Chapter 1.
Ordinary Differential Equations

Just as the twig is bent, the tree's inclined.
— ALEXANDER POPE, *Moral Essays (1734)*

The central topic of this book is time-dependent partial differential equations (PDEs). Even when we come to discuss elliptic PDEs, which are not time-dependent, we shall find that some of the best numerical methods are iterations that behave very much as if there were a time variable. That is why elliptic equations come at the end of the book rather than the beginning.

Ordinary differential equations (ODEs) embody one of the two essential ingredients of this central topic: variation in time, but not in space. Because the state at any moment is determined by a finite set of numbers, ODEs are far easier to solve and far more fully understood than PDEs. In fact, an ODE usually has to be nonlinear before its behavior becomes very interesting—which is not the case at all for a PDE.

The solution of ODEs by discrete methods is one of the oldest and most successful areas of numerical computation. A dramatic example of the need for such computations is the problem of predicting the motions of planets or other bodies in space. The governing differential equations have been known since Newton, and for the case of only two bodies, such as a sun and a planet, Newton showed how to solve them exactly. In the three centuries since then, however, no one has ever found an exact solution for the case of three or more bodies.* Yet the numerical calculation of such orbits is almost effortless by modern standards, and is a routine component of the control of spacecraft.

The most important families of numerical methods for ODEs are

- linear multistep methods,
- Runge-Kutta methods.

This chapter presents the essential features of linear multistep methods, with emphasis on the fundamental ideas of accuracy, stability, and convergence. An important distinction is made between "classical" stability and "eigenvalue stability". All of these notions will prove indispensable when we move on to discuss PDEs in Chapter 3. Though we do not describe Runge-Kutta methods except briefly in §1.8, most of the analytical ideas described here apply with minor changes to Runge-Kutta methods too.

*In fact there are theorems to the effect that such an exact solution can never be found. This unpredictability of many-body motions is connected with the phenomenon of chaos.

1.1. Initial value problems

An **ordinary differential equation**, or **ODE**, is an equation of the form

$$u_t(t) = f(u(t),t), \qquad (1.1.1)$$

where t is the time variable, u is a real or complex scalar or vector function of t ($u(t) \in \mathbb{C}^N$, $N \geq 1$), and f is a function that takes values in $\mathbb{C}^N$.* We also say that (1.1.1) is a **system of ODEs of dimension** N. The symbol u_t denotes du/dt, and if $N > 1$, it should be interpreted componentwise: $(u^{(1)},\dots,u^{(N)})_t^T = (u_t^{(1)},\dots,u_t^{(N)})^T$. Similarly, u_{tt} denotes d^2u/dt^2, and so on. Where clarity permits, we shall leave out the arguments in equations like (1.1.1), which then becomes simply $u_t = f$.

The study of ODEs goes back to Newton and Leibniz in the 1670's, and like so much of mathematics, it owes a great deal also to the work of Euler in the 18th century. Systems of ODEs were first considered by Lagrange in the 1750s, but the use of vector notation did not become standard until around 1890.

If $f(u,t) = \alpha(t)u + \beta(t)$ for some functions $\alpha(t)$ and $\beta(t)$, the ODE is **linear**, and if $\beta(t) = 0$ it is linear and **homogeneous**. (In the vector case $\alpha(t)$ is an $N \times N$ matrix and $\beta(t)$ is an N-vector.) Otherwise it is **nonlinear.** If $f(u,t)$ is independent of t, the ODE is **autonomous**. If $f(u,t)$ is independent of u, the ODE reduces to an indefinite integral.

To make (1.1.1) into a fully specified problem, we shall provide **initial data** at $t = 0$ and look for solutions on some interval $t \in [0,T]$, $T > 0$. The choice of $t = 0$ as a starting point introduces no loss of generality, since any other t_0 could be treated by the change of variables $t' = t - t_0$.

Initial Value Problem. *Given f as described above, $T > 0$, and $u_0 \in \mathbb{C}^N$, find a differentiable function $u(t)$ defined for $t \in [0,T]$ such that*

(a) $u(0) = u_0$,

(b) $u_t(t) = f(u(t),t)$ *for all* $t \in [0,T]$. (1.1.2)

*$\mathbb{C}^N$ is the space of complex column vectors of length N. In practical ODE problems the variables are usually real, so that for many purposes we could write $\mathbb{R}^N$ instead. When we come to Fourier analysis of linear partial differential equations, however, the use of complex variables will be very convenient.

Numerical methods for solving ODE initial-value problems are the subject of this chapter. We shall not discuss boundary-value problems, in which various components of u are specified at two or more distinct points of time; see Keller (1978) and Ascher, et al. (1988).

EXAMPLE 1.1.1. The scalar initial-value problem $u_t = Au$, $u(0) = a$ has the solution $u(t) = ae^{tA}$, which decays to 0 as $t \to \infty$ provided that $A < 0$, or $\operatorname{Re} A < 0$ if A is complex. This ODE is linear, homogeneous, and autonomous.

EXAMPLE 1.1.2. The example above becomes a vector initial-value problem if u and a are N-vectors and A is an $N \times N$ matrix. The solution can still be written $u(t) = e^{tA}a$, if e^{tA} now denotes the $N \times N$ matrix defined by applying the usual Taylor series for the exponential to the matrix tA. For generic initial vectors a, this solution $u(t)$ decays to the zero vector as $t \to \infty$, i.e. $\lim_{t\to\infty} \|u(t)\| = 0$, if and only if each eigenvalue λ of A satisfies $\operatorname{Re}\lambda < 0$.

EXAMPLE 1.1.3. The scalar initial-value problem $u_t = u\cos t$, $u(0) = 1$ has the solution $u(t) = e^{\sin t}$. One can derive this by separation of variables by integrating the equation $du/u = \cos t\, dt$.

EXAMPLE 1.1.4. The nonlinear initial-value problem $u_t = u + u^2$, $u(0) = 1$ has the solution $u(t) = 1/(2e^{-t} - 1)$, which is valid until the solution becomes infinite at $t = \log 2 \approx 0.693$. This ODE is an example of a Bernoulli differential equation. One can derive the solution by the substitution $w(t) = 1/u(t)$, which leads to the linear initial-value problem $w_t = -1 - w$, $w(0) = 1$, with solution $w(t) = 2e^{-t} - 1$.

EXAMPLE 1.1.5. The Lorenz equations, with the most familiar choice of parameters, can be written

$$u_t = -10u + 10v, \qquad v_t = 28u - v - uw, \qquad w_t = -\tfrac{8}{3}w + uv.$$

This is the classical example of a nonlinear system of ODEs whose solutions are chaotic. The solution cannot be written in closed form.

Equation (1.1.1) is an ODE of **first order**, for it contains only a first derivative with respect to t. Many ODEs that occur in practice are of second order or higher, so the form we have chosen may seem unduly restrictive. However, any higher-order ODE can be reduced to an equivalent system of first-order ODEs in a mechanical fashion by introducing additional variables representing lower-order terms. The following example illustrates this reduction.

EXAMPLE 1.1.6. Suppose that a mass m at the end of a massless spring of length y experiences a force $F = -K(y - y^*)$, where K is the spring constant and y^* is the rest position (Hooke's Law). By Newton's First Law, the motion is governed by the autonomous equation

$$y_{tt} = -\frac{K}{m}(y - y^*), \qquad y(0) = a, \quad y_t(0) = b$$

for some a and b, a scalar second-order initial-value problem. Now let u be the 2-vector with $u^{(1)} = y$, $u^{(2)} = y_t$. Then the equation can be rewritten as the first-order system

$$\begin{aligned} u_t^{(1)} &= u^{(2)}, & u^{(1)}(0) &= a, \\ u_t^{(2)} &= -\frac{K}{m}(u^{(1)} - y^*), & u^{(2)}(0) &= b. \end{aligned}$$

It should be clear from this example, and Exercise 1.1.1 below, how to reduce an arbitrary collection of higher-order equations to a first-order system. More formal treatments of this process can be found in the references.

Throughout the fields of ordinary and partial differential equations—and their numerical analysis—there are a variety of procedures in which one problem is reduced to another. We shall see inhomogeneous terms reduced to initial data, initial data reduced to boundary values, and so on. But this reduction of higher-order ODEs to first-order systems is unusual in that it is not just a convenient theoretical device, but highly practical. In fact, most of the general-purpose ODE software currently available assumes that the equation is written as a first-order system. One pays some price in efficiency for this, but it is usually not too great. For PDEs, on the other hand, such reductions are less often practical, and indeed there is less general-purpose software available of any kind.

It may seem obvious that (1.1.2) should have a unique solution for all $t > 0$; after all, the equation tells us exactly how u changes at each instant. But in fact, solutions can fail to exist or fail to be unique, and an example of nonexistence for $t \ge \log 2$ appeared already in Example 1.1.4 (see also Exercises 1.1.4 and 1.1.5). To ensure existence and uniqueness, we must make some assumptions concerning f. The standard assumption is that f is continuous with respect to t and satisfies a (uniform) **Lipschitz condition** with respect to u. This means that there exists a constant $L > 0$ such that for all $u, v \in \mathbb{C}^N$ and $t \in [0,T]$,

$$\|f(u,t) - f(v,t)\| \le L\|u - v\|, \tag{1.1.3}$$

where $\|\cdot\|$ denotes some norm on the set of N-vectors. (For $N = 1$, i.e. a scalar system of equations, $\|\cdot\|$ is usually just the absolute value $|\cdot|$. For $N \ge 2$, the most important examples of norms are $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$, and the reader should make sure that he or she is familiar with these examples. See Appendix B for a review of norms.) A sufficient condition for Lipschitz continuity is that the partial derivative of $f(u,t)$ with respect to u exists and is bounded in norm by L for all $u \in \mathbb{C}^N$ and $t \in [0,T]$.

The following result goes back to Cauchy in 1824.

EXISTENCE AND UNIQUENESS FOR THE INITIAL VALUE PROBLEM

Theorem 1.1. *Let $f(u,t)$ be continuous with respect to t and uniformly Lipschitz continuous with respect to u for $t \in [0,T]$. Then there exists a unique differentiable function $u(t)$ that satisfies the initial-value problem (1.1.2).*

The standard proof of Theorem 1.1 nowadays, which can be found in many books on ODEs, is based on a construction called the method of successive approximations or the Picard iteration. This construction can be realized as a numerical method, but not one of the form we shall discuss in this book. Following Cauchy's original idea, however, a more elementary proof can also be based on more standard numerical methods, such as Euler's formula, which is mentioned in the next section. See Henrici (1962) or Hairer, Nørsett & Wanner (1987).

Theorem 1.1 can be strengthened by allowing f to be Lipschitz continuous not for all u, but for u confined to an open subset D of $\mathbb{C}^N$. Unique solutions then still exist on the interval $[0,T']$, where T' is either the first point at which the solution hits the boundary of D, or T, whichever is smaller.

EXERCISES

▷ *1.1.1. Reduction to first-order system.* Consider the system of ODEs

$$u_{ttt} = u_{tt} + v_t, \qquad v_t t = u^2 + \sin(v) + e^t u_t v_t$$

with initial data

$$u(0) = u_t(0) = u_{tt}(0) = v_t(0) = 0, \qquad v(0) = 1.$$

Reduce this initial-value problem to a first-order system in the standard form (1.1.2).

▷ *1.1.2. The planets.* M planets (or suns, or spacecraft) orbit about each other in three space dimensions according to Newton's laws of gravitation and acceleration. What are the dimension and order of the corresponding system of ODEs— *(a)* When written in their most natural form? *(b)* When reduced to a first-order system? (This problem is intended to be straightforward; do not attempt clever tricks such as reduction to center-of-mass coordinates.)

▷ *1.1.3. Existence and uniqueness.* Apply Theorem 1.1 to show existence and uniqueness of the solutions we have given for the following initial-value problems, stating explicitly your choices of suitable Lipschitz constants L:

(a) Example 1.1.1.

(b) Example 1.1.3.

(c) Example 1.1.4. First, by considering $u_t = u + u^2$ itself, explain why Theorem 1.1 does not guarantee existence or uniqueness for all $t > 0$. Next, by considering the transformed equation $w_t = -1 - w$, show that Theorem 1.1 *does* guarantee existence and uniqueness

until such time as u may become infinite. Exactly how does the proof of the theorem fail at that point?

(d) Example 1.1.2.

▷ *1.1.4. Nonexistence and nonuniqueness.* Consider the scalar initial-value problem

$$u_t = u^\alpha, \qquad u(0) = u_0 \geq 0$$

for some constant $\alpha > 0$.

(a) For which α does Theorem 1.1 guarantee existence and uniqueness of solutions for all $t > 0$?

(b) For $\alpha = 2$ and $u_0 = 1$, no solution for all $t > 0$ exists. Verify this informally by finding an explicit solution that blows up to infinity in a finite time. (Of course such an example by itself does not constitute a proof of nonexistence.)

(c) For $\alpha = \frac{1}{2}$ and $u_0 = 0$, there is more than one solution. Find one of them, an "obvious" one. Then find another "obvious" one. Now construct an infinite family of distinct solutions.

▷ *1.1.5. Continuity with respect to t.* Theorem 1.1 requires continuity with respect to t as well as Lipschitz continuity with respect to u. Show that this assumption cannot be dispensed with by finding an initial-value problem, independent of T, in which f is uniformly Lipschitz continuous with respect to u but discontinuous with respect to t, and for which no solution exists on any interval $[0,T]$ with $T > 0$. *(Hint:* consider the degenerate case in which $f(u,t)$ is independent of u.)

1.2. Linear multistep formulas

Most numerical methods in every field are based on discretization. For the solution of ordinary differential equations, one of the most powerful discretization strategies goes by the name of **linear multistep methods**.

Suppose we are given an initial-value problem (1.1.2) that satisfies the hypotheses of Theorem 1.1 on some interval $[0,T]$. Then it has a unique solution $u(t)$ on that interval, but the solution can rarely be found analytically. Let $k>0$ be a real number, the **time step**, and let $t_0,t_1,\dots$ be defined by $t_n=nk$. Our goal is to construct a sequence of values $v^0,v^1,\dots$ such that

$$v^n \approx u(t_n), \qquad n\ge 0. \tag{1.2.1}$$

(The superscripts are not exponents!—we are leaving room for subscripts to accommodate spatial discretization in later chapters.) We also sometimes write $v(t_n)$ instead of v^n. Let f^n be the abbreviation

$$f^n = f(v^n,t_n). \tag{1.2.2}$$

A linear multistep method is a formula for calculating each new value v^{n+1} from some of the previous values $v^0,\dots,v^n$ and $f^0,\dots,f^n$.* Equation (1.2.11) below will make this more precise.

We shall take the attitude, standard in this field, that the aim in solving an initial-value problem numerically is to achieve a prescribed accuracy with the use of as few function evaluations $f(v^n,t_n)$ as possible. In other words, obtaining function values is assumed to be so expensive that all subsequent manipulations—all "overhead" operations—are essentially free. For easy problems this assumption may be unrealistic, but it is the harder problems that matter more. For hard problems values of f may be very expensive to determine, particularly if they are obtained by solving an inner algebraic or differential equation.

Linear multistep methods are designed to minimize function evaluations by using the approximate values v^n and f^n repeatedly.

The simplest linear multistep method is a one-step method: the **Euler** formula, defined by

$$v^{n+1} = v^n + kf^n. \tag{1.2.3}$$

The motivation for this formula is linear extrapolation, as suggested in Figure 1.2.1a. If v^0 is given (presumably set equal to the initial value u_0), it is a straightforward matter to apply (1.2.3) to compute successive values $v^1,v^2,\dots.$ Euler's method is an example of an **explicit one-step formula**.

A related linear multistep method is the **backward Euler** (or **implicit Euler**) formula, also a one-step formula, defined by

$$v^{n+1} = v^n + kf^{n+1}. \tag{1.2.4}$$

The switch from f^n to f^{n+1} is a big one, for it makes the backward Euler formula **implicit**. According to (1.2.2), f^{n+1} is an abbreviation for $f(v^{n+1},t_{n+1})$. Therefore, it would

*In general, v^n and f^n are vectors in $\mathbb{C}^N$, but it is safe to think of them as scalars; the extension to systems of equations is easy.

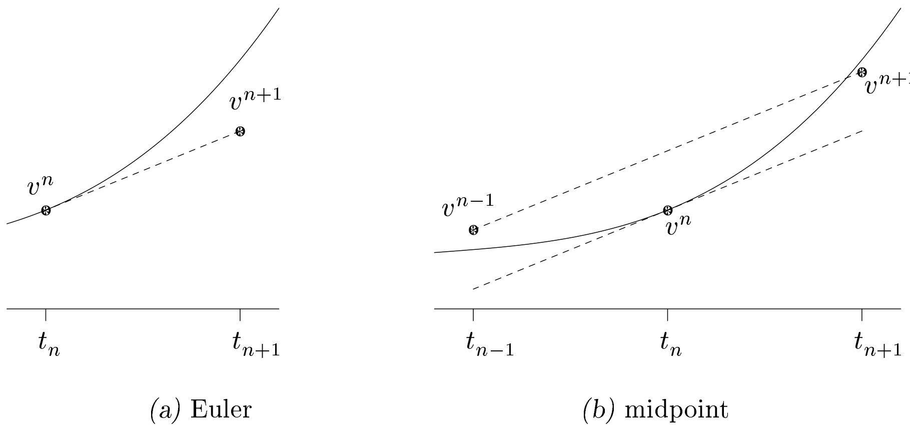


Figure 1.2.1. One step of the Euler and midpoint formulas. The solid curves represent not the exact solution to the initial-value problem, but the exact solution to the problem $u_t = f$, $u(t_n) = v^n$.

appear that v^{n+1} cannot be determined from (1.2.4) unless it is already known! In fact, to implement an implicit formula, one must employ an iteration of some kind to solve for the unknown v^{n+1}, and this involves extra work, not to mention the questions of existence and uniqueness.* But as we shall see, the advantage of (1.2.4) is that in some situations it may be stable when (1.2.3) is catastrophically unstable. Throughout the numerical solution of differential equations, there is a tradeoff between explicit methods, which tend to be easier to implement, and implicit ones, which tend to be more stable. Typically this tradeoff takes the form that an explicit method requires less work per time step, while an implicit method is able to take larger and hence fewer time steps without sacrificing accuracy to unstable oscillations.*

An example of a more accurate linear multistep formula is the **trapezoid** rule,

$$v^{n+1} = v^n + \frac{k}{2}(f^n + f^{n+1}), \tag{1.2.5}$$

also an implicit one-step formula. Another is the **midpoint** rule,

$$v^{n+1} = v^{n-1} + 2kf^n, \tag{1.2.6}$$

an explicit **two-step** formula (Figure 1.2.1b). The fact that (1.2.6) involves multiple time levels raises a new difficulty. We can set $v^0 = u_0$, but to compute v^1 with this formula, where

*In so-called predictor-corrector methods, not discussed in this book, the iteration is terminated before convergence, giving a class of methods intermediate between explicit and implicit.

*But don't listen to people who talk about "conservation of difficulty" as if there were never any clear winners! We shall see that sometimes explicit methods are vastly more efficient than implicit ones (large non-stiff systems of ODEs), and sometimes it is the reverse (small stiff systems).

shall we get v^{-1}? Or if we want to begin by computing v^2, where shall we get v^1? This initialization problem is a general one for linear multistep formulas, and it can be addressed in several ways. One is to calculate the missing values numerically by some simpler formula such as Euler's method—possibly applied several times with a smaller value of k to avoid loss of accuracy. Another is to handle the initialization process by Runge-Kutta methods, to be discussed in Section 1.9.

In Chapter 3 we shall see that the formulas (1.2.3)–(1.2.6) have important analogs for partial differential equations.[†] For the easier problems of ordinary differential equations, however, they are too primitive for most purposes. Instead one usually turns to more complicated and more accurate formulas, such as the **fourth-order Adams-Bashforth** formula,

$$v^{n+1} = v^n + \frac{k}{24}\left(55f^n - 59f^{n-1} + 37f^{n-2} - 9f^{n-3}\right), \tag{1.2.7}$$

an explicit four-step formula, or the **fourth-order Adams-Moulton** formula,

$$v^{n+1} = v^n + \frac{k}{24}\left(9f^{n+1} + 19f^n - 5f^{n-1} + f^{n-2}\right), \tag{1.2.8}$$

an implicit three-step formula. Another implicit three-step formula is the **third-order backwards differentiation** formula,

$$v^{n+1} = \tfrac{18}{11}v^n - \tfrac{9}{11}v^{n-1} + \tfrac{2}{11}v^{n-2} + \tfrac{6}{11}kf^{n+1}, \tag{1.2.9}$$

whose advantageous stability properties will be discussed in Section 1.8.

EXAMPLE 1.2.1. Let us perform an experiment to compare some of these methods. The initial-value problem will be

$$u_t = u, \qquad t \in [0,2], \quad u(0) = 1, \tag{1.2.10}$$

whose solution is simply $u(t) = e^t$. Figure 1.2.2 compares the exact solution with the numerical solutions obtained by the Euler and midpoint formulas with $k = 0.2$ and $k = 0.1$. (For simplicity, we took v^1 equal to the exact value $u(t_1)$ to start the midpoint formula.) A solution with the fourth-order Adams-Bashforth formula was also calculated, but is indistinguishable from the exact solution in the plot. Notice that the midpoint formula does much better than the Euler formula, and that cutting k in half improves both. To make this precise, Table 1.2.1 compares the various errors $u(2) - v(2)$. The final column of ratios suggests that the errors for the three formulas have magnitudes $\Theta(k)$, $\Theta(k^2)$, and $\Theta(k^4)$ as $k \to 0$. The latter figure accounts for the name "fourth-order;" see the next section.

Up to this point we have listed a few formulas, some with rather complicated coefficients, and shown that they work at least for one problem. Fortunately, the subject of linear multistep methods has a great deal more order in it than this. A number of questions suggest themselves:

- What is the general form of a linear multistep method?

[†] Euler, Backward Euler, Crank-Nicolson, and Leap Frog.

