We can also reorganize the equations into the following matrix form

$$\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}\begin{bmatrix} \ddot{x}_1(t) \\ \ddot{x}_2(t) \end{bmatrix} + \begin{bmatrix} F & -F \\ -F & F \end{bmatrix}\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} + \begin{bmatrix} K & -K \\ -K & K \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} f(t). \quad (2.11)$$

Models of the form (2.11) are very typical for mechanical systems. Such a model is called a **second-order model**.

Next let us consider a pendulum shown in Figure 2.6. Here a torque $\tau_i(t)$ can be applied around the pivot point and we are concerned with the angle $\theta(t)$ between the pendulum and the vertical downward direction. The length of the pendulum is L and the mass M of the pendulum is concentrated at its tip.

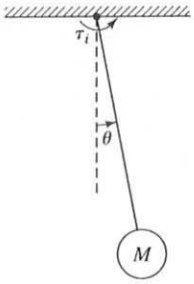

FIGURE 2.6: A pendulum.

In a rotational motion, Newton's second law takes the form

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

where J is called the moment of inertia, $\theta(t)$ is the angular displacement, and $\tau(t)$ is the total torque applied.

Applying this to the pendulum system, we know that the moment of inertia is $J = ML^2$ and there are two torques applied to the system: the externally applied torque $\tau_i(t)$, and the torque due to the gravity of the mass which is $MgL\sin\theta(t)$. Therefore, the equation governing the motion is given by

$$ML^2\frac{d^2\theta(t)}{dt^2} = \tau_i(t) - MgL\sin\theta(t).$$

This is a second-order differential equation. Here the input is the torque $\tau_i(t)$ and the output is the angle $\theta(t)$.

2.1.3 Electromechanical systems

A simple electromechanical system is an armature-controlled direct current (DC) motor with a load, shown in Figure 2.7.

A DC motor has two sets of windings. One set is mounted on the stator and is used to generate the magnetic field. In an armature-controlled DC motor,

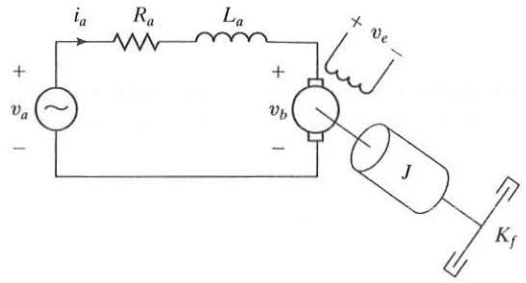

FIGURE 2.7: An armature-controlled DC motor system.

the current to this set of windings is set to be constant so that the magnetic field in the motor is constant. The other set is mounted on the rotor and is used to generate the torque through the magnetic force. The current through this winding is controllable so a controlled torque can be obtained. When the motor shaft turns, the magnetic field also generates a potential in the rotor winding as a result of the Faraday induction. This potential is called the back electro-motive force (back emf). There are two basic relations in a DC motor. One is that the torque in the motor shaft is proportional to the armature current via the torque constant K_t, i.e.,

$$\tau_m(t) = K_t i_a(t).$$

The other is that the back emf $v_b(t)$ is proportional to the motor velocity $\omega(t)$ via the back emf constant K_b, i.e.,

$$v_b(t) = K_b \omega(t).$$

The torque in the motor shaft then drives the load, which consists of a mass with a moment of inertia J and a counteractive friction torque proportional to the motor velocity via the friction coefficient K_f. Therefore, the whole DC motor system including the armature circuit and the mechanical load can be described by the following parameters and variables:

R_a:	armature resistance
L_a:	armature inductance
J:	moment of inertia of the load
K_f:	friction coefficient
K_t:	torque constant
K_b:	back emf constant
$v_a(t)$:	armature voltage
$i_a(t)$:	armature current
$v_b(t)$:	back electro-motive force (back emf)
$\tau_m(t)$:	motor torque
$\theta(t)$:	angular position of the motor shaft
$\omega(t)$:	angular velocity of the motor shaft $(= \dot{\theta}(t))$

Applying KVL to the armature circuit, we obtain

$$R_a i_a(t) + L_a \frac{di_a(t)}{dt} + K_b \frac{d\theta(t)}{dt} = v_a(t). \tag{2.12}$$

Applying the rotational version of Newton's second law to the motor shaft, we obtain

$$J\frac{d^2\theta(t)}{dt^2} + K_f\frac{d\theta(t)}{dt} = K_t i_a(t). \tag{2.13}$$

These two equations give the mathematical model of the DC motor system with input $v_a(t)$ and output $\theta(t)$.

In some applications, we are concerned with the angular velocity (speed) of the motor, instead of the angular position. Such cases are called **speed control cases**. Replacing $\dfrac{d\theta(t)}{dt}$ by $\omega(t)$ in (2.12) and (2.13), we get the differential equation model of a DC motor system in the speed control case.

$$R_a i_a(t) + L_a\frac{di_a(t)}{dt} + K_b\omega(t) = v_a(t) \tag{2.14}$$

$$J\frac{d\omega(t)}{dt} + K_f\omega(t) = K_t i_a(t). \tag{2.15}$$

These two equations give the mathematical model of the DC motor system with input $v_a(t)$ and output $\omega(t)$.

Another interesting electromechanical system is a magnetic-ball suspension system shown in Figure 2.8. The coil at the top, after being fed with current, produces a magnetic field. The magnetic field generates an attracting force on the steel ball.

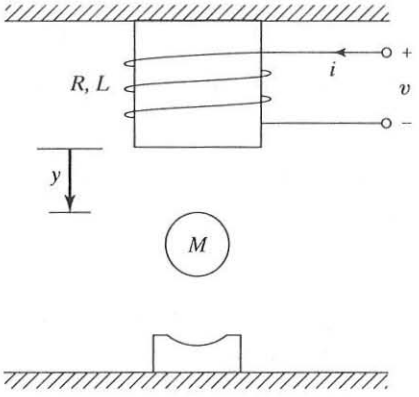

FIGURE 2.8: A magnetic-ball suspension system.

Here, the voltage applied to the coil $v(t)$ is the input, the distance from the ball to the coil $y(t)$ is the output, and the lifting force generated by the magnetic field on the ball is approximately given by $f(t) = K\dfrac{i^2(t)}{y(t)}$. The other parameters are mass of the ball M, winding resistance R, and winding inductance L.

Applying KVL to the coil, we obtain

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

Applying Newton's second law to the ball, we obtain

$$M\frac{d^2y(t)}{dt^2} = -K\frac{i^2(t)}{y(t)} + Mg.$$

These two equations then give the mathematical model of the system. It is noted that the variables $y(t)$ and $i(t)$ are involved nonlinearly in the equations and this system is called a **nonlinear system**.

2.2 STATE SPACE MODEL AND LINEARIZATION

The mathematical models obtained in the previous sections consist of sets of differential equations with different orders and these equations involve variables other than the inputs and outputs. For the sake of systematic study, we need to put them in standard forms. One of the commonly used standard forms is the state space form.

Definition 2.1. The **state variables** of a system are a set of independent variables whose values at time t_0, together with input for all $t \geq t_0$, determine the behavior of the system for all $t \geq t_0$.

