Chapter 7
Dynamic Systems: Ordinary Differential Equations

7.1 Introduction

The mathematical modeling of physiological systems will often result in ordinary or partial differential equations. The fundamental reason underlying this is that biosystems are dynamic in nature. Their behavior constantly evolves with time or varies with respect to position in space. In this chapter we will consider the numerical solution of ordinary differential equations. These are the models that arise from the study of the dynamics of physiological systems that have one independent variable. The latter may be either the space variable x, or the time variable t, depending on the geometry of the system and its boundary conditions. Ordinary differential equations may arise from modeling the metabolic pathways of living cells, the complex interactions of pharmacokinetics, the kinetics of the oxygen/ hemoglobin system, the transfer of nutrients across cells, the dynamics of membrane and nerve cell potentials, the transformation and replication of stem cells, the mechanism of migration and binding of tissue cells, or the dynamics of interacting populations of bacteria and the human species.

The material in this chapter will enable the student to accomplish the following:

- Model the dynamics of physiological systems using ordinary differential equations.
- Obtain numerical solutions of the differential equations, plot the numerical results, and interpret the dynamic behavior of the biosystems under a variety of conditions.
- Appreciate the accuracy and stability of the models and the numerical solutions obtained from these models.

7.1.1 Pharmacokinetics: The dynamics of drug absorption

Pharmacokinetics is the study of the processes that affect drug distribution and the rate of drug concentrations within the body (Fournier, 1999). Drugs can enter the body through the gastrointestinal tract, referred to as the enteral route, or through a variety of other pathways that include intravenous injection, inhalation, subcutaneous penetration, etc. These are referred to as parenteral routes. The drug distribution throughout the body is affected by several factors, such as blood perfusion rate, capillary permeability, drug biological affinity, the metabolism of the drug, and renal excretion. The drug is eliminated from the body by enzymatic reactions in the liver and by excretion into the urine stream via the kidneys. A simplified model for drug absorption and elimination is shown in Fig. 7.1. This model treats all body fluids as a single-compartment unit. A mathematical simulation of this model results in a set of linear ordinary differential equations. Methods for the solution of such a set are developed in Sec. 7.4 of this chapter, and are demonstrated in Example 7.2.

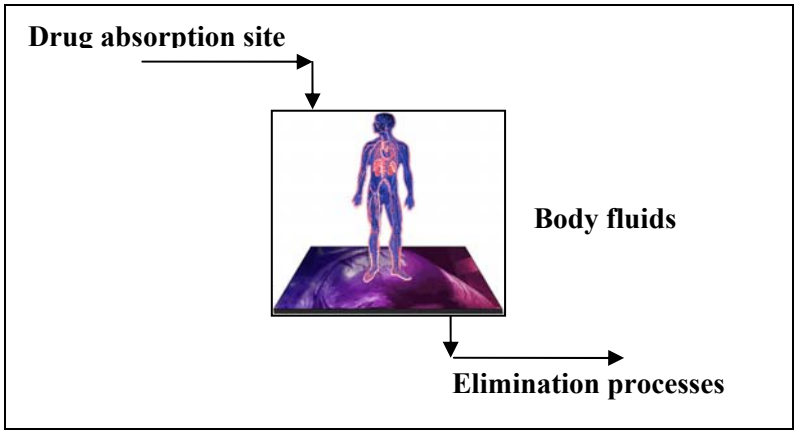


Figure 7.1 Simplified drug absorption model.

7.1.2 Tissue engineering: Stem cell differentiation, cell migration, adhesion

Cell differentiation is a critical dynamic process that underlies the progressive specialization of the various embryonic and progenitor cells to multifunctional tissues in the body. For example, embryonic stem cells in a growing fetus replicate and differentiate to develop into specialized types of cells, such as bone cells, skin cells, liver cells, muscle cells, etc. The differentiation process involves a series of changes in cell phenotype and morphology that typically become more pronounced and easier to observe directly at the later stages of the process (Palsson and Bhatia, 2004). This process begins with the stem cells commitment to differentiation, followed by a coordinated series of gene-expression events, causing the cell to differentiate to a new state. A series of such progressive states leads to fully mature specialized cells. These mature cells perform their intended function in the body and eventually die, or undergo change to another type of cell through a process called *transdifferentiation*. The progressive series of events that converts a stem cell to a fully mature specialized cell may be modeled as a multi-compartment model. The unsteady state balances on these compartments result in a set of simultaneous ordinary differential equations. The solution of such a set of equations is demonstrated in Example 7.6 that presents and discusses stem cell differentiation.

An important aspect of tissue engineering is the proper design and manufacture of porous matrices that imitate the properties of the epidermis and may be used as prosthetic scaffolding to promote dermal regeneration, thus enhancing the healing process of wounded or burned skin. A cellular dynamic process, relevant to wound repair and tissue regeneration, is cell migration (Lauffenburger and Horowitz, 1996). Cell migration is necessary for cells to repopulate a healing wound and an implanted scaffold for tissue regeneration, and during embryogenesis for cell sorting and organ development. Cell migration is also relevant to cancer and tumor metastasis.

Cellular migration is a coordinated process that results from the interaction of specific cell surface receptors with ligands, which are typically biomolecules of an extracellular matrix (Fig. 7.2). Quantitative descriptions of the cell migration process involve establishing relationships between the cell motility response (e.g, cell speed, cell directional persistence, population cell motility) and the various attributes of the ligands. A number of ligand properties, such as ligand surface concentration, degree of receptor occupancy, and ligand affinity, affect the activation of cell motility. An interesting mode of complex cell migration has been quantitatively analyzed by Moghe and coworkers (Tjia and Moghe, 2002a, 2002b). This migration involves cellular internalization (*endocytosis* or *phagocytosis*, depending on the nature of ligand carriers) of the ligands after receptor-ligand binding. The dynamics of cell-ligand interactions have been modeled from a kinetic-mechanistic point of view (Tjia and Moghe, 2002c) using diffusion-reaction descriptions and equations similar to

those in the traditional Michaelis-Menten kinetics. A model of cell migration is presented and solved in Example 7.7.

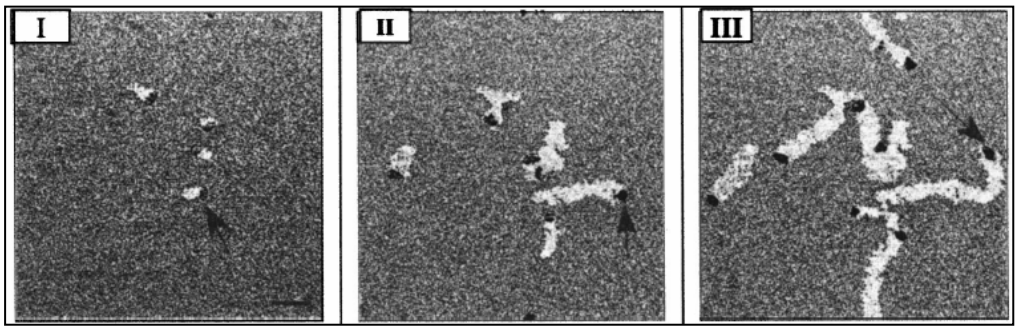


Figure 7.2 The migration of keratinocytes is enhanced by the presence of ligand-bound microcarriers (from Tjia and Moghe, 2002c) .

7.1.3 Glycolysis pathways of living cells

Living cells break down glucose to produce carbon dioxide and water in a complex process called *glycolysis* that involves several enzyme catalyzed reactions. This process generates chemical energy, which is in turn used in the biological synthesis of other compounds, such as proteins. The energy produced in glycolysis is stored by the cell in the form of adenosine triphosphate (ATP). The net effect of this pathway is:

$$C_6H_{12}O_6 + 6O_2 \longrightarrow 6CO_2 + 6H_2O + \text{energy}$$

Many of the chemical reactions in the glycolysis pathway are catalyzed by enzymes, such as the reaction shown here:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} [ES] \xrightarrow{k_2} P + E$$

An enzyme, E, catalyzes the conversion of a substrate, S, to form a product, P, via the formation of an intermediate complex, $[ES]$. The steady state analysis of such reactions results in algebraic equations whose solution may be obtained by the methods discussed in Chapter 5 of this book. On the other hand, the dynamic behavior of enzymatic reactions is modeled by ordinary differential equations. Methods of solution for sets of ordinary differential equations are developed in Sec. 7.5 of this chapter, and are applied to obtain the solution of an enzyme catalysis problem in Example 7.3.

7.1.4 Transport of molecules in biological membranes

The transport of molecules across biological membranes is vital to the operation and survival of living cells. The supply of nutrients to the cell, for growth and reproduction, and the transfer of waste products from cell to the extracellular medium, is a complex process that is facilitated by many mechanisms (Fig. 7.3). There is passive transport of molecules due to the combined effects of concentration gradients and electrical potential differences that exist across the cell membrane. Neutral molecules diffuse from regions of high concentration to regions of low concentration. In addition, charged molecules move along a voltage gradient that normally exists across a cell membrane, such as in neural cells and axons. Carrier-mediated transport and active transport are additional mechanisms that facilitate the movement of molecules across cell boundaries. The transport mechanism of molecules may be model using ordinary and partial differential equations. In this chapter we will discuss dynamic transport systems of one independent variable that may be modeled by ordinary differential equations. In Example 7.5, we solve the Hodgkin-Huxley model that simulates the dynamics of membrane and nerve cell potentials. In Chapter 8 we will examine transport systems of two or more independent variables that result in partial differential equations.

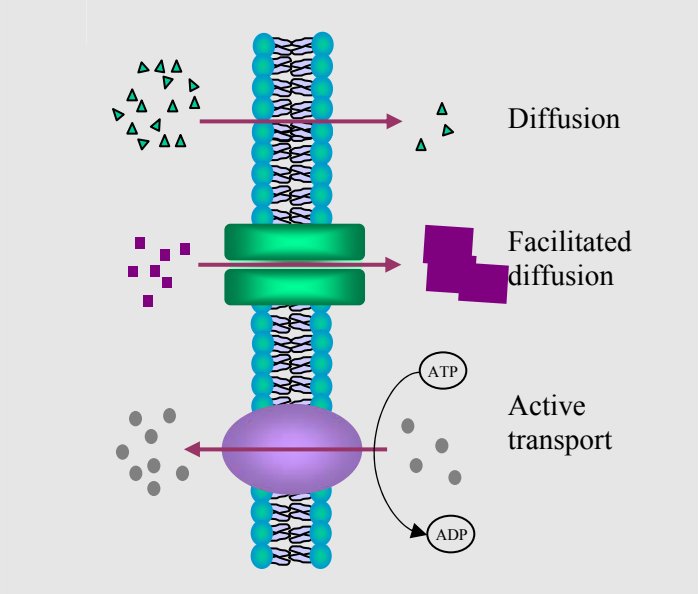


Figure 7.3 Diffusion across biological membranes

7.2 Classification of Ordinary Differential Equations

Ordinary differential equations are classified according to their *order*, *linearity*, *homogeneity*, and *boundary conditions*. The order of a differential equation is the order of the highest derivative present in that equation. Ordinary differential equations may be categorized as *linear* and *nonlinear*. A differential equation is nonlinear if it contains products of the dependent variable, or its derivatives, or of both. In this chapter, as much as possible, we will use the symbol y to represent the dependent variable, and the symbol t to designate the independent variable. The student should remember that either t, or x, is customarily used to represent the independent variable in ordinary differential equations.

The general form of a linear ordinary differential equation of order n may be written as

$$b_n(t)\frac{d^n y}{dt^n} + b_{n-1}(t)\frac{d^{n-1} y}{dt^{n-1}} + \ldots + b_1(t)\frac{dy}{dt} + b_0(t)\,y = R(t) \tag{7.1}$$

If $R(t) = 0$, the equation is called *homogeneous*. If $R(t) \neq 0$, the equation is *nonhomogeneous*. The coefficients $\{b_i \mid i = n, \ldots, 1\}$ are called *variable coefficients* when they are functions of x, and *constant coefficients* when they are scalars. A differential equation is *autonomous* if the independent variable does not appear explicitly in that equation. For example, if Eq. (7.1) is homogeneous with constant coefficients, it is also autonomous. Examples of first, second, and third order differential equations are given below:

First order, linear, homogeneous: $$\frac{dy}{dt} + y = 0 \tag{7.2}$$

First order, linear, nonhomogeneous: $$\frac{dy}{dt} + y = kt \tag{7.3}$$

First order, nonlinear, nonhomogeneous: $$\frac{dy}{dt} + y^2 = kt \tag{7.4}$$

Second order, linear, nonhomogeneous: $$\frac{d^2y}{dt^2}+\frac{dy}{dt}+y=e^t \tag{7.5}$$

Second order, nonlinear, nonhomogeneous: $$y\frac{d^2y}{dt^2}+\frac{dy}{dt}+y=\cos(t) \tag{7.6}$$

Third order, linear, homogeneous: $$\frac{d^3y}{dt^3}+a\frac{d^2y}{dt^2}+b\frac{dy}{dt}+y=0 \tag{7.7}$$

Third order, nonlinear, nonhomogeneous: $$\frac{d^3y}{dt^3}+a\left(\frac{d^2y}{dt^2}\right)^2+\frac{dy}{dt}+y=\sin(t) \tag{7.8}$$

Eqs. (7.4), (7.6), and (7.8) are nonlinear because they contain the terms y^2, $y(d^2y/dt^2)$ and $(d^2y/dt^2)^2$, respectively, whereas Eqs. (7.2), (7.3), (7.5), and (7.7) are linear.

To obtain a unique solution of an nth-order differential equation, or of a set of n simultaneous first-order differential equations, it is necessary to specify n values of the dependent variables (or their derivatives) at specific values of the independent variable.

Ordinary differential equations may be classified as *initial-value* problems or *boundary-value* problems. In initial-value problems, the values of the dependent variables and/or their derivatives are *all* known at the initial value of the independent variable. A problem whose dependent variables, and/or their derivatives, are all known at the final value of the independent variable (rather than the initial value) is identical to the initial-value problem, because only the direction of integration must be reversed. Therefore, the term initial-value problem refers to either case. In boundary-value problems, the dependent variables and/or their derivatives are known at more than one point of the independent variable. If some of the dependent variables (or their derivatives) are specified at the initial value of the independent variable, and the remaining variables (or their derivatives) are specified at the final value of the independent variable, then this is a *two-point boundary-value* problem.

The methods of solution of initial-value problems are developed in Sec. 7.5. The methods for solution of boundary-value problems will not be covered in this book. The interested student is referred to Constantinides and Mostoufi (1999).

7.3 Transformation to Canonical Form

Numerical integration of ordinary differential equations is most conveniently performed when the system consists of a set of n simultaneous first-order ordinary differential equations of the form:

$$\begin{aligned} \frac{dy_1}{dt} &= f_1\left(t, y_1, y_2, \ldots, y_n\right) & y_1\left(t_0\right) &= y_{1,0} \\ \frac{dy_2}{dt} &= f_2\left(t, y_1, y_2, \ldots, y_n\right) & y_2\left(t_0\right) &= y_{2,0} \\ &\vdots & & \\ \frac{dy_n}{dt} &= f_n\left(t, y_1, y_2, \ldots, y_n\right) & y_n\left(t_0\right) &= y_{n,0} \end{aligned} \tag{7.9}$$

This is called the *canonical* form of the equations. When the initial conditions are given at a common point, t_0, then the set of equations (7.40) has solutions of the form

$$\begin{aligned} y_1 &= F_1\left(t\right) \\ y_2 &= F_2\left(t\right) \\ &\vdots \\ y_n &= F_n\left(t\right) \end{aligned} \tag{7.10}$$

The above problem can be condensed into matrix notation, where the system equations are represented by

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}\left(t, \mathbf{y}\right) \tag{7.11}$$

the vector of initial conditions is

$$\mathbf{y}\left(t_0\right) = \mathbf{y_0} \tag{7.12}$$

and the vector of solutions is

$$\mathbf{y} = \mathbf{F}\left(t\right) \tag{7.13}$$

Differential equations of higher order, or systems containing equations of mixed order, can be transformed to the canonical form by a series of substitutions. For example, consider the nth-order differential equation

$$\frac{d^n z}{dt^n} = G\left(z,\ \frac{dz}{dt},\ \frac{d^2 z}{dt^2},\ \ldots\ \frac{d^n z}{dt^n},\ t\right) \tag{7.14}$$

The following transformations

$$\begin{aligned} z &= y_1 \\ \frac{dz}{dt} &= \frac{dy_1}{dt} = y_2 \\ \frac{d^2z}{dt^2} &= \frac{dy_2}{dt} = y_3 \\ &\vdots \\ \frac{d^{n-1}z}{dt^{n-1}} &= \frac{dy_{n-1}}{dt} = y_n \\ \frac{d^nz}{dt^n} &= \frac{dy_n}{dt} \end{aligned} \tag{7.15}$$

when substituted into the *n*th-order equation (7.45), give the equivalent set of *n* first-order equations of canonical form:

$$\begin{aligned} \frac{dy_1}{dt} &= y_2 \\ \frac{dy_2}{dt} &= y_3 \\ &\vdots \\ \frac{dy_n}{dt} &= G\left(y_1,\ y_2,\ y_3, \ldots, y_n,\ t\right) \end{aligned} \tag{7.16}$$

If the right-hand side of the differential equations is not a function of the independent variable, that is,

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}\left(\mathbf{y}\right) \tag{7.17}$$

then the set is *autonomous*. A *nonautonomous* set may be transformed to an autonomous set by an appropriate substitution (see Example 7.1 (*b*)).

If the functions **f**(**y**) are linear in terms of **y**, then the equations can be written in matrix form:

$$\mathbf{y}' = \mathbf{A}\mathbf{y} \tag{7.18}$$

as in Example 7.1 (*a*) and (*b*). Solutions for linear sets of ordinary differential equations are developed in Sec. 7.4. The methods for solution of nonlinear sets are discussed in Sec. 7.5.

A more restricted form of differential equation is

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t) \tag{7.19}$$

where **f**(t) are functions of the independent variable only. Solution methods for these equations were developed in Chapter 6.

The next example demonstrates the technique for converting higher-order linear and nonlinear differential equations to canonical form.

Example 7.1 Transformation of ordinary differential equations into their canonical form.

Statement of the problem

Apply the transformations defined by Eqs. (7.15) and (7.16) to the following ordinary differential equations:

(a) $$\frac{d^4z}{dt^4} + 5\frac{d^3z}{dt^3} - 2\frac{d^2z}{dt^2} - 6\frac{dz}{dt} + 3z = 0 \qquad \text{(Linear, autonomous)}$$

With initial conditions

$$\text{at } t = 0, \quad \left.\frac{d^3z}{dt^3}\right|_0 = 2, \quad \left.\frac{d^2z}{dt^2}\right|_0 = 1.5, \quad \left.\frac{dz}{dt}\right|_0 = 1, \quad z|_0 = 0.5$$

(b) $$\frac{d^4z}{dt^4} + 5\frac{d^3z}{dt^3} - 2\frac{d^2z}{dt^2} - 6\frac{dz}{dt} + 3z = e^{-t} \qquad \text{(Linear, nonhomogeneous)}$$

With initial conditions

$$\text{at } t = 0, \quad \left.\frac{d^3z}{dt^3}\right|_0 = 2, \quad \left.\frac{d^2z}{dt^2}\right|_0 = 1.5, \quad \left.\frac{dz}{dt}\right|_0 = 1, \quad z|_0 = 0.5$$

(c) $$\frac{d^3z}{dt^3} + z^2\frac{d^2z}{dt^2} - \left(\frac{dz}{dt}\right)^3 - 2z = 0 \qquad \text{(Nonlinear, autonomous)}$$

With boundary conditions

$$\text{at t} = 0, \quad \left.\frac{d^2z}{dt^2}\right|_0 = 1, \quad \left.\frac{dz}{dt}\right|_0 = 2, \quad z|_0 = 3$$

Solution

(a) Apply the transformation according to Eqs. (7.15) to obtain the following four equations:

$$\frac{dy_1}{dt} = y_2 \qquad y_1(0) = 0.5$$

$$\frac{dy_2}{dt} = y_3 \qquad y_2(0) = 1$$

$$\frac{dy_3}{dt} = y_4 \qquad y_3(0) = 1.5$$

$$\frac{dy_4}{dt} = -3y_1 + 6y_2 + 2y_3 - 5y_4 \qquad y_4(0) = 2$$

This is a set of linear ordinary differential equations that can be represented in matrix form by Eq. (7.18), where matrix **A** is given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 6 & 2 & -5 \end{bmatrix}$$

The method for obtaining the solution of sets of linear ordinary differential equations is discussed in Sec. 7.4.

(b) The presence of the term e^{-t} on the right-hand side of this equation makes it nonhomogeneous. The left-hand side is identical to that of Eq. (a), so that the transformations of Eq. (a) are applicable. An additional transformation is needed to replace the e^{-t} term. This transformation is

$$y_5 = e^{-t}$$

$$\frac{dy}{dt} = -e^{-t} = -y_5$$

Make the substitutions into Eq. (b) to obtain the following set of five linear ordinary differential equations:

$$\frac{dy_1}{dt} = y_2 \qquad y_1(0) = 0.5$$

$$\frac{dy_2}{dt} = y_3 \qquad y_2(0) = 1$$

$$\frac{dy_3}{dt} = y_4 \qquad y_3(0) = 1.5$$
$$\frac{dy_4}{dt} = -3y_1 + 6y_2 + 2y_3 - 5y_4 + y_5 \qquad y_4(0) = 2$$
$$\frac{dy_5}{dt} = -y_5 \qquad y_5(0) = 1$$

The above set condenses into the matrix form of Eq. (7.18), with the matrix $\boldsymbol{A}$ given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -3 & 6 & 2 & -5 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

(c) This problem is nonlinear, however, similar transformations may be applied:

$$z = y_1$$
$$\frac{dz}{dt} = \frac{dy_1}{dt} = y_2$$
$$\frac{d^2z}{dt^2} = \frac{dy_2}{dt} = y_3$$
$$\frac{d^3z}{dt^3} = \frac{dy_3}{dt}$$

Make the substitutions into Eq. (c) to obtain the set

$$\frac{dy_1}{dt} = y_2 \qquad y_1(0) = 3$$
$$\frac{dy_2}{dt} = y_3 \qquad y_2(0) = 2$$
$$\frac{dy_3}{dt} = 2y_1 + y_2^3 - y_1^2 y_3 \qquad y_3(0) = 1$$

As expected, this is a set of *nonlinear* differential equations, which cannot be expressed in matrix form. The methods of solution of nonlinear differential equations are developed in Sec. 7.5.

7.4 Linear Ordinary Differential Equations

The analysis of many bioengineering systems yields mathematical models that are sets of *linear* ordinary differential equations with constant coefficients and can be reduced to the form

$$\mathbf{y}' = \mathbf{A}\mathbf{y} \tag{7.18}$$

with given initial conditions

$$\mathbf{y}(0) = \mathbf{y_0} \tag{7.20}$$

Sets of linear ordinary differential equations with constant coefficients have closed-form solutions that can be readily obtained from the eigenvalues and eigenvectors of matrix **A**. In order to develop this solution, let us first consider a single linear differential equation of the type

$$\frac{dy}{dt} = ay \tag{7.21}$$

with the given initial condition

$$y(0) = y_0 \tag{7.22}$$

Eq. (7.21) is essentially the scalar form of the matrix set of Eq. (7.18). The solution of the scalar equation can be obtained by separating the variables and integrating both sides of the equation

$$\begin{aligned} \int_{y_0}^{y} \frac{dy}{y} &= \int_0^t a\,dt \\ \ln\frac{y}{y_0} &= at \\ y &= e^{at} y_0 \end{aligned} \tag{7.23}$$

In an analogous fashion, the matrix set can be integrated to obtain the solution

$$\mathbf{y} = \mathbf{e}^{\mathbf{At}}\mathbf{y_0} \tag{7.24}$$

In this case, **y** and $\mathbf{y_0}$ are *vectors* of the dependent variables and the initial conditions, respectively. The term $\mathbf{e}^{\mathbf{At}}$ is the matrix exponential function, which can be obtained from Eq. (7.25):

$$\mathbf{e}^{\mathbf{At}} = \mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2!} + \frac{\mathbf{A}^3t^3}{3!} + \frac{\mathbf{A}^4t^4}{4!} + \ldots \tag{7.25}$$

It can be demonstrated that Eq. (7.25) is a solution of Eq. (7.18) by differentiating it:

$$\begin{aligned}
\frac{d\mathbf{y}}{dt} &= \frac{d}{dt}\left(\mathbf{e}^{\mathbf{A}t}\right)\mathbf{y_0} \\
&= \frac{d}{dt}\left(\mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2!} + \frac{\mathbf{A}^3t^3}{3!} + \frac{\mathbf{A}^4t^4}{4!} + \ldots\right)\mathbf{y_0} \\
&= \left(\mathbf{A} + \mathbf{A}^2t + \frac{\mathbf{A}^3t^2}{2!} + \frac{\mathbf{A}^4t^3}{3!} + \ldots\right)\mathbf{y_0} \\
&= \mathbf{A}\left(\mathbf{I} + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2!} + \frac{\mathbf{A}^3t^3}{3!} + \ldots\right)\mathbf{y_0} \\
&= \mathbf{A}\left(\mathbf{e}^{\mathbf{A}t}\right)\mathbf{y_0} \\
&= \mathbf{A}y
\end{aligned} \tag{7.26}$$

The solution of the set of linear ordinary differential equations is very cumbersome to evaluate in the form of Eq. (7.25) because it requires the evaluation of the infinite series of the exponential term $\mathbf{e}^{\mathbf{At}}$. However, this solution can be modified by further algebraic manipulation to express it in terms of the eigenvalues and eigenvectors of the matrix **A**. In Chapter 4, we showed that a nonsingular matrix **A** of order n has n eigenvectors and n nonzero eigenvalues, whose definitions are given by

$$\begin{aligned}
\mathbf{A}\mathbf{x}_1 &= \lambda_1\mathbf{x}_1 \\
\mathbf{A}\mathbf{x}_2 &= \lambda_2\mathbf{x}_2 \\
&\vdots \\
\mathbf{A}\mathbf{x}_n &= \lambda_n\mathbf{x}_n
\end{aligned} \tag{7.27}$$

All the above eigenvectors and eigenvalues can be represented in a more compact form as follows:

$$\mathbf{AX} = \mathbf{X\Lambda} \tag{7.28}$$

where the columns of matrix **X** are the individual eigenvectors:

$$\mathbf{X} = \left[\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \ldots, \mathbf{x_n}\right] \tag{7.29}$$

and $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues of **A** on its diagonal:

$$\Lambda = \begin{bmatrix} \lambda_1 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \lambda_2 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \lambda_3 & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \lambda_n \end{bmatrix} \tag{7.30}$$

Through a series of matrix operations, Eqs. (7.25) and (7.28) can be combined to express the matrix exponential as follows:

$$\mathbf{e}^{\mathbf{At}} = \mathbf{X}\mathbf{e}^{\mathbf{\Lambda t}}\mathbf{X}^{-1} \tag{7.31}$$

For a complete derivation of this equation see Constantinides and Mostoufi (1999).