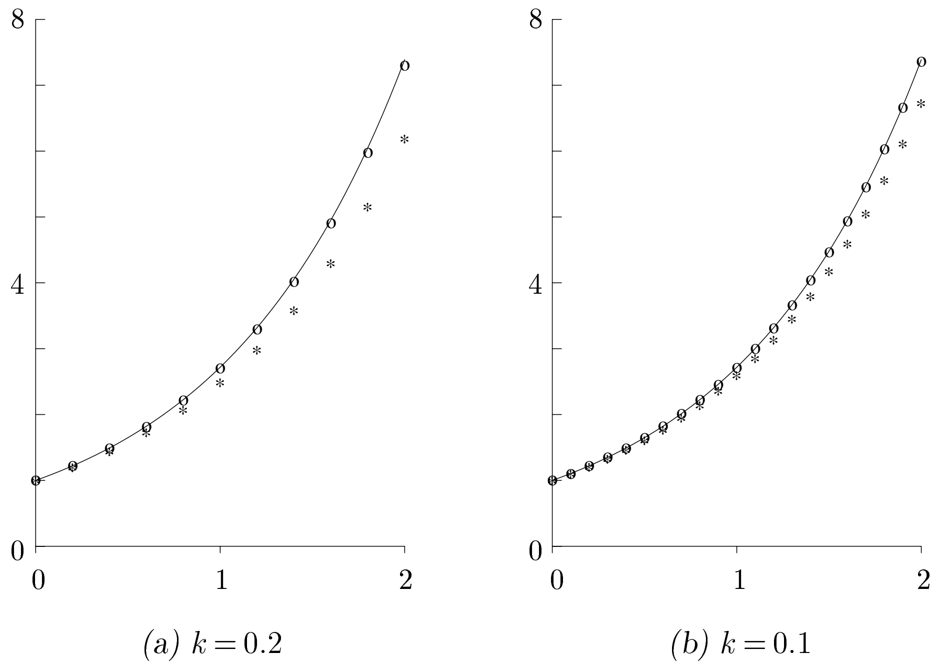


Figure 1.2.2. Solution of (1.2.10) by the Euler ($*$) and midpoint (o) formulas.

	$k = 0.2$	$k = 0.1$	$k = 0.05$	ratio ($k = 0.1$ to $k = .05$)
Euler	1.19732	0.66156	0.34907	1.90
Midpoint	0.09055	0.02382	0.00607	3.92
AB4	0.00422	0.00038	0.00003	12.7

Table 1.2.1. Errors $u(2) - v(2)$ for the experiment of Figure 1.2.2.

- Can appropriate coefficients be derived in a systematic way?
- How accurate are these methods?
- Can anything go wrong?

In the following pages we shall see that these questions have very interesting answers.

We can take care of the first one right away by defining the general linear multistep formula:

*An s-***step linear multistep formula** *is a formula*

$$\sum_{j=0}^{s} \alpha_j v^{n+j} = k \sum_{j=0}^{s} \beta_j f^{n+j} \qquad (1.2.11)$$

for some constants $\{\alpha_j\}$ and $\{\beta_j\}$ with $\alpha_s = 1$ and either $\alpha_0 \neq 0$ or $\beta_0 \neq 0$. If $\beta_s = 0$ the formula is **explicit***, and if $\beta_s \neq 0$ it is* **implicit***.*

Readers familiar with electrical engineering may notice that (1.2.11) looks like the definition of a **recursive** or **IIR digital filter** (Oppenheim & Schafer, 1989). Linear multistep formulas can indeed be thought of in this way, and many of the issues to be described here also come up in digital signal processing, such as the problem of stability and the idea of testing for it by looking for zeros of a polynomial in the unit disk of the complex plane. A linear multistep formula is not quite a digital filter of the usual kind, however. In one respect it is more general, since the function f depends on u and t rather than on t alone. In another respect it is more narrow, since the coefficients are chosen so that the formula has the effect of integration. Digital filters are designed to accomplish a much wider variety of tasks, such as high-pass or low-pass filtering (= smoothing), differentiation, or channel equalization.

What about the word "linear"? Do linear multistep formulas apply only to linear differential equations? Certainly not. Equation (1.2.11) is called linear because the quantities v^n and f^n are related linearly; f may very well be a nonlinear function of u and t. Some authors refer simply to "multistep formulas" to avoid this potential source of confusion.

EXERCISES

▷ *1.2.1.* What are s, $\{\alpha_j\}$, and $\{\beta_j\}$ for formulas (1.2.7)–(1.2.9)?

▷ *1.2.2. Linear multistep formula for a system of equations.* One of the footnotes above claimed that the extension of linear multistep methods to systems of equations is easy. Verify this by writing down exactly what the 2×2 system of Example 1.1.5 becomes when it is approximated by the midpoint rule (1.2.6).

▷ *1.2.3. Exact answers for low-degree polynomials.* If L is a nonnegative integer, the initial-value problem

$$u_t(t) = \frac{L}{t+1} u(t), \qquad u(0) = 1$$

has the unique solution $u(t) = (t+1)^L$. Suppose we calculate an approximation $v(2)$ by a linear multistep formula, using exact values where necessary for the initialization.

(a) For which L does (1.2.3) reproduce the exact solution? *(b)* (1.2.6)? *(c)* (1.2.7)?

▷ *1.2.4. Extrapolation methods.*

(a) The values $V_k = v(2) \approx u(2)$ computed by Euler's method in Figure 1.2.2 are $V_{0.1} \approx 6.72750$ and $V_{0.2} \approx 6.19174$. Use a calculator to compute the analogous value $V_{0.4}$.

(b) It can be shown that these quantities V_k satisfy

$$V_k = u(2) + C_1 k + C_2 k^2 + O(k^3)$$

for some constants C_1, C_2 as $k \to 0$. In the process known as **Richardson extrapolation**, one constructs the higher-order estimates

$$V_k' = V_k + (V_k - V_{2k}) = u(2) + O(k^2)$$

and

$$V_k'' = V_k' + \tfrac{1}{3}(V_k' - V_{2k}') = u(2) + O(k^3).$$

Apply these formulas to compute $V_{0.1}''$ for this problem. How accurate is it? This is an example of an **extrapolation method** for solving an ordinary differential equation (Gragg 1965; Bulirsch & Stoer 1966). The analogous method for calculating integrals is known as **Romberg integration** (Davis & Rabinowitz 1975).

1.3. Accuracy and consistency

In this section we define the consistency and order of accuracy of a linear multistep formula, as well as the associated characteristic polynomials $\rho(z)$ and $\sigma(z)$, and show how these definitions can be applied to derive accurate formulas by a method of undetermined coefficients. We also show that linear multistep formulas are connected with the approximation of the function $\log z$ by the rational function $\rho(z)/\sigma(z)$ at $z=1$.

With $\{\alpha_j\}$ and $\{\beta_j\}$ as in (1.2.11), the **characteristic polynomials** (or **generating polynomials**) for the linear multistep formula are defined by

$$\rho(z)=\sum_{j=0}^{s}\alpha_j z^j, \qquad \sigma(z)=\sum_{j=0}^{s}\beta_j z^j. \tag{1.3.1}$$

The polynomial ρ has degree exactly s, and σ has degree s if the formula is implicit or $<s$ if it is explicit. Specifying ρ and σ is obviously equivalent to specifying the linear multistep formula by its coefficients. We shall see that ρ and σ are convenient for analyzing accuracy and indispensable for analyzing stability.

EXAMPLE 1.3.1. Here are the characteristic polynomials for the first four formulas of the last section:

Euler (1.2.3):	$s=1$,	$\rho(z)=z-1$,	$\sigma(z)=1$.
Backward Euler (1.2.4):	$s=1$,	$\rho(z)=z-1$,	$\sigma(z)=z$.
Trapezoid (1.2.5):	$s=1$,	$\rho(z)=z-1$,	$\sigma(z)=\frac{1}{2}(z+1)$.
Midpoint (1.2.6):	$s=2$,	$\rho(z)=z^2-1$,	$\sigma(z)=2z$.

Now let Z denote a **time shift operator** that acts both on discrete functions according to

$$Zv^n=v^{n+1}, \tag{1.3.2}$$

and on continuous functions according to

$$Zu(t)=u(t+k). \tag{1.3.3}$$

(In principle we should write $[Zv]^n$ and $[Zu](t)$, but expressions like Zv^n and even $Z(v^n)$ are irresistibly convenient.) The powers of Z have the obvious meanings, e.g., $Z^2v^n=v^{n+2}$ and $Z^{-1}u(t)=u(t-k)$.

Equation (1.2.11) can be rewritten compactly in terms of Z, ρ, and σ:

$$\rho(Z)v^n-k\sigma(Z)f^n\,=\,0. \tag{1.3.4}$$

When a linear multistep formula is applied to solve an ODE, this equation is satisfied exactly since it is the definition of the numerical approximation $\{v^n\}$.

If the linear multistep formula is a good one, the analogous equation ought to be nearly satisfied when $\{v^n\}$ and $\{f^n\}$ are replaced by discretizations of any well-behaved function $u(t)$ and its derivative $u_t(t)$. With this in mind, let us define the **linear multistep difference operator** $\mathcal{L}$, acting on the set of continuously differentiable functions $u(t)$, by

$$\mathcal{L} = \rho(Z) - k\mathcal{D}\sigma(Z), \tag{1.3.5}$$

where $\mathcal{D}$ is the time differentiation operator, that is,

$$\begin{aligned} \mathcal{L}u(t_n) &= \rho(Z)u(t_n) - k\sigma(Z)u_t(t_n) \\ &= \sum_{j=0}^{s} \alpha_j u(t_{n+j}) - k\sum_{j=0}^{s} \beta_j u_t(t_{n+j}). \end{aligned} \tag{1.3.6}$$

If the linear multistep formula is accurate, $\mathcal{L}u(t_n)$ ought to be small, and we make this precise as follows. Let a function $u(t)$ and its derivative $u_t(t)$ be expanded formally in Taylor series about t_n,

$$u(t_{n+j}) = u(t_n) + jku_t(t_n) + \tfrac{1}{2}(jk)^2 u_{tt}(t_n) + \cdots, \tag{1.3.7}$$

$$u_t(t_{n+j}) = u_t(t_n) + jku_{tt}(t_n) + \tfrac{1}{2}(jk)^2 u_{ttt}(t_n) + \cdots. \tag{1.3.8}$$

Inserting these formulas in (1.3.6) gives the **formal local discretization error** (or **formal local truncation error**) for the linear multistep formula,

$$\mathcal{L}u(t_n) = C_0 u(t_n) + C_1 k u_t(t_n) + C_2 k^2 u_{tt}(t_n) + \cdots, \tag{1.3.9}$$

where

$$\begin{aligned} C_0 &= \alpha_0 + \cdots + \alpha_s, \\ C_1 &= (\alpha_1 + 2\alpha_2 + \cdots + s\alpha_s) - (\beta_0 + \cdots + \beta_s), \\ C_2 &= \tfrac{1}{2}(\alpha_1 + 4\alpha_2 + \cdots + s^2\alpha_s) - (\beta_1 + 2\beta_2 + \cdots + s\beta_s) \\ &\vdots \\ C_m &= \sum_{j=0}^{s} \frac{j^m}{m!}\alpha_j - \sum_{j=0}^{s} \frac{j^{m-1}}{(m-1)!}\beta_j. \end{aligned} \tag{1.3.10}$$

We now define:

A linear multistep formula has **order of accuracy** *p if*

$$\mathcal{L}u(t_n) = \Theta(k^{p+1}) \qquad \text{as } k \to 0,$$

i.e., if $C_0 = C_1 = \cdots = C_p = 0$ but $C_{p+1} \neq 0$. The **error constant** *is C_{p+1}. The formula is* **consistent** *if $C_0 = C_1 = 0$, i.e., if it has order of accuracy $p \geq 1$.*

EXAMPLE 1.3.1, CONTINUED. First let us analyze the local accuracy of the Euler formula in a lowbrow fashion that is equivalent to the definition above: we suppose that $v^0,\dots,v^n$ are exactly equal to $u(t_0),\dots,u(t_n)$, and ask how close v^{n+1} will then be to $u(t_{n+1})$. If $v^n = u(t_n)$ exactly, then by definition,

$$v^{n+1} = v^n + kf^n = u(t_n) + ku_t(t_n),$$

while the Taylor series (1.3.7) gives

$$u(t_{n+1}) = u(t_n) + ku_t(t_n) + \frac{k^2}{2}u_{tt}(t_n) + O(k^3).$$

Subtraction gives

$$u(t_{n+1}) - v^{n+1} = \frac{k^2}{2}u_{tt}(t_n) + O(k^3)$$

as the formal local discretization error of the Euler formula.

Now let us restate the argument in terms of the operator $\mathcal{L}$. By combining (1.3.6) and (1.3.7), or by calculating $C_0 = C_1 = 0$, $C_2 = \frac{1}{2}$, $C_3 = \frac{1}{6}$ from the values $\alpha_0 = -1$, $\alpha_1 = 1$, $\beta_0 = 1$, $\beta_1 = 0$ in (1.3.10), we obtain

$$\begin{aligned}\mathcal{L}u(t_n) &= u(t_{n+1}) - u(t_n) - ku_t(t_n)\\ &= \frac{k^2}{2}u_{tt}(t_n) + \frac{k^3}{6}u_{ttt}(t_n) + \cdots.\end{aligned} \qquad (1.3.11)$$

Thus again we see that the order of accuracy is 1, and evidently the error constant is $\frac{1}{2}$.

Similarly, the trapezoid rule (1.2.5) has

$$\begin{aligned}\mathcal{L}u(t_n) &= u(t_{n+1}) - u(t_n) - \frac{k}{2}\big(u_t(t_{n+1}) + u_t(t_n)\big)\\ &= -\frac{k^3}{12}u_{ttt}(t_n) - \frac{k^4}{24}u_{tttt}(t_n) + \cdots,\end{aligned} \qquad (1.3.12)$$

with order of accuracy 2 and error constant $-\frac{1}{12}$, and the midpoint rule (1.2.6) has

$$\mathcal{L}u(t_n) = \tfrac{1}{3}k^3u_{ttt}(t_n) + \tfrac{1}{3}k^4u_{tttt}(t_n) + \cdots, \qquad (1.3.13)$$

with order of accuracy 2 and error constant $\frac{1}{3}$.

The idea behind the definition of order of accuracy is as follows. Equation (1.3.9) suggests that if a linear multistep formula is applied to a problem with a sufficiently smooth solution $u(t)$, an error of approximately $C_{p+1}k^{p+1}\frac{d^{p+1}u}{dt^{p+1}}(t_n) = O(k^{p+1})$ will be introduced locally at step n.* This error is known as the **local discretization** (or **truncation**) **error** at step n for the given initial-value problem (as opposed to the formal local discretization error, which is a

*As usual, the limit associated with the Big-O symbol ($k \to 0$) is omitted here because it is obvious.

formal expression that does not depend on the initial-value problem). Since there are $T/k = \Theta(k^{-1})$ steps in the interval $[0,T]$, these local errors can be expected to add up to a global error $O(k^p)$. We shall make this argument more precise in Section 1.6.

With hindsight, it is obvious by symmetry that the trapezoid and midpoint formulas had to have even orders of accuracy. Notice that in (1.2.6), the terms involving v^{n+j} are antisymmetric about t_n, and those involving f^{n+j} are symmetric. In (1.3.13) the effect of these symmetries is disguised, because t_{n-1} was shifted to t_n for simplicity of the formulas in passing from (1.2.6) to (1.2.11) and (1.3.4). However, if we had expressed $\mathcal{L}u(t_n)$ as a Taylor series about t_{n+1} instead, we would have obtained

$$\mathcal{L}u(t_n) = \tfrac{1}{3}k^3 u_{ttt}(t_{n+1}) + \tfrac{1}{60}k^5 u_{ttttt}(t_{n+1}) + O(k^7),$$

with all the even-order derivatives in the series vanishing due to symmetry. Now it can be shown that to leading order, it doesn't matter what point one expands $\mathcal{L}u(t_n)$ about: $C_0,\dots,C_{p+1}$ are independent of this point, though the subsequent coefficients $C_{p+2}, C_{p+3},\dots$ are not. Thus the vanishing of the even-order terms above is a valid explanation of the second-order accuracy of the midpoint formula. For the trapezoid rule (1.2.5), we can make an analogous argument involving an expansion of $\mathcal{L}u(t_n)$ about $t_{n+1/2}$.

As a rule, symmetry arguments do not go very far in the analysis of discrete methods for ODEs, since most of the formulas used in practice are of high order and fairly complicated. They go somewhat further with the simpler formulas used for PDEs.

The definition of order of accuracy suggests a **method of undetermined coefficients** for deriving linear multistep formulas: having decided somehow which α_j and β_j are permitted to be nonzero, simply adjust these parameters to make p as large as possible. If there are q parameters, not counting the fixed value $\alpha_s = 1$, then the equations (1.3.10) with $0 \le m \le q-1$ constitute a $q \times q$ linear system of equations. If this system is nonsingular, as it usually is, it must have a unique solution that defines a linear multistep formula of order at least $p = q-1$. See Exercise 1.3.1.

The method of undetermined coefficients can also be described in another, equivalent way that was hinted at in Exercise 1.2.3. It can be shown that any consistent linear multistep formula computes the solution to an initial-value problem exactly in the special case when that solution is a polynomial $p(t)$ of degree L, provided that L is small enough. The order of accuracy p is the largest such L (see Exercise 1.3.6). To derive a linear multistep formula for a particular choice of available parameters α_j and β_j, one can choose the parameters in such a way as to make the formula exact when applied to polynomials of as high a degree as possible.

At this point it may appear to the reader that the construction of linear multistep formulas is a rather uninteresting matter. Given $s>0$, why not simply pick $\alpha_0,\dots,\alpha_{s-1}$ and $\beta_0,\dots,\beta_s$ so as to achieve order of accuracy $2s$? The answer is that for $s>2$, the resulting formulas are unstable and consequently useless (see Exercise 1.5.4). In fact, as we shall see in Section 1.5, a famous theorem of Dahlquist asserts that a stable s-step linear multistep formula can have order of accuracy at most $s+2$. Consequently, nothing can be gained by permitting all $2s+1$ coefficients in an s-step formula to be nonzero. Instead, the linear multistep formulas used in practice usually have most of the α_j or most of the β_j equal to zero. In the next section we shall describe several important families of this kind.

And there is another reason why the construction of linear multistep formulas is interesting. It can be made exceedingly slick! We shall now describe how the formulas above can be analyzed more compactly, and the question of order of accuracy reduced to a question of rational approximation, by manipulation of formal power series.

Taking $j=1$ in (1.3.7) gives the identity

$$u(t_{n+1}) = u(t_n)+ku_t(t_n)+\tfrac{1}{2}k^2u_{tt}(t_n)+\cdots.$$

Since $u(t_{n+1})=Zu(t_n)$, here is another way to express the same fact:

$$Z = 1+(k\mathcal{D})+\tfrac{1}{2!}(k\mathcal{D})^2+\tfrac{1}{3!}(k\mathcal{D})^3+\cdots = e^{k\mathcal{D}}. \qquad (1.3.14)$$

Like (1.3.7), this formula is to be interpreted as a formal identity; the idea is to use it as a tool for manipulation of terms of series without making any claims about convergence. Inserting (1.3.14) in (1.3.5) and comparing with (1.3.9) gives

$$\mathcal{L} = \rho(e^{k\mathcal{D}})-k\mathcal{D}\sigma(e^{k\mathcal{D}}) = C_0+C_1(k\mathcal{D})+C_2(k\mathcal{D})^2+\cdots. \qquad (1.3.15)$$

In other words, the coefficients C_j of (1.3.9) are nothing else than the Taylor series coefficients of the function $\rho(e^{k\mathcal{D}})-k\mathcal{D}\sigma(e^{k\mathcal{D}})$ with respect to the argument $k\mathcal{D}$. If we let κ be an abbreviation for $k\mathcal{D}$, this equation becomes even simpler:

$$\mathcal{L} = \rho(e^{\kappa})-\kappa\sigma(e^{\kappa}) = C_0+C_1\kappa+C_2\kappa^2+\cdots. \qquad (1.3.16)$$

With the aid of a symbolic algebra system such as Macsyma, Maple, or Mathematica, it is a trivial matter to make use of (1.3.16) to compute the coefficients C_j for a linear multistep formula if the parameters α_j and β_j are given. See Exercise 1.3.7.

By definition, a linear multistep formula has order of accuracy p if and only if the the term between the equal signs in (1.3.16) is $\Theta(\kappa^{p+1})$ as $\kappa\to 0$.

Since $\sigma(e^\kappa)$ is an analytic function of κ, if it is nonzero at $\kappa=0$ we can divide through to conclude that the linear multistep formula has order of accuracy p if and only if

$$\frac{\rho(e^\kappa)}{\sigma(e^\kappa)} = \kappa + \Theta(\kappa^{p+1}) \qquad \text{as } \kappa \to 0. \tag{1.3.17}$$

The following theorem restates this conclusion in terms of the variable $z=e^\kappa$.

LINEAR MULTISTEP FORMULAS AND RATIONAL APPROXIMATION

Theorem 1.2. *A linear multistep formula with $\sigma(1)\neq 0$ has order of accuracy p if and only if*

$$\begin{aligned}\frac{\rho(z)}{\sigma(z)} &= \log z + \Theta((z-1)^{p+1}) \\ &= \Big[(z-1) - \tfrac{1}{2}(z-1)^2 + \tfrac{1}{3}(z-1)^3 - \cdots\Big] + \Theta((z-1)^{p+1})\end{aligned} \tag{1.3.18}$$

as $z\to 1$. It is consistent if and only if

$$\rho(1)=0 \qquad \textit{and} \qquad \rho'(1)=\sigma(1). \tag{1.3.19}$$

Proof. To get from (1.3.17) to the first equality of (1.3.18), we make the change of variables $z=e^\kappa$, $\kappa=\log z$, noting that $\Theta(\kappa^{p+1})$ as $\kappa\to 0$ has the same meaning as $\Theta((z-1)^{p+1})$ as $z\to 1$ since $e^\kappa=1$ and $d(e^\kappa)/d\kappa\neq 0$ at $\kappa=0$. The second equality of (1.3.18) is just the usual Taylor series for $\log z$.

