Chapter 2

Letting It Fly: Initial Value Problems

2.1 Examples of Initial Value Problems

In this section we shall derive the mathematical models for two initial-value problems, one from the field of ecology and the other with aerospace applications.

A Predator-Prey Problem

We consider the population dynamics of two interacting species that have a predator-prey relationship. That is, the prey is able to find sufficient food but is killed by the predator whenever they encounter each other. Examples of such species interaction are wolves and rabbits, and parasites and certain hosts. What we want to investigate is how the predator and prey populations vary with time.

Let $x = x(t)$ and $y = y(t)$ designate the number of prey and predators, respectively, at time t. To derive mathematical equations that approximate the population dynamics we make several simplifying assumptions. First, we assume that the prey population, if left alone, increases at a rate proportional to x. Second, we assume that the number of times that the predator kills the prey depends on the probability of the two coming together and is therefore proportional to xy. Combining these two assumptions, the prey population is governed by the ordinary differential equation

$$\frac{dx}{dt} = \alpha x + \beta xy, \tag{2.1.1}$$

where $\alpha > 0$ and $\beta < 0$.

For the predator equation we assume that the number of predators would decrease by natural causes if the prey were removed, contributing a γy term. However, the number of predators increases as a result of encounters with prey, leading to

$$\frac{dy}{dt} = \gamma y + \delta xy \tag{2.1.2}$$

with $\gamma < 0$ and $\delta > 0$. In summary, we have the system of two nonlinear ordinary differential equations

$$\frac{dx}{dt} = \alpha x + \beta xy, \qquad \frac{dy}{dt} = \gamma y + \delta xy, \tag{2.1.3}$$

with the assumptions $\alpha > 0$, $\beta < 0$, $\gamma < 0$, and $\delta > 0$. These equations were first formulated in 1925 and are known as the *Lotka-Volterra equations*. The problem statement is not complete; we must start the process at some time (for example, $t = 0$) with given values for the initial populations $x(0)$ and $y(0)$. Thus we supplement the differential equations by two *initial conditions*:

$$x(0) = x_0 \qquad y(0) = y_0. \tag{2.1.4}$$

Note that the model (2.1.3) -(2.1.4) gives continuous solutions although the number of predators and prey will always be an integer. This is typical of many mathematical models in which discrete quantities are approximated by continuous ones so as to obtain a differential equation.

A Trajectory Problem

Ballistics problems have a long history in scientific computing and were one of the motivations for the development of computers during World War II. We will consider a simple ballistics problem as a special case of the more general problem of rocket trajectories. Suppose that a rocket is launched at a given angle of inclination to the ground (the launch angle). How high will the rocket go? The answer depends on a number of factors: the characteristics of the rocket and its engine, the drag caused by air density, the gravitational forces, and so on. To set up a mathematical model for this problem, we make a number of simplifying assumptions. First, we consider only rockets going to a height and range of no more than 100 kilometers; in this case, we can assume that the earth is flat with little loss of accuracy. Second, we assume that the trajectory of the rocket lies entirely in a plane; for example, we assume no wind effects. With these two assumptions, we set up a two-dimensional coordinate system centered at the launching site, and in Figure 2.1 we depict a typical trajectory.

As shown in Figure 2.1, $x(t)$ and $y(t)$ denote the x and y coordinates of the rocket at time t, where we assume that launch occurs at $t = 0$ and, hence,

$$x(0) = y(0) = 0. \tag{2.1.5}$$

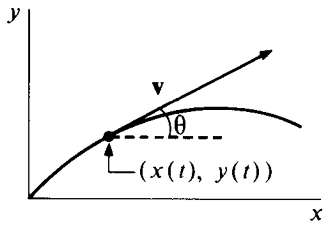


Figure 2.1: *A Typical Trajectory*

If we denote differentiation with respect to time by $\dot{x} = dx/dt$ and $\dot{y} = dy/dt$, then the velocity vector of the rocket at time t is $\mathbf{v}(t) = (\dot{x}(t), \dot{y}(t))$. We denote the magnitude of the velocity vector by $v(t)$ and its angle from the horizontal by $\theta(t)$, as shown in Figure 2.1. These quantities are then given by

$$v(t) = [(\dot{x}(t))^2 + (\dot{y}(t))^2]^{1/2}, \qquad \theta(t) = \tan^{-1} \frac{\dot{y}(t)}{\dot{x}(t)}. \qquad (2.1.6)$$

The basic mathematical model of the trajectory is derived from Newton's laws of motion, which give

$$\frac{d}{dt}(m\mathbf{v}) = F. \qquad (2.1.7)$$

Here $m(t)$ is the mass of the rocket, and F denotes the forces acting on the rocket and is composed of three terms: (1) the thrust, $T(t)$, when the rocket engine is firing; (2) the drag force

$$\tfrac{1}{2}c\rho s v^2, \qquad (2.1.8)$$

where c is the coefficient of drag, ρ is air density, and s is the cross-sectional area of the rocket; and (3) the gravitational force, gm, where g is the acceleration of gravity.

To write (2.1.7) in terms of x and y, we note that the part of the force F that consists of the thrust and the drag acts along the axis of the rocket. If we call this part F_1, then

$$F_1 = T - \tfrac{1}{2}c\rho s v^2 \qquad (2.1.9)$$

and (2.1.7) can be written

$$\dot{m}\dot{x} + m\ddot{x} = F_1 \cos\theta, \qquad \dot{m}\dot{y} + m\ddot{y} = F_1 \sin\theta - mg, \qquad (2.1.10)$$

since the gravitational force acts only in the vertical direction. Using (2.1.9) and rearranging terms, we rewrite (2.1.10) as

$$\ddot{x} = \frac{1}{m}(T - \tfrac{1}{2}c\rho s v^2)\cos\theta - \frac{\dot{m}}{m}\dot{x}, \tag{2.1.11a}$$

$$\ddot{y} = \frac{1}{m}(T - \tfrac{1}{2}c\rho s v^2)\sin\theta - \frac{\dot{m}}{m}\dot{y} - g. \tag{2.1.11b}$$

This is a coupled system of two second-order nonlinear [recall equation (2.1.6)] differential equations. We are assuming that c and s are known constants, ρ is a known function of y (height above the surface), and T and m (and hence $\dot{m}$) are known functions of t. (The change in mass is caused by the expenditure of fuel.)

The solution of (2.1.11) must satisfy (2.1.5), and this gives two of the four initial conditions that are needed. The other two are

$$v(0) = 0, \qquad \theta(0) = \theta_0. \tag{2.1.12}$$

Thus, for a given rocket, the only "free parameter" is the launch angle θ_0, and changes in the launch angle obviously cause changes in the trajectory.

Equations (2.1.11) also serve as the mathematical model for the "projectile problem", examples of which are a shell being shot from a cannon or a rock launched from a slingshot. In this case we assume that the projectile starts with a given velocity v_0, and thus (2.1.12) is changed to

$$v(0) = v_0, \qquad \theta(0) = \theta_0. \tag{2.1.13}$$

There is now no thrust, and hence no change of mass, so (2.1.11) simplifies to

$$\ddot{x} = \frac{-c\rho s v^2}{2m}\cos\theta, \qquad \ddot{y} = \frac{-c\rho s v^2}{2m}\sin\theta - g, \tag{2.1.14}$$

which in the context of our simplified model shows that given the initial velocity and launch angle, the trajectory depends only on the drag and gravitational forces.

Our task, now, is to solve the equations (2.1.11) with the initial conditions (2.1.5) and (2.1.13). [Henceforth we shall use (2.1.13) since it includes the special case $v_0 = 0$ of (2.1.12).] In the trivial case in which there is neither thrust nor drag, the equations can be solved explicitly (Exercise 2.1.3). However, for any realistic specification of the air density ρ and the thrust this is not possible, and an approximate numerical solution is required.

For the numerical solutions to be discussed later, it will be convenient to reformulate the two second-order equations (2.1.11) as a system of four first-order equations. By differentiating the relations

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \tag{2.1.15}$$

which are equivalent to (2.1.6), we have

$$\ddot{x} = \dot{v}\cos\theta - v\dot{\theta}\sin\theta, \qquad \ddot{y} = \dot{v}\sin\theta + v\dot{\theta}\cos\theta. \tag{2.1.16}$$

If we substitute (2.1.15) and (2.1.16) into (2.1.11) and solve for $\dot{v}$ and $\dot{\theta}$, we obtain

$$\dot{v} = \frac{1}{m}(T - \tfrac{1}{2}c\rho s v^2) - g\sin\theta - \frac{\dot{m}}{m}v \tag{2.1.17}$$

$$\dot{\theta} = -\frac{g}{v}\cos\theta. \tag{2.1.18}$$

Equations (2.1.17) and (2.1.18), together with (2.1.15), constitute a system of four first-order equations in the variables x, y, v, and θ. Again, the initial conditions are given by (2.1.5) and (2.1.13).

We shall return to the numerical solution of both the predator-prey problem and the trajectory problem after we have discussed the basic methods used for the solution.

Supplementary Discussion and References: 2.1

There is no known nontrivial analytical solution of the problem given by (2.1.3), (2.1.4), and we must use approximation methods. The primary concern of this book is with numerical methods that replace a continuous problem with a discrete problem that is solved on a computer. But we will consider here another approach to solving (2.1.3), (2.1.4). These *perturbation* methods replace the original continuous problem with a slightly different and simpler continuous problem which can be solved analytically.

The first step is to identify *stationary* or *equilibrium* states (x_s, y_s). In our case, the equations

$$x = x_s \equiv \frac{-\gamma}{\delta}, \qquad y = y_s \equiv \frac{-\alpha}{\beta}$$

represent stationary states because

$$\left.\frac{dx}{dt}\right|_{(x_s,y_s)} = x_s(\alpha + \beta y_s) = 0, \qquad \left.\frac{dx}{dt}\right|_{(x_s,y_s)} = y_s(\gamma + \delta x_s) = 0.$$

By expanding the right-hand sides of (2.1.3) in a Taylor series about (x_s, y_s), we obtain

$$x(\alpha + \beta y) = \beta x_s(y - y_s) + \cdots, \qquad y(\gamma + \delta x) = \delta y_s(x - x_s) + \cdots .$$

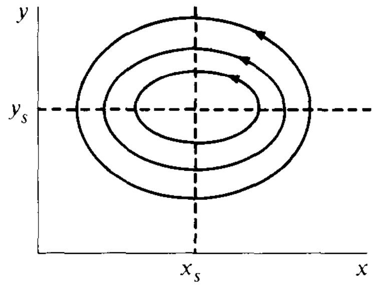
Thus in the neighborhood of (x_s, y_s) we approximate (2.1.3) by the linear equations

$$\dot{x} = \beta x_s(y - y_s), \qquad \dot{y} = \delta y_s(x - x_s). \tag{2.1.19}$$

These equations may be solved in the form

$$\frac{(x - x_s)^2}{-\beta x_s} + \frac{(y - y_s)^2}{\delta y_s} = c, \tag{2.1.20}$$

where c is some constant determined by the initial conditions. This is the equation of an ellipse whose center is at (x_s, y_s), and different starting values of $x(0)$ and $y(0)$ determine different ellipses. The figure that follows shows a family of ellipses about (x_s, y_s), with the arrows indicating the direction of increasing time. It can be seen that the populations are cyclical: after a certain time, they return to their original levels.



This type of perturbation analysis can provide useful information about the solution of (2.1.3) in the neighborhood of a stationary point. Because the equations (2.1.3) are approximated by equations (2.1.19), we might expect that the solutions to (2.1.3) would be close to the ellipses that solve (2.1.19). Such a relationship is verified by the numerical approximations described in the remainder of this chapter.

For further information on the predator-prey problem and other topics in mathematical biology, see Rubinow [1975], and for additional information on the theory of rocket trajectories, see, for example, Rosser, Newton, and Gross [1974].

EXERCISES 2.1

2.1.1. What relationships among the coefficients α, β, γ, and δ and the population levels x and y of (2.1.3) would guarantee stable populations for x and y (that is, $x(t+\Delta t) = x(t)$ and $y(t+\Delta t) = y(t)$ for all $\Delta t > 0$)?

2.1.2. Verify (2.1.6).

2.1.3. Show that the solution of $\ddot{x} = 0$, $\ddot{y} = -mg$, with initial conditions $x(0) = y(0) = 0$, $v(0) = v_0$, $\theta(0) = \theta_0$ is given by $x(t) = (v_0 \cos\theta_0)t$ and $y(t) = -mgt^2/2 + (v_0 \sin\theta_0)t$.

2.2 One-Step Methods

In the previous section we gave two examples of initial-value problems for systems of ordinary differential equations. We will now consider such problems in the general form

$$\frac{dy_i}{dx} = f_i(x, y_1(x), \dots, y_n(x)), \qquad i = 1, \dots, n, \quad a \le x, \tag{2.2.1}$$

with initial conditions

$$y_i(a) = \hat{y}_i, \quad i = 1, \dots, n. \tag{2.2.2}$$

Here, the f_i are given functions, x is the independent variable and the $\hat{y}_i$ are given initial conditions. In the previous section the predator-prey problem gave rise to two equations, whereas in the trajectory problem there were four first-order equations (see Exercise 2.2.1). More generally, as shown in Appendix 1, a single higher-order equation or a system of higher-order equations may always be reduced to a system of first-order equations; thus, the problem (2.2.1), (2.2.2) is very general. For simplicity in the subsequent presentation, we shall restrict our attention to a single equation

$$\frac{dy}{dx} = f(x, y), \quad a \le x, \tag{2.2.3}$$

in the single unknown function y, and with the initial condition

$$y(a) = \hat{y}. \tag{2.2.4}$$

Later in the section we shall show how the methods extend easily to systems of the form (2.2.1).

As indicated in (2.2.3), we wish to find the solution for $x \ge a$. In some problems we will wish to find the solution on a prescribed interval $[a, b]$. In other problems we may wish to know the behavior of the solution as $x \to \infty$; of course, we can only compute the solution numerically in a finite interval, but we may wish to continue the computations until the behavior for large x becomes clear. And, in still other problems, we may wish to integrate until a prescribed condition is satisfied; for example, in the trajectory problem discussed in the previous section we may wish to know when the missile hits the ground.

Although some initial-value problems have solutions that can be obtained analytically, many problems, including most of those of practical interest, cannot be solved in this manner. The purpose of the chapter is to describe methods for approximating solutions by using numerical methods, particularly by what are known as *finite difference methods.*

The first step in the numerical solution is to introduce the *grid points* $a = x_0 < x_1 < \cdots < x_N$ as shown in Figure 2.2. Although unequal spacing of the grid points presents no particular difficulties, we shall assume that they are equally spaced in order to simplify the discussion and analysis. If we let h denote the spacing, then $x_k = a + kh$, $k = 0, 1, \dots, N$. In what follows, $y(x_k)$ will denote the value of the exact solution of (2.2.3) at the point x_k, and

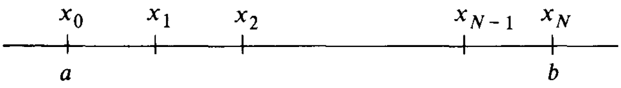


Figure 2.2: *Grid Points*

y_k will denote the approximation generated by the numerical method under consideration.