This definition looks very abstract but in many situations the state variables can be chosen intuitively. For electrical circuits, we can always choose the voltages across independent capacitors and the currents through independent inductors as state variables. For mechanical systems, we can always choose the positions and velocities of independent rigid bodies as state variables. Suppose that a differential equation model of a system is already obtained, the variables in the differential equations are the input $u(t)$ and the internal variables $v_1(t), \ldots, v_p(t)$, and the highest order of the derivatives of $v_i(t)$ in the differential equations is q_i. Then we can choose

$$v_i(t), \dot{v}_i(t), \ddot{v}_i(t), \ldots, v_i^{(q_i-1)}(t) \qquad i = 1, 2, \ldots, p$$

as the state variables. In this case, the total number of state variables is $\sum_{i=1}^{p} q_i$.

After the state variables are chosen, usually named $x_1(t), x_2(t), \ldots, x_n(t)$, we put them into a vector

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \in \mathbb{R}^n.$$

This vector is called a **state vector**. Here and in the sequel, we use bold font letters to denote vectors (or matrices) and vector-valued (or matrix-valued) functions, whereas we use normal font letters to denote scalars and scalar-valued functions. Then the set of mixed ordered differential equations can be converted into a set of first-order differential equations plus an algebraic equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), u(t), t] \tag{2.16}$$

$$y(t) = g[\boldsymbol{x}(t), u(t), t] \tag{2.17}$$

where $u(t) \in \mathbb{R}$ is the input, $y(t) \in \mathbb{R}$ is the output,

$$\boldsymbol{f}[\boldsymbol{x}(t), u(t), t] = \begin{bmatrix} f_1[\boldsymbol{x}(t), u(t), t] \\ f_2[\boldsymbol{x}(t), u(t), t] \\ \vdots \\ f_n[\boldsymbol{x}(t), u(t), t] \end{bmatrix} : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^n$$

is a vector-valued function, and

$$g[\boldsymbol{x}(t), u(t), t] : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

is a scalar-valued function. The number of state variables n is called the **order** of the system. The set of first-order differential equations (2.16) is called the **state equation** of the system. The algebraic equation (2.17) is called the **output equation** of the system. Together they form the state space model of the system.

We always assume that the system starts operation at time $t = 0$; namely, we assume that the input $u(t)$ is a unilateral signal whose value before the initial time is zero. To determine the state vector $\boldsymbol{x}(t)$ from the differential equation (2.16), the input $u(t)$ alone is not sufficient. According to Definition 2.1, the initial value of the state $\boldsymbol{x}(0)$ is also needed. This initial value is called the **initial condition**. To conform to our standard mathematical treatment of signals, we also view $\boldsymbol{x}(t)$ as a unilateral function. If the initial condition is nonzero, then $\boldsymbol{x}(t)$ has a jump discontinuity at $t = 0$ and its derivative $\dot{\boldsymbol{x}}(t)$ contains impulse functions.

Among the models of the systems discussed in Section 2.1, the one for the active RLC circuit has already been put in the state space form.

EXAMPLE 2.2

For the pendulum system introduced in the last section, the differential equation model directly obtained from Newton's second law is

$$ML^2\ddot{\theta}(t) = \tau_i(t) - MgL\sin\theta(t)$$

with input $\tau_i(t)$ and output $\theta(t)$. Renaming $x_1(t) = \theta(t), x_2(t) = \dot{\theta}(t)$, $u(t) = \tau_i(t)$, $y(t) = \theta(t)$, we obtain the state space model

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ \dfrac{-MgL\sin x_1(t) + u(t)}{ML^2} \end{bmatrix}$$

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

with

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} \theta(0) \\ \dot{\theta}(0) \end{bmatrix}.$$

EXAMPLE 2.3

The magnetic suspension system has the differential equation model

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

$$M\frac{d^2y(t)}{dt^2} = -K\frac{i^2(t)}{y(t)} + Mg.$$

Choose $x_1(t) = i(t), x_2(t) = y(t), x_3(t) = \dot{y}(t), u(t) = v(t)$, and we get the state space model

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} \dfrac{u(t) - Rx_1(t)}{L} \\ x_3(t) \\ -\dfrac{Kx_1^2(t)}{Mx_2(t)} + g \end{bmatrix}$$

$$y(t) = x_2(t)$$

with

$$\begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} i(0) \\ y(0) \\ \dot{y}(0) \end{bmatrix}.$$

Definition 2.4. A system is said to be **linear** if it can be described by linear differential equations, in particular, if the functions $\boldsymbol{f}$ and g in its state space model are linear functions of $\boldsymbol{x}(t)$ and $u(t)$.

For a linear system, the state space model takes the following matrix form:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{b}(t)u(t)$$

$$y(t) = \boldsymbol{c}(t)\boldsymbol{x}(t) + d(t)u(t)$$

where $\boldsymbol{A}(t) \in \mathbb{R}^{n\times n}$ is an $n \times n$ matrix, possibly depending on time t, and $\boldsymbol{b}(t) \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{c}(t) \in \mathbb{R}^{1\times n}$ are, respectively, column and row vectors depending possibly on time t. For example, the RLC circuit in Section 2.1 is a linear system and its state space equations were already in the matrix form as in (2.7) and (2.8)

Theorem 2.5 (Superposition Principle). Assume that a linear system has zero initial condition. If input $u_1(t)$ produces output $y_1(t)$ and input $u_2(t)$ produces output $y_2(t)$, then input $\alpha_1 u_1(t) + \alpha_2 u_2(t)$ produces output $\alpha_1 y_1(t) + \alpha_2 y_2(t)$ for all $\alpha_1, \alpha_2 \in \mathbb{R}$.

Proof. With zero initial condition, if input $u_1(t)$ produces output $y_1(t)$ and input $u_2(t)$ produces output $y_2(t)$, then there are $\boldsymbol{x}_1(t)$ and $\boldsymbol{x}_2(t)$ with $\boldsymbol{x}_1(0) = \boldsymbol{0}$ and $\boldsymbol{x}_2(0) = \boldsymbol{0}$ satisfying

$$\dot{\boldsymbol{x}}_1(t) = \boldsymbol{A}(t)\boldsymbol{x}_1(t) + \boldsymbol{b}(t)u_1(t)$$

$$y_1(t) = \boldsymbol{c}(t)\boldsymbol{x}_1(t) + d(t)u_1(t)$$

and

$$\dot{\boldsymbol{x}}_2(t) = \boldsymbol{A}(t)\boldsymbol{x}_2(t) + \boldsymbol{b}(t)u_2(t)$$
$$y_2(t) = \boldsymbol{c}(t)\boldsymbol{x}_2(t) + d(t)u_2(t).$$

If we add the two state equations and the two output equations, respectively, and define $u(t) = \alpha_1 u_1(t) + \alpha_2 u_2(t)$, $\boldsymbol{x}(t) = \alpha_1 \boldsymbol{x}_1(t) + \alpha_2 \boldsymbol{x}_2(t)$, and $y(t) = \alpha_1 y_1(t) + \alpha_2 y_2(t)$, then we obtain $\boldsymbol{x}(0) = \boldsymbol{0}$ and

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{b}(t)u(t)$$
$$y(t) = \boldsymbol{c}(t)\boldsymbol{x}(t) + d(t)u(t).$$

This implies that $y(t)$ is the output of the system with zero initial condition and input $u(t)$. □

Definition 2.6. A system is said to be **time-invariant** if it can be described by differential equations with constant coefficients, in particular, if the functions $\boldsymbol{f}$ and g in its state space model do not depend on the time t explicitly.

All examples of real physical systems considered so far are time-invariant systems.

Theorem 2.7. Assume that a time-invariant system has zero initial condition. Also assume that zero input generates zero output. If input $u(t)$ produces output $y(t)$, then input $u(t-\tau)$ produces output $y(t-\tau)$ for all $\tau \geq 0$.