To prove (1.3.19), let (1.3.18) be written in the form

$$\rho(z) = \sigma(z)(z-1) + O((z-1)^2) + \Theta((z-1)^{p+1}),$$

or by expanding $\rho(z)$ and $\sigma(z)$ about $z=1$,

$$\rho(1) + (z-1)\rho'(1) = (z-1)\sigma(1) + O((z-1)^2) + \Theta((z-1)^{p+1}). \tag{1.3.20}$$

Matching successive powers of $z-1$, we obtain $\rho(1)=0 \Leftrightarrow p\geq 0$ and $p\geq 1 \Rightarrow \rho'(1)=\sigma(1)\Rightarrow p\neq 0$. Thus (1.3.19) is equivalent to $p\geq 1$, which is the definition of consistency. ∎

In Theorem 1.2 the ODE context of a linear multistep formula has vanished entirely, leaving a problem in the mathematical field known as **approximation theory**. Questions of approximation underlie most discrete methods for differential equations, both ordinary and partial.

EXAMPLE 1.3.2. The trapezoid rule (1.2.5) has $\rho(z)=z-1$ and $\sigma(z)=\frac{1}{2}(z+1)$. Since $\rho(1)=0$ and $\rho'(1)=1=\sigma(1)$, the formula is consistent. Comparing (1.3.18) with the expansion

$$\frac{\rho(z)}{\sigma(z)} = \frac{z-1}{\frac{1}{2}(z+1)} = \frac{z-1}{1+\frac{1}{2}(z-1)} = (z-1)\left[1-\frac{z-1}{2}+\frac{(z-1)^2}{4}-\cdots\right]$$

confirms that the trapezoid rule has order 2 and error constant $-\frac{1}{12}$.

This approach to linear multistep formulas by rational approximation is closely related to the methods of **generating functions** described in several of the references. It is also the basis of the method of analysis by **order stars** described later in this chapter. For ordinary differential equations with special structure, such as highly oscillatory behavior, it is sometimes advantageous to approximate $\log z$ at one or more points other than $z=1$; this is the idea behind the **frequency-fitted** or **mode-dependent** formulas discussed by Gautschi (1961), Liniger and Willoughby (1967), Kuo and Levy (1990), and others. See Exercise 1.3.3.

EXERCISES

▷ *1.3.1.* Show by the method of undetermined coefficients that

$$v^{n+1} = -4v^n + 5v^{n-1} + k\big(4f^n + 2f^{n-1}\big) \qquad (1.3.21)$$

is the most accurate 2-step explicit linear multistep formula, with order of accuracy $p=3$. In Section 1.5 we shall see that (1.3.21) is unstable and hence useless in practice.

▷ *1.3.2.* Consider the third-order backwards differentiation formula (1.2.9).
(*a*) What are $\rho(z)$ and $\sigma(z)$?
(*b*) Apply Theorem 1.2 to verify consistency.
(*c*) Apply Theorem 1.2 to verify that the order of accuracy is 3.

▷ *1.3.3. Optimal formulas with finite step size.* The concept of order of accuracy is based on the limit $k\to 0$, but one can also devise formulas on the assumption of a finite step size $k>0$. For example, an Euler-like formula might be defined by

$$v^{n+1} = v^n + \gamma(k)kf^n \qquad (1.3.22)$$

for some function $\gamma(k)$ with $\gamma(k)\to 1$ as $k\to 0$.
(*a*) What choice of $\gamma(k)$ makes (1.3.22) exact when applied to the equation $u_t=u$?
(*b*) ODEs of practical interest contain various time scales, so it is not really appropriate to consider just $u_t=u$. Suppose the goal is to approximate all problems $u_t=au$ with $a\in[0,1]$ as accurately as possible. State a definition of "as accurately as possible" based on the L^∞ norm of the error over a *single* time step, and determine the resulting function $\gamma(k)$.

▶ *1.3.4. Numerical experiments.* Consider the scalar initial-value problem

$$u_t(t) = e^{\cos(tu(t))} \quad \text{for} \quad t \in [0,3], \qquad u(0) = 0.$$

In this problem you will test four numerical methods: (i) Euler, (ii) Midpoint, (iii) Fourth-order Adams-Bashforth, and (iv) Fourth-order Runge-Kutta, defined by

$$\begin{aligned} a &:= kf(v^n, t_n), \\ b &:= kf(v^n + a/2, t_n + k/2), \\ c &:= kf(v^n + b/2, t_n + k/2), \\ d &:= kf(v^n + c, t_n + k), \\ v^{n+1} &:= v^n + \tfrac{1}{6}(a + 2b + 2c + d) \end{aligned} \qquad . \qquad (1.3.23)$$

(a) Write a computer program to implement (1.3.23) in high precision arithmetic (16 digits or more). Run it with $k = 1/2, 1/4, \ldots$ until you are confident that you have a computed value $v(3)$ accurate to at least 6 digits. This will serve as your "exact solution." Make a computer plot or a sketch of $v(t)$. Store appropriate values from your Runge-Kutta computations for use as starting values for the multistep formulas (ii) and (iii).

(b) Modify your program so that it computes $v(3)$ by each of the methods (i)–(iv) for the sequence of time steps $k = 1/2, 1/4, \ldots, 1/256$. Make a table listing $v(3)$ and $v(3) - u(3)$ for each method and each k.

(c) Draw a plot on a log-log scale of four curves representing $|v(3) - u(3)|$ as a function of the *number of evaluations of f*, for each of the four methods. (Make sure you calculate each f^n only once, and count the number of function evaluations for the Runge-Kutta formula rather than the number of time steps.)

(d) What are the approximate slopes of the lines in (c), and why? (If you can't explain them, there may be a bug in your program—very likely in the specification of initial conditions.) Which of the four methods is most efficient?

(e) If you are programming in Matlab, solve this same problem with the programs `ode23` and `ode45` with eight or ten different error tolerances. Measure how many time steps are required for each run and how much accuracy is achieved in the value $u(3)$, and add these new results to your plot of (c). What are the observed orders of accuracy of these adaptive codes? How do they compare in efficiency with your non-adaptive methods (i)–(iv)?

▷ *1.3.5. Statistical effects?* It was stated above that if local errors of magnitude $O(k^{p+1})$ are made at each of $\Theta(k^{-1})$ steps, then the global error will have magnitude $O(k^p)$. However, one might argue that more likely, the local errors will behave like random numbers of order $O(k^{p+1})$, and will accumulate in a square-root fashion according to the principles of a random walk, giving a smaller global error $O(k^{p+1/2})$. Experiments show that for most problems, including those of Example 1.2.1 and Exercise 1.3.4, this optimistic prediction is invalid. What is the fallacy in the random walk argument? Be specific, citing an equation in the text to support your answer.

▷ *1.3.6.* Prove the lemma alluded to on p. 23: *An s-step linear multistep formula has order of accuracy p if and only if, when applied to an ordinary differential equation $u_t = q(t)$, it gives exact results whenever q is a polynomial of degree $\le p$, but not whenever q is a polynomial* *of degree* $p+1$. (Assume arbitrary continuous initial data u_0 and exact numerical initial data $v_0, \dots, v^{s-1}$.)

▷ *1.3.7.* If you have access to a symbolic computation system, carry out the suggestion on p. 26: write a short program which, given the parameters α_j and β_j for a linear multistep formula, computes the coefficients $C_0, C_1, \dots$. Use your program to verify the results of Exercises 1.3.1 and 1.3.2, and then explore other linear multistep formulas that interest you.

1.4. Derivation of linear multistep formulas

The oldest linear multistep formulas are the Adams formulas,

$$v^{n+s} - v^{n+s-1} = k\sum_{j=0}^{s} \beta_j f^{n+j}, \qquad (1.4.1)$$

which date to the work of J. C. Adams* as early as 1855. In the notation of (1.2.11) we have $\alpha_s = 1$, $\alpha_{s-1} = -1$, and $\alpha_0 = \cdots = \alpha_{s-2} = 0$, and the first characteristic polynomial is $\rho(z) = z^s - z^{s-1}$. For each $s \geq 1$, the s-step **Adams-Bashforth** and **Adams-Moulton** formulas are the optimal explicit and implicit formulas of this form, respectively. "Optimal" means that the available coefficients $\{\beta_j\}$ are chosen to maximize the order of accuracy, and in both Adams cases, this choice turns out to be unique.

We have already seen the 1-step Adams-Bashforth and Adams-Moulton formulas: they are Euler's formula (1.2.3) and the trapezoid formula (1.2.5), respectively. The fourth-order Adams-Bashforth and Adams-Moulton formulas, with $s = 4$ and $s = 3$, respectively, were listed above as (1.2.7) and (1.2.8). The coefficients of these and other formulas are listed in Tables 1.4.1–1.4.3 on the next page. The "stencils" of various families of linear multistep formulas are summarized in Figure 1.4.1, which should be self-explanatory.

To calculate the coefficients of Adams formulas, there is a simpler and more enlightening alternative to the method of undetermined coefficients mentioned in the last section. Think of the values $f^n, \ldots, f^{n+s-1}$ (A-B) or $f^n, \ldots, f^{n+s}$ (A-M) as discrete samples of a continuous function $f(t) = f(u(t), t)$ that we want to integrate,

$$u(t_{n+s}) - u(t_{n+s-1}) = \int_{t_{n+s-1}}^{t_{n+s}} u_t(t)\,dt = \int_{t_{n+s-1}}^{t_{n+s}} f(t)\,dt,$$

as illustrated in Figure 1.4.2a. (Of course $f^n, \ldots, f^{n+s}$ will themselves be inexact due to earlier errors in the computation, but we ignore this for the moment.) Let $q(t)$ be the unique polynomial of degree at most $s-1$ (A-B) or s (A-M) that interpolates these data, and set

$$v^{n+s} - v^{n+s-1} = \int_{t_{n+s-1}}^{t_{n+s}} q(t)dt. \qquad (1.4.2)$$

Since the integral is a linear function of the data $\{f^{n+j}\}$, with coefficients that can be computed once and for all, (1.4.2) implicitly defines a linear multistep formula of the Adams type (1.4.1).

EXAMPLE 1.4.1. Let us derive the coefficients of the 2nd-order Adams-Bashforth formula, which are listed in Table 1.4.1. In this case the data to be interpolated are f^n and

*the same Adams who first predicted the existence of the planet Neptune

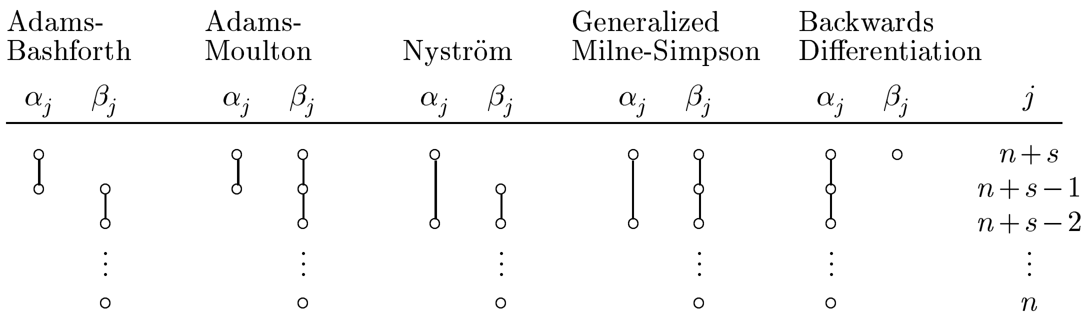


Figure 1.4.1. Stencils of various families of linear multistep formulas.

number of steps s	order p	β_s	β_{s-1}	β_{s-2}	β_{s-3}	β_{s-4}
1	1	0	1		(EULER)	
2	2	0	$\frac{3}{2}$	$-\frac{1}{2}$		
3	3	0	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$	
4	4	0	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$

Table 1.4.1. Coefficients $\{\beta_j\}$ of Adams-Bashforth formulas.

number of steps s	order p	β_s	β_{s-1}	β_{s-2}	β_{s-3}	β_{s-4}
1	1	1		(BACKWARD EULER)		
1	2	$\frac{1}{2}$	$\frac{1}{2}$		(TRAPEZOID)	
2	3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$		
3	4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$	
4	5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$

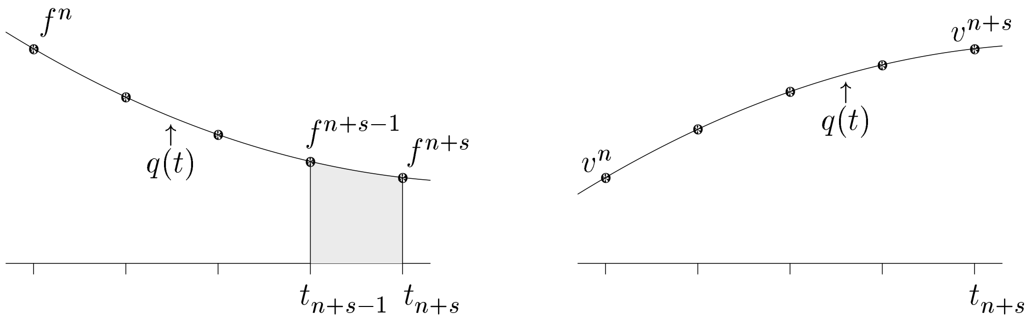
Table 1.4.2. Coefficients $\{\beta_j\}$ of Adams-Moulton formulas.

number of steps s	order p	α_s	α_{s-1}	α_{s-2}	α_{s-3}	α_{s-4}	β_s
1	1	1	-1		(BACKWARD EULER)		1
2	2	1	$-\frac{4}{3}$	$\frac{1}{3}$			$\frac{2}{3}$
3	3	1	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$		$\frac{6}{11}$
4	4	1	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$	$\frac{12}{25}$

Table 1.4.3. Coefficients $\{\alpha_j\}$ and β_s of backwards differentiation formulas.

f^{n+1}, and the interpolant is the linear polynomial $q(t) = f^{n+1} - k^{-1}(f^{n+1} - f^n)(t_{n+1} - t)$. Therefore (1.4.2) becomes

$$\begin{aligned} v^{n+2} - v^{n+1} &= \int_{t_{n+1}}^{t_{n+2}} \Big[f^{n+1} - k^{-1}(f^{n+1} - f^n)(t_{n+1} - t) \Big] dt \\ &= kf^{n+1} - k^{-1}(f^{n+1} - f^n) \int_{t_{n+1}}^{t_{n+2}} (t_{n+1} - t) dt \\ &= kf^{n+1} - k^{-1}(f^{n+1} - f^n)(-\tfrac{1}{2}k^2) \\ &= \frac{3}{2}kf^{n+1} - \underline{\frac{1}{2}kf^n.} \end{aligned} \tag{1.4.3}$$



(a) Adams (b) backwards differentiation

Figure 1.4.2. Derivation of Adams and backwards differentiation formulas via polynomial interpolation.

More generally, the coefficients of the interpolating polynomial q can be determined with the aid of the **Newton interpolation formula.** To begin with, consider the problem of interpolating a discrete function $\{y^n\}$ in the points $0,\dots,\nu$ by a polynomial $q(t)$ of degree at most ν. Let Δ and ∇ denote the **forward** and **backward difference operators,**

$$\Delta = Z - 1, \qquad \nabla = 1 - Z^{-1}, \tag{1.4.4}$$

where 1 represents the identity operator. For example,

$$\Delta y^n = y^{n+1} - y^n, \qquad \Delta^2 y^n = y^{n+2} - 2y^{n+1} + y^n.$$

Also, let $\binom{a}{j}$ denote the binomial coefficient "a choose j",

$$\binom{a}{j} = \frac{a(a-1)(a-2)\cdots(a-j+1)}{j!}, \tag{1.4.5}$$

defined for integers $j \ge 0$ and arbitrary $a \in \mathbb{C}$. The following is a standard result that can be found in many books of numerical analysis and approximation theory:

NEWTON INTERPOLATION FORMULA

Theorem 1.3. *The polynomial*

$$q(t) = \left[1 + \binom{t}{1}\Delta + \binom{t}{2}\Delta^2 + \cdots + \binom{t}{\nu}\Delta^\nu\right] y^0 \qquad (1.4.6)$$

is the unique polynomial of degree at most ν that interpolates the data $y^0,\dots,y^\nu$ in the points $0,\dots,\nu$.

Proof. First of all, from (1.4.5) it is clear that $\binom{t}{j}$ is a monomial of degree j, and since (1.4.6) describes a linear combination of such terms with $0 \le j \le \nu$, $q(t)$ is evidently a polynomial of degree at most ν.

We need to show that $q(t)$ interpolates $y^0,\dots,y^\nu$. To this end, note that $Z = 1+\Delta$, and therefore

$$Z^j = (1+\Delta)^j = 1 + \binom{j}{1}\Delta + \binom{j}{2}\Delta^2 + \cdots + + \binom{j}{j}\Delta^j$$

for any integer $j \ge 0$ (the binomial formula). If $0 \le j \le \nu$ we may equally well extend the series to term ν,

$$Z^j = 1 + \binom{j}{1}\Delta + \binom{j}{2}\Delta^2 + \cdots + \binom{j}{\nu}\Delta^\nu,$$

since $\binom{j}{m} = 0$ for $m > j$. By taking $t = j$ in (1.4.6), this identity implies that $q(j) = Z^j y^0$ for $0 \le j \le \nu$. In other words, $q(t)$ interpolates the data as required.

Finally, uniqueness of the interpolating polynomial is easy to prove. If $q_1(t)$ and $q_2(t)$ are two polynomials of degree $\le \nu$ that interpolate the data, then $q_1 - q_2$ is polynomial of degree $\le \nu$ that vanishes at the interpolation points, which implies $q_1 - q_2 = 0$ identically since a nonzero polynomial of degree $\le \nu$ can have at most ν zeros. ▮

We want to apply Theorem 1.3 to the derivation of Adams-Bashforth formulas. To do this, we need a version of the theorem that is normalized differently and considerably uglier, though equivalent. Let the points $0,\dots,\nu$ be replaced by the points $t_0,\dots,t_{-\nu}$. Then from Theorem 1.3, or by a proof from scratch, one can readily show that the polynomial

$$q(t) = \left[1 - \binom{-t/k}{1}\nabla + \binom{-t/k}{2}\nabla^2 - \cdots + (-1)^\nu \binom{-t/k}{\nu}\nabla^\nu\right] y^0 \qquad (1.4.7)$$

is the unique polynomial of degree at most ν that interpolates the data $y^{-\nu},\dots,$ y^0 in the points $t_{-\nu},\dots,t_0$. Note that among other changes, Δ has been replaced by ∇.

Now let us replace y by f, ν by $s-1$, and t by $t-t_{n+s-1}$, hence $t_{-\nu},\dots,t_0$ by $t_n,\dots,t_{n+s-1}$. Equation (1.4.7) then becomes

$$q(t) = \left[1-\binom{(t_{n+s-1}-t)/k}{1}\nabla+\cdots+(-1)^{s-1}\binom{(t_{n+s-1}-t)/k}{s-1}\nabla^{s-1}\right]f^{n+s-1}. \tag{1.4.8}$$

Inserting this expression in (1.4.2) gives

$$v^{n+s}-v^{n+s-1} = k\sum_{j=0}^{s-1}\gamma_j\,\nabla^j f^{n+s-1},$$

where

$$\gamma_j = \frac{(-1)^j}{k}\int_{t_{n+s-1}}^{t_{n+s}}\binom{(t_{n+s-1}-t)/k}{j}dt = (-1)^j\int_0^1\binom{-\tau}{j}d\tau.$$

The first few values γ_j are

$$\begin{array}{lll} \gamma_0 = 1, & \gamma_3 = \dfrac{3}{8}, & \gamma_6 = \dfrac{19087}{60480}, \\[2ex] \gamma_1 = \dfrac{1}{2}, & \gamma_4 = \dfrac{251}{720}, & \gamma_7 = \dfrac{5257}{17280}, \\[2ex] \gamma_2 = \dfrac{5}{12}, & \gamma_5 = \dfrac{95}{288}, & \gamma_8 = \dfrac{1070017}{3628800}. \end{array} \tag{1.4.9}$$

The following theorem summarizes this derivation and some related facts:

ADAMS-BASHFORTH FORMULAS

Theorem 1.4. *For any $s \geq 1$, the s-step Adams-Bashforth formula has order of accuracy s, and is given by*

$$v^{n+s} = v^{n+s-1} + k \sum_{j=0}^{s-1} \gamma_j \nabla^j f^{n+s-1}, \qquad \gamma_j = (-1)^j \int_0^1 \binom{-\tau}{j} d\tau. \qquad (1.4.10)$$

For $j \geq 0$, the coefficients γ_j satisfy the recurrence relation

$$\gamma_j + \frac{1}{2}\gamma_{j-1} + \frac{1}{3}\gamma_{j-2} + \cdots + \frac{1}{j+1}\gamma_0 = 1. \qquad (1.4.11)$$

Proof. We have already shown above that the coefficients (1.4.10) correspond to the linear multistep formula based on polynomial interpolation. To prove the theorem, we must show three things more: that the order of accuracy is as high as possible, so that these are indeed Adams-Bashforth formulas; that the order of accuracy is s; and that (1.4.11) holds.

The first claim follows from the lemma stated in Exercise 1.3.6. If $f(u,t)$ is a polynomial in t of degree $\leq s$, then $q(t) = f(u,t)$ in our derivation above, so the formula (1.4.2) gives exact results and its order of accuracy is accordingly $\geq s$. On the other hand any other linear multistep formula with different coefficients would fail to integrate q exactly, since polynomial interpolants are unique, and accordingly would have order of accuracy $< s$. Thus (1.4.10) is indeed the Adams-Bashforth formula.

