CHAPTER 5

LINEAR STATE-SPACE MODELS AND SOLUTIONS OF THE STATE EQUATIONS

Topics covered

- State-Space Models
- Solutions of the State Equations
- System Responses
- Sensitivity Analysis of the Matrix Exponential Problem
- Numerical Methods for Computing the Matrix Exponential and the Integral involving an Matrix Exponential
- Computation of the Frequency Response Matrix

5.1 INTRODUCTION

A finite-dimensional time-invariant linear continuous-time dynamical system may be described using the following system of first-order ordinary differential equations:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t),\\ y(t) &= Cx(t) + Du(t).\end{aligned}$$

The input and the output of the system are defined in continuous-time over the interval $[0, \infty)$. The system is, therefore, known as a **continuous-time system.** The discrete-time analog of this system is the system of difference equations:

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k),\\ y(k) &= Cx(k) + Du(k).\end{aligned}$$

We will consider in this book only time-invariant systems, that is, the matrices A, B, C, and D will be assumed constant matrices throughout the book.

It is first shown in Section 5.2 how some simple familiar physical systems can be described in state-space forms. Very often the mathematical model of a system is not obtained in first-order form; it may be a system of nonlinear equations, a system of **second-order differential equations** or **partial differential equations.** It is shown how such systems can be reduced to the standard first-order state-space forms. The computational methods for the state equations are then considered both in time and frequency domain.

The major computational component of the time-domain solution of a continuous-time system is the matrix exponential e^{At}. Some results on the sensitivity of this matrix and various well-known methods for its computation: the Taylor series method, the Padé approximation method, the methods based on decompositions of A, the ordinary-differential equation methods, etc., are described in Section 5.3. A comparative study of these methods is also included. **The Padé method (Algorithm 5.3.1) (with scaling and squaring)** and the **method, based on the Real Schur decomposition of A (Algorithm 5.3.2), are recommended for practical use.** This section concludes with an algorithm for numerically computing an integral with an matrix exponential (**Algorithm 5.3.3**).

Section 5.4 describes the state-space solution of a **discrete-time system**. The major computational task here is computation of various powers of A.

In **Section 5.5**, the problem of computing the **frequency response matrix** for many different values of the frequencies is considered. The computation of the frequency response matrix is necessary to study various system responses in frequency domain. A widely used method **(Algorithm 5.5.1)**, based on the one-time reduction of the state matrix A to a Hessenberg matrix, is described in detail and the references to the other recent methods are given.

Reader's Guide for Chapter 5

The readers familiar with basic concepts and results of modeling and state-space systems can skip Sections 5.2, 5.4, and 5.5.1.

5.2 STATE-SPACE REPRESENTATIONS OF CONTROL SYSTEMS

5.2.1 Continuous-Time Systems

Consider the dynamical system represented by means of the following system of ordinary first-order differential equations:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0, \tag{5.2.1}$$

$$y(t) = Cx(t) + Du(t). \tag{5.2.2}$$

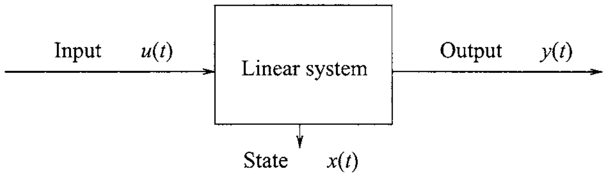


FIGURE 5.1: Representation of a continuous-time state-space model.

In this description,

> $x(t)$ is an n-dimensional vector, called the system **state**,
> $u(t)$ is an m-dimensional vector ($m \leq n$), called the system **input**,
> $y(t)$ is an r-dimensional vector, called the system **output**.

The vector $x(t_0)$ is the **initial condition** of the system. The components of $x(t)$ are called **state variables.**

The matrices A, B, C, and D are **time-invariant matrices**, respectively, of dimensions $n \times n, n \times m, r \times n$, and $r \times m$. The above representation is known as a time-invariant **continuous-time state-space model** of a dynamical system.

Schematically, the model is represented in Figure 5.1.

Clearly, at a given time t, the variables arriving at the system would form the input, those internal to the system form the state, while the others that can be measured directly comprise the output.

The space $X \subseteq \mathbb{R}^n$, where all the states lie for all $t \geq 0$ is called the **state-space**, the Eq. (5.2.1) is called the **state equation** and the Eq. (5.2.2) is called the **output equation**. If $m = r = 1$, the system is said to be a **single-input single-output (SISO) system. A multi-input multi-output (MIMO) system** is similarly defined. If a system has more than one input or more than one output it is referred to be a **multivariable** system. *The system represented by the Eqs.* (5.2.1) *and* (5.2.2) *is sometimes written compactly as (A, B, C, D) or as (A, B, C), in case D is not used in modeling. Sometimes $\dot{x}(t)$ and $x(t)$ will be written just as $\dot{x}$ and x for the sake of convenience.* Similarly, $u(t)$ and $y(t)$ will be written as u and y, respectively.

We provide below a few examples to illustrate the state-space representations of some simple systems.

Example 5.2.1 (A Parallel RLC Circuit). Consider a parallel RLC circuit excited by the current source $u(t)$ and with output $y(t)$ (Figure 5.2).

The current and voltage equations governing the circuit are:

$$u = i_R + i_L + i_C; \qquad i_C = C\frac{de_C}{dt}; \quad e_C = L\frac{di_L}{dt} = Ri_R.$$

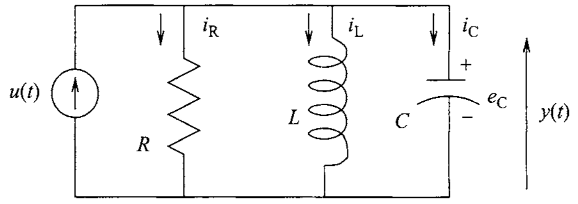


FIGURE 5.2: A parallel RLC circuit.

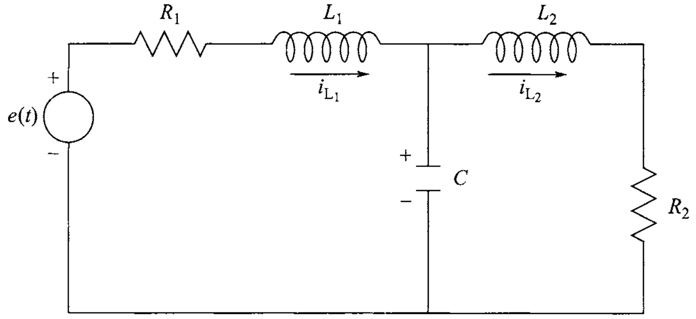


FIGURE 5.3: An expanded RLC circuit.

Defining the states by $x_1 := i_L$ and $x_2 := e_C$, the state and output equations are, respectively:

$$\dot{x} = Ax + bu \quad \text{and} \quad y = cx,$$

where $x = [x_1, x_2]^T$,

$$A = \begin{bmatrix} 0 & 1/L \\ -1/C & -1/RC \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 1/C \end{bmatrix}, \qquad c = [0\ 1].$$

Example 5.2.2. Consider again another electric circuit, as shown in Figure 5.3:

The state variables here are taken as voltage across the capacitor and the current through the inductor. The state equations are

$$L_1 \frac{di_{L_1}(t)}{dt} = -R_1 i_{L_1}(t) - e_C(t) + e(t),$$

$$L_2 \frac{di_{L_2}(t)}{dt} = -R_2 i_{L_2}(t) + e_C(t),$$

$$C \frac{de_C(t)}{dt} = i_{L_1}(t) - i_{L_2}(t).$$

Setting $x_1 = i_{L_1}, x_2 = i_{L_2}, x_3 = e_C, b = \begin{pmatrix} 1/L_1 \\ 0 \\ 0 \end{pmatrix}, u = e(t)$, the matrix form of the state-space representation of the above system is given by:

$$\dot{x}(t) = \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} -\frac{R_1}{L_1} & 0 & \frac{1}{L_1} \\ 0 & \frac{R_2}{L_2} & \frac{1}{L_2} \\ \frac{1}{C} & \frac{1}{C} & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} + bu(t).$$

First-Order State-Space Representation of Second-Order Systems

Mathematical models of several practical problems, especially those arising in vibrations of structures, are second-order differential equations.

We show by means of a simple example of a spring-mass system how the equations of motion represented by a second-order differential equation can be converted to a first-order state-space representation.

Example 5.2.3. (A Spring-Mass System). Consider the spring-mass system shown in Figure 5.4 with equal spring constants k and masses m_1 and m_2. Let force u_1 be applied to mass m_1 and u_2 be applied to mass m_2.

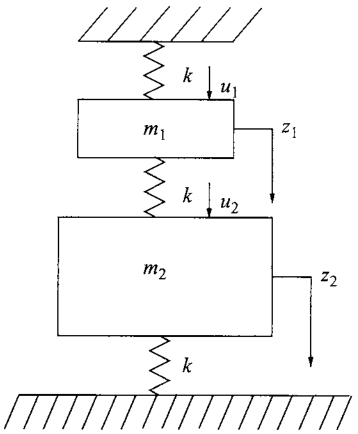


FIGURE 5.4: A spring-mass system.

The equations of motion for the system are:

$$\begin{aligned} m_1\ddot{z}_1 + k(z_1 - z_2) + kz_1 &= u_1, \\ m_2\ddot{z}_2 - k(z_1 - z_2) + kz_2 &= u_2, \end{aligned} \tag{5.2.3}$$

or in matrix form:

$$\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \begin{pmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{pmatrix} + \begin{pmatrix} 2k & -k \\ -k & 2k \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}. \tag{5.2.4}$$

Set

$$z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}.$$

Then, we have

$$M\ddot{z} + Kz = u, \tag{5.2.5}$$

where

$$M = \operatorname{diag}(m_1, m_2), \qquad K = \begin{pmatrix} 2k & -k \\ -k & k \end{pmatrix}, \quad \text{and} \quad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

Let us make a change of variables from z to x as follows:
Set

$$x_1 = z \quad \text{and} \quad x_2 = \dot{z}.$$

Then, in terms of the new variables, the equations of motion become

$$\begin{aligned} \dot{x}_1 &= x_2, \\ M\dot{x}_2 &= -Kx_1 + u, \end{aligned} \tag{5.2.6}$$

or

$$\dot{x} = \begin{pmatrix} 0 & I \\ -M^{-1}K & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ M^{-1} \end{pmatrix} u, \tag{5.2.7}$$

where

$$x = (x_1, x_2)^{\mathrm{T}} = (z, \dot{z})^{\mathrm{T}}.$$

Equation (5.2.7) is a first-order representation of the second-order system (5.2.4).

State-Space Representations of Nonlinear Systems

Mathematical models of many real-life applications are nonlinear systems of differential equations. Very often it is possible to linearize a nonlinear system, and then after linearization, the first-order state-space representation of transformed linear system can be obtained. We will illustrate this by means of the following well-known examples (see Luenberger 1979; Chen 1984; Szidarovszky and Bahill 1991; etc.).

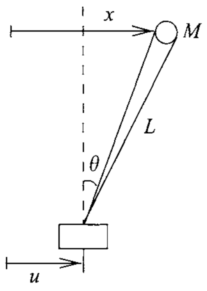


FIGURE 5.5: Balancing of a stick.

Example 5.2.4. (Balancing a Stick). Consider the simple problem of balancing a stick on your hand as shown in the Figure 5.5:

Here L is the length of the stick, and M is the mass of the stick concentrated on the top. The input $u(t)$ is the position of the hand. Then, the position of the top of the stick is

$$x(t) = L\sin\theta(t) + u(t).$$

The torque due to gravity acting on the mass is $MgL \sin\,\theta(t)$. The rotational inertia of the mass on the stick is $ML^2\ddot{\theta}(t)$. The shift of the inertial term down to the pivot point is $\ddot{u}(t)ML \cos\,\theta(t)$. Thus, we have:

$$MgL\sin\,\theta(t) = ML^2\ddot{\theta}(t) + \ddot{u}(t)ML\cos\,\theta(t).$$

The above equations are clearly nonlinear. We now linearize these equations by assuming that θ is small. We then can take $\cos\,\theta = 1$, $\sin\,\theta = \theta$.
This gives us

$$x(t) = L\theta(t) + u(t)$$

and

$$MgL\theta(t) = ML^2\ddot{\theta}(t) + \ddot{u}(t)ML.$$

Eliminating $\theta(t)$ from these two equations, we obtain

$$\ddot{x}(t) = (g/L)\,(x(t) - u(t))\,.$$

We can now write down the first-order state-space representation by setting $v(t) = \dot{x}(t)$. The first-order system is then:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{v}(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{g}{L} & 0 \end{pmatrix} \begin{pmatrix} x(t) \\ v(t) \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{g}{L} \end{pmatrix} u(t).$$

Example 5.2.5. (A Cart with an Inverted Pendulum). Next we consider a similar problem (Figure 5.6), but this time with some more forces exerted (taken from Chen (1984, pp. 96–98)).