Euler's Method

Perhaps the simplest numerical scheme is *Euler's method*, which is defined by

$$y_0 = \hat{y}, \qquad y_{k+1} = y_k + hf(x_k, y_k), \quad k = 0, 1, \ldots, N-1. \tag{2.2.5}$$

The derivation of Euler's method is straightforward. By the Taylor expansion (see Appendix 1) of y about x_k, we have

$$\begin{aligned} y(x_{k+1}) &= y(x_k) + hy'(x_k) + \frac{h^2}{2}y''(z_k) \qquad (2.2.6)\\ &= y(x_k) + hf(x_k, y(x_k)) + \frac{h^2}{2}y''(z_k), \end{aligned}$$

where z_k is some point in the interval (x_k, x_{k+1}). Here, and henceforth, we will always assume that all derivatives shown do exist. Now if y'' is bounded and h is small, we may ignore the last term and we have, using the notation $\doteq$ to mean "approximately equal to,"

$$y(x_{k+1}) \doteq y(x_k) + hf(x_k, y(x_k)).$$

This is the basis for (2.2.5). Geometrically, Euler's method consists of approximating the solution at x_{k+1} by following the tangent to the solution curve at x_k (see Figure 2.3).

Euler's method is very easy to carry out on a computer: at the k-th step we evaluate $f(x_k, y_k)$ and use this in (2.2.5). Hence, essentially all of the computation required is in the evaluation of $f(x_k, y_k)$. We now give a simple example of the use of the method. Consider the equation

$$y'(x) = y^2(x) + 2x - x^4, \quad y(0) = 0. \tag{2.2.7}$$

It is easily verified that the exact solution of this equation is $y(x) = x^2$. Here $f(x, y) = y^2 + 2x - x^4$, and therefore Euler's method for (2.2.7) becomes

$$y_{k+1} = y_k + h(y_k^2 + 2kh - k^4h^4), \quad k = 0, 1, \ldots, \qquad y_0 = 0, \tag{2.2.8}$$

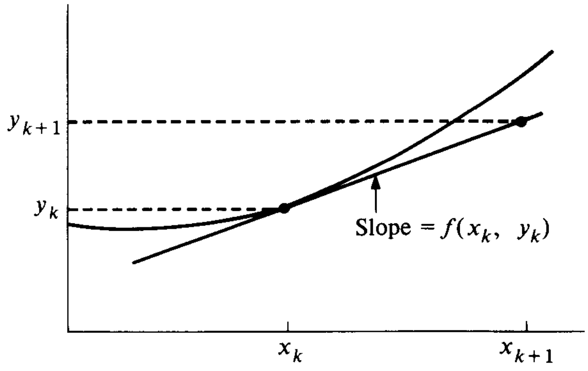


Figure 2.3: *One Step of Euler's Method*

since $x_k = kh$. In Table 2.1 we give some computed values for (2.2.8) for $h = 0.1$, as well as the corresponding values of the exact solution.

Table 2.1: *Computed and Exact Solutions for* (2.2.7) *by Euler's Method*

x	Computed Solution	Exact Solution
0.1	0.00	0.01
0.2	0.02	0.04
0.3	0.06	0.09
0.4	0.12	0.16
0.5	0.20	0.25
0.6	0.30	0.36

As Table 2.1 shows, the computed solution is in error, as is to be expected, and a major question regarding the use of Euler's method, or any other numerical method, is the accuracy of the approximations y_k. In general, the error in these approximations will come from two sources: (1) the discretization error that results from the replacement of the differential equation (2.2.3) by the approximation (2.2.5); and (2) the rounding error made in carrying out the arithmetic operations of the method (2.2.5). We shall consider the rounding error later, and for the moment we shall assume that the y_k of (2.2.5) are computed exactly so that the only error is the discretization error.

Discretization Error

We first consider the discretization error

$$E(h;b) = |y_N - y(b)| \tag{2.2.9a}$$

at a fixed point

$$b = a + hN. \tag{2.2.9b}$$

Note that N, and thus y_N, is function of h; in particular, for fixed b, as h decreases, then N increases and if $h \to 0$, then $N \to \infty$. We only allow h to vary, however, in such a way that (2.2.9b) is satisfied for an integer N. The maximum discretization error on the whole interval $[a, b]$,

$$E(h;[a,b]) = \max_{1\le i\le N} |y_i - y(a+ih)|, \tag{2.2.9c}$$

is called the *global discretization error* (sometimes called the *global truncation error*) on the interval $[a, b]$. When the interval is clear, we will usually denote $E(h;[a,b])$ by $E(h)$. Intuitively, we expect – and certainly hope – that $E(h) \to 0$ as $h \to 0$.

We next show how the global discretization error can be bounded. First, we will assume that the exact solution y has a bounded second derivative y'' on the interval $[a, b]$ and set

$$\max_{a\le x\le b} |y''(x)| = M. \tag{2.2.10}$$

We then consider the expression

$$L(x,h) = \frac{1}{h}[y(x+h) - y(x)] - f(x, y(x)), \tag{2.2.11}$$

which is called the *local discretization error for Euler's method at point* x and is a measure of how much the difference quotient for $y'(x)$ differs from $f(x, y(x))$. Now suppose that y_k equals the exact solution $y(x_k)$. Then the difference between the Euler approximation y_{k+1} and the exact solution $y(x_{k+1})$ is simply

$$y(x_{k+1}) - y_{k+1} = y(x_{k+1}) - y(x_k) - hf(x_k, y(x_k)) = hL(x_k, h). \tag{2.2.12}$$

That is, h times the local discretization error is the error produced in a single step of Euler's method starting from the exact solution.

We shall be interested in the maximum size of $L(x, h)$ for any value of x, and we define the local discretization error for Euler's method by

$$L(h) = \max_{a\le x\le b-h} |L(x,h)|. \tag{2.2.13}$$

Note that $L(h)$ depends on the step length, h, as well as on the function f of the differential equation and the interval $[a, b]$. The only dependence we have explicitly delineated, however, is that on h, since under the assumption (2.2.10) and using a Taylor expansion analogous to (2.2.6) we obtain the bound

$$L(h) \le \frac{h}{2}M = 0(h). \tag{2.2.14}$$

Here we have used the standard notation $0(h)$ to denote a quantity that goes to zero as rapidly as h goes to zero. More generally, we will say that a function g of h is $0(h^p)$ if $g(h)/h^p$ is bounded as $h \to 0$ but $g(h)/h^q$ is unbounded if $q > p$.

The problem now is to relate the local discretization error to the global discretization error. If we denote the error $y(x_k) - y_k$ by e_k, then we have, by using (2.2.5) and (2.2.11),

$$\begin{aligned} e_{k+1} &= y(x_{k+1}) - y_{k+1} \\ &= y(x_k) + hf(x_k, y(x_k)) + hL(x_k, h) - y_k - hf(x_k, y_k) \\ &= e_k + h[f(x_k, y(x_k)) - f(x_k, y_k)] + hL(x_k, h). \end{aligned} \tag{2.2.15}$$

Now assume that the function f has a bounded partial derivative with respect to its second variable:

$$\left|\frac{\partial f}{\partial y}(x, y)\right| \le M_1, \quad a \le x \le b, \quad |y| < \infty. \tag{2.2.16}$$

Then by the mean-value theorem (Appendix 1), we have for some $0 < \theta < 1$,

$$\begin{aligned} |f(x_k, y(x_k)) - f(x_k, y_k)| &= \left|\frac{\partial f}{\partial y}(x_k, \theta y(x_k) + (1-\theta)y_k)(y(x_k) - y_k)\right| \\ &\le M_1|e_k|. \end{aligned}$$

Substituting this into (2.2.15) and bounding $L(x_k, h)$ by $L(h)$ gives

$$|e_{k+1}| \le (1 + hM_1)|e_k| + h|L(h)|. \tag{2.2.17}$$

This is an inequality of the form

$$\bar{e}_{k+1} \le c\bar{e}_k + d,$$

where $\bar{e}_k = |e_k|$, $c = 1 + hM_1$, $d = hL(h)$ and $\bar{e}_0 = 0$. Thus,

$$\begin{aligned} \bar{e}_n &\le c(c\bar{e}_{n-2} + d) + d \le \cdots \le (1 + c + \cdots c^{n-1})d \\ &= \left(\frac{c^n - 1}{c - 1}\right) d = \frac{[(1 + hM_1)^n - 1]}{hM_1} hL(h) \le \frac{(1 + hM_1)^n}{M_1} L(h) \\ &\le \frac{e^{hnM_1}}{M_1} L(h), \end{aligned} \tag{2.2.18}$$

where the last inequality follows from $1 + x \le e^x$ for $x \ge 0$. Now $x_n = x_0 + nh$ and if we keep x_n fixed at some point x^+ in the interval $[a, b]$ as $h \to 0$, then (2.2.18) becomes

$$\bar{e}_n \le \frac{L(h)}{M_1} e^{M_1(x^+ - x_0)}.$$

This estimate is an upper bound on the absolute value of the error and may be rather pessimistic. Nevertheless, it does show that $e_n \to 0$ since $L(h) = 0(h)$. Moreover, $E(h) = 0(h)$. Thus, we have proved the following result.

THEOREM 2.2.1 (Euler Discretization Error) *If the function f has a bounded partial derivative with respect to its second variable, and if the solution of* (2.2.3), (2.2.4) *has a bounded second derivative, then the Euler approximations converge to the exact solution as $h \to 0$, and the global discretization error of Euler's method satisfies $E(h) = 0(h)$.*

The fact that the global discretization error is $0(h)$ is usually expressed by saying that Euler's method is *first order.* The practical consequence of this is that as we decrease h, we expect that the approximate solution will become more accurate and converge to the exact solution at a linear rate in h as h tends to zero; that is, if we halve the step size, h, we expect that the error will decrease by about a factor of 2. This error behavior is shown in the following example.

Consider the equation $y' = y$, $y(0) = 1$, for which the exact solution is $y(x) = e^x$. We compute the solution at $x = 1$ by Euler's method using various values of h (see Table 2.2). The exact solution at $x = 1$ is $e = 2.718\ldots$; the errors for the different step sizes are given in the middle column. The ratios of the errors for successive halvings of h are given in the right-hand column, and it is seen that these ratios tend to $\frac{1}{2}$, as expected.

Table 2.2: *Error in Euler's Method*

h	Computed Value	Error	Error Ratio
1	2.000	0.718	
1/2	2.250	0.468	0.65
1/4	2.441	0.277	0.59
1/8	2.566	0.152	0.55
1/16	2.638	0.080	0.53

Runge-Kutta Methods

The very slow rate of convergence shown in Table 2.2 as h decreases is typical of first-order methods and militates against their use. Much of the rest of this chapter will be devoted to studying other methods for which the error tends to zero at a faster rate as h tends to zero. As an example of one of the approaches to such methods, we next discuss the *Heun method*, which is given by

$$y_{k+1} = y_k + \frac{h}{2}[f(x_k, y_k) + f(x_{k+1}, y_k + hf(x_k, y_k))]. \qquad (2.2.19)$$

Note that we have just replaced $f(x_k, y_k)$ in Euler's method by an average of f evaluated at two different places. This is illustrated in Figure 2.4. The Heun method is also known as a second-order *Runge-Kutta method* and has a local discretization error that is $0(h^2)$, as we will show shortly.

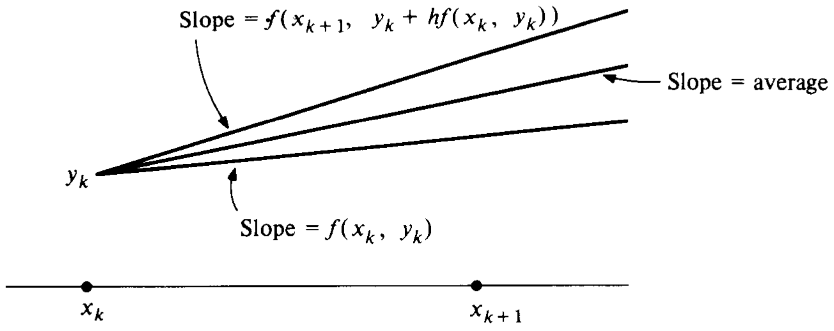


Figure 2.4: *The Heun Method*

The most famous of the Runge-Kutta methods is the classical fourth-order method, given by

$$y_{k+1} = y_k + \frac{h}{6}(F_1 + 2F_2 + 2F_3 + F_4) \tag{2.2.20}$$

where

$$\begin{aligned} F_1 &= f(x_k, y_k), \qquad F_2 = f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2}F_1\right), \\ F_3 &= f\left(x_k + \frac{h}{2}, y_k + \frac{h}{2}F_2\right), \qquad F_4 = f(x_{k+1}, y_k + hF_3). \end{aligned}$$

Here the $f(x_k, y_k)$ in Euler's method has been replaced by a weighted average of f evaluated at four different points. It is instructive to draw the figure corresponding to Figure 2.4; this is left to Exercise 2.2.9.

One-Step Methods

In Section 2.4 we will consider methods based on using information from prior steps so that y_{k+1} will be a function not only of y_k but also of y_{k-1} and, perhaps, other prior values. The present section deals with methods that depend only on y_k. Such methods are called *one step methods* and can be written in the general form

$$y_{k+1} = y_k + h\phi(x_k, y_k) \tag{2.2.21}$$

for some suitable function ϕ. In the case of Euler's method ϕ is just f itself, whereas for the Heun method

$$\phi(x, y) = \tfrac{1}{2}[f(x, y) + f(x + h, y + hf(x, y))]. \tag{2.2.22}$$

The fourth-order Runge-Kutta method (2.2.20) is also a one-step method, and the corresponding function ϕ can be written in a manner similar to (2.2.22) (see Exercise 2.2.5).

For any one-step method (2.2.21), we define the local discretization error in a manner analogous to that for Euler's method by

$$L(h) = \max_{a \le x \le b-h} |L(x,h)|, \quad L(x,h) = \frac{1}{h}[y(x+h) - y(x)] - \phi(x, y(x)) \quad (2.2.23)$$

where, again, $y(x)$ is the exact solution of the differential equation. If, for a given ϕ, $L(h) = 0(h^p)$ for some integer p, then it is possible to show, under suitable assumptions on ϕ and f, that the global discretization error will also be of order p in h:

$$E(h) \equiv \max_{1 \le k \le N} |y(x_k) - y_k| = 0(h^p). \quad (2.2.24)$$

The *order* of the method (2.2.21) is defined to be the integer p for which $L(h) = 0(h^p)$. This definition of order is a statement about the method and assumes that the solution y of the differential equation has bounded derivatives of suitably high order. For example, we showed that $p = 1$ for Euler's method under the assumption (2.2.10), and Table 2.2 illustrated that the error decreased by a factor of about 2^{-1} when the step length h was halved. For a pth order method, we expect the error to decrease by a factor of about 2^{-p} when we halve h, at least for h sufficiently small.