A linear time-invariant (LTI) system has the following form of state space model:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}u(t)$$
$$y(t) = \boldsymbol{c}\boldsymbol{x}(t) + du(t)$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{b} \in \mathbb{R}^{n\times 1}$, $\boldsymbol{c} \in \mathbb{R}^{1\times n}$, and $d \in \mathbb{R}$ are constant matrices.

EXAMPLE 2.8

The DC motor system is an LTI system. Indeed, in the position control case, if we choose the state variables, input and output variables as $x_1(t) = i_a(t), x_2(t) = \theta(t), x_3(t) = \omega(t), u(t) = v_a(t), y(t) = \theta(t)$, then the state space equation can be written as

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} -\dfrac{R_a}{L_a} & 0 & -\dfrac{K_b}{L_a} \\ 0 & 0 & 1 \\ \dfrac{K_t}{J} & 0 & -\dfrac{K_f}{J} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} \dfrac{1}{L_a} \\ 0 \\ 0 \end{bmatrix} u(t)$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}.$$

In the speed control case, if we choose $x_1(t) = i_a(t), x_2(t) = \omega(t), u(t) = v_a(t)$, $y(t) = \omega(t)$, then we obtain

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_b}{L_a} \\ \frac{K_t}{J} & -\frac{K_f}{J} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} u(t)$$

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

In the rest of this section, we will study how to approximate a nonlinear system by a linear one. Such a process is called **linearization**.

We will deal only with time-invariant systems. Assume that a system is described by a state space model.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}[\boldsymbol{x}(t), u(t)]$$
$$y(t) = g[\boldsymbol{x}(t), u(t)]$$

and $\boldsymbol{f}$ and g are continuously differentiable functions, i.e., $\boldsymbol{f}$ and g are sufficiently smooth functions.

Definition 2.9. A triple of constant vectors $(u_0, \boldsymbol{x}_0, y_0) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}$ is said to be an **operating point** of the system if

$$0 = \boldsymbol{f}(\boldsymbol{x}_0, u_0)$$
$$y_0 = g(\boldsymbol{x}_0, u_0).$$

The physical meaning of an operating point is that if the system has initial condition $\boldsymbol{x}_0$ and a constant input u_0 is applied, then the state and output will stay at constant values $\boldsymbol{x}_0$ and y_0, respectively, for all time, i.e.,

$$u(t) = u_0, \boldsymbol{x}(0) = \boldsymbol{x}_0 \Longrightarrow \boldsymbol{x}(t) = \boldsymbol{x}_0, y(t) = y_0.$$

Since $\boldsymbol{f}$ and g are sufficiently smooth, we can conclude that

$$u(t) - u_0 \text{ and } \boldsymbol{x}(0) - \boldsymbol{x}_0 \text{ are small} \Longrightarrow \boldsymbol{x}(t) - \boldsymbol{x}_0 \text{ and } y(t) - y_0 \text{ are small.}$$

Denote

$$\tilde{u}(t) = u(t) - u_0$$
$$\tilde{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_0$$
$$\tilde{y}(t) = y(t) - y_0.$$

Replace $\boldsymbol{f}$ and g by their differentials:

$$\dot{\tilde{\boldsymbol{x}}}(t) = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} \tilde{\boldsymbol{x}}(t) + \left.\frac{\partial \boldsymbol{f}}{\partial u}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} \tilde{u}(t) + \text{high-order terms}$$

$$\tilde{y}(t) = \left.\frac{\partial g}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} \tilde{\boldsymbol{x}}(t) + \left.\frac{\partial g}{\partial u}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} \tilde{u}(t) + \text{high-order terms}$$

where

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix}, \quad \frac{\partial \boldsymbol{f}}{\partial u} = \begin{bmatrix} \dfrac{\partial f_1}{\partial u} \\ \vdots \\ \dfrac{\partial f_n}{\partial u} \end{bmatrix}, \quad \frac{\partial g}{\partial \boldsymbol{x}} = \begin{bmatrix} \dfrac{\partial g}{\partial x_1} & \cdots & \dfrac{\partial g}{\partial x_n} \end{bmatrix}.$$

Since $\tilde{u}(t), \tilde{\boldsymbol{x}}(t), \tilde{y}(t)$ are small, we can neglect the high-order terms and approximate the original system by the following linear system:

$$\dot{\tilde{\boldsymbol{x}}}(t) = \boldsymbol{A}\tilde{\boldsymbol{x}}(t) + \boldsymbol{b}\tilde{u}(t)$$
$$\tilde{y}(t) = \boldsymbol{c}\tilde{\boldsymbol{x}}(t) + d\tilde{u}(t)$$

where

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}}, \qquad \boldsymbol{b} = \left.\frac{\partial \boldsymbol{f}}{\partial u}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}},$$
$$\boldsymbol{c} = \left.\frac{\partial g}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}}, \qquad d = \left.\frac{\partial g}{\partial u}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}}.$$

This linear system is called a **linearized system** of the original nonlinear system.

EXAMPLE 2.10

The magnetic suspension system is a nonlinear system described by state space equation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} \dfrac{u(t) - Rx_1(t)}{L} \\ x_3(t) \\ -\dfrac{x_1^2(t)}{Mx_2(t)} + g \end{bmatrix}$$
$$y(t) = x_2(t).$$

A usual control problem is to lift the ball to a certain height and suspend it at that height. Hence we wish to linearize it around an operating point with $y(t) = y_0$.

To get the operating point, solve equations

$$0 = \frac{u_0 - Rx_{10}}{L}$$
$$0 = x_{30}$$
$$0 = -\frac{x_{10}^2}{Mx_{20}} + g$$
$$y_0 = x_{20}.$$

This gives

$$u_0 = R\sqrt{Mgy_0}, \quad \begin{bmatrix} x_{10} \\ x_{20} \\ x_{30} \end{bmatrix} = \begin{bmatrix} \sqrt{Mgy_0} \\ y_0 \\ 0 \end{bmatrix}, \quad y_0 = y_0$$

which means that to suspend the ball at height y_0 in the steady state, one need to apply a constant voltage $u(t) = R\sqrt{Mgy_0}$ to the coil. Denote the deviations o the input, state, and output variables from the operating point by

$$\tilde{u}(t) = u(t) - u_0 = u(t) - R\sqrt{Mgy_0}$$

$$\tilde{\boldsymbol{x}}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_0 = \begin{bmatrix} x_1(t) - \sqrt{Mgy_0} \\ x_2(t) - y_0 \\ x_3(t) \end{bmatrix}$$

$$\tilde{y}(t) = y(t) - y_0.$$

Now, the linearized model of the deviation variables is

$$\dot{\tilde{\boldsymbol{x}}}(t) = \boldsymbol{A}\tilde{\boldsymbol{x}}(t) + \boldsymbol{b}\tilde{u}(t)$$

$$\tilde{y}(t) = \boldsymbol{c}\tilde{\boldsymbol{x}}(t) + d\tilde{u}(t)$$

where

$$\boldsymbol{A} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} = \begin{bmatrix} -\dfrac{R}{L} & 0 & 0 \\ 0 & 0 & 1 \\ -2\sqrt{\dfrac{g}{My_0}} & \dfrac{g}{y_0} & 0 \end{bmatrix}, \quad \boldsymbol{b} = \left.\frac{\partial \boldsymbol{f}}{\partial u}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} = \begin{bmatrix} \dfrac{1}{L} \\ 0 \\ 0 \end{bmatrix},$$

$$\boldsymbol{c} = \left.\frac{\partial g}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \qquad d = \left.\frac{\partial g}{\partial u}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ u=u_0}} = 0.$$