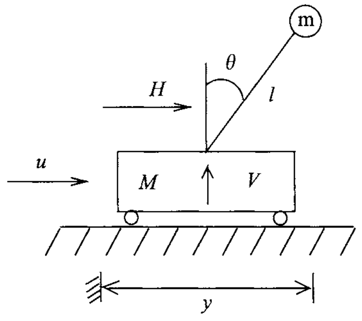


FIGURE 5.6: A cart with an inverted pendulum.

In Figure 5.6, a cart is carrying an inverted pendulum with mass m and length l. Let M be the mass of the cart. Let H and V be, respectively, the horizontal and vertical forces exerted by the cart on the pendulum. Newton's law applied to the linear movements gives:

$$M\ddot{y}(t) = u - H,$$

$$H = m\ddot{y} + ml\cos\theta\ddot{\theta} - ml\sin\theta(\dot{\theta})^2,$$

and

$$mg - V = ml\left(-\sin\theta\ddot{\theta} - \cos\theta\left(\dot{\theta}\right)^2\right).$$

Newton's law applied to the rotational movement of the pendulum gives:

$$ml^2\ddot{\theta} = mgl\sin\theta + Vl\sin\theta - Hl\cos\theta.$$

These are nonlinear equations. We now linearize them by making the same assumptions as before; that is, we assume that θ is small so that we can take $\sin\theta = \theta, \cos\theta = 1$. Dropping the terms involving $\theta^2, \dot{\theta}^2, \theta\dot{\theta}$, and $\theta\ddot{\theta}$, and setting $\sin\theta = \theta$, and $\cos\theta = 1$, we obtain, from above, by eliminating V and H

$$(M + m)\ddot{y} + ml\ddot{\theta} = u$$

and

$$2l\ddot{\theta} - 2g\theta + \ddot{y} = 0.$$

Solving for $\ddot{y}$ and $\ddot{\theta}$, we obtain

$$\ddot{y} = -\frac{2gm}{2M + m}\theta + \frac{2}{2M + m}u,$$

$$\ddot{\theta} = \frac{2g(M + m)\theta}{(2M + m)l} - \frac{1}{(2M + m)l}u.$$

The state-space representations of these linear equations can now be written down by setting $x_1 = y,\ x_2 = \dot{y},\ x_3 = \theta$, and $x_4 = \dot{\theta}$, as follows:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-2gm}{2M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{2g(M+m)}{(2M+m)l} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 \\ \dfrac{2}{2M+m} \\ 0 \\ -\dfrac{1}{(2M+m)l} \end{pmatrix} u,$$

$$y = (1, 0, 0, 0)x.$$

The nonlinear equations in Examples 5.2.4 and 5.2.5 are special cases of the general nonlinear equations of the form:

$$\begin{aligned} \dot{\tilde{x}}(t) &= f(\tilde{x}(t), \tilde{u}(t), t), \qquad \tilde{x}(t_0) = \tilde{x}_0, \\ \tilde{y}(t) &= h(\tilde{x}(t), \tilde{u}(t), t), \end{aligned}$$

where f and h are vector functions. We will now show how these equations can be written in the standard first-order state-space form (Sayed 1994).
Assume that the nonlinear differential equation:

$$\dot{\tilde{x}}(t) = f(\tilde{x}(t), \tilde{u}(t), t), \qquad \tilde{x}(t_0) = x_0$$

has a unique solution and this unique solution is also continuous with respect to the initial condition.

Let $\tilde{x}_{\text{nom}}(t)$ denote the unique solution corresponding to the given input $\tilde{u}_{\text{nom}}(t)$ and the given initial condition $\tilde{x}_{\text{nom}}(t_0)$.

Let the nominal data $\{\tilde{u}_{\text{nom}}(t), \tilde{x}_{\text{nom}}(t)\}$ be perturbed so that

$$\tilde{u}(t) = \tilde{u}_{\text{nom}}(t) + u(t)$$

and $\tilde{x}(t_0) = \tilde{x}_{\text{nom}}(t_0) + x(t_0)$, where $\|u(t)\|$ and $\|x(t_0)\|$ are small; $\|u(t)\| = \sup_t \|u(t)\|_2$.

Assume further that

$$\tilde{x}(t) = \tilde{x}_{\text{nom}}(t) + x(t), \qquad \tilde{y}(t) = \tilde{y}_{\text{nom}}(t) + y(t),$$

where $\|x\|$ and $\|y\|$ are small.

These nonlinear equations can then be linearized (assuming that f and h are smooth enough) by expanding f and h around $(\tilde{u}_{\text{nom}}(t),\ \tilde{x}_{\text{nom}}(t))$, giving rise to a time-invariant linear state-space model of the form:

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

where

$$A = \left.\frac{\partial f}{\partial \tilde{x}}\right|_{(\tilde{x}_{\text{nom}}(t),\tilde{u}_{\text{nom}}(t))}, \qquad B = \left.\frac{\partial f}{\partial \tilde{u}}\right|_{(\tilde{x}_{\text{nom}}(t),\tilde{u}_{\text{nom}}(t))},$$

$$C = \left.\frac{\partial h}{\partial \tilde{x}}\right|_{(\tilde{x}_{\text{nom}}(t),\tilde{u}_{\text{nom}}(t))}, \qquad D = \left.\frac{\partial h}{\partial \tilde{u}}\right|_{(\tilde{x}_{\text{nom}}(t),\tilde{u}_{\text{nom}}(t))}.$$

Example 5.2.6. (The Motion of a Satellite (Sayed 1994)). Suppose that a satellite of unit mass orbits the earth at a distance $d(t)$ from its center (figure 5.7). Let $\theta(t)$ be the angular position of the satellite at time t, and the three forces acting on the satellite are: a radial force $u_1(t)$, a tangential force $u_2(t)$, and an attraction force $\alpha/d^2(t)$, where α is a constant.

The equations of motion are given by

$$\begin{aligned}\ddot{d}(t) &= d(t)\dot{\theta}^2(t) - \frac{\alpha}{d^2(t)} + u_1(t),\\ \ddot{\theta}(t) &= \frac{-2\dot{d}(t)\dot{\theta}(t)}{d(t)} + \frac{u_2(t)}{d(t)}.\end{aligned}$$

Let's define the state variable as

$$\tilde{x}_1(t) = d(t), \qquad \tilde{x}_2(t) = \dot{d}(t), \qquad \tilde{x}_3(t) = \theta(t), \qquad \tilde{x}_4(t) = \dot{\theta}(t)$$

and the output variables as

$$\tilde{y}_1(t) = d(t), \qquad \tilde{y}_2(t) = \theta(t).$$

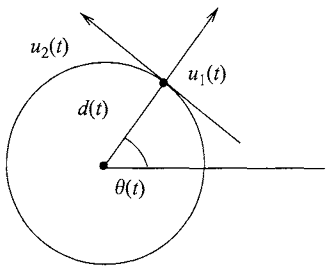


FIGURE 5.7: The motion of a satellite.

The state-space model is then given by:

$$\begin{pmatrix} \dot{\tilde{x}}_1(t) \\ \dot{\tilde{x}}_2(t) \\ \dot{\tilde{x}}_3(t) \\ \dot{\tilde{x}}_4(t) \end{pmatrix} = \begin{pmatrix} \tilde{x}_2(t) \\ \tilde{x}_1(t)\tilde{x}_4^2(t) - \dfrac{\alpha}{\tilde{x}_1^2(t)} + u_1(t) \\ \tilde{x}_4(t) \\ \dfrac{-2\tilde{x}_2(t)\tilde{x}_4(t)}{\tilde{x}_1(t)} + \dfrac{u_2(t)}{\tilde{x}_1(t)} \end{pmatrix},$$

$$\begin{pmatrix} \tilde{y}_1(t) \\ \tilde{y}_2(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1(t) \\ \tilde{x}_2(t) \\ \tilde{x}_3(t) \\ \tilde{x}_4(t) \end{pmatrix}$$

and the initial conditions are:

$$\tilde{x}_0 = \tilde{x}(0) = \begin{pmatrix} \tilde{x}_1(0) \\ \tilde{x}_2(0) \\ \tilde{x}_3(0) \\ \tilde{x}_4(0) \end{pmatrix} = \begin{pmatrix} d(0) \\ 0 \\ \theta(0) \\ \omega_0 \end{pmatrix} = \begin{pmatrix} d_0 \\ 0 \\ \theta_0 \\ \omega_0 \end{pmatrix}.$$

The above is still a nonlinear model of the form:

$$\begin{aligned} \dot{\tilde{x}}(t) &= f(\tilde{x}(t), \tilde{u}(t), t), \qquad \tilde{x}(t_0) = \tilde{x}_0, \\ \tilde{y}(t) &= h(\tilde{x}(t), \tilde{u}(t), t). \end{aligned}$$

Linearizing this nonlinear model around the initial point $(\tilde{x}(0), \tilde{u}(0))$, where $\tilde{u}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, we obtain the linear model:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned}$$

where

$$A = \left.\frac{\partial f}{\partial \tilde{x}}\right|_{(\tilde{x}(0),\ \tilde{u}(0))}$$

$$= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 3\omega_0^2 & 0 & 0 & 2d_0\omega_0 \\ 0 & 0 & 0 & 1 \\ 0 & -2\dfrac{\omega_0}{d_0} & 0 & 0 \end{pmatrix},$$

$$B = \left.\frac{\partial f}{\partial \tilde{u}}\right|_{(\tilde{x}(0),\tilde{u}(0))} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{d_0} \end{pmatrix},$$

and

$$C = \left.\frac{\partial h}{\partial \tilde{x}}\right|_{(\tilde{x}(0),\tilde{u}(0))} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

State-Space Representation of Systems Modeled by Partial Differential Equations

Mathematical models of many engineering problems such as those arising in fluid dynamics, mechanical systems, heat transfer, etc., are partial differential equations. The discretizations of these equations naturally lead to state-space models. We illustrate the idea by means of the following example.

Example 5.2.7. Consider the partial differential equation

$$\frac{\partial^4 y}{\partial x^4} + \frac{P}{\text{EI}}\frac{\partial^2 y}{\partial t^2} = \frac{1}{\text{EI}}F(x,t),$$

which models the **deflection of a prismatic beam** (Soong (1990, pp. 180–181)).
Let $y(x,t)$ be the transverse displacement of a typical segment of the beam that is located at a distance x from the end, and $F(x,t)$ be the applied force. EI is the flexural rigidity, and P is the density of the material of the beam per unit length. Let L be the length of the beam.

Assume that the solution $y(x,t)$ can be written as

$$y(x,t) = \sum_{j=1}^{n} v_j(x)p_j(t)$$

($n = \infty$ in theory, but in practice it is large but finite). Also assume that

$$F(x,t) = \sum_{j=1}^{r} \delta(x - a_j)u_j(t),$$

where $\delta(\cdot)$ is the Dirac delta function.

That is, we assume that the force is point-wise and is exerted at r points of the beam.

Substituting these expressions of $y(x,t)$ and $F(x,t)$ in the partial differential equation, it can be shown that the state equation for the beam in the standard form is:

$$\dot{z}(t) = \mathbf{A}z(t) + Bu(t),$$

where $z(t) = (p_1, \dot{p}_1, p_2, \dot{p}_2, \ldots, p_n, \dot{p}_n)^{\mathrm{T}}$ is the $2n$-dimensional state vector,

$$\mathbf{A} = \operatorname{diag}(\Lambda_1, \Lambda_2, \ldots, \Lambda_n), \qquad B = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix},$$

The matrices B_j and the vector u are defined by:

$$B_j = \frac{1}{\mathrm{EI}} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ v_j(a_1) & v_j(a_2) & \cdots & v_j(a_r) \end{pmatrix}_{n\times r} \quad \text{and} \quad u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{pmatrix}.$$

5.2.2 Discrete-Time Systems

A linear time-invariant discrete-time system may be represented by means of a system of difference equations:

$$x(k+1) = Ax(k) + Bu(k), \tag{5.2.8}$$

$$y(k) = Cx(k) + Du(k). \tag{5.2.9}$$

As before, $x(k)$ is the n-dimensional **state vector**, $u(k)$ is the m-dimensional **input vector**; and A, B, C and D are time-invariant matrices of dimensions $n \times n$, $n \times m$, $r \times n$, and $r \times m$, respectively. The inputs and outputs of a discrete-time system are defined only at discrete time instants.

Sometimes we will write the above equations in the form:

$$x_{k+1} = Ax_k + Bu_k, \tag{5.2.10}$$

$$y_k = Cx_k + Du_k. \tag{5.2.11}$$

5.2.3 Descriptor Systems

The system represented by Eqs. (5.2.1) and (5.2.2) is a special case of a more general system, known as the **descriptor system**.

A continuous-time linear descriptor system has the form:

$$E\dot{x}(t) = Ax(t) + Bu(t), \tag{5.2.12}$$

$$y(t) = Cx(t) + Du(t). \tag{5.2.13}$$

Similarly, a discrete-time linear descriptor system has the form:

$$Ex(k+1) = Ax(k) + Bu(k), \tag{5.2.14}$$

$$y(k) = Cx(k) + Du(k). \tag{5.2.15}$$

If the matrix E is nonsingular, then, of course, the descriptor system represented by (5.2.12) and (5.2.13) is reduced to the standard form (5.2.1) – (5.2.2). Similarly, for the system (5.2.14) – (5.2.15). However, the case when E is singular or nearly singular is more interesting. A book devoted to singular systems of differential equations is by Campbell (1980). We will now give an example to show how a singular descriptor systems arises in practice.

