatrix} = |A| \cdot |B - CA^{-1}D|$$

AE1.5 Suppose that X is $n \times m$ and Y is $m \times n$. With I_k denoting the $k \times k$ identity matrix for any integer $k > 0$, show that

$$|I_n - XY| = |I_m - YX|$$

Explain the significance of this result when $m = 1$. Hint: Apply AE1.4 to

$$\begin{bmatrix} I_m & Y \\ X & I_n \end{bmatrix} \text{ and } \begin{bmatrix} I_n & X \\ Y & I_m \end{bmatrix}$$

AE1.6 Show that the determinant of a square upper triangular matrix (zeros everywhere below the main diagonal) equals the product if its diagonal entries.

AE1.7 Suppose that A and C are nonsingular $n \times n$ and $m \times m$ matrices, respectively. Verify that

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[C^{-1} + DA^{-1}B]^{-1}DA^{-1}$$

What does this formula reduce to when $m = 1$ and $C = 1$?

AE1.8 Suppose that X is $n \times m$ and Y is $m \times n$. With I_k denoting the $k \times k$ identity matrix for any integer $k > 0$, show that

$$(I_n - XY)^{-1}X = X(I_m - YX)^{-1}$$

when the indicated inverses exist.

AE1.9 Suppose that A and B are nonsingular matrices, not necessarily of the same dimension. Show that

$$\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & B^{-1} \end{bmatrix}$$

AE1.10 Continuing AE1.8, derive expressions for

$$\begin{bmatrix} A & 0 \\ C & B \end{bmatrix}^{-1} \text{ and } \begin{bmatrix} A & D \\ 0 & B \end{bmatrix}^{-1}$$

AE1.11 Suppose that A is nonsingular and show that

$$\begin{bmatrix} A & D \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + E\Delta^{-1}F & -E\Delta^{-1} \\ -\Delta^{-1}F & \Delta^{-1} \end{bmatrix}$$

in which $\Delta = B - CA^{-1}D$, $E = A^{-1}D$, and $F = CA^{-1}$.

AE1.12 Compute the inverse of the $k \times k$ Jordan block matrix

$$J_k(\lambda) = \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ 0 & 0 & \lambda & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \cdots & \lambda \end{bmatrix}$$

AE1.13 Suppose that $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear transformation and $\mathbb{S}$ is a subspace of $\mathbb{R}^m$. Verify that the set

$$A^{-1}\mathbb{S} = \{x \in \mathbb{R}^n | Ax \in \mathbb{S}\}$$

is a subspace of $\mathbb{R}^n$. This subspace is referred to as the *inverse image* of the subspace $\mathbb{S}$ under the linear transformation A.

AE1.14 Show that for conformably dimensioned matrices A and B, any induced matrix norm satisfies

$$\|AB\| \leq \|A\|\|B\|$$

AE1.15 Show that for A nonsingular, any induced matrix norm satisfies

$$\|A^{-1}\| \geq \frac{1}{\|A\|}$$

AE1.16 Show that for any square matrix A, any induced matrix norm satisfies

$$\|A\| \geq \rho(A)$$

where $\rho(A) \triangleq \max_{\lambda_i \in \sigma(A)} |\lambda_i|$ is the spectral radius of A.

Continuing MATLAB Exercises

CME1.1 Given the following open-loop single-input, single-output two−dimensional linear time-invariant state equations, namely,

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ \sqrt{2} \end{bmatrix} u(t)$$

$$y(t) = [\,1 \quad -\sqrt{2}/2\,] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + [0]u(t)$$

find the associated open-loop transfer function $H(s)$.

CME1.2 Given the following open-loop single-input, single-output three−dimensional linear time-invariant state equations, namely

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -52 & -30 & -4 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = [\,20 \quad 1 \quad 0\,] \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + [0]u(t)$$

find the associated open-loop transfer function $H(s)$.

