Ordinary Differential Equations

Stability and Stiffness

There are basically three things that can go wrong when we try to solve an ODE $\frac{dy}{dx} = f(x,y)$ numerically:

- Either f or y may be ill-behaved. This may be hard to spot, as it may [well] depend on the initial conditions. The remedy is as for quadrature; change variables, or use analytic approximations to the solution near singularities. More 'interesting' are the other possibilities:

- The chosen solution method may interact badly with the ODE. This is a *stability* problem, and there are several categories. We look in particular at *partial*, *weak* and *strong* stability.

- The desired solution may be difficult to find because of the nature of the family of solutions. This is a *stiffness* problem. There is normally little that can be done for direct solution. Sometimes it helps to run the solution 'backwards'; compare the I_n problem that we looked at in lecture 1.

Partial [in]stability

A partial instability is caused by taking too large a step-length, h. For example, suppose we try to solve

$$\frac{dy}{dx} = \lambda y;\ y = 1 \text{ when } x = 0$$

by using the Modified Euler Method with step-length $h = 0.1$. The exact solution is $y = e^{\lambda x}$. The MEM tells us that $y_1 = 1 + 0.1\lambda + 0.005\lambda^2$ [check!], and $y(1) = y_{10} = (1 + 0.1\lambda + 0.005\lambda^2)^{10}$, when it should be e^{λ}. The following table shows, for various values of λ, how accurate MEM is:

λ	e^{λ}	MEM value	error(%)
0	1.0000	1.0000	0
0.5	1.6487	1.6483	–0.02
1	2.7183	2.7141	–0.2
2	7.3891	7.3046	–1.1
5	148.41	128.39	–13
10	22026	9536.7	–57
25	7e10	1.6e8	–99.8
–0.5	0.6065	0.6067	0.02
–1	0.3679	0.3685	0.2
–2	0.1353	0.1374	1.6
–5	0.0067	0.0091	34
–10	0.000045	0.00098	2000
–25	1.4e–11	128	9e14

Note that for $|\lambda| < 2$, the results are reasonable; but for larger $|\lambda|$ the results get steadily worse. The reason is simple; the truncation error in MEM for finding $y(1)$ is $-\frac{1}{6}h^2y'''$ [exercise!], so will be relatively large unless h^2 is small compared with a typical third derivative, here λ^3. Note that it doesn't matter much whether λ is positive or negative; a rapidly-decaying exponential is as hard to follow as a rapidly-growing one. The cure is to take h smaller, until the error is 'small enough'.

Much the same will happen to all Runge–Kutta and Adams–Bashforth methods; a higher power of h may be involved, but equally a higher derivative of y. Transient behaviour of an ODE may force you to use a much smaller step-length than you might expect, even if transient solutions rapidly become negligible.

Strong and weak stability

Details beyond the scope of this module. The idea is that any method for solving ODEs has 'spurious' solutions. When we build up a table of results, the values in that table are subject to the usual truncation errors and rounding errors. The effect is that we have added in to our solution a small [we hope!] error function, which then participates in the solution for the next step. What happens to this error function?

We cannot normally expect it to go away. [Imagine solving the simplest possible ODE, $\mathrm{d}y/\mathrm{d}x = 0$, solution $y =$ constant. If we get one of the y values wrong, then the constant is just as wrong, and the error will persist.] In general, if we have a series of 'wrong' y values, then our process [such as RK or AB] will continues that series with further 'wrong' values that will take on a life of their own, depending not so much on f [which is just zero in this very simple case] as on previous y's. If the *only* life that persists is either the constant error or a decaying error, then the method is *strongly stable*. If there are other error terms that do not decay, but do not grow either, then the method is *weakly* stable. If there are growing error terms, then the method is *unstable*.

Unstable methods are Bad News. Basically, the instability is structural, within the method, not broadly to do with the ODE itself; but it is possible for 'resonance' to afflict any method, so there are no cast-iron guarantees. Taking shorter step lengths just means you take more steps, giving the errors more chance to grow. So the only solution is to change methods. No-one voluntarily uses unstable methods. Luckily, the usual RK/AB methods are strongly stable except in bizarre cases.

Stiffness

For example, let us tweak our usual ODE:

$$\frac{dy}{dx} = 10xy - (10x+1)e^{-x};\ y = 1 \text{ when } x = 0,$$

of which the solution is $y = e^{-x}$. What happens if we try [eg] Runge–Kutta on this equation? Well, with $h = 0.1$, we find $y(1) \approx 0.36732$, not bad compared with $e^{-1} \approx 0.36788$. But we find $y(2) \approx -1530.3$, which is rather a long way from $e^{-2} \approx 0.13534$. How about $h = 0.01$? That gives $y(2) \approx -0.08045$; still not terribly good! This is a form of partial instability; by the time we take $h = 0.002$, we find $y(2) \approx 0.13500$, and we are doing better.

But why are we having to do such an incredible amount of work? The *general* solution of the given ODE is $y = e^{-x} + Ae^{5x^2}$. We are looking for the solution with $A = 0$; but the slightest rounding or truncation error, and we have added in an exponentially-growing solution. For $x = 1$, $e^5 \approx 150$, so we can get reasonable accuracy provided A is much less than [say] 10^{-6}. For $x = 2$, $e^{20} \approx 5\times10^8$, so we need A to be something like 10^{-12} to get even 3dp accuracy, and we are really straining calculator accuracy. For $x = 3$, $e^{45} \approx 3\times10^{19}$, and you will need 24sf accuracy in both rounding and truncation errors for reasonable accuracy in the result; you won't get that from a calculator, and nor from a computer program unless you use high precision arithmetic together with a highly accurate method.

So the problem really is that $y(3)$, for example, is depending amazingly critically on the value of $y(0)$; no numerical process whatsoever is going to find that easy. [Computer demos of this 'instability' are quite pretty.]

Note: *(a)* Solutions that look like [for example] e^{-5x^2} are just as bad numerically, as discussed previously, even though the true solution now depends only amazingly weakly on A and hence on $y(0)$. *(b)* If you are solving simultaneous ODEs, then ill-conditioning, as discussed for [even] 2×2 matrices, can lead to unexpectedly large/small eigenvalues, which can give the same sorts of critical behaviour, but in much less obvious form.