Example 5.2.8. (Simplified Samuelson's Model of Economics). Let NI(k), CS(k), IV(k), and GE(k), denote, respectively, the national income, consumption, investment, and government expenditure of a country at a given year k.

The economist P.A. Samuelson proposed a model of the national economy of a country, which is based on the following assumptions:

1. National income NI(k) is equal to the sum of the consumption CS(k), investment IV(k), and the government expenditure GE(k) at a given year k.
2. Consumption CS($k+1$) at the year $k+1$ is proportional to the national income NI(k) at the previous year k.
3. Investment IV($k+1$) at the year $k+1$ is proportional to the difference of the consumer spending CS($k+1$) at that year and that of the previous year CS(k).

The state-space representation of Samuelson's model, then, can be written in the form:

$$\begin{aligned} \mathrm{NI}(k) &= \mathrm{CS}(k) + \mathrm{IV}(k) + \mathrm{GE}(k), \\ \mathrm{CS}(k+1) &= \alpha \mathrm{NI}(k), \\ \mathrm{IV}(k+1) &= \beta\,[\mathrm{CS}(k+1) - \mathrm{CS}(k)]. \end{aligned} \tag{5.2.16}$$

These equations in matrix form are:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & -\beta & 0 \end{pmatrix} \begin{pmatrix} \mathrm{IV}(k+1) \\ \mathrm{CS}(k+1) \\ \mathrm{NI}(k+1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 0 & \alpha \\ 0 & -\beta & 0 \end{pmatrix} \begin{pmatrix} \mathrm{IV}(k) \\ \mathrm{CS}(k) \\ \mathrm{NI}(k) \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \mathrm{GE}(k).$$

or

$$Ex(k+1) = Ax(k) + Bu(k),$$

where $x(k) = \begin{pmatrix} IV(k) \\ CS(k) \\ NI(k) \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad u(k) = \mathrm{GE}(k).$

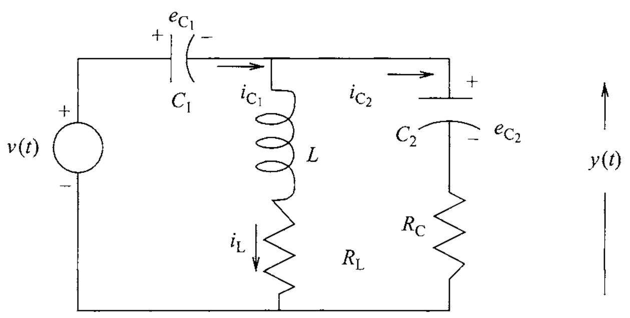


FIGURE 5.8: An electric circuit for a descriptor system.

(Note that E is singular).
For details, see Luenberger (1979, pp. 122–123).

Example 5.2.9. Consider another electric circuit given in Figure 5.8 driven by a voltage source $v(t)$.

The state variables are taken as $x_1 := e_{C_1}, x_2 := i_L$, and $x_3 := e_{C_2}$. By applying Kirchhoff's current and voltage laws we have:

$$i_{C_1} = i_L + i_{C_2}, \qquad i_{C_1} = C_1 \frac{de_{C_1}}{dt}, \qquad i_{C_2} = C_2 \frac{de_{C_2}}{dt},$$

$$v(t) = e_{C_1} + L\frac{di_L}{dt} + R_L i_L = e_{C_1} + e_{C_2} + R_C i_{C_2}.$$

Manipulating these equations we have the state equation:

$$E\dot{x} = Ax + bu,$$

where $u := v$, $E =$ diag $(R_C C_1, L, R_C C_2)$

$$x = [x_1, x_2, x_3]^T$$

$$A = \begin{bmatrix} -1 & R_C & -1 \\ -1 & -R_L & 0 \\ -1 & 0 & -1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

The output is defined by $y(t) = e_{C_2} + R_C i_{C_2}$, so the output equation becomes

$$y = cx + du,$$

where $c = (-1\ 0\ 0)$, $d = 1$.

5.3 SOLUTIONS OF A CONTINUOUS-TIME SYSTEM: SYSTEM RESPONSES

Theorem 5.3.1. *Continuous-Time State-Space Solution. The solutions of the continuous-time dynamical equations*

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(t_0) = x_0. \tag{5.3.1}$$

$$y(t) = Cx(t) + Du(t) \tag{5.3.2}$$

are given by

$$x(t) = e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-s)}Bu(s)\,ds, \tag{5.3.3}$$

$$y(t) = Ce^{A(t-t_0)}x_0 + \int_{t_0}^{t} Ce^{A(t-s)}Bu(s)\,ds + Du(t). \tag{5.3.4}$$

Remark

- If $u(t) = 0$, then $x(t) = e^{A(t-t_1)}x(t_1)$ for every $t \geq t_0$ and any $t_1 \geq t_0$.

Definition 5.3.1. *The matrix $e^{A(t-t_1)}$ is called the* ***state transition matrix.***

Since the state at any time can be obtained from that of any other time through the state transition matrix, **it will be assumed, without any loss of generality, that $t_0 = 0$, unless otherwise mentioned.**

Assuming $t_0 = 0$, the Eqs. (5.3.3) and (5.3.4) will, respectively, be reduced to

$$x(t) = e^{At}x_0 + \int_{0}^{t} e^{A(t-s)}Bu(s)\,ds \tag{5.3.5}$$

and

$$y(t) = Ce^{At}x_0 + \int_{0}^{t} Ce^{A(t-s)}Bu(s)\,ds + Du(t). \tag{5.3.6}$$

Definition 5.3.2. *The matrix e^{At} defined above has the form:*

$$e^{At} = \sum_{k=0}^{\infty} \frac{(At)^k}{k!}$$

and is called the ***matrix exponential.***

Proof. Proof of (5.3.5) and (5.3.6): Noting that $(d/dt)(e^{At}) = Ae^{At}$ (see **Section 5.3.1**), we first verify that the expression (5.3.5) satisfies (5.3.1) with

$t = 0$. Differentiating (5.3.5), we have

$$\begin{aligned}\dot{x}(t) &= Ae^{At}x_0 + Bu(t) + \int_0^t \frac{d}{dt}e^{A(t-s)}Bu(s)\,ds,\\ &= Ae^{At}x_0 + Bu(t) + A\int_0^t e^{A(t-s)}Bu(s)\,ds,\\ &= A\left[e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)\,ds\right] + Bu(t),\\ &= Ax(t) + Bu(t).\end{aligned}$$

Also, note that at $t = 0$,

$$x(0) = x_0.$$

Thus, the solution $x(t)$ also satisfies the initial condition.

The expression for $y(t)$ in (5.3.6) follows immediately by substituting the expression for $x(t)$ from (5.3.5) into $y(t) = Cx(t) + Dx(t)$. ■

Free, Forced, and Steady-State Responses

Given the initial condition x_0 and the control input $u(t)$, the vectors $x(t)$ and $y(t)$ determine the **system time responses** for $t \geq 0$. The system time responses determine the behavior of the dynamical system for particular classes of inputs. System characteristics such as **overshoot, rise-time, settling time,** etc., can be determined from the time responses.

In the expression (5.3.6), the first term $Ce^{At}x_0$ represents the response of the system due to the initial condition x_0 with zero input **(zero-input response)**. This is called the **free response** of the system.

On the other hand, the second term in (5.3.6) determines, what is known, as the **forced response** of the system. It is due to the forcing function $u(t)$ applied to the system with zero initial conditions. A special case of the forced response is known as the **impulse response** which is obtained from (5.3.6) by setting $x_0 = 0$ and $u(t) = \delta(t)$, where $\delta(t)$ is the unit impulse or **Dirac delta function**. Then,

$$\begin{aligned}y(t) &= \int_0^t (Ce^{A(t-s)}B + D\delta(t-s))u(s)\,ds,\\ &= \int_0^t H(t-s)u(s)\,ds, \qquad (5.3.7)\end{aligned}$$

where the matrix $H(t)$, **the impulse response matrix**, is defined by

$$H(t) := Ce^{At}B + D\delta(t). \qquad (5.3.8)$$

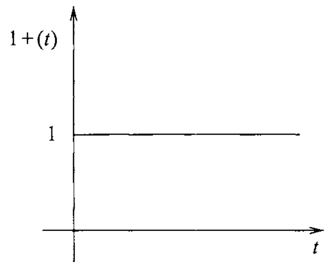


FIGURE 5.9: A unit step function.

In fact, if $x_0 = 0$, then the (i, j)th element of the impulse response matrix $H(t)$ of the system (5.3.1) – (5.3.2) is the response at time t at the output i due to a unit impulse applied at time $t = 0$ at the input j of the system, while all other inputs are kept at zero. Similarly, the **unit step response** is defined to be the output using the input as the unit step function in the manner done for an impulse response, assuming again that the initial state is zero; that is, $x_0 = 0$.

A unit step function $1_+(t)$ (Figure 5.9) is given by

$$1_+(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

For any finite value of time t, the response $y(t)$, that is, the right-hand side of (5.3.6), will contain terms consisting of $e^{\alpha_i t}e^{j\omega_i t}$ if $\lambda_i = \alpha_i + j\omega_i$, $j = \sqrt{-1}$, is a **simple eigenvalue** of A, and the other terms are governed by the nature of the input function $u(t)$.

When t is finite, the part of the response in $y(t)$ which is governed by $e^{\alpha_i t}e^{j\omega_i t}$ is called the **transient response** of the system. As t tends to infinity, this transient part of the response tends to zero if all α_is are negative or it grows to become unbounded if at least one of α_is is positive. Thus, $y_{ss}(t) \equiv \lim_{t\to\infty} y(t)$ will be called the **steady-state response** of the system. The speed with which the response $y(t)$ will reach the steady-state value $y_{ss}(t)$ will be determined by the largest value of αs.

MATLAB note: MATLAB functions **step, impulse**, and **initial** in MATLAB CONTROL TOOLBOX can be used to generate plots for step, impulse, and initial condition responses, respectively. Their syntax are:

step (sys),
impulse (sys),
initial (sys, x_0).

Example 5.3.1. (Baldwin (1961, pp. 29–44)). The dynamic behavior of a moving coil galvanometer, see Figure 5.10,

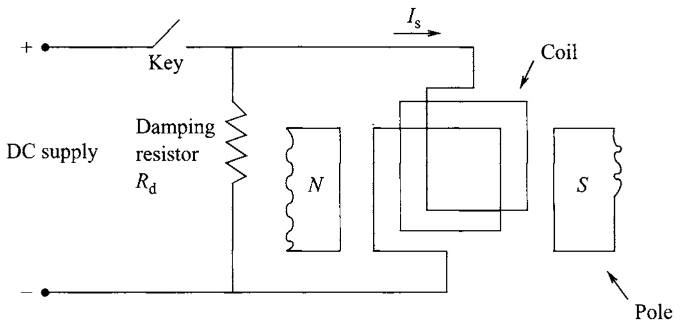


FIGURE 5.10: Basic Circuit of a Moving Coil Galvanometer.

is governed by

$$J\frac{d^2\theta}{dt^2} + D\frac{d\theta}{dt} + C\theta = GI_s, \tag{5.3.9}$$

where

J = the moment of inertia about the axis of rotation of moving parts of the instrument,

D = the damping constant,

C = stiffness of suspension,

G = galvanometer constant which represents the electromagnetic torque developed on the coil by 1 A of current flowing through it,

θ = the deflection of the coil about its axis of rotation,

I_s = the steady-state current flowing through the galvanometer coil, and

R_g = galvanometer resistance.

It can be shown that D is given by

$$D = \frac{G^2}{R_g + R_d} + D_{\text{air}},$$

where

R_g = resistance of galvanometer coil,

R_d = damping resistor,

D_{air} = damping to the coil due to air.

If the key is opened interrupting supply current I_s to the galvanometer, the response of the coil is determined by (5.3.9) with $I_s = 0$ and is shown in Figure 5.11 where θ_0 is the steady-state deflection with I_s flowing through the coil.