CME1.3 Given the following open-loop single-input, single-output fourth−order linear time-invariant state equations, namely,

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\\ \dot{x}_4(t)\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -962 & -126 & -67 & -4\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\\ x_4(t)\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix} u(t)$$

$$y(t) = [300 \quad 0 \quad 0 \quad 0]\begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\\ x_4(t)\end{bmatrix} + [0]u(t)$$

find the associated open-loop transfer function $H(s)$.

CME1.4 Given the following open-loop single-input, single-output four−dimensional linear time-invariant state equations, namely,

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\\ \dot{x}_4(t)\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -680 & -176 & -86 & -6\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\\ x_4(t)\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix} u(t)$$

$$y(t) = [\,100 \quad 20 \quad 10 \quad 0\,]\begin{bmatrix}x_1(t)\\ x_2(t)\\ x_3(t)\\ x_4(t)\end{bmatrix} + [0]u(t)$$

find the associated open-loop transfer function $H(s)$.

Continuing Exercises

CE1.1a A mechanical system is represented by the three degree-of-freedom linear time-invariant system shown in Figure 1.14. There are three input forces $u_i(t)$ and three output displacements $y_i(t), i = 1, 2, 3$. The constant parameters are the masses $m_i, i = 1, 2, 3$, the spring coefficients k_j, and the damping coefficients $c_j, j = 1, 2, 3, 4$. Derive the mathematical model for this system, i.e., draw the free-body diagrams and write the correct number of independent ordinary differential equations. All motion is constrained to be horizontal. Outputs $y_i(t)$ are each measured from the neutral spring equilibrium location of each mass m_i.

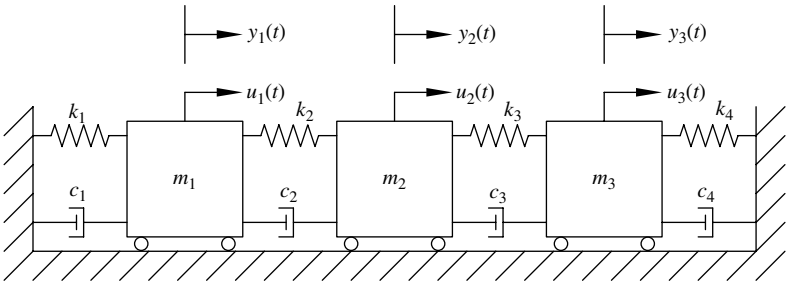

FIGURE 1.14 Diagram for Continuing Exercise 1.

Also express the results in matrix-vector form $M\ddot{y}(t) + C\dot{y}(t) + Ky(t) = u(t)$.

CE1.1b Derive a valid state-space realization for the CE1.1a system. That is, specify the state variables and derive the coefficient matrices A, B, C, and D. Write out your results in matrix-vector form. Give the system order and matrix/vector dimensions of your result. Consider three distinct cases:

i. Multiple-input, multiple-output: three inputs, three displacement outputs.

ii. Multiple-input, multiple-output: two inputs [$u_1(t)$ and $u_3(t)$ only], all three displacement outputs.

iii. Single-input, single-output: input $u_2(t)$ and output $y_3(t)$.

CE1.2a The nonlinear, inherently unstable inverted pendulum is shown in Figure 1.15. The goal is to maintain the pendulum angle $\theta(t) = 0$ by using a feedback controller with a sensor (encoder or potentiometer) for $\theta(t)$ and an actuator to produce an input force $f(t)$. The cart mass is m_1, the pendulum point mass is m_2, and we assume that the pendulum rod is massless. There are two possible outputs, the pendulum angle $\theta(t)$ and the cart displacement $w(t)$. The classical inverted pendulum has only one input, the force $f(t)$. We will consider a second case, using a motor to provide a second input $\tau(t)$ (not shown) at the rotary joint of Figure 1.15. For both cases (they will be very similar), derive the nonlinear model for this system, i.e., draw the free-body diagrams and write the correct number of independent ordinary differential equations. Alternately, you may use the Lagrangian dynamics approach that does not require free-body diagrams. Apply the steps outlined in Section 1.4 to derive a linearized model about the unstable equilibrium condition corresponding to zero angular displacement.

