# The Initial Value Problem for Ordinary Differential Equations

In this chapter we begin a study of time-dependent differential equations, beginning with the initial value problem (IVP) for a time-dependent ordinary differential equation (ODE). Standard introductory texts are Ascher and Petzold [5], Lambert [59], [60], and Gear [33]. Henrici [45] gives the details on some theoretical issues, although stiff equations are not discussed. Butcher [12] and Hairer, Nørsett, and Wanner [43, 44] provide more recent surveys of the field.

The IVP takes the form

$$u'(t) = f(u(t), t) \quad \text{for } t > t_0 \tag{5.1}$$

with some initial data

$$u(t_0) = \eta. \tag{5.2}$$

We will often assume $t_0 = 0$ for simplicity.

In general, (5.1) may represent a system of ODEs, i.e., $u$ may be a vector with $s$ components $u_1, \ldots, u_s$, and then $f(u,t)$ also represents a vector with components $f_1(u,t), \ldots, f_s(u,t)$, each of which can be a nonlinear function of all the components of $u$.

We will consider only the first order equation (5.1), but in fact this is more general than it appears since we can reduce higher order equations to a system of first order equations.

**Example 5.1.** Consider the IVP for the ODE,

$$v'''(t) = v'(t)v(t) - 2t(v''(t))^2 \quad \text{for } t > 0.$$

This third order equation requires three initial conditions, typically specified as

$$\begin{aligned} v(0) &= \eta_1, \\ v'(0) &= \eta_2, \\ v''(0) &= \eta_3. \end{aligned} \tag{5.3}$$

We can rewrite this as a system of the form (5.1), (5.2) by introducing the variables

$$\begin{aligned} u_1(t) &= v(t), \\ u_2(t) &= v'(t), \\ u_3(t) &= v''(t). \end{aligned}$$

Then the equations take the form

$$\begin{aligned} u_1'(t) &= u_2(t), \\ u_2'(t) &= u_3(t), \\ u_3'(t) &= u_1(t)u_2(t) - 2tu_3^2(t), \end{aligned}$$

which defines the vector function $f(u,t)$. The initial condition is simply (5.2), where the three components of $\eta$ come from (5.3). More generally, any single equation of order $m$ can be reduced to $m$ first order equations by defining $u_j(t) = v^{(j-1)}(t)$, and an $m$th order system of $s$ equations can be reduced to a system of $ms$ first order equations.

See Section D.3.1 for an example of how this procedure can be used to determine the general solution of an $r$th order linear differential equation.

It is also sometimes useful to note that any explicit dependence of $f$ on $t$ can be eliminated by introducing a new variable that is simply equal to $t$. In the above example we could define

$$u_4(t) = t$$

so that

$$u_4'(t) = 1 \quad \text{and} \quad u_4(t_0) = t_0.$$

The system then takes the form

$$u'(t) = f(u(t)) \tag{5.4}$$

with

$$f(u) = \begin{bmatrix} u_2 \\ u_3 \\ u_1u_2 - 2u_4u_3^2 \\ 1 \end{bmatrix} \quad \text{and} \quad u(t_0) = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ t_0 \end{bmatrix}.$$

The equation (5.4) is said to be *autonomous* since it does not depend explicitly on time. It is often convenient to assume $f$ is of this form since it simplifies notation.

## 5.1 Linear ordinary differential equations

The system of ODEs (5.1) is *linear* if

$$f(u,t) = A(t)u + g(t), \tag{5.5}$$

where $A(t) \in \mathbb{R}^{s\times s}$ and $g(t) \in \mathbb{R}^s$. An important special case is the *constant coefficient linear system*

$$u'(t) = Au(t) + g(t). \tag{5.6}$$

where $A \in \mathbb{R}^{s\times s}$ is a constant matrix. If $g(t) \equiv 0$, then the equation is *homogeneous*. The solution to the homogeneous system $u' = Au$ with data (5.2) is

$$u(t) = e^{A(t-t_0)}\eta, \tag{5.7}$$

where the matrix exponential is defined as in Appendix D. In the scalar case we often use $\lambda$ in place of $A$.

### 5.1.1 Duhamel's principle

If $g(t)$ is not identically zero, then the solution to the constant coefficient system (5.6) can be written as

$$u(t) = e^{A(t-t_0)}\eta + \int_{t_0}^{t} e^{A(t-\tau)}g(\tau)\,d\tau. \tag{5.8}$$

This is known as *Duhamel's principle*. The matrix $e^{A(t-\tau)}$ is the solution operator for the homogeneous problem; it maps data at time $\tau$ to the solution at time $t$ when solving the homogeneous equation. Duhamel's principle states that the inhomogeneous term $g(\tau)$ at any instant $\tau$ has an effect on the solution at time $t$ given by $e^{A(t-\tau)}g(\tau)$. Note that this is very similar to the idea of a Green's function for the boundary value problem (BVP).

As a special case, if $A = 0$, then the ODE is simply

$$u'(t) = g(t) \tag{5.9}$$

and of course the solution (5.8) reduces to the integral of $g$:

$$u(t) = \eta + \int_{t_0}^{t} g(\tau)\,d\tau. \tag{5.10}$$

As another special case, suppose $A$ is constant and so is $g(t) \equiv g \in \mathbb{R}^s$. Then (5.8) reduces to

$$u(t) = e^{A(t-t_0)}\eta + \left(\int_{t_0}^{t} e^{A(t-\tau)}\,d\tau\right) g. \tag{5.11}$$

This integral can be computed, e.g., by expressing $e^{A(t-\tau)}$ as a Taylor series as in (D.31) and then integrating term by term. This gives

$$\int_{t_0}^{t} e^{A(t-\tau)}\,d\tau = A^{-1}\left(e^{A(t-t_0)} - I\right) \tag{5.12}$$

and so

$$u(t) = e^{A(t-t_0)}\eta + A^{-1}\left(e^{A(t-t_0)} - I\right) g. \tag{5.13}$$

This may be familiar in the scalar case and holds also for constant coefficient systems (provided $A$ is nonsingular). This form of the solution is used explicitly in exponential time differencing methods; see Section 11.6.

## 5.2 Lipschitz continuity

In the last section we considered linear ODEs, for which there is always a unique solution. In most applications, however, we are concerned with nonlinear problems for which there is usually no explicit formula for the solution. The standard theory for the existence of a solution to the initial value problem

$$u'(t) = f(u,t), \quad u(0) = \eta \tag{5.14}$$

is discussed in many texts, e.g., [15]. To guarantee that there is a unique solution it is necessary to require a certain amount of smoothness in the function $f(u,t)$ of (5.14). We say that the function $f(u,t)$ is *Lipschitz continuous* in $u$ over some domain

$$\mathcal{D} = \{(u,t) : |u-\eta| \le a, t_0 \le t \le t_1\}$$

if there exists some constant $L \ge 0$ so that

$$|f(u,t) - f(u^*,t)| \le L|u-u^*| \tag{5.15}$$

for all $(u,t)$ and $(u^*,t)$ in $\mathcal{D}$. This is slightly stronger than mere continuity, which only requires that $|f(u,t) - f(u^*,t)| \to 0$ as $u \to u^*$. Lipschitz continuity requires that $|f(u,t) - f(u^*,t)| = O(|u-u^*|)$ as $u \to u^*$.

If $f(u,t)$ is differentiable with respect to $u$ in $\mathcal{D}$ and this derivative $f_u = \partial f/\partial u$ is bounded then we can take

$$L = \max_{(u,t)\in\mathcal{D}} |f_u(u,t)|,$$

since

$$f(u,t) = f(u^*,t) + f_u(v,t)(u-u^*)$$

for some value $v$ between $u$ and $u^*$.

**Example 5.2.** For the linear problem $u'(t) = \lambda u(t) + g(t)$, $f'(u) = \lambda$ and we can take $L = |\lambda|$. This problem of course has a unique solution for any initial data $\eta$ given by (5.8) with $A = \lambda$.

In particular, if $\lambda = 0$ then $L = 0$. In this case $f(u,t) = g(t)$ is independent of $u$. The solution is then obtained by simply integrating the function $g(t)$, as in (5.10).

### 5.2.1 Existence and uniqueness of solutions

The basic existence and uniqueness theorem states that if $f$ is Lipschitz continuous over some region $\mathcal{D}$ then there is a unique solution to the initial value problem (5.14) at least up to time $T^* = \min(t_1, t_0 + a/S)$, where

$$S = \max_{(u,t)\in\mathcal{D}} |f(u,t)|.$$

Note that this is the maximum modulus of the slope that the solution $u(t)$ can attain in this time interval, so that up to time $t_0 + a/S$ we know that $u(t)$ remains in the domain $\mathcal{D}$ where (5.15) holds.

**Example 5.3.** Consider the initial value problem

$$u'(t) = (u(t))^2, \qquad u(0) = \eta > 0.$$

The function $f(u) = u^2$ is independent of $t$ and is Lipschitz continuous in $u$ over any finite interval $|u - \eta| \le a$ with $L = 2(\eta + a)$, and the maximum slope over this interval is $S = (\eta + a)^2$. The theorem guarantees that a unique solution exists at least up to time $a/(\eta + a)^2$. Since $a$ is arbitrary, we can choose $a$ to maximize this expression, which yields $a = \eta$ and so there is a solution at least up to time $1/4\eta$.

In fact this problem can be solved analytically and the unique solution is

$$u(t) = \frac{1}{\eta^{-1} - t}.$$

Note that $u(t) \to \infty$ as $t \to 1/\eta$. There is no solution beyond time $1/\eta$.

If the function $f$ is not Lipschitz continuous in any neighborhood of some point then the initial value problem may fail to have a unique solution over any time interval if this initial value is imposed.

**Example 5.4.** Consider the initial value problem

$$u'(t) = \sqrt{u(t)}$$

with initial condition

$$u(0) = 0.$$

The function $f(u) = \sqrt{u}$ is not Lipschitz continuous near $u = 0$ since $f'(u) = 1/(2\sqrt{u}) \to \infty$ as $u \to 0$. We cannot find a constant $L$ so that the bound (5.15) holds for all $u$ and $u^*$ near 0.

As a result, this initial value problem does not have a unique solution. In fact it has two distinct solutions:

$$u(t) \equiv 0$$

and

$$u(t) = \frac{1}{4}t^2.$$

### 5.2.2 Systems of equations

For systems of $s > 1$ ordinary differential equations, $u(t) \in \mathbb{R}^s$ and $f(u, t)$ is a function mapping $\mathbb{R}^s \times \mathbb{R} \to \mathbb{R}^s$. We say the function $f$ is Lipschitz continuous in $u$ in some norm $\|\cdot\|$ if there is a constant $L$ such that

$$\|f(u, t) - f(u^*, t)\| \le L\|u - u^*\| \tag{5.16}$$

for all $(u, t)$ and $(u^*, , t)$ in some domain $\mathcal{D} = \{(u, t) : \|u - \eta\| \le a, t_0 \le t \le t_1\}$. By the equivalence of finite-dimensional norms (Appendix A), if $f$ is Lipschitz continuous in one norm then it is Lipschitz continuous in any other norm, although the Lipschitz constant may depend on the norm chosen.

The theorems on existence and uniqueness carry over to systems of equations.

**Example 5.5.** Consider the pendulum problem from Section 2.16,

$$\theta''(t) = -\sin(\theta(t)),$$

which can be rewritten as a first order system of two equations by introducing $v(t) = \theta'(t)$:

$$u = \begin{bmatrix} \theta \\ v \end{bmatrix}, \qquad \frac{d}{dt}\begin{bmatrix} \theta \\ v \end{bmatrix} = \begin{bmatrix} v \\ -\sin(\theta) \end{bmatrix}.$$

Consider the max-norm. We have

$$\|u - u^*\|_\infty = \max(|\theta - \theta^*|, |v - v^*|)$$

and

$$\|f(u) - f(u^*)\|_\infty = \max(|v - v^*|, |\sin(\theta) - \sin(\theta^*)|).$$

To bound $\|f(u) - f(u^*)\|_\infty$, first note that $|v - v^*| \le \|u - u^*\|_\infty$. We also have

$$|\sin(\theta) - \sin(\theta^*)| \le |\theta - \theta^*| \le \|u - u^*\|_\infty$$

since the derivative of $\sin(\theta)$ is bounded by 1. So we have Lipschitz continuity with $L = 1$:

$$\|f(u) - f(u^*)\|_\infty \le \|u - u^*\|_\infty.$$

### 5.2.3 Significance of the Lipschitz constant

The Lipschitz constant measures how much $f(u,t)$ changes if we perturb $u$ (at some fixed time $t$). Since $f(u,t) = u'(t)$, the slope of the line tangent to the solution curve through the value $u$, this indicates how the slope of the solution curve will vary if we perturb $u$. The significance of this is best seen through some examples.

**Example 5.6.** Consider the trivial equation $u'(t) = g(t)$, which has Lipschitz constant $L = 0$ and solutions given by (5.10). Several solution curves are sketched in Figure 5.1. Note that all these curves are "parallel"; they are simply shifted depending on the initial data. Tangent lines to the curves at any particular time are all parallel since $f(u,t) = g(t)$ is independent of $u$.

**Example 5.7.** Consider $u'(t) = \lambda u(t)$ with $\lambda$ constant and $L = |\lambda|$. Then $u(t) = u(0)\exp(\lambda t)$. Two situations are shown in Figure 5.2 for negative and positive values of $\lambda$. Here the slope of the solution curve does vary depending on $u$. The variation in the slope with $u$ (at fixed $t$) gives an indication of how rapidly the solution curves are converging toward one another (in the case $\lambda < 0$) or diverging away from one another (in the case $\lambda > 0$). If the magnitude of $\lambda$ were increased, the convergence or divergence would clearly be more rapid.

The size of the Lipschitz constant is significant if we intend to solve the problem numerically since our numerical approximation will almost certainly produce a value $U^n$ at time $t_n$ that is not exactly equal to the true value $u(t_n)$. Hence we are on a different solution curve than the true solution. The best we can hope for in the future is that we stay close to the solution curve that we are now on. The size of the Lipschitz constant gives an indication of whether solution curves that start close together can be expected to stay close together or might diverge rapidly.