A galvanometer with a very low damping constant is known as a ballistic galvanometer. If a charged capacitor is discharged through a ballistic galvanometer such that

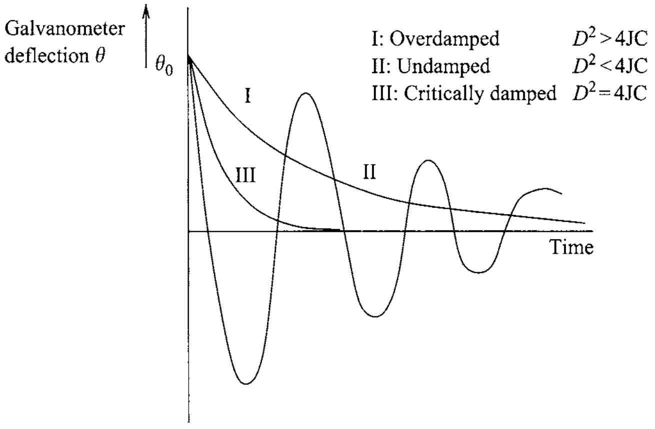


FIGURE 5.11: Step-response of the galvanometer.

the whole charge should have passed before the coil has time to move appreciably, we have the **torque impulse** due to the whole charge equal to $\int Gi\,dt = GQ$, the integral being taken over the time of passage of the charge Q and i is the instantaneous current flowing through galvanometer coil. The subsequent time response of the galvanometer will be similar to that shown in Figure 5.11 but will differ in the fact that the response now starts from the origin. The responses in three cases: damped, undamped, and critically damped, are shown in Figure 5.11.

Causality: If the output of the system at time t does not depend upon the input applied to the system after time t, but depends only upon the present and the past inputs, the system is said to be **causal.**

In this book, all systems will be assumed to be causal.

5.3.1 Some Important Properties of the Matrix e^{At}

Since the matrix exponential e^{At} plays a fundamental role in the solution of the state equations, we will now discuss the various methods for computing this matrix. Before doing that, we list some important properties of this matrix. These properties are easily verifiable and left as Exercises (5.8–5.10) for the readers.

1. $e^{A(t+s)} = e^{At} \cdot e^{As}$
2. $d/dt(e^{At}) = Ae^{At} = e^{At}A$
3. $e^{(A+B)t} = e^{At} \cdot e^{Bt}$, if and only if A and B commute; that is, if and only if $AB = BA$
4. e^{At} is nonsingular and $(e^{At})^{-1} = e^{-At}$
5. $\left(e^{A/m}\right)^m = e^A$, where m is an integer
6. $e^{P^{-1}APt} = P^{-1}e^{At}P$.

5.3.2 Sensitivity of e^{At}

We know that the accuracy of a solution obtained by an algorithm is greatly influenced by the sensitivity of the problem. We will, therefore, consider the sensitivity aspect of the problem of computing e^{At} first. We just state a result due to Van Loan (1977) without proof, which will help identify the condition number of the problem.

Let E be a perturbation matrix. We are interested in knowing how large the **relative error**

$$\rho = \frac{\|e^{(A+E)t} - e^{At}\|_2}{\|e^{At}\|_2}$$

can be.

Differentiating $e^{(A+E)s}e^{A(t-s)}$ with respect to s, we obtain

$$e^{(A+E)t} - e^{At} = \int_0^t e^{A(t-s)}Ee^{(A+E)s}ds.$$

It then follows that

$$\rho \leq \frac{\|E\|_2}{\|e^{At}\|_2}\int_0^t \|e^{A(t-s)}\|_2\|e^{(A+E)s}\|_2 ds$$

Further simplification of this result is possible.

Van Loan (1977) has shown that, for a given t, there exists a perturbation matrix E such that

$$\rho = \frac{\|e^{(A+E)t} - e^{At}\|_2}{\|e^{At}\|_2} \approx \kappa(A,t)\frac{\|E\|_2}{\|A\|_2},$$

where

$$\kappa(A,t) = \max_{\|E\|_2\leq 1}\left\|\int_0^t e^{A(t-s)}Ee^{As}ds\right\|_2 \frac{\|A\|_2}{\|e^{At}\|_2}.$$

This result shows that $\kappa(A,t)$ **is the condition number for the problem** e^{At}**. If this number is large, then a small change in** A **can cause a large change in** e^{At}**, for a given** t**.**

Though determining $\kappa(A,t)$ involves computation of a matrix integral, it can be easily verified that

$$\kappa(A,t) \geq t\|A\|_2,$$

with equality holding for all nonnegative t if and only if A is a normal matrix, that is, if $A^{\mathrm{T}}A = AA^{\mathrm{T}}$. **"When** A **is not normal,** $\kappa(A,t)$ **can grow like a high degree polynomial in** t**"** (Moler and Van Loan 1978).

Example 5.3.2. Consider computing e^A, where

$$A = \begin{pmatrix} -1 & 1000 \\ 0 & -1 \end{pmatrix}.$$

Since $\|A\|_2 = 10^3$, the problem of computing the matrix exponential e^A is expected to be ill-conditioned.

Let's verify this as follows. The matrix e^A computed by using the MATLAB function **expm** is

$$e^A = \begin{pmatrix} 0.3679 & 367.8794 \\ 0 & 0.3679 \end{pmatrix}.$$

Now change the (2, 1) entry of A to 10^{-8} and call this perturbed matrix A_{new}. We now obtain

$$e^{A_{new}} = \begin{pmatrix} 0.3679 & 367.8801 \\ 0 & 0.3679 \end{pmatrix}.$$

The relative error in the solution is

$$\frac{\|e^A - e^{A_{\text{new}}}\|_2}{\|e^A\|_2} = O(10^{-6}).$$

On the other hand, the relative error in the data is

$$\frac{\|A - A_{\text{new}}\|_2}{\|A\|_2} = O(10^{-11}).$$

(The matrix $e^{A_{\text{new}}}$ was also computed by the MATLAB function **expm**).

5.3.3 Computational Methods for e^{At}

There is a wide variety of methods to compute the matrix exponential. Several of these methods have been carefully analyzed, with respect to efficiency, and numerical stability, in an authoritative paper on the subject by Moler and Van Loan (1978). We discuss the following ones briefly.

- **The eigenvalue–eigenvector method**
- **Series methods**
- **Ordinary differential equations (ODE) Methods**
- **Matrix decomposition methods.**

The Eigenvalue–Eigenvector Method

We have seen that the solution of the unforced system:

$$\dot{x}(t) = Ax(t), \qquad x(0) = x_0 \tag{5.3.10}$$

is given by

$$x(t) = e^{At}x_0. \tag{5.3.11}$$

Equation (5.3.11) shows that the ith column of the matrix e^{At} is just the vector $x(t)$ with $x(0) = e_i$, the ith unit vector.

Again, $x(t)$ can be expressed in terms of the eigenvalues and eigenvectors of A:

$$x(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2 + \cdots + c_n e^{\lambda_n t} v_n, \tag{5.3.12}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the **simple eigenvalues of** A and v_1 through v_n are a set of linearly independent eigenvectors associated with λ_1 through λ_n. The scalar cs are computed from the given initial condition.

Thus, when the eigenvalues of A are simple, the matrix e^{At} is completely determined by the eigenvalues and eigenvectors of the matrix A. The same can be shown to be true when A has multiple eigenvalues.

A difficulty with this approach arises when A has some nearly equal eigenvalues. This can be seen from the following theorem (Moler and Van Loan 1978).

Theorem 5.3.2. *Let A be an $n \times n$ nondefective matrix; that is, it has a set of n linearly independent eigenvectors. Let $X^{-1}AX = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A. Then,*

$$\|fl(e^{At}) - e^{At}\|_2 \leq n\mu e^{\rho(A)t}\operatorname{Cond}_2(X),$$

where $\rho(A) = \max |\lambda_i|$ is the spectral radius of A.

Interpretation of Theorem 5.3.2. Theorem 5.3.2 shows that **there might be a large error in computing e^{At} whenever there is a coalescence of eigenvalues of A,** because, in this case, $\operatorname{Cond}_2(X)$ will be large.

The following simple 2×2 example taken from Moler and Van Loan (1978) illustrates the difficulty.

Let

$$A = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}.$$

Then,

$$e^{At} = \begin{pmatrix} e^{\lambda t} & \alpha \dfrac{e^{\lambda t} - e^{\mu t}}{\lambda - \mu} \\ 0 & e^{\mu t} \end{pmatrix}.$$

Clearly, the result will be inaccurate if λ is close to μ, but not exactly equal to μ. A large round-off error can be expected in this case.

Series Method for Computing the Matrix Exponential

In this section, we briefly state two series methods: **The Taylor Series Method** and the **Padé Approximation Method.** When properly implemented, these methods become numerically effective for computing the matrix exponential.

The Taylor Series Method

An obvious way to approximate e^A is to evaluate a finite-series sum by truncating the Taylor series:

$$e^A = I + A + \frac{A^2}{2} + \frac{A^3}{6} + \cdots$$

after k terms. Thus, if

$$T_k(A) = \sum_{j=0}^{k} \frac{A^j}{j!}$$

and if $\mathrm{fl}(T_k(A))$ denotes the floating point evaluation of $T_k(A)$, then it is natural to choose k so that $\mathrm{fl}(T_k(A)) = \mathrm{fl}(T_{k+1}(A))$. **The drawbacks to this method are that a large number of terms is needed for convergence, and even when convergence occurs, the answer can be totally wrong.**

Consider the following example from Moler and Van Loan (1978).

Example 5.3.3. Consider computing e^A using the Taylor series methods with the following 2×2 matrix A and a relative accuracy of about 10^{-5}.

$$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix}.$$

A total of $k = 59$ terms were required for convergence and the computed output was

$$e^A \approx \begin{pmatrix} -22.25880 & -1.432766 \\ -61.49931 & -3.474280 \end{pmatrix},$$

which is nowhere close to the true answer (to 6 decimal places)

$$e^A \approx \begin{pmatrix} -0.735759 & 0.551819 \\ -1.471518 & 1.103638 \end{pmatrix}.$$

The source of error here is the catastrophic cancellation that took place in the evaluation of $(A^{16}/16!) + (A^{17}/17!)$, using finite-precision arithmetic (see **Chapter 3**). These two terms have almost the same magnitude but are of opposite signs, as seen from the following expressions of $(A^{16}/16!)$ and $(A^{17}/17!)$:

$$\frac{A^{16}}{16!} = 10^6 \begin{pmatrix} 6.9773 & -3.4886 \\ 9.3030 & -4.6515 \end{pmatrix},$$

$$\frac{A^{17}}{17!} = 10^6 \begin{pmatrix} -6.9772 & 3.4885 \\ -9.3030 & 4.6515 \end{pmatrix}.$$

For relative accuracy of 10^{-5}, "the elements of these intermediate results have absolute errors larger than the final result".

The Padé Approximation Method
Suppose that $f(x)$ is a power series represented by

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots .$$

Then the (p, q) Padé approximation to $f(x)$ is defined as

$$f(x) \approx \frac{c(x)}{d(x)} = \frac{\sum_{k=0}^{p} c_k x^k}{\sum_{k=0}^{q} d_k x^k}.$$

The coefficients $c_k, k = 0, \ldots, p$, and $d_k, k = 0, \ldots, q$ are chosen such that the terms containing $x^0, x^1, x^2, \ldots, x^{p+q}$ are cancelled out in $f(x)d(x) - c(x)$. The order of the Padé approximation is $p + q$. The ratio $(c(x)/d(x))$ is unique and the coefficients $c_0, c_1, \ldots, c_p$ and $d_0, d_1, \ldots, d_q$ always exist. The (p, q) Padé approximation to e^A is given by

$$R_{pq}(A) = \left[D_{pq}(A)\right]^{-1} N_{pq}(A),$$

where

$$D_{pq}(A) = \sum_{j=0}^{q} \frac{(p+q-j)!q!}{(p+q)!j!(q-j)!}(-A)^j$$

and

$$N_{pq}(A) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!}A^j.$$

It can be shown (**Exercise 5.16**) that if p and q are large, or if A is a matrix having all its eigenvalues **with negative real parts**, then $D_{pq}(A)$ is nonsingular.

Round-off errors due to catastrophic cancellation is again a major concern for this method. It is less when $\|A\|$ is not too large and the diagonal approximants $(p = q)$ are used.

Scaling and Squaring in Taylor Series and Padé Approximations

The difficulties with the round-off errors in Taylor series and Padé approximation methods can somehow be controlled by a technique known as **Scaling and Squaring**. Since $e^A = (e^{A/m})^m$, the idea here is to choose m to be a power of 2 so that $e^{A/m}$ can be computed rather efficiently and accurately using the method under consideration, and then to form the matrix $(e^{A/m})^m$ by repeated squaring.

Moler and Van Loan (1978) have remarked **"When properly implemented, the resulting algorithm is one of the most effective we know."** The method may fail but it is **reliable** in the sense that it tells the user when it does.

The method has favorable numerical properties when $p = q$. We will, therefore, describe the algorithm for this case only.

Let $m = 2^j$ be chosen such that $\|A\|_\infty \leq 2^{j-1}$, then Moler and Van Loan (1978) have shown that there exists an E such that

$$\left[R_{pp}(A/2^j)\right]^{2^j} = e^{A+E},$$

where $\|E\|_\infty \leq \epsilon \|A\|_\infty$, with

$$\epsilon = 2^{3-2p}\left(\frac{(p!)^2}{(2p)!(2p+1)!}\right).$$

Given an error-tolerance δ, the above expression, therefore, gives a criterion to choose p such that $\|E\|_\infty \leq \delta \|A\|_\infty$.

The above discussion leads to the following algorithm.