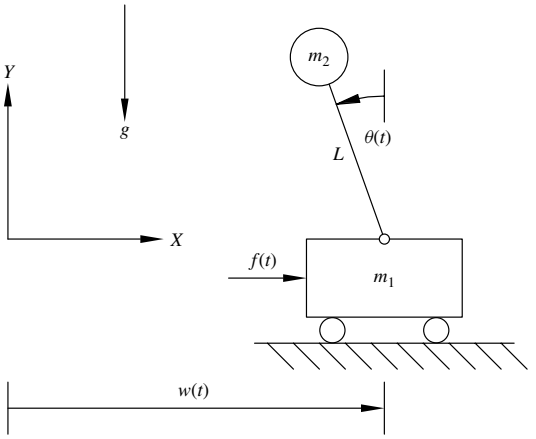

FIGURE 1.15 Diagram for Continuing Exercise 2.

CE1.2b Derive a valid state-space description for the system of Figure 1.15. That is, specify the state variables and derive the coefficient matrices A, B, C, and D. Write out your results in matrix-vector form. Give the system order and matrix-vector dimensions of your result. Consider three distinct cases:

i. Single-input, single-output: input $f(t)$ and output $\theta(t)$.

ii. Single-input, multiple-output: one input $f(t)$ and two outputs $w(t)$ and $\theta(t)$.

iii. Multiple-input, multiple-output: two inputs $f(t)$ and $\tau(t)$ (add a motor to the inverted pendulum rotary joint, traveling with the cart) and two outputs $w(t)$ and $\theta(t)$.

CE1.3a Figure 1.16 shows a single robot joint/link driven through a gear ratio n by an armature-controlled dc servomotor. The input is the dc armature voltage $v_A(t)$ and the output is the load-shaft angle $\theta_L(t)$. Derive the mathematical model for this system; i.e., develop the circuit differential equation, the electromechanical coupling equations, and the rotational mechanical differential equation. Eliminate intermediate variables and simplify; it will be convenient to use a transfer-function approach. Assume the mass-moment of inertia of all outboard links plus any load $J_L(t)$ is a constant (a reasonable assumption when the gear ratio $n = \omega_M/\omega_L$ is much greater than 1, as it is in the case of industrial robots). The parameters in Figure 1.16 are summarized below.

CE1.3b Derive a valid state-space description for the system of Figure 1.16. That is, specify the state variables and derive the

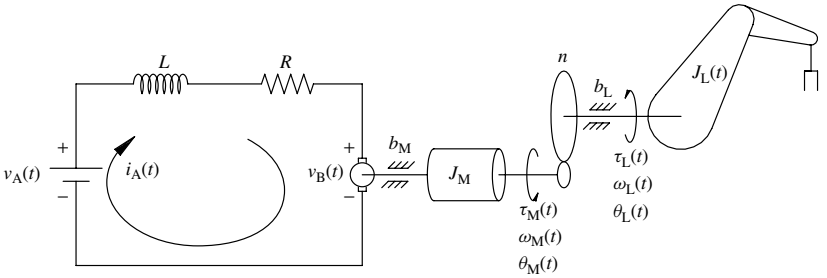

FIGURE 1.16 Diagram for Continuing Exercise 3.

