Timestepping PDEs

Chris Rackauckas

March 13, 2019

1 Backwards Differentiation Formulae

Polynomial interpolation

$$y' = \frac{3y_i - 4y_{i-1} + y_{i-2}}{2\Delta t} + \mathcal{O}(\Delta t^2)$$

$$y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2}{3}hf(t_{n+2}, y_{n+2})$$

Starting method is Euler. Solving:

$$y_{n+2} - \frac{2}{3}hf(t_{n+2}, y_{n+2}) - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = 0$$

$$J(x_n) = x_n - \frac{2}{3}hf'(x_n)$$

$$x_{n+1} = x_n - J^{-1}(x_n)f(x_n)$$

Steps for Newton-Krylov BDF2:

$$\begin{aligned} f_n &= f(x_n) \\ J(x_n)z_n &= f_n \\ z_n &= gmres(J, f_n) \\ x_{n+1} &= x_n + z_n \end{aligned}$$

Using the non-uniform BDF discretization, we get

$$y_{n+2} - \frac{(1+\omega_{n+1})^2}{1+2\omega_{n+1}}y_{n+1} + \frac{\omega_{n+1}^2}{1+2\omega_{n+1}}y_n = h_{n+2}\frac{1+\omega_{n+1}}{1+2\omega_{n+1}}f_{n+2}$$

where $\omega_{n+1} = h_{n+2}/h_{n+1}$, $h_{n+2} = t_{n+2} - t_{n+1}$, $h_{n+1} = t_{n+1} - t_n$. By checking against the 3rd derivative, one can derive:

$$LTE \approx est = \frac{(h_{n+1} + h_{n+2})}{6}\left[\frac{y_{n+2} - y_{n+1}}{h_{n+2}} - \left(1 + \frac{h_{n+2}}{h_{n+1}}\right)\frac{y_{n+1} - y_n}{h_{n+1}} + \frac{h_{n+2}}{h_{n+1}h_n}(y_n - y_{n-1})\right]$$

where $h_n = t_n - t_{n-1}$. Then we check

$$q = \left(0.9\|\frac{y * reltol - abstol}{est}\|\right)^{\frac{1}{3}}$$

and then $h_{n+1} = qh_n$, where if $q < 1$ you reject the step.

2 Pseudospectral

$$u_t = u_{xx} + f$$

write in the Fourier basis, get an ODE. But how to compute f?

$$\overline{u}' = A\overline{u} + \mathcal{F}\left(f\left(\mathcal{F}^{-1}(u)\right)\right)$$

is an ODE for the coefficients.

3 Norsett-Euler and ExpRB

$$u' = Au + f(u)$$

$$u(t) = e^{-tA}u_0 + \int_0^t e^{-(t-\tau)A} f(u(\tau))d\tau$$

$$u_{n+1} = e^{-hA}u_n + h\varphi_1(-hA) f(u_n)$$

where

$$\varphi_1(z) = \frac{e^z - 1}{z}$$

or

$$u_{n+1} = e^{-hA}u_n + A\left(e^{-hA} - 1\right) f(u_n)$$

This can be made into a Runge-Kutta method on the exponentials. For example, one well-known method due to Hochbruck, known as exprb32, is:

$$k = u_n + h_n\varphi_1(h_nA) f(u_n)$$
$$u_{n+1} = u_n + h_n\varphi_1(h_nA_n) f(u_n) + h_n 2\varphi_3(h_nA) D$$

where

$$D = f(k) - f(u_n)$$

Notice that this method has k as the Norsett-Euler method, and thus $u_{n+1} - k$ is an error estimator which can be used for adaptive time stepping. As a side note, this method can be applied to fully nonlinear ODEs/PDEs as well via linearization. If you let

$$u' = g(u)$$

and then you can split via

$$u' = g'(u) + h(u)$$

where $h(u) = g(u) - g'(u)$. This is known as an exponential Rosenbrock method.

