Chapter 8

Stiff Ordinary Differential Equations

The problem of *stiffness* leads to computational difficulty in many practical problems. The classic example is the case of a stiff ordinary differential equation (ODE), which we will examine in this chapter. In general a problem is called *stiff* if, roughly speaking, we are attempting to compute a particular solution that is smooth and slowly varying (relative to the time interval of the computation), but in a context where the nearby solution curves are much more rapidly varying. In other words, if we perturb the solution slightly at any time, the resulting solution curve through the perturbed data has rapid variation. Typically this takes the form of a short-lived "transient" response that moves the solution back toward a smooth solution.

Example 8.1. Consider the ODE (7.2) from the previous chapter,

$$u'(t) = \lambda(u - \cos t) - \sin t. \tag{8.1}$$

One particular solution is the function $u(t) = \cos t$, and this is the solution with the initial data $u(0) = 1$ considered previously. This smooth function is a solution for any value of λ. If we consider initial data of the form $u(t_0) = \eta$ that does not lie on this curve, then the solution through this point is a different function, of course. However, if $\lambda < 0$ (or $\mathrm{Re}(\lambda) < 0$ more generally), this function approaches $\cos t$ exponentially quickly, with decay rate λ. It is easy to verify that the solution is

$$u(t) = e^{\lambda(t-t_0)}(\eta - \cos(t_0)) + \cos t. \tag{8.2}$$

Figure 8.1 shows a number of different solution curves for this equation with different choices of t_0 and η, with the fairly modest value $\lambda = -1$. Figure 8.1b shows the corresponding solution curves when $\lambda = -10$.

In this scalar example, when we perturb the solution at some point it quickly relaxes toward the particular solution $u(t) = \cos t$. In other stiff problems the solution might move quickly toward some different smooth solution, as seen in the next example.

Example 8.2. Consider the kinetics model $A \to B \to C$ developed in Example 7.9. The system of equations is given by (7.10). Suppose that $K_1 \gg K_2$ so that a typical solution appears as in Figure 8.2(a). (Here $K_1 = 20$ and $K_2 = 1$. Compare this to

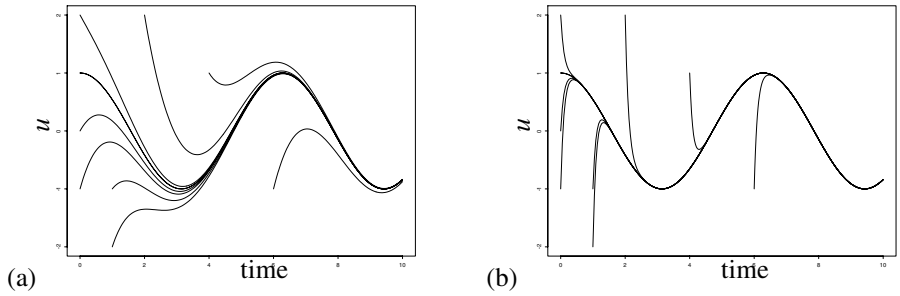

Figure 8.1. *Solution curves for the ODE* (8.1) *for various initial values.* (a) *With* $\lambda = -1$. (b) *With* $\lambda = -10$ *and the same set of initial values.*

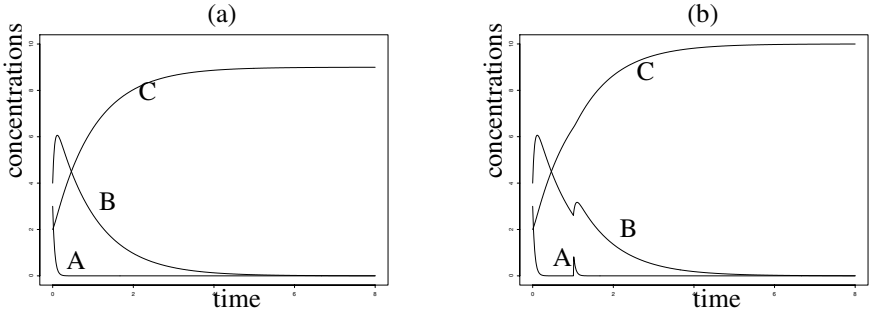

Figure 8.2. *Solution curves for the kinetics problem in Example* 7.9 *with* $K_1 = 20$ *and* $K_2 = 1$. *In* (b) *a perturbation has been made by adding one unit of species* A *at time* $t = 1$. *Figure 8.3 shows similar solutions for the case* $K_1 = 10^6$.

Figure 7.4.) Now suppose at time $t = 1$ we perturb the system by adding more of species A. Then the solution behaves as shown in Figure 8.2(b). The additional A introduced is rapidly converted into B (the fast transient response) and then slowly from B into C. After the rapid transient the solution is again smooth, although it differs from the original solution since the final asymptotic value of C must be higher than before by the same magnitude as the amount of A introduced.