The second claim can also be based on Exercise 1.3.6. If the s-step Adams-Bashforth formula had order of accuracy $> s$, it would be exact for any problem $u_t = q(t)$ with $q(t)$ equal to a polynomial of degree $s+1$. But there are nonzero polynomials of this degree that interpolate the values $f^0 = \cdots = f^s = 0$, from which we can readily derive counterexamples in the form of initial-value problems with $v(t_{s+1}) = 0$ but $u(t_{s+1}) \neq 0$.

Finally, for a derivation of (1.4.11), the reader is referred to Henrici (1962) or Hairer, Nørsett & Wanner (1987). ▮

EXAMPLE 1.4.1, CONTINUED. To rederive the 2nd-order Adams-Bashforth formula directly from (1.4.10), we calculate

$$v^{n+2} = v^{n+1} + k\gamma_0 f^{n+1} + k\gamma_1 (f^{n+1} - f^n) = v^{n+1} + k\left(\tfrac{3}{2} f^{n+1} - \tfrac{1}{2} f^n\right).$$

EXAMPLE 1.4.2. To obtain the third-order Adams-Bashforth formula, we increment n

to $n+1$ in the formula above and then add one more term $k\gamma_2\nabla^2 f^{n+2}$ to get

$$\begin{aligned} v^{n+3} &= v^{n+2} + k\left(\tfrac{3}{2}f^{n+2} - \tfrac{1}{2}f^{n+1}\right) + \tfrac{5}{12}k\left(f^{n+2} - 2f^{n+1} + f^n\right) \\ &= v^{n+2} + k\left(\tfrac{23}{12}f^{n+2} - \tfrac{16}{12}f^{n+1} + \tfrac{5}{12}f^n\right), \end{aligned}$$

which confirms the result listed in Table 1.4.1.

For Adams-Moulton formulas the derivation is entirely analogous. We have

$$v^{n+s} - v^{n+s-1} = k\sum_{j=0}^{s} \gamma_j^* \nabla^j f^{n+s},$$

where

$$\gamma_j^* = \frac{(-1)^j}{k}\int_{t_{n+s-1}}^{t_{n+s}} \binom{(t_{n+s}-t)/k}{j} dt = (-1)^j \int_{-1}^{0} \binom{-\tau}{j} d\tau,$$

and the first few values γ_j^* are

$$\begin{aligned} &\gamma_0^* = +1, && \gamma_3^* = -\frac{1}{24}, && \gamma_6^* = -\frac{863}{60480}, \\ &\gamma_1^* = -\frac{1}{2}, && \gamma_4^* = -\frac{19}{720}, && \gamma_7^* = -\frac{275}{24192}, \\ &\gamma_2^* = -\frac{1}{12}, && \gamma_5^* = -\frac{3}{160}, && \gamma_8^* = -\frac{33953}{3628800}. \end{aligned} \tag{1.4.12}$$

Notice that these numbers are smaller than before, an observation which is related to the fact that Adams-Moulton formulas generally have smaller error constants than the corresponding Adams-Bashforth formulas.

The analog of Theorem 1.4 is as follows:

ADAMS-MOULTON FORMULAS

Theorem 1.5. *For any $s \ge 0$, the s-step Adams-Moulton formula[*] has order of accuracy $s+1$, and is given by*

$$v^{n+s} = v^{n+s-1} + k\sum_{j=0}^{s} \gamma_j^* \nabla^j f^{n+s}, \qquad \gamma_j^* = (-1)^j \int_{-1}^{0} \binom{-\tau}{j} d\tau, \tag{1.4.13}$$

For $j \ge 1$, the coefficients γ_j^* satisfy the recurrence relation

$$\gamma_j^* + \frac{1}{2}\gamma_{j-1}^* + \frac{1}{3}\gamma_{j-2}^* + \cdots + \frac{1}{j+1}\gamma_0^* = 0. \tag{1.4.14}$$

[*] The "0-step Adams-Moulton formula" of this theorem actually has $s=1$, as indicated in Table 1.4.2, because of the nonzero coefficient α_0.

Proof. Analogous to the proof of Theorem 1.4. ▮

Both sets of coefficients $\{\gamma_j\}$ and $\{\gamma_j^*\}$ can readily be converted into coefficients $\{\beta_j\}$ of our standard representation (1.2.11), and the results for $s \le 4$ are listed above in Tables 1.4.1 and 1.4.2.

Besides Adams formulas, the most important family of linear multistep formulas dates to Curtiss and Hirschfelder in 1952, and is also associated with the name of C. W. Gear. The s-step **backwards differentiation formula** is the optimal implicit linear multistep formula with $\beta_0 = \cdots = \beta_{s-1} = 0$. (An example was given in (1.2.9).) Unlike the Adams formulas, the backwards differentiation formulas allocate the free parameters to the $\{\alpha_j\}$ rather than the $\{\beta_j\}$. These formulas are "maximally implicit" in the sense that the function f enters the calculation only at the level $n+1$. For this reason they are the hardest to implement of all linear multistep formulas, but as we shall see in §§1.7–1.8, they are also the most stable.

To derive the coefficients of backwards differentiation formulas, one can again make use of polynomial interpolation. Now, however, the data are samples of v rather than of f, as suggested in Figure 1.4.2b. Let q be the unique polynomial of degree $\le s$ that interpolates $v^n, \dots, v^{n+s}$. The number v^{n+s} is unknown, but it is natural to define it implicitly by imposing the condition

$$q_t(t_{n+s}) = f^{n+s}. \tag{1.4.15}$$

Like the integral in (1.4.2), the derivative in (1.4.15) represents a linear function of the data, with coefficients that can be computed once and for all, and so this equation constitutes an implicit definition of a linear multistep formula. The coefficients for $s \le 4$ were listed above in Table 1.4.3.

To convert this prescription into numerical coefficients, one can again apply the Newton interpolation formula (see Exercise 1.4.2). However, the proof below is a slicker one based on rational approximation and Theorem 1.2.

BACKWARDS DIFFERENTIATION FORMULAS

Theorem 1.6. *For any $s \ge 1$, the s-step backwards differentiation formula has order of accuracy s, and is given by*

$$\sum_{j=1}^{s} \frac{1}{j} \nabla^j v^{n+s} = k f^{n+s}. \tag{1.4.16}$$

(Note that (1.4.16) is not quite in the standard form (1.2.11); it must be normalized by dividing by the coefficient of v^{n+s}.)

Proof. Let $\rho(z)$ and $\sigma(z) = z^s$ be the characteristic polynomials corresponding to the s-step backwards differentiation formula. (Again we have normalized differently from usual.) By Theorem 1.2, since $\log z = -\log z^{-1}$, the order of accuracy is p if and only if

$$\begin{aligned}\frac{\rho(z)}{z^s} &= -\log z^{-1} + \Theta((z-1)^{p+1}) \\ &= -\Big[(z^{-1}-1) - \tfrac{1}{2}(z^{-1}-1)^2 + \tfrac{1}{3}(z^{-1}-1)^3 - \cdots\Big] + \Theta((z-1)^{p+1}),\end{aligned}$$

that is,

$$\rho(z) = z^s\Big[(1-z^{-1}) + \tfrac{1}{2}(1-z^{-1})^2 + \tfrac{1}{3}(1-z^{-1})^3 + \cdots\Big] + \Theta((z-1)^{p+1}).$$

By definition, $\rho(z)$ is a polynomial of degree at most s with $\rho(0) \neq 0$; equivalently, it is z^s times a polynomial in z^{-1} of degree exactly s. Since $\rho(z)$ maximizes the order of accuracy among all such polynomials, the last formula makes it clear that we must have

$$\rho(z) = z^s\left[(1-z^{-1}) + \cdots + \frac{1}{s}(1-z^{-1})^s\right],$$

with order of accuracy s. This is precisely (1.4.16). ▮

These three families of linear multistep formulas—Adams-Bashforth, Adams-Moulton, and backwards differentiation—are the most important for practical computations. Other families, however, have also been developed over the years. The s-step **Nyström** formula is the optimal explicit linear multistep formula with $\rho(z) = z^s - z^{s-2}$, that is, $\alpha_s = 1$, $\alpha_{s-2} = -1$, and $\alpha_j = 0$ otherwise. The s-step **generalized Milne-Simpson** formula is the optimal implicit linear multistep formula of the same type. Coefficients for these formulas can be obtained by the same process described in Figure 1.4.2a, except that now $q(t)$ is integrated from t_{n+s-2} to t_{n+s}. Like the Adams and backwards differentiation formulas, the s-step Nyström and generalized Milne-Simpson formulas have exactly the order of accuracy one would expect from the number of free parameters (s and $s+1$, respectively), with one exception: the generalized Milne-Simpson formula with $s=2$ has order 4, not 3. This formula is known as the **Simpson** formula for ordinary differential equations:

$$v^{n+2} = v^n + \frac{1}{3}k(f^n + 4f^{n+1} + f^{n+2}). \qquad (1.4.17)$$

EXERCISES

▷ *1.4.1. Second-order backwards differentiation formula.* Derive the coefficients of the 2-step backwards differentiation formula in Table 1.4.3:
(a) By the method of undetermined coefficients;
(b) By Theorem 1.2, making use of the expansion $z^2 = (z-1)^2 + 2(z-1) + 1$;
(c) By interpolation, making use of Theorem 1.3.

▷ *1.4.2. Backwards differentiation formulas.* Derive (1.4.16) from Theorem 1.3.

▷ *1.4.3. Third-order Nyström formula.* Determine the coefficients of the third-order Nyström formula by interpolation, making use of Theorem 1.3.

▷ *1.4.4. Quadrature formulas.* What happens to linear multistep formulas when the function $f(u,t)$ is independent of u? To be specific, what happens to Simpson's formula (1.4.17)? Comment on the effect of various strategies for initializing the point v^1 in an integration of such a problem by Simpson's formula.

1.5. Stability

It is time to introduce one of the central themes of this book: stability. Before 1950, the word stability rarely if ever appeared in papers on numerical methods, but by 1960, at which time computers were widely distributed, its importance had become universally recognized.* Problems of stability affect almost every numerical method for solving differential equations, and they must be confronted head-on.

For both ordinary and partial differential equations, there are two main stability questions that have proved important over the years:

Stability:† If $t > 0$ is held fixed, do the computed values $v(t)$ remain bounded as $k \to 0$?

Eigenvalue stability: If $k > 0$ is held fixed, do the computed values $v(t)$ remain bounded as $t \to \infty$?

These two questions are related, but distinct, and each has important applications. We shall consider the first in this and the following section, and the second in §§1.7,1.8.

The motivation for all discussions of stability is the most fundamental question one could ask about a numerical method: will it give the right answer? Of course one can never expect exact results, so a reasonable way to make the question precise for the initial-value problem (1.1.2) is to ask: if $t > 0$ is a fixed number, and the computation is performed with various step sizes $k > 0$ in exact arithmetic, will $v(t)$ converge to $u(t)$ as $k \to 0$?

A natural conjecture might be that for any consistent linear multistep formula, the answer must be yes. After all, as pointed out in §1.3, such a method commits local errors of size $O(k^{p+1})$ with $p \ge 1$, and there are a total of $\Theta(k^{-1})$ time steps. But a simple argument shows that this conjecture is false. Consider a linear multistep formula based purely on extrapolation of previous values $\{v^n\}$, such as

$$v^{n+2} = 2v^{n+1} - v^n. \qquad (1.5.1)$$

This is a first-order formula with $\rho(z) = (z-1)^2$ and $\sigma(z) = 0$, and in fact we can construct extrapolation formulas of arbitrary order of accuracy by taking $\rho(z) = (z-1)^s$. Yet such a formula cannot possibly converge to the correct solution, for it uses no information from the differential equation!*

Thus local accuracy cannot be sufficient for convergence. The surprising fact is that accuracy plus an additional condition of stability *is* sufficient, and necessary too. In fact this is a rather general principle of numerical analysis, first formulated precisely by Dahlquist for ordinary differential equations and by Lax and Richtmyer for partial differential equations, both in the 1950's. Strang (1985) calls it the "fundamental theorem of numerical analysis."

*See G. Dahlquist, "33 years of numerical instability," *BIT 25 (1985)*, 188–204.

†Stability is also known as "zero-stability" or sometimes "D-stability" for ODEs, and "Lax stability" or "Lax-Richtmyer stability" for PDEs. Eigenvalue stability is also known as "weak stability" or "absolute stability" for ODEs, and "time-stability," "practical stability" or "P-stability" for PDEs. The reason for the word "eigenvalue" will become apparent in §§1.7,1.8.

*Where exactly did the argument suggesting global errors $O(k^p)$ break down? Try to pinpoint it yourself; the answer is in the next section.

Theorem 1.10 in the next section gives a precise statement for the case of linear multistep formulas, and for partial differential equations, see Theorem 4.1.

We begin with a numerical experiment.

EXAMPLE 1.5.1. In the spirit of Example 1.2.1, suppose we solve the initial-value problem

$$u_t = u, \qquad t \in [0,1], \qquad u(0) = 1 \tag{1.5.2}$$

by three different 2-step explicit methods: the extrapolation formula (1.5.1), the second-order Adams-Bashforth formula (1.4.3), and the "optimal" 2-step formula (1.3.21). Again we take exact quantities e^{nk} where needed for starting values. Figure 1.5.1 shows the computed functions $v(t)$ for $k=0.2$ and 0.1. In the first plot, both of the higher-order formulas appear to be performing satisfactorily. In the second, however, large oscillations have begun to develop in the solution based on (1.3.21). Obviously (1.3.21) is useless for this problem.

Table 1.5.1 makes this behavior quantitative by listing the computed results $v(1)$ for $k = 0.2, 0.1, 0.05, 0.025$. As k decreases, the solution based on (1.5.1) converges to an incorrect solution, namely the function $t+1$, and the solution based on (1.3.21) diverges explosively. Notice that none of the numbers are preceded by a *. In this example the instability has been excited by discretization errors, not rounding errors.

Figure 1.5.2 gives a fuller picture of what is going on in this example by plotting the error $|v(1)-e|$ as a function of k (on a log scale, with smaller values of k to the right) for the same three linear multistep formulas as well as the fourth-order Adams-Bashforth formula (1.2.7). The two Adams-Bashforth formulas exhibit clean second-order and fourth-order convergence, as one would expect, showing in the plot as lines of slope approximately -2 and -4, respectively. The extrapolation formula (1.5.1) exhibits zeroth-order convergence—in other words divergence, with the error approaching the constant $e-2$. The formula (1.3.21) diverges explosively.

It is not difficult to see what has gone wrong with (1.3.21). For simplicity, assume k is negligible. Then (1.3.21) becomes

$$v^{n+2} + 4v^{n+1} - 5v^n = 0, \tag{1.5.3}$$

a second-order recurrence relation for $\{v^n\}$. It is easy to verify that both $v^n = 1$ and $v^n = (-5)^n$ are solutions of (1.5.3). (*Caution:* the n in v^n is a superscript, but the n in $(-5)^n$ is an exponent.) Since an arbitrary solution to (1.5.3) is determined by two initial values v^0 and v^1, it follows that any solution can be written in the form

$$v^n = a(1)^n + b(-5)^n \tag{1.5.4}$$

for some constants a and b. What has happened in our experiment is that b has ended up nonzero—there is some energy in the mode $(-5)^n$, and an explosion has taken place. In fact, if the computation with $k=0.025$ is carried out to one more time step, v^n takes the value 4.60×10^{19}, which is $-4.93 \approx -5$ times the final quantity listed in the table. So the assumption that k was negligible was not too bad.

Although Example 1.5.1 is very simple, it exhibits the essential mechanism of instability in the numerical solution of ordinary differential equations by linear multistep formulas: a recurrence relation that admits an exponentially

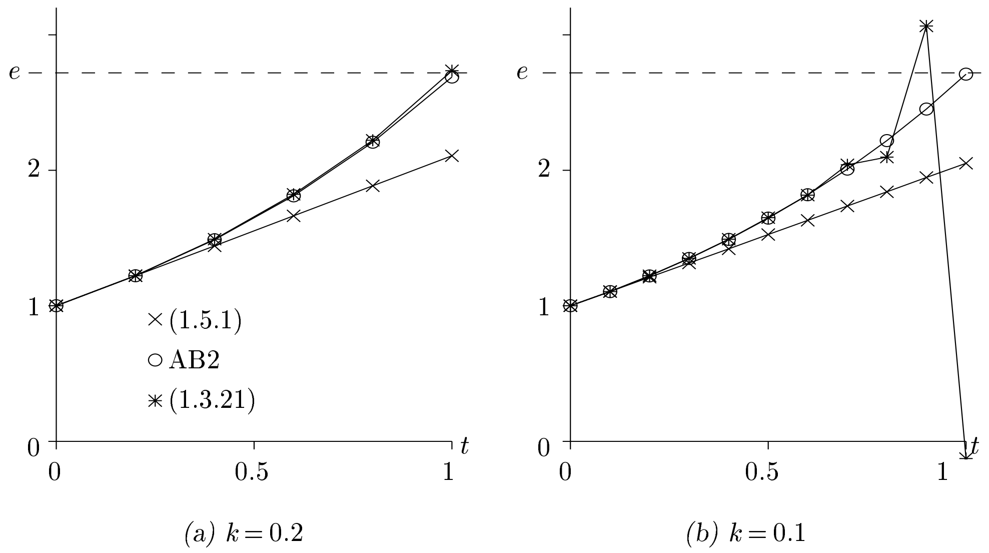


Figure 1.5.1. Solution of (1.5.2) by three explicit two-step linear multistep formulas. The high-order formula (1.3.21) looks good at first, but becomes unstable when the time step is halved.

	$k=0.2$	$k=0.1$	$k=0.05$	$k=0.025$
Extrapolation (1.5.1)	2.10701	2.05171	2.02542	2.01260
2nd-order A-B (1.4.4)	2.68771	2.70881	2.71568	2.71760
"optimal" (1.3.21)	2.73433	-0.12720	-1.62×10^6	-9.34×10^{18}

Table 1.5.1. Computed values $v(1)\approx e$ for the initial-value problem (1.5.2).

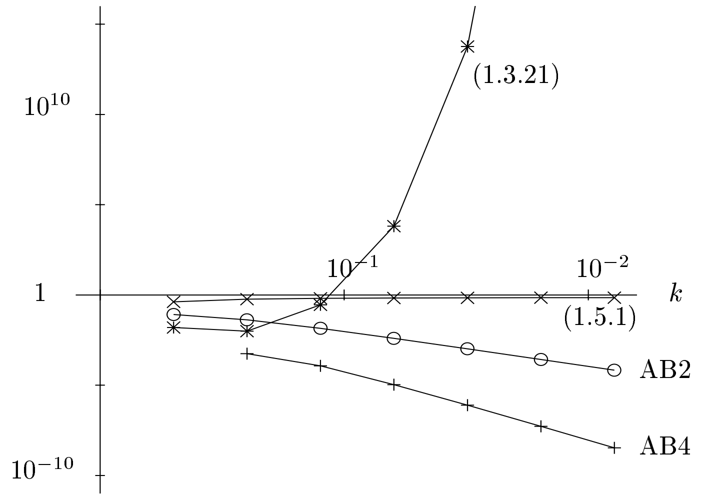


Figure 1.5.2. Error $|v(1)-e|$ as a function of time step k for the solution of (1.5.2) by four explicit linear multistep formulas.

growing solution z^n for some $|z| > 1$. Such a solution is sometimes known as a **parasitic solution** of the numerical method, since it is introduced by the discretization rather than the ordinary differential equation itself. It is a general principle that if such a mode exists, it will almost always be excited by either discretization or rounding errors, or by errors in the initial conditions. Of course in principle, the coefficient b in (1.5.4) might turn out to be identically zero, but in practice this possibility can usually be ignored. Even if b were zero at $t = 0$, it would soon become nonzero due to variable coefficients, nonlinearity, or rounding errors, and the z^n growth would then take over sooner or later.*

The analysis of Example 1.5.1 can be generalized as follows. Given any linear multistep formula, consider the associated recurrence relation

$$\rho(Z)v^n = \sum_{j=0}^{s} \alpha_j v^{n+j} = 0 \tag{1.5.5}$$

obtained from (1.3.4) with $k = 0$. We define:

A linear multistep formula is **stable** *if all solutions $\{v^n\}$ of the recurrence relation (1.5.5) are bounded as $n \to \infty$.*

This means that for any function $\{v^n\}$ that satisfies (1.5.5), there exists a constant $M > 0$ such that $|v^n| \le M$ for all $n \ge 0$. We shall refer to the recurrence relation itself as **stable**, as well as the linear multistep formula.

There is an elementary criterion for determining stability of a linear multistep formula, based on the characteristic polynomial ρ.

ROOT CONDITION FOR STABILITY

Theorem 1.7. *A linear multistep formula is stable if and only if all the roots of $\rho(z)$ satisfy $|z| \le 1$, and any root with $|z| = 1$ is simple.*

A "simple" root is a root of multiplicity 1.

First proof. To prove this theorem, we need to investigate all possible solutions $\{v^n\}$, $n \ge 0$, of the recurrence relation (1.5.5). Since any such solution is determined by its initial values $v^0, \dots, v^{s-1}$, the set of all of them is a

*A general observation is that when any linear process admits an exponentially growing solution in theory, that growth will almost invariably appear in practice. Only in nonlinear situations can the existence of exponentially growing modes fail to make itself felt. A somewhat far-flung example comes from numerical linear algebra. In the solution of an $N \times N$ matrix problems by Gaussian elimination with partial pivoting—a highly nonlinear process—an explosion of rounding errors at the rate 2^{N-1} can occur in principle, but almost never does in practice (Trefethen & Schreiber, "Average-case stability of Gaussian elimination," *SIAM J. Matrix Anal. Applics.*, 1990).

vector space of dimension s. Therefore if we can find s linearly independent solutions v^n, these will form a basis of the space of all possible solutions, and the recurrence relation will be stable if and only if each of the basis solutions is bounded as $n \to \infty$.