2.3 TRANSFER FUNCTIONS AND IMPULSE RESPONSES

Consider an LTI system described by state space equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{b}u(t)$$

$$y(t) = \boldsymbol{c}\boldsymbol{x}(t) + du(t).$$

Take the Laplace transform with zero initial conditions:

$$s\boldsymbol{X}(s) = \boldsymbol{A}\boldsymbol{X}(s) + \boldsymbol{b}U(s) \tag{2.18}$$

$$Y(s) - \boldsymbol{c}\boldsymbol{X}(s) + dU(s). \tag{2.19}$$

Now a set of differential equations in the time domain becomes a set of algebraic equations in the frequency domain. There are a total of $n + 1$ equations in (2.18)–(2.19) and we can use them to eliminate the n variables in $\boldsymbol{X}(s)$ to obtain an equation relating the input $U(s)$ and the output $Y(s)$. Linear algebra now gives

a formal method of capturing this process. From (2.18), we get

$$\boldsymbol{X}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b}U(s).$$

Plugging it into (2.19), we obtain

$$\frac{Y(s)}{U(s)} = \boldsymbol{c}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d,$$

i.e., the ratio of the Laplace transform of the output over that of the input is a fixed function independent of the input.

Definition 2.11. The **transfer function** of an LTI system is the ratio of the Laplace transform of the output over that of the input when the initial condition is zero, i.e.,

$$G(s) = \frac{Y(s)}{U(s)}.$$

EXAMPLE 2.12

Let us continue to consider the DC motor system in Example 2.8. In the position control case, the transfer function is

$$\begin{aligned} G(s) &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s + \frac{R_a}{L_a} & 0 & \frac{K_b}{L_a} \\ 0 & s & -1 \\ -\frac{K_t}{J} & 0 & s + \frac{K_f}{J} \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \end{bmatrix} \\ &= \frac{K_t/(L_a J)}{s(s + R_a/L_a)(s + K_f/J) + K_t K_b/(L_a J)s} \\ &= \frac{K_t}{L_a J s^3 + (R_a J + K_f L_a)s^2 + (R_a K_f + K_t K_b)s}. \end{aligned}$$

One may feel that the inverse of the 3×3 matrix is hard to compute, but the computation can be significantly simplified if one notices that only the element in the second row and the first column of the inverse is needed since all other elements will be multiplied by zero when forming the transfer function. Computing only one element is, of course, much simpler than computing all elements in the inverse. In the speed control case, the transfer function is

$$\begin{aligned} G(s) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s + \frac{R_a}{L_a} & \frac{K_b}{L_a} \\ -\frac{K_t}{J} & s + \frac{K_f}{J} \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} \\ &= \frac{K_t/(L_a J)}{(s + R_a/L_a)(s + K_f/J) + K_t K_b/(L_a J)} \\ &= \frac{K_t}{L_a J s^2 + (R_a J + K_f L_a)s + (R_a K_f + K_t K_b)}. \end{aligned}$$

For systems with a small number of state variables, it is probably more convenient to obtain the transfer function by directly manipulating the Laplace transform of the state space model (2.18) and (2.19). For example, in the speed control case, the Laplace transform of the state space model is

$$\begin{aligned} sX_1(s) &= -\frac{R_a}{L_a}X_1(s) - \frac{K_b}{L_a}X_2(s) + \frac{1}{L_a}U(s) \\ sX_2(s) &= \frac{K_t}{J}X_1(s) - \frac{K_f}{J}X_2(s) \\ Y(s) &= X_2(s). \end{aligned}$$

Substitute $X_1(s)$ from the second equation into the first equation and note that $Y(s) = X_2(s)$ from the third equation. We then get

$$\left[\frac{J}{K_t}\left(s + \frac{R_a}{L_a}\right)\left(s + \frac{K_f}{J}\right) + \frac{K_b}{L_a}\right] Y(s) = \frac{1}{L_a}U(s).$$

Consequently,

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_t}{L_a J s^2 + (R_a J + K_f L_a)s + (R_a K_f + K_t K_b)},$$

the same result as the one obtained by matrix inversion.

EXAMPLE 2.13

Let us continue with Example 2.10, the magnetic suspension system. The transfer function of the linearized model is

$$G(s) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s + \frac{R}{L} & 0 & 0 \\ 0 & s & -1 \\ 2\sqrt{\frac{g}{My_0}} & -\frac{g}{y_0} & s \end{bmatrix}^{-1} \begin{bmatrix} \frac{1}{L} \\ 0 \\ 0 \end{bmatrix}$$

$$= \frac{-2\sqrt{\frac{g}{My_0}}\frac{1}{L}}{\left(s + \frac{R}{L}\right)\left(s^2 - \frac{g}{y_0}\right)} = \frac{-2\sqrt{gy_0}}{\sqrt{M}(Ls + R)(y_0 s^2 - g)}.$$

The transfer function of an LTI system with a state space model is always a ratio of two polynomials

$$G(s) = \frac{b(s)}{a(s)}$$

where $b(s)$ is called the numerator polynomial and $a(s)$ is called the denominator polynomial. We assume that polynomials $b(s)$ and $a(s)$ are **coprime**, i.e., they do not have common factors.

The ratio of two polynomials is also called a **rational function**. So the transfer function of an LTI system with a state space model is a rational function. We often denote a transfer function, or any rational function, in either of the following forms

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

or

$$G(s) = K \frac{(s - z_1)(s - z_2) \cdots (s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_n)}.$$

The first form is called the **unfactored form** and the second form is called the **factored form**. Here $z_1, z_2, \ldots, z_m$, the roots of $b(s)$, are called the **zeros** of $G(s)$ and $p_1, p_2, \ldots, p_n$, the roots of $a(s)$, are called the **poles** of $G(s)$. K is called the **(high frequency) gain** of $G(s)$. The factored form is also called the **zero-pole-gain form**.

Several additional definitions will be needed. A transfer function or system $G(s)$ is said to be **proper** if $\deg b(s) \leq \deg a(s)$, or equivalently $|G(\infty)| < \infty$. It is said to be **strictly proper** if $\deg b(s) < \deg a(s)$, or equivalently $G(\infty) = 0$. It is said to be **bi-proper** if $\deg b(s) = \deg a(s)$, or equivalently $0 \neq |G(\infty)| \neq \infty$. Transfer functions obtained from state space models are always proper, but occasionally nonproper transfer functions appear in abnormal cases. For a proper transfer function, the difference $\deg a(s) - \deg b(s)$ is called the **relative degree** of $G(s)$, and $\deg a(s)$ is called the **order** or **degree** of $G(s)$. $G(0)$ is called the **DC gain** of $G(s)$. Notice the difference between (high frequency) gain and DC gain.

In the transfer function of a state space model

$$G(s) = \boldsymbol{c}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d = \frac{\boldsymbol{c}\,\mathrm{adj}(s\boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{b}}{\det(s\boldsymbol{I} - \boldsymbol{A})} + d$$

where adj means the adjugate of a matrix (see Appendix B), if there is no common factor on the above denominator and numerator, then $a(s) = \det(s\boldsymbol{I} - \boldsymbol{A})$, i.e., $a(s)$ is the characteristic polynomial of matrix $\boldsymbol{A}$, and the poles of the system are the eigenvalues of $\boldsymbol{A}$. If there are common factors on the above denominator and numerator, then $a(s)$ is only a factor of $\det(s\boldsymbol{I} - \boldsymbol{A})$ and the poles of the system are part of the eigenvalues of $\boldsymbol{A}$. In this case, some of the eigenvalues of $\boldsymbol{A}$ do not appear as the poles of the transfer function $G(s)$. They become hidden from the transfer function and hence are called the **hidden poles**. This again is an abnormal phenomenon and is prone to trouble. Extra care needs to be taken in this case. We will assume that this does not happen in our development.