It is a relatively simple matter to show that the local discretization error for Heun's method is $0(h^2)$, but this will be a consequence of the following more general analysis. Consider a function ϕ defined by

$$\phi(x,y) = c_2 f(x,y) + c_3 f(x + c_1 h, y + c_1 h f(x,y)),$$

where we wish to determine the constants c_1, c_2, and c_3 so as to maximize the order of the one-step method (2.2.21); that is, we wish the best linear combination, as determined by c_2 and c_3, of two values of f, and how far along the interval the second evaluation of f should be done, as determined by c_1.

We expand f in a Taylor series in two variables about the point (x, y). First, in the x variable, we have

$$\phi = c_2 f + c_3[f(x, y + c_1 h f) + c_1 h f_x(x, y + c_1 h f) + 0(h^2)],$$

where we have denoted $f(x,y)$ simply by f and the partial derivative of f with respect to x by f_x. Next, expand in y where all partial derivatives shown are evaluated at (x, y):

$$\begin{aligned}\phi &= c_2 f + c_3[f + c_1 h f f_y + 0(h^2) + c_1 h f_x + 0(h^2)] \qquad (2.2.25)\\ &= (c_2 + c_3) f + c_1 c_3 h (f f_y + f_x) + 0(h^2).\end{aligned}$$

On the other hand, the exact solution $y(x)$ of the differential equation satisfies

$$\begin{aligned}\frac{1}{h}[y(x+h) - y(x)] &= y'(x) + \tfrac{1}{2} y''(x) h + 0(h^2) \qquad (2.2.26)\\ &= f + \tfrac{1}{2} h \frac{df}{dx} + 0(h^2)\\ &= f + \tfrac{1}{2} h (f f_y + f_x) + 0(h^2).\end{aligned}$$

Therefore, (2.2.25) and (2.2.26) combine to yield

$$\begin{aligned}&\frac{1}{h}[y(x+h) - y(x)] - \phi(x, y(x)) \qquad (2.2.27)\\ &\quad = (1 - c_2 - c_3) f + h(\tfrac{1}{2} - c_1 c_3)(f f_y + f_x) + 0(h^2).\end{aligned}$$

If we require that

$$c_2 + c_3 = 1, \qquad c_1 c_3 = \frac{1}{2}, \qquad (2.2.28)$$

then the first two terms of (2.2.27) vanish for any f. Therefore $L(h) = 0(h^2)$. Moreover, by carrying out the Taylor expansions one more term, it can be shown that, in general, we cannot achieve $L(h) = 0(h^3)$ no matter what choice of the constants c_1, c_2, c_3 is made. Hence we have

$$\frac{1}{h}[y(x+h) - y(x)] - \phi(x, y(x)) = 0(h^2), \qquad (2.2.29)$$

which will hold whenever (2.2.28) is satisfied and the various derivatives we have used are bounded. Therefore, the methods delineated by (2.2.28) are all second order so that there is not a unique second-order method of this type.

If we set $c_1 = \gamma/2$ and solve the two equations of (2.2.28) in terms of γ, we obtain a function that will always satisfy (2.2.28). Therefore, the method

$$y_{k+1} = y_k + h\left[\left(1 - \frac{1}{\gamma}\right) f(x_k, y_k) + \frac{1}{\gamma} f\left(x_k + \frac{\gamma}{2} h, y_k + \frac{\gamma h}{2} f(x_k, y_k)\right)\right]$$

is second-order accurate for any $\gamma \neq 0$. The special choice $\gamma = 2$ gives the second-order Runge-Kutta method (2.2.19). The derivation of higher-order Runge-Kutta methods, and in particular the fourth-order method (2.2.20), can proceed in an analogous, but more complicated, manner.

Systems of Equations

We next indicate how the above methods can be used for systems of equations. Consider the system (2.2.1), which we will write in the vector form

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)). \tag{2.2.30}$$

Here $\mathbf{y}(x)$ denotes the vector with components $y_1(x), \ldots, y_n(x)$, and $\mathbf{f}$ is the vector with components $f_1, \ldots, f_n$. The vector $\hat{\mathbf{y}}$ will denote the initial values (2.2.2). Then Euler's method (2.2.5) can be written for the system (2.2.30) as

$$\mathbf{y}_0 = \hat{\mathbf{y}} \qquad \mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}(x_k, \mathbf{y}_k), \quad k = 0, 1, \ldots \tag{2.2.31}$$

where $\mathbf{y}_1, \mathbf{y}_2, \ldots$ are vector approximations to the solution $\mathbf{y}$. We could, of course, write out (2.2.31) in component form; for $n = 2$, this would be

$$y_{1,0} = \hat{y}_1 \qquad y_{2,0} = \hat{y}_2$$

$$\left.\begin{aligned} y_{1,k+1} &= y_{1,k} + hf_1(x_k, y_{1,k}, y_{2,k}) \\ y_{2,k+1} &= y_{2,k} + hf_2(x_k, y_{1,k}, y_{2,k}) \end{aligned}\right\}, \quad k = 0, 1 \ldots .$$

Clearly, the succinct vector notation (2.2.31) is advantageous.

Similarly, Heun's method (2.2.19) can be written in vector form for (2.2.30) by

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{2}[\mathbf{f}(x_k, \mathbf{y}_k) + \mathbf{f}(x_{k+1}, \mathbf{y}_k + h\mathbf{f}(x_k, \mathbf{y}_k))]. \tag{2.2.32}$$

It is left to Exercise 2.2.7 to write the fourth-order Runge-Kutta method (2.2.20) in vector form.

Rounding Error

We now turn to a brief discussion of the rounding error in the methods of this section. Consider Euler's method, in which there are two sources of rounding error. The first is the error that occurs in the evaluation of $f(x_k, y_k)$; we will denote this error by ε_k. The second error, η_k, is the error made in the Euler formula. Thus, the computed approximations y_k satisfy

$$y_{k+1} = y_k + h[f(x_k, y_k) + \varepsilon_k] + \eta_k, \quad k = 0, 1, \ldots . \tag{2.2.33}$$

It is possible to bound the effects of these errors in terms of bounds on the ε_k and η_k. However, we will content ourselves with the following intuitive discussion. As we have seen, the global discretization error in Euler's method goes to zero as h goes to zero. Hence we can make the discretization error as small as we like by making h sufficiently small. However, the smaller h is, the more steps of Euler's method that will be required and, in general, the larger the effect of the rounding error on the computed solution. In practice, for a fixed word length in the computer arithmetic there will be a size of h below which the rounding error will become the dominant contribution to the overall error. The situation is depicted schematically in Figure 2.5, in which the step-size h_0 is the practical minimum that can be used. This minimum step-size is very difficult to ascertain in advance, but for problems for which only a moderate accuracy is required, the step size used will be far larger than this minimum, and the discretization error will be the dominant contributor to the error. The same general behavior occurs in all the methods, although the minimum step size, h_0, will depend on the method and the problem.

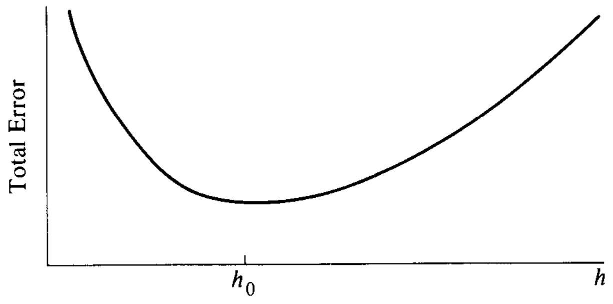


Figure 2.5: *Error in Euler's and Other Methods*

Change in Step Size

So far in this section we have considered a fixed step-size h. However, in practice it is usually beneficial to allow the step to vary. By the bound (2.2.14) on the local discretization error, it follows that the error is small if the second derivative of the solution is small. In regions where this is the case, suitable accuracy may be obtained by using a relatively large step-size. On the other hand, if the second derivative is large a smaller step-size will be necessary to control the discretization error. For the higher order Runge-Kutta methods, the same idea will hold but higher-order derivatives of the solution will be the determining factor. Generally, one will not know much about these higher derivatives, but many times it will be known in advance that the solution is varying slowly in some region and rapidly in another, and the step-size can be changed accordingly. For further discussion on ways to estimate the discretization error so as to ascertain the proper step-size, see the Supplementary Discussion.

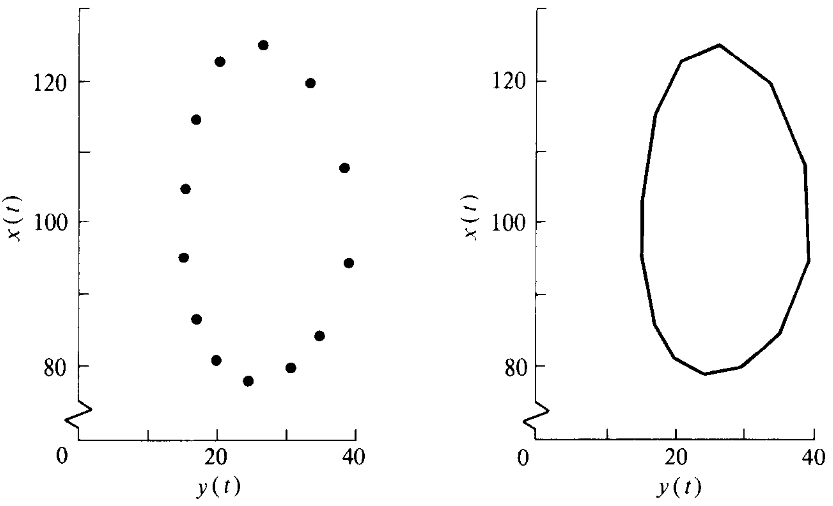
Numerical Examples

We complete this section with some simple calculations for the predator-prey equations introduced in Section 2.1. Recall that these equations are

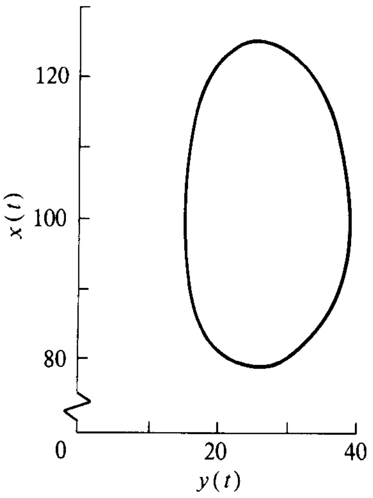
$$\frac{dx}{dt} = \alpha x + \beta xy, \qquad \frac{dy}{dt} = \gamma y + \delta xy, \tag{2.2.34}$$

with the initial conditions

$$x(0) = x_0, \qquad y(0) = y_0. \tag{2.2.35}$$



(a) Data only (b) Data connected by lines



(c) Data interpolated by a function

Figure 2.6: *Options for Graphical Output. Solution to* (2.2.34), (2.2.35)

The Supplementary Discussion of Section 2.1 showed that

$$x_s = \frac{-\gamma}{\delta}, \qquad y_s = \frac{-\alpha}{\beta}$$

is a stationary point of (2.2.34), and that in the neighborhood of (x_s, y_s) the path traced out by $(x(t),\ y(t))$ for $t > 0$ is approximately an ellipse. For illustration purposes, we have chosen initial values x_0 and y_0 that are near stationary points. We have used the following values for the parameters: $\alpha = 0.25$, $\beta = -0.01$, $\gamma = -1.00$, and $\delta = 0.01$. For these parameter values, there is a stationary point at $(x_s, y_s) = (100, 25)$. Initial values of $x_0 = 80$ and $y_0 = 30$ were used in all cases.

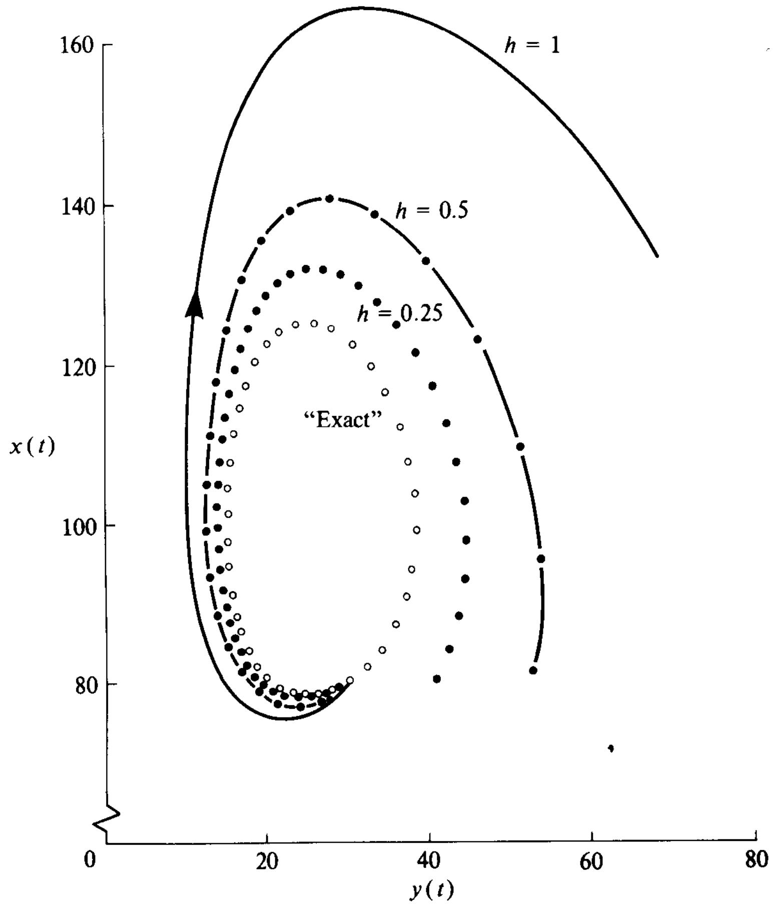


Figure 2.7: *Euler's Method for* (2.2.34), (2.2.35) *Using Different Step Sizes*

Figures 2.6 – 2.8 are the plotted approximations to solutions of (2.2.34), (2.2.35) generated by several of the numerical methods in this section. In all cases we have plotted x (the prey) versus y (the predator), both as functions of time, t. The motion is in a clockwise direction as t increases. Figure 2.6 demonstrates three options that are usually available for graphics. Part (a) plots just the discrete values (x_i, y_i) generated by the numerical method, emphasizing the discrete nature of the methods. In part (b) the points are connected by straight lines, giving a polygonal shape to the approximation. Part (c) shows the dots connected by smooth curves. This option requires some special software based on approximation methods like those discussed in Section 6.2.