Algorithm 5.3.1. *Padé Approximation to e^A using Scaling and Squaring.*

```
Input. $A \in \mathbb{R}^{n\times n}$, $\delta > 0$, an error-tolerance.
Output. $F = e^{A+E}$ with $\|E\|_\infty \leq \delta\|A\|_\infty$.
Step 1. Choose $j$ such that $\|A\|_\infty \leq 2^{j-1}$. Set $A \equiv A/2^j$.
Step 2. Find $p$ such that $p$ is the smallest nonnegative integer satisfying
        $\left(\frac{8}{2^{2p}}\right)\frac{(p!)^2}{(2p)!(2p+1)!} \leq \delta.$
Step 3. Set $D \equiv I$, $N \equiv I$, $Y \equiv I$, $c = 1$.
Step 4. For $k = 1, 2, \ldots, p$ do
            $c \equiv c(p-k+1)/[(2p-k+1)k]$
            $Y \equiv AY$, $N \equiv N + cY$, $D = D + (-1)^k cY$.
End
Step 5. Solve for $F$: $DF = N$.
```

Step 6. *For* $k = 1, 2, \ldots, j$ *do*

$$F \equiv F^2.$$

End

Flop-count: · The algorithm requires about $2(p + j + (1/3)n^3$ flops.

Numerical Stability Property: **The algorithm is numerically stable when** A **is normal. When** A **is non-normal,** an analysis of the stability property becomes difficult, because, in this case e^A **may grow before it decays during the squaring process;** which is known as **"hump"** phenomenon. For details, see Golub and Van Loan (1996, p. 576).

MATLAB note: The MATLAB function **expm** computes the exponential of a matrix A, using Algorithm 5.3.1.

MATCONTROL note: Algorithm 5.3.1 has been implemented in MATCONTROL function **expmpade**.

Example 5.3.4. Consider computing e^A using Algorithm *5.3.1* with

$$A = \begin{pmatrix} 5 & 1 & 0 \\ 0 & 2 & 0 \\ 2 & 3 & 1 \end{pmatrix}.$$

Set $\delta = 0.50$.

Step 1. Since $\|A\|_\infty = 7$, choose $j = 4$.

Then,

$$A \equiv \frac{A}{2^4} = \begin{pmatrix} 0.3125 & 0.8625 & 0 \\ 0 & 0.1750 & 0 \\ 0.1250 & 0.1875 & 0.6625 \end{pmatrix}.$$

Step 2. $p = 1$.
Step 3. $D = N = Y = I$.
Step 4. $k = 1, c = 0.5$.

$$Y = \begin{pmatrix} 0.3125 & 0.0625 & 0 \\ 0 & 0.1250 & 0 \\ 0.1250 & 0.7875 & 0.0625 \end{pmatrix},$$

$$N = \begin{pmatrix} 1.1563 & 0.0313 & 0 \\ 0 & 1.0625 & 0 \\ 0.0625 & 0.09838 & 1.0313 \end{pmatrix}, \quad D = \begin{pmatrix} 0.8438 & -0.0313 & 0 \\ 0 & 0.9375 & 0 \\ -0.0625 & -0.0938 & 0.9638 \end{pmatrix},$$

$$F = \begin{pmatrix} 1.3704 & 0.0790 & 0 \\ 0 & 1.1333 & 0 \\ 0 & 0.2115 & 1.0645 \end{pmatrix}.$$

Step 5. $$F \equiv F^2 = \begin{pmatrix} 154.6705 & 49.0874 & 0 \\ 0 & 7.4084 & 0 \\ 75.9756 & 36.2667 & 2.7192 \end{pmatrix}.$$

The matrix e^A given by F in Step 5 is different from what would have been obtained by using MATLAB function $expm(A)$, which is:

$$e^A = \begin{pmatrix} 148.4132 & 47.0080 & 0 \\ 0 & 7.3891 & 0 \\ 72.8474 & 35.1810 & 2.7183 \end{pmatrix}.$$

This is because MATLAB uses much smaller tolerance than what was prescribed for this example.

ODE Methods

Consider solving

$$\dot{x}(t) = f(x,t), \qquad x(0) = x_0$$

with

$$f(x,t) = Ax(t).$$

Then the kth column of e^{At} becomes equal to the solution $x(t)$ by setting $x(0)$ as the kth column of the identity matrix. Thus, any ODE solver can be used to compute e^{At}.

However, computing e^{At} using a general-purpose ODE solver will be rather expensive, because such a method does not take advantage of the special form of $f(x,t) = Ax(t)$. **An ODE routine will treat $Ax(t)$ as any function $f(x,t)$ and the computation will be carried on.**

However, a single-step ODE method such as the fourth order **Taylor** or **Runge–Kutta** method and a multistep solver such as the **Adams formulas** with variable step size, could be rather computationally practically feasible (and also reliable and stable) for the matrix vector problem of computing $e^{At}x(0)$, when such a problem is to be solved for many different values of t and also when A is large and sparse.

Matrix Decomposition Methods

The basic idea behind such a method is to first transform the matrix A to a suitable canonical form R so that e^R can be easily computed, and then compute e^A from e^R. Thus, if P is the transforming matrix such that

$$P^{-1}AP = R,$$

then $e^A = Pe^RP^{-1}$.

The convenient choices for R include the **Jordan canonical** form (JCF), **the companion form, the Hessenberg form, and the real Schur form (RSF) of a matrix.** Because of the difficulties in using the JCF and the companion form, stated in Section 4.1, we will not discuss computing e^A via these forms here.

Though the Hessenberg form can be obtained in a numerically stable way, computation of e^H via an upper Hessenberg matrix H will involve divisions by the superdiagonal entries, and if the product of these entries is small, large round-off errors can contaminate the computation (**see Exercise 5.12**). Thus, **our choice for *R* here is the RSF.**

Computing e^A via the Real Schur Form
Let A be transformed to a real Schur matrix R using an orthogonal similarity transformation:

$$P^{\mathrm{T}}AP = R,$$

then $e^A = Pe^RP^{\mathrm{T}}$.

The problem is now how to compute e^R. Parlett (1976) has given an elegant formula to compute $f(R)$ for an analytic function $f(x)$, where R is **upper triangular**. The formula is derived from the commutativity property: $Rf(R) = f(R)R$.

Adapting this formula for computing $F = e^R$, we have the following algorithm when R is an upper triangular matrix. The algorithm needs to be modified when R is a **quasi-triangular matrix (Exercise 5.21).**

Algorithm 5.3.2. *The Schur Algorithm for e^A.*
Input. $A \in \mathbb{R}^{n\times n}$
Output. e^A.
Step 1. *Transform A to R, an* ***upper triangular*** *matrix, using the QR iteration algorithm (Chapter 4):*

$$P^{\mathrm{T}}AP = R.$$

(Note that when the eigenvalues of A are all real, the RSF is upper triangular.)
Step 2. *Compute* $e^R = G = (g_{ij})$:

```
For $i = 1, \dots, n$ do
$g_{ii} = e^{r_{ii}}$
End
For $k = 1, 2, \dots, n-1$ do
    For $i = 1, 2, \dots, n-k$ do
        Set $j = i + k$
        $$g_{ij} = \frac{1}{(r_{ii} - r_{jj})}\left[r_{ij}(g_{ii} - g_{jj}) + \sum_{p=i+1}^{j-1}(g_{ip}r_{pj} - r_{ip}g_{pj})\right].$$
    End
End
```

Step 3. *Compute* $e^A = Pe^RP^{\mathrm{T}}$.

Flop-count: Computation of e^R in Step 2 requires about $(2n^3/3)$ flops.

MATCONTROL note: Algorithm 5.3.2 has been implemented in MATCONTROL function **expmschr**.

Example 5.3.5. Consider computing e^A using Algorithm *5.3.2* with the matrix A of Example 5.3.3.

$$A = \begin{pmatrix} 5 & 1 & 0 \\ 0 & 2 & 0 \\ 2 & 3 & 1 \end{pmatrix}.$$

Using MATLAB function $[P, R] = \mathbf{schur}(A)$, we obtain

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 1 \\ 0 & 0 & 2 \end{pmatrix}.$$

$$g_{11} = e^{r_{11}} = 2.7183, \qquad g_{22} = e^{r_{22}} = 148.4132, \qquad g_{33} = e^{r_{33}} = 7.3891,$$

$$\mathbf{k = 1}, i = 1, j = 2: \qquad g_{12} = \frac{1}{(r_{11} - r_{22})}[r_{12}(g_{11} - g_{22})] = 72.8474,$$

$$\mathbf{k = 1}, i = 2, j = 3: \qquad g_{23} = \frac{1}{(r_{22} - r_{33})}[r_{23}(g_{22} - g_{33})] = 47.0090,$$

$$\mathbf{k = 2}, i = 1, j = 3: \qquad g_{13} = \frac{1}{(r_{11} - r_{33})}[r_{13}(g_{11} - g_{33}) + (g_{12}r_{23} - r_{12}g_{23})]$$
$$= 35.1810.$$

So,

$$e^R = \begin{pmatrix} 2.7183 & 72.8474 & 35.1810 \\ 0 & 148.4132 & 47.0080 \\ 0 & 0 & 7.3891 \end{pmatrix}.$$

Thus,

$$e^A = Pe^R P^{\mathrm{T}} = \begin{pmatrix} 148.4132 & 47.0080 & 0 \\ 0 & 7.3891 & 0 \\ 72.8474 & 35.1810 & 2.7183 \end{pmatrix}.$$

Note that e^A obtained here is exactly the same (in four-digit arithmetic) as given by MATLAB function **expm** (A), which is based on Algorithm 5.3.1.

Numerical stability of the schur algorithm: Numerical difficulties clearly arise when A has equal or nearly equal confluent eigenvalues, even though the transformation matrix P is orthogonal.

5.3.4 Comparison of Different Methods for Computing the Exponential Matrix

From our discussions in previous sections, it is clear that the **Padé approximation method** (with scaling and squaring) and the **Schur method** should, in general, be attractive from computational viewpoints.

The Taylor series method and the methods based on reduction of A to a companion matrix or to the JCF should be avoided for the reason of numerical instability. The ODE techniques should be preferred when the matrix A is large and sparse.

5.3.5 Evaluating an Integral with the Matrix Exponential

We have discussed methods for computing the matrix exponential. We now present a method due to Van Loan (1978) to compute integrals involving exponential matrices.

The method can be used, in particular, to compute the state-space solution (5.3.3) of the Eq. (5.3.1), and the **controllability** and **observability Grammians**, which will be discussed in the next chapter.

The method uses diagonal Padé approximation discussed in the last section.

Let

$$H(\Delta) = \int_0^{\Delta} e^{As} B\, ds, \qquad M(\Delta) = \int_0^{\Delta} e^{A^T s} Q H(s)\, ds,$$

$$N(\Delta) = \int_0^{\Delta} e^{A^T s} Q e^{As} ds, \qquad W(\Delta) = \int_0^{\Delta} H(s)^T Q H(s)\, ds$$

where A and B are matrices of order n and $n \times m$, respectively, and Q is a symmetric positive semidefinite matrix.

Algorithm 5.3.3. *An Algorithm for Computing Integrals involving Matrix Exponential.*

Inputs.

A—The $n \times n$ state matrix

B—An $n \times m$ matrix

Q—A symmetric positive semidefinite matrix.

Outputs. *F, H, Q, M, and W which are, respectively, the approximations to $e^{A\Delta}$, $H(\Delta)$, $N(\Delta)$, $M(\Delta)$, and $W(\Delta)$.*

Step 1. *Form the* $(3n+m)\times(3n+m)$ *matrix*

$$\hat{C} = \begin{pmatrix} -A^{\mathrm{T}} & I & 0 & 0 \\ 0 & -A^{\mathrm{T}} & Q & 0 \\ 0 & 0 & A & B \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Find the smallest positive integer j *such that* $(\|\hat{C}\Delta\|_{\mathrm{F}}/2^j) \le \frac{1}{2}$. *Set* $t_0 = (\Delta/2^j)$.

Step 2. *For some* $q \ge 1$, *compute*

$$Y_0 = R_{qq}\left(\frac{\hat{C}\Delta}{2^j}\right),$$

where R_{qq} *is the* (q,q) *Padé approximant to* e^z:

$$R_{qq}(z) = \frac{\sum_{k=0}^{z} c_k z^k}{\sum_{k=0}^{q} c_k(-z)^k}, \qquad \text{where } c_k = \frac{(2q-k)!q!}{(2q)!k!(q-k)!}.$$

Write

$$Y_0 = \begin{pmatrix} F_1(t_0) & G_1(t_0) & H_1(t_0) & K_1(t_o) \\ 0 & F_2(t_0) & G_2(t_0) & H_2(t_0) \\ 0 & 0 & F_3(t_0) & G_3(t_0) \\ 0 & 0 & 0 & F_4(t_0) \end{pmatrix}$$

and set

$$\begin{aligned} F_0 &= F_3(t_0) & M_0 &= F_3(t_0)^{\mathrm{T}} H_2(t_0) \\ H_0 &= G_3(t_0) & W_0 &= [B^{\mathrm{T}} F_3(t_0)^{\mathrm{T}} K_1(t_0)] + (B^{\mathrm{T}} F_3(t_0)^{\mathrm{T}} K_1(t_0)]^{\mathrm{T}}. \\ Q_0 &= F_3(t_0)^{\mathrm{T}} G_2(t_0). \end{aligned}$$

Step 3. *For* $k = 0, 1, \ldots, j-1$ *do*

$$\begin{aligned} W_{k+1} &= 2W_k + H_k^{\mathrm{T}} M_k + M_k^{\mathrm{T}} H_k + H_k^{\mathrm{T}} Q_k H_k \\ M_{k+1} &= M_k + F_k^{\mathrm{T}}[Q_k H_k + M_k] \\ Q_{k+1} &= Q_k + F_k^{\mathrm{T}} Q_k F_k \\ H_{k+1} &= H_k + F_k H_k \\ F_{k+1} &= F_k^2 \end{aligned}$$

End

Step 4. *Set* $F \equiv F_j$, $H \equiv H_j$, $Q \equiv Q_j$, $M \equiv M_j$, *and* $W \equiv W_j$.