8.1 Numerical difficulties

Stiffness causes numerical difficulties because any finite difference method is constantly introducing errors. The local truncation error acts as a perturbation to the system that moves us away from the smooth solution we are trying to compute. Why does this cause more difficulty in a stiff system than in other systems? At first glance it seems like the stiffness might work to our advantage. If we are trying to compute the solution $u(t) = \cos t$ to the ODE (8.1) with initial data $u(0) = 1$, for example, then the fact that any errors introduced decay exponentially should help us. The true solution is very robust and the solution is almost completely insensitive to errors made in the past. In fact, this *stability* of the true

solution does help us, as long as the numerical method is also stable. (Recall that the results in Example 7.2 were much better than those in Example 7.1.)

The difficulty arises from the fact that many numerical methods, including all explicit methods, are unstable (in the sense of absolute stability) *unless the time step is small relative to the time scale of the rapid transient*, which in a stiff problem is much smaller than the time scale of the smooth solution we are trying to compute. In the terminology of Section 7.5, this means that $k_{\text{stab}} \ll k_{\text{acc}}$. Although the true solution is smooth and it seems that a reasonably large time step would be appropriate, the numerical method must always deal with the rapid transients introduced by truncation error in every time step and may need a very small time step to do so stably.

8.2 Characterizations of stiffness

A stiff ODE can be characterized by the property that $f'(u)$ is much larger (in absolute value or norm) than $u'(t)$. The latter quantity measures the smoothness of the solution being computed, while $f'(u)$ measures how rapidly f varies as we move away from this particular solution. Note that stiff problems typically have large Lipschitz constants too.

For *systems* of ODEs, stiffness is sometimes defined in terms of the "stiffness ratio" of the system, which is the ratio

$$\frac{\max |\lambda_p|}{\min |\lambda_p|} \tag{8.3}$$

over all eigenvalues of the Jacobian matrix $f'(u)$. If this is large, then a large range of time scales is present in the problem, a necessary component for stiffness to arise. While this is often a useful quantity, one should not rely entirely on this measure to determine whether a problem is stiff.

For one thing, it is possible even for a scalar problem to be stiff (as we have seen in Example 8.1), although for a scalar problem the stiffness ratio is always 1 since there is only one eigenvalue. Still, more than one time scale can be present. In (8.1) the fast time scale is determined by λ, the eigenvalue, and the slow time scale is determined by the inhomogeneous term $\sin(t)$. For systems of equations there also may be additional time scales arising from inhomogeneous forcing terms or other time-dependent coefficients that are distinct from the scales imposed by the eigenvalues.

We also know that for highly nonnormal matrices the eigenvalues don't always tell the full story (see Section D.4). Often they give adequate guidance, but there are examples where the problem is more stiff than (8.3) would suggest. An example arises when certain spectral approximations to spatial derivatives are used in discretizing hyperbolic equations; see Section 10.13.

On the other hand, it is also important to note that a system of ODEs which has a large "stiffness ratio" is not necessarily stiff! If the eigenvalue with large amplitude lies close to the imaginary axis, then it leads to highly oscillatory behavior in the solution rather than rapid damping. If the solution is rapidly oscillating, then it will probably be necessary to take small time steps for accuracy reasons and k_{acc} may be roughly the same magnitude as k_{stab} even for explicit methods, at least with the sort of methods discussed here. (Special methods for highly oscillatory problems have been developed that allow one to take larger time steps; see, e.g., [50], [74].)

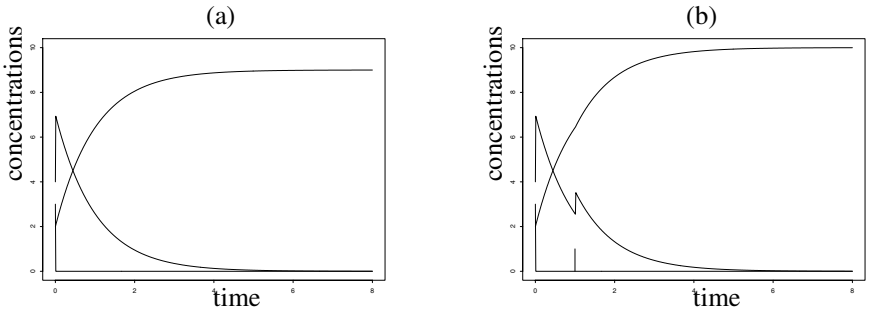

Figure 8.3. *Solution curves for the kinetics problem in Example* 7.9 *with* $K_1 = 10^6$ *and* $K_2 = 1$. *In* (b) *a perturbation has been made by adding one unit of species A at time* $t = 1$.