The solution of the linear differential equations can now be expressed in terms of eigenvalues and eigenvectors by combining Eqs. (7.24) and (7.31):

$$\mathbf{y} = \left[\mathbf{X}\mathbf{e}^{\mathbf{\Lambda t}}\mathbf{X}^{-1}\right]\mathbf{y}_0 \tag{7.32}$$

This method will always work provided that we can find n linearly independent eigenvectors of the ($n \times n$) matrix **A**. This is equivalent to saying that matrix **X** must be nonsingular so that its inverse may be calculated. The eigenvalues and eigenvectors of matrix **A** can be calculated using the techniques developed in Chapter 4, or simply by applying the built-in MATLAB functions described below.

MATLAB functions: MATLAB has several functions that may be used to calculate matrix exponentials and eigenvalues/eigenvectors:

expm(A): Calculates the matrix exponential of **A** using a scaling and squaring algorithm with a Pade approximation (Burden *et al.*, 1981).

expm2(A): Calculates the matrix exponential of **A** via Taylor series. As a practical numerical method, this is often slow and inaccurate.

expm3(A): Calculates the matrix exponential of **A** via eigenvalues and eigenvectors. The accuracy of this method is determined by the condition of the eigenvector matrix.

eig(A): Calculates the eigenvalues of matrix **A**.

[X, LAMBDA] = *eig*(A): Produces a diagonal matrix LAMBDA of eigenvalues, as in Eq. (7.30), and a full matrix **X** whose columns are the corresponding eigenvectors, as in Eq. (7.29), so that Eq. (7.28) is satisfied, i.e., A*X = X*LAMDA

Eq. (7.32) may be evaluated using some of the above MATLAB functions as follows:

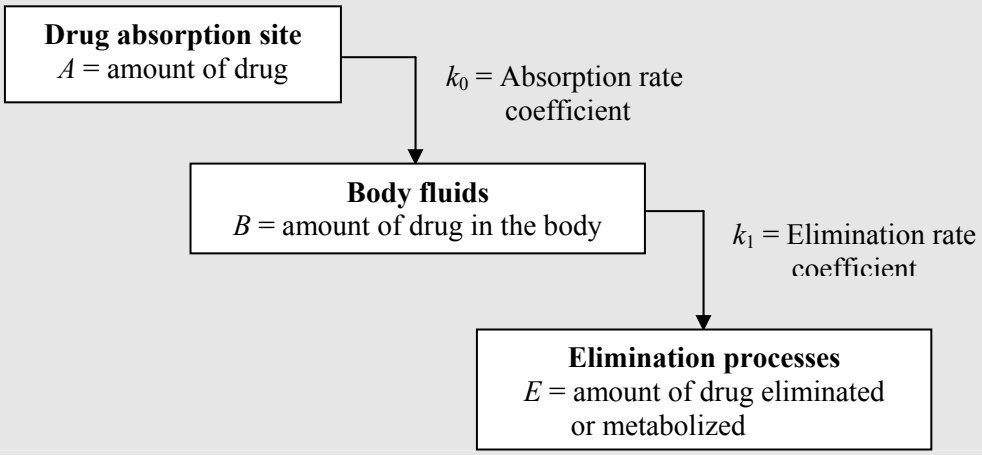
```
syms t
A = [define the elements of matrix A]
y0 = [define the elements of vector y0]
[X, LAMBDA] = eig(A)
y = X*expm(LAMBDA*t)*X^-1*y0
```

The use of these functions is demonstrated in Example 7.2.

Example 7.2 The dynamics of drug absorption.

Statement of the problem

The drug absorption mechanism in the body may be modeled, in its simplest form, as a three-step process, shown diagrammatically below:



All body fluids are treated as a single unit. Unsteady state mass balances around each of the three steps yield three linear ordinary differential equations. The equation that describes the rate of change of the amount of drug at the absorption site is

$$\frac{dA}{dt} = -k_0 A, \qquad A(0) = A_0 \tag{7.33}$$

The rate of change of the amount of drug in the body is described by

$$\frac{dB}{dt} = k_0 A - k_1 B, \qquad B(0) = 0 \tag{7.34}$$

and the rate of change of the amount of drug eliminated is measured by

$$\frac{dE}{dt} = k_1 B, \qquad E(0) = 0 \tag{7.35}$$

Equations (7.33), (7.34), and (7.35) constitute a set of simultaneous first order linear ordinary differential equations, whose solution, $A(t)$, $B(t)$, $E(t)$, correspond to the drug concentrations being fed, in the body, and being eliminated, respectively. It has been determined that values of $k_0 = 0.01$ min^{-1} and $k_1 = 0.035$ min^{-1} are reasonable values for this system. Use the analytical and numerical solution of these equations to calculate the time, $t_{\max}$, at which the concentration of drug in the body reaches its maximum value, $B_{\max} = B(t_{\max})$, and plot the profiles for all three concentrations as functions of time.

Solution

(a) The analytical solutions to the differential equations may be obtained with the MATLAB command `dsolve`:

```
>> [A,B,E]=dsolve('DA=-k0*A','DB=k0*A-k1*B','DE=k1*B', 'A(0)=A0',
'B(0)=0', 'E(0)=0');
>> A=simplify(A)
A =
A0*exp(-k0*t)
>> B=simplify(B)
B =
k0*A0*(-exp(-k1*t)+exp(-k0*t))/(-k0+k1)
>> E=simplify(E)
E =
-A0*(exp(-k0*t)*k1-k1+k0-exp(-k1*t)*k0)/(-k0+k1)
```

From this output we conclude that the analytical solutions for A, B, and E are

$$A(t) = A_0 e^{-k_0 t}$$

$$B(t) = \frac{k_0 A_0}{k_1 - k_0}(e^{-k_0 t} - e^{-k_1 t})$$

$$E(t) = \frac{-A_0(k_1 e^{-k_0 t} - k_0 e^{-k_1 t}) + A_0(k_1 - k_0)}{(k_1 - k_0)}$$

The law of conservation of mass predicts that

$$A(t) + B(t) + E(t) = A_0 + B_0 + E_0$$

This is easily verified by the MATLAB command (remember that B_0 and E_0 are equal to zero in this problem):

```
>> simplify(A+B+E)
ans =
A0
```

The value of t_{max} is obtained by taking the derivative of $B(t)$, equating it to zero, and solving for t, using the values k_0=0.01 and k_1=0.035:

```
>> dB = diff(B)
dB =
k0*A0*(k1*exp(-k1*t)-k0*exp(-k0*t))/(-k0+k1)
>> tmax = solve(dB,'t')
tmax =
log(k1/k0)/(-k0+k1)
>> k0=.01;k1=0.035;
>> eval(tmax)
ans =
   50.1105
```

This predicts that the maximum concentration of the drug in the body is reached at approximately 50 minutes after injection.

(b) This problem will now be solved using the eigenvalue-eigenvector method of Eq. (7.32), and the matrix exponential method of Eq. (7.24). The following MATLAB script was written for this purpose. This program is called `example7_2b.m` and is included in the biosystems software that accompanies this book:

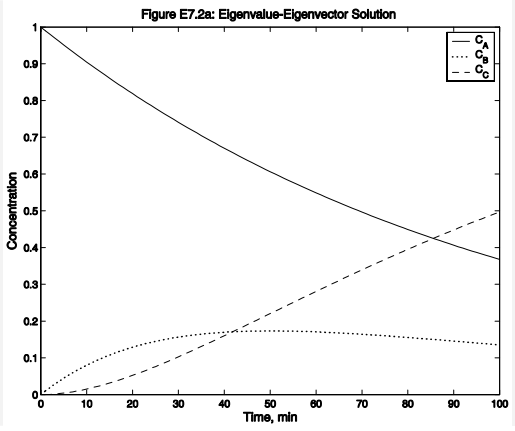
```
% example7_2b.m - Solution of the drug absorption problem,
% both symbolically and numerically, using the eigenvalue-
% eigenvector method and the matrix exponential method.%

clc; clear all;
syms c t
% Constants
k0=0.01; k1=0.035;
disp('Initial concentrations:')
c0=[1; 0; 0]
disp(' '); disp('Matrix of coefficients:')
K=[-k0 0 0; k0 -k1 0; 0 k1 0]
% Eigenvalue-eigenvector method
[X,lambda]=eig(K);
disp(' '), disp('Eigenvectors:'),   X
```

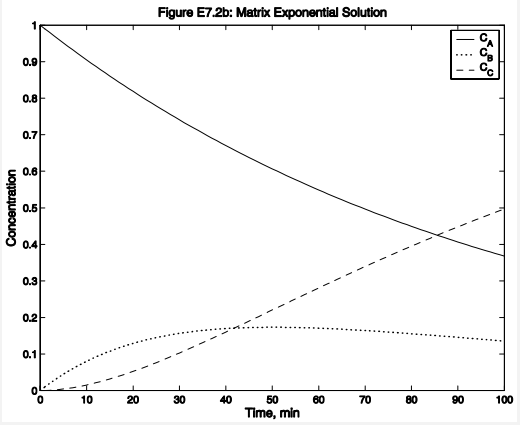
```
disp(' '), disp('Eigenvalues:'),   lambda
disp(' '), disp('Inverse of X:'),   X^-1
disp(' ');
disp('Concentrations using eigenvalue-eigenvector method:')
c=X*expm(lambda*t)*X^-1*c0

% Evaluate concentration profiles
t=[0:100]; c=eval(c);

% Find the maximum concentration and time of drug in the body
[Cmax,tm]=max(c(2,:));
fprintf('\nMaximum concentration in the body = %6.4f at tmax =
%4.2f min.\n',Cmax, tm-1)

% Plot the results
clf; figure(1); h=plot(t,c(1,:), t,c(2,:),':',t,c(3,:),'--');
title('Figure E7.2a: Eigenvalue-Eigenvector Solution')
ylabel('Concentration'); xlabel('Time, min');
legend('C_A','C_B','C_C')

% Matrix exponential method
disp(' '); disp('Concentrations using matrix exponential method:')
syms t
c=expm(K*t)*c0
t=[0:100]; c=eval(c);

% Plot the results
figure(2); h=plot(t,c(1,:), t,c(2,:),':',t,c(3,:),'--');
title('Figure E7.2b: Matrix Exponential Solution')
xlabel('Time, min'); ylabel('Concentration');
legend('C_A','C_B','C_C')
```

Output of results

```
Initial concentrations:
c0 =
     1
     0
     0

Matrix of coefficients:
K =
   -0.0100         0         0
    0.0100   -0.0350         0
         0    0.0350         0

Eigenvectors:
X =
         0         0    0.5661
         0    0.7071    0.2265
```

```
    1.0000   -0.7071   -0.7926

Eigenvalues:
lambda =
         0         0         0
         0   -0.0350         0
         0         0   -0.0100

Inverse of X:
ans =
    1.0000    1.0000    1.0000
   -0.5657    1.4142         0
    1.7664         0         0

Concentrations using eigenvalue-eigenvector method:
c =
[                          exp(-1/100*t)]
[  -2/5*exp(-7/200*t)+2/5*exp(-1/100*t)]
[ 1+2/5*exp(-7/200*t)-7/5*exp(-1/100*t)]
Maximum concentration in the body = 0.1731 at tmax = 50.00 min.

Concentrations using matrix exponential method:
c =
[                          exp(-1/100*t)]
[  -2/5*exp(-7/200*t)+2/5*exp(-1/100*t)]
[ 1+2/5*exp(-7/200*t)-7/5*exp(-1/100*t)]
```





Discussion of results
As expected, the results from the two methods are identical, and they also confirm the results of the analytical method. The values of t_{max} and B_{max} are 50 min and 0.1731, respectively.

7.5 Nonlinear Ordinary Differential Equations

In this section, we develop numerical solutions for a set of ordinary differential equations in their canonical form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}\left(t, \mathbf{y}\right) \tag{7.11}$$

with the vector of initial conditions given by

$$\mathbf{y}\left(t_0\right) = \mathbf{y_0} \tag{7.12}$$

In order to be able to illustrate these methods graphically, we treat $\boldsymbol{y}$ as a single variable rather than as a vector of variables. The formulas developed for the solution of a single differential equation are readily expandable to those for a set of differential equations, which must be solved *simultaneously*. This concept is demonstrated in Sec. 7.5.4.

We begin the development of these methods by first rearranging Eq. (7.11) and integrating both sides between the limits of $t_i \le t \le t_{i+1}$ and $y_i \le y \le y_{i+1}$:

$$\int_{y_i}^{y_{i+1}} dy = \int_{t_i}^{t_{i+1}} f\left(t, y\right) dt \tag{7.36}$$

The left side integrates readily to obtain

$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t, y)\, dt \tag{7.37}$$

One method for integrating Eq. (7.37) is to take the left-hand side of this equation and use finite differences for its approximation. This technique works directly with the tangential trajectories of the dependent variable y rather than with the areas under the function $f(t, y)$. This is the technique applied in Secs. 7.5.1 and 7.5.2.

In Chapter 6, we developed the integration formulas by first replacing the function $f(t)$ with an interpolating polynomial and then evaluating the integral of $f(t)dt$ between the appropriate limits. A similar technique could be applied here to integrate the right-hand side of Eq. (7.37). This approach is followed in Sec. 7.5.3.

MATLAB functions: There are several functions in MATLAB that may be used for the integration of sets of ordinary differential equations of the form of (7.42). These solvers, along with their method of solution, are listed in Table 7.1. Any one of the following statements may be used to call an ODE solver

```
[T, Y] = solver(@name_func, tspan, y0)
[T, Y] = solver(@name_func, tspan, y0, options)
[T, Y] = solver(@name_func, tspan, y0, options, p1, p2,...)
```

where "`solver`" is one of `ode23, ode45, ode113, ode15s, ode23s, ode23t,` or `ode23tb`.

The arguments that are passed to the solver are:

`name_func`: The name of the m-file containing the function that evaluates the right-hand side of the differential equations. Function *name_func*(t, y) must return a column vector corresponding to **f**(t, **y**).

`tspan`: A vector specifying the interval of integration, `[t0,tf]`. To obtain solutions at specific points of `t` (all increasing or all decreasing), use `tspan=[t0,t1,...,tf]`, or to obtain solutions at equally spaced intervals, specify `tspan = [t0:delt:tf]`, where `delt` is the user's choice of spacing between points where output will be given.

`y0`: The vector containing the initial conditions of the differential equations.

`options`: Optional integration argument created using the `odeset` function. See `odeset` for details.

`p1, p2, ...`: Optional parameters that the solver passes to `name_func` and all the functions specified in options.

`[T, Y]`: The solver returns the values of independent and dependent variables in the vectors `T, Y,` respectively. The vector of independent variable is not equally spaced, because the integrating solver controls the step size, unless the user has specified the `tspan`, as described above.

For example:

```
[T,Y] = ode45(@test1_func,[0:10],[1,0],[],0.1, 0.02, 0.1)

function dydt = test1_func(x, y, p1, p2, p3)
dydt = [p1*y(1)-p2*y(2)^2; p3*exp(y(1))];
```

This function should return the value(s) of the derivative(s) as a column vector. The first input to this function has to be the independent variable, *x*, even if it is not explicitly used in the definition of the derivative (autonomous equations). The second input argument to the function is the vector of dependent variables, *y*. The additional parameters, `p1, p2, p3`, are the last three values in the `ode45` call, (…, 0.1, 0.02, 0.1), which get passed on to the `test1_func` function.

An alternate way of using these functions is:

```
[T,Y] = ode45('test2_func',[0:10],[1,0],[],0.1, 0.02, 0.1)
function dydt = test2_func(x, y, flag, p1, p2, p3)
dydt = [p1*y(1)-p2*y(2)^2; p3*exp(y(1))];
```

It should be noted that in this case the third input to *test2_func* has to be an empty variable, *flag*, and the additional parameters are introduced starting with the fourth argument.

Table 7.1 Ordinary differential equation solvers in MATLAB

Solver	Method of solution
ode23	Runge-Kutta lower-order (2^{nd} order, 3stages)
ode45	Runge-Kutta higher-order (4^{th} order, 5stages)
ode113	Adams-Bashforth-Moulton of varying order (1-13)
ode15s	Implicit, multistep of varying order (1-5), for stiff differential equations
ode23s	Modified Rosenbrock of order 2, for stiff differential equations
ode23t	Implementation of the trapezoidal rule using a "free" interpolant, for moderately stiff differential equations

ode23tb	Implementation of an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order two, for stiff differential equations

7.5.1 The Euler and modified Euler methods

One of the earliest techniques developed for the solution of ordinary differential equations is the *Euler method*. This is simply obtained by recognizing that the left side of Eq. (7.37) is the first forward finite difference of y at position i:

$$y_{i+1} - y_i = \Delta y_i \tag{7.38}$$

which, when rearranged, gives a "forward marching" formula for evaluating y:

$$y_{i+1} = y_i + \Delta y_i \tag{7.39}$$

The forward difference term Δy_i is obtained from Eq. (???) applied to y at position i:

$$\Delta y_i = hDy_i + \frac{h^2 D^2 y_i}{2} + \frac{h^3 D^3 y_i}{6} + \ldots \tag{7.40}$$

In the Euler method, the above series is truncated after the first term to obtain

$$\Delta y_i = hDy_i + O\left(h^2\right) \tag{7.41}$$

The combination of Eqs. (7.39) and (7.41) gives the *explicit Euler formula* for integrating differential equations

$$y_{i+1} = y_i + hDy_i + O\left(h^2\right) \tag{7.42}$$

The derivative Dy_i is replaced by its equivalent y'_i or $f(t_i, y_i)$ to give the more commonly used form of the explicit Euler method[1]

$$y_{i+1} = y_i + hf\left(t_i, y_i\right) + O\left(h^2\right) \tag{7.43}$$

The Euler method, Eq. (7.43), simply states that the next value of y is obtained from the previous value by moving a step of width h in the tangential direction of y. This is demonstrated graphically in Fig. 7.4*a*. This Euler formula is rather inaccurate

[1] From here on the term y'_i and $f(t_i, y_i)$ will be used interchangeably. The student should remember that these are equal to each other through the differential equation (7.42).

because it has a truncation error of only $O(h^2)$. If h is large the trajectory of y can quickly deviate from its true value, as demonstrated in Fig. 7.4*b*.

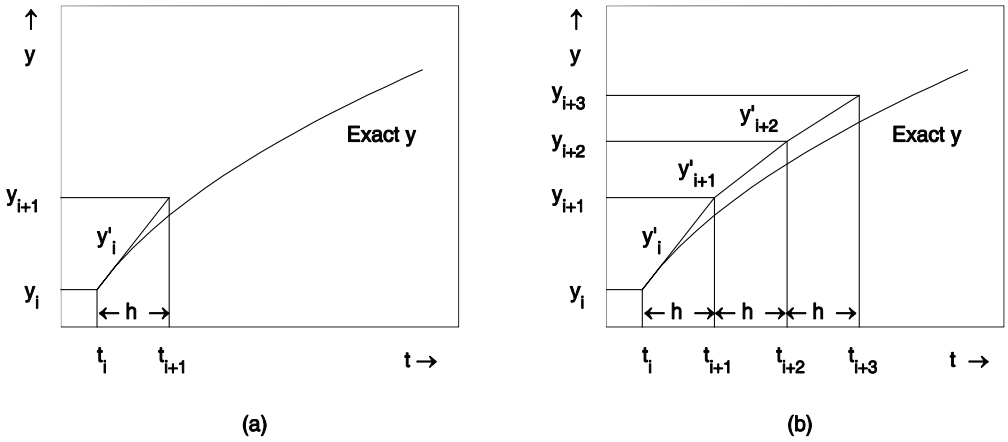


Figure 7.4 The explicit Euler method of integration. (a) Single step. (b) Several steps.

The accuracy of the Euler method can be improved by utilizing a combination of forward and backward differences. Note that the first forward difference of y at i is equal to the first backward difference of y at $(i + 1)$:

$$\Delta y_i = y_{i+1} - y_i = \nabla y_{i+1} \tag{7.44}$$

Therefore, the forward marching formula in terms of backward differences is

$$y_{i+1} = y_i + \nabla y_{i+1} \tag{7.45}$$

The backward difference term ∇y_{i+1} is obtained from Eq. (???) applied to y at position $(i + 1)$:

$$\nabla y_i = hDy_{i+1} - \frac{h^2 D^2 y_{i+1}}{2} + \frac{h^3 D^3 y_{i+1}}{6} - \ldots \tag{7.46}$$

Combining Eqs. (7.45) and (7.46), we obtain:

$$y_{i+1} = y_i + hf\left(t_{i+1}, y_{i+1}\right) + O\left(h^2\right) \tag{7.47}$$

This is called the *implicit Euler formula* (or backward Euler), because it involves the calculation of function f at an unknown value of y_{i+1}. Eq. (7.47) can be viewed as taking a step forward from position i to $(i + 1)$ in a gradient direction that must be evaluated at $(i + 1)$.

Implicit equations cannot be solved individually but must be set up as sets of simultaneous algebraic equations. When these sets are linear, the problem can be solved by the application of the Gauss elimination methods developed in Chapter 4. If the set consists of nonlinear equations, the problem is much more difficult and must be solved using Newton's method for simultaneous nonlinear algebraic equations developed in Chapter 5.

In the case of the Euler methods, the problem can be simplified by first applying the explicit method to *predict* a value y_{i+1}:

$$\left(y_{i+1}\right)_{\text{Predicted}} = y_i + hf\left(t_i, y_i\right) + O\left(h^2\right) \tag{7.48}$$

and then using this predicted value in the implicit method to get a *corrected* value:

$$\left(y_{i+1}\right)_{\text{Corrected}} = y_i + hf\left(t_{i+1}, \left(y_{i+1}\right)_{\text{Predicted}}\right) + O\left(h^2\right) \tag{7.49}$$

This combination of steps is known as the *Euler predictor-corrector* (or *modified Euler*) method. Correction by Eq. (7.49) may be applied more than once until the corrected value converges, that is, the difference between the two consecutive corrected values becomes less than the convergence criterion. However, not much more accuracy is achieved after the second application of the corrector.

The explicit, as well as the implicit, forms of the Euler methods have error of order (h^2). However, when used in combination, as predictor-corrector, their accuracy is enhanced, yielding an error of order (h^3). This conclusion can be reached by adding Eqs. (7.39) and (7.45):

$$y_{i+1} = y_i + \frac{h}{2}\left(\Delta y_i + \nabla y_{i+1}\right) \tag{7.50}$$

and utilizing (7.40) and (7.46) to obtain

$$y_{i+1} = y_i + \frac{h}{2}\left[f\left(t_i, y_i\right) + f\left(t_{i+1}, y_{i+1}\right)\right] + O\left(h^3\right) \tag{7.51}$$

The terms of order (h^2) cancel out because they have opposite sign, thus giving a formula of higher accuracy. Eq. (7.51) is essentially the same as the trapezoidal rule (Eq. (???)), the only difference is in the way the function is evaluated at (t_{i+1}, y_{i+1}).

It has been shown (Finlayson, 1980) that the Euler implicit formula is more stable than the explicit one. The stability of these methods is discussed in Sec. 7.7.