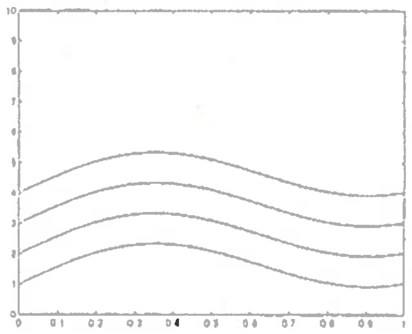

**Figure 5.1.** *Solution curves for Example* 5.6, *where* $L = 0$.

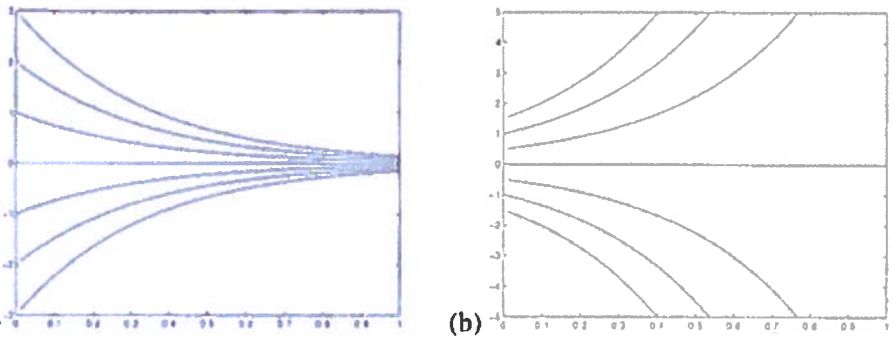

**Figure 5.2.** *Solution curves for Example* 5.7 *with* (a) $\lambda = -3$ *and* (b) $\lambda = 3$.

### 5.2.4 Limitations

Actually, the Lipschitz constant is not the perfect tool for this purpose, since it does not distinguish between rapid divergence and rapid convergence of solution curves. In both Figure 5.2(a) and Figure 5.2(b) the Lipschitz constant has the same value $L = |\lambda| = 3$. But we would expect that rapidly convergent solution curves as in Figure 5.2(a) should be easier to handle numerically than rapidly divergent ones. If we make an error at some stage, then the effect of this error should decay at later times rather than growing. To some extent this is true and as a result error bounds based on the Lipschitz constant may be orders of magnitude too large in this situation.

However, rapidly converging solution curves can also give serious numerical difficulties, which one might not expect at first glance. This is discussed in detail in Chapter 8, which covers stiff equations.

One should also keep in mind that a small value of the Lipschitz constant does not necessarily mean that two solution curves starting close together will stay close together forever.

**Example 5.8.** Consider two solutions to the pendulum problem from Example 5.5, one with initial data

$$\theta_1(0) = \pi - \epsilon, \qquad v_1(0) = 0,$$

and the other with

$$\theta_2(0) = \pi + \epsilon, \qquad v_2(0) = 0.$$

The Lipschitz constant is 1 and the data differ by $2\epsilon$, which can be arbitrarily small, and yet the solutions eventually diverge dramatically, as Solution 1 falls toward $\theta = 0$, while in Solution 2 the pendulum falls the other way, toward $\theta = 2\pi$.

In this case the IVP is very ill conditioned: small changes in the data can lead to order 1 changes in the solution. As always in numerical analysis, the solution of ill-conditioned problems can be very hard to compute accurately.

## 5.3 Some basic numerical methods

We begin by listing a few standard approaches to discretizing (5.1). Note that the IVP differs from the BVP considered before in that we are given all the data at the initial time $t_0 = 0$ and from this we should be able to march forward in time, computing approximations at successive times $t_1, t_2, \ldots$. We will use $k$ to denote the time step, so $t_n = nk$ for $n \geq 0$. It is convenient to use the symbol $k$, which is different from the spatial grid size $h$, since we will soon study PDEs which involve both spatial and temporal discretizations. Often the symbols $\Delta t$ and $\Delta x$ are used.

We are given initial data

$$U^0 = \eta \tag{5.17}$$

and want to compute approximations $U^1, U^2, \ldots$ satisfying

$$U^n \approx u(t_n).$$

We will use superscripts to denote the time step index, again anticipating the notation of PDEs where we will use subscripts for spatial indices.

The simplest method is *Euler's method* (also called *forward Euler*), based on replacing $u'(t_n)$ with $D_+U^n = (U^{n+1} - U^n)/k$ from (1.1). This gives the method

$$\frac{U^{n+1} - U^n}{k} = f(U^n), \qquad n = 0, 1, \ldots. \tag{5.18}$$

Rather than viewing this as a system of simultaneous equations as we did for the BVP, it is possible to solve this explicitly for $U^{n+1}$ in terms of $U^n$:

$$U^{n+1} = U^n + kf(U^n). \tag{5.19}$$

From the initial data $U^0$ we can compute $U^1$, then $U^2$, and so on. This is called a *time-marching method*.

The *backward Euler* method is similar but is based on replacing $u'(t_{n+1})$ with $D_-U^{n+1}$:

$$\frac{U^{n+1} - U^n}{k} = f(U^{n+1}) \tag{5.20}$$

or

$$U^{n+1} = U^n + kf(U^{n+1}). \tag{5.21}$$

Again we can march forward in time since computing $U^{n+1}$ requires only that we know the previous value $U^n$. In the backward Euler method, however, (5.21) is an equation that must be solved for $U^{n+1}$, and in general $f(u)$ is a nonlinear function. We can view this as looking for a zero of the function

$$g(u) = u - kf(u) - U^n,$$

which can be approximated using some iterative method such as *Newton's method*.

Because the backward Euler method gives an equation that must be solved for $U^{n+1}$, it is called an *implicit* method, whereas the forward Euler method (5.19) is an *explicit* method.

Another implicit method is the *trapezoidal method*, obtained by averaging the two Euler methods:

$$\frac{U^{n+1} - U^n}{k} = \frac{1}{2}(f(U^n) + f(U^{n+1})). \tag{5.22}$$

As one might expect, this symmetric approximation is second order accurate, whereas the Euler methods are only first order accurate.

The above methods are all *one-step methods*, meaning that $U^{n+1}$ is determined from $U^n$ alone and previous values of $U$ are not needed. One way to get higher order accuracy is to use a *multistep* method that involves other previous values. For example, using the approximation

$$\frac{u(t+k) - u(t-k)}{2k} = u'(t) + \frac{1}{6}k^2u'''(t) + O(k^3)$$

yields the *midpoint method* (also called the *leapfrog method*),

$$\frac{U^{n+1} - U^{n-1}}{2k} = f(U^n) \tag{5.23}$$

or

$$U^{n+1} = U^{n-1} + 2kf(U^n), \tag{5.24}$$

which is a second order accurate explicit 2-step method. The approximation $D_2u$ from (1.11), rewritten in the form

$$\frac{3u(t+k) - 4u(t) + u(t-k)}{2k} = u'(t+k) + \frac{1}{12}k^2u'''(t+k) + \cdots,$$

yields a second order implicit 2-step method

$$\frac{3U^{n+1} - 4U^n + U^{n-1}}{2k} = f(U^{n+1}). \tag{5.25}$$

This is one of the backward differentiation formula (*BDF*) methods that will be discussed further in Chapter 8.

## 5.4 Truncation errors

The truncation error for these methods is defined in the same way as in Chapter 2. We write the difference equation in the form that directly models the derivatives (e.g., in the form

(5.23) rather than (5.24)) and then insert the true solution to the ODE into the difference equation. We then use Taylor series expansion and cancel out common terms.

**Example 5.9.** The local truncation error (LTE) of the midpoint method (5.23) is defined by

$$\begin{aligned}\tau^n &= \frac{u(t_{n+1}) - u(t_{n-1})}{2k} - f(u(t_n)) \\ &= \left[u'(t_n) + \frac{1}{6}k^2 u'''(t_n) + O(k^4)\right] - u'(t_n) \\ &= \frac{1}{6}k^2 u'''(t_n) + O(k^4).\end{aligned}$$

Note that since $u(t)$ is the true solution of the ODE, $u'(t_n) = f(u(t_n))$. The $O(k^3)$ term drops out by symmetry. The truncation error is $O(k^2)$ and so we say the method is *second order accurate*, although it is not yet clear that the global error will have this behavior. As always, we need some form of *stability* to guarantee that the global error will exhibit the same rate of convergence as the local truncation error. This will be discussed below.

## 5.5 One-step errors

In much of the literature concerning numerical methods for ODEs, a slightly different definition of the local truncation error is used that is based on the form (5.24), for example, rather than (5.23). Denoting this value by $\mathcal{L}^n$, we have

$$\begin{aligned}\mathcal{L}^n &= u(t_{n+1}) - u(t_{n-1}) - 2kf(u(t_n)) \\ &= \frac{1}{3}k^3 u'''(t_n) + O(k^5).\end{aligned} \tag{5.26}$$

Since $\mathcal{L}^n = 2k\tau^n$, this local error is $O(k^3)$ rather than $O(k^2)$, but of course the global error remains the same and will be $O(k^2)$. Using this alternative definition, many standard results in ODE theory say that a $p$th order accurate method should have an LTE that is $O(k^{p+1})$. With the notation we are using, a $p$th order accurate method has an LTE that is $O(k^p)$. The notation used here is consistent with the standard practice for PDEs and leads to a more coherent theory, but one should be aware of this possible source of confusion.

In this book $\mathcal{L}^n$ will be called the *one-step error*, since this can be viewed as the error that would be introduced in one time step if the past values $U^n$, $U^{n-1}$, ... were all taken to be the exact values from $u(t)$. For example, in the midpoint method (5.24) we suppose that

$$U^n = u(t_n) \quad \text{and} \quad U^{n-1} = u(t_{n-1})$$

and we now use these values to compute $U^{n+1}$, an approximation to $u(t_{n+1})$:

$$\begin{aligned}U^{n+1} &= u(t_{n-1}) + 2kf(u(t_n)) \\ &= u(t_{n-1}) + 2ku'(t_n).\end{aligned}$$

Then the error is

$$u(t_{n+1}) - U^{n+1} = u(t_{n+1}) - u(t_{n-1}) - 2ku'(t_n) = \mathcal{L}^n.$$

From (5.26) we see that in one step the error introduced is $O(k^3)$. This is consistent with second order accuracy in the global error if we think of trying to compute an approximation to the true solution $u(T)$ at some fixed time $T > 0$. To compute from time $t = 0$ up to time $T$, we need to take $T/k$ time steps of length $k$. A rough estimate of the error at time $T$ might be obtained by assuming that a new error of size $\mathcal{L}^n$ is introduced in the $n$th time step and is then simply carried along in later time steps without affecting the size of future local errors and without growing or diminishing itself. Then we would expect the resulting global error at time $T$ to be simply the sum of all these local errors. Since each local error is $O(k^3)$ and we are adding up $T/k$ of them, we end up with a global error that is $O(k^2)$.

This viewpoint is in fact exactly right for the simplest ODE (5.9), in which $f(u,t) = g(t)$ is independent of $u$ and the solution is simply the integral of $g$, but it is a bit too simplistic for more interesting equations since the error at each time feeds back into the computation at the next step in the case where $f(u,t)$ depends on $u$. Nonetheless, it is essentially right in terms of the expected order of accuracy, provided the method is stable. In fact, it is useful to think of *stability* as exactly what is needed to make this naive analysis correct, by ensuring that the old errors from previous time steps do not grow too rapidly in future time steps. This will be investigated in detail in the following chapters.

## 5.6 Taylor series methods

The forward Euler method (5.19) can be derived using a Taylor series expansion of $u(t_{n+1})$ about $u(t_n)$:

$$u(t_{n+1}) = u(t_n) + ku'(t_n) + \frac{1}{2}k^2u''(t_n) + \cdots. \tag{5.27}$$

If we drop all terms of order $k^2$ and higher and use the differential equation to replace $u'(t_n)$ with $f(u(t_n), t_n)$, we obtain

$$u(t_{n+1}) \approx u(t_n) + kf(u(t_n), t_n).$$

This suggests the method (5.19). The 1-step error is $O(k^2)$ since we dropped terms of this order.

A *Taylor series method* of higher accuracy can be derived by keeping more terms in the Taylor series. If we keep the first $p+1$ terms of the Taylor series expansion

$$u(t_{n+1}) \approx u(t_n) + ku'(t_n) + \frac{1}{2}k^2u''(t_n) + \cdots + \frac{1}{p!}k^pu^{(p)}(t_n)$$

we obtain a $p$th order accurate method. The problem is that we are given only

$$u'(t) = f(u(t), t)$$

and we must compute the higher derivatives by repeated differentiation of this function. For example, we can compute

$$\begin{aligned} u''(t) &= f_u(u(t),t)u'(t) + f_t(u(t),t) \\ &= f_u(u(t),t)f(u(t),t) + f_t(u(t),t). \end{aligned} \tag{5.28}$$

This can result in very messy expressions that must be worked out for each equation, and as a result this approach is not often used in practice. However, it is such an obvious approach that it is worth mentioning, and in some cases it may be useful. An example should suffice to illustrate the technique and its limitations.

**Example 5.10.** Suppose we want to solve the equation

$$u'(t) = t^2 \sin(u(t)). \tag{5.29}$$

Then we can compute

$$\begin{aligned} u''(t) &= 2t\sin(u(t)) + t^2\cos(u(t))\,u'(t) \\ &= 2t\sin(u(t)) + t^4\cos(u(t))\sin(u(t)). \end{aligned}$$

A second order method is given by

$$U^{n+1} = U^n + kt_n^2\sin(U^n) + \frac{1}{2}k^2[2t_n\sin(U^n) + t_n^4\cos(U^n)\sin(U^n)].$$

Clearly higher order derivatives can be computed and used, but this is cumbersome even for this simple example. For systems of equations the method becomes still more complicated.

This Taylor series approach does get used in some situations, however—for example, in the derivation of the Lax–Wendroff method for hyperbolic PDEs; see Section 10.3. See also Section 11.3.

## 5.7 Runge–Kutta methods

Most methods used in practice do not require that the user explicitly calculate higher order derivatives. Instead a higher order finite difference approximation is designed that typically models these terms automatically.

A multistep method of the sort we will study in Section 5.9 can achieve high accuracy by using high order polynomial interpolation through several previous values of the solution and/or its derivative. To achieve the same effect with a 1-step method it is typically necessary to use a *multistage* method, where intermediate values of the solution and its derivative are generated and used within a single time step.

