Week 13: Monday, Apr 23

Ordinary differential equations

Consider ordinary differential equations of the form

$$y' = f(t, y) \tag{1}$$

together with the initial condition $y(0) = y_0$. These are equivalent to integral equations of the form

$$y(t) = y_0 + \int_0^t f(s, y(s))\, ds. \tag{2}$$

Only the simplest sorts of differential equations can be solved in closed form, so our goal for the immediate future is to try to solve these ODEs numerically.

For the most part, we will focus on equations of the form (1), even though many equations of interest are not posed in this form. We do this, in part, because we can always convert higher-order differential to first-order form by adding variables. For example,

$$my'' + by' + ky = f(t)$$

becomes

$$\begin{bmatrix} y \\ v \end{bmatrix}' = \begin{bmatrix} v \\ f(t) - \frac{b}{m}v - \frac{k}{m}y \end{bmatrix}.$$

Thus, while there are methods that work directly with higher-order differential equations (and these are sometimes preferable), we can always solve high-order equations by first converting them to first-order form by introducing auxiliary variables. We can also always convert equations of the form

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

into *autonomous systems* of the form

$$u' = g(u)$$

by defining an auxiliary equation for the evolution of time:

$$u = \begin{bmatrix} y \\ t \end{bmatrix}, \quad g(u) = \begin{bmatrix} f(t, y) \\ 1 \end{bmatrix}.$$

Basic methods

Maybe the simplest numerical method for solving ordinary differential equations is *Euler's method*. Euler's method can be written in terms of finite difference approximation of the derivative, Hermite interpolation, Taylor series, numerical quadrature using the left-hand rule, or the method of undetermined coefficients. For the moment, we will start with a derivation by Taylor series. If $y(t)$ is known and $y' = f(t, y)$, then

$$y(t+h) = y(t) + hf(t, y(t)) + O(h^2)$$

Now, drop the $O(h^2)$ term to get a recurrence for $y_k \approx y(t_k)$:

$$y_{k+1} = y_k + h_k f(t_k, y_k)$$

where $t_{k+1} = t_k + h_k$.

We can derive another method based on first-order Taylor expansion about the point $t + h$ rather than t:

$$y(t) = y(t+h) - hf(t+h, y(t+h)) + O(h^2).$$

If we drop the $O(h^2)$ term, we get the approximation $y(t_k) = y_k$ where

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}).$$

This is the *backward Euler* method, sometimes also called implicit Euler. Notice that in the backward Euler step, the unknown y_{k+1} appears on both sides of the equations, and in general we will need a nonlinear equation solver to take a step.

Another basic method is the *trapezoidal rule*. Let's think about this one by quadrature. If we apply the trapezoidal rule to (2), we have

$$\begin{aligned} y(t+h) &= y(t) + \int_t^{t+h} f(s, y(s))\, ds \\ &= y(t) + \frac{h}{2}\left(f(t, y(t)) + f(t+h, y(t+h))\right) + O(h^3) \end{aligned}$$

If we drop the $O(h^3)$ term, we have

$$y_{k+1} = y_k + \frac{h}{2}\left(f(t_k, y_k) + f(t_{k+1}, y_{k+1})\right).$$

Like the backward Euler rule, the trapezoidal rule is implicit: in order to compute y_{k+1}, we have to solve a nonlinear equation.

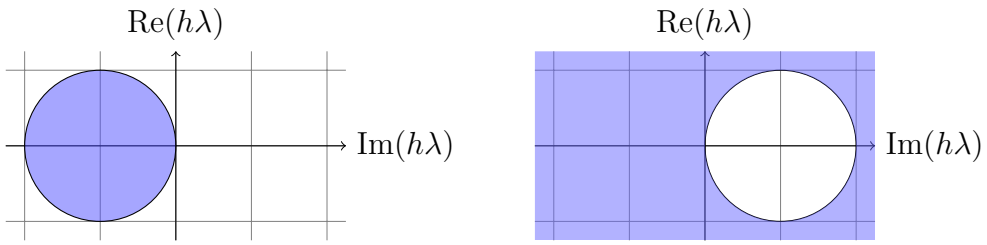

Figure 1: Regions of absolute stability for Euler (left) and backward Euler (right). For values of $h\lambda$ in the colored region, the numerical method produces decaying solutions to the test problem $y' = \lambda y$.