It can be seen by writing Eq. (7.51) in the form

$$y_{i+1} = y_i + \frac{h}{2} f\left(t_i, y_i\right) + \frac{h}{2} f\left(t_{i+1}, y_{i+1}\right) + O\left(h^3\right) \tag{7.52}$$

that this Euler method uses the weighted trajectories of the function y evaluated at two positions that are located one full step of width h apart and weighted equally. In this form, Eq. (7.52) is also known as the Crank-Nicolson method.

Eq. (7.52) can be written in a more general form as

$$y_{i+1} = y_i + w_1 k_1 + w_2 k_2 \tag{7.53}$$

where, in this case:

$$k_1 = hf\left(t_i, y_i\right) \tag{7.54}$$

$$k_2 = hf\left(t_i + c_2 h, y_i + a_{21} k_1\right) \tag{7.55}$$

The choice of the weighting factors, w_1 and w_2, and the positions i and $(i + 1)$ at which to evaluate the trajectories is dictated by the accuracy required of the integration formula, that is, by the number of terms retained in the infinite series expansion.

This concept forms the basis for a whole series of integration formulas, with increasingly higher accuracies, for ordinary differential equations. These are discussed in the following section.

7.5.2 The Runge-Kutta methods

The most widely used methods of integration for ordinary differential equations are the series of methods called Runge-Kutta second, third, fourth, and fifth order, plus a number of other techniques that are variations on the Runge-Kutta theme. These methods are based on the concept of weighted trajectories formulated at the end of Sec. 7.5.1. In a more general fashion, the forward marching integration formula for the differential equation (7.11) is given by the recurrence equation

$$y_{i+1} = y_i + w_1 k_1 + w_2 k_2 + w_3 k_3 + \ldots + w_m k_m \tag{7.56}$$

where each of the trajectories k_i are evaluated by

$$
\begin{aligned}
k_1 &= hf\left(t_i, y_i\right) \\
k_2 &= hf\left(t_i + c_2 h, y_i + a_{21} k_1\right) \\
k_3 &= hf\left(x_i + c_3 h, y_i + a_{31} k_1 + a_{32} k_2\right) \\
&\vdots \\
k_3 &= hf\left(x_i + c_m h, y_i + a_{m1} k_1 + a_{m2} k_2 + \ldots + a_{m,m-1} k_{m-1}\right)
\end{aligned} \tag{7.57}
$$

These equations can be written in a compact form as

$$y_{i+1} = y_i + \sum_{i=1}^{m} w_i k_i \tag{7.58}$$

$$k_j = hf\left(x_i + c_j h, y_i + \sum_{l=1}^{j-1} a_{jl} k_l \right) \tag{7.59}$$

where $c_1 = 0$ and $a_{1j} = 0$. The value of m, which determines the complexity and accuracy of the method, is set when $(m + 1)$ terms are retained in the infinite series expansion of y_{i+1}

$$y_{i+1} = y_i + hy_i' + \frac{h^2 y_i''}{2!} + \frac{h^3 y_i'''}{3!} + \ldots \tag{7.60}$$

or

$$y_{i+1} = y_i + hDy_i + \frac{h^2 D^2 y_i}{2!} + \frac{h^3 D^3 y_i}{3!} + \ldots \tag{7.61}$$

The procedure for deriving the Runge-Kutta methods can be divided into five steps that are demonstrated below in the derivation of the *second-order Runge-Kutta* formulas.

Step 1: Choose the value of m, which fixes the accuracy of the formula to be obtained. For second-order Runge-Kutta, $m = 2$. Truncate the series (7.61) after the $(m + 1)$ term:

$$y_{i+1} = y_i + hDy_i + \frac{h^2 D^2 y_i}{2!} + O\left(h^3\right) \tag{7.62}$$

Step 2: Replace each derivative of y in Eq. (7.62) by its equivalent in f, remembering that f is a function of both t and $y(t)$:

$$Dy_i = f_i \tag{7.63}$$

$$D^2 y_i = \frac{df}{dt} = \left(\frac{\partial f}{\partial t}\frac{dt}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \right)_i = \left(f_t + f f_y \right)_i \tag{7.64}$$

Combine Eqs. (7.62) to (7.64) and regroup the terms:

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2} f_{t_i} + \frac{h^2}{2} f_i f_{y_i} + O\left(h^3\right) \tag{7.65}$$

Step 3: Write Eq. (7.58) with *m* terms in the summation:

$$y_{i+1} = y_i + w_1 k_1 + w_2 k_2 \tag{7.66}$$

where

$$k_1 = hf\left(t_i, y_i\right) \tag{7.67}$$

$$k_2 = hf\left(t_i + c_2 h, y_i + a_{21} k_1\right) \tag{7.68}$$

Step 4: Expand the *f* function in Taylor series:

$$f\left(t_i + c_2 h, y_i + a_{21} k_1\right) = f_i + c_2 h f_{t_i} + a_{21} h f_{y_i} f_i + O\left(h^2\right) \tag{7.69}$$

Combine Eqs. (7.66) to (7.69) and regroup the terms:

$$y_{i+1} = y_i + \left(w_1 + w_2\right) hf_i + \left(w_2 c_2\right) h^2 f_{t_i} + \left(w_2 a_{21}\right) h^2 f_i f_{y_i} + O\left(h^3\right) \tag{7.70}$$

Step 5: In order for Eqs. (7.65) and (7.70) to be identical, the coefficients of the corresponding terms must be equal to one another. This results in a set of simultaneous nonlinear algebraic equations in the unknown constants w_j, c_j, and a_{jl}. For the second-order Runge-Kutta method, there are three equations and four unknowns:

$$\begin{aligned} w_1 + w_2 &= 1 \\ w_2 c_2 &= \frac{1}{2} \\ w_2 a_{21} &= \frac{1}{2} \end{aligned} \tag{7.71}$$

It turns out that there are always more unknowns than equations. The degree of freedom allows us to choose some of the parameters. For second-order Runge-Kutta, there is one degree of freedom. For third- and fourth-order Runge-Kutta, there are two degrees of freedom. For fifth-order Runge-Kutta, there are at least five degrees

of freedom. This freedom of choice of parameters gives rise to a very large number of different forms of the Runge-Kutta formulas. It is usually desirable to first choose the values of the c_j constants, thus fixing the positions along the independent variable, where the functions

$$f\left(t_i + c_j h, y_i + \sum_{l=1}^{j-1} a_{jl} k_l\right) \tag{7.72}$$

are to be evaluated. An important consideration in choosing the free parameters is to minimize the *truncation error* of the calculation.

For the second-order Runge-Kutta method, which we are currently deriving, let us choose $c_2 = 1$. The rest of the parameters are evaluated from Eqs. (7.71):

$$w_1 = w_2 = \frac{1}{2} \qquad a_{21} = 1 \tag{7.73}$$

With this set of parameters, the second-order Runge-Kutta formula is

$$\left.\begin{aligned} y_{i+1} &= y_i + \frac{1}{2}(k_1 + k_2) \\ k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + h, y_i + k_1) \end{aligned}\right\} O(h^3) \tag{7.74}$$

This method is essentially identical to the Crank-Nicolson method (see Eq. (7.52)). Higher-order Runge-Kutta formulas are derived in an analogous manner. Several of these are listed in Table 7.2. The fourth-order Runge-Kutta, which has an

Table 7.2 Summary of the Runge-Kutta integration formulas

Second order Runge-Kutta method (same as Crank-Nicolson method)

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{2}(k_1 + k_2) + O(h^3) \\ k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + h, y_i + k_1) \end{aligned} \tag{7.74}$$

Third order Runge-Kutta method

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 4k_2 + k_3\right) + O\left(h^4\right)$$

$$k_1 = hf\left(t_i, y_i\right)$$

$$k_2 = hf\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \qquad (7.75)$$

$$k_3 = hf\left(t_i + h, y_i + 2k_2 - k_1\right)$$

Fourth order Runge-Kutta method

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) + O\left(h^5\right)$$

$$k_1 = hf\left(t_i, y_i\right)$$

$$k_2 = hf\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \qquad (7.76)$$

$$k_3 = hf\left(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = hf\left(t_i + h, y_i + k_3\right)$$

Table 7.2 Summary of the Runge-Kutta integration formulas (continued)

Fifth order Runge-Kutta method

$$
\begin{aligned}
y_{i+1} &= y_i + \frac{1}{90}\left(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6\right) + O\left(h^6\right) \\
k_1 &= hf\left(t_i, y_i\right) \\
k_2 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \\
k_3 &= hf\left(t_i + \frac{h}{4}, y_i + \frac{3k_1}{16} + \frac{k_2}{16}\right) \\
k_4 &= hf\left(t_i + \frac{h}{2}, y_i + \frac{k_3}{2}\right) \\
k_5 &= hf\left(t_i + \frac{3h}{4}, y_i - \frac{3k_2}{16} + \frac{6k_3}{16} + \frac{9k_4}{16}\right) \\
k_6 &= hf\left(t_i + h, y_i + \frac{k_1}{7} + \frac{4k_2}{7} + \frac{6k_3}{7} - \frac{12k_4}{7} + \frac{8k_5}{7}\right)
\end{aligned}
\tag{7.77}
$$

Runge-Kutta-Felfberg method

$$
\begin{aligned}
y_{i+1} &= y_i + \left(\frac{25}{256}k_1 + \frac{1048}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right) + O\left(h^5\right) \\
k_1 &= hf\left(t_i, y_i\right) \\
k_2 &= hf\left(t_i + \frac{h}{4}, y_i + \frac{k_1}{4}\right) \\
k_3 &= hf\left(t_i + \frac{3}{8}h, y_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \\
k_4 &= hf\left(t_i + \frac{12}{13}h, y_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\
k_5 &= hf\left(t_i + h, y_i + \frac{439}{216}k_1 - 8k_2 + \frac{3860}{513}k_3 - \frac{845}{4104}k_4\right) \\
k_6 &= hf\left(t_i + \frac{h}{2}, y_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right) \\
T_E &= \frac{1}{360}k_1 - \frac{128}{4275}k_2 - \frac{2197}{75240}k_4 + \frac{1}{50}k_5 + \frac{2}{55}k_6
\end{aligned}
\tag{7.78}
$$

error of $O(h^5)$, is probably the most widely used numerical integration method for ordinary differential equations. Implicit Runge-Kutta methods, that offer wider regions of stability than the explicit methods, have been developed and are

thoroughly discussed by Hairer (Hairer *et al.*, 1980), (Hairer and Wanner, 1991, 1991). These methods, such as Radau5 that uses an implicit Runge-Kutta method of order 5 with step size control, are recommended for the solution of stiff differential equations. Discussion of these methods is outside the scope of this book. The interested user may read the aforementioned references for more details.

7.5.3 The Adams and Adams-Moulton methods

The Runge-Kutta family of integration techniques, developed above, are called *single-step* methods. The value of y_{i+1} is obtained from y_i and the trajectories of y within the single step from (t_i, y_i) to (t_{i+1}, y_{i+1}). This procedure marches forward, taking single step of width h, over the entire interval of integration. These methods are very suitable for solving initial-value problems because they are *self-starting* from a given initial point of integration.

Other categories of integration techniques, called *multiple-step* methods, have been developed. These compute the value of y_{i+1} utilizing several previously unknown, or calculated, values of y (y_i, y_{i-1}, y_{i-2}, etc.) as the base points. For this reason, the multiple-step methods are *non-self-starting*. For the solution of initial-value problems, where only y_0 is known, the multiple-step methods must be "primed" by first utilizing a self-starting procedure to obtain the requisite number of base points. There are several multiple-step methods. Two of these, the Adams and Adams-Moulton methods, are covered in this section.

Once again, let us start by evaluating y_{i+1} by integrating the derivative function over the interval $[t_i, t_{i+1}]$

$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t, y)\, dt \tag{7.37}$$

In order to evaluate the right-hand side of Eq. (7.37), $f(t, y)$ may be approximated by an nth-degree polynomial. In the Adams method, a quadratic polynomial is passed through the three past points, that is, (t_{i-2}, y_{i-2}), (t_{i-1}, y_{i-1}), and (t_i, y_i), and is used to extrapolate the value of $f(t_{i+1}, y_{i+1})$. If we choose a uniform step size, a second-degree backward Gregory-Newton interpolating polynomial may be applied to this problem, and Eq. (7.37) becomes

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} \left[f_i - \frac{(t - t_i)}{h} \nabla f_i + \frac{(t - t_i)(t - t_{i+1})}{2! h^2} \nabla^2 f_i \right] dt + \int_{t_i}^{t_{i+1}} R_n(t)\, dt \tag{7.79}$$

where $f_i = f(t_i, y_i)$, and it may be considered a function of t only. Noting that $(t_{i+1} - t_i) = h$, Eq. (7.79) reduces to

$$y_{i+1} = y_i + h\left(f_i + \frac{1}{2}\nabla f_i + \frac{5}{12}\nabla^2 f_i \right) + O\left(h^4\right) \tag{7.80}$$

This equation would be easier to use by expanding the backward differences in terms of the function values given in Table ???. Replacing the backward differences, followed by further rearrangements, results in the following formula known as the Adams method for solution of the ordinary differential equations:

$$y_{i+1} = y_i + \frac{h}{12}\left[23 f\left(t_i, y_i\right) - 16 f\left(t_{i-1}, y_{i-1}\right) + 5 f\left(t_{i-2}, y_{i-2}\right) \right] + O\left(h^4\right) \tag{7.81}$$

Eq. (7.81) shows that prior to evaluating y_{i+1}, the values of the function at three points before that have to be known. Because in an initial-value problem only the value of the function at the start of the solution interval is known, two additional succeeding values should be calculated by a single-step method, such as Runge-Kutta. Solution of the ordinary differential equation from the fourth point may then be continued with Eq. (7.81).

In order to derive the Adams-Moulton technique, we repeat the same procedure by applying a third-degree Gregory-Newton interpolating polynomial (using four past points) instead of a second-degree polynomial to approximate $f(t, y)$ in Eq. (7.37). This procedure results in the prediction of y_{i+1}

$$\left(y_{i+1}\right)_{\mathrm{Pr}} = y_i + \frac{h}{24}\left[55 f\left(t_i, y_i\right) - 59 f\left(t_{i-1}, y_{i-1}\right) + 37 f\left(t_{i-2}, y_{i-2}\right) - 9 f\left(t_{i-3}, y_{i-3}\right) \right] + O\left(h^5\right) \tag{7.82}$$

In the Adams-Moulton method we do not stop here, but we further correct y_{i+1} before moving to the next step. The value of y_{i+1} calculated from Eq. (7.82) is a good approximation of the dependent variable at position $(i + 1)$; therefore, almost the correct value of $f(t_{i+1}, y_{i+1})$ may be evaluated from $f(t_{i+1}, (y_{i+1})_{Pr})$ at this stage. We now interpolate the function $f(t, y)$, using a cubic Gregory-Newton backward interpolating polynomial over the range from t_{i-2} to t_{i+1} and calculate the corrected value of y_{i+1} by the integral of Eq. (7.37):

$$\left(y_{i+1}\right)_{\mathrm{Cor}} = y_i + \frac{h}{24}\left[9 f\left(t_{i+1}, \left(y_{i+1}\right)_{\mathrm{Pr}}\right) + 19 f\left(t_i, y_i\right) - 5 f\left(t_{i-1}, y_{i-1}\right) + f\left(t_{i-2}, y_{i-2}\right) \right] + O\left(h^5\right) \tag{7.83}$$

Eqs. (7.82) and (7.83) should be used as predictor and corrector, respectively. Correction by Eq. (7.83) may be applied more than once until the corrected value converges; that is, the difference between the two consecutive corrected values becomes less than the convergence criterion. However, two applications of the

corrector is probably optimum in terms of computer time and the accuracy gained. Once again, solution of the ordinary differential equation by this technique may start from the fifth point; therefore, some other technique should be applied at the beginning of the solution to evaluate y_1 to y_3.

7.5.4 Simultaneous differential equations

It was mentioned at the beginning of Sec. 7.5 that the methods of solution of a single differential equation are readily adaptable for solving sets of simultaneous differential equations. To illustrate this, we use the set of n simultaneous ordinary differential equations in their canonical form:

$$\begin{aligned}
\frac{dy_1}{dt} &= f_1\left(t, y_1, y_2, \ldots, y_n\right) \\
\frac{dy_2}{dt} &= f_2\left(t, y_1, y_2, \ldots, y_n\right) \\
&\vdots \\
\frac{dy_n}{dt} &= f_n\left(t, y_1, y_2, \ldots, y_n\right)
\end{aligned} \tag{7.84}$$

and expand, for example, the fourth-order Runge-Kutta formulas to

$$\begin{aligned}
y_{i+1,j} &= y_i + \frac{1}{6}\left(k_{1j} + 2k_{2j} + 2k_{3j} + k_{4j}\right) + O\left(h^5\right) && j = 1, 2, \ldots, n \\
k_{1j} &= hf_j\left(t_i, y_{i1}, y_{i2}, \ldots, y_{in}\right) && j = 1, 2, \ldots, n \\
k_{2j} &= hf_j\left(t_i + \frac{h}{2}, y_{i1} + \frac{k_{11}}{2}, y_{i2} + \frac{k_{12}}{2}, \ldots, y_{in} + \frac{k_{1n}}{2}\right) && j = 1, 2, \ldots, n \\
k_{3j} &= hf_j\left(t_i + \frac{h}{2}, y_{i1} + \frac{k_{21}}{2}, y_{i2} + \frac{k_{22}}{2}, \ldots, y_{in} + \frac{k_{2n}}{2}\right) && j = 1, 2, \ldots, n \\
k_{4j} &= hf_j\left(t_i + h, y_{i1} + k_{31}, y_{i2} + k_{32}, \ldots, y_{in} + k_{3n}\right) && j = 1, 2, \ldots, n
\end{aligned} \tag{7.85}$$

This method is programmable using nested loops. In MATLAB, the values of k and y_i can be put in vectors, thus easily evaluating Eq. (7.85) in matrix form.

7.6 Steady State Solutions and Stability Analysis

Before we attempt to obtain the numerical solution of a set of differential equations, it is strongly recommended that we examine the steady state solution of the problem. The steady state is reached when variations with respect to time become zero. To accomplish this mathematically, we force the time-derivatives to become zero and solve the resulting algebraic equations. It is likely that the set of equations will have multiple steady states, including the trivial case, where all variables are zero. We demonstrate these concepts by analyzing a set of two simultaneous nonlinear ordinary differential equations of the form

$$\begin{aligned} \frac{dN_1}{dt} &= f_1\left(N_1, N_2\right) \\ \frac{dN_2}{dt} &= f_2\left(N_1, N_2\right) \end{aligned} \tag{7.86}$$

At steady state the derivatives are set to zero to obtain

$$f_1\left(N_1^*, N_2^*\right) = 0 \qquad f_2\left(N_1^*, N_2^*\right) = 0 \tag{7.87}$$

where N_1^* and N_2^* are the steady state values of the dependent variables. We also define the small deviations (perturbations), $\bar{N}_1$ and $\bar{N}_2$, away from the steady state, so that

$$N_1 = N_1^* + \bar{N}_1 \qquad N_2 = N_2^* + \bar{N}_2 \tag{7.88}$$

By direct substitution of Eqs. (7.88) into Eqs. (7.86), we obtain

$$\begin{aligned} \frac{d\left(N_1^* + \bar{N}_1\right)}{dt} &= f_1\left(N_1^* + \bar{N}_1, N_2^* + \bar{N}_2\right) \\ \frac{d\left(N_2^* + \bar{N}_2\right)}{dt} &= f_2\left(N_1^* + \bar{N}_1, N_2^* + \bar{N}_2\right) \end{aligned} \tag{7.89}$$

The left-hand sides are expanded into the corresponding two derivatives, and the right-hand sides into Taylor series:

$$\begin{aligned} \frac{dN_1^*}{dt} + \frac{d\bar{N}_1}{dt} &= f_1\left(N_1^*, N_2^*\right) + \left(\frac{\partial f_1}{\partial N_1}\right)^* \bar{N}_1 + \left(\frac{\partial f_1}{\partial N_2}\right)^* \bar{N}_2 + \text{higher order terms} \\ \frac{dN_2^*}{dt} + \frac{d\bar{N}_2}{dt} &= f_2\left(N_1^*, N_2^*\right) + \left(\frac{\partial f_2}{\partial N_1}\right)^* \bar{N}_1 + \left(\frac{\partial f_2}{\partial N_2}\right)^* \bar{N}_2 + \text{higher order terms} \end{aligned} \tag{7.90}$$

We apply the condition of steady state (time-derivatives and functions at steady state are zero), and assume that the perturbations around the steady state are small. The latter assumption enables us to drop the higher order terms that involve $\bar{N}_1^2, \bar{N}_2^2, \bar{N}_1^3, \bar{N}_2^3,$ etc., thus essentially linearizing the equations that describe the perturbation around the steady state. Eqs. (7.90) simplify to:

$$\begin{aligned} \frac{d\bar{N}_1}{dt} &= \left(\frac{\partial f_1}{\partial N_1}\right)^* \bar{N}_1 + \left(\frac{\partial f_1}{\partial N_2}\right)^* \bar{N}_2 \\ \frac{d\bar{N}_2}{dt} &= \left(\frac{\partial f_2}{\partial N_1}\right)^* \bar{N}_1 + \left(\frac{\partial f_2}{\partial N_2}\right)^* \bar{N}_2 \end{aligned} \tag{7.91}$$

The matrix of partial derivatives is the *Jacobian* of the original set of differential equations evaluated near the neighborhood of the steady state:

$$\mathbf{J}^* = \begin{bmatrix} \left(\frac{\partial f_1}{\partial N_1}\right)^* & \left(\frac{\partial f_1}{\partial N_2}\right)^* \\ \left(\frac{\partial f_2}{\partial N_1}\right)^* & \left(\frac{\partial f_2}{\partial N_2}\right)^* \end{bmatrix} \tag{7.92}$$

It should be obvious that Eq. (7.91) is a set of simultaneous linear ordinary differential equations of the form

$$\bar{\mathbf{N}}' = \mathbf{J}^* \bar{\mathbf{N}} \tag{7.93}$$

It was demonstrated in Sec. 7.4 that the solution of a set of linear ordinary differential equations of the form of Eq. (7.18) can be obtained from the eigenvalues of the matrix A. Similarly, the solution of Eq. (7.93) will depend on the eigenvalues of the Jacobian matrix. The eigenvalues could be real positive, real negative, and/or complex with positive or negative real parts.

Let us show the eigenvalues in their most general form

$$\lambda_k = a_k \pm b_k i \qquad k = 1, 2, \ldots, n \tag{7.94}$$

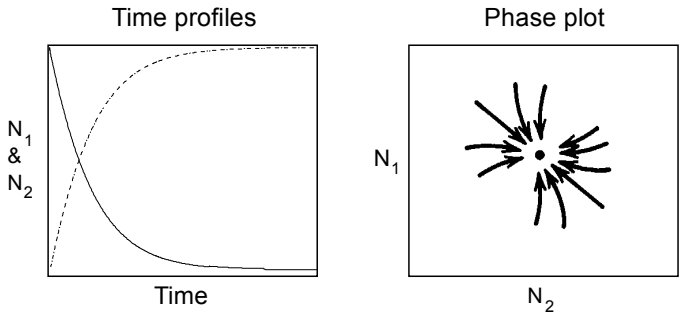
where a_k are the real parts, b_k are the coefficients of the imaginary parts of the eigenvalues, and $i = \sqrt{-1}$; remembering that complex eigenvalues appear as conjugate pairs. We now summarize all possible cases and their stability analysis in Table 7.3, and show time profiles and phase plots of (N_1 vs. N_2) for the corresponding

cases in Fig. 7.5. Negative eigenvalues result in stable solutions (Cases 1 & 2), while positive eigenvalues cause instability (Cases 3 & 4). The presence of complex eigenvalues introduces oscillatory behavior in the solutions (Cases 2, 4, & 6). If both positive and negative real values exist, the solution is a metastable saddle point (Case 5). Finally, if the eigenvalues are complex and the real parts are zero, the results are neutrally stable oscillatory (Case 6).

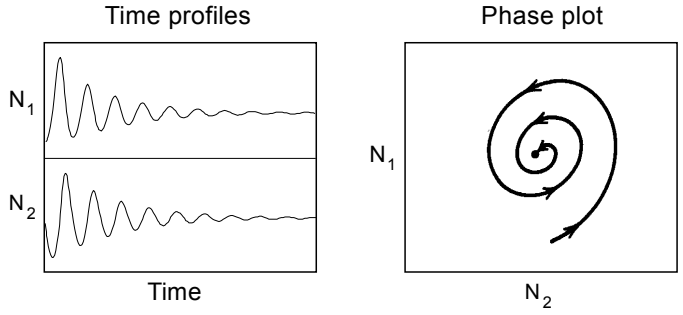
Similar analysis applies to sets of equations that contain n dependent variables (where $n > 2$). In that case, the Jacobian is of size ($n \times n$), and phase plots of pairs of variables are constructed. Three-dimensional phase plots may also be constructed, if their use is deemed instructive.

Table 7.3 Stability Analysis Based on the Eigenvalues of the Jacobian Matrix.