In MATLAB, one can represent a polynomial

$$p(s) = p_0 s^n + p_1 s^{n-1} + \cdots + p_n$$

by a vector

$$\boldsymbol{p} = \begin{bmatrix} p_0 & p_1 & \cdots & p_n \end{bmatrix}.$$

To find the roots of $p(s)$, we can type

```
>> roots(p)
```

One often needs to compute the sums and products of polynomials. Let

$$p(s) = s^3 + 2s^2 + 3s + 4, \quad q(s) = 5s + 6.$$

Represent them in MATLAB by

```
>> p=[1 2 3 4];
>> q=[5 6];
```

However, neither of the following commands

```
>> p+q
>> p*q
```

would give you what you want since the first requires the two vectors to have the same dimension and the second is, in general, not defined at all. For polynomial addition, one has to augment either p or q with enough zeros so that they have the same dimension. For the example above, we should do

```
>> p+[0 0 q];
```

For polynomial multiplication, one has to use the command "conv":

```
>> conv(p,q)
```

One may find these inconvenient and counterintuitive. In Section 2.6, we will present an alternative way of representing and operating polynomials in MATLAB.

Since the transfer function $G(s)$ of an LTI system is the ratio of the output Laplace transform $Y(s)$ and the input Laplace transform $U(s)$, if $U(s) = 1$, i.e., $u(t) = \delta(t)$, then $Y(s) = G(s)$ and $y(t) = \mathcal{L}^{-1}[G(s)] = g(t)$. Hence the inverse Laplace transform of $G(s)$, denoted by $g(t)$, is called the **impulse response**.

2.4 SIMPLIFYING BLOCK DIAGRAMS

Interconnected systems are often conveniently represented by block diagrams. For example, the system in the last section $Y(s) = G(s)U(s)$ can be represented using the block diagram in Figure 2.9.

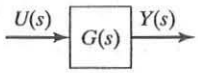

FIGURE 2.9: Block diagram representation.

Block diagrams are particularly useful when dealing with complex systems consisting of collections of interconnected subsystems. They can be simplified using the equivalence relationships in Table 2.1.

We shall now see how a complex system block diagram can be simplified.

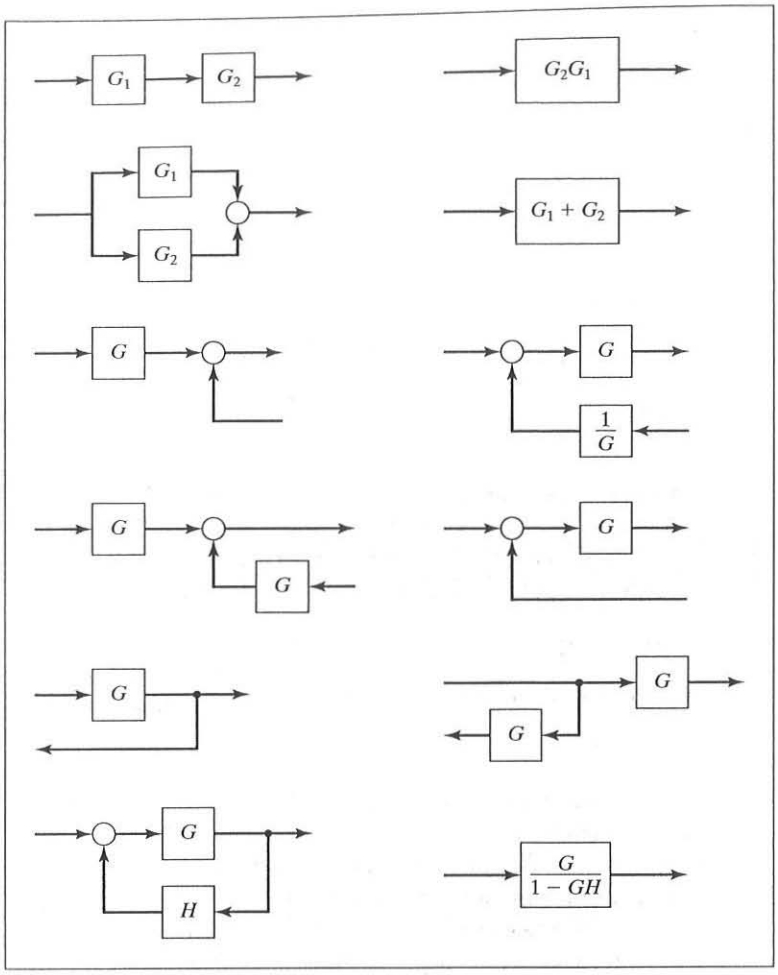

TABLE 2.1: Equivalent block diagrams.

EXAMPLE 2.14

Consider the simple feedback system shown in Figure 2.10. To find the relationship between r and y, we shall write down all the equations:

$$Y(s) = P(s)U(s), \quad U(s) = C(s)E(s), \quad E(s) = F(s)R(s) - H(s)Y(s).$$

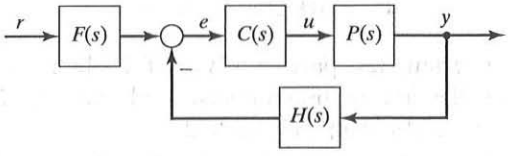

FIGURE 2.10: A simple feedback system.

We shall now eliminate the intermediate variables, $E(s)$ and $U(s)$, to get

$$Y(s) = P(s)C(s)\left[F(s)R(s) - H(s)Y(s)\right].$$

Thus solving $Y(s)$, we get

$$Y(s) = \frac{P(s)C(s)F(s)}{1 + P(s)C(s)H(s)}R(s)$$

i.e., the transfer function from $R(s)$ to $Y(s)$ is given by

$$\frac{Y(s)}{R(s)} = \frac{P(s)C(s)F(s)}{1 + P(s)C(s)H(s)}.$$

EXAMPLE 2.15

Consider the feedback control system shown in Figure 2.11. Here we omitted the variable s in the transfer function notation to make the expressions more compact.

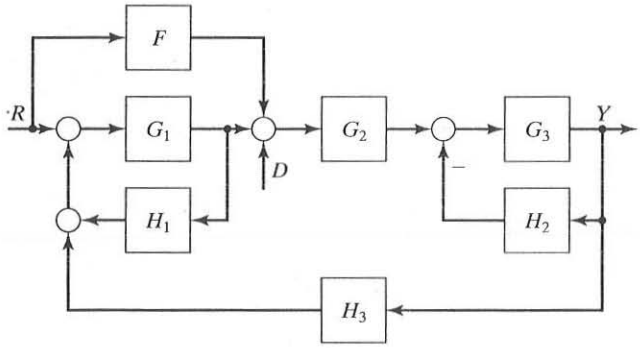

FIGURE 2.11: Original diagram of Example 2.15.