It is easy to verify that if z is any root of $\rho(z)$, then

$$v^n = z^n \tag{1.5.6}$$

is one solution of (1.5.5). (Again, on the left n is a superscript and on the right it is an exponent. If $z=0$ we define $z^0=1$.) If ρ has s distinct roots, then these functions constitute a basis. Since each function (1.5.6) is bounded if and only if $|z| \le 1$, this proves the theorem in the case of distinct roots.

On the other hand suppose that $\rho(z)$ has a root z of multiplicity $m \ge 2$. Then it can readily be verified that each of the functions

$$v^n = nz^n, \qquad v^n = n^2 z^n, \quad \dots, \quad v^n = n^{m-1} z^n \tag{1.5.7}$$

is an additional solution of (1.5.5), and clearly they are all linearly independent since degree-$(m-1)$ polynomial interpolants in m points are unique, to say nothing of ∞ points! (If $z=0$, we replace $n^j z^n$ by the function that takes the value 1 at $n=j$ and 0 elsewhere.) These functions are bounded if and only if $|z|<1$, and this finishes the proof of the theorem in the general case. ▮

Alternative proof based on linear algebra. The proof above is simple and complete, but there is another way of looking at Theorem 1.7 that involves a technique of general importance. Let us rewrite the s-step recurrence relation as a 1-step matrix operation on vectors $\mathbf{v}$ of length s:

$$\begin{pmatrix} v^{n+1} \\ v^{n+2} \\ \\ \vdots \\ \\ v^{n+s} \end{pmatrix} = \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & & & & \\ & & & \ddots & \ddots & \\ & & & & 0 & 1 \\ -\alpha_0 & & \cdots & & -\alpha_{s-2} & -\alpha_{s-1} \end{pmatrix} \begin{pmatrix} v^n \\ v^{n+1} \\ \\ \vdots \\ \\ v^{n+s-1} \end{pmatrix}. \tag{1.5.8}$$

That is,

$$\mathbf{v}^{n+1} = A\mathbf{v}^n, \tag{1.5.9}$$

or after n steps,

$$\mathbf{v}^n = A^n \mathbf{v}^0, \tag{1.5.10}$$

where A^n denotes the nth power of the matrix A. This is a discrete analog of the reduction of higher-order ODEs to first-order systems described in §1.1.

The scalar sequence $\{v^n\}$ will be bounded as $n \to \infty$ if and only if the vector sequence $\{\mathbf{v}^n\}$ is bounded, and $\{\mathbf{v}^n\}$ will in turn be bounded if and only if the elements of A^n are bounded. Thus we have reduced stability to a problem of growth or boundedness of the powers of a matrix.

A matrix A of the form (1.5.8) is known as a **companion matrix**, and one can verify that $\det(zI - A) = \rho(z)$ for any z, where $\rho(z)$ is defined by (1.3.1) as usual.* In other words, the characteristic polynomial of the matrix A is the same as the characteristic polynomial of the linear multistep formula. Therefore the set of eigenvalues of A is the same as the set of roots of ρ, and these eigenvalues determine how the powers A^n behave asymptotically as $n \to \infty$. To make the connection precise one can look at the similarity transformation that brings A into **Jordan canonical form**,

$$A = SJS^{-1}.$$

Here S is an $s \times s$ nonsingular matrix, and J is an $s \times s$ matrix consisting of all zeros except for a set of **Jordan blocks** J_i along the diagonal with the form

$$J_i = \begin{pmatrix} z_i & 1 & & & \mathbf{0} \\ & z_i & 1 & & \\ & & \ddots & \ddots & \\ \mathbf{0} & & & z_i & 1 \\ & & & & z_i \end{pmatrix},$$

where z_i is one of the eigenvalues of A. Every matrix has a Jordan canonical form, and for matrices in general, each eigenvalue may appear in several Jordan blocks. For a companion matrix, however, there is exactly one Jordan block for each eigenvalue, with dimension equal to the multiplicity m_i of that eigenvalue. (Proof: $zI - A$ has rank $\geq s-1$ for any z, since its upper-right $(s-1)\times(s-1)$ block is obviously nonsingular. Such a matrix is called **nonderogatory**.) Now the powers of A are

$$A^n = (SJS^{-1})\cdots(SJS^{-1}) = SJ^nS^{-1},$$

so their growth or boundedness is determined by the growth or boundedness

* One way to verify it is to look for eigenvectors of the form $(1, z, \ldots, z^{s-1})^T$, where z is the eigenvalue.

of the powers J^n, which can be written down explicitly:

$$J_i^n = \begin{pmatrix} z_i^n & \binom{n}{1} z_i^{n-1} & & \cdots & \binom{n}{m_i-1} z_i^{n+1-m_i} \\ & z_i^n & \binom{n}{1} z_i^{n-1} & & \\ & & \ddots & \ddots & \vdots \\ & & & z_i^n & \binom{n}{1} z_i^{n-1} \\ & \mathbf{0} & & & z_i^n \end{pmatrix}.$$

If $|z_i| < 1$, these elements approach 0 as $n \to \infty$. If $|z_i| > 1$, they approach ∞. If $|z_i| = 1$, they are bounded in the case of a 1×1 block, but unbounded if $m_i \geq 2$. ∎

The reader should study Theorem 1.7 and both of its proofs until he or she is quite comfortable with them. These tricks for analyzing recurrence relations come up so often that they are worth remembering.

EXAMPLE 1.5.2. Let us test the stability of various linear multistep formulas considered up to this point. Any Adams-Bashforth or Adams-Moulton formula has $\rho(z) = z^s - z^{s-1}$, with roots $\{1, 0, \ldots, 0\}$, so these methods are stable. The Nyström and generalized Milne-Simpson formulas have $\rho(z) = z^s - z^{s-2}$, with roots $\{1, -1, 0, \ldots, 0\}$, so they are stable too. The scheme (1.3.21) that caused trouble in Example 1.5.1 has $\rho(z) = z^2 + 4z - 5$, with roots $\{1, -5\}$, so it is certainly unstable. As for the less dramatically unsuccessful formula (1.5.1), it has $\rho(z) = z^2 - 2z + 1 = (z-1)^2$, with a multiple root $\{1, 1\}$, so it counts as unstable too since it admits the growing solution $v^n = n$. The higher-order extrapolation formula defined by $\rho(z) = (z-1)^s$, $\sigma(z) = 0$ admits the additional solutions $n^2, n^3, \ldots, n^{s-1}$, making for an instability more pronounced but still algebraic rather than exponential.

Note that by Theorem 1.2, any consistent linear multistep formula has a root of $\rho(z)$ at $z = 1$, and if the formula is stable, then Theorem 1.7 ensures that this root is simple. It is called the **principal root** of ρ, for it is the one that tracks the differential equation. The additional consistency condition $\rho'(1) = \sigma(1)$ of Theorem 1.2 amounts to the condition that if z is perturbed away from 1, the principal root behaves correctly to leading order.

In §1.3 an analogy was mentioned between linear multistep formulas and recursive digital filters. In digital signal processing, one demands $|z| < 1$ for stability; why then does Theorem 1.7 contain the weaker condition $|z| \leq 1$? The answer is that unlike the usual filters, an ODE integrator must remember the past: even for values of t where f is zero, u is in general nonzero. If all roots z satisfied $|z| < 1$, the influence of the past would die away exponentially.

Theorem 1.7 leaves us in a remarkable situation, summarized in Figure 1.5.3. By Theorem 1.2, consistency is the condition that $\rho(z)/\sigma(z)$ matches $\log z$ to at least second order at $z = 1$. By Theorem 1.7, stability is the condition that all roots of $\rho(z)$ lie in $|z| \le 1$, with simple roots only permitted on $|z| = 1$. Thus the two crucial properties of linear multistep formulas have been reduced completely to algebraic questions concerning a rational function. The proofs of many results in the theory of linear multistep formulas, such as Theorems 1.8 and 1.9 below, consist of arguments of pure complex analysis of rational functions, having nothing superficially to do with ordinary differential equations.

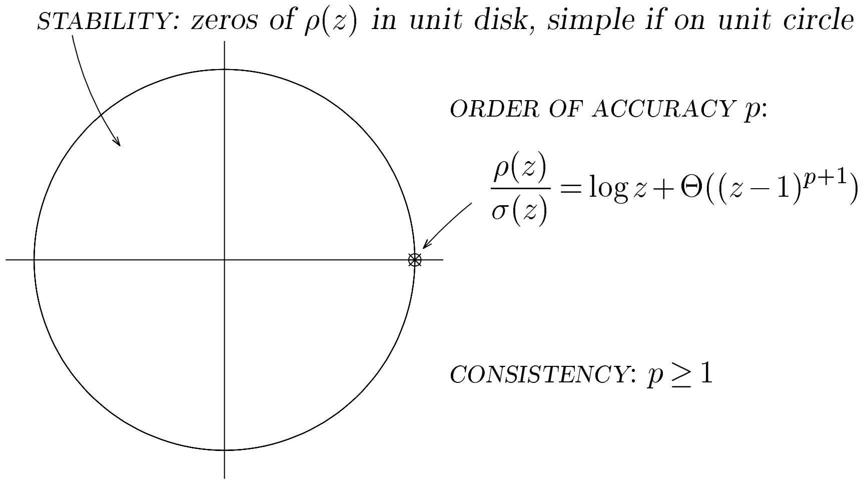

STABILITY: zeros of $\rho(z)$ in unit disk, simple if on unit circle
ORDER OF ACCURACY p:
$\frac{\rho(z)}{\sigma(z)} = \log z + \Theta((z-1)^{p+1})$
CONSISTENCY: $p \ge 1$

Figure 1.5.3. Stability, consistency, and order of accuracy as algebraic conditions on the rational function $\rho(z)/\sigma(z)$.

The following theorem summarizes the stability of the standard families of linear multistep formula that we have discussed.

STABILITY OF STANDARD LINEAR MULTISTEP FORMULAS

Theorem 1.8. *The s-step Adams-Bashforth, Adams-Moulton, Nyström, and generalized Milne-Simpson formulas are stable for all $s \ge 1$. The s-step backwards differentiation formulas are stable for $1 \le s \le 6$, but unstable for $s \ge 7$.*

Proof. The results for the first four families of linear multistep formulas follow easily from Theorem 1.7, as described in Example 1.5.2. The analysis of backwards differentiation formulas is a more complicated matter, since the polynomials $\rho(z)$ are no longer trivial. Instability for $s \ge 7$ was recognized numerically in the 1950's (Mitchell and Craggs, 1953), but not proved mathematically until 1971 (Cryer, 1972). An elegant short proof of instability for $s \ge 12$ was devised by Hairer and Wanner in 1983 and can be found on pp. 328–331 of (Hairer, Nørsett & Wanner, 1987). ▮

To close this section we shall present an important result that is known as the **first Dahlquist stability barrier**. In §1.3 we mentioned that an s-step linear multistep formula can have order $2s$; why not simply use that high-order formula and forget special classes like Adams and Nyström methods? Dahlquist's famous theorem confirms that the answer is an impassable barrier of stability.

FIRST DAHLQUIST STABILITY BARRIER

Theorem 1.9. *The order of accuracy p of a stable s-step linear multistep formula satisfies*

$$p \le \begin{cases} s+2 & \textit{if } s \textit{ is even,} \\ s+1 & \textit{if } s \textit{ is odd,} \\ s & \textit{if the formula is explicit.} \end{cases} \tag{1.5.11}$$

Proof. Various proofs of Theorem 1.9 have been published, beginning with the original one by Dahlquist in 1956. More recent proofs have been based on the beautiful idea of "order stars" introduced by Wanner, Hairer, and Nørsett (*BIT,* 1978). The following argument is adapted from "A proof of the first Dahlquist barrier by order stars," by A. Iserles and S. P. Nørsett, *BIT,* 1984. Though fundamentally correct, it is somewhat casual about geometric details; for a more detailed treatment see that paper or the book *Order Stars* by the same authors.

Let $\rho(z)$ and $\sigma(z)$ be the characteristic polynomials corresponding to a stable s-step linear multistep formula of order of accuracy p. The key idea is to look at level lines of the function we have been dealing with since (1.3.18),

$$\varphi(z) = \frac{\rho(z)}{\sigma(z)} - \log z. \tag{1.5.12}$$

This function is analytic throughout the complex plane except near zeros of $\sigma(z)$, provided one introduces a branch cut going to the point $z=0$. (Alternatively, one can work with $e^{\varphi(z)}$ and eliminate the need for a branch cut.) In particular, $\varphi(z)$ is analytic at $z=1$. (If $\sigma(1)=0$, $\rho(z)$ and $\sigma(z)$ have a

Figure 1.5.4. Order star defined by (1.5.14) for the 5th-order Adams-Bashforth formula. The zeros of $\rho(z)$ (at $z=0$) are marked by o and the zeros of $\sigma(z)$ are marked by $*$. The 6-fold daisy at $z=1$ reflects the order of accuracy 5 of this formula. If there were a bounded, shaded "finger" that did not intersect the unit disk (dashed), the formula would be unstable.

common factor $z-1$ and the linear multistep formula is not in simplest form.) By Theorem 1.2, its behavior near there is

$$\varphi(z) = C(z-1)^{p+1} + O((z-1)^{p+2}) \tag{1.5.13}$$

for $C = C_{p+1}/\sigma(1) \neq 0$. Now let A, the **order star** for the linear multistep formula, be the set defined by

$$A = \{z \in \mathbb{C}\colon \operatorname{Re}\varphi(z) > 0\}. \tag{1.5.14}$$

In other words, it is the inverse image of the right half-plane under φ. Figure 1.5.4 shows the order star for the 5th-order Adams-Bashforth formula.

The power of order stars comes from their ability to couple local and global properties of a function by way of geometric arguments. The global property that concerns us is stability: the zeros of $\rho(z)$ must lie in the unit disk. The local property that concerns us is order of accuracy: from (1.5.13) it follows that near $z=1$, A must look like a daisy with $p+1$ evenly spaced petals—or "fingers", as they are sometimes called. In Figure 1.5.4 the number of fingers is 6.

Since the linear multistep formula is stable, all the zeros of $\rho(z)$ must lie in A (or possibly on its boundary in the case of a zero with $|z|=1$). This follows from (1.5.12) and (1.5.14) since the zeros have to satisfy $|z|\le 1$, hence $\operatorname{Re}\log z \le 0$. [At this point some reasoning by the argument principle of complex analysis is needed, which will be filled in later.] The conclusion is this: any bounded finger of A (i.e., not extending to ∞) must contain one of the zeros of $\rho(z)$. Consequence: *for stability, every bounded finger has to intersect the unit disk.*

To finish the argument, let us now assume that the linear multistep formula is explicit; similar arguments apply in the implicit case (Exercise 1.5.7). Our goal is then to prove $p\le s$. To do this we shall count zeros of $\operatorname{Re}\varphi(z)$ on the unit circle $|z|=1$. Let M be this number of zeros, counted with multiplicity; obviously M must be even. How big can M be? Note first that

$$2\operatorname{Re}\varphi(z) = \frac{\rho(z)}{\sigma(z)} + \frac{\overline{\rho(z)}}{\overline{\sigma(z)}} = \frac{\rho(z)\overline{\sigma(z)} + \overline{\rho(z)}\sigma(z)}{\sigma(z)\overline{\sigma(z)}}$$

on the unit circle $|z|=1$, since $\operatorname{Re}\log z = 0$ there. Since the linear multistep formula is explicit, $\sigma(z)$ has degree $\le s-1$, so the numerator is a polynomial of degree $\le 2s-1$, which implies that M, being even, satisfies

$$M \le 2s-2. \tag{1.5.15}$$

Now, how many zeros are implied by order of accuracy p? First, there is a zero of multiplicity $p+1$ at $z=1$. In addition, there is a zero wherever the boundary of A crosses the unit circle—in Figure 1.5.4, four such zeros altogether. To count these, we note that $(p+1)/2$ fingers of A begin at $z=1$ outside the unit circle. One of these may be unbounded and go to infinity, but the other $(p-1)/2$ must intersect the unit disk and cross the unit circle on the way. Two of those may cross the unit circle just once (one each in the upper and lower half-planes), but all the remaining $(p-5)/2$ fingers are trapped inside those fingers and must cross the unit circle twice. All told, we count

$$M \ge p+1+2+2(p-5)/2 = 2p-2. \tag{1.5.16}$$

Combining (1.5.15) and (1.5.16) gives $p\le s$, as required.

Similar arguments apply if the linear multistep formula is implicit. ∎

The restrictions of Theorem 1.9 are tight! In effect, they show, half of the available parameters in a linear multistep formula are wasted.

Theorems 1.3, 1.8, and 1.9 imply that the Adams-Bashforth and Nyström formulas are all optimal in the sense that they attain the bound $p=s$ for stable explicit formulas, and the Adams-Moulton and generalized Milne-Simpson formulas of odd step number s are also optimal, with $p=s+1$. Simpson's rule, with $p=s+2$, is an example of an optimal implicit formula with even step number. It can be shown that for any optimal implicit formula with even step number s, the roots of $\rho(z)$ all lie on the unit circle $|z|=1$ (Henrici, 1962, p. 232). Thus the stability of these formulas is always of a borderline kind.

EXERCISES

▷ *1.5.1. Backwards differentiation formulas.* Show that the following backwards differentiation formulas from Table 1.4.3 are stable: *(a)* $s=2$, *(b)* $s=3$.

▷ *1.5.2.* Prove:
(a) Any consistent 1-step linear multistep formula is stable.
(b) Any consistent linear multistep formula with $\rho(z)=\sigma(z)$ is unstable.

▷ *1.5.3.* Consider the s-step explicit linear multistep formula of optimal order of accuracy of the form $v^{n+s}=v^n+k\sum_{j=0}^{s-1}\beta_j f^{n+j}$.
(a) For which s is it stable?
(b) Derive the coefficients for the case $s=2$.
(c) Likewise for $s=3$.

▷ *1.5.4. Padé approximation.* The *type* (μ,ν) *Padé approximant* to $\log z$ at $z=1$ is defined as the unique rational function $\rho(z)/\sigma(z)$ of type (μ,ν) (i.e., with numerator of degree $\le\mu$ and denominator of degree $\le\nu$) that satisfies

$$\frac{\rho(z)}{\sigma(z)}=\log z+O((z-1)^{\mu+\nu+1}) \qquad \text{as } z\to 1. \tag{1.5.17}$$

By Theorem 1.2, taking $\mu=\nu=s$ gives the maximally accurate implicit s-step formula, with order of accuracy at least $p=2s$, and taking $\mu=s$, $\nu=s-1$ gives the maximally accurate explicit s-step formula, with order of accuracy at least $p=2s-1$.

Without performing any calculations, but appealing only to the uniqueness of Padé approximants and to theorems stated in this text, determine whether each of the following Padé schemes is stable or unstable, and what it is called if we have given a name for it. In each case, state exactly what theorems you have used.
(a) $s=1$, explicit. *(b)* $s=1$, implicit.
(c) $s=2$, explicit. *(d)* $s=2$, implicit.
(e) $s=3$, explicit. *(f)* $s=3$, implicit.
(g) If you have access to a symbolic calculator such as Macsyma, Maple, or Mathematica, use it to calculate the coefficients of the linear multistep formulas *(a)*–*(f)*, and confirm that the names you have identified above are correct.

▷ *1.5.5. Linear combinations of linear multistep formulas.* In Example 1.3.1 we showed that the trapezoid formula has error constant $-\frac{1}{12}$ and the midpoint formula has error constant $\frac{1}{3}$.

(a) Devise a linear combination of these two formulas that has order of accuracy higher than 2. What is the order of accuracy?

(b) Show that the formula of *(a)* is stable.

(c) In general, is a convex linear combination of two stable linear multistep formulas always stable? (A convex linear combination has the form $a \times \mathit{formula}_1 + (1-a) \times \mathit{formula}_2$ with $0 \le a \le 1$.) Prove it or give a counterexample.

▶ *1.5.6. Order stars.* Write a program to generate order star plots like that of Figure 1.5.4. This may be reasonably easy in a higher-level language like Matlab. Plot order stars for the following linear multistep formulas, among others:

(a) 4th-order Adams-Bashforth;

(b) 4th-order Adams-Moulton;

(c) 4th-order backwards differentiation;

(d) the unstable formula (1.3.21).

▷ *1.5.7.* The proof of Theorem 1.9 in the text covered only the case of an explicit linear multistep formula. Show what modifications are necessary for the implicit case.

1.6. Convergence and the Dahlquist equivalence theorem

Up to this point we have talked about accuracy, consistency, and stability, but we have yet to establish that a linear multistep formula with these admirable properties will actually work. After one more definition we shall be able to remedy that. To set the stage, let us return to the footnote on p. 41. If a linear multistep formula has local accuracy $O(k^{p+1})$, how can it fail to have global accuracy $O(k^p)$? The answer has nothing to do with the fact that f may be nonlinear, for we have assumed that f is Lipschitz continuous, and that is enough to keep the behavior of small perturbations essentially linear, so long as they remain small.

The flaw in the $O(k^p)$ argument is as follows. Even though a discretization error may be small when it is first introduced, *from that point on it may grow*—often exponentially. The global error at step n consists of the superposition not simply of the local errors at all previous steps, but of what these local errors *have become* at step n. Consistency is the condition that the local errors are small at the time that they are introduced, provided that the function being dealt with is smooth. The additional condition of stability is needed to make sure that they do not become bigger as the calculation proceeds.

The purpose of this section is to describe the ideas related to the Dahlquist Equivalence Theorem that have been built to describe these phenomena. The rigorous statement of the argument above appears in the proof of Theorem 1.10, below. If f is linear, the argument becomes simpler; see Exercise 1.6.3.

To begin we must define the notion of **convergence**. The standard definition requires the linear multistep formula to be applicable not just to a particular initial-value problem, but to an arbitrary initial-value problem with Lipschitz continuous data f. It must also work for any starting values $v^0,\dots,v^{s-1}$ that satisfy the consistency condition*

$$\|v^n - u_0\| = o(1) \quad \text{as } k\to 0, \qquad 0\le n\le s-1. \tag{1.6.1}$$

Equivalent statements of the same condition would be

$$\lim \|v^n - u_0\| = 0, \qquad \lim v^n = u_0, \qquad \text{or} \quad v^n = u_0 + o(1)$$

as $k\to 0$ for $0\le n\le s-1$ (see Exercise α.3(b)).