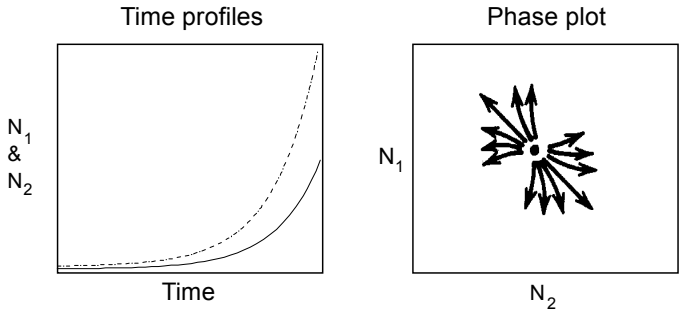
Case	a_k	b_k	**Stability analysis**
1	All negative	Zero	Stable, nonoscillatory
2	All negative	Nonzero	Stable, oscillatory
3	All positive	Zero	Unstable, nonoscillatory
4	All positive	Nonzero	Unstable, oscillatory
5	Positive and negative	Zero	Metastable, saddle point
6	Zero	Nonzero	Neutrally stable, oscillatory



Case 1 - Stable node: no oscillations
Eigenvalues: negative real parts
zero complex parts

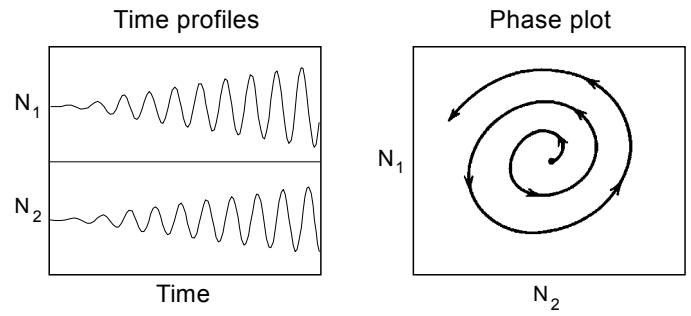


Case 2 - Stable focus: stable damped oscillations
Eigenvalues: negative real parts
nonzero complex parts

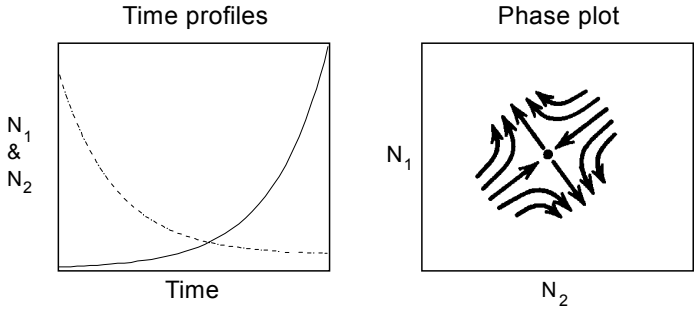


Case 3 - Unstable node: no oscillations
Eigenvalues: positive real parts
zero complex parts

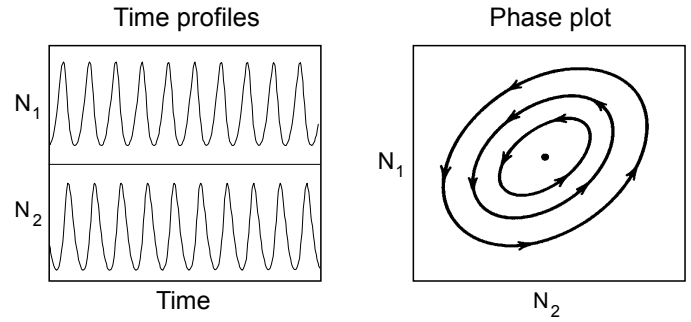
Figure 7.5 Time profiles and phase plots for stability analysis.



Case 4 - Unstable focus: unstable oscillations
Eigenvalues: positive real parts
nonzero complex parts



Case 5 - Metastable saddle point
Eigenvalues: one positive real
one negative real
zero complex parts



Case 6 - Neutrally stable oscillations
Eigenvalues: zero real parts
nonzero complex parts

Figure 7.5 (cont.) Time profiles and phase plots for stability analysis.

Example 7.3 Solution of enzyme catalysis reactions.

Statement of the problem

An enzyme, *E*, catalyzes the conversion of a substrate, *S*, to form a product, *P*, via the formation of an intermediate complex, *ES*, as shown below:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_2} P + E$$

Apply the law of mass action to this simple enzymatic reaction to obtain the differential equations that describe the dynamics of the reaction. Use the following values of initial conditions and rate constants to integrate the differential equations and plot the time profiles for all variables in the model:

Initial Conditions: $[S]_0 = 1.0\ \mu M$ $[E]_0 = 0.1\ \mu M$ $[ES]_0 = 0$ $[P]_0 = 0$

Constants: $k_1 = 0.1\ (\mu M)^{-1}s^{-1}$ $k_{-1} = 0.1\ s^{-1}$ $k_2 = 0.3\ s^{-1}$

Determine the time (in seconds) it takes for the reaction to reach 99.9% conversion of the substrate.

Solution

The law of mass action states that the rate of molecular collision of two chemical species in a dilute gas or solution is proportional to the product of the two concentrations. Based on this, the model equations are:

$$\frac{d[S]}{dt} = -k_1[S][E] + k_{-1}[ES] \qquad [S]_0 = 1.0$$
$$\frac{d[E]}{dt} = -k_1[S][E] + k_{-1}[ES] + k_2[ES] \qquad [E]_0 = 0.1$$
$$\frac{d[ES]}{dt} = k_1[S][E] - k_{-1}[ES] - k_2[ES] \qquad [ES]_0 = 0$$
$$\frac{d[P]}{dt} = k_2[ES] \qquad [P]_0 = 0$$

We integrate the equations for the period 0 to 1000 seconds using the program listed below as `example7_3.m` and the function `enzyme_kinetics_equations.m`:

```
% example7_3.m - Integration of simple enzyme kinetics model
% using MATLAB function ode45.m to integrate the differential
```

```
% equations that are contained in the file:
% enzyme_kinetics_equations.m

clc; clear all;
% Set the initial conditions, constants, & time span
yzero=[1, 0.1, 0, 0];
k1=0.1; k_1=0.1; k2=0.3;
tspan=[0 1000];

% Integrate the equations
[t,y]=ode45('enzyme_kinetics_equations',tspan,yzero,[],k1,k_1,k2);
n=length(t);

% Print out the results
n=length(y);
for i=1:n
    if y(i,1)<=0.001*yzero(1)
    fprintf('Reaction is 99.9 percent complete at time = %4.0f
seconds',t(i));
    break
    end
end
% Plot concentration profiles
clf; figure(1); plot(t,y(:,1),'-',t,y(:,4),'-.')
title('Figure E7.3a: Concentration Profiles of Substrate and
Product', 'FontSize',12)
xlabel('Time, s','FontSize',12);
ylabel('Concentration, \muM', 'FontSize',12);
legend('S','P');
figure(2); plot(t,y(:,2),'-',t,y(:,3),'-.')
title('Figure E7.3b: Concentration Profiles of Enzyme and
Complex', 'FontSize',12)
xlabel('Time, s','FontSize',12);
ylabel('Concentration, \muM', 'FontSize',12);
legend('E','ES')
```

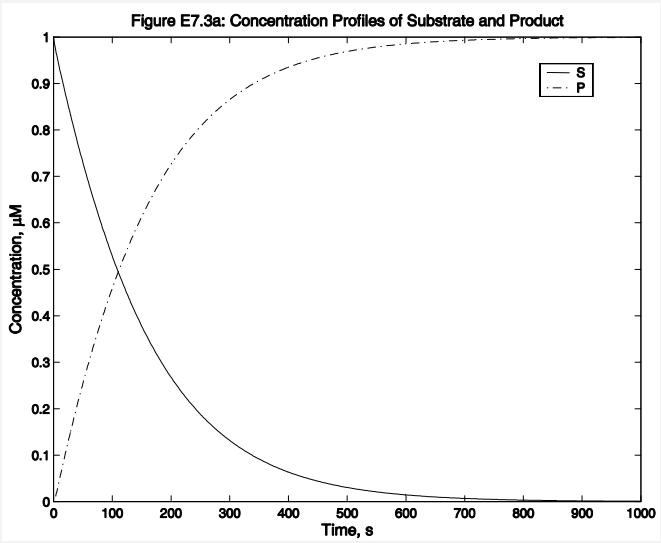
Function that contains equations (`enzyme_kinetics_equations.m`)

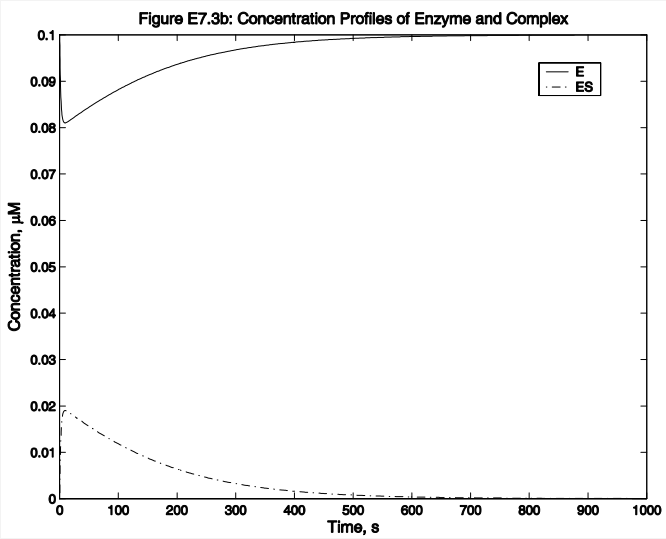
```
function dy=enzyme_kinetics_equations(t,y,flag,k1,k_1,k2)
% enzyme_kinetics_equations.m
% Contains the equations for example7_3

% Variables
S=y(1); E=y(2); ES=y(3);
% Equations
dy=[-k1*S*E+k_1*ES
    -k1*S*E+k_1*ES+k2*ES
    k1*S*E-k_1*ES-k2*ES
    k2*ES];
```

Results

The plots show that the enzyme complex, [*ES*], forms quickly within the first few seconds of the reaction. The substrate gets converted steadily to product. The program determines that the reaction reaches 99.9% conversion at 960 seconds. By this time, the enzyme complex disappears and the enzyme returns back to its original free state.

```
Reaction is 99.9 percent complete at time = 960 seconds
```





7.7 Numerical Stability and Error Propagation

Topics of paramount importance in the numerical integration of differential equations are the error propagation, stability, and convergence of these solutions. Two types of stability considerations enter in the solution of ordinary differential equations: inherent stability (or instability) and numerical stability (or instability). Inherent stability is determined by the mathematical formulation of the problem and is dependent on the eigenvalues of the Jacobian matrix of the differential equations, as was shown in Sec. 7.6. On the other hand, numerical stability is a function of the error propagation in the numerical integration method. The behavior of error propagation depends on the values of the characteristic roots of the difference equations that yield the numerical solution. In this section, we concern ourselves with numerical stability considerations as they apply to the numerical integration of ordinary differential equations.

There are three types of errors present in the application of numerical integration methods. These are the *truncation error*, the *roundoff error*, and the *propagation error*. The truncation error is a function of the number of terms that are retained in the approximation of the solution from the infinite series expansion. The truncation error may be reduced by retaining a larger number of terms in the series or by reducing the step size of integration h. The plethora of available numerical methods of integration of ordinary differential equations provides a choice of increasingly higher accuracy (lower truncation error), at an escalating cost in the number of arithmetic operations to be performed, and with the concomitant accumulation of roundoff errors.

Computers carry numbers using a finite number of significant figures. A roundoff error is introduced in the calculation when the computer rounds up or down (or just chops) the number to n significant figures. Roundoff errors may be reduced significantly by the use of double precision. However, even a very small roundoff error may affect the accuracy of the solution, especially in numerical integration methods that march forward (or backward) for hundreds or thousands of steps, each step being performed using rounded numbers.

The truncation and roundoff errors in numerical integration accumulate and propagate, creating the propagation error, which, in some cases, may grow in exponential or oscillatory pattern, thus causing the calculated solution to deviate drastically from the correct solution.

Fig. 7.6 illustrates the propagation of error in a numerical integration method. Starting with a known initial condition y_0, the method calculates the value y_1, which contains the truncation error for this step and a small roundoff error introduced by the computer. The error has been magnified in order to illustrate it more clearly. The next step starts with y_1 as the initial point and calculates y_2. But because y_1 already contains truncation and roundoff errors, the value obtained for y_2 contains these

errors propagated, in addition to the new truncation and roundoff errors from the second step. The same process occurs in subsequent steps.

Error propagation in numerical integration methods is a complex operation that depends on several factors. Roundoff error, which contributes to propagation error, is entirely determined by the accuracy of the computer being used. The truncation error is fixed by the choice of method being applied, by the step size of integration, and by the values of the derivatives of the functions being integrated. For these reasons, it is necessary to examine the error propagation and stability of each method individually and in connection with the differential equations to be integrated. Some techniques work well with one class of differential equations but fail with others.

In the sections that follow, we examine systematically the error propagation and stability of several numerical integration methods and suggest ways of reducing these errors by the appropriate choice of step size and integration algorithm.

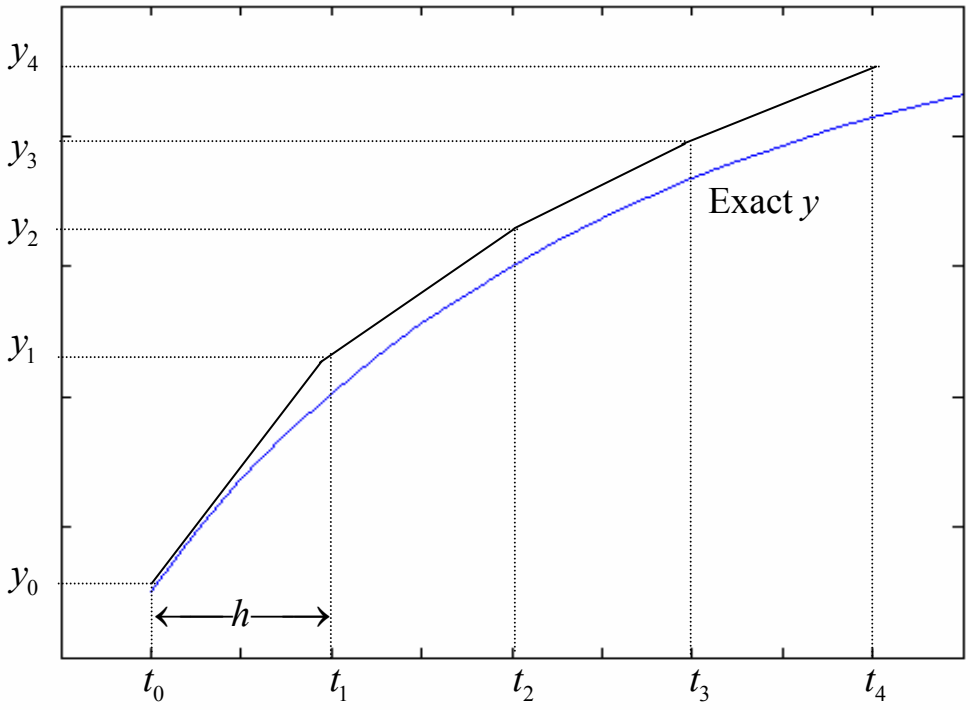


Figure 7.6 Error propagation in numerical integration methods. The error has been magnified in order to illustrate it more clearly.

7.7.1 Stability of the Euler methods

Let us consider the initial-value differential equation in the linear form:

$$\frac{dy}{dt} = \lambda y \tag{7.95}$$

where the initial condition is given as

$$y(t_0) = y_0 \tag{7.96}$$

We assume that λ is real and y_0 is finite. The analytical solution of this differential equation is

$$y(t) = y_0 e^{\lambda t} \tag{7.97}$$

This solution is *inherently stable* for $\lambda < 0$. Under these conditions:

$$\lim_{t\to\infty} y(t) = 0 \tag{7.98}$$

Next, we examine the stability of the numerical solution of this problem obtained from using the explicit Euler method. Momentarily we ignore the truncation and roundoff errors. Applying Eq. (7.42), we obtain the recurrence equation

$$y_{n+1} = y_n + h\lambda y_n \tag{7.99}$$

which rearranges to the following *first-order homogeneous difference equation*

$$y_{n+1} - (1 + h\lambda) y_n = 0 \tag{7.100}$$

Using the methods described in Sec. 6.???, we obtain the characteristic equation

$$E - (1 + h\lambda) = 0 \tag{7.101}$$

whose root is

$$\mu_1 = (1 + h\lambda) \tag{7.102}$$

From this, we obtain the solution of the difference equation (7.100) as

$$y_n = C(1 + h\lambda)^n \tag{7.103}$$

The constant C is calculated from the initial condition, at $t = t_0$:

$$n = 0 \qquad y_n = y_0 = C \tag{7.104}$$

Therefore, the final form of the solution is

$$y_n = y_0 \left(1 + h\lambda\right)^n \tag{7.105}$$

The differential equation is an initial-value problem; therefore, n can increase without bound. Because the solution y_n is a function of $\left(1 + h\lambda\right)^n$, its behavior is determined by the value of $\left(1 + h\lambda\right)$. A numerical solution is said to be *absolutely stable* if

$$\lim_{n \to \infty} y_n = 0 \tag{7.106}$$

The numerical solution of the differential equation (7.95) using the explicit Euler method is absolutely stable if

$$\left|1 + h\lambda\right| \le 1 \tag{7.107}$$

Because $\left(1 + h\lambda\right)$ is the root of the characteristic equation (7.101), an alternative definition of absolute stability is

$$\left|\mu_i\right| \le 1 \qquad i = 1, 2, \ldots, k \tag{7.108}$$

where more than one root exists in the multi-step numerical methods.

Returning to the problem at hand, the inequality (7.107) is rearranged to

$$-2 \le h\lambda \le 0 \tag{7.109}$$

This inequality sets the limits of the integration step size for a stable solution as follows: Because h is positive, then $\lambda < 0$ and

$$h \le \frac{2}{\left|\lambda\right|} \tag{7.110}$$

Inequality (7.110) is a finite *general stability boundary*, and for this reason, the explicit Euler method is called *conditionally stable*. Any method with an infinite general stability boundary can be called *unconditionally stable*.

At the outset of our discussion, we assumed that λ was real in order to simplify the derivation. This assumption is not necessary: λ can be a complex number. In the earlier discussion of the stability of difference equations (Sec. ???), we mentioned that a solution is stable, converging with damped oscillations, when complex roots are present, and the moduli of the roots are less than or equal to unity:

$$|r| \le 1 \tag{7.111}$$

The two inequalities (7.109) and (7.111) describe the circle with a radius of unity on the complex plane shown in Fig. 7.7. Since the explicit Euler method can be categorized as a first-order Runge-Kutta method, the corresponding curve in this figure is marked by RK1. The set of values of $h\lambda$ inside the circle yields stable numerical solutions of Eq. (7.95) using the Euler integration method.

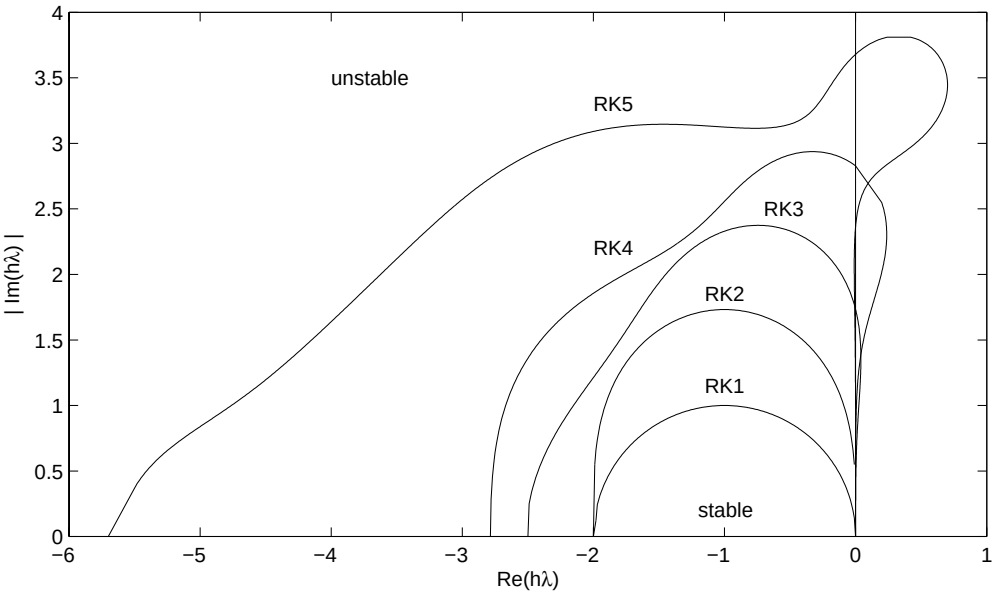


Figure 7.7 Stability regions in the complex plane for Runge-Kutta methods of order 1 (explicit Euler), 2, 3, 4, and 5.

We now return to the consideration of the truncation and roundoff errors of the Euler method and develop a difference equation, which describes the propagation of the error in the numerical solution. We work with the nonlinear form of the initial-value problem

$$\frac{dy}{dt} = f(t, y) \tag{7.112}$$

where the initial condition is given by

$$y(t_0) = y_0 \tag{7.113}$$

We define the accumulated error of the numerical solution at step $(n+1)$ as

$$\varepsilon_{n+1} = y_{n+1} - y(t_{n+1}) \tag{7.114}$$

where $y(t_{n+1})$ is the *exact* value of y, and y_{n+1} is the *calculated* value of y at t_{n+1}. We then write the exact solution, $y(t_{n+1})$, as a Taylor series expansion, showing as many terms as needed for the Euler method:

$$y(t_{n+1}) = y(t_n) + hf\left(t_n, y(t_n)\right) + T_{E,n+1} \tag{7.115}$$

where $T_{E,\,n\,+\,1}$ is the local truncation error for step $(n+1)$. We also write the calculated value y_{n+1} obtained from the explicit Euler formula

$$y_{n+1} = y_n + hf\left(t_n, y_n\right) + R_{E,n+1} \tag{7.116}$$

where $R_{E,n+1}$ is the roundoff error introduced by the computer in step $(n+1)$. Combining Eqs. (7.114)-(7.116) we have

$$\varepsilon_{n+1} = y_n - y(t_n) + h\left[f\left(t_n, y_n\right) - f\left(t_n, y(t_n)\right)\right] - T_{E,n+1} + R_{E,n+1} \tag{7.117}$$

which simplifies to

$$\varepsilon_{n+1} = \varepsilon_n + h\left[f\left(t_n, y_n\right) - f\left(t_n, y(t_n)\right)\right] - T_{E,n+1} + R_{E,n+1} \tag{7.118}$$

The mean-value theorem

$$f\left(t_n, y_n\right) - f\left(t_n, y(t_n)\right) = \left.\frac{\partial f}{\partial y}\right|_{\alpha, x_n} \left[y_n - y(t_n)\right] \qquad y_n < \alpha < y(t_n) \tag{7.119}$$

can be used to further modify the error equation (7.118) to

$$\varepsilon_{n+1} - \left[1 + h\left.\frac{\partial f}{\partial y}\right|_{\alpha, x_n}\right]\varepsilon_n = -T_{E,n+1} + R_{E,n+1} \tag{7.120}$$

This is a *first-order nonhomogeneous difference equation with varying coefficients,* which can be solved only by iteration. However, by making the following simplifying assumptions

$$T_{E,n+1} = T_E = \text{constant} \tag{7.121}$$

$$R_{E,n+1} = R_E = \text{constant} \tag{7.122}$$

$$\left.\frac{\partial f}{\partial y}\right|_{\alpha, x_n} = \lambda = \text{constant} \tag{7.123}$$

Eq. (7.120) simplifies to

$$\varepsilon_{n+1} - (1 + h\lambda)\varepsilon_n = -T_E + R_E \tag{7.124}$$

whose solution is given by the sum of the homogeneous and particular solutions:

$$\varepsilon_n = C_1 (1 + h\lambda)^n + \frac{-T_E + R_E}{1 - (1 + h\lambda)} \tag{7.125}$$

Comparison of Eqs. (7.100) and (7.124) reveals that the characteristic equations for the solution y_n and the error ε_n are identical. The truncation and roundoff error terms in Eq.(7.124) introduce the particular solution. The constant C_1 is calculated by assuming that the initial condition of the differential equation has no error; that is, $\varepsilon_0 = 0$. The final form of the equation that describes the behavior of the propagation error is

$$\varepsilon_n = \frac{-T_E + R_E}{h\lambda}\left[(1 + h\lambda)^n - 1\right] \tag{7.126}$$

A great deal of insight can be gained by thoroughly examining Eq. (7.126). As expected, the value of $(1 + h\lambda)$ is the determining factor in the behavior of the propagation error. Consider first the case of a fixed finite step size *h*, with the number of integration steps increasing to a very large *n*. The limit on the error as $n \to \infty$ is

$$\lim_{n\to\infty} |\varepsilon_n| = \frac{-T_E + R_E}{h\lambda} \qquad \text{for} \quad |1 + h\lambda| < 1 \tag{7.127}$$

$$\lim_{n\to\infty} |\varepsilon_n| = \infty \qquad \text{for} \quad |1 + h\lambda| > 1 \tag{7.128}$$

In the first situation [Eq. (7.127)], $\lambda < 0$, $0 < h < \dfrac{2}{|\lambda|}$, the error is bounded, and the numerical solution is stable. The numerical solution differs from the exact solution by only the finite quantity $\dfrac{-T_E + R_E}{h\lambda}$, which is a function of the truncation error, the roundoff error, the step size, and the eigenvalue of the differential equation.