We shall compute the transfer function from R to Y. Thus, we shall assume $D = 0$ and we can simplify the block diagram by first closing the two inner loops: $G_1 \to H_1 \to G_1$ loop and $G_3 \to H_2 \to G_3$ loop, which results in the block diagram in Figure 2.12. We then move the first summing junction to the place of the second summing junction to get the block diagram in Figure 2.13. Finally, closing the loop, we have

$$\frac{Y}{R} = \left(F + \frac{G_1}{1 - G_1H_1}\right) \frac{\dfrac{G_2G_3}{1 + G_3H_2}}{1 - \dfrac{G_2G_3}{1 + G_3H_2}\dfrac{G_1H_3}{1 - G_1H_1}}$$

$$= \frac{G_2G_3(F - FG_1H_1 + G_1)}{1 - G_1H_1 + G_3H_2 - G_1G_3H_1H_2 - G_1G_2G_3H_3}.$$

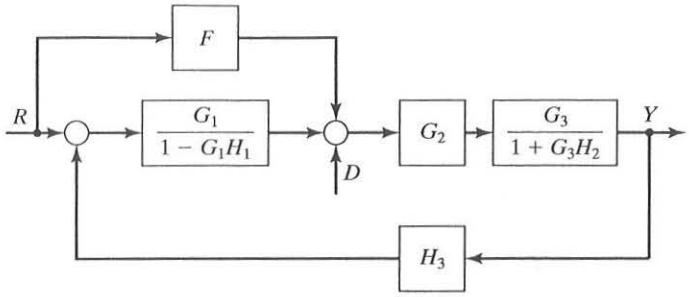

FIGURE 2.12: Block diagram of Example 2.15 with inner loops closed.

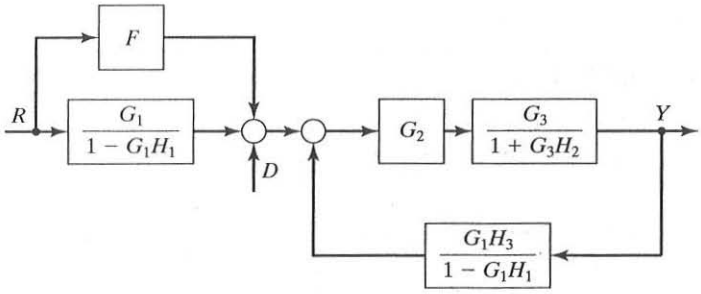

FIGURE 2.13: Further simplified block diagram of Example 2.15.

We can also find the transfer function from D to Y as

$$\frac{Y}{D} = \frac{\dfrac{G_2G_3}{1+G_3H_2}}{1-\dfrac{G_2G_3}{1+G_3H_2}\dfrac{G_1H_3}{1-G_1H_1}}$$

$$= \frac{G_2G_3(1-G_1H_1)}{1-G_1H_1+G_3H_2-G_1G_3H_1H_2-G_1G_2G_3H_3}.$$

In general, block diagrams can always be simplified by using the block diagram algebra as shown in Table 2.1.

2.5 TRANSFER FUNCTION MODELING

The modeling of complicated interconnected LTI systems can be done in the Laplace transform domain using transfer functions and block diagrams in cases when the transfer functions of the subsystems are known.

Let us use an armature-controlled DC motor with a load torque, shown in Figure 2.7, as an example to see how this can be done. Let us consider the case when the load not only contains an inertia torque proportional to the angular acceleration and a friction torque proportional to the angular velocity, but also includes

a nonzero possibly time-varying torque $\tau_d(t)$ independent of the angular position, velocity, and acceleration. Such a system can be considered as an interconnected system with electrical, magnetic, and mechanical subsystems.

First notice that in the electrical part, we have

$$I_a(s) = \frac{1}{L_a s + R_a}[V_a(s) - V_b(s)]. \tag{2.20}$$

Then the torque generated by the motor is given by

$$T_m(s) = K_t I_a(s). \tag{2.21}$$

We also know that the back emf voltage is given by

$$V_b(s) = K_b \Omega(s). \tag{2.22}$$

The mechanical part has relations

$$\Omega(s) = \frac{1}{Js + K_f}[T_m(s) - T_d(s)] \tag{2.23}$$

and

$$\Theta(s) = \frac{1}{s}\Omega(s)$$

where $T_d(s)$ is the Laplace transform of a possible load torque. Combining all of these equations, we can see that the block diagram of the whole system is as in Figure 2.14. Simplifying the block diagram gives

$$\Theta(s) = \frac{1}{s\,[(L_a s + R_a)(Js + K_f) + K_t K_b]} \begin{bmatrix} K_t & -(L_a s + R_a) \end{bmatrix} \begin{bmatrix} V_a(s) \\ T_d(s) \end{bmatrix}.$$

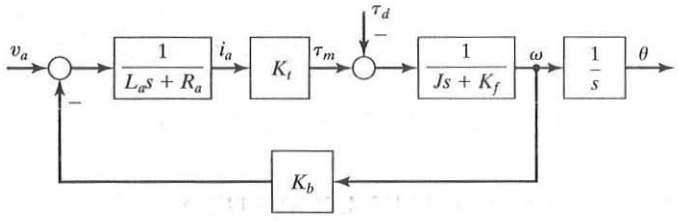

FIGURE 2.14: Block diagram of an armature-controlled DC motor.

Another illustrative example is a field-controlled DC motor shown in Figure 2.15. A field-controlled DC motor has the same structure as an armature-controlled DC motor. The difference is that in the field-controlled case, the armature current $i_a(t)$ is set to be constant but the field circuit is used to control the varying torque. In this case, the torque is related to the field current as

$$\tau_m(t) = K_t i_f(t) \tag{2.24}$$

where $i_f(t)$ is the field current. In the Laplace domain, we have

$$T_m(s) = K_t I_f(s) \tag{2.25}$$

where $I_f(s)$ is the Laplace transform of $i_f(t)$.

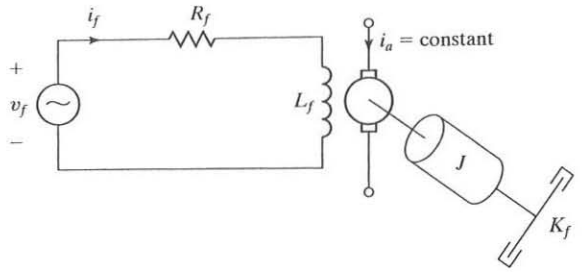

FIGURE 2.15: A field-controlled DC motor system.

The field current is generated by a field voltage through a field circuit and satisfies the following equation:

$$v_f(t) = L_f \frac{di_f(t)}{dt} + R_f i_f(t) \tag{2.26}$$

which, in terms of Laplace transforms, gives

$$\frac{I_f(s)}{V_f(s)} = \frac{1}{L_f s + R_f}. \tag{2.27}$$

Combining equations (2.23), (2.25), and (2.27), we get

$$\Theta(s) = \frac{1}{(L_f s + R_f)(Js + K_f)s} \begin{bmatrix} K_t & -(L_f s + R_f) \end{bmatrix} \begin{bmatrix} V_f(s) \\ T_d(s) \end{bmatrix}.$$

An interconnection block diagram for the system is shown in Figure 2.16.

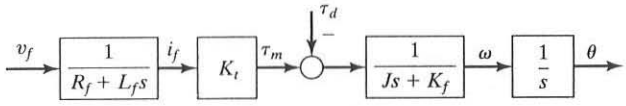

FIGURE 2.16: Block diagram of a field-controlled DC motor.