**Example 5.11.** A two-stage explicit Runge–Kutta method is given by

$$\begin{aligned} U^* &= U^n + \frac{1}{2}kf(U^n), \\ U^{n+1} &= U^n + kf(U^*). \end{aligned} \tag{5.30}$$

In the first stage an intermediate value is generated that approximates $u(t_{n+1/2})$ via Euler's method. In the second step the function $f$ is evaluated at this midpoint to estimate the slope over the full time step. Since this now looks like a centered approximation to the derivative we might hope for second order accuracy, as we'll now verify by computing the LTE.

Combining the two steps above, we can rewrite the method as

$$U^{n+1} = U^n + kf\left(U^n + \frac{1}{2}kf(U^n)\right).$$

Viewed this way, this is clearly a 1-step explicit method. The truncation error is

$$\tau^n = \frac{1}{k}(u(t_{n+1}) - u(t_n)) - f\left(u(t_n) + \frac{1}{2}kf(u(t_n))\right). \tag{5.31}$$

Note that

$$\begin{aligned} f\left(u(t_n) + \frac{1}{2}kf(u(t_n))\right) &= f\left(u(t_n) + \frac{1}{2}ku'(t_n)\right) \\ &= f(u(t_n)) + \frac{1}{2}ku'(t_n)f'(u(t_n)) + \frac{1}{8}k^2(u'(t_n))^2 f''(u(t_n)) + \cdots. \end{aligned}$$

Since $f(u(t_n)) = u'(t_n)$ and differentiating gives $f'(u)u' = u''$, we obtain

$$f\left(u(t_n) + \frac{1}{2}kf(u(t_n))\right) = u'(t_n) + \frac{1}{2}ku''(t_n) + O(k^2).$$

Using this in (5.31) gives

$$\begin{aligned} \tau^n &= \frac{1}{k}\left(ku'(t_n) + \frac{1}{2}k^2u''(t_n) + O(k^3)\right) \\ &\qquad - \left(u'(t_n) + \frac{1}{2}ku''(t_n) + O(k^2)\right) \\ &= O(k^2) \end{aligned}$$

and the method is second order accurate. (Check the $O(k^2)$ term to see that this does not vanish.)

*Remark*: Another way to determine the order of accuracy of this simple method is to apply it to the special test equation $u' = \lambda u$, which has solution $u(t_{n+1}) = e^{\lambda k}u(t_n)$, and determine the error on this problem. Here we obtain

$$\begin{aligned} U^{n+1} &= U^n + k\lambda\left(U^n + \frac{1}{2}k\lambda U^n\right) \\ &= U^n + (k\lambda)U^n + \frac{1}{2}(k\lambda)^2U^n \\ &= e^{k\lambda}U^n + O(k^3). \end{aligned}$$

The one-step error is $O(k^3)$ and hence the LTE is $O(k^2)$. Of course we have checked only that the LTE is $O(k^2)$ on one particular function $u(t) = e^{\lambda t}$, not on all smooth solutions, and for general Runge–Kutta methods for nonautonomous problems this approach gives only an upper bound on the method's order of accuracy. Applying a method to this special equation is also a fundamental tool in stability analysis—see Chapter 7.

**Example 5.12.** The Runge–Kutta method (5.30) can be extended to nonautonomous equations of the form $u'(t) = f(u(t), t)$:

$$\begin{aligned} U^* &= U^n + \frac{1}{2}kf(U^n, t_n), \\ U^{n+1} &= U^n + kf\left(U^*, t_n + \frac{k}{2}\right). \end{aligned} \tag{5.32}$$

This is again second order accurate, as can be verified by expanding as above, but it is slightly more complicated since Taylor series in two variables must be used.

**Example 5.13.** One simple higher order Runge–Kutta method is the fourth order four-stage method given by

$$\begin{aligned}
Y_1 &= U^n,\\
Y_2 &= U^n + \frac{1}{2}kf(Y_1, t_n),\\
Y_3 &= U^n + \frac{1}{2}kf\left(Y_2, t_n + \frac{k}{2}\right),\\
Y_4 &= U^n + kf\left(Y_3, t_n + \frac{k}{2}\right),\\
U^{n+1} &= U^n + \frac{k}{6}\left[f(Y_1, t_n) + 2f\left(Y_2, t_n + \frac{k}{2}\right)\right.\\
&\qquad \left. +2f\left(Y_3, t_n + \frac{k}{2}\right) + f(Y_4, t_n + k)\right].
\end{aligned} \tag{5.33}$$

Note that if $f(u, t) = f(t)$ does not depend on $u$, then this reduces to Simpson's rule for the integral. This method was particularly popular in the precomputer era, when computations were done by hand, because the coefficients are so simple. Today there is no need to keep the coefficients simple and other Runge–Kutta methods have advantages.

A general $r$-stage Runge–Kutta method has the form

$$\begin{aligned}
Y_1 &= U^n + k\sum_{j=1}^{r} a_{1j} f(Y_j, t_n + c_j k),\\
Y_2 &= U^n + k\sum_{j=1}^{r} a_{2j} f(Y_j, t_n + c_j k),\\
&\vdots\\
Y_r &= U^n + k\sum_{j=1}^{r} a_{rj} f(Y_j, t_n + c_j k),\\
U^{n+1} &= U^n + k\sum_{j=1}^{r} b_j f(Y_j, t_n + c_j k).
\end{aligned} \tag{5.34}$$

Consistency requires

$$\begin{aligned}
\sum_{j=1}^{r} a_{ij} &= c_i, \quad i = 1, 2, \ldots, r,\\
\sum_{j=1}^{r} b_j &= 1.
\end{aligned} \tag{5.35}$$

If these conditions are satisfied, then the method will be at least first order accurate.

The coefficients for a Runge–Kutta method are often displayed in a so-called Butcher tableau:

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1r} \\ \vdots & \vdots & & \vdots \\ c_r & a_{r1} & \dots & a_{rr} \\ \hline & b_1 & \dots & b_r \end{array} \tag{5.36}$$

For example, the fourth order Runge–Kutta method given in (5.33) has the following tableau (entries not shown are all 0):

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

An important class of Runge–Kutta methods consists of the *explicit methods* for which $a_{ij} = 0$ for $j \geq i$. For an explicit method, the elements on and above the diagonal in the $a_{ij}$ portion of the Butcher tableau are all equal to zero, as, for example, with the fourth order method displayed above. With an explicit method, each of the $Y_i$ values is computed using only the previously computed $Y_j$.

Fully implicit Runge–Kutta methods, in which each $Y_i$ depends on all the $Y_j$, can be expensive to implement on systems of ODEs. For a system of $s$ equations (where each $Y_i$ is in $\mathbb{R}^s$), a system of $sr$ equations must be solved to compute the $r$ vectors $Y_i$ simultaneously.

One subclass of implicit methods that are simpler to implement are the *diagonally implicit* Runge–Kutta methods (DIRK methods) in which $Y_i$ depends on $Y_j$ for $j \leq i$, i.e., $a_{ij} = 0$ for $j > i$. For a system of $s$ equations, DIRK methods require solving a sequence of $r$ implicit systems, each of size $s$, rather than a coupled set of $sr$ equations as would be required in a fully implicit Runge–Kutta method. DIRK methods are so named because their tableau has zero values above the diagonal but possibly nonzero diagonal elements.

**Example 5.14.** A second order accurate DIRK method is given by

$$\begin{aligned} Y_1 &= U^n, \\ Y_2 &= U^n + \frac{k}{4}\left[f(Y_1, t_n) + f\left(Y_2, t_n + \frac{k}{2}\right)\right], \\ Y_3 &= U^n + \frac{k}{3}\left[f(Y_1, t_n) + f\left(Y_2, t_n + \frac{k}{2}\right) + f(Y_3, t_n + k)\right], \\ U^{n+1} = Y_3 &= U^n + \frac{k}{3}\left[f(Y_1, t_n) + f\left(Y_2, t_n + \frac{k}{2}\right) + f(Y_3, t_n + k)\right]. \end{aligned} \tag{5.37}$$

This method is known as the TR-BDF2 method and is derived in a different form in Section 8.5. Its tableau is

$$\begin{array}{c|ccc}
0 & & & \\
1/2 & 1/4 & 1/4 & \\
1 & 1/3 & 1/3 & 1/3 \\
\hline
 & 1/3 & 1/3 & 1/3
\end{array}$$

In addition to the conditions (5.35), a Runge–Kutta method is second order accurate if

$$\sum_{j=1}^{r} b_j c_j = \frac{1}{2}, \tag{5.38}$$

as is satisfied for the method (5.37). Third order accuracy requires two additional conditions:

$$\begin{aligned}
\sum_{j=1}^{r} b_j c_j^2 &= \frac{1}{3}, \\
\sum_{i=1}^{r} \sum_{j=1}^{r} b_i a_{ij} c_j &= \frac{1}{6}.
\end{aligned} \tag{5.39}$$

Fourth order accuracy requires an additional four conditions on the coefficients, and higher order methods require an exponentially growing number of conditions.

An $r$-stage explicit Runge–Kutta method can have order at most $r$, although for $r \geq 5$ the order is strictly less than the number of stages. Among implicit Runge–Kutta methods, $r$-stage methods of order $2r$ exist. There typically are many ways that the coefficients $a_{ij}$ and $b_j$ can be chosen to achieve a given accuracy, provided the number of stages is sufficiently large. Many different classes of Runge–Kutta methods have been developed over the years with various advantages. The order conditions are quite complicated for higher-order methods and an extensive theory has been developed by Butcher for analyzing these methods and their stability properties. For more discussion and details see, for example, [13], [43], [44].

Using more stages to increase the order of a method is an obvious strategy. For some problems, however, we will also see that it can be advantageous to use a large number of stages to increase the *stability* of the method while keeping the order of accuracy relatively low. This is the idea behind the *Runge–Kutta–Chebyshev methods*, for example, discussed in Section 8.6.

### 5.7.1 Embedded methods and error estimation

Most practical software for solving ODEs does not use a fixed time step but rather adjusts the time step during the integration process to try to achieve some specified error bound. One common way to estimate the error in the computation is to compute using two different methods and compare the results. Knowing something about the error behavior of each method often allows the possibility of estimating the error in at least one of the two results.

A simple way to do this for ODEs is to take a time step with two different methods, one of order $p$ and one of a different order, say, $p + 1$. Assuming that the time step is small enough that the higher order method is really generating a better approximation, then the difference between the two results will be an estimate of the one-step error in the lower order method. This can be used as the basis for choosing an appropriate time step for the lower order approximation. Often the time step is chosen in this manner, but then the higher order solution is used as the actual approximation at this time and as the starting point for the next time step. This is sometimes called *local extrapolation*. Once this is done there is no estimate of the error, but presumably it is even smaller than the error in the lower order method and so the approximation generated will be even better than the required tolerance. For more about strategies for time step selection, see, for example, [5], [43], [78].

Note, however, that the procedure of using two different methods in every time step could easily double the cost of the computation unless we choose the methods carefully. Since the main cost in a Runge–Kutta method is often in evaluating the function $f(u,t)$, it makes sense to reuse function values as much as possible and look for methods that provide two approximations to $U^{n+1}$ of different order based on the same set of function evaluations, by simply taking different linear combinations of the $f(Y_j, t_n + c_j k)$ values in the final stage of the Runge–Kutta method (5.34). So in addition to the value $U^{n+1}$ given there we would like to also compute a value

$$\hat{U}^{n+1} = U^n + k \sum_{j=1}^{r} \hat{b}_j f(Y_j, t_n + c_j k) \tag{5.40}$$

that gives an approximation of a different order that can be used for error estimation. These are called *embedded Runge–Kutta methods* and are often displayed in a tableau of the form

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1r} \\ \vdots & \vdots & & \vdots \\ c_r & a_{r1} & \dots & a_{rr} \\ \hline & b_1 & \dots & b_r \\ \hline & \hat{b}_1 & \dots & \hat{b}_r \end{array} \tag{5.41}$$

As a very simple example, the second order Runge–Kutta method (5.32) could be combined with the first order Euler method:

$$\begin{aligned} Y_1 &= U^n, \\ Y_2 &= U^n + \frac{1}{2} k f(Y_1, t_n), \\ U^{n+1} &= U^n + k f\left(Y_2, t_n + \frac{k}{2}\right), \\ \hat{U}^{n+1} &= U^n + k f(Y_1, t_n). \end{aligned} \tag{5.42}$$

Note that the computation of $\hat{U}^{n+1}$ reuses the value $f(Y_1, t_n)$ obtained in computing $Y_2$ and is essentially free. Also note that

$$\begin{aligned}\hat{U}^{n+1} - U^{n+1} &= k\left(f(Y_1, t_n) - f\left(Y_2, t_n + \frac{k}{2}\right)\right) \\ &\approx k\left(u'(t_n) - u'(t_{n+1/2})\right) \qquad (5.43) \\ &\approx \frac{1}{2}k^2 u''(t_n),\end{aligned}$$

which is approximately the one-step error for Euler's method.

Most software based on Runge–Kutta methods uses embedded methods of higher order. For example, the `ode45` routine in MATLAB uses a pair of embedded Runge-Kutta methods of order 4 and 5 due to Dormand and Prince [25]. See Shampine and Reichelt [78] for implementation details (or `typeode45` in MATLAB).

## 5.8 One-step versus multistep methods

Taylor series and Runge–Kutta methods are *one-step methods*; the approximation $U^{n+1}$ depends on $U^n$ but not on previous values $U^{n-1}, U^{n-2}, \ldots$. In the next section we will consider a class of multistep methods where previous values are also used (one example is the midpoint method (5.24)).

One-step methods have several advantages over multistep methods:

- The methods are *self-starting*: from the initial data $U^0$ the desired method can be applied immediately. Multistep methods require that some other method be used initially, as discussed in Section 5.9.3.
- The time step $k$ can be changed at any point, based on an error estimate, for example. The time step can also be changed with a multistep method but more care is required since the previous values are assumed to be equally spaced in the standard form of these methods given below.
- If the solution $u(t)$ is not smooth at some isolated point $t^*$ (for example, because $f(u, t)$ is discontinuous at $t^*$), then with a one-step method it is often possible to get full accuracy simply by ensuring that $t^*$ is a grid point. With a multistep method that uses data from both sides of $t^*$ in approximating derivatives, a loss of accuracy may occur.