In the second situation [Eq. (7.128)], $\lambda > 0$, $h > 0$, the error is unbounded and the numerical solution is unstable. For $\lambda > 0$, however, the exact solution itself is *inherently unstable*. For this reason we introduce the concept of *relative error* defined as

$$\text{relative error} = \frac{\varepsilon_n}{y_n} \tag{7.129}$$

Utilizing Eqs. (7.105) and (7.126), we obtain the relative error as

$$\frac{\varepsilon_n}{y_n} = \frac{-T_E + R_E}{y_0 h\lambda}\left[1 - \frac{1}{(1+h\lambda)^n}\right] \tag{7.130}$$

The relative error is bounded for $\lambda > 0$ and unbounded for $\lambda < 0$. So we conclude that for inherently stable differential equations, the absolute propagation error is the pertinent criterion for numerical stability, whereas for inherently unstable differential equations, the relative propagation error must be investigated.

Let us now consider a fixed interval of integration, $0 < t < \alpha$, so that

$$h = \frac{\alpha}{n} \tag{7.131}$$

and we increase the number of integration steps to a very large n. This, of course, causes $h \to 0$. A numerical method is said to be *convergent* if

$$\lim_{h\to 0} |\varepsilon_n| = 0 \tag{7.132}$$

In the absence of roundoff error, the Euler method, and most other integration methods, would be convergent because

$$\lim_{h\to 0} T_E = 0 \tag{7.133}$$

therefore, Eq. (7.132) would be true. However, roundoff error is *never* absent in numerical calculations. As $h \to 0$ the roundoff error is the crucial factor in the propagation of error:

$$\lim_{h\to 0}\left|\varepsilon_n\right| = R_E \lim_{h\to 0}\frac{\left(1+h\lambda\right)^n - 1}{h\lambda} \tag{7.134}$$

Application of L'Hôpital's rule shows that the roundoff error propagates unbounded as the number of integration steps becomes very large:

$$\lim_{h\to 0}\varepsilon_n = R_E\left[\infty\right] \tag{7.135}$$

This is the "catch 22" of numerical methods: A smaller step size of integration reduces the truncation error but requires a large number of steps, thereby increasing the roundoff error.

A similar analysis of the *implicit Euler method* (backward Euler) results in the following two equations, for the solution

$$y_{n+1} = \frac{y_0}{\left(1-h\lambda\right)^n} \tag{7.136}$$

and the propagation error

$$\varepsilon_{n+1} = \frac{-T_E + R_E}{h\lambda}\left(1-h\lambda\right)\left[\frac{1}{\left(1-h\lambda\right)^n} - 1\right] \tag{7.137}$$

For $\lambda < 0$ and $0 < h < \infty$, the solution is stable:

$$\lim_{n\to\infty} y_n = 0 \tag{7.138}$$

and the error is bounded:

$$\lim_{n\to\infty}\varepsilon_n = -\frac{-T_E + R_E}{h\lambda}\left(1-h\lambda\right) \tag{7.139}$$

No limitation is placed on the step size; therefore, the implicit Euler method is *unconditionally stable* for $\lambda < 0$. On the other hand, when $\lambda > 0$, the following inequality must be true for a stable solution:

$$\left|1-h\lambda\right| \le 1 \tag{7.140}$$

This imposes the limit on the step size:

$$-2 \le h\lambda < 0 \tag{7.141}$$

It can be concluded that the implicit Euler method has a wider range of stability than the explicit Euler method (see Table 7.4).

7.7.2 Stability of the Runge-Kutta methods

Using methods parallel to those of the previous section, the recurrence equations and the corresponding roots for the Runge-Kutta methods can be derived (Lapidus and Sienfeld, 1971). For the differential equation (7.95), these are:

Second-order Runge-Kutta:

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2\right) y_n \tag{7.142}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 \tag{7.143}$$

Third-order Runge-Kutta:

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3\right) y_n \tag{7.144}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 \tag{7.145}$$

Fourth-order Runge-Kutta:

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4\right) y_n \tag{7.146}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 \tag{7.147}$$

Fifth-order Runge-Kutta:

$$y_{n+1} = \left(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 + \frac{1}{120}h^5\lambda^5 + \frac{0.5625}{720}h^6\lambda^6\right) y_n \tag{7.148}$$

$$\mu_1 = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4 + \frac{1}{120}h^5\lambda^5 + \frac{0.5625}{720}h^6\lambda^6 \tag{7.149}$$

The last term in the right-hand side of Eqs. (7.148) and (7.149) is specific to the fifth-order Runge-Kutta, which appears in Table 7.2 and varies for different fifth-order formulas. The condition for absolute stability

$$|\mu_i| \le 1 \qquad i = 1, 2, \ldots, k \tag{7.150}$$

applies to all the above methods. The absolute real stability boundaries for these methods are listed in Table 7.4, and the regions of stability in the complex plane are shown on Fig. 7.6. In general, as the order increases, so do the stability limits.

Table 7.4 Real stability boundaries

Method	Boundary
Explicit Euler	$-2 \le h\lambda < 0$
Implicit Euler	$\begin{cases} 0 < h < \infty & \text{for} \quad \lambda < 0 \\ -2 \le h\lambda < 0 & \text{for} \quad \lambda > 0 \end{cases}$
Modified Euler (predictor-corrector)	$-1.077 \le h\lambda < 0$
Second-order Runge-Kutta	$-2 \le h\lambda < 0$
Third-order Runge-Kutta	$-2.5 \le h\lambda < 0$
Fourth-order Runge-Kutta	$-2.785 \le h\lambda < 0$
Fifth-order Runge-Kutta	$-5.7 \le h\lambda < 0$
Adams	$-0.546 \le h\lambda < 0$
Adams-Moulton	$-1.285 \le h\lambda < 0$

7.7.3 Stability of multistep methods

Using methods parallel to those of the previous section, the recurrence equations and the corresponding roots for the modified Euler, Adams, and Adams-Moulton methods can be derived (Lapidus and Sienfeld, 1971). For the differential equation (7.95), these are:

Modified Euler (combination of predictor and corrector):

$$y_{n+1} = \left(1 + h\lambda + h^2\lambda^2\right) y_n \tag{7.151}$$

$$\mu_1 = 1 + h\lambda + h^2\lambda^2 \tag{7.152}$$

Adams:

$$y_{n+1} = \left(1 + \frac{23}{12}h\lambda\right) y_n - \left(\frac{4}{3}h\lambda\right) y_{n-1} + \left(\frac{5}{12}h\lambda\right) y_{n-2} \tag{7.153}$$

$$\mu^3 - \left(1 + \frac{23}{12}h\lambda\right)\mu^2 + \left(\frac{4}{3}h\lambda\right)\mu - \left(\frac{5}{12}h\lambda\right) = 0 \tag{7.154}$$

Adams-Moulton (combination of predictor and corrector):

$$\begin{aligned} y_{n+1} = &\left(1 + \frac{7}{6}h\lambda + \frac{55}{64}h^2\lambda^2\right) y_n - \left(\frac{5}{24}h\lambda + \frac{59}{64}h^2\lambda^2\right) y_{n-1} \\ &+ \left(\frac{1}{24}h\lambda + \frac{37}{64}h^2\lambda^2\right) y_{n-2} - \left(\frac{9}{64}h^2\lambda^2\right) y_{n-3} \end{aligned} \tag{7.155}$$

$$\begin{aligned} \mu^4 - &\left(1 + \frac{7}{6}h\lambda + \frac{55}{64}h^2\lambda^2\right)\mu^3 + \left(\frac{5}{24}h\lambda + \frac{59}{64}h^2\lambda^2\right)\mu^2 \\ &- \left(\frac{1}{24}h\lambda + \frac{37}{64}h^2\lambda^2\right)\mu + \left(\frac{9}{64}h^2\lambda^2\right) = 0 \end{aligned} \tag{7.156}$$

The condition for absolute stability

$$|\mu_i| \le 1 \qquad i = 1, 2, \ldots, k \tag{7.150}$$

applies to all the above methods. The absolute real stability boundaries for these methods are also listed in Table 7.4, and the regions of stability in the complex plane are shown on Fig. 7.8.

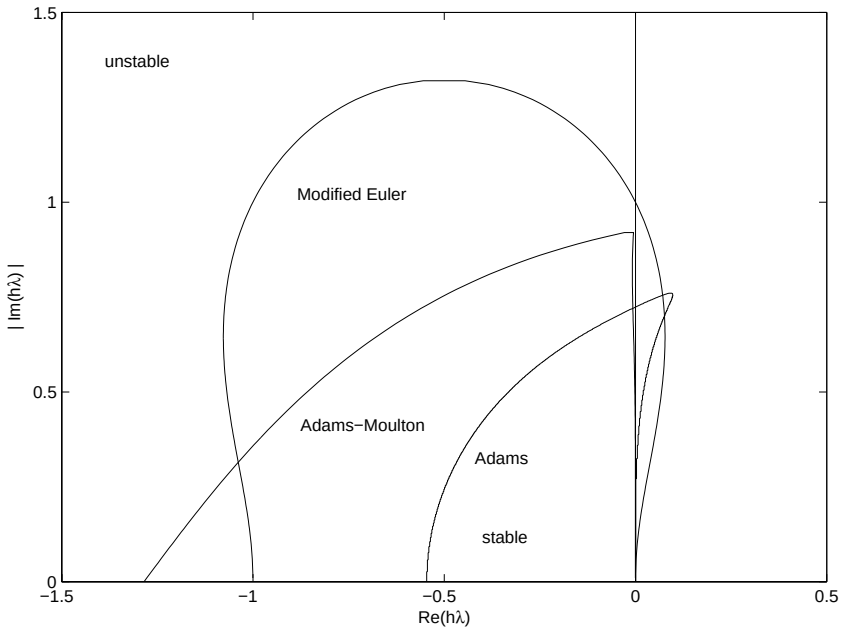


Figure 7.8 Stability regions in the complex plane for the modified Euler (Euler predictor-corrector), Adams, and Adams-Moulton methods.

7.8 Step Size Control

The discussion of stability analysis in the previous sections made the simplifying assumption that the value of λ remains constant throughout the integration. This is true for linear equations such as Eq. (7.95); however, for the nonlinear equation (7.11), the value of λ may vary considerably over the interval of integration. The step size of integration must be chosen using the maximum possible value of λ, thus resulting in the minimum step size. This, of course, will guarantee stability at the expense of computation time. For problems in which computation time becomes excessive, it is possible to develop strategies for automatically adjusting the step size at each step of the integration.

A simple test for checking the step size is to do the calculations at each interval twice: Once with the full step size, and then repeat the calculations over the same interval with a smaller step size, usually half that of the first one. If at the end of the interval, the difference between the predicted values of y by both approaches is less than the specified convergence criterion, the step size may be increased. Otherwise, a larger than acceptable difference between the two calculated y values suggests that the step size is large, and it should be shortened in order to achieve an acceptable truncation error.

Another method of controlling the step size is to obtain an estimation of the truncation error at each interval. A good example of such an approach is the Runge-Kutta-Fehlberg method (see Table 7.2), which provides the estimation of the local truncation error. This error estimate can be easily introduced into the computer program, and let the program automatically change the step size at each point until the desired accuracy is achieved.

As mentioned before, the optimum number of application of corrector is two. Therefore, in the case of using a predictor-corrector method, if the convergence is achieved before the second corrected value, the step size may be increased. On the other hand, if the convergence is not achieved after the second application of the corrector, the step size should be reduced.

7.9 Stiff Differential Equations

In Sec. 7.7, we showed that the stability of the numerical solution of differential equations depends on the value of $h\lambda$, and that λ together with the stability boundary of the method determine the step size of integration. In the case of the linear differential equation

$$\frac{dy}{dt} = \lambda y \tag{7.95}$$

λ is the eigenvalue of that equation, and it remains constant throughout the integration. The nonlinear differential equation

$$\frac{dy}{dt} = f\left(t, y\right) \tag{7.11}$$

can be linearized at each step using the mean-value theorem (7.119), so that λ can be obtained from the partial derivative of the function with respect to y:

$$\lambda = \left.\frac{\partial f}{\partial y}\right|_{\alpha, t_n} \tag{7.157}$$

The value of λ is no longer a constant but varies in magnitude at each step of the integration.

This analysis can be extended to a set of simultaneous nonlinear differential equations:

$$\begin{aligned}\frac{dy_1}{dt} &= f_1\left(t, y_1, y_2, \ldots, y_n\right)\\ \frac{dy_2}{dt} &= f_2\left(t, y_1, y_2, \ldots, y_n\right)\\ &\vdots\\ \frac{dy_n}{dt} &= f_n\left(t, y_1, y_2, \ldots, y_n\right)\end{aligned} \tag{7.84}$$

Linearization of the set produces the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_1}{\partial y_1} & \cdots & \dfrac{\partial f_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial y_1} & \cdots & \dfrac{\partial f_n}{\partial y_n} \end{bmatrix} \tag{7.158}$$

The eigenvalues $\{\lambda \mid i = 1, 2, \ldots, n\}$ of the Jacobian matrix are the determining factors in the stability analysis of the numerical solution. The step size of integration is determined by the stability boundary of the method and the maximum eigenvalue.

When the eigenvalues of the Jacobian matrix of the differential equations are all of the same order of magnitude, no unusual problems arise in the integration of the set. However, when the maximum eigenvalue is several orders of magnitude larger than the minimum eigenvalue, the equations are said to be *stiff*. The *stiffness ratio* (SR) of such a set is defined as

$$\mathrm{SR} = \frac{\max\limits_{1 \le i \le n} \left|\mathrm{Real}\left(\lambda_i\right)\right|}{\min\limits_{1 \le i \le n} \left|\mathrm{Real}\left(\lambda_i\right)\right|} \tag{7.159}$$

The step size of integration is determined by the largest eigenvalue, and the final time of integration is usually fixed by the smallest eigenvalue; therefore, integration of stiff differential equations using explicit methods may be time intensive.

The MATLAB functions *ode23s* and *ode15s* are solvers suitable for the solution of stiff ordinary differential equations (see Table 7.1).

7.10 Advanced examples

Example 7.4 Modeling the glycolysis pathways of living cells.

Statement of the problem

An important step in the glycolytic pathway is the phosphorylation of fructose 6-phosphate to fructose 1,6-biphosphate. This reaction is catalyzed by the enzyme phosphofructokinase. This enzyme is an example of an allosteric enzyme that is inhibited by ATP and stimulated by adenosine diphosphate (ADP) or by adenosine-monophosphate (AMP). The enzyme becomes active when it combines with γ molecules of ADP:

$$\text{Enzyme} + \gamma\text{ADP} \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} \text{Enzyme-ADP}^{\gamma}$$

The active complex catalyzes the reaction of fructose 6-phosphate to fructose 1,6-biphosphate, and in this process it converts one molecule of ATP to one molecule of ADP, as follows:

$$\begin{array}{c} \text{Fructose 6-phosphate} \\ \Updownarrow \\ \text{ATP} + \text{Enzyme-ADP}^{\gamma} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \text{ATP-Enzyme-ADP}^{\gamma} \xrightarrow{k_2} \text{Enzyme-ADP}^{\gamma} + \text{ADP} \\ \Updownarrow \\ \text{Fructose 1,6-biphosphate} \end{array}$$

This is the Sel'kov model as discussed by Keener and Sneyd (1998). Since the net result of this reaction is the formation of an additional ADP molecule that may further activate the enzyme, this reaction has a positive feedback effect on itself. Assuming that there is a steady supply of the ATP available to this reaction at the rate of v_1, and an irreversible flow of ADP away from the reaction at the rate of v_2, the steps of the reaction that involve the consumption of ATP and the formation of ADP, via the formation of enzyme complexes, may be shown schematically as:

$$\begin{aligned} &\xrightarrow{v_1} S_1 \\ \gamma S_2 + E &\underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} ES_2^{\gamma} \\ S_1 + ES_2^{\gamma} &\underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} S_1ES_2^{\gamma} \xrightarrow{k_2} ES_2^{\gamma} + S_2 \\ S_2 &\xrightarrow{v_2} \end{aligned}$$

where S_1 represents the ATP molecule, S_2 stands for the ADP molecule, and E represents the enzyme phosphofructokinase.

Keener and Sneyd applied the law of mass action to this reaction scheme to obtain the following set of ordinary differential equations that describe the dynamics of the reactions:

$$\begin{aligned}
\frac{ds_1}{dt} &= \nu_1 - k_1 s_1 x_1 + k_{-1} x_2 \\
\frac{ds_2}{dt} &= k_2 x_2 - k_3 s_2^{\gamma} e + k_{-3} x_1 - \nu_2 s_2 \\
\frac{dx_1}{dt} &= -k_1 s_1 x_1 + (k_{-1} + k_2) x_2 + k_3 s_2^{\gamma} e - k_{-3} x_1 \\
\frac{dx_2}{dt} &= k_1 s_1 x_1 - (k_{-1} + k_2) x_2 \\
\frac{de}{dt} &= -\frac{dx_1}{dt} - \frac{dx_2}{dt}
\end{aligned} \tag{7.160}$$

where $s_1 = [\text{ATP}]$, $s_2 = [\text{ADP}]$, $e = [\text{E}]$, $x_1 = \left[\text{ES}_2^{\gamma}\right]$, $x_2 = \left[\text{S}_1\text{ES}_2^{\gamma}\right]$. The square brackets are used to denote concentration of the particular compound in the cell. The last equation that describes the rate of change (de/dt) of the free enzyme is obtained from the balance equation for the total enzyme in the cell (e_0), assuming that the total amount of enzyme remains constant:

$$e + x_1 + x_2 = e_0 \tag{7.161}$$

The above equations are a set of simultaneous first order nonlinear ordinary differential equations. Methods of solution for such a set were developed in Sec. 7.5, and are applied here to obtain the solution of the glycolysis problem in this example.

Solution

(a) It is well known in the literature that the rate of glycolysis is oscillatory in nature. To show this, integrate the above set of differential equations with the following initial conditions and constants:

Initial Conditions: $s_1(0) = 1.0$ $s_2(0) = 0.2$ $x_1(0) = 0$ $x_2(0) = 0$ $e_0(0) = 1.4$

Constants: $\gamma = 2.0$ $\nu_1 = 0.003$ $\nu_2 = 2.5*\nu_1$ $k_1 = 0.1$
$k_{-1} = 0.2$ $k_2 = 0.1$ $k_3 = 0.2$ $k_{-3} = 0.2$

Note: The constants contain units of time (seconds) and concentrations (nM) as needed for unit consistency of the equations.

Plot the concentration profiles of all five dependent variables and discuss the results. Plot the phase plot of s_1 and s_2, and discuss what this phase plot demonstrates.

(b) Perform a stability analysis of these equations by examining the eigenvalues of the Jacobian matrix evaluated around the steady state. How do the eigenvalues predict the oscillatory behavior of the concentration vs. time profiles?

(a) **Integration of equations**

The program `example7_4a.m`, listed below, integrates the differential equations using *ode45* and plots the results.

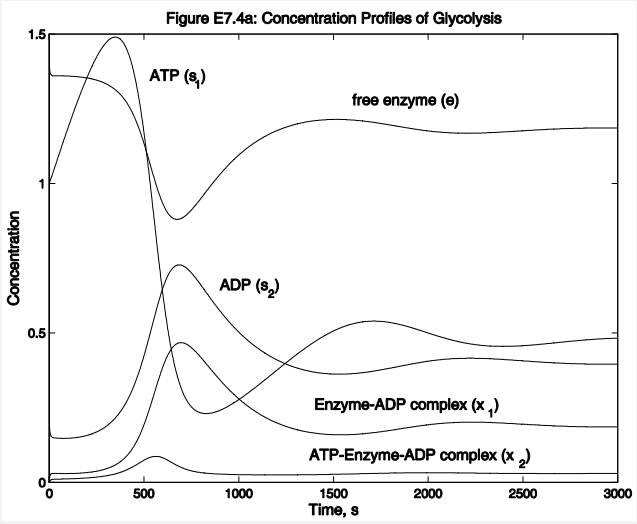
```
% example7_4a.m - Integration of the glycolysis model
% using the MATLAB function ode45.m to integrate the
% differential equations that are contained in the file:
% glycolysis_equations.m

clc; clear all;

% Set the initial conditions & time span
yzero=[1, .2, 0, 0, 1.4];
tspan=[0 3000];

% Integrate the equations
[t,y]=ode45(@glycolysis_equations,tspan,yzero);
n=length(t);

% Plot concentration profiles
clf; figure(1); plot(t,y)
title('Figure E7.4a: Concentration Profiles of Glycolysis')
xlabel('Time, s'); ylabel('Concentration')
text(530,1.35,'ATP (s_1)'); text(900,0.65,'ADP (s_2)')
text(1600,0.25,'Enzyme-ADP complex (x_1)')
text(1600,0.09,'ATP-Enzyme-ADP complex (x_2)')
text(1600,1.28,'free enzyme (e)')

% Plot phase diagrams
figure(2); plot(y(:,1),y(:,2))
title('Figure E7.4b: Phase Plot of Glycolysis')
xlabel('ATP (s_1)'); ylabel('ADP (s_2)')
```

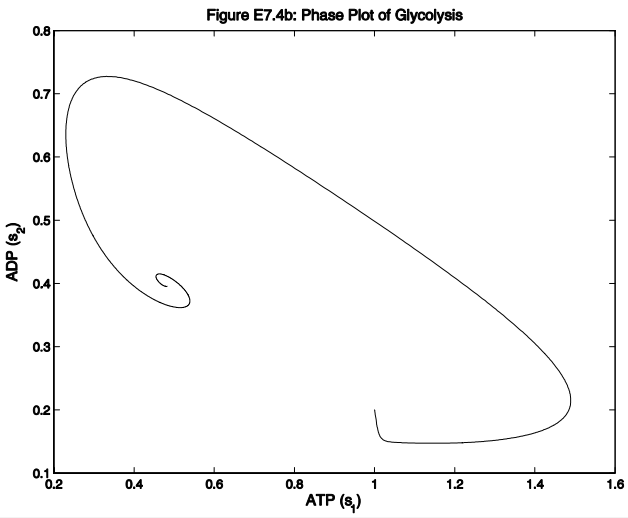
Function that contains equations (`glycolysis_equations.m`):

```
function dy=glycolysis_equations(t,y)
% glycolysis_equations.m
% Contains the glycolysis model for example7_4a

% Constants
```

```
gamma=2; neu1=0.003; neu2=2.5*neu1;
k1=0.1; k_1=2*k1; k2=0.1; k3=0.2; k_3=k3;
s1=y(1); s2=y(2); x1=y(3); x2=y(4); e =y(5);
% Equations
dy=[neu1-k1*s1*x1+k_1*x2
    k2*x2-k3*s2^gamma*e+k_3*x1-neu2*s2
    -k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1
    k1*s1*x1-(k_1+k2)*x2
    -(-k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1)-(k1*s1*x1-
(k_1+k2)*x2)];
```

Results of integration





The concentration profiles (Fig. E7.4a) of the glycolysis system of equations indicate that the above set of constants and initial conditions represent a case that is oscillatory at first, but approaches steady state within 3000 seconds (50 minutes). The phase plot of ADP vs. ATP (Fig. E7.4b) exhibits a stable focus of the type shown on Fig. 7.5 Case (2).