4 IMEX

IMEX methods are also known as "multi-rate solvers" given there ability to directly handle equations in different forms.

$$\frac{dy}{dt} = f(y,p,t) + g(y,p,t)$$

Do a different method on the two parts. Simplest is IMEX Euler: do explicit Euler on g while you do implicit Euler on f, which looks like:

$$y_{n+1} = y_n + \Delta t f(y_{n+1}) + \Delta t g(y_{n+1})$$

Notice that the linear structure for solving these equations is exactly the same, $G(y_{n+1}) = 0$ gives a nonlinear system with $G' = I - \Delta t f'$ which is the same Jacobian as implicit Euler, but now only on the f part. This can be extended to some higher order methods as well. For example, Crank-Nicholson Adams Bashforth 2.

$$y_{n+1} = y_n + \Delta t\left[\frac{3}{2}g(y_n) - \frac{1}{2}g(y_{n-1})\right] + \frac{\Delta t}{2}\left[f(y_{n+1}) + f(y_n)\right]$$

SBDF2

$$y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} + \frac{2}{3}\Delta t\left[2g(y_n) - g(y_{n-1})\right] + \frac{2}{3}\Delta t f(y_{n+1})$$

However, you can only do "trivial" mixing up to second order. To get higher than second order, you need the methods to satisfy extra order conditions, and thus have to directly derive a method that can satisfy the conditions. A popular class of IMEX methods are the ESDIRK methods of Kennedy and Carpenter.

5 Two-Dimensional Crank-Nicholson Improvement: ADI

Now let's try to see if there's a more clever way to solve the 2-dimensional semilinear heat equation which utilizes some of the structure of the equation itself. Our equation is

$$u_t = \Delta u + f(u) = u_{xx} + u_{yy} + f(u)$$

When we discretize via finite differences (or pseudospectral), we get:

$$u_t = A_x u + A_y u + f(u)$$

Crank-Nicholson-Euler is then

$$\frac{u_{n+1} - u_n}{\Delta t} = \frac{A_x u_{n+1} + A_y u_{n+1}}{2} + \frac{A_x u_n + A_y u_n}{2} + f(u_n)$$

$$\left(I - \frac{\Delta t}{2}A_x - \frac{\Delta t}{2}A_y\right)u_{n+1} = u_n + \Delta t\left(\frac{A_x u_n + A_y u_n}{2} + f(u_n)\right)$$

The trick is now to notice that

$$I-\frac{\Delta t}{2}A_x-\frac{\Delta t}{2}A_y-\frac{\Delta t^2}{4}A_xA_y=\left(I-\frac{\Delta t}{2}A_x\right)\left(I-\frac{\Delta t}{2}A_y\right)$$

and so

$$\left(I-\frac{\Delta t}{2}A_x-\frac{\Delta t}{2}A_y\right)=\left(I-\frac{\Delta t}{2}A_x\right)\left(I-\frac{\Delta t}{2}A_y\right)+\mathcal{O}\left(\Delta t^2\right)$$

and thus

$$\left(I-\frac{\Delta t}{2}A_x\right)\left(I-\frac{\Delta t}{2}A_y\right)u_{n+1}=u_n+\Delta t\left(\frac{A_xu_n+A_yu_n}{2}+f(u_n)\right)$$

keeps second order accuracy. If you write u_n as a matrix, this is much easier to solve because $A=A_yu+uA_x$ as tridiagonal matrices, and so we can keep A_x and A_y as small tridiagonals and do,

$$\begin{aligned} v&=u_n+\Delta t\left(\frac{A_xu_n+A_yu_n}{2}+f(u_n)\right)\\ w&=v\backslash\left(I-\frac{\Delta t}{2}A_x\right)\\ u_{n+1}&=\left(I-\frac{\Delta t}{2}A_y\right)\backslash w \end{aligned}$$

This is known as the ADI scheme, or the Alternative Direction Implicit scheme.

But this idea can be applied much more broadly as a matrix factorization approximation. Notice that the crux of the method is

$$\left(I-\frac{\Delta t}{2}A_x-\frac{\Delta t}{2}A_y\right)=\left(I-\frac{\Delta t}{2}A_x\right)\left(I-\frac{\Delta t}{2}A_y\right)+\mathcal{O}\left(\Delta t^2\right)$$

This applies even when the linear operators are Jacobians. So take for example the implicit Euler scheme. If we know of a good splitting for our equation:

$$u'=f_1(u)+f_2(u)$$

then the implicit Euler scheme gives Newton iterations like

$$G'(x_n)=(I-\Delta tf_1'(x_n)-\Delta tf_2'(x_n))=(I-\Delta tf_1')(I-\Delta tf_2')+\mathcal{O}\left(\Delta t^2\right)$$

This means you can solve two easy linear systems instead of one hard one! This can be applied to any of the implicit schemes, such as BDF2, and can be incorporated into the IMEX methods as well.