Remark

- It has been proved by Van Loan (1978) that the accuracy of the algorithm can be controlled by selecting q properly. For **"how to choose q properly"** and other computational details, see the paper of Van Loan (1978).

MATCONTROL note: Algorithm 5.3.3 has been implemented in MATCONTROL function **intmexp**.

5.4 STATE-SPACE SOLUTION OF THE DISCRETE-TIME SYSTEM

In this section, we state a discrete-analog of Theorem 5.3.1, and then discuss how to compute the discrete-time solution.

Theorem 5.4.1. *Discrete-Time State-Space Solution. The solutions to the linear time-invariant discrete-time system of equations*

$$x(k+1) = Ax(k) + Bu(k), \quad x(0) = x_0 \tag{5.4.1}$$

and

$$y(k) = Cx(k) + Du(k) \tag{5.4.2}$$

are

$$x(k) = A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} Bu(i) \tag{5.4.3}$$

and

$$y(k) = CA^k x_0 + \left\{ \sum_{i=0}^{k-1} CA^{k-i-1} Bu(i) \right\} + Du(k). \tag{5.4.4}$$

Proof. From

$$x(k+1) = Ax(k) + Bu(k), \tag{5.4.5}$$

we have

$$\begin{aligned} x(k) &= A[Ax(k-2) + Bu(k-2)] + Bu(k-1), \\ &= A^2 x(k-2) + ABu(k-2) + Bu(k-1), \\ &= A^2[Ax(k-3) + Bu(k-3)] + ABu(k-2) + Bu(k-1), \\ &\vdots \\ &= A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} Bu(i). \end{aligned}$$

This proves (5.4.3).

Equation (5.4.4) is now obtained by substituting the expression of $x(t)$ from (5.4.3) into (5.4.2). ■

Computing the Powers of a Matrix

Theorem 5.4.1 shows that, to find the state-space solution of a discrete-time system, one needs to compute the various powers of A. The powers of a matrix A are more easily computed if A is first decomposed into a condensed form by similarity. Thus, if $P^{\mathrm{T}}AP = R$, where P is orthogonal, then $A^s = PR^sP^{\mathrm{T}}$. **For the sake of numerical stability, only those condensed forms such as the Hessenberg form or the RSF of a matrix should be considered.** The matrix R^s can be easily calculated by exploiting the Hessenberg or Schur structure of R. Furthermore, the reduction to R can be achieved using a numerically stable procedure such as Householder's or Givens' method (**Chapter 4**).

5.5 TRANSFER FUNCTION AND FREQUENCY RESPONSE

In this section, we introduce the important concepts of **transfer function** and **frequency response** matrices and describe a numerical algorithm for computing the frequency response matrix.

5.5.1 Transfer Function

Consider

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0, \tag{5.5.1}$$

$$y(t) = Cx(t) + Du(t). \tag{5.5.2}$$

Let $\hat{x}(s)$, $\hat{y}(s)$, and $\hat{u}(s)$, respectively, denote the Laplace transforms of $x(t)$, $y(t)$, and $u(t)$. Then taking the Laplace transform of (5.5.1) and (5.5.2), we obtain

$$s\hat{x}(s) - x_0 = A\hat{x}(s) + B\hat{u}(s), \tag{5.5.3}$$

$$\hat{y}(s) = C\hat{x}(s) + D\hat{u}(s). \tag{5.5.4}$$

From (5.5.3) and (5.5.4), we have

$$\hat{x}(s) = R(s)x(0) + R(s)B\hat{u}(s) \tag{5.5.5}$$

$$\hat{y}(s) = CR(s)x(0) + G(s)\hat{u}(s), \tag{5.5.6}$$

where

$$R(s) = (sI - A)^{-1} \tag{5.5.7}$$

and

$$G(s) = C(sI - A)^{-1}B + D. \tag{5.5.8}$$

If $x(0) = 0$, then (5.5.6) gives

$$\hat{y}(s) = G(s)\hat{u}(s).$$

Definition 5.5.1. *The matrix $R(s)$ is called the* **resolvent** *and $G(s)$ is called the* **transfer function.**

The transfer function $G(s)$ is a matrix transfer function of dimension $r \times m$. Its (i, j)th entry denotes the transfer function from the jth input to the ith output. That is why, it is also referred to as the **transfer function matrix** or simply the **transfer matrix.**

Definition 5.5.2. *The points p at which $G(p) = \infty$ are called the* **poles** *of the system.*

If $G(\infty) = 0$, then the transfer function is called **strictly proper** *and is* ***proper*** *if $G(\infty)$ is a constant matrix.*

State-Space Realization

For computational convenience, the transfer function matrix $G(s)$ will be written sometimes as

$$G(s) = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right].$$

The state-space model (A, B, C, D) having $G(s)$ as its transfer function matrix is called a realization of $G(s)$. For more on this topic, see **Chapter 9. In general,** it will be assumed that $G(s)$ **is a real-rational transfer matrix that is proper.**

Operations with Transfer Function Matrices

Let $G_1(s)$ and $G_2(s)$ be the transfer functions of the two systems S_1 and S_2. Then the transfer function matrix of the **parallel connection** of S_1 and S_2 is $G_1(s) + G_2(s)$. Using our notation above, we obtain:

$$G_1(s)+G_2(s) = \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right] + \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array}\right] = \left[\begin{array}{cc|c} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ \hline C_1 & C_2 & D_1 + D_2 \end{array}\right].$$

Similarly, the transfer function matrix of the **series** or **cascade connection** of S_1 and S_2 (i.e., a system with the output of the second system as the input of the first system) is $G_1(s)G_2(s)$, given by

$$G_1(s)G_2(s) = \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right]\left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array}\right] = \left[\begin{array}{c|cc} A_2 & 0 & B_1 \\ B_1C_2 & A_1 & B_1D_2 \\ \hline D_1C_2 & C_1 & D_1D_2 \end{array}\right].$$

The **transpose** of a transfer function matrix $G(s)$ is defined as:

$$G^{\mathrm{T}}(s) = B^{\mathrm{T}}(sI - A^{\mathrm{T}})^{-1}C^{\mathrm{T}} + D^{\mathrm{T}},$$

or equivalently,

$$G^{\mathrm{T}}(s) = \left[\begin{array}{c|c} A^{\mathrm{T}} & B^{\mathrm{T}} \\ \hline C^{\mathrm{T}} & D^{\mathrm{T}} \end{array}\right].$$

The **conjugate** of $G(s)$ is defined as:

$$G^{\sim}(s) \equiv G^{\mathrm{T}}(-s) = B^{\mathrm{T}}(-sI - A^{\mathrm{T}})^{-1}C^{\mathrm{T}} + D^{\mathrm{T}},$$

or equivalently,

$$G^{\sim}(s) = \left[\begin{array}{c|c} -A^{\mathrm{T}} & -C^{\mathrm{T}} \\ \hline -B^{\mathrm{T}} & D^{\mathrm{T}} \end{array}\right].$$

The **inverse** of a transfer function matrix $G(s)$, denoted by $\hat{G}(s)$ is such that $G(s)\hat{G}(s) = \hat{G}(s)G(s) = I$. If $G(s)$ is square and D is invertible, then,

$$\hat{G}(s) \equiv G^{-1}(s) = \left[\begin{array}{c|c} A - BD^{-1}C & -BD^{-1} \\ \hline D^{-1}C & D^{-1} \end{array}\right].$$

MATLAB notes: MATLAB functions **parallel, series, transpose, inv (ss/inv.m)** can be used to compute parallel, series, transpose, and inverse, respectively.

Transfer Function of Discrete-Time System

The transfer function matrix of the discrete-time system (5.4.1)–(5.4.2) is

$$G(z) = C(zI - A)^{-1}B + D.$$

It is obtained by taking the z-transform of the system.

5.5.2 The Frequency Response Matrix and its Computation

In this section, we describe a computationally viable approach for computing the **frequency response matrix**.

Definition 5.5.3. *For the continuous-time state-space model (5.5.1–5.5.2), the matrix*

$$G(j\omega) = C(j\omega I - A)^{-1}B + D \tag{5.5.9}$$

is called the **frequency response matrix**; $\omega \in \mathbb{R}$ *is called* ***frequency***.

The frequency response matrix of a discrete-time system is similarly defined by using the transformation:

$$z = e^{j\omega T},$$

where T is the sample time. This transformation maps the frequencies ω to points on the unit circle. The frequency response matrix is then evaluated at the resulting z values.

Computing the Frequency Response Matrix

In order to study the different responses of the system, it is necessary to compute $G(j\omega)$ for many different values of ω. Furthermore, the singular values of the **return difference matrix** $I + L(j\omega)$ and of the inverse return difference matrix $I + L^{-1}(j\omega)$, where $L(j\omega)$ is square and of the form $L = KGM$ for appropriate K and M, **provide robustness measure of a linear system with respect to stability, noise, disturbance attenuation, sensitivity, etc.** (Safonov *et al.* 1981).

We therefore describe a numerical approach to compute $G(j\omega)$. **For simplicity, since D does not have any computational role in computing $G(j\omega)$, we will assume that $D = 0$.**

The computation of $(j\omega I - A)^{-1}B$ is equivalent to solving m systems:

$$(j\omega I - A)X = B,$$

A usual scheme for computing the frequency response matrix is:

Step 1. Solve the m systems for m columns $x_1, x_2, \ldots, x_m$ of X:

$$(j\omega I - A)x_i = b_i, \qquad i = 1, 2, \ldots, m,$$

where b_i is the ith column of B.

Step 2. Compute CX.

If A is $n \times n$, B is $n \times m$ ($m \le n$), and C is $r \times n$ ($r \le n$), and if LU factorization (**Chapter 3**) is used to solve the systems $(j\omega I - A)x_i = b_i$, $i = 1, 2, \ldots, m$, then the total flop-count **(complex) for each ω** is about $\frac{2}{3}n^3 + 2mn^2 + 2mnr$. Note that, since the matrix $j\omega I - A$ is the same for each linear system for a given ω, the matrix $j\omega I - A$ has to be factored into LU only once, and the same factorization can be used to solve all the m systems. Since the matrix $G(j\omega)$ needs to be computed for many different values of ω, the computation in this way will be fairly expensive.

Laub (1981) noted that the computational cost can be reduced significantly when $n > m$ (which is normally the case in practice), if the matrix A is initially transformed to an upper Hessenberg matrix H by orthogonal similarity, before the frequency response computation begins. The basic observation here is that if $P^{\mathrm{T}}AP = H$, an upper Hessenberg matrix, then

$$\begin{aligned} G(j\omega) &= C(j\omega I - A)^{-1}B, \\ &= C(j\omega I - PHP^{\mathrm{T}})^{-1}B, \\ &= C(P(j\omega I - H)P^{\mathrm{T}})^{-1}B, \\ &= CP(j\omega I - H)^{-1}P^{\mathrm{T}}B. \end{aligned}$$

Thus, if A is transformed initially to an upper Hessenberg matrix H, and $CP = C'$ and $P^{\mathrm{T}}B = B'$ are computed once for all, then for each ω, we have the following algorithm.

Algorithm 5.5.1. *A Hessenberg Algorithm for the Frequency Response Matrix.*

Input. A—**The** $n \times n$ **state matrix**
ω—Frequency, a real number
B—The $n \times m$ input matrix
C—The $r \times n$ output matrix.

Output. *The Frequency Response Matrix $G(j\omega) = C(j\omega I - A)^{-1}B$.*

Step 1. *Transform A to an upper Hessenberg matrix H (**Section 3.5.3**):* $P^{\mathrm{T}}AP = H$.

Step 2. *Compute $B' = P^{\mathrm{T}}B$ and $C' = CP$*

Step 3. *Solve the m **Hessenberg** systems:*

$$(j\omega I - H)x_i = b_i', \quad i = 1, \ldots, m,$$

where b_i' is the ith column of B'.

Step 4. *Compute $C'X$.*

Flop-Count: Since the system matrices A, B, and C are real, Steps 1 and 2 can be done using real arithmetic and require approximately $\frac{10}{3}n^3 + 2mn^2 + 2rn^2$ (real) flops. Steps 3 and 4 require complex arithmetic and require approximately $2mn^2 + 2rnm$ **complex** flops.