(b) **Steady state analysis of glycolysis equations**

The program `example7_4b.m`, listed below, performs the stability analysis of Eqs. (7.160) by first evaluating the Jacobian matrix of the differential equations using the MATLAB command `jacobian(dy,v)`, where `dy` is the vector of derivatives and v is the vector of variables. Next, it calculates the steady state solution of the differential equations by setting the derivatives equal to zero and solving for the unknown variables using the MATLAB command *solve*. Finally, the stability of the steady state is examined by obtaining the eigenvalues of the Jacobian matrix around the steady state, using the command `eig`.

```
% example7_4b.m - Steady state analysis of the glycolysis model
% using MATLAB functions jacobian.m and eig.m

clc; clear all;

% Set the constants
e0=1.4; gamma=2; neu1=0.003; neu2=2.5*neu1;
k1=0.1; k_1=2*k1; k2=0.1; k3=0.2; k_3=k3;

% Evaluate the Jacobian matrix
syms s1 s2 x1 x2 e
disp('Steady State Analysis of the Glycolysis Equations:')
v=[s1, s2, x1, x2, e];
dy=[neu1-k1*s1*x1+k_1*x2;
    k2*x2-k3*s2^gamma*e+k_3*x1-neu2*s2;
    -k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1;
    k1*s1*x1-(k_1+k2)*x2;
    -(-k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1)-(k1*s1*x1-
(k_1+k2)*x2)];
J=jacobian(dy,v);
disp('The Jacobian matrix is:'), J
% Evaluate the steady state solution
[SteadyState]=solve('neu1-k1*s1*x1+k_1*x2=0',...
    'k2*x2-k3*s2^gamma*e+k_3*x1-neu2*s2=0',...
    '-k1*s1*x1+(k_1+k2)*x2+k3*s2^gamma*e-k_3*x1=0',...
    'k1*s1*x1-(k_1+k2)*x2=0',...
    'e+x1+x2=e0', 's1,s2,x1,x2,e');
disp(' '), disp('The steady state values of each variable are:')
disp('s1'),disp(SteadyState.s1),disp(' ')
disp('s2'),disp(SteadyState.s2),disp(' ')
```

```
disp('x1'),disp(SteadyState.x1),disp(' ')
disp('x2'),disp(SteadyState.x2),disp(' ')
disp('e '),disp(SteadyState.e), disp(' ')
n=length(SteadyState.s1);
disp('Value of each variable at the steady state(s):')
disp('           s1         s2         x1         x2         e')
for i=1:n;
    s1=eval(SteadyState.s1); s2=eval(SteadyState.s2);
    x1=eval(SteadyState.x1); x2=eval(SteadyState.x2);
    e =eval(SteadyState.e);
    fprintf(' %2i   %9.4f  %9.4f  %9.4f  %9.4f  %9 .4f   \n',...
       i, s1(i), s2(i), x1(i), x2(i), e);
end
for i=1:n
    s1=eval(SteadyState.s1); s2=eval(SteadyState.s2);
    x1=eval(SteadyState.x1); x2=eval(SteadyState.x2);
    e =eval(SteadyState.e);
   fprintf('\nSteady state %2i \n',i)
   disp(' '); disp('Jacobian matrix at steady state:'), eval(J)
   disp(' '); disp('Eigenvalues of Jacobian at steady state:');
eig(eval(J))
end
```

Results of steady state analysis

```
Steady State Analysis of the Glycolysis Equations:
The Jacobian matrix is:
J =
[        -1/10*x1,               0,        -1/10*s1,
1/5,               0]
[               0, -2/5*s2*e-3/400,             1/5,
1/10,       -1/5*s2^2]
[        -1/10*x1,        2/5*s2*e,    -1/10*s1-1/5,
3/10,        1/5*s2^2]
[         1/10*x1,               0,         1/10*s1,
-3/10,              0]
[               0,       -2/5*s2*e,             1/5,
0,       -1/5*s2^2]

The steady state values of each variable are:
s1
neu1*(k_1+k2)*(k3*exp(log(neu1/neu2)*gamma)+k_3)/k1/exp(log(neu1/n
eu2)*gamma)/k3/(-neu1+e0*k2)

s2
neu1/neu2

x1
exp(log(neu1/neu2)*gamma)*k3*(-
neu1+e0*k2)/k2/(k3*exp(log(neu1/neu2)*gamma)+k_3)
```

```
x2
neu1/k2

e
k_3*(-neu1+e0*k2)/k2/(k3*exp(log(neu1/neu2)*gamma)+k_3)


Value of each variable at the steady state(s):
           s1         s2         x1         x2         e
  1      0.4763     0.4000     0.1890     0.0300
Steady state  1

Jacobian matrix at steady state:
ans =
   -0.0189         0   -0.0476    0.2000         0
         0   -0.1965    0.2000    0.1000   -0.0320
   -0.0189    0.1890   -0.2476    0.3000    0.0320
    0.0189         0    0.0476   -0.3000         0
         0   -0.1890    0.2000         0   -0.0320

Eigenvalues of Jacobian at steady state:
ans =
  -0.4859
  -0.3060
  -0.0015 + 0.0044i
  -0.0015 - 0.0044i
  -0.0000
```

For this system of equations and constants, the steady state analysis shows that one steady state exists at which the concentrations of the main components are:

[ATP] = 0.4763 [ADP] = 0.4000 [Enzyme-ADP complex] = 0.1890
[ATP-Enzyme-ADP complex] = 0.0300 [free Enzyme] = 1.1810

The eigenvalues of the Jacobian matrix of this system are: two real negative, two complex with negative real parts, and one zero eigenvalue. Such a combination of eigenvalues predicts an oscillatory behavior with dumped oscillations approaching a steady state (see Sec. 0). The zero eigenvalue is a direct consequence of the conservation of mass principle applied to the enzyme (see Eq. (7.9)). These results confirm the evolution of the system shown by the concentration profiles.

Example 7.5 The dynamics of membrane and nerve cell potentials.

Formulation of the problem

The activation and inactivation of the potassium/sodium channels and the role they play in the generation of nerve action potential formed the basis of the Nobel Prize winning work of Hodgkin and Huxley in the 1940s and 50s (Hodgkin and Huxley, 1952). They studied the effect of the application of voltage potentials on the Na+ and K+ channels on the squid giant axon and developed mathematical models that describe the dynamics of the processes.

Numerous papers and books have been written on the Hodgkin-Huxley model. A very concise description of this model is that of Keener and Sneyd (1998). They begin by showing that the cell membrane can be modeled as a capacitor in parallel with an ionic current, and since there can be no buildup of charge on either side of the membrane, the sum of the ionic and capacitive currents must be zero, resulting in the equation

$$C_m \frac{dV}{dt} + I_{ion} = 0 \tag{7.162}$$

where V denotes the internal minus the external potential. In the squid giant axon, and in many nerve cells, the principal ionic currents are the sodium current, I_{Na}, and the potassium current, I_{K}. Other currents that are present, such as the chloride current, are lumped together into one current called the *leakage current*, I_{L}. The ionic currents for sodium and potassium ions can be modeled by the current-voltage relationships

$$I_{\mathrm{Na}} = g_{\mathrm{Na}}(V - V_{\mathrm{Na}}) \tag{7.163}$$

$$I_{\mathrm{K}} = g_{\mathrm{K}}(V - V_{\mathrm{K}}) \tag{7.164}$$

and the leakage current may be shown as

$$I_{\mathrm{L}} = g_{\mathrm{L}}(V - V_{\mathrm{L}}) \tag{7.165}$$

where g_{Na} and g_{K} are the membrane conductances for sodium and potassium ions, respectively, and g_{L} is a combined conductance for leakage current. V_{Na} and V_{K} are the equilibrium membrane potentials due to concentration differences of the two ions, sodium and potassium, and V_{L} is the potential at which the leakage current due to chloride and other ions is zero. The sodium and potassium potentials are calculated from the Nernst equation:

$$V_{\mathrm{Na}} = \frac{RT}{zF} \ln\left(\frac{\left[\mathrm{Na}^+\right]_e}{\left[\mathrm{Na}^+\right]_i} \right) \tag{7.166}$$

$$V_{\mathrm{K}} = \frac{RT}{zF} \ln\left(\frac{\left[\mathrm{K}^+\right]_e}{\left[\mathrm{K}^+\right]_i} \right) \tag{7.167}$$

Ionic channels open and close in response to a voltage. This behavior of ionic channels in response to changes in membrane potential is the basis for electrical excitability, and is of fundamental significance to neurophysiology. According to Keener and Sneyd (1998), the current flow through a population of channels is the product of two terms

$$I = \eta(V,t)\, \phi(V) \tag{7.168}$$

where $\eta(V,t)$ is the proportion of open channels in a population, and $\phi(V)$ is the *I-V* curve of a single channel. The simplest model for the K+ channel assumes that the channel can exist either in the closed state in the proportion of $(1-\eta)$, or in the open state in the proportion of η:

$$\overbrace{(1-\eta)}^{Closed} \underset{\beta(V)}{\overset{\alpha(V)}{\rightleftarrows}} \overbrace{(\eta)}^{Open} \tag{7.169}$$

Then the rate of change of the open channels may be modeled by the differential equation

$$\frac{d\eta}{dt} = \alpha(V)(1-\eta) - \beta(V)\eta \tag{7.170}$$

It is sometimes instructive to write Eq. (7.170) in the form

$$\tau_\eta(V)\frac{d\eta}{dt} = \eta_\infty(V) - \eta \tag{7.171}$$

where $\eta_\infty(V)$ is the steady state value of η, that may be obtained from Eq. (7.170) as

$$\eta_\infty(V) = \frac{\alpha}{\alpha+\beta} \tag{7.172}$$

and τ_η is the time constant of approach to steady state:

$$\tau_\eta = \frac{1}{\alpha + \beta} \tag{7.173}$$

Hodgkin and Huxley used a *voltage clamp* in their studies of the giant squid axon. The user of a voltage clamp fixes the membrane potential by applying a rapid step from one voltage to another and then measures the current that must be applied, I_{app}, to hold the voltage constant. Based on their experimental data, Hodgkin and Huxley modified the potassium conductance, g_K, in order to obtain sigmoidal increase and exponential decrease:

$$g_{\mathrm{K}} = \overline{g}_K n^4 \tag{7.174}$$

They also modified the sodium conductance, g_{Na}, to account for two processes at work, one that turns on the sodium current and one that turns it off:

$$g_{\mathrm{Na}} = \overline{g}_{Na} m^3 h \tag{7.175}$$

Keener and Sneyd (1998) interpret the potassium mechanism to be equivalent to having four "n" gates per potassium channel, all of which must be open for potassium to flow. They also elucidate the mechanism of the Na^+ channel as consisting of three "m" gates and one "h" gate, each of which can be either closed or open. Combining equations (7.162)-(7.165), (7.170), (7.174), and (7.175) results in the complete Hodgkin-Huxley model:

$$\begin{aligned}
C_m \frac{dv}{dt} &= -\overline{g}_K n^4 (v - v_{\mathrm{K}}) - \overline{g}_{Na} m^3 h (v - v_{\mathrm{Na}}) - \overline{g}_L (v - v_{\mathrm{L}}) + I_{\mathrm{app}} \\
\frac{dn}{dt} &= \alpha_n (1 - n) - \beta_n n \\
\frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\
\frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h
\end{aligned} \tag{7.176}$$

The potential, v, is the deviation from rest potential ($v = V - V_{eq}$) measured in units of mV, current density I is in units of $\mu A/cm^2$, conductances are in units of mS/cm^2, and capacitance C_m is in $\mu F/cm^2$. The rate constants of α and β are, in units of $(ms)^{-1}$,

$$\alpha_n = 0.01\frac{10-v}{e^{\left(\frac{10-v}{10}\right)}-1} \qquad \beta_n = 0.125e^{\left(\frac{-v}{80}\right)}$$
$$\alpha_m = 0.1\frac{25-v}{e^{\left(\frac{25-v}{10}\right)}-1} \qquad \beta_m = 4e^{\left(\frac{-v}{18}\right)} \tag{7.177}$$
$$\alpha_h = 0.07e^{\left(\frac{-v}{20}\right)} \qquad \beta_h = \frac{1}{e^{\left(\frac{30-v}{10}\right)}+1}$$

The steady state values of the gating variables and the time constants are:

$$n_\infty = \frac{\alpha_n}{\alpha_n+\beta_n} \qquad m_\infty = \frac{\alpha_m}{\alpha_m+\beta_m} \qquad h_\infty = \frac{\alpha_h}{\alpha_h+\beta_h}$$
$$\tau_n = \frac{1}{\alpha_n+\beta_n} \qquad \tau_m = \frac{1}{\alpha_m+\beta_m} \qquad \tau_h = \frac{1}{\alpha_h+\beta_h} \tag{7.178}$$

The constants and initial conditions for this simulation are:

$\bar{g}_K = 36$ mS/cm^2 $\quad \bar{g}_{Na} = 120$ mS/cm^2 $\quad \bar{g}_L = 0.3$ mS/cm^2

$v_K = $-12 mV $\quad v_{Na} = 115$ mV $\quad v_L = 10.6$ mV

$v(0) = 8$ mV $\quad n(0) = 0.3177$ $\quad m(0) = 0.0529$ $\quad h(0) = 0.5961$

The initial conditions for the four variables (v, n, m, and h) in Eq. (7.176) are chosen based on the following statement made by Hodgkin and Huxley:

> "By a membrane action potential is meant one in which the membrane potential is uniform, at each instant, over the whole of the length of the fibre considered. There is no current along the axis of the cylinder and the net membrane current must therefore always be zero, except during the stimulus. If the stimulus is a short shock at $t = 0$, the form of the action potential should be given by solving Eq. (7.176) with $I = 0$ and the initial conditions that $V=V_0$ and m, n, and h have their resting steady state values, when $t = 0$."

(a) First verify the values of the initial conditions, n(0), m(0), and h(0); they must be the resting steady state values of these variables (when v = 0). Integrate the differential equations for the time span of 0 to 20 ms, using an initial voltage of 8 mV. There is no current along the axis of the cylinder and the net membrane current must always be zero, therefore, use a current density of 0 μA/cm^2 and a membrane capacitance of 1 μF/cm^2. Plot the time profiles of the potential, v, the

gating variables, n, m, and h, and the conductances, g_K and g_{Na} (Eqs. (7.174) and (7.175)).

(b) Calculate and plot the steady state values of the time constants and the gating variables (Eqs. (7.178)) as functions of the potential in the range of voltages from –100 mV to +100 mV.

Solution

(a) **Integration of equations**

The program `example7_5.m`, listed below, first calculates the initial conditions of the gating variables using Eqs. (7.178), and then integrates the differential equations that are contained in the function `hodgkin_huxley_equations.m` using the MATLAB function `ode45.m`. The program also uses the function `rate_constants.m` to calculate the values of α and β. The same program also calculates the steady state values of the time constants and the gating variables and plots the results.

```
% example7_5.m - Simulation of the Hodgkin-Huxley model
% using MATLAB function ode45.m to integrate the differential
% equations that are contained in the file:
% hodgkin_huxley_equations.m

clc; clear all;
warning off MATLAB:divideByZero

% Evaluate the initial conditions for gating variables
v=0;
[alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=rate_constants(v);
tau_n=1./(alpha_n+beta_n);
n_ss=alpha_n.*tau_n;
tau_m=1./(alpha_m+beta_m);
m_ss=alpha_m.*tau_m;
tau_h=1./(alpha_h+beta_h);
h_ss=alpha_h.*tau_h;
fprintf('\n The following initial conditions of the gating
variables are used:')
fprintf('\n n_ss= %5.4g \n m_ss= %5.4g \n h_ss= %5.4g ',
n_ss,m_ss,h_ss)
fprintf('\n They are the resting steady state values of these
variables (when v=0).')

% Integrate the equations
yzero=[8,n_ss,m_ss,h_ss];
tspan=[0,20];
[t,y]=ode45(@hodgkin_huxley_equations,tspan,yzero);
% Evaluate the conductances
ggK=36; ggNa=120;
gK=ggK*y(:,2).^4; gNa=ggNa*y(:,3).^3.*y(:,4);

% Plot the results
```

```
clf; figure(1); plot(t,y(:,1),'k');
title('Figure E7.5a: Time Profile of Membrane Potential in Nerve
Cells')
xlabel('Time (ms)'); ylabel('Potential (mV)')
figure(2); plot(t,y(:,2:4));
title('Figure E7.5b: Time Profiles of Gating Variables')
xlabel('Time (ms)'); ylabel('Gating variables')
text(7,0.6,'\leftarrow n(t)'); text(4.5,0.9,'\leftarrow m(t)');
text(7,0.25,'\leftarrow h(t)')
figure(3); plot(t,gK,t,gNa);
title('Figure E7.5c: Time Profiles of Conductances')
xlabel('Time (ms)'); ylabel('Conductances')
text(7,6,'g _K'); text(3.6,25,'g _{Na}');

% Evaluate the rate constants
v=[-100:1:100];
[alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=rate_constants(v);

% Evaluating time constants and gating variables at steady state
tau_n=1./(alpha_n+beta_n);
n_ss=alpha_n.*tau_n;
tau_m=1./(alpha_m+beta_m);
m_ss=alpha_m.*tau_m;
tau_h=1./(alpha_h+beta_h);
h_ss=alpha_h.*tau_h;

% Plot the time constants
figure(4); plot(v,tau_n,v,tau_m,v,tau_h)
axis([-100 100 0 10])
title('Figure E7.5d: Time Constants as Functions of Potential')
xlabel('Potential (mV)'); ylabel('Time constants (ms)')

text(-75,4,'\tau _n'); text(0,0.8,'\tau _m'); text(15,8,'\tau
_h');

% Plot the gating variables at steady state
figure(5); plot(v,n_ss,v,m_ss,v,h_ss)
axis([-100 100 0 1])
title('Figure E7.5e: Gating Variables at Steady State as Functions
of Potential')
xlabel('Potential (mV)'); ylabel('Gating variables at steady
state')
text(-35,0.1,'n_\infty'); text(25,0.4,'m_\infty');
text(-20,0.8,'h_\infty');
```

Function that contains equations (`hodgkin_huxley_equations.m`)

```
function dy=hodgkin_huxley_equations(t,y)
% hodgkin_huxley_equations.m
% Contains the Hodgkin-Huxley model for example7_5

% Constants
ggK=36; ggNa=120; ggL=0.3;
```

```
vK=-12; vNa=115; vL=10.6;
Iapp=0;  Cm=1;
% Equations
v=y(1); n=y(2); m=y(3); h=y(4);
[alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h]=rate_constants(v);

dy=[(-ggK*n^4*(v-vK)-ggNa*m^3*h*(v-vNa)-ggL*(v-vL)+Iapp)/Cm
    alpha_n*(1-n)-beta_n*n
    alpha_m*(1-m)-beta_m*m
    alpha_h*(1-h)-beta_h*h];
```

Function that calculates the rate constants (`rate_constants.m`)

```
function [alpha_n,beta_n,alpha_m,beta_m,alpha_h,beta_h] =
 rate_constants(v)
% rate_constants.m
% Calculates the rate constants for the Hodgkin-Huxley model

alpha_n=0.01*(10-v)./(exp((10-v)/10)-1);
beta_n=0.125*exp(-v/80);
alpha_m=0.1*(25-v)./(exp((25-v)/10)-1);
beta_m=4*exp(-v/18);
alpha_h=0.07*exp(-v/20);
beta_h=1./(exp((30-v)/10)+1);
```

Results

```
The following initial conditions of the gating variables are used:
n_ss= 0.3177
m_ss= 0.05293
h_ss= 0.5961
They are the resting steady state values of these variables (when
v=0).
```

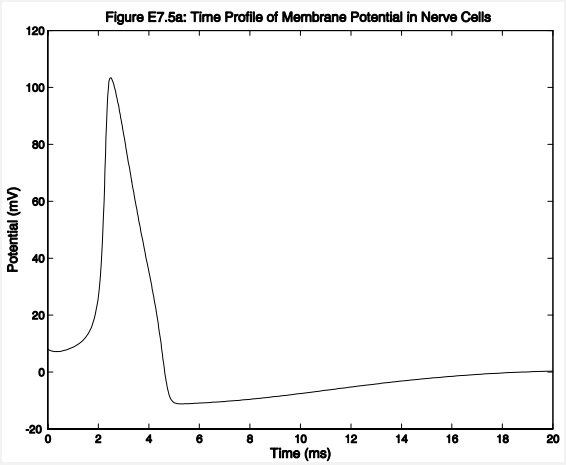


Figure E7.5b: Time Profiles of Gating Variables

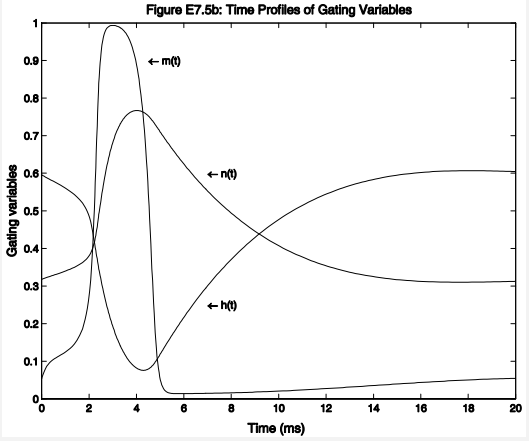


Figure E7.5c: Time Profiles of Conductances

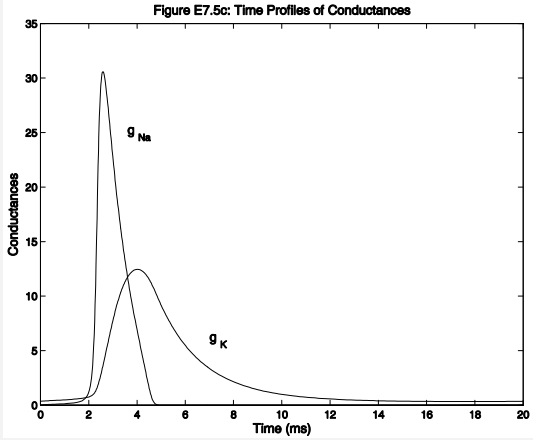
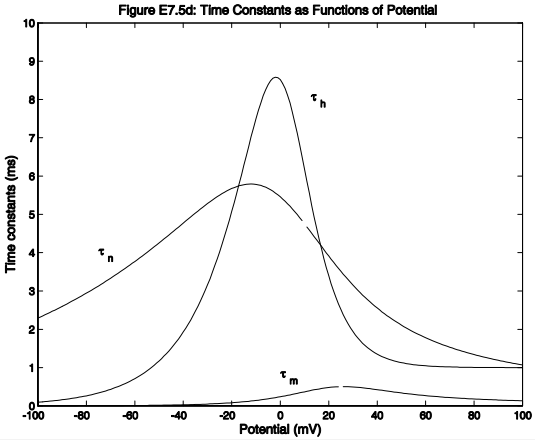
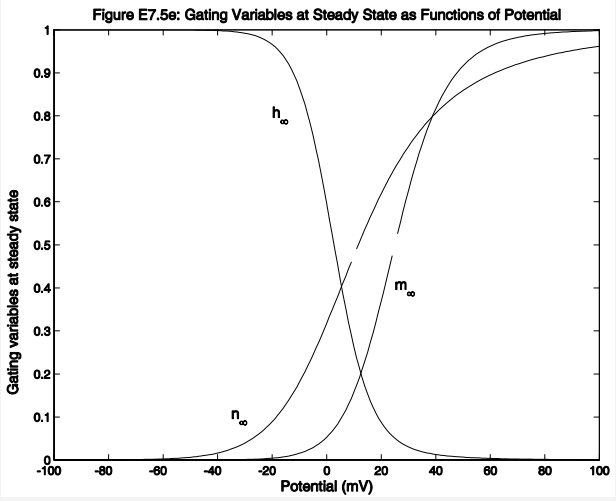


Figure E7.5d: Time Constants as Functions of Potential





Discussion of results

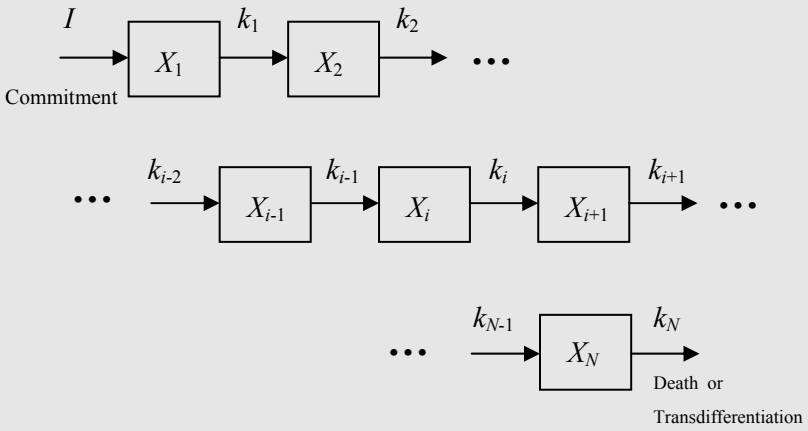
The application of a stimulus to the cell, in the form of a voltage of 8 mV at t = 0, raises the membrane potential above the threshold value and causes the generation of a self-propagating action potential, as shown in Fig. E7.5a. The membrane potential rises rapidly to over 100 mV and then drops back to its resting potential, all in a matter of less than 20 milliseconds. This action is explained as follows: the sodium gates have a much smaller time constant, τ_m, (Fig. E7.5d), therefore $m(t)$ responds faster, i.e. the sodium channels open faster allowing the flow of Na^+ into the cell, thus making the potential more positive. As the potential rises, the value of h_∞ goes to zero (Fig. E7.5e), thus causing the sodium current to inactivate because its conductance, g_{Na}, goes to zero (Fig. E7.5c). This mechanism, however, has a higher time constant, thus it is slower to show its effect. The voltage-gated potassium channels also open when the membrane potential becomes more positive than during the resting state; however, unlike the sodium channels, they open more slowly and become fully opened only after the sodium channels have closed. The potassium channels then remain open until the membrane potential has returned to near its resting value.

The student is encouraged to work out Problem 7.1 (at end of this chapter), which applies a constant current of 10 $\mu A/cm^2$, and to observe and interpret the results.

Example 7.6 The dynamics of stem cell differentiation.