On the other hand, one-step methods have some disadvantages. The disadvantage of Taylor series methods is that they require differentiating the given equation and are cumbersome and often expensive to implement. Runge–Kutta methods only use evaluations of the function $f$, but a higher order multistage method requires evaluating $f$ several times each time step. For simple equations this may not be a problem, but if function values are expensive to compute, then high order Runge–Kutta methods may be quite expensive as well. This is particularly true for implicit methods, where an implicit nonlinear system must be solved in each stage.

An alternative is to use a multistep method in which values of $f$ already computed in previous time steps are reused to obtain higher order accuracy. Typically only one new $f$ evaluation is required in each time step. The popular class of *linear multistep methods* is discussed in the next section.

## 5.9 Linear multistep methods

All the methods introduced in Section 5.3 are members of a class of methods called linear multistep methods (LMMs). In general, an $r$-step LMM has the form

$$\sum_{j=0}^{r} \alpha_j U^{n+j} = k \sum_{j=0}^{r} \beta_j f(U^{n+j}, t_{n+j}). \tag{5.44}$$

The value $U^{n+r}$ is computed from this equation in terms of the previous values $U^{n+r-1}$, $U^{n+r-2}, \dots, U^n$ and $f$ values at these points (which can be stored and reused if $f$ is expensive to evaluate).

If $\beta_r = 0$, then the method (5.44) is explicit; otherwise it is implicit. Note that we can multiply both sides by any constant and have essentially the same method, although the coefficients $\alpha_j$ and $\beta_j$ would change. The normalization $\alpha_r = 1$ is often assumed to fix this scale factor.

There are special classes of methods of this form that are particularly useful and have distinctive names. These will be written out for the autonomous case where $f(u,t) = f(u)$ to simplify the formulas, but each can be used more generally by replacing $f(U^{n+j})$ with $f(U^{n+j}, t_{n+j})$ in any of the formulas.

**Example 5.15.** The *Adams methods* have the form

$$U^{n+r} = U^{n+r-1} + k \sum_{j=0}^{r} \beta_j f(U^{n+j}). \tag{5.45}$$

These methods all have

$$\alpha_r = 1, \quad \alpha_{r-1} = -1, \quad \text{and } \alpha_j = 0 \text{ for } j < r-1.$$

The $\beta_j$ coefficients are chosen to maximize the order of accuracy. If we require $\beta_r = 0$ so the method is explicit, then the $r$ coefficients $\beta_0, \beta_1, \dots, \beta_{r-1}$ can be chosen so that the method has order $r$. This can be done by using Taylor series expansion of the local truncation error and then choosing the $\beta_j$ to eliminate as many terms as possible. This gives the explicit *Adams–Bashforth methods*.

Another way to derive the Adams–Bashforth methods is by writing

$$\begin{aligned} u(t_{n+r}) &= u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} u'(t)\,dt \\ &= u(t_{n+r-1}) + \int_{t_{n+r-1}}^{t_{n+r}} f(u(t))\,dt \end{aligned} \tag{5.46}$$

and then applying a quadrature rule to this integral to approximate

$$\int_{t_{n+r-1}}^{t_{n+r}} f(u(t))\,dt \approx k \sum_{j=1}^{r-1} \beta_j f(u(t_{n+j})). \tag{5.47}$$

This quadrature rule can be derived by interpolating $f(u(t))$ by a polynomial $p(t)$ of degree $r-1$ at the points $t_n, t_{n+1}, \dots, t_{n+r-1}$ and then integrating the interpolating polynomial.

Either approach gives the same $r$-step explicit method. The first few are given below.

**Explicit Adams–Bashforth methods**

1-step: $U^{n+1} = U^n + kf(U^n)$ (forward Euler)

2-step: $U^{n+2} = U^{n+1} + \frac{k}{2}(-f(U^n) + 3f(U^{n+1}))$

3-step: $U^{n+3} = U^{n+2} + \frac{k}{12}(5f(U^n) - 16f(U^{n+1}) + 23f(U^{n+2}))$

4-step: $U^{n+4} = U^{n+3} + \frac{k}{24}(-9f(U^n) + 37f(U^{n+1}) - 59f(U^{n+2}) + 55f(U^{n+3}))$

If we allow $\beta_r$ to be nonzero, then we have one more free parameter and so we can eliminate an additional term in the LTE. This gives an implicit method of order $r+1$ called the $r$-step *Adams–Moulton*. These methods can again be derived by polynomial interpolation, now using a polynomial $p(t)$ of degree $r$ that interpolates $f(u(t))$ at the points $t_n, t_{n+1}, \dots, t_{n+r}$ and then integrating the interpolating polynomial.

**Implicit Adams–Moulton methods**

1-step: $U^{n+1} = U^n + \frac{k}{2}(f(U^n) + f(U^{n+1}))$ (trapezoidal method)

2-step: $U^{n+2} = U^{n+1} + \frac{k}{12}(-f(U^n) + 8f(U^{n+1}) + 5f(U^{n+2}))$

3-step: $U^{n+3} = U^{n+2} + \frac{k}{24}(f(U^n) - 5f(U^{n+1}) + 19f(U^{n+2}) + 9f(U^{n+3}))$

4-step: $U^{n+4} = U^{n+3} + \frac{k}{720}(-19f(U^n) + 106f(U^{n+1}) - 264f(U^{n+2})$
$+ 646f(U^{n+3}) + 251f(U^{n+4}))$

**Example 5.16.** The explicit *Nyström methods* have the form

$$U^{n+r} = U^{n+r-2} + k\sum_{j=0}^{r-1}\beta_j f(U^{n+j})$$

with the $\beta_j$ chosen to give order $r$. The midpoint method (5.23) is a two-step explicit Nyström method. A two-step implicit Nyström method is *Simpson's rule*,

$$U^{n+2} = U^n + \frac{2k}{6}(f(U^n) + 4f(U^{n+1}) + f(U^{n+2})).$$

This reduces to Simpson's rule for quadrature if applied to the ODE $u'(t) = f(t)$.

## 5.9.1 Local truncation error

For LMMs it is easy to derive a general formula for the LTE. We have

$$\tau(t_{n+r}) = \frac{1}{k}\left(\sum_{j=0}^{r}\alpha_j u(t_{n+j}) - k\sum_{j=0}^{r}\beta_j u'(t_{n+j})\right).$$

We have used $f(u(t_{n+j})) = u'(t_{n+j})$ since $u(t)$ is the exact solution of the ODE. Assuming $u$ is smooth and expanding in Taylor series gives

$$u(t_{n+j}) = u(t_n) + jku'(t_n) + \frac{1}{2}(jk)^2u''(t_n) + \cdots,$$
$$u'(t_{n+j}) = u'(t_n) + jku''(t_n) + \frac{1}{2}(jk)^2u'''(t_n) + \cdots,$$

and so

$$\begin{aligned}\tau(t_{n+r}) = \frac{1}{k}\left(\sum_{j=0}^{r}\alpha_j\right)u(t_n) + \left(\sum_{j=0}^{r}(j\alpha_j - \beta_j)\right)u'(t_n) \\ + k\left(\sum_{j=0}^{r}\left(\frac{1}{2}j^2\alpha_j - j\beta_j\right)\right)u''(t_n) \\ + \cdots + k^{q-1}\left(\sum_{j=0}^{r}\left(\frac{1}{q!}j^q\alpha_j - \frac{1}{(q-1)!}j^{q-1}\beta_j\right)\right)u^{(q)}(t_n) + \cdots.\end{aligned}$$

The method is *consistent* if $\tau \to 0$ as $k \to 0$, which requires that at least the first two terms in this expansion vanish:

$$\sum_{j=0}^{r}\alpha_j = 0 \quad \text{and} \quad \sum_{j=0}^{r} j\alpha_j = \sum_{j=0}^{r}\beta_j. \tag{5.48}$$

If the first $p+1$ terms vanish, then the method will be $p$th order accurate. Note that these conditions depend only on the coefficients $\alpha_j$ and $\beta_j$ of the method and not on the particular differential equation being solved.

### 5.9.2 Characteristic polynomials

It is convenient at this point to introduce the so-called characteristic polynomials $\rho(\zeta)$ and $\sigma(\zeta)$ for the LMM:

$$\rho(\zeta) = \sum_{j=0}^{r}\alpha_j\zeta^j \quad \text{and} \quad \sigma(\zeta) = \sum_{j=0}^{r}\beta_j\zeta^j. \tag{5.49}$$

The first of these is a polynomial of degree $r$. So is $\sigma(\zeta)$ if the method is implicit; otherwise its degree is less than $r$. Note that $\rho(1) = \sum\alpha_j$ and also that $\rho'(\zeta) = \sum j\alpha_j\zeta^{j-1}$, so that the consistency conditions (5.48) can be written quite concisely as conditions on these two polynomials:

$$\rho(1) = 0 \quad \text{and} \quad \rho'(1) = \sigma(1). \tag{5.50}$$

This, however, is not the main reason for introducing these polynomials. The location of the roots of certain polynomials related to $\rho$ and $\sigma$ plays a fundamental role in stability theory as we will see in the next two chapters.

**Example 5.17.** The two-step Adams–Moulton method

$$U^{n+2} = U^{n+1} + \frac{k}{12}(-f(U^n) + 8f(U^{n+1}) + 5f(U^{n+2})) \tag{5.51}$$

has characteristic polynomials

$$\rho(\zeta) = \zeta^2 - \zeta, \qquad \sigma(\zeta) = \frac{1}{12}(-1 + 8\zeta + 5\zeta^2). \tag{5.52}$$

### 5.9.3 Starting values

One difficulty with using LMMs if $r > 1$ is that we need the values $U^0, U^1, \dots, U^{r-1}$ before we can begin to apply the multistep method. The value $U^0 = \eta$ is known from the initial data for the problem, but the other values are not and typically must be generated by some other numerical method or methods.

**Example 5.18.** If we want to use the midpoint method (5.23), then we need to generate $U^1$ by some other method before we begin to apply (5.23) with $n = 1$. We can obtain $U^1$ from $U^0$ using any one-step method, such as Euler's method or the trapezoidal method, or a higher order Taylor series or Runge–Kutta method. Since the midpoint method is second order accurate we need to make sure that the value $U^1$ we generate is sufficiently accurate so that this second order accuracy will not be lost. Our first impulse might be to conclude that we need to use a second order accurate method such as the trapezoidal method rather than the first order accurate Euler method, but this is wrong. The overall method is second order in either case. The reason that we achieve second order accuracy even if Euler is used in the first step is exactly analogous to what was observed earlier for boundary value problems, where we found that we can often get away with one order of accuracy lower in the local error at a single point than what we have elsewhere.

In the present context this is easiest to explain in terms of the one-step error. The midpoint method has a one-step error that is $O(k^3)$ and because this method is applied in $O(T/k)$ time steps, the global error is expected to be $O(k^2)$. Euler's method has a one-step error that is $O(k^2)$, but we are applying this method only once.

If $U^0 = \eta = u(0)$, then the error in $U^1$ obtained with Euler will be $O(k^2)$. If the midpoint method is stable, then this error will not be magnified unduly in later steps and its contribution to the global error will be only $O(k^2)$. The overall second order accuracy will not be affected.

More generally, with an $r$-step method of order $p$, we need $r$ starting values

$$U^0, U^1, \dots, U^{r-1}$$

and we need to generate these values using a method that has a *one-step error* that is $O(k^p)$ (corresponding to an LTE that is $O(k^{p-1})$). Since the number of times we apply this method $(r-1)$ is independent of $k$ as $k \to 0$, this is sufficient to give an $O(k^p)$ global error. Of course somewhat better accuracy (a smaller error constant) may be achieved by using a $p$th order accurate method for the starting values, which takes little additional work.

In software for the IVP, multistep methods generally are implemented in a form that allows changing the time step during the integration process, as is often required to efficiently solve the problem. Typically the order of the method is also allowed to vary, depending on how the solution is behaving. In such software it is then natural to solve the starting-value problem by initially taking a small time step with a one-step method and then ramping up to higher order methods and longer time steps as the integration proceeds and more past data are available.

### 5.9.4 Predictor-corrector methods

The idea of comparing results obtained with methods of different order as a way to choose the time step, discussed in Section 5.7.1 for Runge–Kutta methods, is also used with LMMs. One approach is to use a *predictor-corrector method*, in which an explicit Adams–Bashforth method of some order is used to predict a value $\hat{U}^{n+1}$ and then the Adams–Moulton method of the same order is used to "correct" this value. This is done by using $\hat{U}^{n+1}$ on the right-hand side of the Adams–Moulton method inside the $f$ evaluation, so that the Adams–Moulton formula is no longer implicit. For example, the one-step Adams–Bashforth (Euler's method) and the one-step Adams–Moulton method (the trapezoidal method) could be combined into

$$\begin{aligned}\hat{U}^{n+1} &= U^n + kf(U^n),\\ U^{n+1} &= U^n + \frac{1}{2}k(f(U^n) + f(\hat{U}^{n+1})).\end{aligned} \tag{5.53}$$

It can be shown that this method is second order accurate, like the trapezoidal method, but it also generates a lower order approximation and the difference between the two can be used to estimate the error. The MATLAB routine `ode113` uses this approach, with Adams–Bashforth–Moulton methods of orders 1–12; see [78].

Chapter 6

# Zero-Stability and Convergence for Initial Value Problems

## 6.1 Convergence

To discuss the convergence of a numerical method for the initial value problem, we focus on a fixed (but arbitrary) time $T > 0$ and consider the error in our approximation to $u(T)$ computed with the method using time step $k$. The method converges on this problem if this error goes to zero as $k \to 0$. Note that the number of time steps that we need to take to reach time $T$ increases as $k \to 0$. If we use $N$ to denote this value ($N = T/k$), then convergence means that

$$\lim_{\substack{k\to 0\\ Nk=T}} U^N = u(T). \tag{6.1}$$

In principle a method might converge on one problem but not on another, or converge with one set of starting values but not with another set. To speak of a *method* being *convergent* in general, we require that it converges on *all* problems in a reasonably large class with *all* reasonable starting values. For an $r$-step method we need $r$ starting values. These values will typically depend on $k$, and to make this clear we will write them as $U^0(k), U^1(k), \dots, U^{r-1}(k)$. While these will generally approximate $u(t)$ at the times $t_0 = 0, t_1 = k, \dots, t_{r-1} = (r-1)k$, respectively, as $k \to 0$, each of these times approaches $t_0 = 0$. So the weakest condition we might put on our starting values is that they converge to the correct initial value $\eta$ as $k \to 0$:

$$\lim_{k\to 0} U^\nu(k) = \eta \quad \text{for } \nu = 0, 1, \dots, r-1. \tag{6.2}$$

We can now state the definition of convergence.