> *A linear multistep formula is* **convergent** *if, for all initial-value problems (1.1.2) satisfying the conditions of Theorem 1.1 on an interval* $[0,T]$, *and all starting values* $v^0,\dots,v^{s-1}$ *satisfying (1.6.1), the solution* v^n *satisfies*
>
> $$\|v(t) - u(t)\| = o(1) \qquad \textit{as } k\to 0 \tag{1.6.2}$$
>
> *uniformly for all* $t\in[0,T]$.

*The choice of the norm $\|\cdot\|$ doesn't matter, since all norms on a finite-dimensional space are equivalent. For a scalar ODE the norm can be replaced by the absolute value.

("Uniformly" means that $\|v(t)-u(t)\|$ is bounded by a fixed function $\phi(k)=o(1)$ as $k\to 0$, independent of t.) To put it in words, a convergent linear multistep formula is one that is guaranteed to get the right answer in the limit $k\to 0$, for each t in a bounded interval $[0,T]$—assuming there are no rounding errors.

Some remarks on this definition are in order. First, (1.6.2) is a statement about the limit $k\to 0$ for each *fixed* value of t. Since $v(t)$ is defined only on a discrete grid, this means that k is implicitly restricted to values t/n in this limit process. However, it is possible to loosen this restriction by requiring only $t_{n_k}\to t$ as $k\to 0$.

Second, to be convergent a linear multistep formula must work for *all* well-posed initial-value problems, not just one. This may seem an unnaturally strict requirement, but it is not. A formula that worked for only a restricted class of initial-value problems—for example, those with sufficiently smooth coefficients—would be a fragile object, sensitive to perturbations.

Third, v^n refers to the exact solution of the linear multistep formula; rounding errors are not accounted for in the definition of convergence. This is a reasonable simplification because discretization errors and errors in the starting values *are* included, via (1.6.1), and the stability phenomena that govern these various sources of error are nearly the same. Alternatively, the theory can be broadened to include rounding errors explicitly.

Finally, note that condition (1.6.1) is quite weak, requiring only $o(1)$ accuracy of starting values, or $O(k)$ if the errors happen to follow an integral power of k. This is in contrast to the $O(k^2)$ accuracy required at subsequent time steps by the definition of consistency. The reason for this discrepancy is simple enough: starting errors are introduced only at s time steps, while subsequent errors are introduced $\Theta(k^{-1})$ times. Therefore one can get away with one order lower accuracy in starting values than in the time integration. For partial differential equations we shall find that analogously, one can usually get away with one order lower accuracy in discretizing boundary conditions.

We come now to a remarkable result that might be called the fundamental theorem of linear multistep formulas. Like much of the material in this and the last section, it first appeared in a classic paper by Germund Dahlquist in 1956.*

DAHLQUIST EQUIVALENCE THEOREM

Theorem 1.10. *A linear multistep formula is convergent if and only if it is consistent and stable.*

Before proving this theorem, let us pause for a moment to consider what it says. It seems obvious that a linear multistep formula that admits unstable solutions is unlikely to be useful, but it is not so obvious that instability is the *only* thing that can go wrong. For example, clearly the extrapolation formula (1.5.1) of Example 1.5.1 must be useless, since it ignores $\{f^n\}$, but why should that have anything to do with instability? Yet Theorem 1.10 asserts that any useless consistent formula must be unstable! And indeed, we have verified this prediction for (1.5.1) in Example 1.5.2.

* G. Dahlquist, "Convergence and stability in the numerical integration of ordinary differential equations," *Math. Scand. 4* (1956), 33–53. There are important earlier references in this area, too, notably an influential article by Richard von Mises in 1930, who proved convergence of Adams methods. The classic reference in book form on this material is P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations,* Wiley, 1962. A modern classic is the two-volume series by Hairer, Nørsett and Wanner.

An indication of the power of Theorem 1.10 is that here, as in all of the developments of this chapter, the initial-value problems under consideration may be linear or nonlinear. For partial differential equations we shall see that the best known analog of Theorem 1.10—the Lax Equivalence Theorem—requires linearity.

Proof. Theorem 1.10 is not mathematically deep; the art is in the definitions. To prove it we shall establish three implications:

(a) convergent $\Rightarrow$ stable;

(b) convergent $\Rightarrow$ consistent;

(c) consistent + stable $\Rightarrow$ convergent.

In outline, *(a)* and *(b)* are proved by applying the linear multistep formula to particular initial-value problems (1.1.2), and *(c)* is proved by verifying that so long as one has stability, local errors cannot add up too much.

(a) Convergent $\Rightarrow$ stable. If the linear multistep formula is convergent, then (1.6.2) must hold in particular for the initial-value problem $u_t = 0$, $u(0) = 0$, whose solution is $u(t) \equiv 0$. Now suppose the formula is unstable. Then by the proof of Theorem 1.7, it admits a particular solution $V^n = z^n$ with $|z| > 1$, or $V^n = nz^n$ with $|z| = 1$. In either case, suppose the starting values for the time integration are taken as

$$v^n = \sqrt{k}\,V^n, \qquad 0 \le n \le s-1, \tag{1.6.3}$$

where k is as always the time step. Then the computed solution for all n will be precisely $\sqrt{k}V^n$. But whereas the factor $\sqrt{k}$ ensures that the starting values approach $u_0 = 0$ as $k \to 0$, as required by (1.6.1), $|\sqrt{k}V^n|$ approaches ∞ for any $t = nk > 0$. Thus (1.6.2) does not hold, and the formula is not convergent.

(b) Convergent $\Rightarrow$ consistent. To prove consistency, by Theorem 1.2, we must show $\rho(1) = 0$ and $\rho'(1) = \sigma(1)$. For the first, consider the particular initial-value problem $u_t = 0$, $u(0) = 1$, whose solution is $u(t) \equiv 1$, and the particular initial values $v^0 = \cdots = v^{s-1} = 1$. Since $f(u,t) = 0$, the linear multistep formula reduces to (1.5.5), and convergence implies that the solutions to this recurrence relation for the given initial data satisfy $v(1) \to 1$ as $k \to 0$. Since v^n does not depend on k, this is the same as saying $v^n \to 1$ as $n \to \infty$ for fixed $k > 0$, and by (1.5.5), this implies $\rho(1) = 0$.

To show $\rho'(1) = \sigma(1)$, consider the particular initial-value problem $u_t = 1$, $u(0) = 0$, with exact solution $u(t) = t$. Since $f(u,t) = 1$, the linear multistep formula (1.3.4) reduces to $\rho(Z)v^n = k\sigma(1)$. Now since $\rho(Z)$ is a polynomial of degree s with $\rho(1) = 0$, it has a finite Taylor expansion

$$\rho(Z) = \rho'(1)(Z-1) + \frac{1}{2}\rho''(1)(Z-1)^2 + \cdots + \frac{1}{s!}\rho^{(s)}(1)(Z-1)^s.$$

Let us apply $\rho(Z)$ to the particular function $V^n = kn\sigma(1)/\rho'(1) = t_n\sigma(1)/\rho'(1)$. Since V^n is linear in n, all but the first term in the series are 0, and we get $\rho(Z)V^n = \rho'(1)(Z-1)V^n$, which reduces to $k\sigma(1)$. In other words, V^n is a solution to the linear multistep formula if the prescribed initial values are $v^n = V^n$ for $0 \le n \le s-1$. Obviously this solution satisfies condition (1.6.1) for the given initial data $u(0) = 0$. For the linear multistep formula to be convergent, it follows that V^n must satisfy (1.6.2), and thus we must have $\sigma(1) = \rho'(1)$, as claimed.

(c) Consistent + stable $\Rightarrow$ convergent. [This part of the proof is not yet written; see the references, such as pp. 342–345 of Hairer, Nørsett, & Wanner (1987). Following work of

Butcher (1966) and Skeel (1976), it is now standard to carry out the proof by first reducing the linear multistep formula to a one-step recursion, as in (1.5.8). The underlying idea is *backward error analysis*—i.e., one shows that the numerical method for a fixed time step computes the exact solution to a problem with a slightly perturbed function $f(u,t)$.] ∎

Besides convergence, it is desirable to know something about the accuracy of solutions computed with linear multistep formulas. To ensure pth-order accuracy, condition (1.6.1) may be strengthened as follows:

$$\|v^n - u(t_n)\| = O(k^p) \quad \text{as } k \to 0, \qquad 0 \le n \le s-1. \tag{1.6.4}$$

The following theorem confirms that for a stable linear multistep formula applied to an initial-value problem with sufficiently smooth coefficients, the local errors add up as expected:

GLOBAL p^{TH}-ORDER ACCURACY

Theorem 1.11. *Consider an initial-value problem (1.1.2) that satisfies the conditions of Theorem 1.1 on an interval $[0,T]$, and in addition, assume that $f(u,t)$ is p times continuously differentiable with respect to u and t. Let an approximate solution be computed by a convergent linear multistep formula of order of accuracy $\ge p$ with starting values satisfying (1.6.4). Then this solution satisfies*

$$\|v(t) - u(t)\| = O(k^p) \qquad \text{as } k \to 0, \tag{1.6.5}$$

uniformly for all $t \in [0,T]$.

EXERCISES

▷ *1.6.1.* Which of the following linear multistep formulas are convergent? Are the nonconvergent ones inconsistent, or unstable, or both?

(a) $v^{n+2} = \frac{1}{2}v^{n+1} + \frac{1}{2}v^n + 2kf^{n+1}$,

(b) $v^{n+1} = v^n$,

(c) $v^{n+4} = v^n + \frac{4}{3}k(f^{n+3} + f^{n+2} + f^{n+1})$,

(d) $v^{n+3} = v^{n+1} + \frac{1}{3}k(7f^{n+2} - 2f^{n+1} + f^n)$,

(e) $v^{n+4} = \frac{8}{19}(v^{n+3} - v^{n+1}) + v^n + \frac{6}{19}k(f^{n+4} + 4f^{n+3} + 4f^{n+1} + f^n)$,

(f) $v^{n+3} = -v^{n+2} + v^{n+1} + v^n + 2k(f^{n+2} + f^{n+1})$.

▷ *1.6.2. Borderline cases.*

(a) Consider the unstable extrapolation formula (1.5.1). What is the order of magnitude (power of k) of the errors introduced locally at each step? What is the order of magnitude factor by which these are amplified after $\Theta(k^{-1})$ time steps? Verify that this factor is large enough to cause nonconvergence.

(b) What are the corresponding numbers for the s-step generalization with $\rho(z) = (z-1)^s$, $\sigma(z) = 0$?

(c) On the other hand, devise another unstable linear multistep formula in which the local discretization errors are of a high enough order of accuracy relative to the unstable amplification factors that they will *not* cause nonconvergence.

(d) Why does the example of *(c)* not contradict Theorem 1.10? Why is it appropriate to consider this example unstable?

▷ *1.6.3. Convergence for scalar linear initial-value problems.* Prove as briefly and elegantly as you can that stability and consistency imply convergence, for the special case in which u is scalar and the function $f(u,t)$ is scalar and linear.

1.7. Stability regions

The results of the last two sections concerned the behavior of linear multistep formulas in the limit $k \to 0$. But for the important class of stiff ODEs, which involve widely varying time scales, it is impractical to take k small enough for those results to apply. Analysis of behavior for finite k becomes indispensable. In Chapter 4 we shall see that consideration of finite k is also essential for the analysis of discretizations of partial differential equations.

Stiffness is a subtle idea, and our discussion of it will be deferred to the next section. Here, we shall simply consider the problem of finite k analysis of linear multistep formulas. The key idea that emerges is the idea of a *stability region.*

EXAMPLE 1.7.1. Let us begin by looking again at the unstable third-order formula (1.3.21). In Example 1.5.1 we applied this formula to the initial-value problem $u_t = u$, $u(0) = 1$ with time step $k = 0.025$ and observed the oscillatory instability shown in Figure 1.5.1(b). From one time step to the next, the solution grew by a factor about -4.93, and to explain this we looked at the recurrence relation

$$\rho(Z)v^n = v^{n+2} + 4v^{n+1} - 5v^n = 0. \tag{1.7.1}$$

The zeros of $\rho(z)$ are 1 and -5, corresponding to solutions $v^n = 1$ and $v^n = (-5)^n$ to (1.7.1), and the second of these solutions explains the factor -4.93 at least approximately.

For this scalar, linear example, we can do much better by retaining the terms f^{n+j} in the analysis. The function f is simply $f(u) = u$, and thus an exact rather than approximate model of the calculation is the recurrence relation

$$\rho(Z)v^n - k\sigma(Z)f^n = (\rho(Z) - k\sigma(Z))v^n = 0, \tag{1.7.2}$$

that is,

$$(Z^2 + (4-4k)Z - (5+2k))v^n = v^{n+2} + (4-4k)v^{n+1} - (5+2k)v^n = 0.$$

Setting $k = 0.025$ gives the zeros

$$z_1 \approx 1.025315, \qquad z_2 \approx -4.925315.$$

And now we have a virtually exact explanation of the ratio ≈ -4.93 of Exercise 1.5.1—accurate, as it happens, to about 20 digits of accuracy (see Exercise 1.7.5).

For an arbitrary ODE, f is not just a multiple of u, and finite k analysis is not so simple as in the example just considered. We want to take the terms $\beta_j f^{n+j}$ into consideration, but we certainly don't want to carry out a separate analysis for each function f. Instead, let us assume that $f(u,t) = au$ for some constant $a \in \mathbb{C}$. In other words, we consider the linear *model equation*

$$u_t = au. \tag{1.7.3}$$

In the next section we shall see that a nonlinear system of ODEs can be reduced to a set of problems of the form (1.7.3) by linearization, freezing of coefficients, and diagonalization.

If a linear multistep formula (1.2.11) is applied to the model equation (1.7.3), it reduces to the recurrence relation

$$\sum_{j=0}^{s} \alpha_j v^{n+j} - \bar{k} \sum_{j=0}^{s} \beta_j v^{n+j} = 0,$$

or equivalently

$$\Big[\rho(Z) - \bar{k}\sigma(Z)\Big] v^n = 0, \tag{1.7.4}$$

where we have defined

$$\bar{k} = ak. \tag{1.7.5}$$

Now let $\pi_{\bar{k}}(z)$ be the **stability polynomial**

$$\pi_{\bar{k}}(z) = \rho(z) - \bar{k}\sigma(z) = \sum_{j=0}^{s} (\alpha_j - \bar{k}\beta_j) z^j, \tag{1.7.6}$$

whose coefficients depend on the parameter $\bar{k}$. Then the solutions to (1.7.4) are related to the zeros of $\pi_{\bar{k}}$ exactly as the solutions to (1.5.5) were related to the zeros of $\rho(z)$. In analogy to the developments of §1.5, we define:

A linear multistep formula is **absolutely stable*** *for a particular value $\bar{k} = ak$ if all solutions $\{v^n\}$ of the recurrence relation (1.7.4) are bounded as $n \to \infty$.*

Just as in Theorem 1.7, it is easy to characterize those linear multistep formulas that are absolutely stable:

ROOT CONDITION FOR ABSOLUTE STABILITY

Theorem 1.12. *A linear multistep formula is absolutely stable for a particular value $\bar{k} = ak$ if and only if all the zeros of $\pi_{\bar{k}}(z)$ satisfy $|z| \le 1$, and any zero with $|z| = 1$ is simple.*

What is different here from what we did in §1.5 is that everything depends on $\bar{k}$. For some $\bar{k}$ a linear multistep formula will be absolutely stable, and for others it will be absolutely unstable. Here now is the key definition:

* Other terms are "weakly stable" and "time-stable."

The **stability region** *S of a linear multistep formula is the set of all $\bar{k} \in \mathbb{C}$ for which the formula is absolutely stable.*

Note that according to this definition, a linear multistep formula is stable if and only if the point 0 belongs to its stability region.

Let us now derive the four most familiar examples of stability regions, illustrated in Figure 1.7.1.

EXAMPLE 1.7.2. For the Euler formula (1.2.3), (1.7.6) becomes

$$\pi_{\bar{k}}(z) = (z-1)-\bar{k} = z-(1+\bar{k}), \tag{1.7.7}$$

with zero $1+\bar{k}$. Therefore S is the set of $\bar{k} \in \mathbb{C}$ with $|1+\bar{k}| \le 1$, that is, the disk $|\bar{k}-(-1)| \le 1$. Figure 1.7.1a plots this region.

EXAMPLE 1.7.3. For the backward Euler formula (1.2.4), the stability polynomial is

$$\pi_{\bar{k}}(z) = (z-1)-\bar{k}z = (1-\bar{k})z-1, \tag{1.7.8}$$

with zero $(1-\bar{k})^{-1}$. S is the set of $\bar{k} \in \mathbb{C}$ with $|1-\bar{k}|^{-1} \le 1$, that is, the exterior of the disk $|\bar{k}-1| \ge 1$. See Figure 1.7.1b.

EXAMPLE 1.7.4. For the trapezoid formula (1.2.5), we have

$$\pi_{\bar{k}}(z) = (z-1)-\tfrac{1}{2}\bar{k}(z+1) = (1-\tfrac{1}{2}\bar{k})z-(1+\tfrac{1}{2}\bar{k}) \tag{1.7.9}$$

with zero $(1+\frac{1}{2}\bar{k})/(1-\frac{1}{2}\bar{k}) = (2+\bar{k})/(2-\bar{k})$. S is the set of points in $\mathbb{C}$ that are no farther from -2 than from 2—i.e., $\operatorname{Re} z \le 0$, the left half-plane. See Figure 1.7.1c.

EXAMPLE 1.7.5. The midpoint formula (1.2.6) has

$$\pi_{\bar{k}}(z) = z^2-2\bar{k}z-1, \tag{1.7.10}$$

which has two zeros z satisfying

$$z-\frac{1}{z} = 2\bar{k}. \tag{1.7.11}$$

Obviously there is always one zero with $|z_1| \le 1$ and another with $|z_2| \ge 1$, so for absolute stability, we must have $|z_1| = |z_2| = 1$ —both zeros on the unit circle, which will occur if and only if $\bar{k}$ lies in the closed complex interval $[-i,i]$. The two extreme values $\bar{k} = \pm i$ give double zeros $z_1 = z_2 = \pm i$, so we conclude that S is the open complex interval $(-i,i)$, as shown in Figure 1.7.1d.

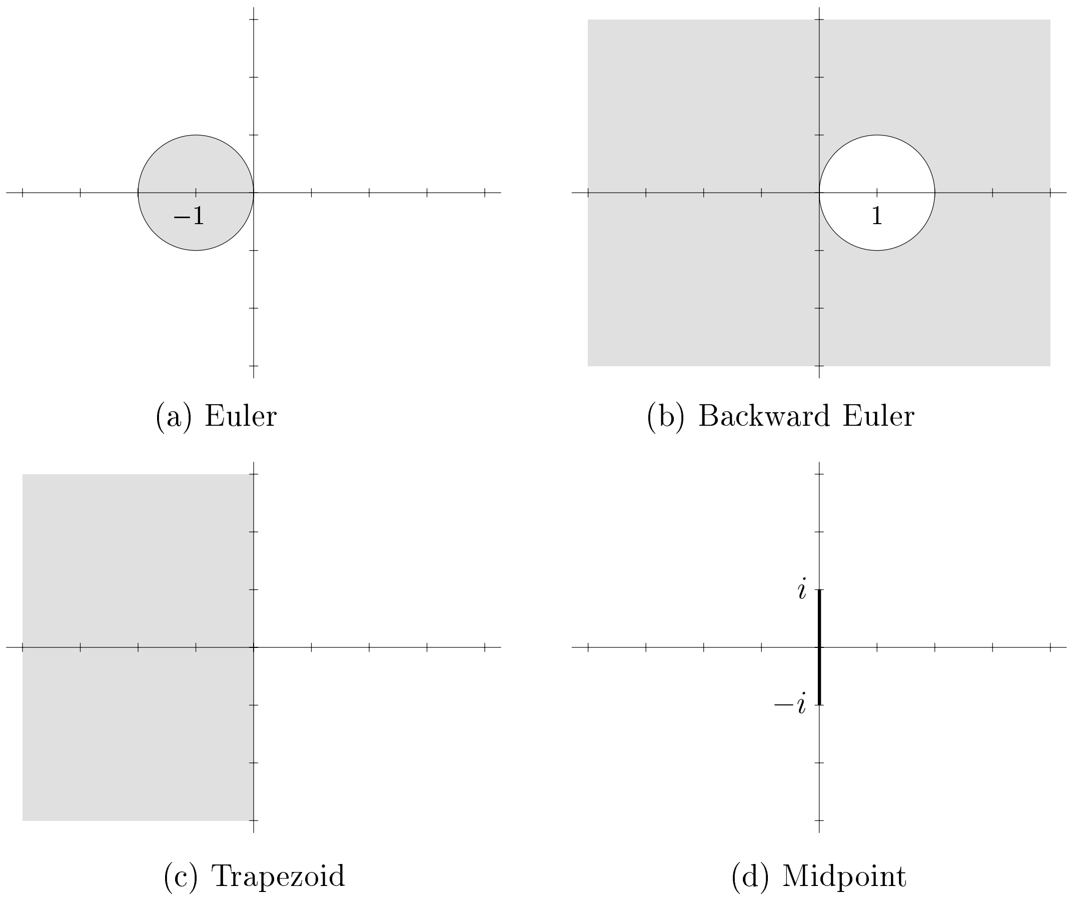


Figure 1.7.1. Stability regions (shaded) for four linear multistep formulas. In case (d) the stability region is the open complex interval $(-i, i)$.

As mentioned on p. 17, the four linear multistep formulas just considered are the bases of four important classes of finite difference formulas for partial differential equations: Euler, backward Euler, trapezoid, and leap frog. We shall refer often to Figure 1.7.1 in examining the stability properties of these finite difference formulas.