Formulation of the problem

Stem cells in a growing fetus replicate and differentiate to develop into specialized types of cells, such as bone cells, skin cells, liver cells, muscle cells, etc. In an adult human body, the bone marrow contains stem cells, such as *hematopoietic* cells that generate red blood cells, and *mesenchymal* cells that produce connective tissue cells. The differentiation process involves a series of changes in cell phenotype and morphology that typically become more pronounced and easier to observe directly at the latter stages of the process (Palsson and Bhatia, 2004). This process begins with the stem cell's commitment to differentiation, followed by a coordinated series of gene-expression events, causing the cell to differentiate to a new state. A series of such progressive states leads to fully mature specialized cells. These mature cells perform their intended function in the body and eventually die, or undergo change to another type of cell through a process called *transdifferentiation*. The progressive series of events that converts a stem cell to a fully mature specialized cell may be depicted schematically as follows:



where X_i = number of cells in stage i of differentiation (cells)
I = number of cells entering the differentiation process (cells/day)
k_i = the transition rate of cells from stage i to stage i+1 (1/day)
N = the total number of stages of differentiation (may be as high as 16 to 18).

The final stage of the process, N, may be considered as the intended goal of the differentiation, i.e., the state of specialized mature cells. This last stage may have a zero transition rate constant. That is, if k_N is equal to zero, then cells do not die or transdifferentiate.

The dynamics of the differentiation process may be easily simulated using a multi-compartment model. Assuming that each stage is homogeneous in its cellular content, an unsteady state balance on each compartment yields the following set of ordinary differential equations:

$$\frac{dX_1}{dt} = I - k_1 X_1$$
$$\frac{dX_2}{dt} = k_1 X_1 - k_2 X_2$$
$$\vdots$$
$$\frac{dX_i}{dt} = k_{i-1} X_{i-1} - k_i X_i$$
$$\vdots$$
$$\frac{dX_N}{dt} = k_{N-1} X_{N-1} - k_N X_N$$

The above model reflects the transition of cells from one stage of differentiation to the next, with no cell division and no self-renewal. These two concepts are explored in Problems 7.9 and 7.10, at the end of this chapter.

Using the above differential equations, simulate numerically the following stem cell differentiation cases:

(a) Stem cells commit to the differentiation process at a continuous rate of I = 5000 cells/day. Assume that these cells undergo 10 stages of differentiation (N = 10). No death occurs at the last step in the process (k_N = 0). Integrate the differential equations and trace the path of these cells through the 10 stages of differentiation, using the following initial conditions and constants:

$$I = 5000 \text{ cells/day}$$
$$X_i(0) = 0 \text{ cells, for } i = 1,\ldots,N$$
$$k_i = 2.2 \text{ day}^{-1}, \text{ for } i = 1,\ldots,(N-1)$$
$$k_N = 0 \quad \text{no death or transdifferentiation}$$

Examine and discuss the time profiles. Does this case reach a steady state?

(b) There are no new cells entering the process, i.e., I = 0, but the initial number of cells in the first stage of differentiation, $X_1(0)$, is 5000. Assume that these cells undergo the same number of stages of differentiation as in case (a). No death occurs at the last stage of the process. Integrate the differential equations and trace the path of these cells through the 10 stages of differentiation, using the following initial

conditions and constants:

$$I = 0 \text{ cells/day}$$
$$X_1(0) = 5000 \text{ cells}, \quad X_i(0) = 0 \text{ cells, for } i = 2,\ldots,N$$
$$k_i = 2.2 \text{ day}^{-1}, \text{ for } i = 1,\ldots,(N-1)$$
$$k_N = 0 \quad \text{no death or transdifferentiation}$$

Examine and discuss the time profiles. How many days does it take for the completion of this process?

(c) This is the same as case (a), except for the occurrence of death at the completion of stage 10. Examine the time profiles and predict the steady state behavior of this system using the following initial conditions and constants:

$$I = 5000 \text{ cells/day}$$
$$X_i(0) = 0 \text{ cells, for } i = 1,\ldots,N$$
$$k_i = 2.2 \text{ day}^{-1}, \text{ for } i = 1,\ldots,N, \text{ with death}$$

Solution

(c) The MATLAB program and function that solve all three cases are listed below:

```
% example7_6.m - Solution of the stem cell differentiation model
% using MATLAB function ode45.m to integrate the differential
% equations that are contained in the file:
% cell_differentiation_equations.m
clc; clear all;
% Set the number of stages & time span
N=10; tzero=0; tmax=10; tspan=[tzero:0.1:tmax];
% Case (a): With continuous input; no death
I=5000;                      % Input
Xzero=zeros(N,1);            % Initial conditions
k=2.2*ones(N-1,1); k(N)=0;  % Transiton rate constants, no death
% Integrate the equations
[t,X]=ode45('cell_differentiation_equations',tspan,Xzero,[],N,I,k);
% Pseudo steady state values for stages 1 to N-1
SS=I/k(1); X_last=X(length(X),N);
disp('Case (a)')
fprintf('The pseudo steady state number of cells in stages %1d to%2d
= %4.0f',1,N-1,SS)
fprintf('\nThe number of cells in stage %2d, at %2d days = %4.0f
\n',N,tmax,X_last)
% Plot concentration profiles
clf; figure(1); subplot(2,1,1), plot(t,X(:,1:1:N-1))
title(['Figure E7.6 (a):  Continuous input (I = ',num2str(I),...
```

```
        '); no death (k(1:', num2str(N-1),') = ',num2str(k(1)),...
        ', k(', num2str(N),') = ',num2str(k(N)),') '])
text(0.4,SS,'i = 1'); text(0.45*tmax,SS/2,['i = ', num2str(N-1)]);
xlabel('Time, days'); ylabel('Number of cells');
subplot(2,1,2), plot(t,X(:,N)/1000)
axis([tzero, tmax, 0, 1.1*X_last/1000])
text(tmax/2,X_last/2000,['i = ', num2str(N)]);
xlabel('Time, days'); ylabel('Number of cells (thousands)');
% Case (b): With no new input; no death
I=0;                              % Input
Xzero=zeros(N,1); Xzero(1)=5000; % Initial conditions
% Integrate the equations
[t,X]=ode45('cell_differentiation_equations',tspan,Xzero,[],N,I,k);
% Steady state values for stages 1 to N-1
SS=I/k(1);
X_last=X(length(X),N);
disp('Case (b)')
fprintf('The steady state number of cells in stages %1d to%2d =
%4.0f',1,N-1,SS)
fprintf('\nThe final number of cells in stage %2d at %2d days = %4.0f
\n',N,tmax,X_last)
% Plot concentration profiles
figure(2); plot(t,X(:,1:1:N))
title(['Figure E7.6 (b):  No new input (I = ', num2str(I),...
        '); no death (k(1:', num2str(N-1),') = ', num2str(k(1)),...
        ', k(', num2str(N),') = ', num2str(k(N)),')'])
text(0.4,0.8*X(1),'i = 1');
text(0.45*tmax,X(1)/2,['i = ',num2str(N)]);
xlabel('Time, days'); ylabel('Number of cells');
% Case (c): With continuous input; with death (or transdiff.)
I=5000;             % Input
Xzero=zeros(N,1);   % Initial conditions
k(N)=k(1);          % reset the death rate constant
% Transiton rate constants, with death
% Integrate the equations
[t,X]=ode45('cell_differentiation_equations',tspan,Xzero,[],N,I,k);
% Pseudo steady state values for stages 1 to N-1
SS=I/k(1);
X_last=X(length(X),N);
disp('Case (c)')
fprintf('The steady state number of cells in all stages = %4.0f',SS)
% Plot concentration profiles
figure(3); plot(t,X(:,1:1:N))
axis([tzero, tmax, 0, 1.1*I/k(1)])
title(['Figure E7.6 (c):  Continuous input (I = ',...
        num2str(I), '); with death (k(1:', num2str(N),...
        ') = ',num2str(k(1)),')'])
text(0.4,SS,'i = 1'); text(0.5*tmax,SS/2,['i = ', num2str(N)]);
xlabel('Time, days'); ylabel('Number of cells');
```

Function that contains the equations (`cell_differentiation_equations.m`)

```
function dX=cell_differentiation_equations(t,X,flag,N,I,k)
% cell_differentiation_equations.m
% Contains the equations for example7_6

% Equations
dX(1)=I-k(1)*X(1);
for i=2:N
    dX(i)=k(i-1)*X(i-1)-k(i)*X(i);
end
% Convert to column vector
dX=dX';
```

Case (a) Results and discussion

The results of this case are plotted on Fig. E7.6 (a). The top half of the plot shows the time profiles for stages 1 to 9. The constant input of cells into stage one (I = 5000 cells/day) causes the first 9 stages of differentiation to reach a steady state in less than 10 days, with the number of cells in each stage given by

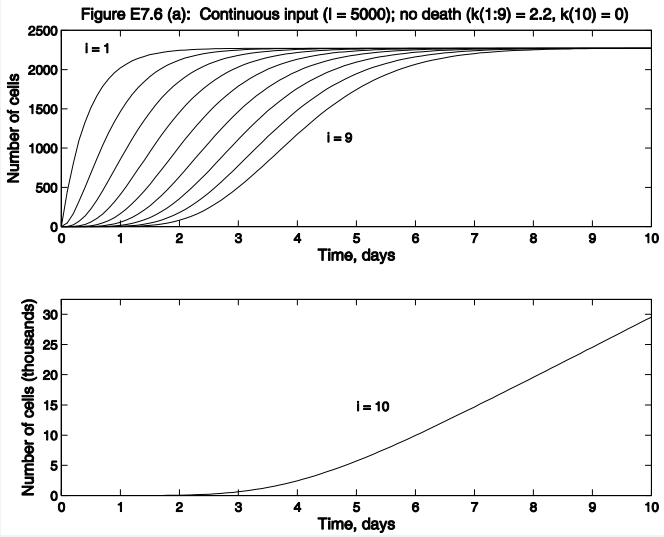
$$X_i^* = \frac{I}{k_i} = \frac{5000 \text{ cells/day}}{2.2 / day} = 2273 \text{ cells}$$

This result is obtained mathematically by setting the derivatives of the first nine differential equations to zero (steady state) and solving for X_i^* (the steady state level of the cells). However, the differential equation for the final stage does not have a steady state because of the no death assumption (k_{10} = 0). Setting $\frac{dX_{10}}{dt} = 0$ yields $I = 0,,$ which we know is incorrect. For this reason, we call this case a *pseudo* steady state. The number of cells in the final stage is 29,547 in 10 days and continues to increase, as shown in the bottom half of Fig. E7.6 (a). This final stage of the process is the intended goal of the differentiation; therefore it is reasonable to expect that cells will continue to accumulate in this stage.

```
Case (a)
The pseudo steady state number of cells in stages 1 to 9 = 2273
The number of cells in stage 10, at 10 days = 29547
```

Case (b) Results and discussion

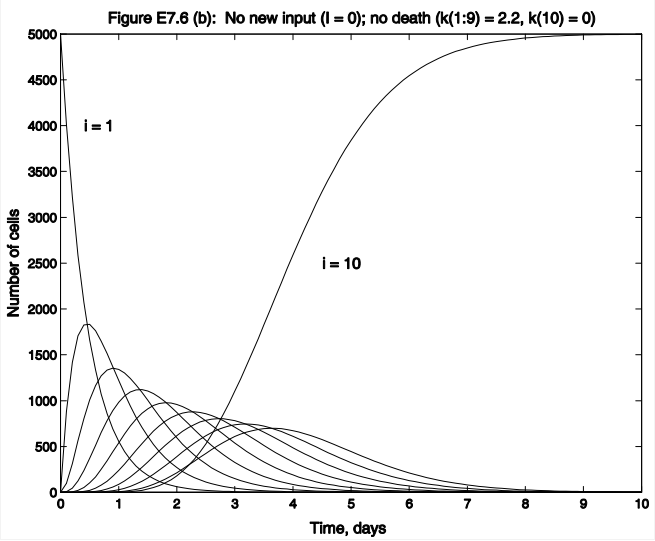
The results of this case are plotted on Fig. E7.6 (b). There is no input of new cells and no death occurs in the last stage of differentiation. Therefore, the cells differentiate



completely from one stage to the next, without any renewal from new cells entering, and finally accumulate in the last compartment of the process, as is clearly shown by Fig. E7.6 (b). Since $I = 0$, the steady states for stages 1 to 9 are all zero, i.e.,

$$X_i^* = \frac{I}{k_i} = \frac{0 \text{ cells/day}}{2.2 / day} = 0 \text{ cells}$$

The final number of cells in stage 10 is ~5000, as expected, remembering that the initial number of cells was 5000, and there is no death of cells anywhere in this pathway.



```
Case (b)
The steady state number of cells in stages 1 to 9 =    0
The final number of cells in stage 10 at 10 days = 4997
```

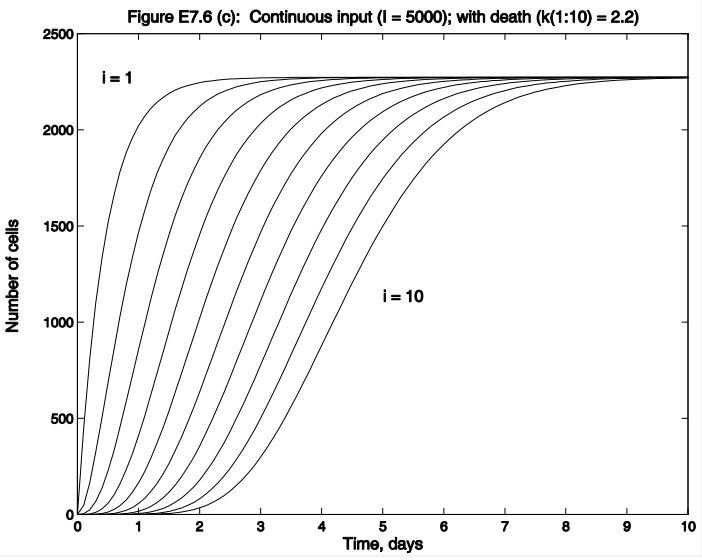
Case (c) Results and discussion

In this case, there is a continuous rate of new stem cells, I = 5000 cells/day, that commit to the differentiation process. There is also the occurrence of death at the completion of stage 10. The results of this case are plotted on Fig. E7.6 (c). Under these circumstances, all 10 stages reach their steady states at

$$X_i^* = \frac{I}{k_i} = \frac{5000 \text{ cells/day}}{2.2 / day} = 2273 \text{ cells}$$

The cells continue to differentiate from one stage to the next, with death occurring after the last stage. It is theoretically possible that this cell differentiation process may continue for the duration of the lifetime of the individual.

```
Case (c)
The steady state number of cells in all stages = 2273
```



Example 7.7 Tissue engineering: models of epidermal cell migration.

Introduction

One aspect of tissue engineering is the proper design and manufacture of porous matrices (membranes) that imitate the properties of the epidermis and may be used as prosthetic scaffolding to promote dermal regeneration, thus enhancing the healing process of wounded or burned skin. During the healing process, cell migration is necessary for cells to repopulate a healing wound, and to imbed themselves in an implanted scaffold for successful tissue regeneration. Cellular migration is known to depend on the interaction of specific cell surface receptors with cell-internalizable ligands that are present on the extracellular matrix. The formation of ligand-receptor bonds between skin epidermal cells (keratinocytes) and ligand presenting microcarriers may initiate and promote the process of endocytosis – the ingestion of molecules by the cells – thus, significantly enhancing the levels of cell motility.

The dynamics of cell-ligand interactions and endocytically-coupled cell motility have been modeled from a kinetic-mechanistic point of view (Tjia and Moghe, 2002c) using diffusion-reaction descriptions and equations similar to those in the traditional Michaelis-Menten kinetics.

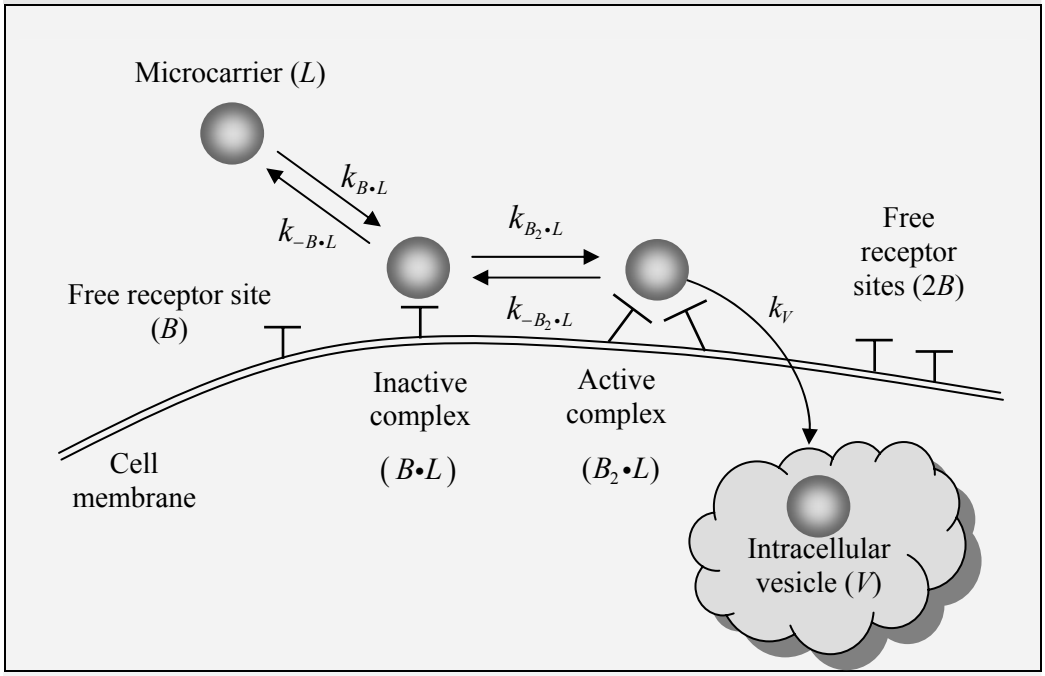


Figure E7.7 Cell-ligand interactions.

Formulation of the mechanism

Fig. 7.7 shows the mechanism of cell-ligand interactions schematically. The individual steps of this process are described below:

1. A ligand-adsorbed microcarrier, (L), interacts with a free receptor site, (B), on the surface of a cell to form an inactive complex, $(B \bullet L)$:

$$L + B \underset{k_{-B \bullet L}}{\overset{k_{B \bullet L}}{\rightleftarrows}} B \bullet L \tag{7.179}$$

2. The inactive complex, in turn, binds reversibly with a second receptor to form the active complex $(B_2 \bullet L)$:

$$B \bullet L + B \underset{k_{-B_2 \bullet L}}{\overset{k_{B_2 \bullet L}}{\rightleftarrows}} B_2 \bullet L \tag{7.180}$$

3. The active complex is ingested by the cell to produce an intracellular vesicle, (V). Once ingested, the microcarrier dissociates itself from the membrane receptors thus freeing the receptors to recycle back to the cell surface. For the purposes of this model, the rates of ingestion and binding site recycling are lumped into one parameter, k_V:

$$B_2 \bullet L \xrightarrow{k_V} V + 2B \tag{7.181}$$

Formulation of the mathematical model

Cell migration will affect the degree of exposure of microcarriers to the cell, as migration would make new microcarriers available for internalization. The rate of cell migration has been derived, based on an analogy to molecular diffusion in a semi-infinite plane, to be

$$\left.\frac{d[L]}{dt}\right|_{\text{Migration}} = \frac{\mu L_0}{A_{\text{cell}}} \tag{7.182}$$

where L = effective ligand density encountered by the cell
μ = the random motility coefficient,
A_{cell} = the spread area of the cell,
L_0 = the overall density of microcarriers.

The composite model that includes both cell migration and ligand-receptor adhesion is given below:

1. The density of local extracellular microcarriers encountered by the cell, [L],

changes according to the following rate equation:

$$\frac{d[L]}{dt} = -k_{B\bullet L}[L][B] + k_{-B\bullet L}[B\bullet L] + \frac{\mu L_0}{A_{\text{cell}}} \tag{7.183}$$

The first term in this equation corresponds to the forward rate in reaction (7.179), the second term corresponds to the reverse rate, and the third term reflects the rate of the cell migration given by Eq. (7.182).

2. The balance of the density of inactive microcarrier-receptor complex, $[B\bullet L]$, gives the rate equation (7.184):

$$\frac{d[B\bullet L]}{dt} = k_{B\bullet L}[L][B] - k_{-B\bullet L}[B\bullet L] - k_{B_2\bullet L}[B\bullet L][B] + k_{-B_2\bullet L}[B_2\bullet L] \tag{7.184}$$

3. The rate of change of the density of activated microcarrier-receptor complex, $[B_2\bullet L]$, is

$$\frac{d[B_2\bullet L]}{dt} = k_{B_2\bullet L}[B\bullet L][B] - k_{-B_2\bullet L}[B_2\bullet L] - k_V[B_2\bullet L] \tag{7.185}$$

4. The density of ingested microcarrier, $[V]$, changes at the following rate:

$$\frac{d[V]}{dt} = k_V[B_2\bullet L] \tag{7.186}$$

5. The total number of binding sites on the cell, $[B_T]$, is assumed to be constant

$$[B_T] = [B] + [B\bullet L] + [B_2\bullet L] \tag{7.187}$$

The net effects of cell migration may be measured in terms of the rate at which cells effectively clear an area covered with ingestible microcarriers. For a given initial surface particle density, the rate of area clearance by a cell is equal to the sum of the rates of bounding and ingesting, divided by the initial particle density:

$$\frac{d[clearance]}{dt} = \frac{1}{L_o}\left(\frac{d[B\bullet L]}{dt} + \frac{d[B_2\bullet L]}{dt} + \frac{d[V]}{dt}\right) \tag{7.188}$$

It should be noted that that the density terms in this model are in units of particle per surface area of cell. The $[clearance]$ term is in $(\text{min})^{-1}$.

In order to simplify the model, we make the following assumptions:

(a) The rate constant of decomposition of the inactivated microcarrier-receptor complex , $k_{-B\bullet L}$, is very small in comparison to the rate of initial binding, $k_{B\bullet L}$, and can be neglected. This effectively makes reaction (7.179) irreversible.

(b) The fully activated complex, $[B_2\bullet L]$, once formed, is highly reactive and is quickly ingested. Thus, this complex would be present only in low densities and may be assumed to be at pseudosteady-state. This assumption causes the rate in Eq. (7.185) to be equal to zero, enabling us to solve for $[B\bullet L]$, as follows:

$$[B_2L] = \frac{[B\bullet L][B]}{K_m} \tag{7.189}$$

where K_m is a dissociation constant of the Michaelis-Menten type, defined as:

$$K_m = \frac{k_{-B_2\bullet L} + k_V}{k_{B_2\bullet L}} \tag{7.190}$$

The above two assumptions are utilized in Eqs. (7.183)-(7.188) to eliminate $[B\bullet L]$ and $[B]$, which simplifies the model to the following set of equations:

$$\begin{aligned}
\frac{d[L]}{dt} &= -k_{B\bullet L}[L]([B_T]-[B\bullet L]) + \frac{\mu L_0}{A_{\text{cell}}} \\
\frac{d[B\bullet L]}{dt} &= k_{B\bullet L}[L]([B_T]-[B\bullet L]) - k_V\left(\frac{([B_T]-[B\bullet L])[B\bullet L]}{K_m + 2[B\bullet L]}\right) \\
\frac{d[V]}{dt} &= k_V\left(\frac{([B_T]-[B\bullet L])[B\bullet L]}{K_m + 2[B\bullet L]}\right) \\
\frac{d[clearance]}{dt} &= \frac{1}{L_o}\left(\frac{d[B\bullet L]}{dt} + \frac{d[V]}{dt}\right)
\end{aligned} \tag{7.191}$$

Equations (7.191) define the mathematical model of the dynamics of cellular migration enhanced by the presence of ligand-associated microcarriers. This is a set of ordinary differential equations that may be integrated to yield the temporal behavior of this process. For this problem, perform the following tasks:

(a) Evaluate and plot the time profiles, and discuss the results of the integration for the period of 300 minutes, using the following initial conditions and

constants, based on the experiment work of Tjia and Moghe (2002c):

Initial conditions:

$$L_0 = 1.0 \text{ particle}/\mu\text{m}^2 \qquad [B{\bullet}L]\big|_0 = 0$$
$$[clearance]\big|_0 = 0 \qquad [V]\big|_0 = 0$$

Constants:

$$B_T = 3.74 \text{ particles}/\mu\text{m}^2 \qquad \mu = 10\ \mu\text{m}^2/\text{min} \qquad A_{\text{cell}} = 3400\ \mu\text{m}^2$$
$$K_m = 0.73 \text{ particles}/\mu\text{m}^2 \qquad k_V = 1.3\times10^{-3}\ \text{min}^{-1}$$
$$k_{B{\bullet}L} = 2.0\times10^{-3}\ \mu\text{m}^2/(\text{particle.min})$$

(b) Define the term "Sampling rate" as

$$\text{(Sampling rate)} = \frac{d[B{\bullet}L]}{dt} - \frac{d[V]}{dt}$$

and show its effect on the internalization rate, d[V]/dt, and the clearance rate, d[clearance]/dt.