**Definition 6.1.** *An $r$-step method is said to be* convergent *if applying the method to any ODE* (5.1) *with $f(u,t)$ Lipschitz continuous in $u$, and with any set of starting values satisfying* (6.2), *we obtain convergence in the sense of* (6.1) *for every fixed time $T > 0$ at which the ODE has a unique solution.*

To be convergent, a method must be *consistent*, meaning as before that the local truncation error (LTE) is $o(1)$ as $k \to 0$, and also *zero-stable*, as described later in this chapter. We will begin to investigate these issues by first proving the convergence of one-step methods, which turn out to be zero-stable automatically. We start with Euler's method on linear problems, then consider Euler's method on general nonlinear problems and finally extend this to a wide class of one-step methods.

## 6.2 The test problem

Much of the theory presented below is based on examining what happens when a method is applied to a simple scalar linear equation of the form

$$u'(t) = \lambda u(t) + g(t) \tag{6.3}$$

with initial data

$$u(t_0) = \eta.$$

The solution is then given by Duhamel's principle (5.8),

$$u(t) = e^{\lambda(t-t_0)}\eta + \int_{t_0}^{t} e^{\lambda(t-\tau)} g(\tau)\,d\tau. \tag{6.4}$$

## 6.3 One-step methods

### 6.3.1 Euler's method on linear problems

If we apply Euler's method to (6.3), we obtain

$$\begin{aligned} U^{n+1} &= U^n + k(\lambda U^n + g(t_n)) \\ &= (1+k\lambda)U^n + kg(t_n). \end{aligned} \tag{6.5}$$

The LTE for Euler's method is given by

$$\begin{aligned} \tau^n &= \left(\frac{u(t_{n+1}) - u(t_n)}{k}\right) - (\lambda u(t_n) + g(t_n)) \\ &= \left(u'(t_n) + \frac{1}{2}ku''(t_n) + O(k^2)\right) - u'(t_n) \\ &= \frac{1}{2}ku''(t_n) + O(k^2). \end{aligned} \tag{6.6}$$

Rewriting this equation as

$$u(t_{n+1}) = (1+k\lambda)u(t_n) + kg(t_n) + k\tau^n$$

and subtracting this from (6.5) gives a difference equation for the global error $E^n$:

$$E^{n+1} = (1+k\lambda)E^n - k\tau^n. \tag{6.7}$$

Note that this has exactly the same form as (6.5) but with a different nonhomogeneous term: $-\tau^n$ in place of $g(t_n)$. This is analogous to equation (2.15) in the boundary value theory and again gives the relation we need between the local truncation error $\tau^n$ (which is easy to compute) and the global error $E^n$ (which we wish to bound). Note again that *linearity* plays a critical role in making this connection. We will consider nonlinear problems below.

Because the equation and method we are now considering are both so simple, we obtain an equation (6.7) that we can explicitly solve for the global error $E^n$. Applying the recursion (6.7) repeatedly we see what form the solution should take:

$$\begin{aligned} E^n &= (1+k\lambda)E^{n-1} - k\tau^{n-1} \\ &= (1+k\lambda)[(1+k\lambda)E^{n-2} - k\tau^{n-2}] - k\tau^{n-1} \\ &= \cdots . \end{aligned}$$

By induction we can easily confirm that in general

$$E^n = (1+k\lambda)^n E^0 - k\sum_{m=1}^{n}(1+k\lambda)^{n-m}\tau^{m-1}. \tag{6.8}$$

(Note that some of the superscripts are powers while others are indices!) This has a form that is very analogous to the solution (6.4) of the corresponding ordinary differential equation (ODE), where now $(1+k\lambda)^{n-m}$ plays the role of the solution operator of the homogeneous problem—it transforms data at time $t_m$ to the solution at time $t_n$. The expression (6.8) is sometimes called the discrete form of Duhamel's principle.

We are now ready to prove that Euler's method converges on (6.3). We need only observe that

$$|1+k\lambda| \le e^{k|\lambda|} \tag{6.9}$$

and so

$$(1+k\lambda)^{n-m} \le e^{(n-m)k|\lambda|} \le e^{nk|\lambda|} \le e^{|\lambda|T}, \tag{6.10}$$

provided that we restrict our attention to the finite time interval $0 \le t \le T$, so that $t_n = nk \le T$. It then follows from (6.8) that

$$\begin{aligned} |E^n| &\le e^{|\lambda|T}\left(|E^0| + k\sum_{m=1}^{n}|\tau^{m-1}|\right) \\ &\le e^{|\lambda|T}\left(|E^0| + nk\max_{1\le m\le n}|\tau^{m-1}|\right). \end{aligned} \tag{6.11}$$

Let $N = T/k$ be the number of time steps needed to reach time $T$ and set

$$\|\tau\|_\infty = \max_{0\le n\le N-1}|\tau^n|.$$

From (6.6) we expect

$$\|\tau\|_\infty \approx \frac{1}{2}k\|u''\|_\infty = O(k),$$

where $\|u''\|_\infty$ is the maximum value of the function $u''$ over the interval $[0,T]$. Then for $t = nk \le T$, we have from (6.11) that

$$|E^n| \le e^{|\lambda|T}(|E^0| + T\|\tau\|_\infty).$$

If (6.2) is satisfied then $E^0 \to 0$ as $k \to 0$. In fact for this one-step method we would generally take $U^0 = u(0) = \eta$, in which case $E^0$ drops out and we are left with

$$|E^n| \le e^{|\lambda|T} T \|\tau\|_\infty = O(k) \quad \text{as } k \to 0 \tag{6.12}$$

and hence the method converges and is in fact first order accurate.

Note where *stability* comes into the picture. The one-step error $\mathcal{L}^{m-1} = k\tau^{m-1}$ introduced in the $m$th step contributes the term $(1 + k\lambda)^{n-m}\mathcal{L}^{m-1}$ to the global error. The fact that $|(1 + k\lambda)^{n-m}| < e^{|\lambda|T}$ is uniformly bounded as $k \to 0$ allows us to conclude that each contribution to the final error can be bounded in terms of its original size as a one-step error. Hence the "naive analysis" of Section 5.5 is valid, and the global error has the same order of magnitude as the local truncation error.

### 6.3.2 Relation to stability for boundary value problems

To see how this ties in with the definition of stability used in Chapter 2 for the BVP, it may be useful to view Euler's method as giving a linear system in matrix form, although this is not the way it is used computationally. If we view the equations (6.5) for $n = 0, 1, \dots, N-1$ as a linear system $AU = F$ for $U = [U^1, U^2, \dots, U^N]^T$, then

$$A = \frac{1}{k}\begin{bmatrix} 1 & & & & & \\ -(1+k\lambda) & 1 & & & & \\ & -(1+k\lambda) & 1 & & & \\ & & & \ddots & & \\ & & & -(1+k\lambda) & 1 & \\ & & & & -(1+k\lambda) & 1 \end{bmatrix}$$

and

$$U = \begin{bmatrix} U^1 \\ U^2 \\ U^3 \\ \vdots \\ U^{N-1} \\ U^N \end{bmatrix}, \qquad F = \begin{bmatrix} (1/k + \lambda)U^0 + g(t_0) \\ g(t_1) \\ g(t_2) \\ \vdots \\ g(t_{N-2}) \\ g(t_{N-1}) \end{bmatrix}.$$

We have divided both sides of (6.5) by $k$ to conform to the notation of Chapter 2. Since the matrix $A$ is lower triangular, this system is easily solved by forward substitution, which results in the iterative equation (6.5).

If we now let $\hat{U}$ be the vector obtained from the true solution as in Chapter 2, then subtracting $A\hat{U} = F + \tau$ from $AU = F$, we obtain (2.15) (the matrix form of (6.7)) with solution (6.8). We are then in exactly the same framework as in Chapter 2. So we have convergence and a global error with the same magnitude as the local error provided that the method is stable in the sense of Definition 2.1, i.e., that the inverse of the matrix $A$ is bounded independent of $k$ for all $k$ sufficiently small.

The inverse of this matrix is easy to compute. In fact we can see from the solution (6.8) that

$$A^{-1} = k \begin{bmatrix} 1 \\ (1+k\lambda) & 1 \\ (1+k\lambda)^2 & (1+k\lambda) & 1 \\ (1+k\lambda)^3 & (1+k\lambda)^2 & (1+k\lambda) & 1 \\ \vdots & & & & \ddots \\ (1+k\lambda)^{N-1} & (1+k\lambda)^{N-2} & (1+k\lambda)^{N-3} & \cdots & (1+k\lambda) & 1 \end{bmatrix}.$$

We easily compute using (A.10a) that

$$\|A^{-1}\|_\infty = k \sum_{m=1}^{N} |(1+k\lambda)^{N-m}|$$

and so

$$\|A^{-1}\|_\infty \le kNe^{|\lambda|T} = Te^{|\lambda|T}.$$

This is uniformly bounded as $k \to 0$ for fixed $T$. Hence the method is stable and $\|E\|_\infty \le \|A^{-1}\|_\infty \|\tau\|_\infty \le Te^{|\lambda|T}\|\tau\|_\infty$, which agrees with the bound (6.12).

### 6.3.3 Euler's method on nonlinear problems

So far we have focused entirely on linear equations. Practical problems are almost always nonlinear, but for the initial value problem it turns out that it is not significantly more difficult to handle this case if we assume that $f(u)$ is Lipschitz continuous, which is reasonable in light of the discussion in Section 5.2.

Euler's method on $u' = f(u)$ takes the form

$$U^{n+1} = U^n + kf(U^n) \tag{6.13}$$

and the truncation error is defined by

$$\begin{aligned}\tau^n &= \frac{1}{k}(u(t_{n+1}) - u(t_n)) - f(u(t_n)) \\ &= \frac{1}{2}ku''(t_n) + O(k^2),\end{aligned}$$

just as in the linear case. So the true solution satisfies

$$u(t_{n+1}) = u(t_n) + kf(u(t_n)) + k\tau^n$$

and subtracting this from (6.13) gives

$$E^{n+1} = E^n + k(f(U^n) - f(u(t_n))) - k\tau^n. \tag{6.14}$$

In the linear case $f(U^n) - f(u(t_n)) = \lambda E^n$ and we get the relation (6.7) for $E^n$. In the nonlinear case we cannot express $f(U^n) - f(u(t_n))$ directly in terms of the error $E^n$ in general. However, using the Lipschitz continuity of $f$ we can get a bound on this in terms of $E^n$:

$$|f(U^n) - f(u(t_n))| \le L|U^n - u(t_n)| = L|E^n|.$$

Using this in (6.14) gives

$$|E^{n+1}| \le |E^n| + kL|E^n| + k|\tau^n| = (1+kL)|E^n| + k|\tau^n|. \tag{6.15}$$

From this inequality we can show by induction that

$$|E^n| \le (1+kL)^n|E^0| + k\sum_{m=1}^{n}(1+kL)^{n-m}|\tau^{m-1}|$$

and so, using the same steps as in obtaining (6.12) (and again assuming $E^0 = 0$), we obtain

$$|E^n| \le e^{LT}T\|\tau\|_\infty = O(k) \quad \text{as } k \to 0 \tag{6.16}$$

for all $n$ with $nk \le T$, proving that the method converges. In the linear case $L = |\lambda|$ and this reduces to exactly (6.12).

## 6.3.4 General one-step methods

A general explicit one-step method takes the form

$$U^{n+1} = U^n + k\Psi(U^n, t_n, k) \tag{6.17}$$

for some function $\Psi$, which depends on $f$ of course. We will assume that $\Psi(u,t,k)$ is continuous in $t$ and $k$ and Lipschitz continuous in $u$, with Lipschitz constant $L'$ that is generally related to the Lipschitz constant of $f$.

**Example 6.1.** For the two-stage Runge–Kutta method of Example 5.11, we have

$$\Psi(u,t,k) = f\left(u + \frac{1}{2}kf(u)\right). \tag{6.18}$$

If $f$ is Lipschitz continuous with Lipschitz constant $L$, then $\Psi$ has Lipschitz constant $L' = L + \frac{1}{2}kL^2$.

The one-step method (6.17) is *consistent* if

$$\Psi(u,t,0) = f(u,t)$$

for all $u$, $t$, and $\Psi$ is continuous in $k$. The local truncation error is

$$\tau^n = \left(\frac{u(t_{n+1}) - u(t_n)}{k}\right) - \Psi(u(t_n), t_n, k).$$

We can show that any one-step method satisfying these conditions is convergent. We have

$$u(t_{n+1}) = u(t_n) + k\Psi(u(t_n), t_n, k) + k\tau^n$$

and subtracting this from (6.17) gives

$$E^{n+1} = E^n + k\left(\Psi(U^n, t_n, k) - \Psi(u(t_n), t_n, k)\right) - k\tau^n.$$

Using the Lipschitz condition we obtain

$$|E^{n+1}| \le |E^n| + kL'|E^n| + k|\tau^n|.$$

This has exactly the same form as (6.15) and the proof of convergence proceeds exactly as from there.

## 6.4 Zero-stability of linear multistep methods

The convergence proof of the previous section shows that for one-step methods, each one-step error $k\tau^{m-1}$ has an effect on the global error that is bounded by $e^{L'T}|k\tau^{m-1}|$. Although the error is possibly amplified by a factor $e^{L'T}$, this factor is bounded independent of $k$ as $k \to 0$. Consequently the method is stable: the global error can be bounded in terms of the sum of all the one-step errors and hence has the same asymptotic behavior as the LTE as $k \to 0$. This form of stability is often called *zero-stability* in ODE theory, to distinguish it from other forms of stability that are of equal importance in practice. The fact that a method is zero-stable (and converges as $k \to 0$) is no guarantee that it will give reasonable results on the particular grid with $k > 0$ that we want to use in practice. Other "stability" issues of a different nature will be taken up in the next chapter.

But first we will investigate the issue of zero-stability for general LMMs, where the theory of the previous section does not apply directly. We begin with an example showing a consistent LMM that is *not* convergent. Examining what goes wrong will motivate our definition of zero-stability for LMMs.