2.6 MATLAB MANIPULATION OF LTI SYSTEMS

In the MATLAB Control Systems Toolbox, a system can be represented by a single variable, no matter whether it is described by a state space model or by transfer function model. Suppose that we have a system described by its state space model:

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

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

and we wish to name it as F. Then the following sequence of commands assigns the variable F with its state space description:

```
>> A=[0 1; -1 -2];
>> B=[0 ; 1];
```

```
>> C=[2 1];
>> D=0;
>> F=ss(A,B,C,D);
```

Suppose we now have a system described by its transfer function model

$$\frac{s+2}{s^2+2s+1}$$

and we wish to name it as G. The following sequence of commands assigns the variable G with its transfer function description:

```
>> num=[1 2];
>> den=[1 2 1];
>> G=tf(num,den);
```

The different descriptions of a system can be easily converted from one to another. The command

```
>> F=tf(F);
```

converts the description of F from state space to transfer function. By doing this we can find that F and G are actually the same system since they have the same transfer function. Also the command

```
>> G=ss(G);
```

converts the description of G from transfer function to state space. However, we do not necessarily get exactly the same state space description as the original F although G and F have the same transfer function. This is because a system may have different state space descriptions. Now F is in transfer function form. Let us run

```
>> F=ss(F);
```

We will observe that this state space description of F is different from the original state space description of F, but is actually the same as the state space description of G obtained from the conversion. This is because by running the transfer function to state space conversion, the computer program chooses, among many possibilities, a particular canonical form of the state space description. If the very original state space description is not of this canonical form, then a state space to transfer function to state space conversion will not give the same thing back. The original state space description of F is forever lost after the "`ss`" to "`tf`" conversion.

For a system F in either the state space form or the transfer function form, commands

```
>> [A,B,C,D]=ssdata(F);
>> [num,den]=tfdata(F);
```

give back the parameter matrices of its state space description and the numerator and denomination coefficients of its transfer function respectively. Owing to the nonuniqueness of the state space model for a given transfer function, one may

wonder which choice the first command takes if F is in transfer function form. It turns out that the same canonical form as that chosen by the command "`ss(F)`" is also chosen here.

The use of the LTI system variable brings much convenience. To find out the poles and zeros of system F, instead of computing the denominator and numerator polynomials of transfer function $F(s)$ and then using the command "roots", we can simply do

```
>> pole(F);
```

and

```
>> zero(F);
```

Here the actual form of F is immaterial. To compute the sum and product of systems, we can simply run

```
>> F+G;
```

and

```
>> F*G;
```

Here F and G may take different forms. The following commands have obvious meanings:

```
>> F-G;
```

and

```
>> F/G;
```

When doing the last operation, one may run into trouble when G is strictly proper and either F or G is in state space form since the inverse of a strictly proper system cannot be represented by a state space system. Nevertheless, no problem will arise if both F and G are in transfer function form unless $G(s) = 0$. The feedback connection of two systems are computed easily:

```
>> feedback(F,G)
```

computes

$$\frac{F}{1+FG}$$

and

```
>> feedback(F,G,1)
```

computes

$$\frac{F}{1-FG}.$$

LTI system variables give another way of representing and operating polynomials, simply by interpreting them as transfer functions with 1 as the denominator

polynomials. In this case, the operations on polynomials, such as addition and multiplication, can be done in a way closer to natural language. Again, take

$$p(s) = s^3 + 2s^2 + 3s + 4, \quad q(s) = 5s + 6$$

as an example. The commands

```
>> p=tf([1 2 3 4],1);
>> q=tf([5 6],1);
```

assign p and q to be appropriate transfer functions with 1 as denominators, i.e., polynomials. Addition and multiplication can then be done in the following natural way without worrying about dimension compatibility and complications caused by vector multiplication:

```
>> p+q
>> p*q
```

2.7 SIMULATION AND IMPLEMENTATION OF SYSTEMS

We have seen how to model a physical system using differential equations and how to convert the model into a transfer function. In many situations, we also need to carry out the inverse process. For example, in system simulation, we often need to build a physical system with a given transfer function to observe the behavior of the system represented by the transfer function. In control implementation, we need to build a physical controller from the designed controller transfer function to connect it with the plant to form a feedback loop. The process of building a real physical system with a given transfer function is called **realization**. Unlike the modeling process of finding transfer functions from physical systems, the inverse realization process is highly nonunique, in terms of the kind of physical components used and the many possible configurations and structures. Although nowadays such a job is more and more accomplished by computer software, the traditional way of using hardware components is still of great theoretical and practical value. In the majority of such system simulation and implementation, op-amp circuits are used.

2.7.1 Hardware simulation and implementation

Let a system be given by a proper nth order transfer function

$$G(s) = \frac{b(s)}{a(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_n}{a_0 s^n + a_1 s^{n-1} + \cdots + a_n}, \quad a_0 \neq 0.$$

The op-amp circuit shown in Figure 2.17 gives a realization of $G(s)$. To show this, notice that

$$a_0 x^{(n)}(t) = -a_1 x^{(n-1)}(t) - \cdots - a_n x(t) + u(t).$$

Taking the Laplace transforms with zero initial conditions, we get

$$a_0 s^n X(s) = -a_1 s^{n-1} X(s) - \cdots - a_n X(s) + U(s).$$

This gives us

$$a(s)X(s) = U(s).$$

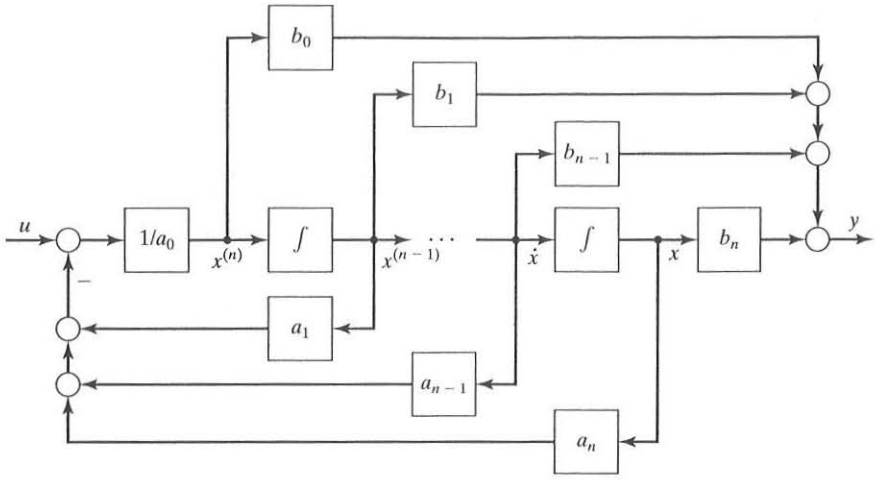

FIGURE 2.17: Controller form realization.

Also, notice that

$$y(t) = b_0 x^{(n)}(t) + b_1 x^{(n-1)}(t) + \cdots + b_n x(t).$$

Taking the Laplace transforms with zero initial conditions, we get

$$Y(s) = b_0 s^n X(s) + b_1 s^{n-1} X(s) + \cdots + b_n X(s) = b(s)X(s).$$

Hence

$$\frac{Y(s)}{U(s)} = \frac{b(s)}{a(s)}.$$

This realization is called a **controller form realization**.