Solution

The program, `example7_7.m`, and the function, `cell_migration_equations.m`, that solve this problem are listed below:

```
% example7_7.m - Solution of the epidermal cell migration
% model using MATLAB function ode45.m to integrate the
% differential equations that are contained in the file:
% cell_migration_equations.m

clc; clear all;
% Set the time span
tspan=[0:1:300];
% Set the constants
BT=3.74; mu=10; A_cell=3400;
Km=0.73; kV=1.3e-3; kBL=2.0e-3;
% Set the initial conditions
yzero=[1, 0, 0, 0];
L0=yzero(1);
% Integrate the equations
[t,y]=ode45('cell_migration_equations',tspan,yzero,[],...
    BT,mu,A_cell,Km,kV,kBL,L0);

% Plot concentration profiles
```

```
figure(1); plot(t,y(:,1),'-',t,y(:,2),':',t,y(:,3),'-.',...
    t,y(:,4),'--')
title('Figure E7.7(a): Time profiles of epidermal cell migration')
xlabel('Time, min'); ylabel('Densities, number/\mum^2');
legend('L','B{\bf\cdot}L','V','clearance',2)
n=length(y);

% Evaluate the derivatives
for i=1:n
dy(:,i)=feval('cell_migration_equations',t(i),y(i,:),flag,...
    BT,mu,A_cell,Km,kV,kBL,L0);
end
dy=dy';
rate_BL=dy(:,2);
rate_V=dy(:,3);
clearance_rate=dy(:,4);
sampling_rate=rate_BL-rate_V;

% Show the effect of microcarrier sampling rate on internalization
% and clearance rates
figure(2);
plot(sampling_rate*1e3,clearance_rate*1e3)
title('Figure E7.7(b): The effect of microcarrier sampling rate on
clearance rate')
ylabel('Clearance rate, d[clearance]/dt  x  10^3')
xlabel('Sampling rate, (d[B{\bf\cdot}L]/dt - d[V]/dt)  x  10^3')

function dy=cell_migration_equations(t,y,flag,BT,mu,A_cell,...
    Km,kV,kBL,L0)
% cell_migration_equations.m
% Contains the equations for example7_7

% Equations
dy=[-kBL*y(1)*(BT-y(2))+mu*L0/A_cell
     kBL*y(1)*(BT-y(2))-kV*((BT-y(2))*y(2))/(Km+2*y(2))
     kV*((BT-y(2))*y(2))/(Km+2*y(2))
     (kBL*y(1)*(BT-y(2))-kV*((BT-y(2))*y(2))/(Km+2*y(2))...
      +kV*((BT-y(2))*y(2))/(Km+2*y(2)))/L0];
```

Discussion

In Fig. E7.7(a), the dynamics of ligand interactions with skin epidermal cells are plotted as a function of time. The free ligand concentration decreased slowly over time indicating the steady depletion of instantaneous ligand concentration due to cell internalization of the ligand. Concurrently, the concentrations of membrane-bound ligand complex (B.L) as well as internalized ligand (V) increase over time. The instantaneous clearance of ligands increases over time as well, indicating the cells are not yet saturated with ligand-microcarriers.

In Fig. E7.7(b), the cell clearance rate is graphed versus the net rate of ligand sampling by the cells (defined as the on-rate of ligand binding minus the off-rate of ligand internalization). It is assumed that internalized ligands can no longer activate the intracellular signaling necessary for increased cell migration. A monotonic increase in cell clearance rate of the ligands was observed with increased ligand sampling rate. This suggests that the migration may be a strong function of the dynamics of ligand sampling processes.

Results

Figure E7.7(a): Time profiles of epidermal cell migration

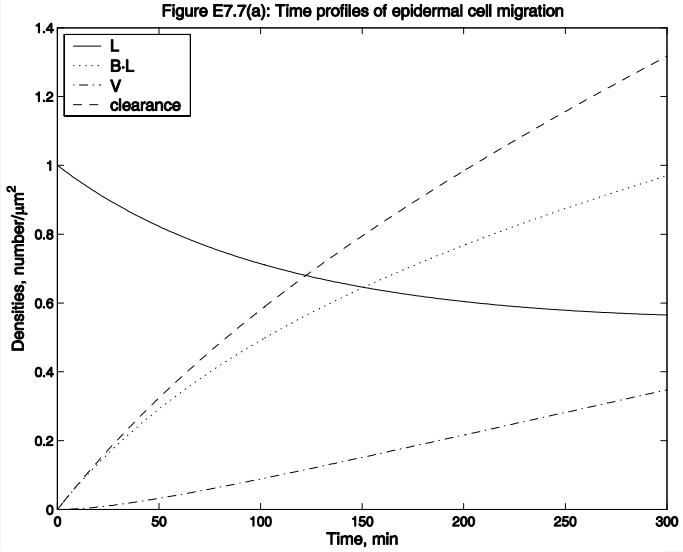
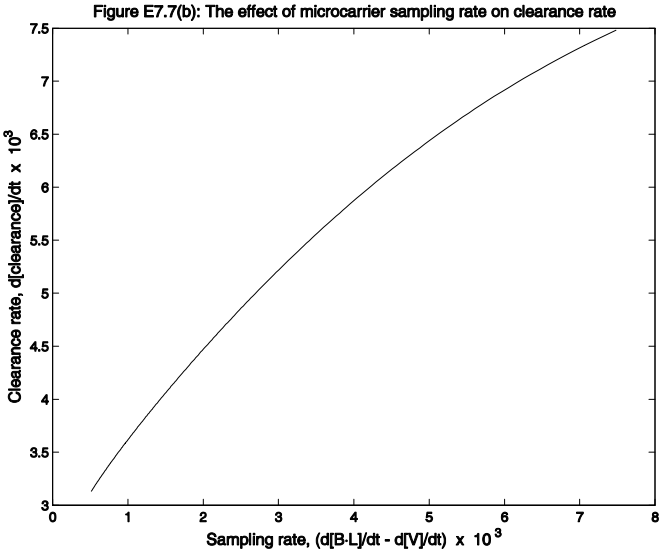


Figure E7.7(b): The effect of microcarrier sampling rate on clearance rate



7.11 Lessons Learned in this Chapter

After studying this chapter, the student should have learned the following:

- The dynamics of physiological systems may be modeled using ordinary differential equations.
- Ordinary differential equations may be classified as:
  - First, second, third order, etc.
  - Linear or nonlinear
  - Homogeneous or nonhomogeneous
  - Autonomous or non-autonomous
  - Initial value or boundary value
- Second order and higher ordinary differential equations may be converted to sets of first order differential equations for numerical integration by the methods discussed in this chapter.
- The solution of linear ordinary differential equations depends on the eigenvalues and eigenvectors of the equations.
- Nonlinear differential equations (as well as linear ones) may be integrated numerically using methods that are based on finite differences.
- Integrating differential equations is like climbing a mountain: You move in the direction of the slope (or the weighted average of the slope at different points), taking many small steps (carefully), until you reach the destination.
- The stability of nonlinear differential equations depends on the eigenvalues of the Jacobian matrix of the equations.
- The stability of the numerical solution depends on the form of the equations, the method of solution, and the step size of integration.

7.12 Problems

7.1 Integrate the Hodgkin-Huxley model (see Example 7.5) for the period 0 to 50 ms using a constant current of 10 $\mu A/cm^2$. Examine and explain the results thoroughly.

7.2 The pool of fluid in the body of a patient undergoing dialysis has been modeled by Enderle et al. (2000) as a two-compartment system, as shown diagrammatically on Figure P7.2, where R is the rate of production of urea by the patient's body, V_1 is the volume of the intracellular fluid, V_2 is the volume of the extracellular fluid (blood and interstitial fluids), C_1 and C_2 are the concentrations of urea in the fluids of the two compartments, respectively, k_{12} and k_{21} are the mass transport parameters between the two compartments, and k_2 is the clearance rate constant for the dialysis unit.

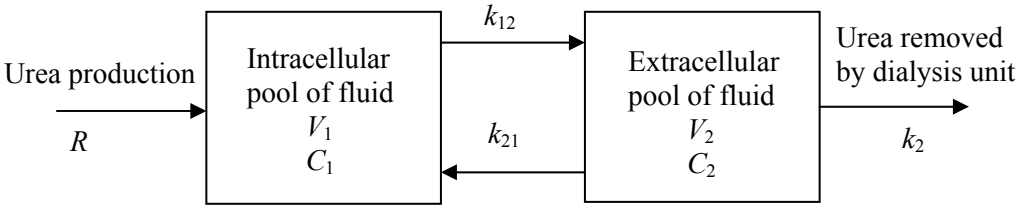


Figure P7.2 A two-compartment model of the fluid of a patient undergoing dialysis.

An unsteady state mass balance of urea on each of the compartments yields the following two differential equations:

$$V_1 \frac{dC_1}{dt} = R - k_{12}C_1 + k_{21}C_2$$
$$V_2 \frac{dC_2}{dt} = k_{12}C_1 - k_{21}C_2 - k_2 C_2 \qquad \text{(P7.2)-(1)}$$

For Patient X, the following parameters apply:

$R = 100$ mg/h $\quad k_{12} = 33$ liters/h $\quad k_{21} = 33$ liters/h
$V_1 = 10$ liters $\quad V_2 = 25$ liters

The dialysis unit clearance rate constant is $k_2 = 8$ liters/h.

When Patient X arrives at the dialysis unit his blood urea nitrogen (BUN) is 150 mg/liter. Integrate the differential equations (P7.2)-(1) to obtain answers to the following:

(a) How many hours of dialysis will the patient require in order to reduce the level of BUN to 75 mg/liter?
(b) After the completion of the treatment, how long will it take for the BUN of the patient to rise back to the 150 mg/liter level?
(c) Experiment with setting the values of k_{12} and k_{21} to be unequal to each other (say $k_{21} = 0.7\ k_{12}$, i.e., slower transfer from the extracellular pool to the intracellular one) and interpret the results.

Show clearly how you obtain your answers and illustrate this by showing the concentrations vs. time profiles of C_1 and C_2 in all parts of the problem.

7.3 A computer simulation of the physiological human knee jerk reflex has been developed by Huang (1994). A strong tap on the patellar ligament of the leg elicits a knee jerk reflex, which follows closely the oscillations of the pendulum. The jerk of the patellar tendon stretches the muscle that sends a barrage of neural impulses to the spinal cord. The reflex signal passes back to the quadriceps muscle via the alpha motor neuron to produce a sudden contraction and forces the leg to move forward with a jerk. As the muscle relaxes, the leg system acts as a damped compound pendulum, swinging back and forth for a few oscillations. Eventually the leg returns to the normal position.

In his analysis, Huang assumed that the extensor and flexor muscles are identical and opposite in action. The numbers of primary and secondary nerve endings are considered equal, and the nervous signals are instantaneous when compared to the system response. Small deflection angles are considered with constant damping coefficient within the range. Based on the equation of the pendulum, and for small oscillations, Huang developed a second order differential equation

$$J\frac{d^2\theta}{dt^2} + c\frac{d\theta}{dt} + \left(\frac{mgL\theta}{2} - T\right)\theta = 0 \qquad \text{(P7.3)-(1)}$$

that describes the angular position, θ, of the leg during the knee jerk reflex, where m is the mass of the leg, g is the gravitational acceleration constant, L is the length of the leg, J is the moment of inertia of the leg, and T is the gain produced by the isometric torque of the muscle. The natural frequency, ω_n, of the system is calculated by

$$\omega_n = \sqrt{\frac{mgL}{2J} - \frac{T}{J}} \qquad \text{(P7.3)-(2)}$$

and the damping factor, α, is given by

$$\alpha = \frac{c}{2\sqrt{J(\frac{mgL}{2} - T)}} \qquad \text{(P7.3)-(3)}$$

The values of ω_n and α are obtained experimentally. Solve the above equations with the following values of m, g, L, J, ω_n and α, and plot the time profile of the angular position of the leg during the jerk reflex.

$m = 4$ kg	$g = 9.81$ m/s^2	$L = 0.34$ m
$J = 0.154$ kg.m^2	$\omega_n = 6.28$ rad/s	$\alpha = 0.228$

Use the following initial conditions $\theta(0) = 0$ rad and $d\theta(0)/dt = 2\pi$ rad/s for the solution of the differential equation (P7.3)-(1). HINTS: Use the MATLAB `solve` command for the algebraic equations and the `dsolve` command for the differential equation.

7.4 The pool of fluid in the body of a patient undergoing dialysis was modeled in Problem 7.2 (above) as a two-compartment system. Change this analysis to a one-compartment model by treating the total fluid of the patient as one unit of volume V_T (see Fig. P7.4).

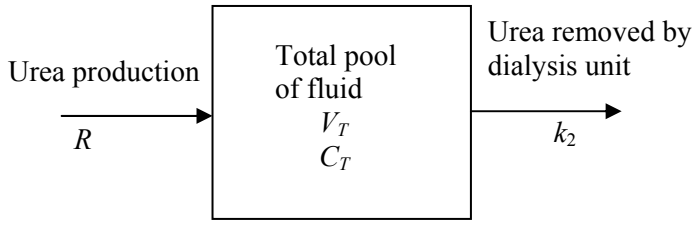


Figure P7.4 A one-compartment model of the fluid of a patient undergoing dialysis.

Derive the unsteady state mass balance of urea for this one-compartment model. For Patient X, the following parameters apply for the one-compartment analysis:

$$R = 100 \text{ mg/h} \quad V_T = 35 \text{ liters} \quad k_2 = 8 \text{ liters/h}$$

When Patient X arrives at the dialysis unit his blood urea nitrogen (BUN) is 150 mg/liter. Integrate the differential equation to obtain answers to the following:

(a) How many hours of dialysis will the patient require in order to reduce the level of BUN to 75 mg/liter?
(b) After the completion of the treatment, how long will it take for the BUN of the patient to rise back to the 150 mg/liter level?

Compare these results with those of the two-compartment model (Problem 7.2).

7.5 A simple model of an epidemic is shown on Fig. P7.5, where S is the number of persons susceptible to the disease, I is the number infected with it, R is the number that have already been affected but have recovered (or died), α is the rate constant for infection, and β is the rate constant of recovery. Those who have recovered develop immunity to the infection.

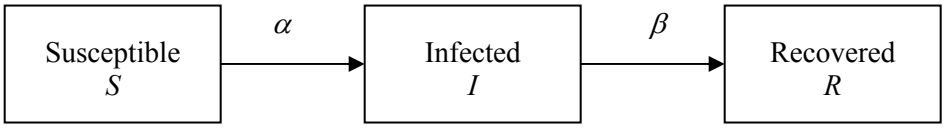


Figure P7.5 A simple model of an epidemic.

A dynamic model of the interactions between these three groups is given by Edelstein-Keshet (1988), as follows:

$$\begin{aligned} \frac{dS}{dt} &= -\alpha SI \\ \frac{dI}{dt} &= \alpha SI - \beta I \\ \frac{dR}{dt} &= \beta I \end{aligned} \qquad \text{(P7.5)-(1)}$$

One person, highly contagious with a new influenza virus, enters a small community that has a population of 5000 individuals that are susceptible to the infection. The virus epidemic spreads quickly and eventually infects all susceptible individuals. The rate constants for this epidemic are

$$\begin{aligned} \alpha &= 0.005 \quad (\text{person})^{-1}(\text{week})^{-1} \\ \beta &= 1 \qquad\quad (\text{week})^{-1} \end{aligned}$$

Integrate the differential equations (P7.5)-(1) and determine the following:

(a) How many weeks does it take for this epidemic to reach its peak?
(b) What is the maximum number of persons sick at the peak of the epidemic?
(c) In how many weeks will the epidemic subside, (when less the 0.5 % of the susceptible population is still infected)?

7.6 Modify the epidemic model in Problem 7.5 to allow loss of immunity that causes recovered individuals to become susceptible to the virus again (see Fig. P7.6). The

loss of immunity rate constant has the following value (α and β remain the same as in Problem 7.5):

$$\gamma = 0.1 \qquad (\text{week})^{-1}$$

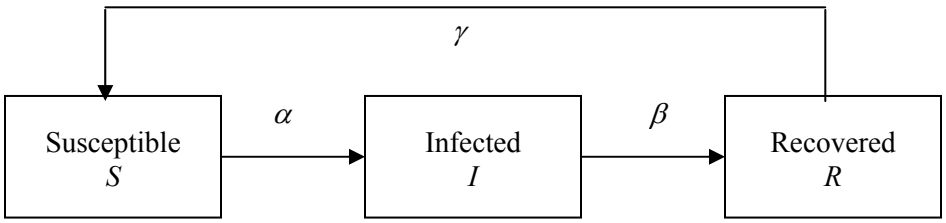


Figure P7.6 A modified model of an epidemic to account for loss of immunity.

Integrate the modified set of differential equations for this epidemic and determine the following:

(a) How many weeks does it take for the epidemic to approach steady state?
(b) How many people will remain infected during steady state?
(c) Show phase plots and discuss the stability of the solutions with respect to the eigenvalues of the Jacobian matrix.

7.7 The well-known van der Pol oscillator is the second order nonlinear differential equation shown below:

$$\frac{d^2u}{dt^2} - k\left(1-u^2\right)\frac{du}{dt} + au = 0 \qquad \text{(P7.7)-(1)}$$

The solution of this equation exhibits stable oscillatory behavior. Van der Pol realized the parallel between the oscillations generated by this equation and certain biological rhythms, such as the heart beat, and proposed this as a model of an oscillatory cardiac pacemaker. Integrate the van der Pol equation with the following value of k and initial conditions

$$k = 1.0 \text{ s}^{-1} \qquad u(0) = 2 \text{ dimensionless} \qquad \left.\frac{du}{dt}\right|_0 = 0 \text{ s}^{-1}$$

and determine the value of *a* that would give a heart rate of 1.25 beats/second (75 beats/minute, which is a typical heart rate in a resting adult).

7.8 It is well known that most living cells – bacteria cells, stem cells, yeasts, etc. – replicate themselves by cell division. The growth of an organism is accompanied by an orderly increase in its mass and all of its chemical constituents, followed by

division of the cell into two identical daughter cells or a mother and daughter cell, as in the case of yeasts. In Example 7.6, we simulated the process of stem cell differentiation without cell division. That was a rather simplistic model of cell differentiation, because most stages of differentiation have cell replication activity. The act of replication marks the completion of one stage of differentiation and the beginning of the next.

The human body produces and consumes approximately 200 billion red blood cells daily. The process of turning a bone marrow stem cell to a red blood cell is called erythropoiesis. The differentiation from the early precursor stage (pronormoblast) to a fully mature enucleated erythrocyte takes approximately one week (Palsson and Bhatia, 2004). This concept is depicted diagrammatically in Fig. P7.9.

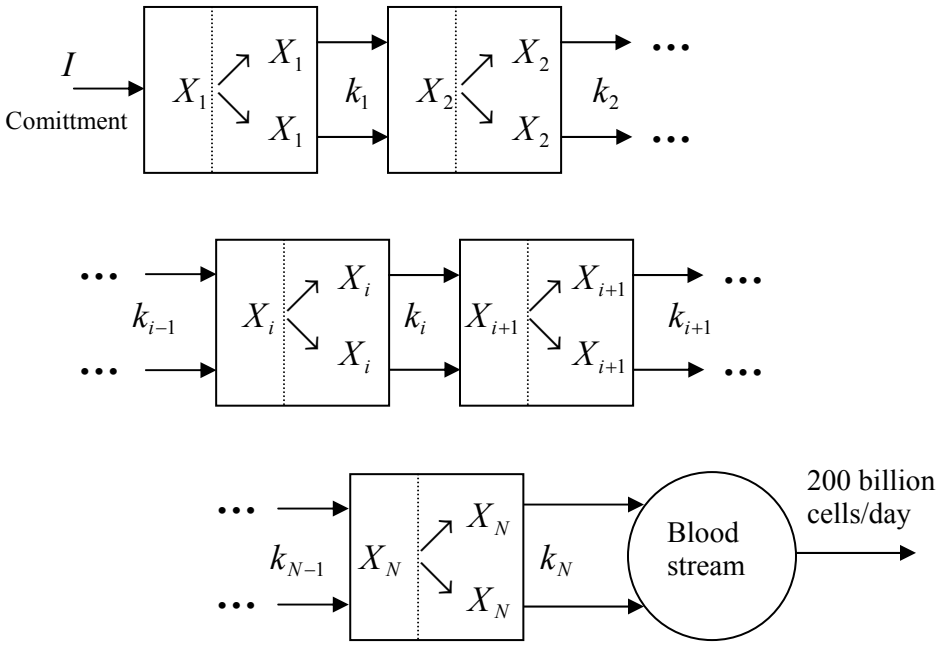


Figure P7.8 Stem cell differentiation with replication.

With the assumptions that a single cell that leaves compartment (i-1) splits into two cells that enter compartment i, we derive the mass balances for the N compartments representing differentiation, as follows:

$$\begin{aligned}
\frac{dX_1}{dt} &= I - k_1 X_1 \\
\frac{dX_2}{dt} &= 2k_1 X_1 - k_2 X_2 \\
&\vdots \\
\frac{dX_i}{dt} &= 2k_{i-1} X_{i-1} - k_i X_i \\
&\vdots \\
\frac{dX_N}{dt} &= 2k_{N-1} X_{N-1} - k_N X_N
\end{aligned} \qquad \text{(P7.8)-(1)}$$

In addition, we make the assumption that all the red blood cells that are formed by this process enter the blood stream, where they serve their purpose and die, at the rate of 200 billion cells per day. The number of red blood cells in a healthy individual remains relatively constant, i.e., there is steady state. Then the balance on the blood stream results in the following equation:

$$\frac{dX_{\text{Blood}}}{dt} = 2k_N X_N - 200 \times 10^9 = 0 \qquad \text{(P7.8)-(2)}$$

Using the above differential equations, simulate numerically the erythropoiesis process and answer the following questions:

(a) What is the total number of stem cells per day, I, that need to commit to the erythropoiesis process in order to produce the required 200 billion red blood cells per day? Assume that the stem cells undergo a total of 10 stages of differentiation (N = 10), and that the initial conditions and transition rate constants are:

$$X_i(0) = 0 \text{ cells, for } i = 1,\ldots,N$$
$$k_i = 2.2 \text{ day}^{-1}, \text{ for } i = 1,\ldots,N$$

Explain carefully how you calculate the value of I.

(b) Show and discuss thoroughly the time profiles in the N stages of the differentiation/replication process and compare these results with those of Example 7.6 Case (c).

7.13 References

Burden, R. L., J. D. Faires, and A. C. Reynolds (1981). *Numerical Analysis*. Prindle, Weber & Schmidt, Boston, MA.

Constantinides, A., and N. Mostoufi (1999). *Numerical Methods for Chemical Engineers with MATLAB Applications*. Prentice Hall PTR, Upper Saddle River, NJ.

Edelstein-Keshet, L. (1988). *Mathematical Models in Biology*. McGraw-Hill Book Company, New York, NY.

Enderle, J., S. Blanchard, and J. Bronzino (2000). *Introduction to Biomedical Engineering*. Academic Press, San Diego, CA.

Finlayson, B. A. (1980). *Nonlinear Analysis in Chemical Engineering*. McGraw-Hill, New York, NY.

Fournier, R. L. (1999). *Basic Transport Phenomena in Biomedical Engineering*. Taylor & Francis, Philadelphia, PA.

Hairer, E., C. Lubich, and M. Roche (1980). *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Springer-Verlag, Berlin.

Hairer, E., and G. Wanner (1991). *Solving Ordfinary Differential Equations I*. Springer, Berlin.

Hairer, E., and G. Wanner (1991). *Solving Ordfinary Differential Equations II*. Springer, Berlin.

Hodgkin, A. L., and A. F. Huxley (1952). *A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve*. J. Physiol,. **117**:500-544.

Huang, B. K. (1994). *Computer Simulation Analysis of Biological and Agricultural Systems*. CRC Press, Boca Raton, FL.

Keener, J., and J. Sneyd (1998). *Mathematical Physiology*. Springer-Verlag, New York, NY.

Lapidus, L., and J. H. Sienfeld (1971). *Numerical Solution of Ordinary Differential Equations*. Academic Press, New York, NY.

Lauffenburger, D. A., and A. F. Horwitz (1996). *Cell Migration: A Physically Integrated Process*. Cell. **84**:359–369.

Palsson, B. Ø., and S. N. Bhatia (2004). *Tissue Engineering*. Pearson Prentice Hall, Upper Saddle River, NJ.

Tjia, J.S., and P.V. Moghe (2002a). *Regulation of Cell Motility on Polymer Substrates via Dynamic, Cell-Internalizable, Ligand Microinterfaces*. Tissue Eng., **8**:247-259.

Tjia, J.S. and P.V. Moghe (2002b). *Cell-Internalizable Ligand Microinterfaces on Biomaterials: Design of Regulatory Determinants Of Cell Migration* in Biomimetic Materials and Design: Interactive Biointerfacial Strategies for Drug Delivery and Tissue Engineering, A. Dillow, T. Lowman (Eds.), Marcel-Dekker, 335-373.

Tjia, J. S., and P.V. Moghe (2002c). *Cell Migration on Cell-Internalizable Ligand Microdepots: A Phenomenological Model*. Annals of Biomedical Engineering, **30**:851-866.

Tortora, G. J., and S. Reynolds Grabowski (2001). *Introduction to the Human Body*. 5th Edition, John Wiley & Sons, Inc., Hoboken, NJ.