**Example 6.2.** The LMM

$$U^{n+2} - 3U^{n+1} + 2U^n = -kf(U^n) \tag{6.19}$$

has an LTE given by

$$\tau^n = \frac{1}{k}[u(t_{n+2}) - 3u(t_{n+1}) + 2u(t_n) + ku'(t_n)] = \frac{5}{2}ku''(t_n) + O(k^2),$$

so the method is consistent and "first order accurate." But in fact the global error will not exhibit first order accuracy, or even convergence, in general. This can be seen even on the trivial initial-value problem

$$u'(t) = 0, \quad u(0) = 0 \tag{6.20}$$

with solution $u(t) \equiv 0$. In this problem, equation (6.19) takes the form

$$U^{n+2} - 3U^{n+1} + 2U^n = 0. \tag{6.21}$$

We need two starting values $U^0$ and $U^1$. If we take $U^0 = U^1 = 0$, then (6.21) generates $U^n = 0$ for all $n$ and in this case we certainly converge to correct solution, and in fact we get the exact solution for any $k$.

But in general we will not have the exact value $U^1$ available and will have to approximate this, introducing some error into the computation. Table 6.1 shows results obtained by applying this method with starting data $U^0 = 0$, $U^1 = k$. Since $U^1(k) \to 0$ as $k \to 0$, this is valid starting data in the context of Definition 6.1 of convergence. If the method is convergent, we should see that $U^N$, the computed solution at time $T = 1$, converges to zero as $k \to 0$. Instead it blows up quite dramatically. Similar results would be seen if we applied this method to an arbitrary equation $u' = f(u)$ and used any one-step method to compute $U^1$ from $U^0$.

The homogeneous linear difference equation (6.21) can be solved explicitly for $U^n$ in terms of the starting values $U^0$ and $U^1$. We obtain

$$U^n = 2U^0 - U^1 + 2^n(U^1 - U^0). \tag{6.22}$$

**Table 6.1.** *Solution $U^N$ to (6.21) with $U^0 = 0$, $U^1 = k$ and various values of $k = 1/N$.*

| $N$ | $U^N$ |
|---|---|
| 5 | 6.2 |
| 10 | 1023 |
| 20 | $5.4 \times 10^4$ |

It is easy to verify that this satisfies (6.21) and also the starting values. (We'll see how to solve general linear difference equations in the next section.)

Since $u(t) = 0$, the error is $E^n = U^n$ and we see that any initial errors in $U^1$ or $U^0$ are magnified by a factor $2^n$ in the global error (except in the special case $U^1 = U^0$). This exponential growth of the error is the instability that leads to nonconvergence. To rule out this sort of growth of errors, we need to be able to solve a general linear difference equation.

### 6.4.1 Solving linear difference equations

We briefly review one solution technique for linear difference equations; see Section D.2.1 for a different approach. Consider the general homogeneous linear difference equation

$$\sum_{j=0}^{r} \alpha_j U^{n+j} = 0. \tag{6.23}$$

Eventually we will look for a particular solution satisfying given initial conditions

$$U^0, U^1, \ldots, U^{r-1},$$

but to begin with we will find the general solution of the difference equation in terms of $r$ free parameters. We will hypothesize that this equation has a solution of the form

$$U^n = \zeta^n \tag{6.24}$$

for some value of $\zeta$ (here $\zeta^n$ is the $n$th power!). Plugging this into (6.23) gives

$$\sum_{j=0}^{r} \alpha_j \zeta^{n+j} = 0$$

and dividing by $\zeta^n$ yields

$$\sum_{j=0}^{r} \alpha_j \zeta^j = 0. \tag{6.25}$$

We see that (6.24) is a solution of the difference equation if $\zeta$ satisfies (6.25), i.e., if $\zeta$ is a root of the polynomial

$$\rho(\zeta) = \sum_{j=0}^{r} \alpha_j \zeta^j.$$

Note that this is just the first characteristic polynomial of the LMM introduced in (5.49). In general $\rho(\zeta)$ has $r$ roots $\zeta_1, \zeta_2, \ldots, \zeta_r$ and can be factored as

$$\rho(\zeta) = \alpha_r(\zeta - \zeta_1)(\zeta - \zeta_2)\cdots(\zeta - \zeta_r).$$

Since the difference equation is linear, any linear combination of solutions is again a solution. If $\zeta_1, \zeta_2, \ldots, \zeta_r$ are distinct ($\zeta_i \neq \zeta_j$ for $i \neq j$), then the $r$ distinct solutions $\zeta_i^n$ are linearly independent and the general solution of (6.23) has the form

$$U^n = c_1\zeta_1^n + c_2\zeta_2^n + \cdots + c_r\zeta_r^n, \tag{6.26}$$

where $c_1, \ldots, c_r$ are arbitrary constants. In this case, every solution of the difference equation (6.23) has this form. If initial conditions $U^0, U^1, \ldots, U^{r-1}$ are specified, then the constants $c_1, \ldots, c_r$ can be uniquely determined by solving the $r \times r$ linear system

$$\begin{aligned} c_1 + c_2 + \cdots + c_r &= U^0, \\ c_1\zeta_1 + c_2\zeta_2 + \cdots + c_r\zeta_r &= U^1, \\ &\vdots \quad \vdots \\ c_1\zeta_1^{r-1} + c_2\zeta_2^{r-1} + \cdots + c_r\zeta_r^{r-1} &= U^{r-1}. \end{aligned} \tag{6.27}$$

**Example 6.3.** The characteristic polynomial for the difference equation (6.21) is

$$\rho(\zeta) = 2 - 3\zeta + \zeta^2 = (\zeta - 1)(\zeta - 2) \tag{6.28}$$

with roots $\zeta_1 = 1$, $\zeta_2 = 2$. The general solution has the form

$$U^n = c_1 + c_2 \cdot 2^n$$

and solving for $c_1$ and $c_2$ from $U^0$ and $U^1$ gives the solution (6.22).

This example indicates that if $\rho(\zeta)$ has any roots that are greater than one in modulus, the method will not be convergent. It turns out that the converse is nearly true: if all the roots have modulus no greater than one, then the method is convergent, with one proviso. There must be no *repeated* roots with modulus equal to one. The next two examples illustrate this.

If the roots are not distinct, say, $\zeta_1 = \zeta_2$ for simplicity, then $\zeta_1^n$ and $\zeta_2^n$ are not linearly independent and the $U^n$ given by (6.26), while still a solution, is not the most general solution. The system (6.27) would be singular in this case. In addition to $\zeta_1^n$ there is also a solution of the form $n\zeta_1^n$ and the general solution has the form

$$U^n = c_1\zeta_1^n + c_2 n\zeta_1^n + c_3\zeta_3^n + \cdots + c_r\zeta_r^n.$$

If in addition $\zeta_3 = \zeta_1$, then the third term would be replaced by $c_3 n^2 \zeta_1^n$. Similar modifications are made for any other repeated roots. Note how similar this theory is to the standard solution technique for an $r$th order linear ODE.

**Example 6.4.** Applying the consistent LMM

$$U^{n+2} - 2U^{n+1} + U^n = \frac{1}{2}k(f(U^{n+2}) - f(U^n)) \tag{6.29}$$

to the differential equation $u'(t) = 0$ gives the difference equation

$$U^{n+2} - 2U^{n+1} + U^n = 0.$$

The characteristic polynomial is

$$\rho(\zeta) = \zeta^2 - 2\zeta + 1 = (\zeta - 1)^2 \tag{6.30}$$

so $\zeta_1 = \zeta_2 = 1$. The general solution is

$$U^n = c_1 + c_2 n.$$

For particular starting values $U^0$ and $U^1$ the solution is

$$U^n = U^0 + (U^1 - U^0)n.$$

Again we see that the solution grows with $n$, although not as dramatically as in Example 6.2 (the growth is linear rather than exponential). But this growth is still enough to destroy convergence. If we take the same starting values as before, $U^0 = 0$ and $U^1 = k$, then $U^n = kn$ and so

$$\lim_{\substack{k \to 0 \\ Nk = T}} U^N = kN = T.$$

The method converges to the function $v(t) = t$ rather than to $u(t) = 0$, and hence the LMM (6.29) is not convergent.

This example shows that if $\rho(\zeta)$ has a *repeated* root of modulus 1, then the method cannot be convergent.

**Example 6.5.** Now consider the consistent LMM

$$U^{n+3} - 2U^{n+2} + \frac{5}{4}U^{n+1} - \frac{1}{4}U^n = \frac{1}{4}hf(U^n). \tag{6.31}$$

Applying this to (6.20) gives

$$U^{n+3} - 2U^{n+2} + \frac{5}{4}U^{n+1} - \frac{1}{4}U^n = 0$$

and the characteristic polynomial is

$$\rho(\zeta) = \zeta^3 - 2\zeta^2 + \frac{5}{4}\zeta - \frac{1}{4} = (\zeta - 1)(\zeta - 0.5)^2. \tag{6.32}$$

So $\zeta_1 = 1$, $\zeta_2 = \zeta_3 = 1/2$ and the general solution is

$$U^n = c_1 + c_2\left(\frac{1}{2}\right)^n + c_3 n\left(\frac{1}{2}\right)^n.$$

Here there is a repeated root but with modulus less than 1. The linear growth of $n$ is then overwhelmed by the decay of $(1/2)^n$.

For this three-step method we need three starting values $U^0$, $U^1$, $U^2$ and we can find $c_1$, $c_2$, $c_3$ in terms of them by solving a linear system similar to (6.27). Each $c_i$ will be a linear combination of $U^0$, $U^1$, $U^2$ and so if $U^\nu(k) \to 0$ as $k \to 0$, then $c_i(k) \to 0$ as $k \to 0$ also. The value $U^N$ computed at time $T$ with step size $k$ (where $kN = T$) has the form

$$U^N = c_1(k) + c_2(k)\left(\frac{1}{2}\right)^N + c_3(k)N\left(\frac{1}{2}\right)^N. \tag{6.33}$$

Now we see that

$$\lim_{\substack{k\to 0\\ Nk=T}} U^N = 0$$

and so the method (6.31) converges on $u' = 0$ with arbitrary starting values $U^\nu(k)$ satisfying $U^\nu(k) \to 0$ as $k \to 0$. (In fact, this LMM is convergent in general.)

More generally, if $\rho(\zeta)$ has a root $\zeta_j$ that is repeated $m$ times, then $U^N$ will involve terms of the form $N^s\zeta_j^N$ for $s = 0, 1, \dots, m-1$. This converges to zero as $N \to \infty$ provided $|\zeta_j| < 1$. The algebraic growth of $N^s$ is overwhelmed by the exponential decay of $\zeta_j^N$. This shows that repeated roots are not a problem as long as they have magnitude strictly less than 1.

With the above examples as motivation, we are ready to state the definition of zero-stability.

**Definition 6.2.** *An $r$-step LMM is said to be zero-stable if the roots of the characteristic polynomial $\rho(\zeta)$ defined by* (5.49) *satisfy the following conditions:*

$$\begin{aligned} &|\zeta_j| \le 1 \quad \textit{for } j = 1, 2, \dots, r. \\ &\textit{If } \zeta_j \textit{ is a repeated root, then } |\zeta_j| < 1. \end{aligned} \tag{6.34}$$

If the conditions (6.34) are satisfied for all roots of $\rho$, then the polynomial is said to satisfy the *root condition*.

**Example 6.6.** The Adams methods have the form

$$U^{n+r} = U^{n+r-1} + k\sum_{j=1}^{r}\beta_j f(U^{n+j})$$

and hence

$$\rho(\zeta) = \zeta^r - \zeta^{r-1} = (\zeta - 1)\zeta^{r-1}.$$

The roots are $\zeta_1 = 1$ and $\zeta_2 = \cdots = \zeta_r = 0$. The root condition is clearly satisfied and all the Adams–Bashforth and Adams–Moulton methods are zero-stable.

The given examples certainly do not prove that zero-stability as defined above is a sufficient condition for convergence. We looked at only the simplest possible ODE $u'(t) = 0$ and saw that things could go wrong if the root condition is *not* satisfied. It turns out, however, that the root condition is all that is needed to prove convergence on the general initial value problem (in the sense of Definition 6.1).

**Theorem 6.3 (Dahlquist [22]).** *For LMMs applied to the initial value problem for $u'(t) = f(u(t), t)$,*

$$\textit{consistency} + \textit{zero-stability} \iff \textit{convergence}. \tag{6.35}$$

This is the analogue of the statement (2.21) for the BVP. A proof of this result can be found in [43].

*Note:* A consistent LMM always has one root equal to 1, say, $\zeta_1 = 1$, called the *principal root*. This follows from (5.50). Hence a consistent one-step LMM (such as Euler, backward Euler, trapezoidal) is certainly zero-stable. More generally we have proved in Section 6.3.4 that any consistent one-step method (that is a Lipschitz continuous) is convergent. Such methods are automatically "zero-stable" and behave well as $k \to 0$. We can think of zero-stability as meaning "stable in the limit as $k \to 0$."

Although a consistent zero-stable method is convergent, it may have other stability problems that show up if the time step $k$ is chosen too large in an actual computation. Additional stability considerations are the subject of the next chapter.

Chapter 7

# Absolute Stability for Ordinary Differential Equations

## 7.1 Unstable computations with a zero-stable method

In the last chapter we investigated zero-stability, the form of stability needed to guarantee convergence of a numerical method as the grid is refined ($k \to 0$). In practice, however, we are not able to compute this limit. Instead we typically perform a single calculation with some particular nonzero time step $k$ (or some particular sequence of time steps with a variable step size method). Since the expense of the computation increases as $k$ decreases, we generally want to choose the time step as large as possible consistent with our accuracy requirements. How can we estimate the size of $k$ required?

Recall that if the method is stable in an appropriate sense, then we expect the global error to be bounded in terms of the local truncation errors at each step, and so we can often use the local truncation error to estimate the time step needed, as illustrated below. But the form of stability now needed is something stronger than zero-stability. We need to know that the error is well behaved for the particular time step we are now using. It is little help to know that things will converge in the limit "for $k$ sufficiently small." The potential difficulties are best illustrated with some examples.