$v_A(t)$	armature voltage	L	armature inductance	R	armature resistance
$i_A(t)$	armature current	$v_B(t)$	back emf voltage	k_B	back emf constant
J_M	motor inertia	b_M	motor viscous damping	$\tau_M(t)$	motor torque
k_T	torque constant	$\omega_M(t)$	motor shaft velocity	$\theta_M(t)$	motor shaft angle
n	gear ratio	$J_L(t)$	load inertia	b_L	load viscous damping
$\tau_L(t)$	load shaft torque	$\omega_L(t)$	load shaft velocity	$\theta_L(t)$	load shaft angle

coefficient matrices A, B, C, and D. Write out your results in matrix-vector form. Give the system order and matrix-vector dimensions of your result. Consider two distinct cases:

i. Single-input, single-output: armature voltage $v_A(t)$ as the input and robot load shaft angle $\theta_L(t)$ as the output.

ii. Single-input, single-output: armature voltage $v_A(t)$ as the input and robot load shaft angular velocity $\omega_L(t)$ as the output.

CE1.4 The nonlinear ball and beam apparatus was introduced in Section 1.4, Example 1.7. This system will form the basis for Continuing Exercise 4. Figure 1.8 shows the ball and beam system geometry, where the goal is to control the position of the ball on the slotted rotating beam by applying a torque $\tau(t)$ to the beam. CE1.4a and CE1.4b are already completed for you; i.e., the nonlinear equations of motion have been presented, and a valid state-space realization has been derived and linearized about the given nominal trajectory. Thus the assignment here is to rederive these steps for Continuing Exercise 4. As in Example 1.7, use the single-input, single-output model with input torque $\tau(t)$ and output ball position $p(t)$.

For all ensuing Continuing Exercise 4 assignments, use a special case of the time-varying linear state equation (1.6) to obtain a linear time-invariant state-space realization of the nonlinear model; use zero velocity $v_0 = 0$ and constant nominal ball position $\tilde{p}(t) = p_0$. Derive the linear time-invariant coefficient matrices A, B, C, and D for this special case.

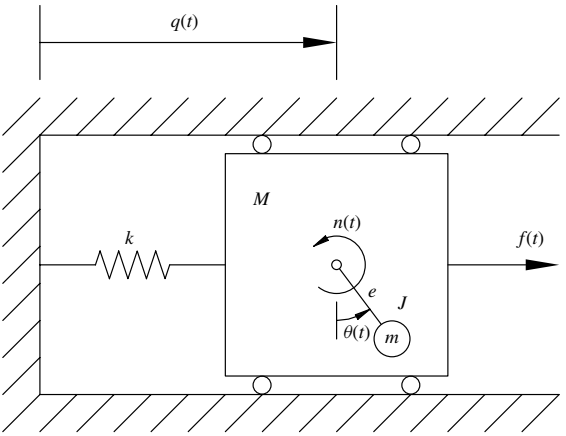

FIGURE 1.17 Diagram for Continuing Exercise 5 (top view).

CE1.5a A nonlinear proof-mass actuator system is shown in Figure 1.17. This system has been proposed as a nonlinear controls benchmark problem (Bupp et al., 1998). However, in this book, the system will be linearized about a nominal trajectory, and the linearization then will be used in all ensuing chapters as Continuing Exercise 5.

This is a vibration-suppression system wherein the control goal is to reject an unknown, unwanted disturbance force $f(t)$ by using the control torque $n(t)$ to drive the unbalanced rotating pendulum (proof mass) to counter these disturbances. The block of mass M is connected to the wall via a spring with spring constant k and is constrained to translate as shown; $q(t)$ is the block displacement. The rotating pendulum has a point mass m at the tip, and the pendulum has mass moment of inertia J. The pendulum length is e and the pendulum angle $\theta(t)$ is measured as shown. Assume that the system is operating in the horizontal plane, so gravity need not be considered. Derive the nonlinear model for this system.

CE1.5b For nominal equilibria corresponding to zero control torque, linearize the nonlinear model from CE1.5a and derive a valid state-space description. That is, follow the procedure of Section 1.4 and derive the linearized coefficient matrices A, B, C, and D. Write out your results in matrix-vector form. Give the system order and matrix-vector dimensions of your result. Consider only the single-input, single-output case with input torque $n(t)$ and output displacement $q(t)$. In ensuing problems, the control objective will be to regulate nonequilibrium initial conditions.