Stability regions

Consider what happens when we apply Euler and backward Euler to a simple linear test problem

$$y' = \lambda y$$

with a fixed step size h. Note that the solutions $y(t) = y_0 \exp(\lambda t)$ decay to zero whenever $\mathrm{Re}(\lambda) < 0$. This is a qualitative property we might like our numerical methods to reproduce. Euler's method yields

$$y_{k+1} = (1 + h\lambda) y_k,$$

which gives decaying solutions only when $|1 + h\lambda| < 1$. The set of values $h\lambda$ where Euler produces a decaying solution is called the *region of absolute stability* for the method. This region is shown in Figure 1.

Backward Euler produces the iterates

$$y_{k+1} = (1 - h\lambda)^{-1} y_k$$

Therefore, the discrete solutions decay whenever $|(1 - h\lambda)^{-1}| < 1$ — or, equivalently, whenever $|1 - h\lambda| > 1$. Thus, the region of absolute stability includes the entire left half plane $\mathrm{Re}(\lambda) < 0$ (see Figure 1), and so backward Euler produces a decaying solution when $\mathrm{Re}(\lambda) < 0$, no matter how large or small h is. This property is known as A-stability.

Euler and trapezoidal rules

So far, we have introduced three methods for solving ordinary differential equations: forward Euler, backward Euler, and the trapezoidal rule:

$$\begin{aligned}
y_{n+1} &= y_n + hf(t_n, y_n) && \text{Euler} \\
y_{n+1} &= y_n + hf(t_{n+1}, y_{n+1}) && \text{Backward Euler} \\
y_{n+1} &= y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right) && \text{Trapezoidal}
\end{aligned}$$

Each of these methods is *consistent* with the ordinary differential equation

$$y' = f(t, y).$$

That is, if we plug solutions to the exact equation into the numerical method, we get a small *local error*. For example, for forward Euler we have *consistency of order 1*,

$$\mathcal{N}_h y_h(t_{n+1}) \equiv \frac{y(t_{n+1}) - y(t_n)}{h_n} - f(t_n, y(t_n)) = O(h_n),$$

and for the trapezoidal rule we have second-order consistency

$$\mathcal{N}_h y_h(t_{n+1}) \equiv \frac{y(t_{n+1}) - y(t_n)}{h_n} - f(t_n, y(t_n)) = O(h_n^2).$$

Consistency + 0-stability = convergence

Each of the numerical methods we have described can be written in the form

$$\mathcal{N}_h y^h = 0,$$

where y^h denotes the numerical solution and $\mathcal{N}_h$ is a (nonlinear) difference operator. If the method is consistent of order p, then the true solution gives a small *residual* error as h goes to zero:

$$\mathcal{N}_h y = O(h^p).$$

As we have seen in the past, however, small *residual* error is not the same as small *forward* error. In order to establish convergence, therefore, we need

one more property. Formally, a method is *zero-stable* if there are constants h_0 and K so for any mesh functions x^h and z^h on an interval $[0, T]$ with $h \leq h_0$,

$$\|d^h\|_\infty \equiv \|x^h - z^h\|_\infty \leq K \left\{ |x_0 - z_0| + \left\| \mathcal{N}_h x^h(t_j) - \mathcal{N}_h z^h(t_j) \right\|_\infty \right\}.$$

Zero stability essentially says that the difference operators $\mathcal{N}_h$ can't become ever more singular as $h \to 0$: they are invertible, and the inverse is bounded by K. If a method is consistent and zero stable, then the error at step n is

$$|y(t_n) - y^h(t_n)| = |e_n| \leq K \max_j |d_j| = O(h^p).$$

The proof is simply a substitution of y and y^h into the definition of zero stability. The only tricky part, in general, is to show that the method is zero stable. Let's at least do this for forward Euler, to see how it's done — but you certainly won't be required to describe the details of this calculation on an exam!