Figure 2.7 demonstrates the dependency of the approximate solution on the value of the step size, h. The numerical method used for this figure is Euler's method defined by (2.2.5). The values of h are 1, 0.5, and 0.25. One sees that as the step size is halved the error is roughly halved also, suggesting $0(h)$ convergence. Clearly, the errors are rather large even for $h = 0.25$. The "exact" solution used for comparison was obtained by a higher-order Runge-Kutta method, and the solution so obtained may be considered to be exact for the purpose of comparing with the lower-order methods.

Figure 2.8 shows the effect of using a second-order method rather than the first-order Euler method. Here the error for a step size of $h = 1$ is less than that for Euler's method with a step size of $h = 0.25$. Note that, as with Euler's method, the approximate solution is spiraling out away from the exact solution.

Supplementary Discussion and References: 2.2

Perhaps the conceptually simplest approach to higher-order one-step methods is Taylor-series expansion of the solution. Consider the "method"

$$y_{k+1} = y_k + hy'(x_k) + \tfrac{1}{2}h^2 y''(x_k) + \cdots + \frac{h^p}{p!}y^{(p)}(x_k), \tag{2.2.36}$$

where y is the exact solution. It is easy to see that the order of this method is p. The higher derivatives of the solution can be obtained in principle from the differential equation itself. Thus $y'(x) = f(x, y(x))$, and

$$y''(x) = \frac{d}{dx}f(x, y(x)) = f_x(x, y(x)) + f_y(x, y(x))y'(x). \tag{2.2.37}$$

We then approximate $y'(x_k)$ by $f(x_k, y_k)$, and proceed similarly for higher derivatives. Thus the method for $p = 1$ is simply Euler's method, whereas for $p = 2$ it becomes

$$y_{k+1} = y_k + hf(x_k, y_k) + \frac{h^2}{2}[f_x(x_k, y_k) + f_y(x_k, y_k)f(x_k, y_k)],$$

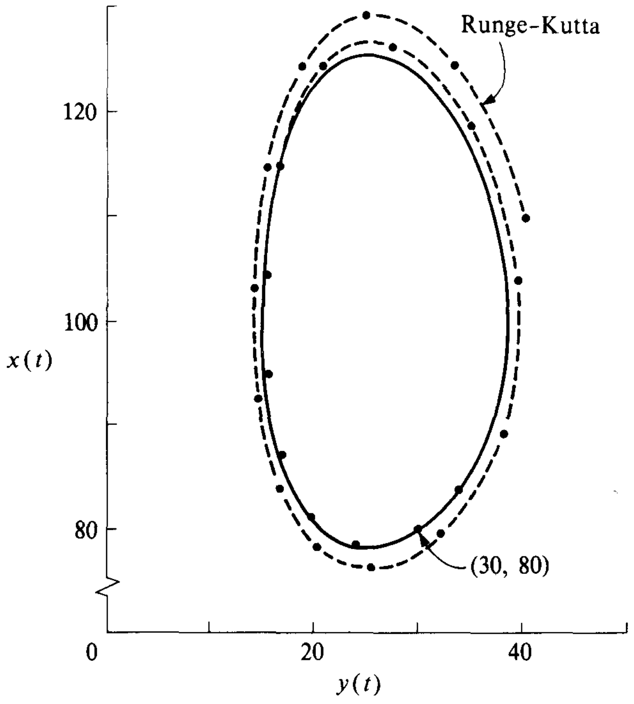


Figure 2.8: *Second-Order Runge-Kutta (Spiraling Outward) with Step-Size $h =$ 1, Compared to the "Exact" Solution of* (2.2.34), (2.2.35)

which is a second-order method. One can continue to differentiate (2.2.37) to obtain higher derivatives of y in terms of higher partial derivatives of f, but the methods become exceedingly cumbersome. Symbol manipulation techniques have proved somewhat useful in generating the derivatives. Note that Taylor series methods achieve higher order by using derivatives of f evaluated at a single point, whereas Runge-Kutta methods evaluate only f at different points. For further discussion of Taylor-series methods, see Daniel and Moore [1970].

Runge-Kutta methods of order higher than four may be obtained but at still additional costs in evaluations of the function f. Runge-Kutta methods of order p require p evaluations of f for $2 \leq p \leq 4$, $p+1$ evaluations for $5 \leq p \leq 7$, and $p+2$ evaluations for $p \geq 8$. For a thorough discussion of Runge-Kutta methods, see, for example, Henrici [1962], Butcher [1987], or Hairer, Norsett, and Wanner [1987].

All good computer codes using Runge-Kutta methods employ some mechanism for automatically changing the step size h as the integration proceeds. The problem is to ascertain the proper step size before the start of the next integration step. The usual approach is to estimate the local discretization error and adjust the step size accordingly. There are several ways to estimate the local error; two simple approaches are to repeat the last step of the integration with a step size half as large and then to compare the two results, or to use two Runge-Kutta formulas of different order. Both of these ways are costly in evaluations of f, and an alternative approach is by means of the Runge-Kutta-Fehlberg formulas (see Butcher [1987] for further discussion). Here one can use, for example, a Runge-Kutta method of order five to estimate the error in a fourth-order Runge-Kutta method in such a way that only six evaluations of f are needed, as opposed to ten if the usual Runge-Kutta formulas were used. One of the best and most used Runge-Kutta codes is RKF45; see Shampine, Watts, and Davenport [1976].

As we have shown, in the absence of rounding error Euler's method will yield the true solution of the differential equation as $h \to 0$. But, as Table 2.2 illustrates, the rate of convergence can be rather slow. An important technique to accelerate the convergence is *Richardson extrapolation to the limit.* For some fixed point x^*, denote the approximation to the solution by $y(x^*;h)$. Under certain assumptions, it can be shown that

$$y(x^*;h) = y(x^*) + c_1h + c_2h^2 + \cdots + c_ph^p + 0(h^{p+1}), \qquad (2.2.38)$$

where the c_i are functions of x^*. Now suppose that we also compute the approximate solution with step length $h/2$. Then we can combine $y(x^*;h)$ and $y(x^*;h/2)$ to obtain a better approximation. In particular,

$$\bar{y}(x^*;h) \equiv 2y(x^*;\frac{h}{2}) - y(x^*;h) = y(x^*) + d_2h^2 + \cdots + 0(h^{p+1}),$$

so that $\bar{y}(x^*;h)$ is a second order approximation. (Note that the more accurate solution is obtained only at the points with spacing h, and not at the intermediate points.) The process can then be repeated, if desired, to eliminate the coefficient d_2 and obtain a third order approximation, and so on. The same idea can be applied to other problems such as numerical integration or differentiation, or other types of differential equations, provided that an "asymptotic expansion" of the form (2.2.38) holds for the approximate solution.

An important modification of a system of differential equations occurs when there are side conditions. For example, a system of the form

$$\begin{aligned} y_i' &= f_i(x, y, (x), \ldots, y_n(x)), \qquad & i = 1, \ldots, m, \\ 0 &= g_i(x, y, (x), \ldots, y_n(x)), \qquad & i = m+1, \ldots, n, \end{aligned}$$

consists of m differential equations and $n-m$ non-differential equations. Such a system is called *differential-algebraic.* For further discussion, see, for example, Hairer, Lubich, and Roche [1989], and Brenan, Campbell, and Petzold [1989].

EXERCISES 2.2

2.2.1. Rewrite the predator-prey equations (2.1.3) in the form (2.2.1); that is, give the functions f_1 and f_2. Do the same for the trajectory equations (2.1.15), (2.1.17), and (2.1.18).

2.2.2. Apply Euler's method (2.2.5) to the initial-value problem $y' = -y$, $0 \le x \le 1$, $y(0) = 1$, with $h = 0.25$. Compare your answers to the exact solution $y(x) = e^{-x}$. Repeat for $h/2$ and $h/4$.

2.2.3. Verify the calculations of Tables 2.1 and 2.2.

2.2.4. Apply the Heun method to the problem of Exercise 2.2.2. Compare your results with Euler's method.

2.2.5. Give the function ϕ of (2.2.21) for the fourth-order Runge-Kutta method (2.2.20).

2.2.6. The method $y_{k+1} = y_k + hf(x_k + (h/2), y_k + (h/2)f(x_k, y_k))$ is known as the *midpoint rule.* Show that it is second-order accurate.

2.2.7. Write the fourth-order Runge-Kutta method (2.2.20) in vector form for the system (2.2.30).

2.2.8. Apply Euler's method and the Heun method to the problem $y'(x) = x^2 + [y(x)]^2, y(0) = 1$, $x \ge 0$, and compute y_2 for $h = 0.1$.

2.2.9. For the fourth-order Runge-Kutta method (2.2.20), draw the figure corresponding to Figure 2.4.

2.2.10. Repeat the calculations of Figure 2.7 using Euler's method with step sizes 0.5 and 0.25. How small a step size do you have to use for the graph of the solution to close back on itself to visual accuracy?

2.2.11. Test the stability of the solution of the predator-prey equations (2.2.34) with respect to changes in the initial conditions by changing $x_0 = 80$, $y_0 = 30$ by a unit amount in each direction (four different cases) and repeating the calculation using the second-order Runge-Kutta method.

2.2.12. Suppose that the initial condition of (2.2.5) is in error. Modify the analysis of the discretization error in Euler's method to obtain a bound on the error caused by using an inexact initial condition $y_0 = \bar{y} \ne \hat{y}$.

2.3 Polynomial Interpolation

The methods described in Section 2.2 were all one-step methods – methods that estimate y at x_{k+1} using information only at the previous point, x_k. In Section 2.4 methods that use information at several previous points will be described. In order to develop such methods, however, we must first describe polynomial interpolation, the subject of this section.

Suppose that one is given a set of points or *nodes* $x_0, x_1, \ldots, x_n$ and a set of corresponding numbers $y_0, y_1, \ldots, y_n$. The *interpolation problem* is to find a function g that satisfies

$$g(x_i) = y_i, \qquad i = 0, 1, \ldots, n; \tag{2.3.1}$$

if (2.3.1) holds we say that g interpolates the data. There are many types of possible functions g that might be used, but the functions of interest in this section will be polynomials.

It is clear that a polynomial of a given degree can not always be found so that (2.3.1) is satisfied. For example, if the data y_i are given at three distinct nodes, no polynomial of degree 1 (a linear function) can satisfy (2.3.1) unless the data lie on a straight line. On the other hand, there is a polynomial of degree 2 and infinitely many polynomials of degree 3 which satisfy (2.3.1). The basic result for polynomial interpolation is given in the following theorem.

THEOREM 2.3.1 (Existence and Uniqueness for Polynomial Interpolation) *If $x_0, x_1, \ldots, x_n$ are distinct nodes, then for any $y_0, y_1, \ldots, y_n$ there exists a unique polynomial $p(x)$ of degree n or less, such that*

$$p(x_i) = y_i, \qquad i = 0, 1, \ldots, n. \tag{2.3.2}$$

Proof: The existence can be proved by constructing the *Lagrange polynomials* defined by

$$\begin{aligned} l_j(x) &= \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)} \\ &= \prod_{\substack{k=0\\k\neq j}}^{n} \left(\frac{x-x_k}{x_j-x_k}\right), \qquad j = 0, 1, \ldots, n. \end{aligned} \tag{2.3.3}$$

It is easy to verify that these polynomials, which are all of degree n, satisfy

$$l_j(x_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases} \tag{2.3.4}$$

Therefore $l_j(x)y_j$ has the value 0 at all nodes x_i, $i = 0, 1, \ldots, n$, except for x_j, where $l_j(x_j)y_j = y_j$. Thus, by defining

$$p(x) = \sum_{j=0}^{n} l_j(x)y_j, \tag{2.3.5}$$

we have a polynomial of degree n or less that interpolates the data.

To prove uniqueness, suppose, on the contrary, that there is another interpolating polynomial of degree n or less, say $q(x)$. By defining

$$r(x) = p(x) - q(x)$$

we obtain a polynomial, r, of degree n or less that is equal to zero at the $n+1$ distinct points $x_0, x_1, \dots, x_n$. By the fundamental theorem of algebra, such a polynomial must be identically equal to zero, and it follows that $p(x) = q(x)$. Thus, uniqueness is proved.

As an example of polynomial interpolation, let us determine the polynomial $p(x)$ of degree 2 or less that satisfies $p(-1) = 4$, $p(0) = 1$, and $p(1) = 0$. The interpolating polynomial (2.3.5) is

$$\begin{aligned} p(x) &= \frac{(x-0)(x-1)}{(-1-0)(-1-1)}4 + \frac{(x-(-1))(x-1)}{(0-(-1))(0-1)}1 + \frac{(x-(-1))(x-0)}{(1-(-1))(1-0)}0 \\ &= 2x^2 - 2x + 1 - x^2 + 0 = x^2 - 2x + 1. \end{aligned}$$

One can easily verify that this $p(x)$ does interpolate the given data.

Error Estimates

We next consider the question of accuracy in polynomial interpolation. In many applications of interpolation there is some function f defined over the entire interval of interest, even though values are used only at discrete points to determine an interpolating polynomial p. Thus it is of interest to discuss the discrepancy between $p(x)$ and $f(x)$ for values of x that lie between the nodes. The following theorem gives an expression for the error in terms of higher derivatives of f:

THEOREM 2.3.2 (Polynomial Interpolation Error) *Let $f(x)$ be a function with $n+1$ continuous derivatives on an interval containing the distinct nodes $x_0 < x_1 < \cdots < x_n$. If $p(x)$ is the unique polynomial of degree n or less satisfying*

$$p(x_i) = f(x_i), \qquad i = 0, 1, \dots, n,$$

then for any $x \in [x_0, x_n]$,

$$f(x) - p(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(z) \tag{2.3.6}$$

for some z, depending on x, in the interval (x_0, x_n).

We indicate a proof of this theorem in the Supplementary Discussion of this section; here we only discuss some of its ramifications. First of all, if n is at all large (even 4 or 5), it will probably be difficult, if not impossible, to compute the $(n+1)$th derivative of f. Even if n is only 1 (linear interpolation) and only the second derivative of f is needed, this also may be impossible if f is an unknown function for which values are known only at some discrete points; at best, we might be able to estimate some bound for the second derivative on the basis of our assumed knowledge of f. In any case, it will almost never be the case that (2.3.6) can be used to give a very precise bound on the error. It can, however, be useful in providing insight into the errors that are produced. As an example of this, suppose that the points x_i are equally spaced with spacing h. Then it is easy to see (Exercise 2.3.2.) that

$$|(x-x_0)(x-x_1)\cdots(x-x_n)| \le n!\ h^{n+1}$$

for any x in the interval $[x_0, x_n]$. Thus (2.3.6) can be bounded by

$$|f(x)-p(x)| \le \frac{Mh^{n+1}}{n+1}, \tag{2.3.7}$$

where

$$M = \max_{x_0 \le x \le x_n} |f^{(n+1)}(x)|.$$

Piecewise Polynomials

The bound (2.3.7) is, of course, still difficult to compute because of the quantity M. But it is useful in the following way. Suppose that we wish to approximate the function f over a given interval $[a,b]$ by means of *piecewise polynomials*, that is, functions that are polynomials on given subintervals of $[a,b]$. For example, if $a=\gamma_0<\gamma_1<\cdots<\gamma_p<\gamma_{p+1}=b$ is a partitioning of the interval $[a,b]$, and g is a function that is continuous on $[a,b]$ and is a polynomial on each of the intervals (γ_i,γ_{i+1}), $i=0,1,\cdots,p$, then g is called a piecewise polynomial function on $[a,b]$.