Finally, note that a particular problem may be stiff over some time intervals and nonstiff elsewhere. In particular, if we are computing a solution that has a rapid transient, such as the kinetics problem shown in Figure 8.3(a), then the problem is not stiff over the initial transient period where the true solution is as rapidly varying as nearby solution curves. Only for times greater than 10^{-6} or so does the problem become stiff, once the desired solution curve is much smoother than nearby curves.

For the problem shown in Figure 8.3(b), there is another time interval just after $t = 1$ over which the problem is again not stiff since the solution again exhibits rapid transient behavior and a small time step would be needed on the basis of accuracy considerations.

8.3 Numerical methods for stiff problems

Over time intervals where a problem is stiff, we would like to use a numerical method that has a large region of absolute stability, extending far into the left half-plane. The problem with a method like Euler's, with a stability region that extends only out to $\mathrm{Re}(\lambda) = -2$, is that the time step k is severely limited by the eigenvalue with largest magnitude, and we need to take $k \approx 2/|\lambda_{\max}|$. Over time intervals where this fastest time scale does not appear in the solution, we would like to be able to take much larger time steps. For example, in the problems shown in Figure 8.3, where $K_1 = 10^6$, we would need to take $k \approx 2 \times 10^{-6}$ with Euler's method, requiring 4 million time steps to compute over the time interval shown in the figure, although the solution is very smooth over most of this time.

An analysis of stability regions shows that there are basically two different classes of methods: those for which the stability region is bounded and extends a distance $O(1)$ from the origin, such as Euler's method, the midpoint method, or any of the Adams methods (see Figures 7.1 and 8.5), and those for which the stability region is unbounded, such as backward Euler or trapezoidal. Clearly the first class of methods is inappropriate for stiff problems.

Unfortunately, all explicit methods have bounded stability regions and hence are generally quite inefficient on stiff problems. An exception is methods such as the Runge–Kutta–Chebyshev methods described in Section 8.6 that may work well for mildly stiff problems. Some implicit methods also have bounded stability regions, such as the Adams–Moulton methods, and are not useful for stiff problems.

8.3.1 A-stability and A(α)-stability

It seems like it would be optimal to have a method whose stability region contains the entire left half-plane. Then any time step would be allowed, provided that all the eigenvalues have negative real parts, as is often the case in practice. The backward Euler and trapezoidal methods have this property, for example. We give methods with this property a special name, as follows.

Definition 8.1. *An ODE method is said to be* A-stable *if its region of absolute stability $\mathcal{S}$ contains the entire left half-plane* $\{z \in \mathbb{C} : Re(z) \leq 0\}$.

For LMMs it turns out that this is quite restrictive. A theorem of Dahlquist [21] (the paper in which the term *A-stability* was introduced) states that any A-stable LMM is at most second order accurate, and in fact the trapezoidal method is the A-stable method with smallest truncation error. This is *Dahlquist's second barrier theorem*, proved, for example, in [44] and by using order stars in [51].

Higher order A-stable implicit Runge–Kutta methods do exist, including diagonally implicit (DIRK) methods; see, for example, [13], [44].

For many stiff problems the eigenvalues are far out in the left half-plane but near (or even exactly on) the real axis. For such problems there is no reason to require that the entire left half-plane lie in the region of absolute stability. If $\arg(z)$ represents the argument of z with $\arg(z) = \pi$ on the negative real axis, and if the wedge $\pi - \alpha \leq \arg(z) \leq \pi + \alpha$ is contained in the stability region, then we say the method is A(α)-stable. An A-stable method is A($\pi/2$)-stable. A method is A(0)-stable if the negative real axis itself lies in the stability region. Note that in general it makes sense to require a wedge to lie in the stability regions, since adjusting the time step k causes $z = k\lambda$ to move in toward the origin on a ray through each eigenvalue.

8.3.2 L-stability

Notice a major difference between the stability regions for trapezoidal and backward Euler: the trapezoidal method is stable only in the left half-plane, whereas backward Euler is also stable over much of the right half-plane. The point at infinity (if we view these stability regions on the Riemann sphere) lies on the boundary of the stability region for the trapezoidal method but in the interior of the stability region for backward Euler.

These are both one-step methods and so on the test problem $u' = \lambda u$ we have $U^{n+1} = R(z)U^n$, where

$$R(z) = \frac{1}{(1-z)} \quad \text{and } |R(z)| \to 0 \text{ as } |z| \to \infty$$

for backward Euler, while

$$R(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z} \quad \text{and } |R(z)| \to 1 \text{ as } |z| \to \infty$$

for the trapezoidal method.

This difference can have a significant effect on the quality of solutions in some situations, in particular if there are rapid transients in the solution that we are not interested in resolving accurately with very small time steps. For these transients we want more than just stability—we want them to be effectively damped in a single time step since we are planning to use a time step that is much larger than the true decay time of the transient. For this purpose a method like backward Euler will perform better than the trapezoidal method. The backward Euler method is said to be L-stable.