Comparison: For N values of ω, the Hessenberg method require $\frac{10}{3}n^3 + 2(m + r)n^2$ real $+[2mn^2 + 2rnm]N$ complex flops compared to $[\frac{2}{3}n^3 + 2mn^2 + +2mnr]N$ complex flops, required by the non-Hessenberg scheme.

Numerical stability: It is possible to show (Laub and Linnemann 1986) that if the data is well-conditioned, then the frequency response of the computed Hessenberg form is $(C + \triangle C)(j\omega I - A - \triangle A)^{-1}(B + \triangle B)$, where $\triangle A$, $\triangle B$, and $\triangle C$ are small. Thus, the **Hessenberg method is stable.**

MATCONTROL note: Algorithm 5.5.1 has been implemented in MATCONTROL function **freqresh**.

Example 5.5.1. Compute the frequency response matrix with

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 0 & 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \omega = 1.$$

Since A is already in upper Hessenberg form, Steps 1 and 2 are skipped.

Step 3. Solve for x_1: $(j\omega I - H)x_1 = b_1' = b_1$,

$$x_1 = \begin{pmatrix} -5.000 - 0.0000i \\ 0.4000 - 0.8000i \\ +0.1000 - 0.7000i \end{pmatrix}.$$

Solve for x_2: $(j\omega I - H)x_2 = b_2' = b_2$,

$$x_2 = \begin{pmatrix} -0.5000 - 0.0000i \\ 0.4000 + 0.8000i \\ 0.1000 - 0.7000i \end{pmatrix}.$$

Step 4. Compute the frequency response matrix:

$$G(j\omega) = C'X = CX,$$
$$= \begin{pmatrix} -0.8000 + 0.1000i & -0.8000 + 0.1000i \\ -0.8000 + 0.1000i & -0.8000 + 0.1000i \end{pmatrix}.$$

Other Methods for Frequency Response Computation

1. A method based on a determinant identity for the computation of the frequency response matrix has been proposed by Misra and Patel (1988). The method seems to be considerably faster and at least as accurate as the Hessenberg method just described. The method uses the controller-Hessenberg and observer-Hessenberg forms which will be described in the next chapter. The Misra–Patel method for computing the frequency response matrix is based on the following interesting observation:

$$g_{lk}(j\omega) = \frac{\det(j\omega I - A + b_k c_l^{\mathrm{T}})}{\det(j\omega I - A)} - 1, \tag{5.5.10}$$

where b_k and c_l denote, respectively, the kth and lth columns of the matrices B and C.

2. An alternative method also based on the reduction to controller-Hessenberg form, has been given by Laub and Linnemann (1986).

3. Another method for the problem has been proposed by Kenney *et al.* (1993). The method uses matrix interpolation techniques and seems to have better efficiency than the Hessenberg method.
4. The frequency response of a discrete time system is obtained by evaluating the transfer function $H(z)$ in (5.5.8) on the unit circle.

Bode Diagram and the Singular Value Plot

Traditionally, Bode diagram is used to measure the magnitude and angle of frequency response of an SISO system. Thus expressing the frequency response function in polar coordinates, we have

$$G(j\omega) = M(\omega)e^{j\sigma(\omega)}.$$

$M(\omega)$ is the **magnitude** and $\sigma(\omega)$ is the **angle**. It is customary to express $M(\omega)$ in decibels (abbreviated by dB). Thus,

$$M(\omega)|_{\mathrm{dB}} = 20\log_{10} M(\omega).$$

The angle is measured in degrees.

The **Singular Value Plot** is the plot of singular values of $H(j\omega)$ as a function of the frequency ω. If the system is an SISO system, the singular value plot is the same as the Bode diagram. The singular value plot is a useful tool in robustness analysis.

MATLAB note: MATLAB functions **bode** and **sigma** can be used to draw the Bode plot and the singular value plot, respectively. For the Bode plot of multi-input, multi-output (MIMO) system, the system is treated as arrays of SISO systems and the magnitudes and phases are computed for each SISO entry h_{ij} independently.

MATLAB function **freqresp** can be used to compute frequency response at some specified individual frequencies or over a grid of frequencies. When numerically safe, the frequency response is computed by diagonalizing the matrix A; otherwise, Algorithm 5.5.1 is used.

5.6 SOME SELECTED SOFTWARE

5.6.1 Matlab Control System Toolbox

Time response

step	Step response
impulse	Impulse response
initial	Response of state-space system with given initial state

lsim	Response to arbitrary inputs
ltiview	Response analysis GUI
gensig	Generate input signal for LSIM
stepfun	Generate unit-step input.

Frequency response

bode	Bode plot for the frequency response
sigma	Singular value frequency plot
nyquist	Nyquist plot
nichols	Nichols chart
ltiview	Response analysis GUI
evalfr	Evaluate frequency response at given frequency
freqresp	Frequency response over a frequency grid
margin	Gain and phase margins

5.6.2 MATCONTROL

EXPMPADE	The Padé approximation to the exponential of a matrix
EXPMSCHR	Computing the exponential of a matrix using Schur decomposition
FREQRESH	Computing the frequency response matrix using Hessenberg decomposition
INTMEXP	Computing an integral involving a matrix exponentials.

5.6.3 SLICOT

MB05MD	Matrix exponential for a real non-defective matrix
MB05ND	Matrix exponential and integral for a real matrix
MB05OD	Matrix exponential for a real matrix with accuracy estimate.

State-space to rational matrix conversion

TB04AD	Transfer matrix of a state-space representation.

State-space to frequency response

TB05AD	Frequency response matrix of a state-space representation
TF	Time response
TF01MD	Output response of a linear discrete-time system
TF01ND	Output response of a linear discrete-time system (Hessenberg matrix).

5.6.4 MATRIX$_X$

Purpose: Gain and phase plots of discrete-time systems.

Syntax:
[GAIN, DB, PHASE]=DBODE (SD, NS, OMEGANMIN, OMEGANMAX, NPTS, 'OPT') OR
[GAIN, DB, PHASE]=DBODE (DNUM, DDEN, OMEGANMIN, OMEGANMAX, NPTS) OR
[GIAN, DB, PHASE]=DBODE (SD, NS, OMEGAN)

Purpose: Initial value response of discrete-time dynamic system.

Syntax: [N, Y]=DINITIAL (SD, NS, XO, NPTS).

Purpose: Step-response of discrete-time system.

Syntax: [N, Y]=DSTEP (SD, NS, NPTS) OR
[N, Y]=DSTEP (DNUM, DDEN, NPTS)

Purpose: Frequency response of dynamic system. FREQ transforms the A matrix to Hessenberg form prior to finding the frequency response.

Syntax: [OMEGA, H]=FREQ (S, NS, RANGE, option) OR
H=FREQ (S, NS, OMEGA, 'options')

Purpose: Compute the impulse response of a linear continuous-time system.
Syntax: [T, Y]=IMPULSE (S, NS, TMAX, NPTS) OR
[T, Y]=IMPULSE (NUM, DEN, TMAX, NPTS)

Purpose: Initial value response of continuous-time dynamic system.

Syntax: [T, Y]=INITIAL (S, NS, XO, TMAX, NPTS)

Purpose: Response of continuous-time system to general inputs.

Syntax: [T, Y]=LSIM (S, NS, U, DELTAT, X0)

Purpose: Step response of continuous-time system.

Syntax: [T, Y]=STEP (S, NS, TMAX, NPTS) OR
[T, Y]=STEP (NUM, DEN, TMAX, NPTS)

Purpose: Gives transfer function form of a state-space system.

Syntax: [NUM, DEN]=TFORM (S, NS)

Purpose: Impulse response of continuous-time system.

Syntax: [T, Y]=TIMR (S, NS, RANGE, 'MODE')

5.7 SUMMARY AND REVIEW

State-Space Representations

A **continuous-time** linear time-invariant control system may be represented by systems of differential equations of the form (5.2.1) – (5.2.2)

$$\dot{x}(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t) + Du(t),$$

where $x(t)$ is the **state vector**, $u(t)$ is the **input vector**, $y(t)$ is the **output vector**.

The matrices A, B, C, and D are **time-invariant matrices** known, respectively, as the **state matrix**, the **input matrix**, the **output matrix**, and the **direct transmission matrix**.

A **discrete-time** linear time-invariant control system may analogously be represented by systems of difference equations (5.4.1) – (5.4.2).

$$x(t+1) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t) + Du(t).$$

where $x(t)$, $u(t)$, $y(t)$, and A, B, C, and D have the same meaning as above.

Solutions of the Dynamical Equations

The solutions of the equations representing the continuous-time system in state-space form are given by (assuming $t_0 = 0$):

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)\,ds$$

and

$$y(t) = Ce^{At}x_0 + \int_0^t Ce^{A(t-s)}Bu(s)\,ds + Du(t).$$

where x_0 is the value of $x(t)$ at $t = 0$. The matrix e^{At} is the **state-transition matrix in this case**.

The solutions of the equations representing the discrete-time system are given by:

$$x(k) = A^k x(0) + \sum_{i=0}^{k-1} A^{k-1-i} Bu(i)$$

and

$$y(k) = CA^k x(0) + \left\{ \sum_{i=0}^{k-1} CA^{k-i-1} Bu(i) \right\} + Du(k).$$

Computing e^{At}

There exist several methods for computing the state-transition matrix e^{At}. These include: The Taylor series method, the Padé approximation method, ODE methods, the eigenvalue–eigenvector method, and the matrix decomposition methods.

Among these, **the Padé approximation method with scaling and squaring (Algorithm 5.3.1)** and the **method based on the Schur decomposition of** A **(Algorithm 5.3.2) are the ones that are recommended for use in practice.** If the problem is ill-conditioned, these methods, however, might not give accurate results. The **ODE methods (Section 5.3.3) should be preferred if** A **is large and sparse.**

Computing Integrals Involving Matrix Exponentials

An algorithm **(Algorithm 5.3.3)** is presented for computing integrals involving matrix exponentials.

Transfer Function Matrix

If $\hat{u}(s)$ and $\hat{y}(s)$ are the Laplace transforms of $u(t)$ and $y(t)$, then assuming zero initial condition, we obtain:

$$\hat{y}(s) = G(s)\hat{u}(s),$$

where

$$G(s) = C(sI - A)^{-1}B + D.$$

The matrix $G(s)$ is called the transfer function matrix and is conveniently written as:

$$G(s) \equiv \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array} \right].$$

The Frequency Response Matrix

The matrix $G(j\omega) = C(j\omega I - A)^{-1}B + D$ is called the **frequency response matrix.**

The frequency response plot for different values of ω is important in the study of different responses of a control system. For this the frequency response matrix needs to be computed.

An efficient method (**Algorithm 5.5.1**), based on transformation of A to Hessenberg form, is described. **The Hessenberg method is nowadays widely used in practice.**

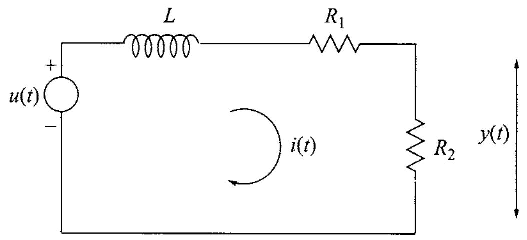
5.8 CHAPTER NOTES AND FURTHER READING

The examples on state-space model in Section 5.2.1 have been taken from various sources. These include the books by Brogan (1991), Chen (1984), Kailath (1980), Luenberger (1979), Szidarovszky and Bahill (1991), Zhou with Doyle (1998). Discussions on system responses can be found in any standard text books. The books mentioned above and also the books by DeCarlo (1989), Dorf and Bishop (2001), Friedland (1986), Patel and Munro (1982), etc., can be consulted. For various ways of computing the matrix exponential e^{At}, the paper by Moler and Van Loan (1978) is an excellent source. Some computational aspects of the matrix exponential have also been discussed in DeCarlo (1989).

The frequency response algorithm is due to Laub (1981). For discussions on applications of frequency response matrix, see Safonov *et al.* (1981). For alternative algorithms for frequency response computation, see Misra and Patel (1988), Kenney *et al.* (1993). The algorithm for computing integrals (**Algorithm 5.3.3**) involving matrix exponential has been taken from the paper of Van Loan (1978). The sensitivity analysis of the matrix e^{At} is due to Van Loan (1977). See also Golub and Van Loan (1996).

Exercises

5.1 (a) Consider the electric circuit

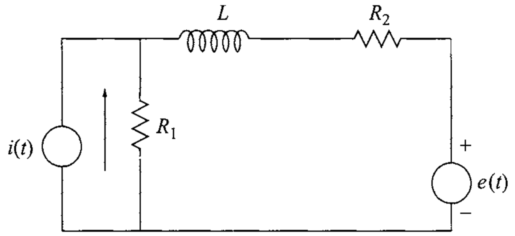


(i) Show that the state-space representation of this circuit is given by

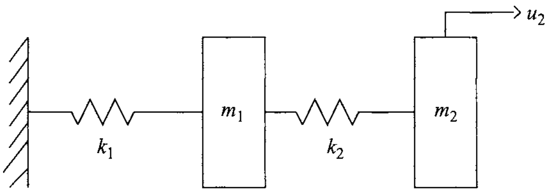
$$L\frac{di(t)}{dt} + (R_1 + R_2)i(t) = u(t), \quad y(t) = R_2 i(t).$$

(ii) Give an explicit expression for the solution of the state equation.