As an example of a piecewise quadratic function, suppose that the values of the function f on the interval $[0,1]$ are given by

x	0	1/6	1/3	1/2	2/3	5/6	1
f	1	3	2	1	0	2	1

Then, the function g defined by

$$\begin{aligned} g(x) &= -54x^2+21x+1, && 0\le x\le \tfrac{1}{3} \\ &= -6x+4, && \tfrac{1}{3}\le x\le \tfrac{2}{3} \\ &= -54x^2+93x-38, && \tfrac{2}{3}\le x\le 1 \end{aligned} \tag{2.3.8}$$

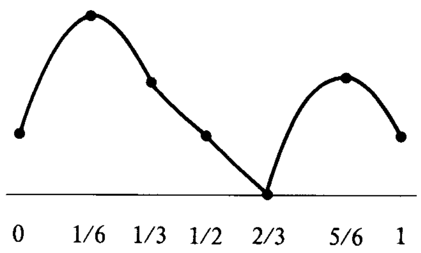


Figure 2.9: *A Piecewise Quadratic Function*

is the piecewise quadratic function on $[0,1]$ that agrees with f at the given nodes, is continuous on the whole interval, and is a quadratic on each of the subintervals $[0,\frac{1}{3}]$, $[\frac{1}{3},\frac{2}{3}]$, $[\frac{2}{3},1]$. This function is shown in Figure 2.9.

Consider now the error in approximating the function f by the function g of (2.3.8). Suppose that M is a bound for the third derivative of f on the entire interval $[0,1]$. Then on each of the intervals $[0,\frac{1}{3}]$, $[\frac{1}{3},\frac{2}{3}]$, and $[\frac{2}{3},1]$, the error bound (2.3.7) can be applied; here $h=\frac{1}{6}$, and $n=2$. Therefore

$$|f(x)-g(x)| \le \frac{h^3M}{3} = \frac{M}{3\cdot 6^3}, \qquad 0\le x\le 1 \tag{2.3.9}$$

Without further information on M, this estimate does not furnish a quantitative bound. It does, however, show how the spacing h enters the error estimate. In particular, approximation by piecewise quadratics has an error estimate that is $0(h^3)$, so that it is third-order. Higher order approximation will result from using piecewise cubics or higher degree polynomials.

The Vandermonde Matrix

Even though the interpolating polynomial is unique, as shown by Theorem 2.3.1, there are several alternative ways to obtain or represent the polynomial, other than by the Lagrange polynomials. Perhaps the most basic approach is the following. Suppose that the interpolation polynomial p is

$$p(x) = a_0 + a_1x + \cdots + a_nx^n.$$

Then we want

$$a_0 + a_1x_i + \cdots + a_nx_i^n = y_i, \qquad i=0,1,\ldots,n. \tag{2.3.10}$$

Since the x_i's and the y_i's are known, this is a system of $n+1$ linear equations in the $n+1$ unknowns $a_0, a_1, \ldots, a_n$. We write this system in the matrix-vector form

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}. \qquad (2.3.11)$$

The coefficient matrix of (2.3.11), which we denote by V, is called the *Vandermonde matrix* and is nonsingular if the x's are distinct. (This statement can be proved directly rather easily, but note that we already have proved it indirectly by means of Theorem 2.3.1, which showed the existence and uniqueness of the interpolating polynomial. For if V were singular, this would imply that either no interpolating polynomial exists for the given data or infinitely many exist.)

The Vandermonde matrix approach is sometimes useful for theoretical purposes, but less so for computation of the polynomial. For the latter the Lagrange polynomials are usually better, but they are not convenient if a node is added or dropped from the data. For example, if (x_{n+1}, y_{n+1}) were added to the set of data (x_i, y_i), $i = 0, 1, \ldots, n$, and we wished to compute the polynomial of degree $n+1$ that interpolated this data, then the Lagrange polynomials would all have to be recomputed. There is another representation of the interpolating polynomial that is very useful in this context; this is the *Newton form*, which we now describe.

The Newton Form

We assume now that the points x_i are equally spaced with spacing h. We define differences of the data y_i by means of $\Delta y_i = y_{i+1} - y_i$, and higher differences by repeated application of this:

$$\begin{aligned} \Delta^2 y_0 &= \Delta y_1 - \Delta y_0 = y_2 - 2y_1 + y_0 \\ \Delta^3 y_0 &= \Delta^2 y_1 - \Delta^2 y_0 = y_3 - 3y_2 + 3y_1 - y_0 \\ &\vdots \\ \Delta^n y_0 &= y_n - \binom{n}{1} y_{n-1} + \binom{n}{2} y_{n-2} - \cdots + (-1)^n y_0, \end{aligned} \qquad (2.3.12)$$

where the binomial coefficients are given by

$$\binom{n}{i} = \frac{n(n-1)\cdots(n-i+1)}{i!}.$$

In terms of the differences (2.3.12), we define a polynomial of degree n by

$$\begin{aligned} p_n(x) &= y_0 + \frac{(x-x_0)}{h}\Delta y_0 + \frac{(x-x_0)(x-x_1)}{2h^2}\Delta^2 y_0 \qquad (2.3.13) \\ &+ \cdots + \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{n!\,h^n}\Delta^n y_0. \end{aligned}$$

Clearly, $p_n(x_0) = y_0$ since all remaining terms in (2.3.13) vanish. Similarly,

$$p_n(x_1) = y_0 + \frac{(x_1 - x_0)}{h}(y_1 - y_0) = y_1,$$

and

$$\begin{aligned} p_n(x_2) &= y_0 + \frac{(x_2 - x_0)}{h}(y_1 - y_0) + \frac{(x_2 - x_0)(x_2 - x_1)}{2h^2}(y_2 - 2y_1 + y_0) \\ &= y_0 + 2(y_1 - y_0) + (y_2 - 2y_1 + y_0) = y_2. \end{aligned}$$

It is easy to verify in an analogous way that $p_n(x_i) = y_i$, $i = 3, \cdots, n$, although the computations become increasingly tedious.

It is of interest to note that the polynomial p_n of (2.3.13) is analogous to the first $n + 1$ terms of a Taylor expansion about x_0. Now suppose that we add (x_{n+1}, y_{n+1}) to the data set. Then the polynomial p_{n+1} that satisfies $p_{n+1}(x_i) = y_i$, $i = 0, 1, \ldots, n + 1$, is given by

$$p_{n+1}(x) = p_n(x) + \frac{(x - x_0)(x - x_1) \cdots (x - x_n)}{(n + 1)!\, h^{n+1}} \Delta^{n+1} y_0,$$

and it is this feature of the Newton form of the interpolating polynomial that is sometimes useful in practice. This is similar to taking one more term in a Taylor expansion.

In the next section we will use interpolating polynomials to derive other methods for the solution of differential equations. This will be done by integrating an interpolating polynomial, but once the polynomial has been determined other manipulations, such as differentiation, can also be performed. Moreover, interpolation is useful for approximating the solution between grid points, if that should be desired. However, it should be pointed out that interpolation may fail to preserve such desirable properties as monotonicity and convexity. It is a more difficult and subtle problem to obtain approximating functions that do maintain such properties.

Supplementary Discussion and References: 2.3

We will indicate the proof of the basic error theorem 2.3.2. Assume that $x \neq x_j$, $j = 0, 1, \ldots, n$; otherwise, both sides of (2.3.6) are zero, and the result is trivially true. Now, for x held fixed, define the function

$$\phi(s) = f(s) - p(s) - q(x)\psi(s),$$

where

$$\psi(s) = (s - x_0)(s - x_1) \cdots (s - x_n), \qquad q(x) = \frac{f(x) - p(x)}{\psi(x)}.$$

It is clear that $\psi(x_i) = 0$, $i = 0, 1, \ldots, n$, and $\phi(x) = 0$; hence ϕ has at least $n + 2$ distinct roots $x_0, x_1, \ldots, x_n, x$. It follows by repeated application of Rolle's theorem that ϕ' has at least $n + 1$ distinct roots, ϕ'' has at least n distinct roots, and so on. In particular, $\phi^{(n+1)}$ has at least one root z in the interval spanned by $x_0, x_1, \ldots, x_n, x$. But

$$\begin{aligned} \phi^{(n+1)}(s) &= f^{(n+1)}(s) - p^{(n+1)}(s) - q(x)\psi^{(n+1)}(s) \\ &= f^{(n+1)}(s) - (n+1)!\, q(x), \end{aligned}$$

since p is a polynomial of degree n and ψ is a polynomial of degree $n+1$. Thus,

$$0 = \phi^{(n+1)}(z) = f^{(n+1)}(z) - (n+1)!\, q(x),$$

and solving for $q(x)$ gives (2.3.6).

For further discussion of interpolation, see, for example, Young and Gregory [1990].

EXERCISES 2.3

2.3.1. Compute the polynomial p of degree 2 that satisfies $p(0) = 0$, $p(1) = 1$, $p(2) = 0$ by all three methods, that is, by using Lagrange polynomials, the Vandermonde matrix, and the Newton representation. Conclude that the polynomial is the same in all three cases.

2.3.2. Verify the bound (2.3.7). Then let $f(x) = \sin \pi x/2$, and let p be the polynomial of Exercise 2.3.1 that agrees with f at the points $x = 0, 1, 2$. Use (2.3.7) to compute a bound for $|f(x) - p(x)|$ on the interval $[0, 2]$. Compare this bound with the actual error at selected points in the interval, and in particular at $x = \frac{1}{4}$ and $\frac{3}{4}$.

2.3.3. Find the piecewise linear and quadratic functions that agree with the following data:

x	0	1/6	1/3	1/2	2/3	5/6	1
f	1	4	1	−1	2	4	0

Compute error bounds for these functions on the interval $[0, 1]$, assuming that the function f satisfies $|f''(z)| \le 4$, $|f'''(z)| \le 10, 0 \le z \le 1$.

2.3.4. Let $f(x) = \sin x$, and let p and q be two polynomials of degree 3 that satisfy $p(k/3) = q(k/3) = f(k/3)$, $k = 0, 1, 2, 3$. Compute a bound for $|p(x) - q(x)|$ that holds on the whole interval $[0, 1]$.

2.3.5. For given $y_0, \ldots, y_n$ and distinct $x_0, \ldots, x_n$, let p be the polynomial of degree n that satisfies $p(x_i) = y_i$, $i = 0, \ldots, n$. Suppose that we wish to write p in the form $p(x) = c_0q_0(x) + \cdots + c_nq_n(x)$, where $q_0(x) \equiv 1$ and $q_i(x) = (x - x_0)\cdots(x - x_i)$. Give an algorithm for finding $c_0, \ldots, c_n$.

2.3.6. Find a polynomial of degree 3 that agrees with $\sqrt{x}$ at $0, 1, 3, 4$. Compare the approximation $p(2)$ with $\sqrt{2} = 1.414216$.

2.3.7. Let p be a polynomial that satisfies $p(-2) = -5$, $p(-1) = 1$, $p(0) = 1$, $p(1) = 1$, $p(2) = 7$, $p(3) = 25$. What can you say about the degree of p?

2.4 Multistep Methods

We return now to the initial-value problem

$$y' = f(x, y), \qquad a \leq x, \qquad y(a) = \hat{y}. \tag{2.4.1}$$

In the methods of Section 2.2, the value of y_{k+1} depended only on information at the previous point, x_k. It seems plausible that more accuracy might be gained if information at several previous points, $x_k, x_{k-1}, \ldots$, were used. Multistep methods do just that.

A large and important class of multistep methods arises from the following approach. If we integrate (2.4.1) for the exact solution $y(x)$ over the interval $[x_k, x_{k+1}]$, we have

$$\begin{aligned} y(x_{k+1}) - y(x_k) &= \int_{x_k}^{x_{k+1}} y'(x)dx = \int_{x_k}^{x_{k+1}} f(x, y(x))dx \\ &\doteq \int_{x_k}^{x_{k+1}} p(x)dx, \end{aligned} \tag{2.4.2}$$

where in the last term we assume that $p(x)$ is a polynomial that approximates $f(x, y(x))$. To obtain this polynomial, suppose that, as in Section 2.2, $y_k, y_{k-1}, \ldots, y_{k-N}$ are approximations to the solution at $x_k, x_{k-1}, \ldots, x_{k-N}$, where we assume that the x_i are equally spaced with spacing h. Then $f_i \equiv f(x_i, y_i)$, $i = k, k-1, \ldots, k-N$, are approximations to $f(x, y(x))$ at $x_k, x_{k-1}, \ldots$. x_{k-N}, and we take p to be the interpolating polynomial for the data set (x_i, f_i), $i = k, k-1, \ldots, k-N$. Thus p is the polynomial of degree N that satisfies $p(x_i) = f_i$, $i = k, k-1, \ldots, k-N$. In principle, we can integrate this polynomial explicitly to give the method

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} p(x)dx. \tag{2.4.3}$$

Adams-Bashforth Methods

As the simplest example, if $N = 0$, then p is the constant f_k and (2.4.3) is simply Euler's method. If $N = 1$, then p is the linear function that interpolates

(x_{k-1}, f_{k-1}) and (x_k, f_k). We may obtain this and the subsequent formulas by applying (2.3.13) backwards from (x_k, f_k); that is, with $\Delta f_k = f_{k-1} - f_k$

$$p(x) = p_1(x) = f_k - \frac{(x - x_k)}{h}\Delta f_k, \tag{2.4.4}$$

where the minus sign occurs since $h = |x_{k-1} - x_k|$. If we integrate (2.4.4) from x_k to x_{k+1}, we obtain the method

$$y_{k+1} = y_k + hf_k - \frac{h}{2}\Delta f_k = y_k + \frac{h}{2}(3f_k - f_{k-1}), \tag{2.4.5}$$

which is a two-step method since it uses information at the two points x_k and x_{k-1}. Note that the first form of (2.4.5) shows how Euler's method is modified to obtain the new method.