Definition 8.2. *A one-step method is* L-stable *if it is A-stable and* $\lim_{z\to\infty} |R(z)| = 0$, *where the stability function* $R(z)$ *is defined in* Section 7.6.2.

The value of L-stability is best illustrated with an example.

Example 8.3. Consider the problem (8.1) with $\lambda = -10^6$. We will see how the trapezoidal and backward Euler methods behave in two different situations.

Case 1: Take data $u(0) = 1$, so that $u(t) = \cos(t)$ and there is no initial transient. Then both trapezoidal and backward Euler behave reasonably and the trapezoidal method gives smaller errors since it is second order accurate. Table 8.1 shows the errors at $T = 3$ with various values of k.

Case 2: Now take data $u(0) = 1.5$ so there is an initial rapid transient toward $u = \cos(t)$ on a time scale of about 10^{-6}. Both methods are still absolutely stable, but the results in Table 8.1 show that backward Euler works much better in this case.

To understand what is happening, see Figure 8.4, which shows the true and computed solutions with each method if we use $k = 0.1$. The trapezoidal method is stable and the results stay bounded, but since $k\lambda = -10^5$ we have $\frac{1-\frac{1}{2}k\lambda}{1+\frac{1}{2}k\lambda} = -.99996 \approx -1$ and the initial deviation from the smooth curve $\cos(t)$ is essentially negated in each time step.

Backward Euler, on the other hand, damps the deviation very effectively in the first time step, since $(1 + k\lambda)^{-1} \approx -10^{-6}$. This is the proper behavior since the true rapid transient decays in a period much shorter than a single time step.

If we are solving a stiff equation with initial data for which the solution is smooth from the beginning (no rapid transients), or if we plan to compute rapid transients accurately by taking suitably small time steps in these regions, then it may be fine to use a method such as the trapezoidal method that is not L-stable.

Table 8.1. *Errors at time* $T = 3$ *for Example* 8.3.

	k	Backward Euler	Trapezoidal
Case 1	0.4	4.7770e−02	4.7770e−02
	0.2	9.7731e−08	4.7229e−10
	0.1	4.9223e−08	1.1772e−10
Case 2	0.4	4.7770e−02	4.5219e−01
	0.2	9.7731e−08	4.9985e−01
	0.1	4.9223e−08	4.9940e−01

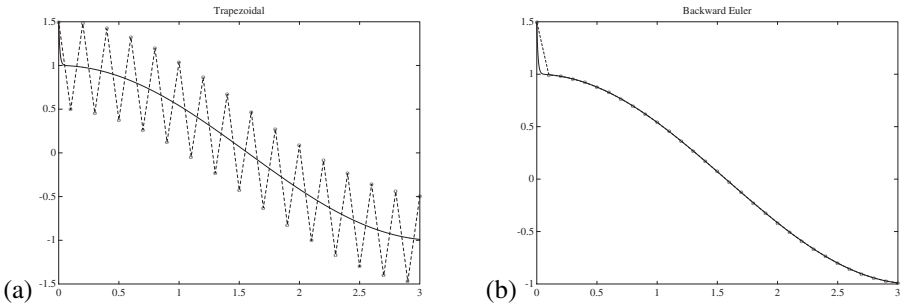

Figure 8.4. *Comparison of* (a) *trapezoidal method and* (b) *backward Euler on a stiff problem with an initial transient (Case* 2 *of Example* 8.3*).*

8.4 BDF methods

One class of very effective methods for stiff problems consists of the backward differentiation formula (BDF) methods. These were introduced by Curtiss and Hirschfelder [20]. See e.g., [33], [44], or [59] for more about these methods.

These methods result from taking $\sigma(\zeta) = \beta_r \zeta^r$, which has all its roots at the origin, so that the point at infinity is in the interior of the stability region. The method thus has the form

$$\alpha_0 U^n + \alpha_1 U^{n+1} + \cdots + \alpha_r U^{n+r} = k\beta_r f(U^{n+r}) \tag{8.4}$$

with $\beta_0 = \beta_1 = \cdots = \beta_{r-1} = 0$. Since $f(u) = u'$, this form of method can be derived by approximating $u'(t_{n+r})$ by a backward difference approximation based on $u(t_{n+r})$ and r additional points going backward in time.