The boundaries of the stability regions for various Adams-Bashforth, Adams-Moulton, and backwards differentiation formulas are plotted in Figures 1.7.2–1.7.4. Note that the scales in these three figures are very different—the Adams-Moulton stability regions are much larger than those for Adams-Bashforth. Note also that for all of the higher-order Adams formulas, the stability regions are bounded, whereas for the backwards differentiation formulas they are unbounded. As will be explained in the next section, this is why backwards differentiation formulas are important.

How does one compute stability regions like these? One approach would be to calculate the zeros of $\pi_{\bar{k}}$ for a large number of values of $\bar{k} \in \mathbb{C}$ and draw a contour plot, but there is a simpler and more accurate method. By (1.7.6), if z is a zero of $\pi_{\bar{k}}(z)$ for some $\bar{k}$ and $\sigma(z) \neq 0$, then

$$\bar{k} = \frac{\rho(z)}{\sigma(z)} \tag{1.7.12}$$

(cf. (1.3.19)). To determine the boundary of the stability region, first calculate the curve of values $\bar{k}$ corresponding to $z = e^{i\theta}$ with $\theta \in [0, 2\pi]$. This **root locus curve** has the property that at each point $\bar{k}$ on the curve, one zero of $\pi_{\bar{k}}$ just touches the unit circle. It follows that the boundary of S is a subset of the root locus curve—only a subset, in general, because the other zeros of $\pi_{\bar{k}}$ might lie either in the disk or outside. By the principle of the argument of complex analysis, one can determine which components are which by checking just one value $\bar{k}$ in each loop enclosed by the root locus curve. See Exercise 1.7.3.

EXERCISES

▷ *1.7.1.* Prove that for the unstable linear multistep formula (1.3.21), the stability region S is the empty set.

▷ *1.7.2.* Find exact formulas for the boundaries of S for *(a)* the second-order Adams-Bashforth formula, *(b)* the third-order backwards differentiation formula.

▶ *1.7.3.* Write a computer program to plot root locus curves. In a high-level language like Matlab, it is most convenient to define a variable $\nabla = 1 - z^{-1}$, where z is a vector of 200 or so points on the unit circle, and then work directly with formulas such as (1.4.10), (1.4.13), (1.4.16) rather than with the coefficients α_j and β_j.

(a) Reproduce Figure 1.7.2, and then generate the root locus curve for $p = 4$. What is S? (Be careful.)

(b) Reproduce Figure 1.7.4, and then generate the root locus curve for $p = 7$. What is S?

(c) Plot the root locus curve for (1.3.21). Label each component by the number of zeros of $\pi_{\bar{k}}$ outside the unit disk, and explain how this pictures relates to Exercise 1.7.1.

▷ *1.7.4.* What is the maximum time step k for which the third-order Adams-Bashforth formula is absolutely stable when applied to *(i)* $u_t = -u$, *(ii)* $u_t = iu$?

(a) First, estimate the time step limits with the aid of Figure 1.7.2 and a ruler.

(b) Now derive the exact answers. For *(ii)* this is hard, but it can be done.

▷ *1.7.5.* Explain the remark about 20-digit accuracy at the end of Example 1.7.1. Approximately where does the figure of 20 digits come from?

▷ *1.7.6.* *True or False:* the stability region for any linear multistep formula is a closed subset of the complex plane.

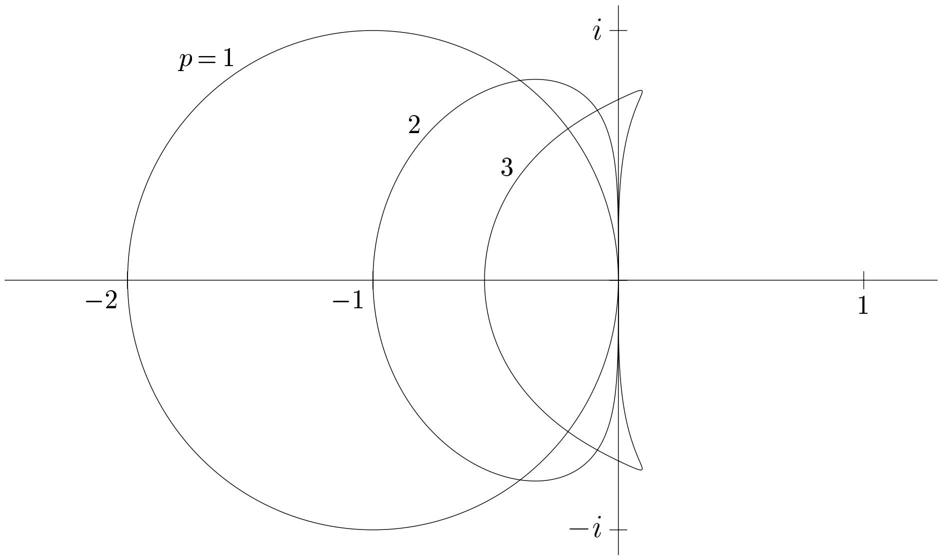


Figure 1.7.2. Boundaries of stability regions for Adams-Bashforth formulas of orders 1–3.

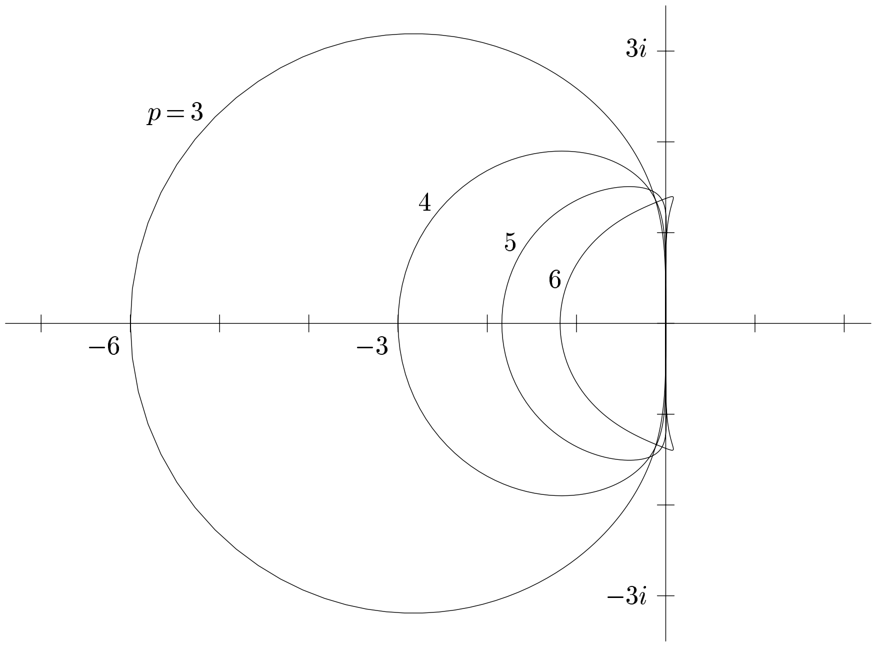


Figure 1.7.3. Boundaries of stability regions for Adams-Moulton formulas of orders 3–6. (Orders 1 and 2 were displayed already in Figure 1.7.1(b,c).) Note that the scale is very different from that of the previous figure.

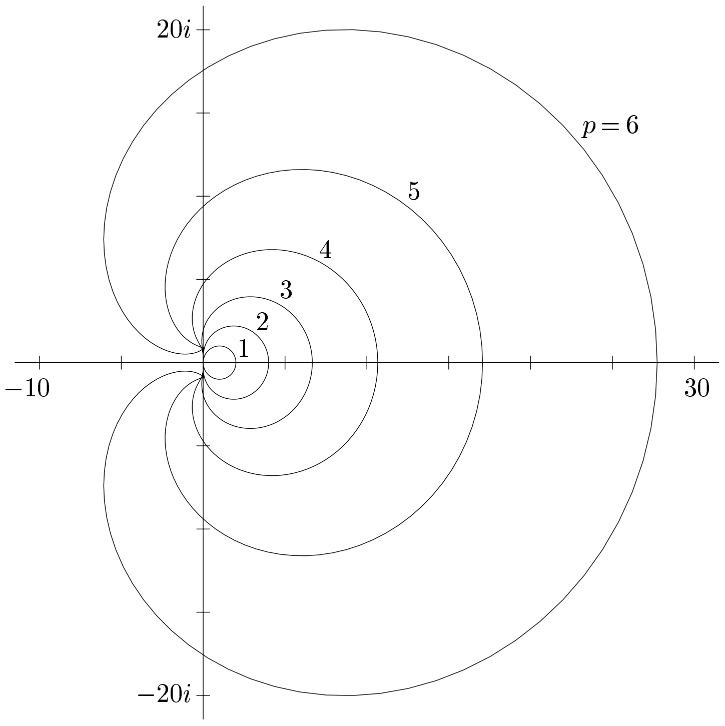


Figure 1.7.4. Boundaries of stability regions for backwards differentiation formulas of orders 1–6 (exteriors of curves shown).

▷ *1.7.7. Simpson's formula.* Determine the stability region for Simpson's formula (1.4.17), and draw a sketch.

▷ *1.7.8.* Prove that any convergent linear multistep formula is absolutely unstable for all sufficiently small positive $\bar{k} > 0$.

1.8. Stiffness

In 1962 Henrici's *Discrete Variable Methods in Ordinary Differential Equations* appeared. This landmark book, presenting the material of the previous sections in great detail, made the "Dahlquist theory" of linear multistep formulas widely known. With the availability of formulas of arbitrarily high order of accuracy and a general convergence theory to prove that they worked, it may have seemed that little was left to do except turn theory into software.

As ODE computations became commonplace in the 1960s, however, it became clear that certain problems were handled badly by the usual methods. Unreasonably small time steps would be required to achieve the desired accuracy. What was missing in the standard theory was the notion of *stiffness.* As it happens, the missing piece had been supplied a decade earlier in an eight-page paper by a pair of chemists at the University of Wisconsin, C.F. Curtiss and J.O. Hirschfelder (*Proc. Nat. Acad. Sci. 38,* 1952). Curtiss and Hirschfelder had identified the phenomenon of stiffness, coined the term, and invented backwards differentiation formulas to cope with it. However, their paper received little attention for a number of years, and for example, it does not appear among Henrici's three hundred references.

What is a stiff ODE? The following are the symptoms most often mentioned:

1. *The problem contains widely varying time scales.*
2. *Stability is more of a constraint on k than accuracy.*
3. *Explicit methods don't work.*[*]

Each of these statements has been used as a characterization of stiffness by one author or another. In fact, they are all correct, and they are not independent statements but part of a coherent whole. After presenting an example, we shall make them more precise and explain the logical connections between them.

EXAMPLE 1.8.1. The linear initial-value problem

$$u_t = -100(u - \cos(t)) - \sin(t), \qquad u(0) = 1 \tag{1.8.1}$$

has the unique solution $u(t) = \cos(t)$, for if $u(t) = \cos(t)$ the first term on the right becomes 0, so that the large coefficient -100 drops out of the equation. That coefficient has a dominant effect on *nearby* solutions of the ODE corresponding to different initial data, however, as illustrated in Figure 1.8.1. A typical trajectory $u(t)$ of a solution to this ODE begins by shooting rapidly towards the curve $\cos(t)$ on a time scale ≈ 0.01. This is the hallmark of stiffness: rapidly changing components that are present in an ODE even when they are absent from the solution being tracked.

[*] This last item is quoted from p. 2 of the book by Hairer and Wanner (1991). For all kinds of information on numerical methods for stiff ODEs, including historical perspectives and lighthearted humor, that is the book to turn to. Another earlier reference worth noting is L. F. Shampine and C. W. Gear, "A user's view of solving stiff ordinary differential equations," *SIAM Review 21* (1979), 1–17.

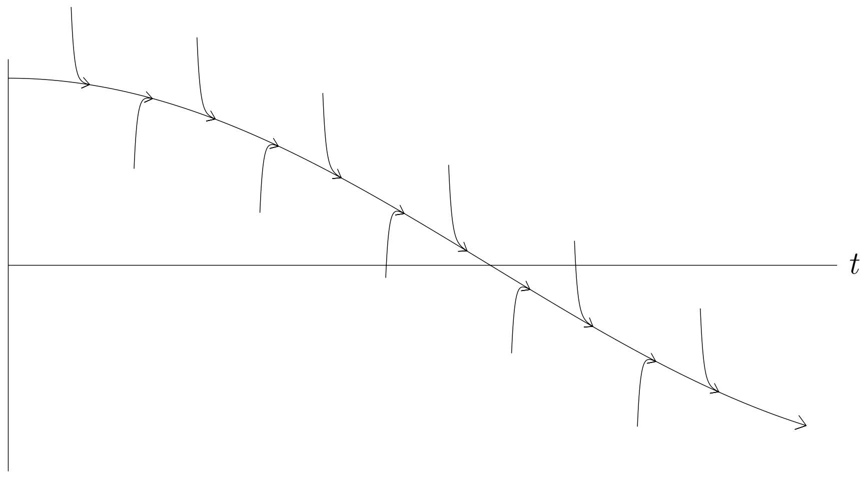


Figure 1.8.1. $u(t) = \cos(t)$ and nearby solutions of the initial-value problem (1.8.1).

k	AB2	BD2	
0.2	14.40	0.5404	
0.1	-5.70×10^4	0.54033	
0.05	-1.91×10^9	0.540309	
0.02	-5.77×10^{10}	0.5403034	
0.01	0.54030196	0.54030258	
0.005	0.54030222	0.54030238	
$\vdots$	$\vdots$	$\vdots$	
0	0.540302306	0.540302306	$= \cos(1)$

Table 1.8.1. Computed values $v(1)$ for the same problem.

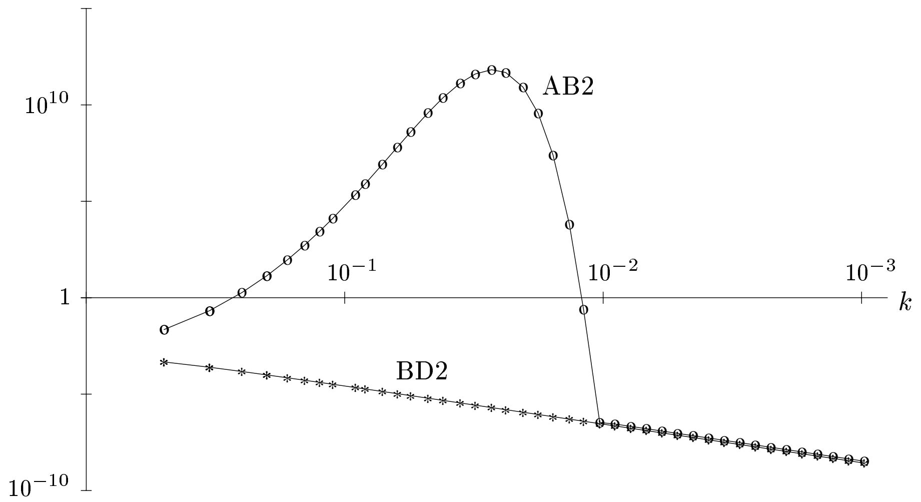


Figure 1.8.2. Computed errors $|v(1) - \cos(1)|$ as a function of step size k for the same problem (log-log scale).

Table 1.8.1 indicates the curious effect that this property of the ODE has on numerical computations. For six values of k, the table compares the results at $t=1$ computed by the second-order Adams-Bashforth and backwards differentiation formulas. (Since the ODE is linear, implementing the backwards differentiation formula is easy.) The difference between the two columns of numbers is striking. The backwards differentiation formula behaves beautifully, converging smoothly and quadratically to the correct answer, but the Adams-Bashforth formula generates enormous and completely erroneous numbers for moderate k. Yet when k becomes small enough it settles down to be just as accurate as backwards differentiation.

This behavior is shown graphically in Figure 1.8.2, which is a log-log plot of the error $|v(1)-\cos(1)|$ as a function of k. The Adams-Bashforth formula is obviously useless for $k>0.01$. What if we only want two or three digits of accuracy? With the Adams-Bashforth formula, that request cannot be satisfied; seven digits is the minimum.

Since the example just considered is linear, one can analyze what went wrong with the Adams-Bashforth formula exactly. In the notation of the last section, the trouble is that there is a root of the recurrence relation $\pi_{\bar{k}}(Z)v^n=0$ that lies outside the unit disk. We make the argument precise as follows.

EXAMPLE 1.8.1, CONTINUED. If $u(t)$ is any solution to $u_t=-100(u-\cos(t))-\sin(t)$, then $(\delta u)(t)=u(t)-\cos(t)$ satisfies the equation

$$(\delta u)_t = -100(\delta u). \tag{1.8.2}$$

This is a linear, scalar, constant-coefficient ODE of the form (1.7.3) of the last section, with $a=-100$. If we apply the 2nd-order Adams-Bashforth formula to it, we get the recurrence relation

$$v^{n+2}-v^{n+1} = -100k(\tfrac{3}{2}v^{n+1}-\tfrac{1}{2}v^n),$$

that is,

$$v^{n+2}+(150k-1)v^{n+1}-50kv^n = 0,$$

with characteristic polynomial

$$\pi_{\bar{k}}(z) = z^2+(150k-1)z-50k = z^2+(\tfrac{3}{2}\bar{k}-1)z-\tfrac{1}{2}\bar{k}.$$

For $k>0.01$, one of the two roots of this polynomial lies in the negative real interval $(-\infty,-1)$, whereas for $k\le 0.01$ that root crosses into the interval $[-1,0)$. This is why $k=0.01$ is the critical value for this problem, as is so strikingly evident in Figure 1.8.2.

Figure 1.8.3 shows how this analysis relates to the stability region of the second-order Adams-Bashforth formula. The shaded region in the figure is the stability region (see Figure 1.7.2), and the crosses mark the quantities $\bar{k}=-100k$ corresponding to the values k (except $k=0.2$) in Table 1.8.1. When k becomes as small as 0.01, i.e., $\bar{k}\ge -1$, the crosses move into the stability region and the computation is successful.

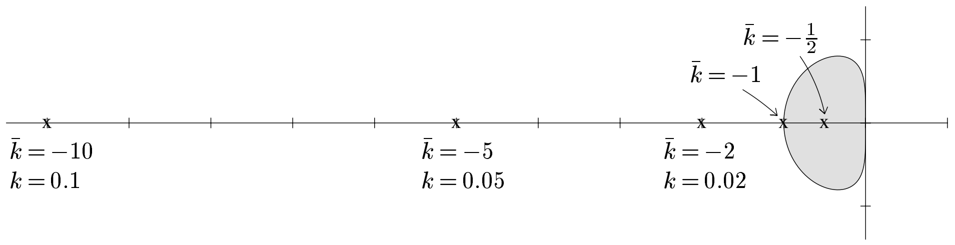


Figure 1.8.3. The stability region for the 2nd-order Adams-Bashforth formula, with crosses marking the values $\bar{k} = -100k$ from Table 1.8.1.

With this example in mind, let us turn to a more careful discussion of statements 1–3 on p. 65. The following summary describes what it means to say that the ODE $u_t = f(u,t)$ is stiff with respect to the solution $u^*(t)$ for times $t \approx t^*$.

1. Widely varying time scales. Stiffness arises when the time scale that characterizes the evolution of $u^*(t)$ for $t \approx t^*$ is much slower than the time scale that characterizes the evolution of small perturbations $(\delta u)(t)$ on that solution for $t \approx t^*$.

2. Stability is more of a constraint than accuracy. If widely varying time scales in this sense are present, and if the discrete ODE formula has a bounded stability region, then there is a mismatch between the relatively large time steps that are adequate to resolve $u^*(t)$ and the relatively small time steps that are necessary to prevent unstable growth of small perturbations $(\delta u)(t)$. A successful computation will necessitate the use of these small time steps.

3. Explicit methods don't work. In particular, since explicit methods always have bounded stability regions (Exercise 1.8.1), the solution of stiff ODEs by explicit methods necessitates the use of small time steps. For better results one must usually turn to implicit formulas with unbounded stability regions.

If the real world supplied ODEs to be solved "at random," then widely varying time scales might not be a common concern. In actuality, the applications that the real world presents us with are often exceedingly stiff. One example is the field of chemical kinetics. Here the ordinary differential equations describe reactions of various chemical species to form other species, and the stiffness is a consequence of the fact that different reactions take place on vastly different time scales. Time scale ratios of 10^6 or more are common. Two other fields in which stiff ODEs arise frequently are control theory, where controlling forces tend to be turned on and off again suddenly, and circuit simulation, since different circuit components tend to have widely differing natural frequencies.* A fourth important class of stiff ordinary differential equations are the systems of ODEs that arise after discretizing time-dependent partial differential equations, particularly parabolic problems, with respect to their space variables—the "method of lines." This example will be discussed at length in Chapter 4.

To be able to determine whether a particular problem is stiff, we need tools for quantifying the notion of the "time scales" present in an ODE. The way this is generally done is by reducing an arbitrary system of ODEs to a collection of scalar, linear model problems of the form (1.7.3), to which the idea of stability regions will be applicable. This is achieved by the following sequence of three steps, a sequence that has been familiar to mathematical scientists for a century:

$$u_t = f(u,t) \tag{1.8.3}$$

$$\downarrow \quad \textit{LINEARIZE}$$

$$u_t = J(t)u \tag{1.8.4}$$

$$\downarrow \quad \textit{FREEZE COEFFICIENTS}$$

$$u_t = Au \tag{1.8.5}$$

$$\downarrow \quad \textit{DIAGONALIZE}$$

$$\{u_t = \lambda u\}, \quad \lambda \in \Lambda(A). \tag{1.8.6}$$

In (1.8.6), $\Lambda(A)$ denotes the spectrum of A, i.e., the set of eigenvalues.

We begin with (1.8.3), a system of N first-order ODEs, with $u^*(t)$ denoting the particular solution that we are interested in. If we make the substitution

$$u(t) = u^*(t) + (\delta u)(t),$$

as in (1.8.2), then stability and stiffness depend on the evolution of $(\delta u)(t)$.