Similarly, if $N = 2$, then p is the interpolating quadratic polynomial for (x_{k-2}, f_{k-2}), (x_{k-1}, f_{k-1}), and (x_k, f_k). If we again use (2.3.13), this polynomial may be written as

$$p_2(x) = p_1(x) + \frac{(x - x_k)(x - x_{k-1})}{2h^2}\Delta^2 f_k, \tag{2.4.6}$$

where $\Delta^2 f_k = f_k - 2f_{k-1} + f_{k-2}$. Thus, by (2.4.3), the method is

$$y_{k+1} = y_k + hf_k - \frac{h}{2}\Delta f_k + \frac{5}{6}h\Delta^2 f_k. \tag{2.4.7}$$

This exhibits how the two-step formula (2.4.5) has been modified. We can also collect terms in (2.4.7) and write it as

$$y_{k+1} = y_k + \frac{h}{12}(23f_k - 16f_{k-1} + 5f_{k-2}). \tag{2.4.8}$$

If $N = 3$, the interpolating polynomial is a cubic, and the method is

$$y_{k+1} = y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}). \tag{2.4.9}$$

Note that (2.4.8) is a three-step method, whereas (2.4.9) is a four-step method. It is left to Exercise 2.4.1 to give a detailed verification of the formulas (2.4.5), (2.4.8), and (2.4.9).

The formulas (2.4.5), (2.4.8), and (2.4.9) are known as *Adams-Bashforth methods.* As we shall see later, (2.4.5) is second-order accurate and hence is known as the *second-order Adams-Bashforth method.* Similarly, (2.4.8) and (2.4.9) are the *third-* and *fourth-order Adams-Bashforth methods*, respectively. We can, in principle, continue the preceding process to obtain Adams-Bashforth methods of arbitrarily high order by increasing the number of prior points and the degree of the interpolating polynomial p. The formulas become increasingly complex as N increases, but the principle is still the same.

The Starting Problem

Multistep methods suffer from a problem not encountered with one-step methods. Consider the fourth-order Adams-Bashforth method of (2.4.9). The initial value y_0 is given, but for $k = 0$ in (2.4.9), information is needed at x_{-1}, x_{-2}, and x_{-3}, which doesn't exist. The problem is that multistep methods need "help" getting started. We cannot use (2.4.9) until $k \geq 3$, nor can we use (2.4.8) until $k \geq 2$. The usual tactic is to use a one-step method, such as Runge-Kutta, of the same order of accuracy until enough values have been computed so that the multistep method is usable. Alternatively, one may use a one-step method at the first step, a two-step method at the second, and so on until enough starting values have been built up. However, it is important that the starting values obtained in this fashion be as accurate as those to be produced by the final method, and if the starting methods are of lower order, this will necessitate using a smaller step size and generating more intermediate points at the outset. The same problem arises if the step length is changed during the calculation. If, for example, at x_k the step length is changed from h to $\frac{h}{2}$, then we will need values of y and f at $x_k - \frac{h}{2}$, which we do not have. Again, a Runge-Kutta method starting at x_{k-1} with step length $\frac{h}{2}$ could be used. An attractive alternative is to use an interpolation formula, but care must be exercised to ensure that suitable accuracy is maintained. In general, it is much easier to change step-size with a one-step method than with a multistep method.

Adams-Moulton Methods

The Adams-Bashforth methods were obtained by using information already computed at x_k and prior points. In principle, we can form the interpolating polynomial by using forward points as well. The simplest situation is to use the points $x_{k+1}, x_k, \ldots, x_{k-N}$ and form the interpolating polynomial of degree $N+1$ that satisfies $p(x_i) = f_i$, $i = k+1, k, \ldots, k-N$. This generates a class of methods known as *Adams-Moulton methods.* If $N = 0$, then p is the linear function that interpolates (x_k, f_k) and (x_{k+1}, f_{k+1}), and the corresponding method is

$$y_{k+1} = y_k + \frac{h}{2}(f_{k+1} + f_k), \tag{2.4.10}$$

which is the *second-order Adams-Moulton method.* If $N = 2$, then p is the cubic polynomial that interpolates (x_{k+1}, f_{k+1}), (x_k, f_k), (x_{k-1}, f_{k-1}), and $x_{k-2}, f_{k-2})$; in this case the corresponding method is

$$y_{k+1} = y_k + \frac{h}{24}(9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}), \tag{2.4.11}$$

which is the *fourth-order Adams-Moulton method.*

Note that in the formulas (2.4.10) and (2.4.11) f_{k+1} is not known, since we need y_{k+1} to evaluate $f(x_{k+1}, y_{k+1}) = f_{k+1}$, but y_{k+1} is not yet known. Hence the Adams-Moulton methods define y_{k+1} only implicitly. For example, (2.4.10) is really an equation,

$$y_{k+1} = y_k + \frac{h}{2}[f(x_{k+1}, y_{k+1}) + f_k], \tag{2.4.12}$$

for the unknown value y_{k+1}, and similarly for (2.4.11). Thus the Adams-Moulton methods are called *implicit*, whereas the Adams-Bashforth methods are called *explicit* since no equation needs to be solved in order to obtain y_{k+1}.

Predictor-Corrector Methods

Implicit methods are useful for so-called stiff equations, to be discussed in the next section. However, another use of implicit methods is to combine an explicit with an implicit formula to form a *predictor-corrector method.* A commonly used predictor-corrector method is the combination of the fourth-order Adams methods (2.4.9) and (2.4.11):

$$\begin{aligned} y_{k+1}^{(p)} &= y_k + \frac{h}{24}(55f_k - 59f_{k-1} + 37f_{k-2} - 9f_{k-3}), \\ f_{k+1}^{(p)} &= f(x_{k+1}, y_{k+1}^{(p)}), \\ y_{k+1} &= y_k + \frac{h}{24}(9f_{k+1}^{(p)} + 19f_k - 5f_{k-1} + f_{k-2}),. \end{aligned} \tag{2.4.13}$$

Note that this method is entirely explicit. First a "predicted" value $y_{k+1}^{(p)}$ of y_{k+1} is computed by the Adams-Bashforth formula, then $y_{k+1}^{(p)}$ is used to give an approximate value of f_{k+1}, which is used in the Adams-Moulton formula. The Adams-Moulton formula "corrects" the approximation given by the Adams-Bashforth formula. We could also take additional corrector steps in (2.4.13). In fact, repeated use of the corrector formula gives an iterative method to solve the nonlinear equation (2.4.11). We will return to this topic in Chapter 5.

Discretization Error

We turn now to the question of the discretization error and, for simplicity, we will consider in detail only the Adams-Bashforth method (2.4.5). In a manner analogous to (2.2.23) for one-step methods, we define the local discretization error at x by

$$hL(x,h) = y(x+h) - y(x) - \frac{h}{2}[3f(x, y(x)) - f(x-h, y(x-h))], \tag{2.4.14}$$

where $y(x)$ is the exact solution of the differential equation. Since $y'(x) = f(x, y(x))$, the term in brackets in (2.4.14) can be expanded in a Taylor series as

$$3y'(x) - y'(x-h) = 2y'(x) + hy''(x) + 0(h^2).$$

Combining this with

$$y(x+h) - y(x) = hy'(x) + \frac{h^2}{2}y''(x) + 0(h^3)$$

yields

$$L(x,h) = 0(h^2). \tag{2.4.15}$$

Therefore, assuming that suitably high-order derivatives of the solution are bounded, we have that the local discretization error $L(h)$ on the interval $[a,b]$ satisfies

$$L(h) = \max_{a \le x \le b-h} |L(x,h)| = 0(h^2), \tag{2.4.16}$$

which shows that the method is second order.

We could define the local discretization error separately for each of the other methods of this section. However, all of these methods are special cases of what are called *linear multistep* methods of the form

$$y_{k+1} = \sum_{i=1}^{m} \alpha_i y_{k+1-i} + h \sum_{i=0}^{m} \beta_i f_{k+1-i}, \tag{2.4.17}$$

where, as usual, $f_j = f(x_j, y_j)$, and m is some fixed integer. The method 2.4.17) is called linear since y_{k+1} is a linear combination of the y_i and f_i. If $\beta_0 = 0$, the method is explicit, and if $\beta_0 \neq 0$, then the method is implicit. In all of the Adams methods $\alpha_1 = 1$ and $\alpha_i = 0$, $i > 1$; in the Adams-Bashforth methods $\beta_0 = 0$ and for Adams-Moulton $\beta_0 \neq 0$.

For the general linear multistep method (2.4.17), we define the local discretization error $L(x,h)$ at x by

$$L(x,h) = \frac{1}{h}[y(x+h) - \sum_{i=1}^{m} \alpha_i y(x-(i-1)h)] - \sum_{i=0}^{m} \beta_i y'(x-(i-1)h).$$

For any given method, that is, for any given choice of m and the constants α_i and β_i, one can compute the local discretization error by expansion of y and y' in Taylor series about x. In particular, under suitable assumptions on the differentiability of the solution, one can show that the Adams-Bashforth methods (2.4.8) and (2.4.9) are third and fourth order, respectively, whereas the Adams-Moulton methods (2.4.10) and (2.4.11) are second and fourth order. The verification of these statements is left to Exercise 2.4.6.

Once the local discretization error is known, there remains the problem of bounding the global discretization error, which is defined by

$$E(h) = \max\{|y(x_k) - y_k| : 1 \le k \le N\}.$$

In general, this is a difficult problem, but under suitable assumptions on f and the solution y, it can be shown for all of the methods of this section that $E(h) = 0(h^p)$ when $L(h) = 0(h^p)$.

Multistep methods constitute an attractive alternative to the one-step methods of Section 2.2. High-order methods can be constructed that require only one evaluation of f at each step, but at the price that the methods are not self-starting. Indeed, high-order Adams methods are the basis of the most efficient computer codes available today (see the Supplementary Discussion).

Numerical Examples

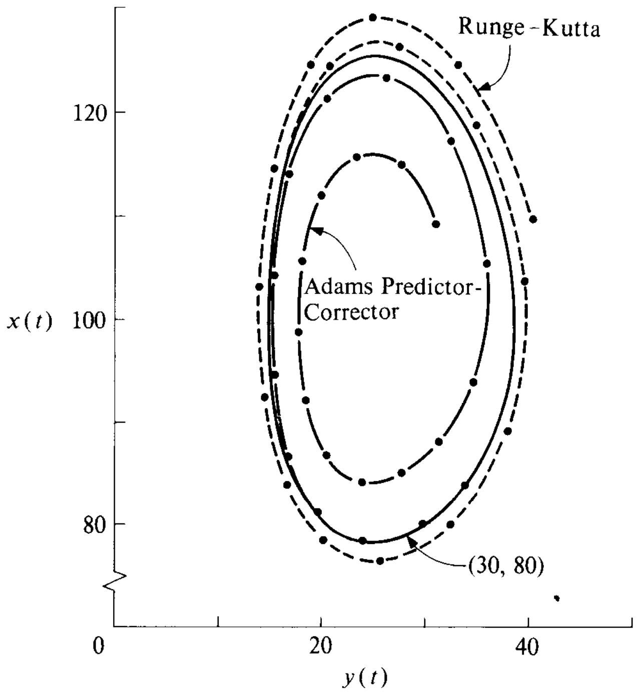


Figure 2.10: *Second-Order Runge-Kutta and Adams Predictor-Corrector*

We conclude this section by returning to the sample problems of Section 2.1. In Figure 2.8 (Section 2.2) we gave an approximate solution of the predator-prey equations (2.2.34), (2.2.35) obtained by the second-order Runge-Kutta method. In Figure 2.10 we superimpose on Figure 2.8 an approximate solution obtained by a second-order Adams-Bashforth/Adams-Moulton predictor-corrector method, which is based on (2.4.5) and (2.4.10) and given explicitly in Exercise 2.4.5. Note that the two methods in Figure 2.10 are in error by a comparable amount, although one approximate solution spirals in from the exact solution whereas the other spirals out. The step size used for both methods was $h = 1$. Both of these methods require two evaluations of f per step. However, the corresponding fourth-order predictor-corrector method still requires only two evaluations of f, whereas the fourth-order Runge-Kutta method requires four.

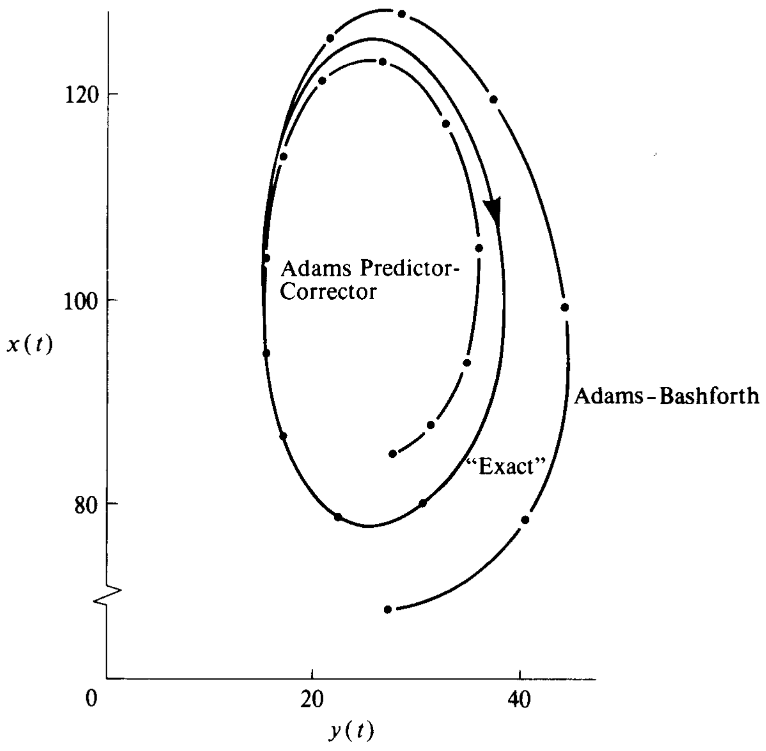


Figure 2.11: *Second-Order Adams-Bashforth and Adams Predictor-Corrector*

Figure 2.11 compares the second-order predictor-corrector method of Fig-re 2.10 with the second-order Adams-Bashforth method. Note the strong ffect that the correction step has on the Adams-Bashforth method; the ac-racy is improved somewhat but, more noticeably, the approximate solution ow spirals in rather than out. Again, the step size for both methods was $= 1$.

We now apply the second-order Adams predictor-corrector method to the trajectory problem discussed in Section 2.1. The system of ordinary differential equations used for the projectile problem is given by (2.1.15), (2.1.17), and (2.1.18) with $T = 0$ and $\dot{m} = 0$:

$$\begin{aligned} \dot{x} &= v\cos\theta, \qquad \dot{y} = v\sin\theta, \\ \dot{v} &= -\frac{1}{2m}c\rho s v^2 - g\sin\theta, \qquad \dot{\theta} = -\frac{g}{v}\cos\theta, \end{aligned} \tag{2.4.18}$$

with initial conditions

$$x(0) = 0, \qquad y(0) = 0, \qquad v(0) = v_0, \qquad \theta(0) = \theta_0. \tag{2.4.19}$$

Values for the parameters in (2.4.20) are $m = 15$ kg, $c = 0.2$, $\rho = 1.29$ kg/m, $s = 0.25$ m^2, and $g = 9.81$ m/s^2. The initial value for v is $v_0 = 50$ m/s, and two different initial angles were used: $\theta_0 = 0.6$ and 1.2 radians. Figure 2.12 is a plot of the height versus range of the projectile.