It is possible to derive an r-step method that is rth order accurate. The one-step BDF method is simply the backward Euler method, $U^{n+1} = U^n + kf(U^{n+1})$, which is first order accurate. The other useful BDF methods are below:

$$\begin{aligned}
r = 2: \quad & 3U^{n+2} - 4U^{n+1} + U^n &= 2kf(U^{n+2}) \\
r = 3: \quad & 11U^{n+3} - 18U^{n+2} + 9U^{n+1} - 2U^n &= 6kf(U^{n+3}) \\
r = 4: \quad & 25U^{n+4} - 48U^{n+3} + 36U^{n+2} - 16U^{n+1} + 3U^n &= 12kf(U^{n+4}) \\
r = 5: \quad & 137U^{n+5} - 300U^{n+4} + 300U^{n+3} - 200U^{n+2} \\
& \qquad + 75U^{n+1} - 12U^n &= 60kf(U^{n+5}) \\
r = 6: \quad & 147U^{n+6} - 360U^{n+5} + 450U^{n+4} - 400U^{n+3} \\
& \qquad + 225U^{n+2} - 72U^{n+1} + 10U^n &= 60kf(U^{n+6})
\end{aligned}$$

These methods have the proper behavior on eigenvalues for which $\mathrm{Re}(\lambda)$ is very negative, but of course we also have other eigenvalues for which $z = k\lambda$ is closer to the origin, corresponding to the active time scales in the problem. So deciding its suitability for a particular problem requires looking at the full stability region of a method. This is shown in Figure 8.5 for each of the BDF methods.

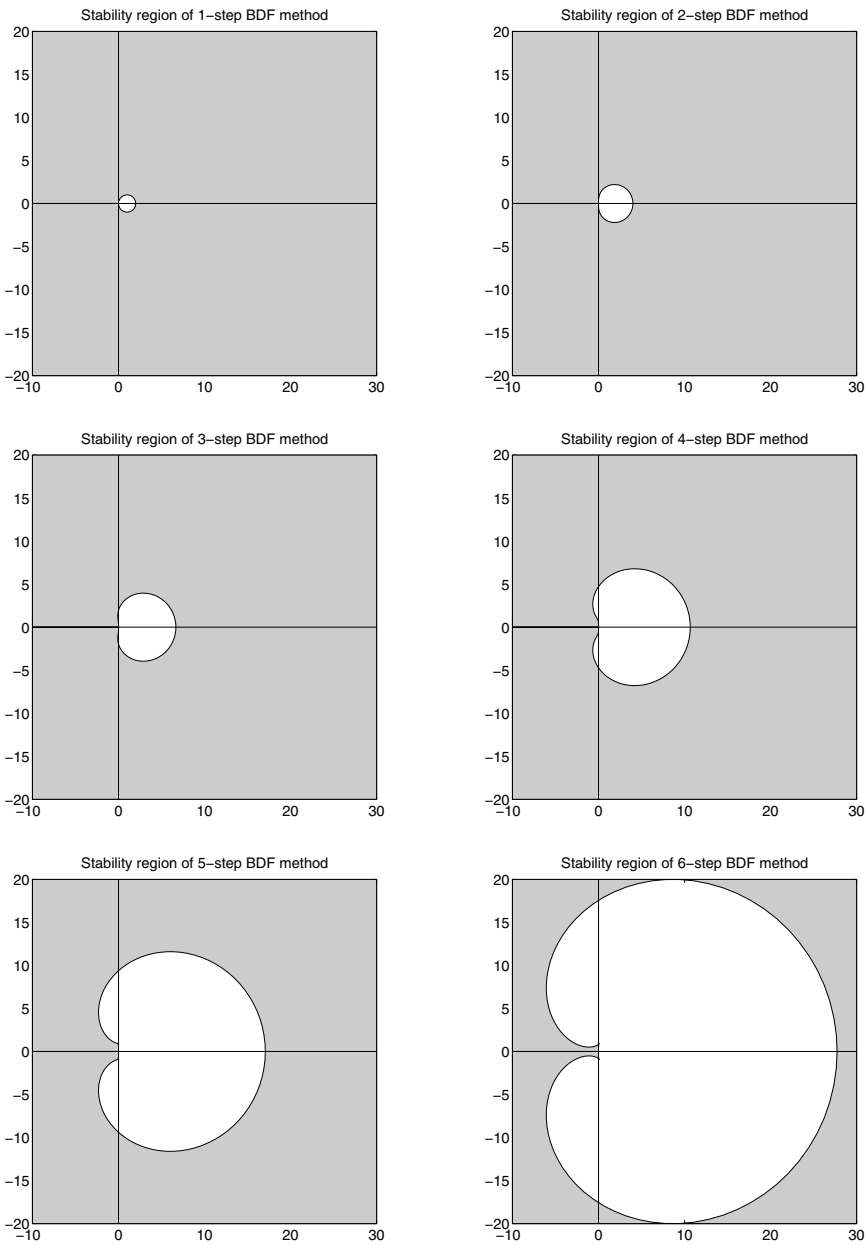

Figure 8.5. *Stability regions for the BDF methods. The stability region is the shaded region* exterior *to the curves.*