The op-amp circuit in Figure 2.17 has a natural state space model. If we follow our tradition of assigning the voltages across capacitors, which are hidden in the integrators, then the state vector becomes

$$\boldsymbol{x}(t) = \begin{bmatrix} x^{(n-1)}(t) \\ \vdots \\ \dot{x}(t) \\ x(t) \end{bmatrix}.$$

The corresponding state space model is

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -\frac{a_1}{a_0} & \cdots & -\frac{a_{n-1}}{a_0} & -\frac{a_n}{a_0} \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} \frac{1}{a_0} \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} b_1 - b_0\frac{a_1}{a_0} & \cdots & b_{n-1} - b_0\frac{a_{n-1}}{a_0} & b_n - b_0\frac{a_n}{a_0} \end{bmatrix} \boldsymbol{x}(t) + \frac{b_0}{a_0}u(t).$$

This state space model, of course, gives a transfer function exactly the same as the given one.

Another realization of the same system is given by the circuit shown in Figure 2.18. To show this, notice that

$$\begin{aligned} a_0 y(t) &= x_1(t) + b_0 u(t) \\ \dot{x}_1(t) &= x_2(t) - a_1 y(t) + b_1 u(t) \\ &\vdots \\ \dot{x}_{n-1}(t) &= x_n(t) - a_{n-1} y(t) + b_{n-1} u(t) \\ \dot{x}_n(t) &= -a_n y(t) + b_n u(t). \end{aligned}$$

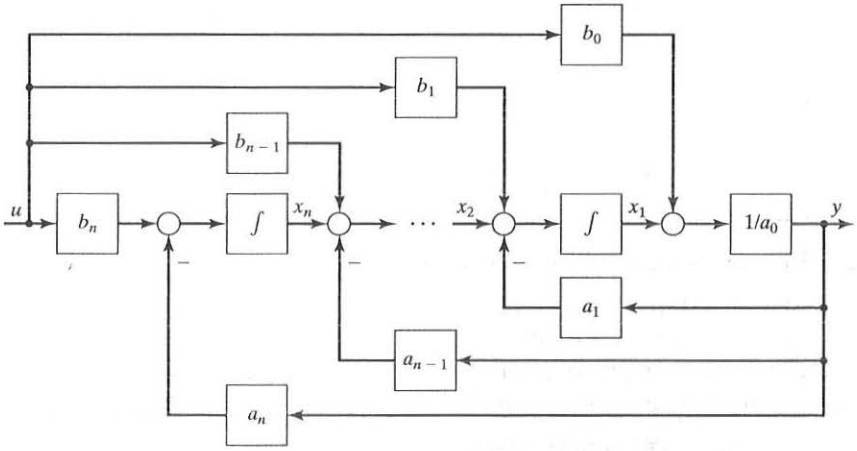

FIGURE 2.18: Observer form realization.

Taking the Laplace transforms with zero initial conditions, we get

$$\begin{aligned} a_0 Y(s) &= X_1(s) + b_0 U(s) \\ sX_1(s) &= X_2(s) - a_1 Y(s) + b_1 U(s) \\ &\vdots \\ sX_{n-1}(s) &= X_n(s) - a_{n-1} Y(s) + b_{n-1} U(s) \\ sX_n(s) &= -a_n Y(s) + b_n U(s). \end{aligned}$$

Multiplying the above equations by $s^n, s^{n-1}, \ldots, s, 1$, respectively, and adding them altogether, one can see that the variables $X_1(s), \ldots, X_n(s)$ are all cancelled and the resulting equation is

$$a(s)Y(s) = b(s)U(s).$$

So we also get

$$\frac{Y(s)}{U(s)} = \frac{b(s)}{a(s)}.$$

This realization is called the **observer form realization**. It also has a state space model. Let us again assign the voltage across capacitors as state variables. The state vector then becomes

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

The corresponding state space model is

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} -\frac{a_1}{a_0} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n-1}}{a_0} & 0 & \cdots & 1 \\ -\frac{a_n}{a_0} & 0 & \cdots & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} b_1 - b_0\frac{a_1}{a_0} \\ \vdots \\ b_{n-1} - b_0\frac{a_{n-1}}{a_0} \\ b_n - b_0\frac{a_n}{a_0} \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} \frac{1}{a_0} & 0 & \cdots & 0 \end{bmatrix} \boldsymbol{x}(t) + \frac{b_0}{a_0} u(t).$$

2.7.2 Software simulation and implementation

MATLAB provides certain tools for the numerical computation of system responses. For a system represented by a variable `G`, regardless of whether it is in the transfer function form or state space form, to compute its impulse response, i.e., its response to the unit impulse input $\delta(t)$, one can use

```
>> [y,t]=impulse(G);
```

To find its step response, i.e., its response to the unit step input $\sigma(t)$, one can use

```
>> [y,t]=step(G);
```

To calculate a system response with respect to a more general input signal than impulse and step, one can use a MATLAB command `lsim`. For example, the following sequence of commands gives the sinusoidal response of the system:

```
>> t=0:0.1:10;
>> u=sin(t);
>> y=lsim(G,u,t);
```

Another software product associated with MATLAB is SIMULINK. It can be used to simulate an interconnected system, represented by a block diagram. Let us demonstrate its use by a couple of examples.

EXAMPLE 2.16

Consider the unity feedback system shown in Figure 2.19 with a loop transfer function

$$L(s) = \frac{1}{s(s+1)}.$$

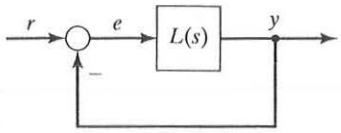

FIGURE 2.19: For Examples 2.16 and 2.17.

The time responses of this system with respect to various kinds of command signals are simulated using a SIMULINK diagram as shown in Figure 2.20 and are plotted in Figure 2.21.

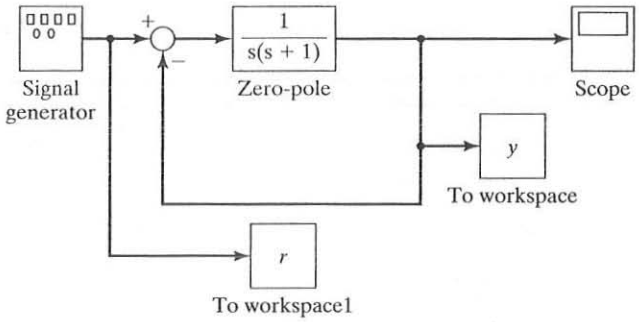

FIGURE 2.20: SIMULINK diagram for Example 2.16.

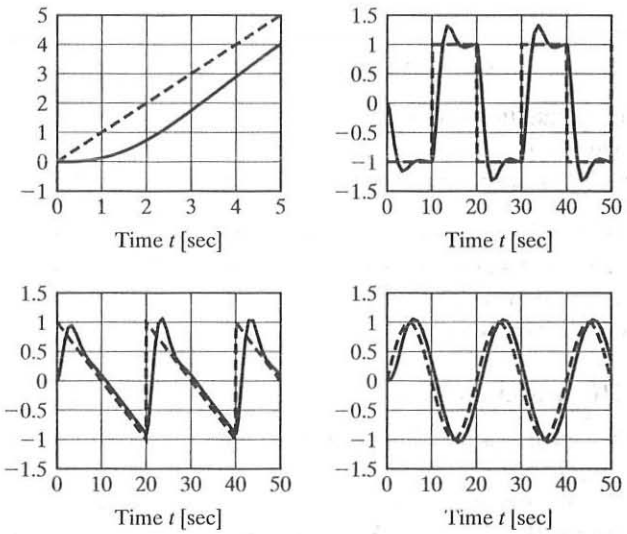

FIGURE 2.21: Responses to a ramp signal, a square wave, a sawtooth signal, and a sine wave for Example 2.16.