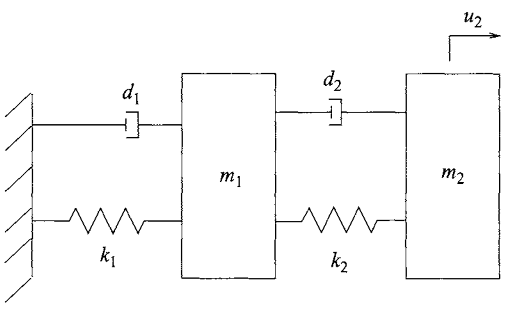
(b) Write the state equations for the following electric network:



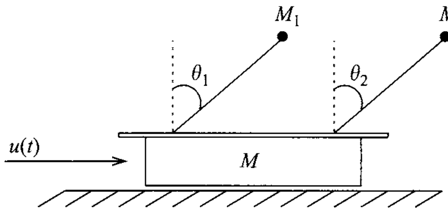
5.2 (a) Write down the equations of motion of the following spring-mass system in state-space form:



(b) Write down the state-space representation of the following spring-mass system:



5.3 Consider the following diagram of a cart of mass M carrying two sticks of masses M_1 and M_2, and lengths l_1 and l_2.



(a) Write down the equations of motion, in terms of the velocity v of the cart, u and $\theta_1, \theta_2, \dot{\theta}_1, \ddot{\theta}_2$.

(b) Assuming θ_1 and θ_2 are small, linearize the equations of motion and then write down a first-order state-space representation.

5.4 (Saad 1988) Consider the differential equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \beta \frac{\partial u}{\partial x} + \nu u + F(x, y, t)$$

on the unit square $\Omega = (0, 1) \times (0, 1)$, that models a chemical reaction process, where u represents the concentration of a chemical component that diffuses and convects. Let the boundary conditions $u(x, y, t) = 0$ for every t. Assume that $F(x, y, t)$ has the form:

$$F(x, y, t) = F(x, y)g(t).$$

The term νu simulates a chemical reaction that results in an increase of the concentration that is proportional to u.

(a) Discretize the region with n interior points in the x-direction and m interior points in the y-direction and show that this leads to the state-space representation of the form:

$$\dot{u} = Au + bg,$$

where the dimension of A is nm.

(b) Solve the above problem on a computer with $\beta = 20, \nu = 180, n = 20, m = 10$.

5.5 **(Lanchester War Model)** The following simple model of Warfare was developed by F. Lanchester in 1916.

Let x_1 and x_2 be the number of units in the two forces which are engaged in a war. Suppose that each unit of the first force has the "hitting" power h_1 and that of the second force has the "hitting" power h_2.

The "hitting" power is defined to be the number of casualties per unit time that one member can inflict on the other.

Suppose further that the hitting power of each force is directed uniformly against all units of the other force.

(a) Develop a state-space representation of the model.

(b) Show that

$$x_1(t) = c_1 e^{t\sqrt{h_1 h_2}} + c_2 e^{-t\sqrt{h_1 h_2}},$$

$$x_2(t) = -c_1\sqrt{\frac{h_1}{h_2}}e^{t\sqrt{h_1 h_2}} + c_2\sqrt{\frac{h_1}{h_2}}e^{-t\sqrt{h_1 h_2}},$$

where c_1 and c_2 are constants to be determined from initial conditions.

5.6 (a) Find an explicit expression for the solution of the initial value problem

$$\dot{x}(t) = \begin{pmatrix} 0 & \lambda \\ -\lambda & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

(b) Find the free response of the system.

5.7 Find an explicit expression for the solution of the homogeneous discrete-time system

$$x(t+1) = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} x(t), \qquad x(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

5.8 Prove the following properties of e^{At}:

(a) $e^{A(t+s)} = e^{At} \cdot e^{As}$,

(b) $e^{(A+B)t} = e^{At} \cdot e^{Bt}$ if and only if A and B commute,

(c) e^{At} is nonsingular and $(e^{At})^{-1} = e^{-At}$,

(d) $\left(e^{A/m}\right)^m = e^A$, where m is an integer,

(e) $e^{P^{-1}APt} = P^{-1}e^{At}P$.

5.9 Prove that the infinite series

$$e^{At} = \sum_{k=0}^{\infty} \frac{1}{k!} A^k t^k$$

converges uniformly and absolutely for t in any bounded interval.

5.10 Prove that $(d/dt)(e^{At}) = Ae^{At}$
$\left(\mathit{Hint}: \text{ Differentiate } e^{At} = \sum_{k=0}^{\infty} 1/k! A^k t^k \text{ term by term}\right)$.

5.11 Illustrate the difficulty with the eigenvalue–eigenvector method for computing e^{At} with the matrix:

$$A = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}$$

by choosing λ, μ, and α appropriately.

5.12 Let $R = (r_{ij})$ be an unreduced lower Hessenberg matrix and let

$$e^R = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_r \end{pmatrix} \quad \text{and} \quad B_i = R - r_{ii}I.$$

Then prove that

$$f_2 = \frac{1}{r_{12}} f_1 B_1$$

and

$$f_{i+1} = \frac{1}{r_{i-1,i}} \left(f_i B_i - \sum_{j=1}^{i-1} r_{ij} f_j \right), \quad i = 2, 3, \ldots, n-1.$$

(Consult Datta and Datta (1976), if necessary.) What difficulties do you foresee in computing e^R using these formulas? Give an illustrative example.

5.13 Compute e^A for

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

using

(a) a fifth-order variable-step Runge–Kutta method;

(b) the Adams–Moulton predictor correct formulas of variable order and variable step;

(c) a general purpose ODE solver. (Use **error tolerance** 10^{-6}.)

Compare the result in each case with that obtained by MATLAB function **expm**.

5.14 (a) Write an algorithm based on the block diagonal decomposition of A to compute e^{At}:

$$A = X \operatorname{diag}(T_1, T_2, \ldots, T_p) X^{-1}.$$

(b) Determine the flop-count of this algorithm.

(c) What numerical difficulty do you expect out of this algorithm?

5.15 Prove that the matrix $D_{pq}(A)$ in the Padé approximation method is nonsingular if all the eigenvalues of A have negative real parts or if p and q are sufficiently large.

5.16 Develop an algorithm to compute A^s, where s is a positive integer, and A is an unreduced lower Hessenberg matrix. Apply your algorithm to compute A^{10}, where $A =$

$$\text{(i)} \begin{pmatrix} 1 & 2 & 0 \\ 1 & 1 & 0.0001 \\ 1 & 1 & 1 \end{pmatrix}, \quad \text{(ii)} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \quad \text{(iii)} \begin{pmatrix} 1 & 10^{-3} & 0 \\ 1 & 1 & 10^{-4} \\ 1 & 1 & 1 \end{pmatrix}.$$

(Consult Datta and Datta 1976.)

5.17 Let $A = X\Lambda X^{-1}$, where $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Prove that $A^k = X\Lambda^k X^{-1}$, for each k. Under what conditions on $\lambda_1, \ldots, \lambda_n$ does the infinite series of matrices $\sum c_k A^k$ converge?

5.18 Develop an algorithm to compute A^s, where A is an upper real Schur matrix, and s is a positive integer. Apply your algorithm to compute A^5, where

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 0.99 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1.99 \end{pmatrix}.$$

5.19 Prove that the Laplace transform of $y(t) = e^{At}$ is $y(s) = (sI - A)^{-1}$.

5.20 Modify Algorithm 5.3.2 to compute e^A, where A is in RSF.

5.21 Show that the transformation $\tilde{x} = Tx$, where T is nonsingular, preserves the transfer function.

5.22 Show that the transfer function of the system:

$$\dot{x}(t) = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u, \qquad x(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$y = (1, 1)x.$$

is $H(s) = \dfrac{s + \omega}{s^2 + \omega^2}$.

5.23 Modify the Hessenberg algorithm (Algorithm 5.5.1) for computation of the frequency response matrix that uses only real arithmetic. Give a flop-count of this modified algorithm.

5.24 Develop an algorithm for computing the frequency response matrix using formula (5.5.10) and the fact that the determinant of a matrix A is just the product of the diagonal entries of the matrix U in its LU factorization. (Consult Misra and Patel 1988.)

5.25 Develop an algorithm for computing the frequency response matrix based on the reduction of A to RSF. Compare the flop-count of this algorithm with that of the Hessenberg algorithm (Algorithm 5.5.1).

5.26 Develop an algorithm for computing the frequency response matrix of the descriptor model:

$$E\dot{x} = Ax + Bu$$

based on the initial reduction of the pair (A, B) to Hessenberg-triangular form described in Chapter 4. (Consult Misra 1989.)

References

Ali Sayed, *Lecture Notes on Dynamical Systems*, University of California, Los Angeles, 1994.

Baldwin C.T. *Fundamentals of Electrical Measurements*, George G. Harp and Co. Ltd., London, 1961.

Brogan W.L. *Modern Control Theory*, 3rd edn, Prentice Hall, Englewood Cliffs, NJ, 1991.

Campbell S.L. *Singular Systems of Differential Equations,* Pitman, Marshfield, MA, 1980.

Chen C.-T. *Linear System Theory and Design*, CBS College Publishing, New York, 1984.

Datta B.N. and Datta, K. "An algorithm to compute the powers of a Hessenberg matrix and applications," *Lin. Alg. Appl.* Vol. 14, pp. 273–284, 1976.

DeCarlo R.A. *Linear Systems: A State Variable Approach with Numerical Implementation*, Prentice Hall, Englewood Cliffs, NJ, 1989.

Dorf R.C. and Bishop R.H. *Modern Control Systems*, 9th ed, Prentice Hall, Upper Saddle River, NJ, 2001

Friedland B. *Control Systems Design: An Introduction to State-Space Methods*, McGraw-Hill, New York, 1986.

Golub G.H. and Van Loan C.F. *Matrix Computations*, 3rd edn, Johns Hopkins University Press, Baltimore, MD, 1996.

Kailath T. *Linear Systems*, Prentice Hall, Englewood Cliffs, NJ, 1980.

Kenney C.S., Laub A.J., and Stubberud, S.C. "Frequency response computation via rational interpolation," *IEEE Trans. Autom. Control*, Vol. 38, pp. 1203–1213, 1993.

Laub A.J. "Efficient multivariable frequency response computations," *IEEE Trans. Autom. Control*, Vol. AC-26, pp. 407–408, 1981.

Laub A.J. and Linnemann A. "Hessenberg and Hessenberg/triangular forms in linear systems theory," *Int. J. Control*, Vol. 44, pp. 1523–1547, 1986.

Luenberger D.G. *Introduction to Dynamic Systems: Theory, Methods, & Applications*, John Wiley & Sons, New York, 1979.

Misra P. "Hessenberg-triangular reduction and transfer function matrices of singular systems," *IEEE Trans. Circuits Syst.*, Vol. CAS-36, pp. 907–912, 1989.

Misra P. and Patel R.V. "A determinant identity and its application in evaluating frequency response matrices," *SIAM J. Matrix Anal. Appl.*, Vol. 9, pp. 248–255, 1988.

Moler C.B. and Van Loan C.F. "Nineteen dubious ways to compute the exponential of a matrix," *SIAM Rev.*, Vol. 20, pp. 801–836, 1978.

Parlett B.N. "A recurrence among the elements of functions of triangular matrices," *Lin. Alg. Appl.*, Vol. 29, pp. 323–346, 1976.

Patel R.V. and Munro N. *Multivariable Systems Theory and Design,* Pergamon Press, Oxford, UK, 1982.

Safonov M.G., Laub A.J. and Hartman, G.L. "Feedback properties of multivariable systems: the role and use of the return difference matrix," *IEEE Trans. Autom. Control*, Vol. AC-26, pp. 47–65, 1981.

Saad Y. "Projection and deflation methods for partial pole assignment in state feedback," IEEE Trans. Automat. Control, Vol. 33, No. 3, pp. 290–297, 1988.

Soong T.T. *Active Structural Control: Theory and Practice*, Longman Scientific and Technical, Essex, UK, 1990.

Szidarovszky F. and Terry Bahill, A. *Linear Systems Theory*, CRC Press, Boca Raton, 1991.

Van Loan C.F. "The sensitivity of the matrix exponential," *SIAM J. Numer. Anal.*, Vol. 14, pp. 971–981, 1977.

Van Loan C.F. "Computing integrals involving the matrix exponential," *IEEE Trans. Autom. Control*, Vol. AC-23, pp. 395–404, 1978.

Zhou K. (with Doyle J.), *Essentials of Robust Control*, Prentice Hall, Upper Saddle River, NJ, 1998.