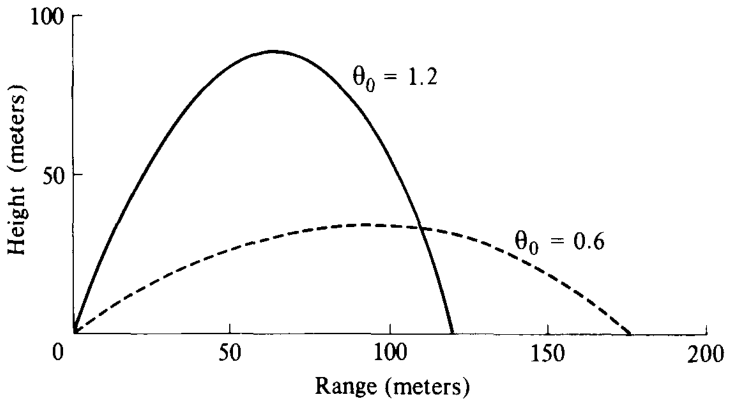


Figure 2.12: *Adams Predictor-Corrector Method* (2.4.18), (2.4.19) *Applied to the Trajectory Problem with Different Initial Angles*

Supplementary Discussion and References: 2.4

The Adams methods form the basis for a number of highly sophisticated computer codes. In these codes the Adams methods are implemented with the capability of changing not only the step size – as we discussed for the Runge-Kutta methods – but also the order of the method. For details on one particular collection of codes – ODEPACK – see Hindmarsh [1983]. For more discussion of the theory and practice of Adams methods, see, for example, Gear [1971] and Butcher [1987]. For an excellent account of how good computer codes are developed, see Shampine, Watts, and Davenport [1976]; this article makes the important point that the way methods are implemented on a computer can be more important than the intrinsic difference between methods.

Another approach to the derivation of multistep methods starts with the general linear method (2.4.17) and requires that it be exact when the solution y of the differential equation is a polynomial of degree q. This then implies that the method is order q. For example, if $q = 1$, then (2.4.17) must be exact whenever the solution is a constant; in this case the f_i all vanish since $f(x, y(x)) = y'(x) = 0$, and we are left with the condition

$$1 = \sum_{i=1}^{m} \alpha_i. \tag{2.4.20}$$

Similarly, the requirement that (2.4.17) be exact when the solution is $y(x) = x$ leads to the condition

$$m + 1 = \alpha_1 m + \alpha_2(m-1) + \cdots + \alpha_m + \sum_{i=0}^{m} \beta_i. \tag{2.4.21}$$

The relations (2.4.20) and (2.4.21) for the coefficients α_i and β_i are known as the *consistency conditions* for the multistep method and are necessary and sufficient conditions that the method be first order. One can continue this process to obtain relations on the α_i and β_i that are necessary and sufficient that the method be of any given order. For further discussions of multistep methods, see, for example, Henrici [1962] and Butcher [1987].

We can combine the one-step methods of Section 2.2 with the multistep methods of this section into the same general formulation:

$$y_{k+1} = \sum_{i=1}^{m} \alpha_i y_{k+1-i} + h\phi(x_{k+1}, \ldots, x_{k+1-m}; y_{k+1}, \ldots, y_{k+1-m}). \tag{2.4.22}$$

For one-step methods $m = 1$, and if $\alpha_1 = 1$ and ϕ is independent of x_{k+1} and y_{k+1}, then (2.4.22) reduces to the one-step method (2.2.24). On the other hand, if ϕ is the function

$$\phi = \sum_{i=0}^{m} \beta_i f(x_{k+1-i}, y_{k+1-i}),$$

then (2.4.22) reduces to the linear multistep method (2.4.17). The formulation (2.4.22) contains virtually all methods in current use.

EXERCISES 2.4

2.4.1. Verify that p_1 of (2.4.4) is the linear interpolating polynomial for (x_k, f_k) and (x_{k-1}, f_{k-1}), and that (2.4.5) follows from (2.4.3) for this p_1. Similarly, verify that (2.4.6) is the quadratic interpolating polynomial for (x_k, f_k), (x_{k-1}, f_{k-1}), and (x_{k-2}, f_{k-2}). Then carry out the integration of p_2 to verify the formula (2.4.7). Finally, give the cubic interpolating polynomial p_3 by adding the appropriate cubic term (see (2.3.13)) to p_2. Then integrate p_3 in (2.4.3) to obtain the formula (2.4.9).

2.4.2. Write a computer program to carry out the second-order Adams-Bashforth method (2.4.5). Use the second-order Runge-Kutta method to supply the missing starting value y_1. Apply your program to the problems of Exercises 2.2.2 and 2.2.8 and compare your results with the Euler and Heun methods.

2.4.3. Repeat Exercise 2.4.2 using the fourth-order Adams-Bashforth method (2.4.9).

2.4.4. Carry out in detail the derivation of the Adams-Moulton method (2.4.10). Do the same for the method (2.4.11).

2.4.5. Use as much as possible of your program of Exercise 2.4.2 to write a computer program to carry out the predictor-corrector method

$$\begin{aligned} y_{k+1}^{(p)} &= y_k + \frac{h}{2}(3f_k - f_{k-1}), \\ f_{k+1}^{(p)} &= f(x_{k+1}, y_{k+1}^{(p)}), \\ y_{k+1} &= y_k + \frac{h}{2}(f_{k+1}^{(p)} + f_k). \end{aligned}$$

Apply this to the problem $y' = -y$, $y(0) = 1$ and compare your results with the methods of Exercise 2.4.2. Similarly, write a program to carry out (2.4.13).

2.4.6. Compute the local discretization errors for the Adams-Bashforth methods (2.4.8) and (2.4.9) and show that they are third and fourth order, respectively. (Assume that the solution is sufficiently differentiable.) Do the same for the Adams-Moulton methods (2.4.10) and (2.4.11) and verify that they are second and fourth order, respectively.

2.4.7. Give the coefficients α_i and β_i in the linear multistep formulation (2.4.17) for the Adams-Bashforth and Adams-Moulton methods of second, third, and fourth order.

2.4.8. Consider the method $y_{k+1} = y_{k-1} + \frac{h}{2}(f_{k+1} + 2f_k + f_{k-1})$.

a. Find the order of the method.

b. Discuss how to apply this method to the *system* of equations $\mathbf{y}' = \mathbf{f}(x, \mathbf{y})$. What difficulties do you expect to encounter in carrying out the method?

2.4.9. Repeat the calculations of Figure 2.11 using the second-order Adams-Bashforth method and the predictor-corrector method of Exercise 2.4.5.

2.4.10. Repeat the calculations of Figure 2.12. Find the value of θ_0 such that the range of the rocket is 150 m.

2.4.11. Consider the problem of evaluating the "normal density"

$$p(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2/2} dt + \frac{1}{2}$$

by solving the differential equation

$$p'(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \qquad p(0) = \frac{1}{2}.$$

Use the second-order Runge-Kutta method and the second-order predictor-corrector method of Exercise 2.4.5 to solve this differential equation. Compare your results.

2.5 Stability, Instability, and Stiff Equations

One of the pervading concerns of scientific computing is that of *stability*, a much overused word that tends to have somewhat different meanings depending on the context. In this section we will discuss several aspects of stability as it pertains to the numerical solution of ordinary differential equations.

Unstable Solutions

Consider the second-order differential equation

$$y'' - 10y' - 11y = 0 \tag{2.5.1}$$

with the initial conditions

$$y(0) = 1, \qquad y'(0) = -1. \tag{2.5.2}$$

The solution of (2.5.1), (2.5.2) is $y(x) = e^{-x}$, as is easily verified. Now suppose we change the first initial condition by a small quantity ε, so that the initial conditions are

$$y(0) = 1 + \varepsilon, \qquad y'(0) = -1. \tag{2.5.3}$$

Then, as is again easily verified, the solution of (2.5.1) with the initial conditions (2.5.3) is

$$y(x) = (1 + \tfrac{11}{12}\varepsilon)e^{-x} + \frac{\varepsilon}{12} e^{11x}. \tag{2.5.4}$$

Therefore for any $\varepsilon > 0$, no matter how small, the second term in (2.5.4) causes the solution to tend to infinity as $x \to \infty$. The two solutions are shown in Figure 2.13. We say that the solution $y(x) = e^{-x}$ of the problem (2.5.1), (2.5.2) is *unstable*, since arbitrarily small changes in the initial conditions can produce arbitrarily large changes in the solution as $x \to \infty$. In the parlance of numerical analysis, one would also say that this problem is *ill-conditioned*; it is extremely difficult to obtain the solution numerically because rounding and discretization error will cause the same effect as changing the initial conditions, and the approximate solution will tend to diverge to infinity (see Exercise 2.5.1).

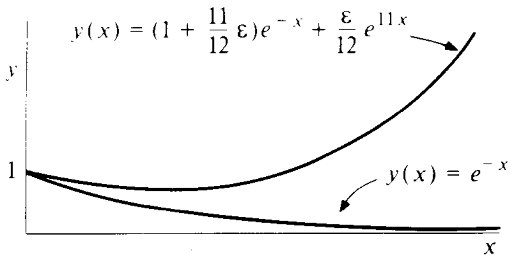


Figure 2.13: *Solutions of Slightly Different Problems*

Even more pronounced instabilities can occur with nonlinear equations. For example, the problem

$$y' = xy(y-2), \qquad y(0) = 2, \tag{2.5.5}$$

has the solution $y(x) \equiv 2$, which is unstable. To see this, note that for the initial condition $y(0) = y_0$ the solution is

$$y(x) = \frac{2y_0}{y_0 + (2 - y_0)e^{x^2}} .$$

Thus if $y_0 < 2$, then $y(x) \to 0$ as $x \to \infty$, and if $y_0 > 2$, the solution increases and has a singularity when $y_0 + (2 - y_0)e^{x^2} = 0$. Typical solutions are shown in Figure 2.14.

Unstable Methods

The two previous examples illustrated instabilities of solutions of the differential equation itself. We now turn to possible instabilities in the numerical method. Let us consider the method

$$y_{n+1} = y_{n-1} + 2hf_n, \tag{2.5.6}$$

which is similar to Euler's method but is a multistep method and is second-order accurate, as is easy to verify (Exercise 2.5.3).

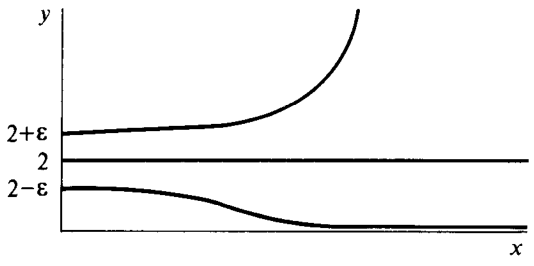


Figure 2.14: *Solutions for Three Slightly Different Initial Conditions*

We now apply (2.5.6) to the problem

$$y' = -2y + 1, \qquad y(0) = 1, \tag{2.5.7}$$

whose exact solution is

$$y(x) = \tfrac{1}{2}e^{-2x} + \tfrac{1}{2}. \tag{2.5.8}$$

This solution is stable, since if the initial condition is changed to $y(0) = 1 + \varepsilon$ the solution becomes

$$y(x) = (\tfrac{1}{2} + \varepsilon)e^{-2x} + \tfrac{1}{2},$$

and the change in (2.5.8) is only εe^{-2x}. The method (2.5.6) applied to (2.5.7) is

$$y_{n+1} = y_{n-1} + 2h(-2y_n + 1) = -4hy_n + y_{n-1} + 2h, \quad y_0 = 1, \tag{2.5.9}$$

with y_0 taken as the initial condition. However, since (2.5.6) is a two-step method, we also need to supply y_1 in order to start the process, and we will take y_1 to be the exact solution (2.5.8) at $x = h$:

$$y_1 = \tfrac{1}{2}e^{-2h} + \tfrac{1}{2}. \tag{2.5.10}$$

If, for any fixed $h > 0$, we now generate the sequence $\{y_n\}$ by (2.5.9) and (2.5.10), we will see that $|y_n| \to \infty$ as $n \to \infty$, rather than mirroring the behavior as $x \to \infty$ of the solution (2.5.8) of the differential equation. Thus the method (2.5.6), although second-order accurate, exhibits unstable behavior and we next wish to address why this is so.

Difference Equations

It is relatively easy to analyze the behavior of the sequence $\{y_n\}$ generated by (2.5.9) by viewing (2.5.9) as a *difference equation.* The theory of difference equations parallels that of differential equations, and we will sketch the basic parts of this theory in the case of *linear difference equations of order m with constant coefficients*; such an equation is of the form

$$y_{n+1} = a_m y_n + \cdots + a_1 y_{n-m+1} + a_0, \qquad n = m-1, m, m+1, \ldots, \tag{2.5.11}$$

with given constants $a_0, a_1, \ldots, a_m$. The *homogeneous part* of (2.5.11) is

$$y_{n+1} = a_m y_n + \cdots + a_1 y_{n-m+1}. \tag{2.5.12}$$

By analogy with differential equations, we attempt to find an exponential type solution of (2.5.12), only now the exponential takes the form $y_k = \lambda^k$ for some unknown constant λ. We see that $y_k = \lambda^k$ indeed is a solution of (2.5.12) provided that λ satisfies

$$\lambda^m - a_m \lambda^{m-1} - \cdots - a_1 = 0, \tag{2.5.13}$$

which is the *characteristic equation* of (2.5.12). If we assume that the m roots $\lambda_1, \ldots, \lambda_m$ of (2.5.13) are distinct, then the *fundamental solutions* of (2.5.12) are $\lambda_1^k, \ldots, \lambda_m^k$, and the general solution of (2.5.12) is

$$y_k = \sum_{i=1}^{m} c_i \lambda_i^k, \qquad k = 0, 1, \ldots, \tag{2.5.14}$$

where the c_i are arbitrary constants. A *particular solution* of (2.5.11) is given by

$$y_k = \frac{a_0}{1 - a_1 - \cdots - a_m}, \tag{2.5.15}$$

provided that the denominator is not zero, as is easily verified. Therefore the general solution of (2.5.11) is the sum of (2.5.14) and (2.5.15):

$$y_k = \sum_{i=1}^{m} c_i \lambda_i^k + \frac{a_0}{1 - a_1 - \cdots - a_m}, \qquad k = 0, 1 \ldots . \tag{2.5.16}$$

The arbitrary constants in (2.5.16) can be determined – just as for differential equations – by imposing additional conditions on the solution. In particular, suppose we are given the initial conditions

$$y_0, y_1, \ldots, y_{m-1}. \tag{2.5.17}$$

Then (2.5.16) requires that

$$\sum_{i=1}^{m} c_i \lambda_i^k + \frac{a_0}{1 - a_1 - \cdots - a_m} = y_k, \quad k = 0, 1, \ldots m-1, \tag{2.5.18}$$

which is a system of m linear equations in the m unknowns $c_1, \ldots, c_m$, and can be used to determine the c_i.