In particular, we need to make sure that the method is zero-stable. Otherwise it would not be convergent. This is not guaranteed from our derivation of the methods, since zero-stability depends only on the polynomial $\rho(\zeta)$, whose coefficients α_j are determined by considering the local truncation error and not stability considerations. It turns out that the BDF methods are zero-stable only for $r \leq 6$. Higher order BDF methods cannot be used in practice. For $r > 2$ the methods are not A-stable but are A(α)-stable for the following values of α:

$$\begin{array}{ll} r = 1: \ \alpha = 90^\circ, & r = 4: \ \alpha = 73^\circ \\ r = 2: \ \alpha = 90^\circ, & r = 5: \ \alpha = 51^\circ, \\ r = 3: \ \alpha = 88^\circ, & r = 6: \ \alpha = 18^\circ. \end{array} \tag{8.5}$$

8.5 The TR-BDF2 method

There are often situations in which it is useful to have a one-step method that is L-stable. The backward Euler method is one possibility, but it is only first order accurate. It is possible to derive higher order implicit Runge–Kutta methods that are L-stable. As one example, we mention the two-stage second order accurate diagonally implicit method

$$\begin{aligned} U^* &= U^n + \frac{k}{4}(f(U^n) + f(U^*)), \\ U^{n+1} &= \frac{1}{3}(4U^* - U^n + kf(U^{n+1})). \end{aligned} \tag{8.6}$$

Each stage is implicit. The first stage is simply the trapezoidal method (or trapezoidal rule, hence *TR* in the name) applied over time $k/2$. This generates a value U^* at $t_{n+1/2}$. Then the two-step BDF method is applied to the data U^n and U^* with time step $k/2$ to obtain U^{n+1}. This method is written in a different form in (5.37).

8.6 Runge–Kutta–Chebyshev explicit methods

While conventional wisdom says that implicit methods should be used for stiff problems, there's an important class of "mildly stiff" problems for which special explicit methods have been developed with some advantages. In this section we take a brief look at the Runge–Kutta–Chebyshev methods that are applicable to problems where the eigenvalues of the Jacobian matrix are on the negative real axis and not too widely distributed. This type of problem arises, for example, when a parabolic heat equation or diffusion equation is discretized in space, giving rise to a large system of ordinary differential equations in time. This problem is considered in detail in Chapter 9.

The idea is to develop an explicit method whose stability region stretches out along the negative real axis as far as possible, without much concern with what's happening away from the real axis. We want the region of absolute stability $\mathcal{S}$ to contain a "stability interval" $[-\beta, 0]$ that is as long as possible. To do this we will consider multistage explicit Runge–Kutta methods as introduced in Section 5.7, but now as we increase the number of stages we will choose the coefficients to make β as large as possible rather than to increase the order of accuracy of the method.

We will consider only first order methods, because they are easiest to study. Second order methods are commonly used in practice, and this is often quite sufficient for the applications we have in mind, such as integrating method of lines (MOL) discretizations of the heat equation. The system of ODEs is obtained by discretizing the original partial differential equations PDE in space, and often this discretization is only second order accurate spatially. Second order methods similar to those described here can be found in [80], [97], [2] and higher order methods in [1], [69], for example.

Consider an explicit r-stage Runge–Kutta method of the form (5.34) with $a_{ij} = 0$ for $i \le j$. If we apply this method to $u' = \lambda u$ we obtain

$$U^{n+1} = R(z)U^n,$$

where $z = k\lambda$ and $R(z)$ is a polynomial of degree r in z, the *stability polynomial*. We will write this as

$$R(z) = d_0 + d_1 z + d_2 z^2 + \cdots + d_r z^r. \tag{8.7}$$

Consistency and first order accuracy require that

$$d_0 = 1, \quad d_1 = 1. \tag{8.8}$$

The region of absolute stability is

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \le 1\}. \tag{8.9}$$

The coefficients d_j depend on the coefficients A and b defining the Runge–Kutta method, but for the moment suppose that for any choice of d_j we can find suitable coefficients A and b that define an explicit r-stage method with stability polynomial (8.7). Then our goal is to choose the d_j coefficients so that $\mathcal{S}$ contains an interval $[-\beta, 0]$ of the negative real axis with β as large as possible. We also require (8.8) for a first order accurate method.

Note that the condition (8.8) amounts to requiring

$$R(0) = 1 \quad \text{and} \quad R'(0) = 1. \tag{8.10}$$

Then we want to choose $R(z)$ as a polynomial of degree r that satisfies $|R(z)| \le 1$ for as long as possible as we go out on the negative real axis. The solution is simply a scaled and shifted Chebyshev polynomial. Figure 8.6 shows the optimal polynomials of degrees 3 and 6. Note that these are plots of $R(x)$ for x real.