**Example 7.1.** Consider the initial value problem (IVP)

$$u'(t) = -\sin t, \qquad u(0) = 1$$

with solution

$$u(t) = \cos t.$$

Suppose we wish to use Euler's method to solve this problem up to time $T = 2$. The local truncation error (LTE) is

$$\begin{aligned} \tau(t) &= \frac{1}{2}ku''(t) + O(k^2) \\ &= -\frac{1}{2}k\cos(t) + O(k^2). \end{aligned} \tag{7.1}$$

Since the function $f(t) = -\sin t$ is independent of $u$, it is Lipschitz continuous with Lipschitz constant $L = 0$, and so the error estimate (6.12) shows that

$$|E^n| \le T\|\tau\|_\infty = k \max_{0\le t\le T} |\cos t| = k.$$

Suppose we want to compute a solution with $|E| \le 10^{-3}$. Then we should be able to take $k = 10^{-3}$ and obtain a suitable solution after $T/k = 2000$ time steps. Indeed, calculating using $k = 10^{-3}$ gives a computed value $U^{2000} = -0.415692$ with an error $E^{2000} = U^{2000} - \cos(2) = 0.4548 \times 10^{-3}$.

**Example 7.2.** Now suppose we modify the above equation to

$$u'(t) = \lambda(u - \cos t) - \sin t, \tag{7.2}$$

where $\lambda$ is some constant. If we take the same initial data as before, $u(0) = 1$, then the solution is also the same as before, $u(t) = \cos t$. As a concrete example, let's take $\lambda = -10$. Now how small do we need to take $k$ to get an error that is $10^{-3}$? Since the LTE (7.1) depends only on the true solution $u(t)$, which is unchanged from Example 7.1, we might hope that we could use the same $k$ as in that example, $k = 10^{-3}$. Solving the problem using Euler's method with this step size now gives $U^{2000} = -0.416163$ with an error $E^{2000} = 0.161 \times 10^{-4}$. We are again successful. In fact, the error is considerably smaller in this case than in the previous example, for reasons that will become clear later.

**Example 7.3.** Now consider the problem (7.2) with $\lambda = -2100$ and the same data as before. Again the solution is unchanged and so is the LTE. But now if we compute with the same step size as before, $k = 10^{-3}$, we obtain $U^{2000} = -0.2453 \times 10^{77}$ with an error of magnitude $10^{77}$. The computation behaves in an "unstable" manner, with an error that grows exponentially in time. Since the method is zero-stable and $f(u,t)$ is Lipschitz continuous in $u$ (with Lipschitz constant $L = 2100$), we know that the method is convergent, and indeed with sufficiently small time steps we achieve very good results. Table 7.1 shows the error at time $T = 2$ when Euler's method is used with various values of $k$. Clearly something dramatic happens between the values $k = 0.000976$ and $k = 0.000952$. For smaller values of $k$ we get very good results, whereas for larger values of $k$ there is no accuracy whatsoever.

The equation (7.2) is a linear equation of the form (6.3) and so the analysis of Section 6.3.1 applies directly to this problem. From (6.7) we see that the global error $E^n$ satisfies the recursion relation

$$E^{n+1} = (1 + k\lambda)E^n - k\tau^n, \tag{7.3}$$

where the local error $\tau^n = \tau(t_n)$ from (7.1). The expression (7.3) reveals the source of the exponential growth in the error—in each time step the previous error is multiplied by a factor of $(1 + k\lambda)$. For the case $\lambda = -2100$ and $k = 10^{-3}$, we have $1 + k\lambda = -1.1$ and so we expect the local error introduced in step $m$ to grow by a factor of $(-1.1)^{n-m}$ by the end of $n$ steps (recall (6.8)). After 2000 steps we expect the truncation error introduced in the first step to have grown by a factor of roughly $(-1.1)^{2000} \approx 10^{82}$, which is consistent with the error actually seen.

Note that in Example 7.2 with $\lambda = -10$, we have $1 + k\lambda = 0.99$, causing a *decay* in the effect of previous errors in each step. This explains why we got a reasonable result in Example 7.2 and in fact a better result than in Example 7.1, where $1 + k\lambda = 1$.

**Table 7.1.** *Errors in the computed solution using Euler's method for Example 7.3, for different values of the time step $k$. Note the dramatic change in behavior of the error for $k < 0.000952$.*

| $k$ | Error |
|---|---|
| 0.001000 | 0.145252E+77 |
| 0.000976 | 0.588105E+36 |
| 0.000950 | 0.321089E-06 |
| 0.000800 | 0.792298E-07 |
| 0.000400 | 0.396033E-07 |

Returning to the case $\lambda = -2100$, we expect to observe exponential growth in the error for any value of $k$ greater than $2/2100 = 0.00095238$, since for any $k$ larger than this we have $|1 + k\lambda| > 1$. For smaller time steps $|1 + k\lambda| < 1$ and the effect of each local error decays exponentially with time rather than growing. This explains the dramatic change in the behavior of the error that we see as we cross the value $k = 0.00095238$ in Table 7.1.

Note that the exponential growth of errors does not contradict zero-stability or convergence of the method in any way. The method does converge as $k \to 0$. In fact the bound (6.12),

$$|E^n| \le e^{|\lambda|T} T \|\tau\|_\infty = O(k) \quad \text{as } k \to 0,$$

that we used to prove convergence allows the possibility of exponential growth with time. The bound is valid for all $k$, but since $Te^{|\lambda|T} = 2e^{4200} = 10^{1825}$ while $\|\tau\|_\infty = \frac{1}{2}k$, this bound does not guarantee any accuracy whatsoever in the solution until $k < 10^{-1825}$. This is a good example of the fact that a mathematical convergence proof may be a far cry from what is needed in practice.

## 7.2 Absolute stability

To determine whether a numerical method will produce reasonable results with a given value of $k > 0$, we need a notion of stability that is different from zero-stability. There are a wide variety of other forms of "stability" that have been studied in various contexts. The one that is most basic and suggests itself from the above examples is *absolute stability*. This notion is based on the linear test equation (6.3), although a study of the absolute stability of a method yields information that is typically directly useful in determining an appropriate time step in nonlinear problems as well; see Section 7.4.3.

We can look at the simplest case of the test problem in which $g(t) = 0$ and we have simply

$$u'(t) = \lambda u(t).$$

Euler's method applied to this problem gives

$$U^{n+1} = (1 + k\lambda)U^n$$

and we say that this method is *absolutely stable* when $|1 + k\lambda| \le 1$; otherwise it is unstable. Note that there are two parameters $k$ and $\lambda$, but only their product $z \equiv k\lambda$ matters. The method is stable whenever $-2 \le z \le 0$, and we say that the *interval of absolute stability* for Euler's method is $[-2, 0]$.

It is more common to speak of the *region of absolute stability* as a region in the complex $z$ plane, allowing the possibility that $\lambda$ is complex (of course the time step $k$ should be real and positive). The region of absolute stability (or simply the *stability region*) for Euler's method is the disk of radius 1 centered at the point $-1$, since within this disk we have $|1 + k\lambda| \le 1$ (see Figure 7.1a). Allowing $\lambda$ to be complex comes from the fact that in practice we are usually solving a system of ordinary differential equations (ODEs). In the linear case it is the eigenvalues of the coefficient matrix that are important in determining stability. In the nonlinear case we typically linearize (see Section 7.4.3) and consider the eigenvalues of the Jacobian matrix. Hence $\lambda$ represents a typical eigenvalue and these may be complex even if the matrix is real. For some problems, looking at the eigenvalues is not sufficient (see Section 10.12.1, for example), but eigenanalysis is generally very revealing.

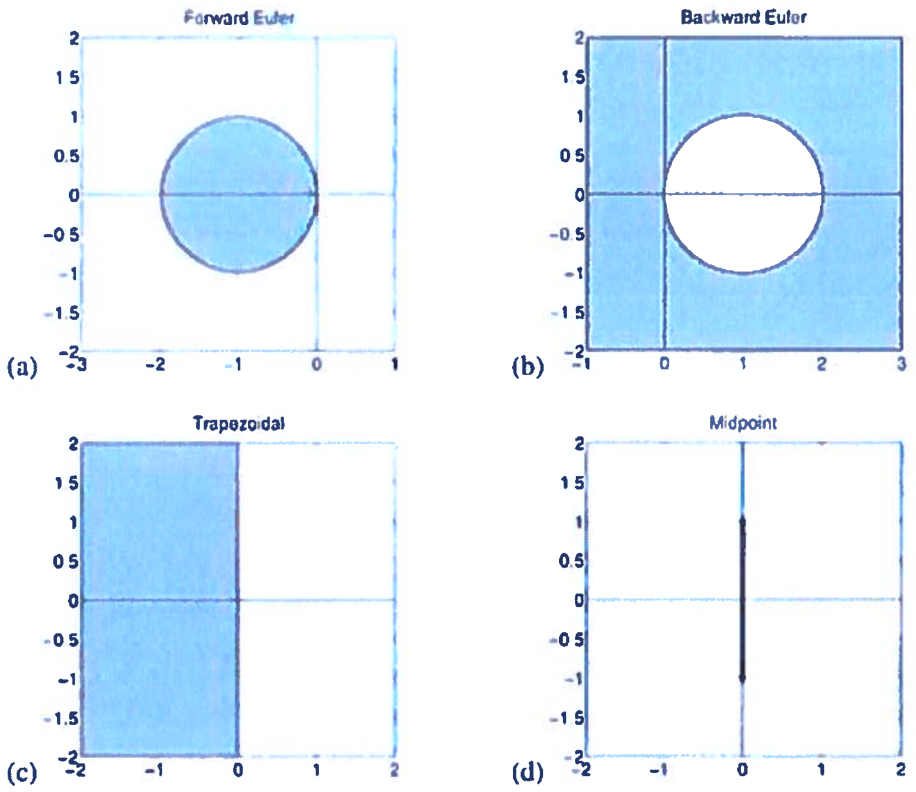

**Figure 7.1.** *Stability regions for* (a) *Euler,* (b) *backward Euler,* (c) *trapezoidal, and* (d) *midpoint (a segment on imaginary axis).*

## 7.3 Stability regions for linear multistep methods

For a general linear multistep method (LMM) of the form (5.44), the region of absolute stability is found by applying the method to $u' = \lambda u$, obtaining

$$\sum_{j=0}^{r} \alpha_j U^{n+j} = k \sum_{j=0}^{r} \beta_j \lambda U^{n+j},$$

which can be rewritten as

$$\sum_{j=0}^{r} (\alpha_j - z\beta_j) U^{n+j} = 0. \tag{7.4}$$

Note again that it is only the product $z = k\lambda$ that is important, not the values of $k$ or $\lambda$ separately, and that this is a dimensionless quantity since the decay rate $\lambda$ has dimensions time$^{-1}$, while the time step has dimensions of time. This makes sense—if we change the units of time (say, from seconds to milliseconds), then the parameter $\lambda$ will decrease by a factor of 1000 and we may be able to increase the numerical value of $k$ by a factor of 1000 and still be stable. But then we also have to solve out to time $1000T$ instead of to time $T$, so we haven't really changed the numerical problem or the number of time steps required.

The recurrence (7.4) is a homogeneous linear difference equation of the same form considered in Section 6.4.1. The solution has the general form (6.26), where the $\zeta_j$ are now the roots of the characteristic polynomial $\sum_{j=0}^{r}(\alpha_j - z\beta_j)\zeta^j$. This polynomial is often called the *stability polynomial* and denoted by $\pi(\zeta; z)$. It is a polynomial in $\zeta$ but its coefficients depend on the value of $z$. The stability polynomial can be expressed in terms of the characteristic polynomials for the LMM as

$$\pi(\zeta; z) = \rho(\zeta) - z\sigma(\zeta). \tag{7.5}$$

The LMM is absolutely stable for a particular value of $z$ if errors introduced in one time step do not grow in future time steps. According to the theory of Section 6.4.1, this requires that the polynomial $\pi(\zeta; z)$ satisfy the root condition (6.34).

**Definition 7.1.** *The* region of absolute stability *for the LMM* (5.44) *is the set of points $z$ in the complex plane for which the polynomial $\pi(\zeta; z)$ satisfies the root condition* (6.34).

Note that an LMM is zero-stable if and only if the origin $z = 0$ lies in the stability region.

**Example 7.4.** For Euler's method,

$$\pi(\zeta; z) = \zeta - (1 + z)$$

with the single root $\zeta_1 = 1 + z$. We have already seen that the stability region is the disk in Figure 7.1(a).

**Example 7.5.** For the backward Euler method (5.21),

$$\pi(\zeta; z) = (1 - z)\zeta - 1$$

with root $\zeta_1 = (1-z)^{-1}$. We have

$$|(1-z)^{-1}| \le 1 \iff |1-z| \ge 1$$

so the stability region is the *exterior* of the disk of radius 1 centered at $z = 1$, as shown in Figure 7.1(b).

**Example 7.6.** For the trapezoidal method (5.22),

$$\pi(\zeta; z) = \left(1 - \frac{1}{2}z\right)\zeta - \left(1 + \frac{1}{2}z\right)$$

with root

$$\zeta_1 = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}.$$

This is a linear fractional transformation and it can be shown that

$$|\zeta_1| \le 1 \iff \text{Re}(z) \le 0,$$

where Re($z$) is the real part. So the stability region is the left half-plane as shown in Figure 7.1(c).

**Example 7.7.** For the midpoint method (5.23),

$$\pi(\zeta; z) = \zeta^2 - 2z\zeta - 1.$$

The roots are $\zeta_{1,2} = z \pm \sqrt{z^2+1}$. It can be shown that if $z$ is a pure imaginary number of the form $z = i\alpha$ with $|\alpha| < 1$, then $|\zeta_1| = |\zeta_2| = 1$ and $\zeta_1 \ne \zeta_2$, and hence the root condition is satisfied. For any other $z$ the root condition is not satisfied. In particular, if $z = \pm i$, then $\zeta_1 = \zeta_2$ is a repeated root of modulus 1. So the stability region consists only of the open interval from $-i$ to $i$ on the imaginary axis, as shown in Figure 7.1(d).

Since $k$ is always real, this means the midpoint method is useful only on the test problem $u' = \lambda u$ if $\lambda$ is pure imaginary. The method is not very useful for scalar problems where $\lambda$ is typically real, but the method is of great interest in some applications with systems of equations. For example, if the matrix is real but skew symmetric ($A^T = -A$), then the eigenvalues are pure imaginary. This situation arises naturally in the discretization of hyperbolic partial differential equations (PDEs), as discussed in Chapter 10.