We now apply this theory to the difference equation (2.5.9), which we write in the form

$$y_{n+1} = -4hy_n + y_{n-1} + 2h, \quad y_0 = 1, \; y_1 = \tfrac{1}{2}e^{-2h} + \tfrac{1}{2}. \tag{2.5.19}$$

The characteristic equation (2.5.13) is $\lambda^2 + 4h\lambda - 1 = 0$ with roots

$$\lambda_1 = -2h + \sqrt{1+4h^2}, \quad \lambda_2 = -2h - \sqrt{1+4h^2}. \tag{2.5.20}$$

The conditions (2.5.18) then become

$$c_1 + c_2 + \tfrac{1}{2} = y_0 = 1, \qquad c_1\lambda_1 + c_2\lambda_2 + \tfrac{1}{2} = y_1 = \tfrac{1}{2}e^{-2h} + \tfrac{1}{2},$$

which can be solved for c_1 and c_2 to give

$$c_1 = \tfrac{1}{4} + \frac{y_1 - \frac{1}{2} + h}{2\sqrt{1+4h^2}}, \qquad c_2 = \tfrac{1}{4} - \frac{(y_1 - \frac{1}{2} + h)}{2\sqrt{1+4h^2}}. \tag{2.5.21}$$

Thus the solution of (2.5.19) is

$$y_n = c_1(-2h + \sqrt{1+4h^2})^n + c_2(-2h - \sqrt{1+4h^2})^n + \tfrac{1}{2}. \tag{2.5.22}$$

Although this representation of the solution is perhaps a little formidable, it allows us to see very easily the behavior of y_n as $n \to \infty$. In particular, for any fixed step size $h > 0$, it is evident that

$$0 < -2h + \sqrt{1+4h^2} < 1, \qquad 2h + \sqrt{1+4h^2} > 1.$$

Therefore the first term in (2.5.22) tends to zero, while the second tends to infinity, in an oscillatory way, as n tends to infinity. Since the exact solution (2.5.8) of the differential equation tends to $1/2$ as x tends to infinity, we see that the error in the approximate solution $\{y_n\}$ diverges to infinity, and the method (2.5.9) is unstable applied to the problem (2.5.7). Note that this divergence of the error has nothing to do with rounding error; (2.5.22) is the exact mathematical representation of y_n, and if the sequence (2.5.19) were computed in exact arithmetic it would correspond precisely with that given by (2.5.22).

Stability of Methods

From the preceding example it is clear that an important property of a method is that it be stable in some sense. The most basic definition of stability may be given in terms of the general method (2.4.22):

$$y_{n+1} = \sum_{i=1}^{m} \alpha_i y_{n+1-i} + h\phi(x_{n+1}, \ldots, x_{n+1-m}, y_{n+1}, \ldots, y_{n+1-m}). \tag{2.5.23}$$

The method (2.5.23) is *stable* provided that all roots λ_i of the polynomial

$$\rho(\lambda) \equiv \lambda^m - \alpha_1\lambda^{m-1} - \cdots - \alpha_m \tag{2.5.24}$$

satisfy $|\lambda_i| \leq 1$, and any root for which $|\lambda_i| = 1$ is simple. The method is *strongly stable* if, in addition, $m-1$ roots of (2.5.24) satisfy $|\lambda_i| < 1$. (We note that some authors use the terms *weakly stable* and *stable* in place of stable and strongly stable.)

Any method that is at least first-order accurate must satisfy the condition $\sum_{i=1}^{m}\alpha_i = 1$, so that 1 is a root of (2.5.24). In this case, a strongly stable method will then have one root of (2.5.24) equal to 1 and all the rest strictly less than 1 in absolute value. For Runge-Kutta methods, $\rho(\lambda) = \lambda - 1$ since the method is one-step; hence there are no roots of (2.5.24) besides the root $\lambda = 1$, and these methods are always strongly stable. For an m-step Adams method, $\rho(\lambda) = \lambda^m - \lambda^{m-1}$, so the other $m-1$ roots of (2.5.24) are zero, and these methods also are strongly stable.

For the method (2.5.6), the polynomial (2.5.24) is $\rho(\lambda) = \lambda^2 - 1$ with roots ± 1; hence this method is stable but not strongly stable, and it is this lack of strong stability that gives rise to the unstable behavior of the sequence $\{y_k\}$ defined by (2.5.19). The reason for this is as follows. The difference equation (2.5.19) is second order (since y_{n+1}, y_n, and y_{n-1} appear in the equation) and has two fundamental solutions, λ_1^n and λ_2^n, where λ_1 and λ_2 are the roots (2.5.20). The sequence $\{y_k\}$ generated by (2.5.19) is meant to approximate the solution of the differential equation (2.5.7), which is a first-order equation with only one fundamental solution. This fundamental solution is approximated by λ_1^n; λ_2^n is spurious and should rapidly go to zero. However, for any $h > 0$, $|\lambda_2| > 1$, and hence λ_2^n tends to infinity and not zero; it is this that causes the instability. Now, note that λ_1 and λ_2 converge to the roots of the stability polynomial (2.5.24) as $h \to 0$; indeed, this polynomial is just the limit, as $h \to 0$, of the characteristic polynomial $\lambda^2 + 4h\lambda - 1$ of (2.5.19). The idea of strong stability now becomes more evident. If all the roots except one of the stability polynomial are less than 1 in magnitude, then all but one of the roots of the characteristic equation of the method must be less than 1 for sufficiently small h; hence powers of these roots – the spurious fundamental solutions of the difference equation – tend to zero and cause no instability.

The stability theory that we have just discussed is essentially stability in the limit as $h \to 0$, and the example of instability that we gave shows what can happen for arbitrarily small h if the method is stable but not strongly stable and the interval is infinite. On a finite interval, a stable method will give accurate results for sufficiently small h. On the other hand, even strongly stable methods can exhibit unstable behavior if h is too large. Although in principle h can be taken sufficiently small to overcome this difficulty, it may be that the computing time then becomes prohibitive. This is the situation with differential equations that are known as *stiff*, and we shall conclude this section with a short discussion of such problems.

Stiff Equations

Consider the equation

$$y' = -100y + 100, \qquad y(0) = y_0. \tag{2.5.25}$$

The exact solution of this problem is

$$y(x) = (y_0 - 1)e^{-100x} + 1. \tag{2.5.26}$$

It is clear that the solution is stable, since if we change the initial condition to $y_0 + \varepsilon$ the solution changes by εe^{-100x}. Euler's method applied to (2.5.25) is

$$y_{n+1} = y_n + h(-100y_n + 100) = (1 - 100h)y_n + 100h, \tag{2.5.27}$$

and the exact solution of this first-order difference equation is

$$y_n = (y_0 - 1)(1 - 100h)^n + 1. \tag{2.5.28}$$

For concreteness, suppose that $y_0 = 2$ so that the exact solutions (2.5.26) and (2.5.28) become

$$y(x) = e^{-100x} + 1, \tag{2.5.29}$$

$$y_n = (1 - 100h)^n + 1. \tag{2.5.30}$$

Now, $y(x)$ decreases very rapidly from $y_0 = 2$ to its limiting value of 1; for example, $y(0.1) \doteq 1 + 5 \times 10^{-5}$. Initially, therefore, we expect to require a small step size h to compute the solution accurately. However, beyond, say, $x = 0.1$, the solution varies slowly and is essentially equal to 1; intuitively we would expect to obtain sufficient accuracy with Euler's method using a relatively large h. However, we see from (2.5.30) that if $h > 0.02$, then $|1 - 100h| > 1$ and the approximation y_n grows rapidly at each step and shows an unstable behavior. If we compare the exact solutions (2.5.29) and (2.5.30), we see that the particular solutions of (2.5.25) and (2.5.27) are identical (and equal to 1). The quantity $(1 - 100h)^n$ is an approximation to the exponential term e^{-100x} and is, indeed, a good approximation for small h but rapidly becomes a poor approximation as h becomes as large as 0.02. Even though this exponential term contributes virtually nothing to the solution after $x = 0.1$, Euler's method still requires that we approximate it with sufficient accuracy to maintain stability. This is the typical problem with stiff equations: the solution contains a component that contributes very little to the solution, but the usual methods require that it be approximated accurately in order to maintain stability.

This problem occurs very frequently in systems of equations. For example, consider the second-order equation

$$y'' + 101y' + 100y = 0. \tag{2.5.31}$$

As discussed in Appendix 1, we can convert (2.5.31) to an equivalent system of two first-order equations, but it is sufficient for our purposes to treat it in its second-order form. The general solution of (2.5.31) is

$$y(x) = c_1 e^{-100x} + c_2 e^{-x},$$

and if we impose the initial conditions

$$y(0) = 1.01, \qquad y'(0) = -2,$$

the solution is

$$y(x) = \tfrac{1}{100} e^{-100x} + e^{-x}. \tag{2.5.32}$$

Clearly, the first term of this solution contributes very little after x reaches a value such as $x = 0.1$. Yet we will have the same problem as in the previous example if we apply Euler's method to the first-order system corresponding to (2.5.31); that is, we will need to make the step size sufficiently small to approximate e^{-100x} accurately even though this term contributes very little to the solution. This example illustrates the essence of the problem of stiffness in systems of equations. Usually the independent variable in such problems is time and the physical problem that is being modeled has transients that decay to zero very rapidly, but the numerical scheme must cope with them even after they no longer contribute to the solution.

The general approach to the problem of stiffness is to use implicit methods. It is beyond the scope of this book to discuss this in detail, and we will only give an indication of the value of implicit methods in this context by applying one of the simplest such methods to the problem (2.5.25). For the general equation $y' = f(x, y)$, the method

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \tag{2.5.33}$$

is known as the *backward Euler method.* It is of the same form as Euler's method except that f is evaluated at (x_{n+1}, y_{n+1}) rather than at (x_n, y_n); hence the method is implicit. If we apply (2.5.33) to (2.5.25), we obtain

$$y_{n+1} = y_n + h(-100y_{n+1} + 100), \tag{2.5.34}$$

which can be put in the form

$$y_{n+1} = (1 + 100h)^{-1}(y_n + 100h). \tag{2.5.35}$$

The exact solution of the difference equation (2.5.35) is

$$y_n = (y_0 - 1)(1 + 100h)^{-n} + 1, \tag{2.5.36}$$

as is easily verified (Exercise 2.5.10). In particular, for the initial condition $y_0 = 2$, which was treated previously, (2.5.36) becomes

$$y_n = \frac{1}{(1 + 100h)^n} + 1, \tag{2.5.37}$$

and we see that there is no unstable behavior regardless of the size of h. Note that with Euler's method we are attempting to approximate the solution by a polynomial, and no polynomial (except 0) can approximate e^{-x} as $x \to \infty$. With the backward Euler's method we are approximating the solution by a rational function, and such functions can indeed go to zero as $x \to \infty$.

The backward Euler method, like Euler's method itself, is only first-order accurate, and a better choice would be the second-order Adams-Moulton method (2.4.10):

$$y_{n+1} = y_n + \frac{h}{2}[f_n + f(x_{n+1}, y_{n+1})], \tag{2.5.38}$$

which is also known as the *trapezoid rule*. The use of this method on (2.5.25) is left to Exercise 2.5.13.

The application of an implicit method to (2.5.25) was deceptively simple since the differential equation is linear and hence we could easily solve for y_{n+1} in (2.5.34). If the differential equation had been nonlinear, however, the method would have required the solution of a nonlinear equation for y_{n+1} at each step. More generally, for a system of differential equations the solution of a system of equations (linear or nonlinear, depending on the differential equations) would be needed at each step. This is costly in computer time, but the effective handling of stiff equations requires that some kind of implicitness be brought into the numerical method.

Supplementary Discussion and References: 2.5

There is a vast literature on the theory of stability of solutions of differential equations. For a readable introduction, see LaSalle and Lefschetz [1961].

We have given the basic result for linear difference equations with constant coefficients only for the case where the roots of the characteristic equation are distinct. If there are multiple roots, then polynomial terms in n enter the solution in a manner entirely analogous to that for differential equations. For more discussion of the theory of linear difference equations, see, for example, Ortega [1987].

The method (2.5.6) arises in a natural way by differentiation of an interpolating polynomial; for the derivation, see Henrici [1962, p. 219].

The basic results of the theory of stability of multistep methods were developed by G. Dahlquist in the 1950s; for a detailed treatment of this theory, see Henrici [1962]. Since then there have been a number of refined definitions of stability; in particular, the terms stiffly-stable and A-stable deal with types of stability needed for methods to handle stiff equations. For more on different definitions of stability, see, for example, Gear [1971], Lambert [1973], and Butcher [1987].

There are a number of codes available for solving stiff equations. One of the best is the VODE package of Brown, Byrne, and Hindmarsh [1989], which may also be used for problems which are not stiff. For stiff equations a so-called Backward Differention Formula (BDF) implicit method is used, which is of the general form (2.4.17) with $\beta_i = 0$ for $i > 0$.

EXERCISES 2.5

2.5.1. By letting $z = y'$, show that the problem (2.5.1), (2.5.2) is equivalent to the first-order system $y' = z$, $z' = 10z + 11y$, with initial conditions $y(0) = 1$ and $z(0) = -1$. Attempt to solve this system numerically by any of the methods of this chapter and discuss your results.

2.5.2. Attempt to solve the problem (2.5.5) numerically by any of the methods of Sections 2.2 or 2.4 and discuss your results.

2.5.3. Verify that the method (2.5.6) is second-order accurate.

2.5.4. Carry out the algorithm (2.5.9), (2.5.10) numerically for various values of h. Discuss your results.

2.5.5. Solve the difference equation $y_{n+1} = \frac{5}{2}y_n + y_{n-1}$, $y_0 = y_1 = 1$, in terms of the roots of its characteristic equation. Discuss the behavior of the sequence $\{y_n\}$ as $n \to \infty$.

2.5.6. Find a value of y_1 such that the resulting solution of (2.5.9) with $y_0 = 1$ tends to zero as n tends to infinity. Write a program to carry out (2.5.9) with y_1 given in this way as well as by (2.5.10). Discuss your results.

2.5.7. Consider the method $y_{n+1} = y_{n-3} + (4h/3)(2f_n - f_{n-1} + 2f_{n-2})$, which is known as *Milne's method.* Ascertain whether this method is stable and strongly stable.

2.5.8. Write a program to carry out Euler's method (2.5.27) for different values of h both less than and greater than 0.02. Discuss your results.

2.5.9. The system $y' = x$, $z' = -100y - 101z$ is the first-order system equivalent to the second-order equation (2.5.31). Using the initial conditions $y(0) = 2$ and $z(0) = -2$, apply Euler's method to this system and determine experimentally how small the step size h must be to maintain stability. Attempt to verify analytically your conclusion about the size of h.