We have seen several other examples where Chebyshev polynomials solve problems of this sort, and as usual the optimal solution equioscillates at a set of points, in this case $r + 2$ points (including $x = -\beta$ and $x = 0$), where $|R(x)| = 1$. If we try to perturb the polynomial so that $|R(-\beta)| < 1$ (and hence the method will be stable further out on the negative real axis), one of the other extrema of the polynomial will move above 1, losing stability for some z closer to the origin.

The optimal polynomial is

$$R(z) = T_r\left(1 + z/r^2\right), \tag{8.11}$$

where $T_r(z)$ is the Chebyshev polynomial of degree r, as in Section B.3.2. For this choice $R(-2r^2) = T_r(-1) = \pm 1$, while $|R(x)| > 1$ for $x < -2r^2$, and so

$$\beta(r) = 2r^2. \tag{8.12}$$

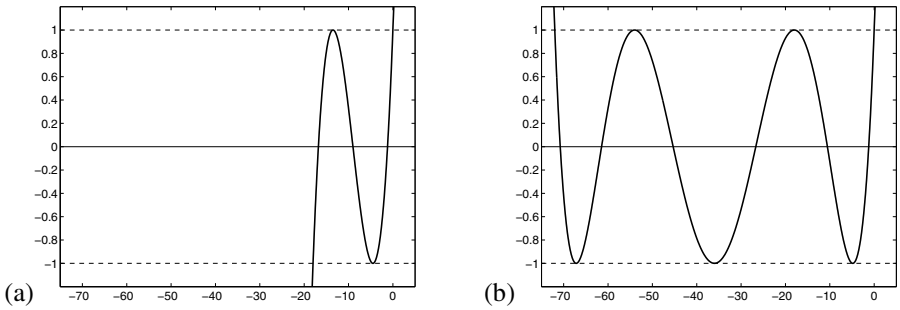

Figure 8.6. *Shifted Chebyshev polynomials corresponding to $R(x)$ along the real axis for the first order Runge–Kutta–Chebyshev methods.* (a) $r = 3$. (b) $r = 6$.

We have optimized along the negative real axis, but it is interesting to see what the stability region $\mathcal{S}$ from (8.9) looks like in the complex plane. Figure 8.7 shows the stability region for the two polynomials shown in Figure 8.6 of degrees 3 and 6. They do include the negative real axis out to $-\beta(r)$, but notice that everywhere $|R(x)| = 1$ on the real axis is a point where the region $\mathcal{S}$ contracts to the axis. Perturbing such an x away from the axis into the complex plane gives a point z where $|R(z)| > 1$. So these methods are suitable only for problems with real eigenvalues, such as MOL discretizations of the heat equation.

Even for these problems it is generally preferable to perturb the polynomial $R(z)$ a bit so that $|R(z)|$ is bounded slightly below 1 on the real axis between $-\beta$ and the origin, which can be done at the expense of decreasing β slightly. This is done by using the shifted Chebyshev polynomial

$$R(z) = \frac{T_r(w_0 + w_1 z)}{T_r(w_0)}, \qquad \text{where } w_1 = \frac{T_r(w_0)}{T_r'(w_0)}.$$

Here $w_0 > 1$ is the "damping parameter" and w_1 is chosen so that (8.8) holds and hence the method remains first order accurate. Now $R(x)$ alternates between $\pm(T_r(w_0))^{-1} < 1$ on the interval $[-\beta, 0]$, where β is now found by solving $w_0 - \beta w_1 = -1$, so

$$\beta = \frac{(w_0 + 1)T_r'(w_0)}{T_r(w_0)}.$$

Verwer [97] suggests taking $w_0 = 1 + \epsilon/r^2$ with $\epsilon = 0.05$, for which

$$\beta \approx \left(2 - \frac{4}{3}\epsilon\right) r^2 \approx 1.93r^2.$$

This gives about 5% damping with little reduction in β. Figure 8.8 shows the stability regions for the damped first order methods, again for $r = 3$ and $r = 6$.

Once we have determined the desired stability polynomial $R(z)$, we must still develop an r-stage Runge–Kutta method with this stability polynomial. In general there are infinitely many such Runge–Kutta methods. Here we discuss one approach that has several desirable properties, the *Runge–Kutta–Chebyshev* method introduced by van der Houwen