We assume without loss of generality that the system is autonomous ($y' = f(y)$). We also assume that f is *Lipschitz continuous*; that is, there is some L so that for any x and z,

$$|f(x) - f(z)| \leq L|x - y|.$$

It turns out that Lipschitz continuity of f plays an important rule not only in the numerical analysis of ODEs, but in the theory of existence and uniqueness of ODEs as well: if f is not Lipschitz, then there might not be a unique solution to the ODE. The standard example of this is $u' = 2\,\mathrm{sign}(u)\sqrt{|u|}$, which has solutions $u = \pm t^2$ that both satisfy the ODE with initial condition $u(0) = 0$.

We can rearrange our description of $\mathcal{N}_h$ to get

$$\begin{aligned} x_{n+1} &= x_n + hf(x_n) + h\mathcal{N}_h[x](t_n) \\ z_{n+1} &= z_n + hf(z_n) + h\mathcal{N}_h[z](t_n). \end{aligned}$$

Subtract the two equations and take absolute values to get

$$|x_{n+1} - z_{n+1}| \leq |x_n - z_n| + h|f(x_n) - f(z_n)| + h|\mathcal{N}_h[x](t_n) - \mathcal{N}_h[z](t_n)|$$

Define $d_n = |x_n - z_n|$ and $\theta = \max_j |\mathcal{N}_h[x](t_j) - \mathcal{N}_h[z](t_j)|$. Note that by Lipschitz continuity, $|f(x_n) - f(z_n)| < Ld_n$; therefore,

$$d_{n+1} \leq (1 + hL)d_n + h\theta.$$

Let's look at the first few steps of this recurrence inequality:

$$\begin{aligned}
d_1 &\leq (1+hL)d_0 + h\theta \\
d_2 &\leq (1+hL)^2 d_0 + \left[(1+hL)+1\right] h\theta \\
d_3 &\leq (1+hL)^3 d_0 + \left[(1+hL)^2 + (1+hL) + 1\right] h\theta
\end{aligned}$$

In general, we have

$$\begin{aligned}
d_n &\leq (1+hL)^n d_0 + \left[\sum_{j=0}^{n-1} (1+hL)^j\right] h\theta \\
&\leq (1+hL)^n d_0 + \left[\frac{(1+hL)^n - 1}{(1+hL)-1}\right] h\theta \\
&\leq (1+hL)^n d_0 + L^{-1}\left[(1+hL)^n - 1\right]\theta
\end{aligned}$$

Now note that

$$(1+hL)^n \leq \exp(Lnh) = \exp(L(t_n - t_0)) \leq \exp(LT),$$

where T is the length of the time interval we consider. Therefore,

$$d_n \leq \exp(LT) d_0 + \frac{\exp(LT)-1}{L} \max_j |\mathcal{N}_h[x](t_j) - \mathcal{N}_h[z](t_j)|.$$

While you need not remember the entire argument, there are a few points that you should take away from this exercise:

1. The basic analysis technique is the same one we used when talking about iterative methods for solving nonlinear equations: take two equations of the same form, subtract them, and write a recurrence for the size of the difference.

2. The Lipschitz continuity of f plays an important role. In particular, if LT is large, $\exp(LT)$ may be very inconvenient, enough so that we have to take very small time steps to get good error results according to our theory.

As it turns out, in practice we will usually give up on global accuracy bounds via analyzing Lipschitz constant. Instead, we will use the same sort of

local error estimates that we described when talking about quadrature: look at the difference between two methods that are solving the same equation with different accuracy, and use the difference of the numerical methods as a proxy for the error. We will discuss this strategy — and more sophisticated Runge-Kutta and multistep methods — next lecture.