The first step is to *linearize* the equation by assuming δu is small. If f is differentiable with respect to each component of u, then we have

$$f(u,t) \;=\; f(u^*,t) + J(t)(\delta u)(t) + o(\|\delta u\|),$$

where $J(t)$ is the $N \times N$ **Jacobian matrix** of partial derivatives of f with respect to u:

$$[J(t)]_{ij} = \frac{\partial f^{(i)}}{\partial u^{(j)}}(u^*(t),t), \qquad 1 \le i,j \le N.$$

*See A. Sangiovanni-Vincentelli and J. K. White, *Relaxation Techniques for the Solution of VLSI Circuits,* Kluwer, 1987.

This means that if δu is small, the ODE can be accurately approximated by a linear problem:

$$u_t = f(u^*,t) + J(t)(\delta u).$$

If we subtract the identity $u_t^* = f(u^*,t)$ from this equation, we get

$$(\delta u)_t = J(t)(\delta u).$$

One can think of this result as approximate, if δu is small, or exact, if δu is infinitesimal. Rewriting δu as a new variable u gives (1.8.4).

The second step is to *freeze coefficients* by setting

$$A = J(t^*)$$

for the particular value t^* of interest. The idea here is that stability and stiffness are local phenomena, which may appear at some times t^* and not others. The result is the constant-coefficient linear problem (1.8.5).

Finally, assuming A is diagonalizable, we *diagonalize* it. In general, any system of N linear ordinary differential equations with a constant, diagonalizable coefficient matrix A is exactly equivalent to N scalar equations (1.7.3) with values a equal to the eigenvalues of A. To exhibit this equivalence, let V be an $N \times N$ matrix such that $V^{-1}AV$ is diagonal. (The columns of V are eigenvectors of A.) Then $u_t = Au$ can be rewritten as $u_t = V(V^{-1}AV)V^{-1}u$, i.e., $(V^{-1}u)_t = (V^{-1}AV)(V^{-1}u)$, a diagonal system of equations in the variables $V^{-1}u$. This diagonal system of ODEs can be interpreted as a representation of the original system (1.8.5) in the basis of eigenvectors of A defined by the columns of V. And thus, since a diagonal system of ODEs is the same as a collection of independent scalar ODEs, we end up with the N problems (1.8.6).

Having reduced (1.8.3) to a collection of N scalar, linear, constant-coefficient model problems, we now estimate stability and stiffness as follows:

RULE OF THUMB. For a successful computation of $u^*(t)$ for $t \approx t^*$, k must be small enough that for each eigenvalue λ of the Jacobian matrix $J(t^*)$, $\bar{k} = k\lambda$ lies inside the stability region S (or at least within a distance $O(k)$ of S).

This Rule of Thumb is not a theorem; exceptions can be found.* But if it is violated, some numerical mode will very likely blow up and obliterate the correct solution. Regarding the figure $O(k)$, see Exercise 1.8.4.

*It is interesting to consider what keeps the Rule of Thumb from being a rigorous statement. The various gaps in the argument can be associated with the three reductions (1.8.3)→(1.8.6). The step (1.8.3)→(1.8.4) may fail if the actual discretization errors or rounding errors present in the problem are large enough that they cannot be considered infinitesimal, i.e., the behavior of δu is nonlinear.

A stiff problem is one for which satisfying the conditions of the Rule of Thumb is more of a headache than resolving the underlying solution $u^*(t)$.

Here are some examples to illustrate the reduction (1.8.3)→(1.8.6).

EXAMPLE 1.8.2. If the equation in Example 1.8.1 had been

$$u_t = -100\sin(u - \cos(t)) - \sin(t),$$

then linearization would have been called for. Setting $(\delta u)(t) = u(t) - \cos(t)$ gives

$$(\delta u)_t = -100\sin(\delta u),$$

and if $u(t) \approx \cos(t)$, then $(\delta u)(t)$ is small, and the linear model becomes (1.8.2) again.

EXAMPLE 1.8.3. For the initial-value problem

$$u_t = -100u^2(t)\sin(u - \cos(t)) - \sin(t),$$

we end up with a model equation with a time-dependent coefficient:

$$(\delta u)_t = -100\cos^2(t)(\delta u).$$

These are scalar examples. However, the usual reason that an ODE has widely varying time scales is that it is a *system* of differential equations. If the solution has N components, it is natural that some of them may evolve much more rapidly than others. Here are three examples.

EXAMPLE 1.8.5. Consider the 2×2 linear initial-value problem

$$\begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}_t = \begin{pmatrix} -1000 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}, \quad t\in[0,1], \qquad \begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{1.8.7}$$

Suppose we decide that an acceptable solution at $t=1$ is any vector with $|v^{(1)}(1) - e^{-1000}| < \epsilon$ and $|v^{(2)}(1) - e^{-1}| < \epsilon$ for some $\epsilon > 0$. If (1.8.7) is solved by a pth order linear multistep formula with time step k, the $u^{(1)}$ and $u^{(2)}$ components decouple completely, so the results

The step (1.8.4)→(1.8.5) may fail if the density of time steps is not large compared with the time scale on which $J(t)$ varies (D.J. Higham 1992). The finite size of the time steps also implies that if an eigenvalue drifts outside the stability region for a short period, the error growth it causes may be small enough not to matter. Finally, the step (1.8.5)→(1.8.6) may fail in the case where the matrix A is far from *normal,* i.e., its eigenvectors are far from orthogonal. In this case the matrix V is ill-conditioned and instabilities may arise even though the spectrum of A lies in the stability region; it becomes necessary to consider the *pseudospectra* of A as well (D.J. Higham and L.N.T., 1992). These effects of non-normality will be discussed in §§4.5–4.6, and turn out to be particularly important in the numerical solution of time-dependent PDEs by spectral methods, discussed in Chapter 8.

in each component will be those for the corresponding model equation (1.7.3). To obtain $v^{(2)}$ sufficiently accurately, we need $k = O(\epsilon^{1/p})$. But to obtain $v^{(1)}$ sufficiently accurately, if the formula has a stability region of finite size like the Euler formula, we need k to be on the order of 10^{-3}. Most likely this is a much tighter restriction.

EXAMPLE 1.8.6. The linear problem

$$\begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}_t = \begin{pmatrix} -5 & 6 \\ 4 & -5 \end{pmatrix} \begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}, \qquad \begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{1.8.8}$$

is only superficially less trivial. The eigenvalues of the matrix are approximately -9.90 and -0.10; if 6 and 4 are changed to 5.01 and 4.99 they become approximately -9.999990 and -0.000010. As a result this system will experience a constraint on k just like that of Example 1.8.5, or worse.

EXAMPLE 1.8.7. The nonlinear ODE

$$\begin{pmatrix} u^{(1)} \\ u^{(2)} \end{pmatrix}_t = \begin{pmatrix} -u^{(1)}u^{(2)} \\ \cos(u^{(1)}) - \exp(u^{(2)}) \end{pmatrix}$$

has Jacobian matrix

$$J = -\begin{pmatrix} u^{(2)} & u^{(1)} \\ \sin(u^{(1)}) & \exp(u^{(2)}) \end{pmatrix}.$$

Near a point t with $u^{(1)}(t) = 0$ and $u^{(2)}(t) \gg 1$, the matrix is diagonal with widely differing eigenvalues, and the behavior will probably be stiff.

In general, a system of ODEs $u_t = f(u,t)$ is likely to be stiff at a point $u = u^*$, $t = t^*$ if the eigenvalues of $J(u^*,t^*)$ differ greatly in magnitude, especially if the large eigenvalues have negative real parts so that the corresponding solution components tend to die away rapidly.

To solve stiff problems effectively, one needs discrete ODE formulas with large stability regions. The particular shape of the stability region required will depend on the problem, but as a rule, the most important property one would like is for the stability region to include a large part of the left half-plane. (Why the left half-plane? See Exercise 1.8.3.) The following definitions are standard:

A linear multistep formula is A-**stable** *if the stability region contains the entire left half-plane* $\operatorname{Re}\bar{k} < 0$. *It is* $A(\alpha)$-**stable**, $\alpha \in (0,\pi/2)$, *if the stability region contains the infinite sector* $|\arg \bar{k} - \pi| < \alpha$. *It is* $A(0)$-**stable** *if it is* $A(\alpha)$*-stable for some* $\alpha > 0$.

Roughly speaking, an A-stable formula will perform well on almost any stiff problem (and of course also on non-stiff problems, at some cost in extra work per time step). An $A(0)$-stable formula will perform well on stiff problems whose component modes exhibit exponential decay (negative real eigenvalues) but not oscillation (imaginary eigenvalues). Discretizations of parabolic and hyperbolic partial differential equations, respectively, will provide good examples of these two situations later in the book.

Figure 1.7.1 showed that both the backward Euler and trapezoid formulas are A-stable. This seems encouraging; perhaps to play it safe, we can simply use A-stable formulas all the time? But the following famous theorem shows that that policy is too limiting:

SECOND DAHLQUIST STABILITY BARRIER

Theorem 1.13. *The order of accuracy of an implicit A-stable linear multistep formula satisfies $p \le 2$. An explicit linear multistep formula cannot be A-stable.*

Proof. (Dahlquist, *BIT* 1963). *[Not yet written]* ∎

For serious computations, second-order accuracy is often not good enough. This is why backwards differentiation formulas are so important. For all of the higher-order Adams formulas, the stability regions are bounded, but the backwards differentiation formulas are $A(0)$-stable for $p \le 6$.* Most of the software in use today for stiff ordinary differential equations is based on these formulas, though Runge-Kutta methods are also contenders (see the next section). Because of Theorem 1.8, the usable backwards differentiation formulas are only those with $p \le 6$. In practice, some codes restrict attention to $p \le 5$.

Of course there are a dozen practical issues of computation for stiff ODEs that we have not touched upon here. For example, how does one solve the nonlinear equations at each time step? (The answer is invariably some form of Newton's method; see the references.) The state of software for stiff problems is highly advanced, as is also true for non-stiff problems. A non-stiff solver applied to a stiff problem will typically get the right answer, but it will take excessively small time steps. Many such codes incorporate devices to detect that stiffness is present and alert the user to that fact, or in some cases, to trigger an automatic change to a stiff solution method.

EXERCISES

▷ *1.8.1.* Prove that the stability region of any explicit linear multistep formula is bounded.

*For $p = 3, 4, 5, 6$ the values of α involved are approximately 86°, 73°, 52°, 18°. See Figure 1.7.4.

▷ *1.8.2.* Suppose Figure 1.8.2 had been based on the error $|v(2)-\cos(2)|$ instead of $|v(1)-\cos(1)|$. Qualitatively speaking, how would the plot have been different?

▷ *1.8.3.* What is special about the left half-plane that makes it appear in the definition of A-stability? Avoid an answer full of waffle; write two or three sentences that hit the nail on the head.

▷ *1.8.4.* The Rule of Thumb of p. 70 mentions the distance $O(k)$ from the stability region. Explain where this figure comes from, perhaps with the aid of an example. Why is it $O(k)$ rather than, say, $O(k^2)$ or $O(k^{1/2})$?

▷ *1.8.5.* Here is a second-order initial-value problem:

$$u_{tt}(t) = \cos(t\,u_t(t)) + (u(t))^2 + t, \qquad u(0) = 1, \qquad u_t(0) = 0.$$

(a) Rewrite it as a first-order initial-value problem of dimension 2.
(b) Write down the precise formula used to obtain v^{n+1} at each step if this initial-value problem is solved by the second-order Adams-Bashforth formula.
(c) Write down the 2×2 Jacobian matrix J for the system *(a)*.

▷ *1.8.6.* An astronomer decides to use a non-stiff ODE solver to predict the motion of Jupiter around the sun over a period of 10,000 years. Saturn must certainly be included in the calculation if high accuracy is desired, and Neptune and Uranus might as well be thrown in for good measure. Now what about Mars, Earth, Venus, and Mercury? Including these inner planets will improve the accuracy slightly, but it will increase the cost of the computation. Estimate as accurately as you can the ratio by which it will increase the computation time on a serial computer. (*Hint:* your favorite almanac or encyclopedia may come in handy.)

▷ *1.8.7.* A linear multistep formula is A-stable. What can you conclude about the roots of $\sigma(z)$?

▷ *1.8.8.* *Van der Pol oscillator.* The equation $\epsilon u_{tt} = -u + (1-u^2)u_t$, known as the Van der Pol equation, represents a simple harmonic oscillator to which has been added a nonlinear term that introduces positive damping for $|u| > 1$ and negative damping for $|u| < 1$. Solutions to this equation approach limit cycles of finite amplitude, and if ϵ is small, the oscillation is characterized by periods of slow change punctuated by short intervals during which $u(t)$ swings rapidly from positive to negative or back again.

(a) Reduce the Van der Pol equation to a system of first-order ODEs, and compute the Jacobian matrix of this system.

(b) What can you say about the stiffness of this problem?

▷ *1.8.9.* Given $\epsilon \ll 1$, suppose you solve an initial-value problem involving the linear equation $u_t = -Au + f(t)$, where A is a constant 3×3 matrix with eigenvalues 1, $\epsilon^{-1/2}$, and ϵ^{-1}, and $f(t)$ is a slowly-varying vector function of dimension 3. You solve this equation by a linear multistep program that is smart enough to vary the step size adaptively as it integrates, and your goal is to compute the solution $u(1)$ accurate to within an absolute error on the order of $\delta \ll 1$. How many time steps (order of magnitude functions of δ and ϵ) will be needed if the program uses *(a)* the second-order Adams-Moulton method, and *(b)* the third-order Adams-Moulton method? (*Hint:* be careful!)

▶ *1.8.10. A nonlinear stiff ODE. [To appear]*

1.9. Runge-Kutta methods

Linear multistep formulas represent one extreme—one function evaluation per time step—and Runge-Kutta methods represent the other. They are *one-step, multistage* methods, in which $f(u,t)$ may be evaluated at any number of *stages* in the process of getting from v^n to v^{n+1}, but those stages are intermediate evaluations that are never used again. As a result, Runge-Kutta methods are often more stable than linear multistep formulas, but at the price of more work per time step. They tend to be easier to implement than linear multistep formulas, since starting values are not required, but harder to analyze. These methods were first developed a century ago by Runge (1895), Heun (1900), and Kutta (1901).

In this book we will not discuss Runge-Kutta methods in any detail; we merely list a few of them, below, and state two theorems without proof. This is not because they are less important than linear multistep methods, but merely to keep the scale of the book manageable. Fortunately, the fundamental concepts of stability, accuracy and convergence are much the same for Runge-Kutta as for linear multistep formulas. The details are quite different, however, and quite fascinating, involving a remarkable blend of ideas of combinatorics and graph theory. For information, see the books by Butcher and by Hairer, Nørsett, and Wanner.

Runge-Kutta formulas tend to be "not very unique". If we define

$$s = \text{numbers of stages}, \qquad p = \text{order of accuracy},$$

then for most values of s there are infinitely many Runge-Kutta formulas with the maximal order of accuracy p. Here are some of the best-known examples [which should properly be presented in tableau form, but I haven't gotten around to that]:

"Modified Euler" or "improved polygon" formula $(s=p=2)$

$$\begin{aligned} a &:= kf(v^n, t_n), \\ b &:= kf(v^n + a/2, t_n + k/2), \\ v^{n+1} &:= v^n + b. \end{aligned} \tag{1.9.1}$$

"Improved Euler" or "Heun" formula $(s=p=2)$

$$\begin{aligned} a &:= kf(v^n, t_n), \\ b &:= kf(v^n + a, t_n + k), \\ v^{n+1} &:= v^n + \tfrac{1}{2}(a+b). \end{aligned} \tag{1.9.2}$$

"Heun's third-order formula" $(s=p=3)$

$$\begin{aligned} a &:= kf(v^n, t_n), \\ b &:= kf(v^n + a/3, t_n + k/3), \\ c &:= kf(v^n + 2b/3, t_n + 2k/3), \\ v^{n+1} &:= v^n + \tfrac{1}{4}(a+3c). \end{aligned} \tag{1.9.3}$$

"Fourth-order Runge-Kutta formula" $(s=p=4)$*

$$\begin{aligned} a &:= kf(v^n, t_n), \\ b &:= kf(v^n + a/2, t_n + k/2), \\ c &:= kf(v^n + b/2, t_n + k/2), \\ d &:= kf(v^n + c, t_n + k), \\ v^{n+1} &:= v^n + \tfrac{1}{6}(a + 2b + 2c + d). \end{aligned} \tag{1.9.4}$$

If f is independent of u, the first two of these formulas reduce to the midpoint and trapezoid formulas, respectively, and the fourth-order formula reduces to Simpson's rule. This last formula is sometimes called "the" Runge-Kutta formula, and is very widely known.

How accurate can a Runge-Kutta formula be? Here is what was known as of 1987:

ORDER OF ACCURACY OF EXPLICIT RUNGE-KUTTA FORMULAS

Theorem 1.14. *An explicit Runge-Kutta formula of order of accuracy p has the following minimum number of stages s:*

Order p	*Minimum number of stages s*
1,2,3,4	1,2,3,4 *(respectively)*
5	6
6	7
7	9
8	11
9	12–17 *(precise minimum unknown)*
10	13–17 *(precise minimum unknown)*

Proof. See Butcher (1987). ▮

Figure 1.9.1 shows the stability regions for the explicit Runge-Kutta formulas with $s = 1,2,3,4$. Since these formulas are not unique, the reader may wonder which choice the figure illustrates. As it turns out, it illustrates *all* the choices, for they all have the same stability regions, which are the level curves $|p(z)| = 1$ for the truncated Taylor series approximations to e^z. (This situation changes for $s \geq 5$.)

The classical Runge-Kutta formulas were explicit, but in recent decades implicit Runge-Kutta formulas have also been studied extensively. Unlike linear multistep formulas, Runge-Kutta formulas are not subject to an A-stability barrier. The following result should be compared with Theorems 1.9 and 1.13.

IMPLICIT RUNGE-KUTTA FORMULAS

Theorem 1.15. *An s-stage implicit Runge-Kutta formula has order of accuracy $p \leq 2s$. For each s, there exists an A-stable implicit Runge-Kutta formula with $p = 2s$.*

*already given in Exercise 1.3.4.

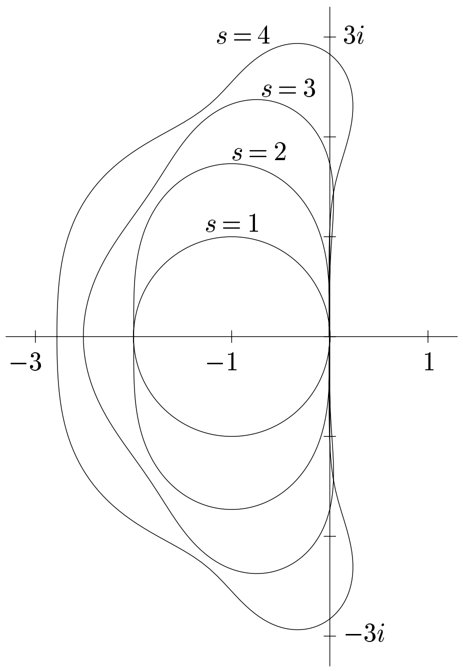


Figure 1.9.1. Boundaries of stability regions for Runge-Kutta formulas with $s=1,2,3,4$.

Theorem 1.15 looks like a panacea; why not use implicit Runge-Kutta methods all the time for stiff problems? The answer is that they lead to large systems of equations to be solved. For an initial-value problem involving N variables, an implicit linear multistep formula requires the solution of a system of N equations at each step, whereas in the Runge-Kutta case the dimension becomes sN. Since the work involved in solving a system of equations usually depends superlinearly on its size, it follows that implicit Runge-Kutta formulas tend to be more advantageous for small systems than for the larger ones that arise, for example, in discretizing partial differential equations. On the other hand, many special tricks have been devised for those problems, especially to take advantage of sparsity, so no judgment can be considered final. The book by Hairer and Wanner contains experimental comparisons of explicit and implicit multistep and Runge-Kutta codes, reaching the conclusion that they all work well.

1.10. Notes and References

This chapter has said little about the practical side of numerical solution of ordinary differential equations. One topic barely mentioned is the actual implementation of implicit methods—the solution of the associated equations at each time step by Newton's method and its variants. Another gap is that we have not discussed the idea of **adaptive step size and order control**, which, together with the development of special methods for stiff problems, are the most important developments in the numerical solution of ODEs during the computer era. The software presently available for solving ODEs is so powerful and reliable that there is little reason other than educational to write one's own program, except for very easy problems or specialized applications. Essentially all of this software makes use of adaptive step size control, and some of it controls the order of the formula adaptively too. Trustworthy codes can be found in the IMSL, NAG, and Harwell libraries, in Matlab, and in numerous other sources. The best-known programs are perhaps those developed at Sandia Laboratories and the Lawrence Livermore Laboratory, which can be obtained from the Netlib facility described in the Preface.

Although this book does not discuss boundary-value problems for ODEs, the reader should be aware that in that area too there is excellent adaptive software, which is far more efficient and reliable than the program a user is likely to construct by combining an initial-value problem solver with the idea of "shooting". Two well-known programs are PASVA and its various derivatives, which can be found in the NAG Library, and COLSYS, which has been published in the *Transactions on Mathematical Software* and is available from Netlib. A standard textbook on the subject is U. M. Ascher, R. M. M. Mattheij and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations,* Prentice Hall, 1988.

Some applications lead to systems of differential equations that must be solved in conjunction with additional non-differential conditions. Such systems are called **differential-algebraic equations (DAEs)** and have been investigated by Gear, Petzold, and many others. Software is available for these problems too, including a well-known program by Petzold known as DASSL.

The scale of ODEs that occur in practice can be enormous. One example of engineering interest arises in the simulation of VLSI circuits, where one encounters sparse stiff systems containing thousands or tens of thousands of variables. In such cases it is of paramount importance to exploit the special properties of the systems, and one special method that has been devised goes by the name of **waveform relaxation;** see the footnote on p. 69.