**Example 7.8.** Figures 7.2 and 7.3 show the stability regions for the $r$-step Adams–Bashforth and Adams–Moulton methods for various values of $r$. For an $r$-step method the polynomial $\pi(\zeta; z)$ has degree $r$ and there are $r$ roots. Determining the values of $z$ for which the root condition is satisfied does not appear simple. However, there is a simple technique called the *boundary locus method* that makes it possible to determine the regions shown in the figures. This is briefly described in Section 7.6.1.

Note that for many methods the shape of the stability region near the origin $z = 0$ is directly related to the accuracy of the method. Recall that the stability polynomial $\rho(\zeta)$ for a consistent LMM always has a principal root $\zeta_1 = 1$. It can be shown that for $z$ near 0 the polynomial $\pi(\zeta; z)$ has a corresponding principal root with behavior

$$\zeta_1(z) = e^z + O(z^{p+1}) \quad \text{as } z \to 0 \tag{7.6}$$

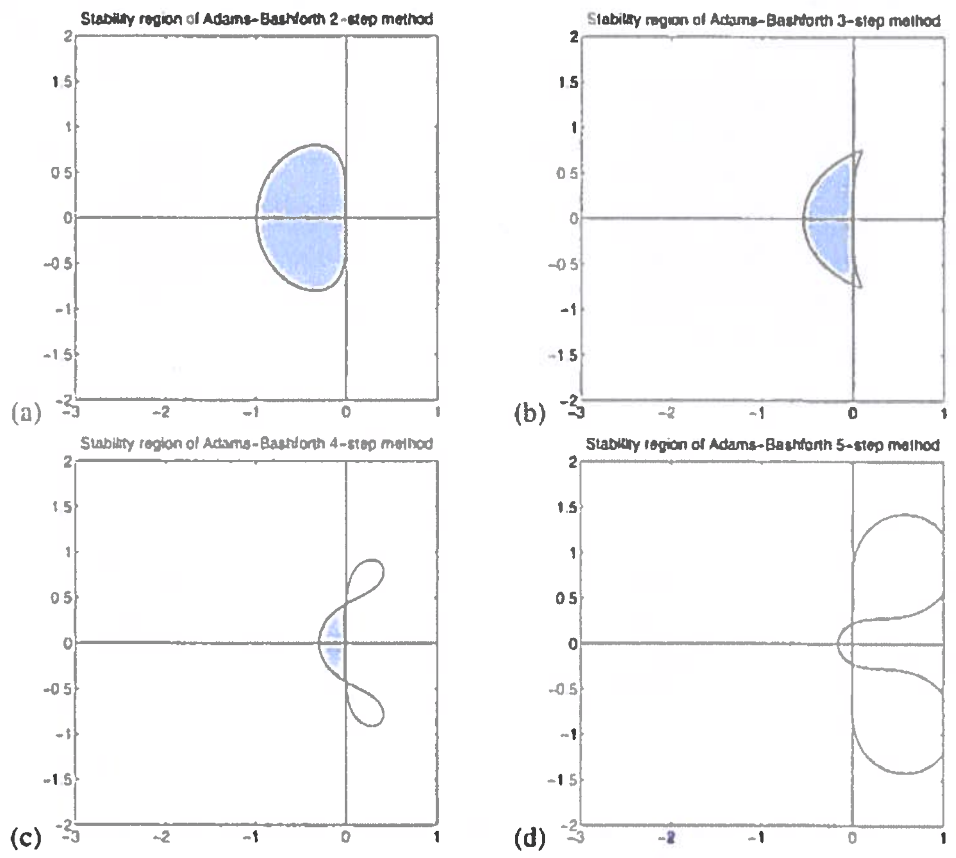

**Figure 7.2.** *Stability regions for some Adams-Bashforth methods. The shaded region just to the left of the origin is the region of absolute stability. See* Section 7.6.1 *for a discussion of the other loops seen in figures (c) and (d).*

if the method is $p$th order accurate. We can see this in the examples above for one-step methods, e.g., for Euler's method $\zeta_1(z) = 1 + z = e^z + O(z^2)$. It is this root that is giving the appropriate behavior $U^{n+1} \approx e^z U^n$ over a time step. Since this root is on the unit circle at the origin $z = 0$, and since $|e^z| < 1$ only when $\text{Re}(z) < 0$, we expect the principal root to move inside the unit circle for small $z$ with $\text{Re}(z) < 0$ and outside the unit circle for small $z$ with $\text{Re}(z) > 0$. This suggests that if we draw a small circle around the origin, then the left half of this circle will lie inside the stability region (unless some other root moves outside, as happens for the midpoint method), while the right half of the circle will lie outside the stability region. Looking at the stability regions in Figure 7.1 we see that this is indeed true for all the methods except the midpoint method. Moreover, the higher the order of accuracy in general, the larger a circle around the origin where this will approximately hold, and so the boundary of the stability region tends to align with the imaginary axis farther and farther from the origin as the order of the method increases, as observed in Figures 7.2 and 7.3. (The trapezoidal method is a bit of an anomaly, as its stability region exactly agrees with that of $e^z$ for all $z$.)

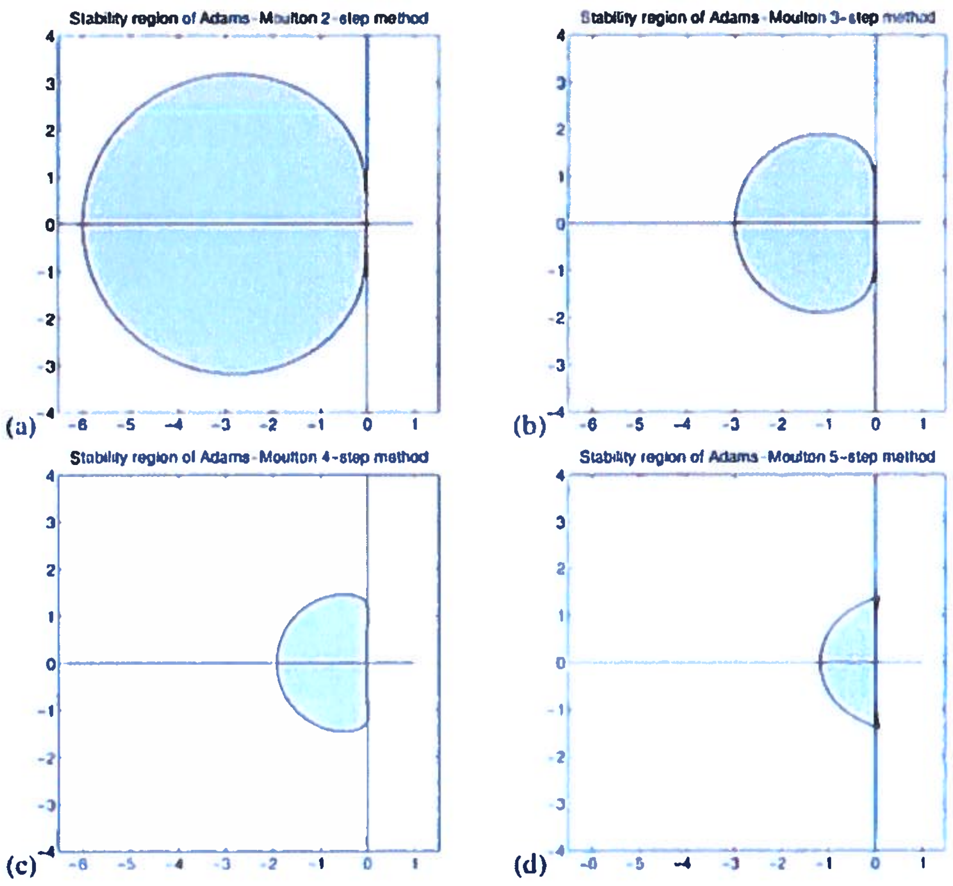

**Figure 7.3.** *Stability regions for some Adams–Moulton methods.*

See Section 7.6 for a discussion of ways in which stability regions can be determined and plotted.

## 7.4 Systems of ordinary differential equations

So far we have examined stability theory only in the context of a scalar differential equation $u'(t) = f(u(t))$ for a scalar function $u(t)$. In this section we will look at how this stability theory carries over to systems of $m$ differential equations where $u(t) \in \mathbb{R}^m$. For a linear system $u' = Au$, where $A$ is an $m \times m$ matrix, the solution can be written as $u(t) = e^{At}u(0)$ and the behavior is largely governed by the eigenvalues of $A$. A necessary condition for stability is that $k\lambda$ be in the stability region for each eigenvalue $\lambda$ of $A$. For general nonlinear systems $u' = f(u)$, the theory is more complicated, but a good rule of thumb is that $k\lambda$ should be in the stability region for each eigenvalue $\lambda$ of the Jacobian matrix $f'(u)$. This may not be true if the Jacobian is rapidly changing with time, or even for constant coefficient linear problems in some highly nonnormal cases (see [47] and Section 10.12.1 for an example), but most of the time eigenanalysis is surprisingly effective.

## 7.6 Plotting stability regions

### 7.6.1 The boundary locus method for linear multistep methods

A point $z \in \mathbb{C}$ is in the stability region $S$ of an LMM if the stability polynomial $\pi(\zeta; z)$ satisfies the root condition for this value of $z$. It follows that if $z$ is on the *boundary* of the stability region, then $\pi(\zeta; z)$ must have at least one root $\zeta_j$ with magnitude exactly equal to 1. This $\zeta_j$ is of the form

$$\zeta_j = e^{i\theta}$$

for some value of $\theta$ in the interval $[0, 2\pi]$. (Beware of the two different uses of $\pi$.) Since $\zeta_j$ is a root of $\pi(\zeta; z)$, we have

$$\pi(e^{i\theta}; z) = 0$$

for this particular combination of $z$ and $\theta$. Recalling the definition of $\pi$, this gives

$$\rho(e^{i\theta}) - z\sigma(e^{i\theta}) = 0 \tag{7.12}$$

and hence

$$z = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}.$$

If we know $\theta$, then we can find $z$ from this.

Since every point $z$ on the boundary of $S$ must be of this form for some value of $\theta$ in $[0, 2\pi]$, we can simply plot the parametrized curve

$$\tilde{z}(\theta) \equiv \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})} \tag{7.13}$$

for $0 \le \theta \le 2\pi$ to find the locus of all points which are *potentially* on the boundary of $S$. For simple methods this yields the region $S$ directly.

**Example 7.13.** For Euler's method we have $\rho(\zeta) = \zeta - 1$ and $\sigma(\zeta) = 1$, and so

$$\tilde{z}(\theta) = e^{i\theta} - 1.$$

This function maps $[0, 2\pi]$ to the unit circle centered at $z = -1$, which is exactly the boundary of $S$ as shown in Figure 7.1(a).

To determine which side of this curve is the *interior* of $S$, we need only evaluate the roots of $\pi(\zeta; z)$ at some random point $z$ on one side or the other and see if the polynomial satisfies the root condition.

Alternatively, as noted on page 155, most methods are stable just to the left of the origin on the negative real axis and unstable just to the right of the origin on the positive real axis. This is often enough information to determine where the stability region lies relative to the boundary locus.

For some methods the boundary locus may cross itself. In this case we typically find that at most one of the regions cut out of the plane corresponds to the stability region. We can determine which region is $S$ by evaluating the roots at some convenient point $z$ within each region.

**Example 7.14.** The five-step Adams–Bashforth method gives the boundary locus seen in Figure 7.2(d). The stability region is the small semicircular region to the left of the

From the definition of absolute stability given at the beginning of this chapter, we see that the region of absolute stability for a one-step method is simply

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \le 1\}. \tag{7.20}$$

This follows from the fact that iterating a one-step method on $u' = \lambda u$ gives $|U^n| = |R(z)|^n |U^0|$ and this will be uniformly bounded in $n$ if $z$ lies in $\mathcal{S}$.

One way to attempt to compute $\mathcal{S}$ would be to compute the boundary locus as described in Section 7.6.1 by setting $R(z) = e^{i\theta}$ and solving for $z$ as $\theta$ varies. This would give the set of $z$ for which $|R(z)| = 1$, the boundary of $\mathcal{S}$. There's a problem with this, however: when $R(z)$ is a higher order polynomial or rational function there will be several solutions $z$ for each $\theta$ and it is not clear how to connect these to generate the proper curve.

Another approach can be taken graphically that is more brute force, but effective. If we have a reasonable idea of what region of the complex $z$-plane contains the boundary of $\mathcal{S}$, we can sample $|R(z)|$ on a fine grid of points in this region and approximate the level set where this function has the value 1 and plot this as the boundary of $\mathcal{S}$. This is easily done with a contour plotter, for example, using the `contour` command in MATLAB. Or we can simply color each point depending on whether it is inside $\mathcal{S}$ or outside.

For example, Figure 7.5 shows the stability regions for the Taylor series methods of orders 2 and 4, for which

$$\begin{aligned} R(z) &= 1 + z + \frac{1}{2}z^2, \\ R(z) &= 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4, \end{aligned} \tag{7.21}$$

respectively. These are also the stability regions of the second order Runge–Kutta method (5.30) and the fourth order accurate Runge–Kutta method (5.33), which are easily seen to have the same stability functions.

Note that for a one-step method of order $p$, the rational function $R(z)$ must agree with $e^z$ to $O(z^{p+1})$. As for LMMs, we thus expect that points very close to the origin will lie in the stability region $\mathcal{S}$ for $\mathrm{Re}(z) < 0$ and outside of $\mathcal{S}$ for $\mathrm{Re}(z) > 0$.

## 7.7 Relative stability regions and order stars

Recall that for a one-step method the stability region $\mathcal{S}$ (more properly called the region of absolute stability) is the region $\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \le 1\}$, where $U^{n+1} = R(z)U^n$ is the relation between $U^n$ and $U^{n+1}$ when the method is applied to the test problem $u' = \lambda u$. For $z = \lambda k$ in the stability region the numerical solution does not grow, and hence the method is absolutely stable in the sense that past errors will not grow in later time steps.

On the other hand, the true solution to this problem, $u(t) = e^{\lambda t}u(0)$, is itself exponentially growing or decaying. One might argue that if $u(t)$ is itself decaying, then it isn't good enough to simply have the past errors decaying, too—they should be decaying at a faster rate. Or conversely, if the true solution is growing exponentially, then perhaps it is fine for the error also to be growing, as long as it is not growing faster.

This suggests defining the *region of relative stability* as the set of $z \in \mathbb{C}$ for which $|R(z)| \le |e^z|$. In fact this idea has not proved to be particularly useful in terms of judging