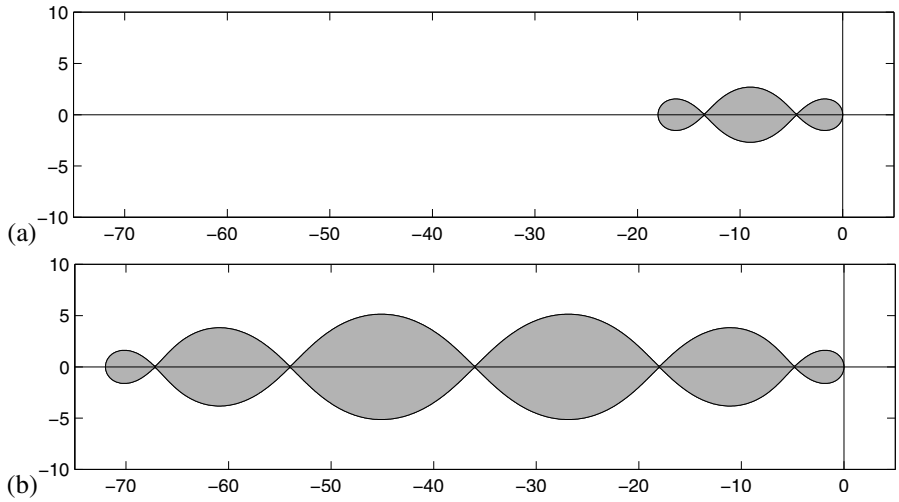

Figure 8.7. *Absolute stability regions for the first order Runge–Kutta–Chebyshev methods.* (a) $r = 3$. (b) $r = 6$.

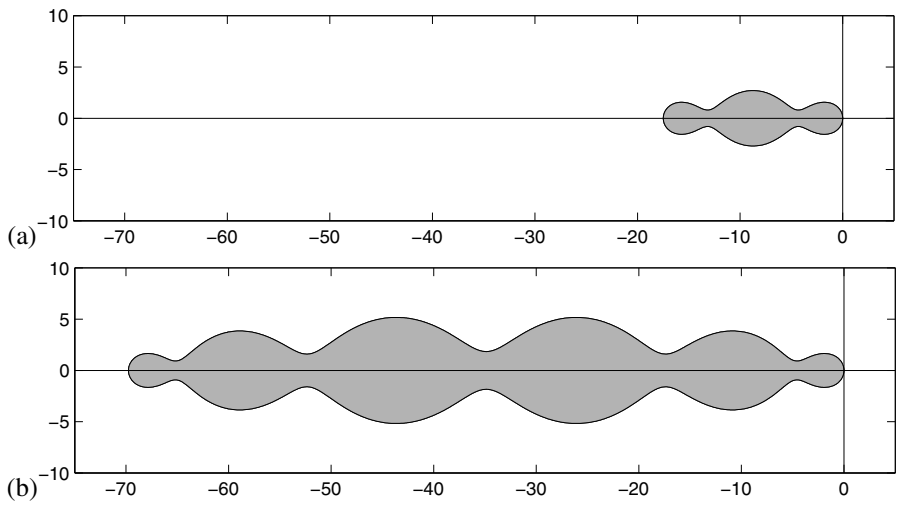

Figure 8.8. *Absolute stability regions for the first order Runge–Kutta–Chebyshev methods with damping.* (a) $r = 3$. (b) $r = 6$.

and Sommeijer [94]. See the review paper of Verwer [97] for a derivation of this method and a discussion and comparison of some other approaches, and see [80] for a discussion of software that implements these methods.

The Runge–Kutta–Chebyshev methods are based on the recurrence relation of Chebyshev polynomials and have the general form

$$
\begin{aligned}
Y_0 &= U^n, \\
Y_1 &= Y_0 + \tilde{\mu}_1 k f(Y_0, t_n), \\
Y_j &= (1 - \mu_j - \nu_j) Y_0 + \mu_j Y_{j-1} \\
&\quad + \nu_j Y_{j-2} + \tilde{\mu}_j k f(Y_{j-1}, t_n + c_{j-1} k) + \tilde{\gamma} k f(Y_0, t_n), \quad j = 2, \ \ldots, \ r, \\
U^{n+1} &= Y_r.
\end{aligned}
\tag{8.13}
$$

The various parameters in these formulas are given in the references above. This is an r-stage explicit Runge–Kutta method, although written in a different form than (5.34). An advantage of this recursive form is that only three intermediate solution vectors Y_0, Y_{j-1}, and Y_{j-2} are needed to compute the next Y_j, no matter how many stages r are used. This is important since large values of r (e.g., $r = 50$) are often used in practice, and for PDE applications each Y_j is a grid function over the entire (possibly multidimensional) spatial domain.

Another advantage of the Runge–Kutta–Chebyshev methods is that they have good *internal stability* properties, as discussed in the references above. We know the stability region of the overall stability polynomial $R(z)$, but some r-stage methods for large r suffer from exponential growth of the solution from one stage to the next in the early stages before ultimately decaying even when $|R(z)| < 1$ (analogous to the transient growth sometimes seen when iterating with a nonnormal matrix even if it is asymptotically stable; see Section D.4). This growth can cause numerical difficulties if methods are not carefully designed.