

Mat1062: Introductory Numerical Methods for PDE

Mary Pugh

April 2, 2009

1 Ownership

These notes are the joint property of Rob Almgren and Mary Pugh.

2 Nonlinear Equations

From the heat equation example, it's clear how to use spectral methods for a linear constant coefficient partial differential equation on a periodic domain. We now turn to nonlinear equations. As a sample equation, we will consider Burger's equation with dissipation:

$$u_t + uu_x = Du_{xx} \quad \text{on } (0, L)$$

with periodic boundary conditions. The first step is to find a spectral representation of this problem. We seek $u(x, t)$ so that

$$\langle u_t + uu_x - Du_{xx}, \phi_\ell \rangle = 0 \quad \forall \ell \in \mathbb{Z}, \forall t > 0$$

for each basis function $\phi_\ell = \exp(i\ell 2\pi x/L)$. That is,

$$\frac{d}{dt} \hat{u}_\ell + \widehat{(uu_x)}_\ell + D\ell^2 \frac{2\pi^2}{L} \hat{u}_\ell = 0 \quad \forall \ell \in \mathbb{Z}, \forall t > 0$$

with initial data $\hat{u}_\ell(0) = \hat{u}_{0\ell}$. And so again, we have reduced the PDE to infinitely many ODE but they are no longer decoupled:

$$\begin{aligned} u(x, t) &= \sum_{\ell=-\infty}^{\infty} \hat{u}_\ell(t) e^{i\ell 2\pi x/L} \implies u_x(x, t) = \sum_{\ell=-\infty}^{\infty} i\ell \frac{2\pi}{L} \hat{u}_\ell(t) e^{i\ell 2\pi x/L} \\ u(x, t) u_x(x, t) &= \sum_{\tilde{m}=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} im \frac{2\pi}{L} \hat{u}_m(t) \hat{u}_{\tilde{m}}(t) e^{i(\tilde{m}+m)2\pi x/L} \\ \implies (\widehat{uu_x})_\ell &= \sum_{m=-\infty}^{\infty} im \frac{2\pi}{L} \hat{u}_m(t) \hat{u}_{\ell-m}(t) \end{aligned}$$

And so the infinite system of ODE is

$$\begin{cases} \frac{d}{dt} \hat{u}_\ell = -D\ell^2 \left(\frac{2\pi}{L}\right)^2 \hat{u}_\ell(t) - \sum_{m=-\infty}^{\infty} im \frac{2\pi}{L} \hat{u}_m(t) \hat{u}_{\ell-m}(t) \\ \hat{u}_\ell(0) = \hat{u}_{0\ell} \end{cases} \quad (1)$$

for all $\ell \in \mathbb{Z}$.

2.1 Time-Stepping

Assuming that we somehow figure out how to deal with the convolution sum in (1), we see that in spectral space we will need to solve a nonlinear ODE at each wave number ℓ . And so we start by considering the ODE

$$\frac{d}{dt} \mathbf{u}(t) = f(\mathbf{u}(t)), \quad \mathbf{u}(0) = \mathbf{u}_0.$$

This is the first time we've needed to timestep a nonlinear ODE. Up until now, all of our ODE were linear because the PDE they came from were linear.

Clearly, we can still easily implement explicit Euler:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + k f(\mathbf{u}^n).$$

However, if we want to implement either the fully implicit or the Crank-Nicolson scheme

$$\mathbf{u}^{n+1} - k f(\mathbf{u}^{n+1}) = \mathbf{u}^n \quad \mathbf{u}^{n+1} - \frac{k}{2} f(\mathbf{u}^{n+1}) = \mathbf{u}^n + \frac{k}{2} f(\mathbf{u}^n)$$

then solving for \mathbf{u}^{n+1} is no longer an exercise in linear algebra — we would need to use some nonlinear solver like Newton-Raphson iteration. Suddenly,

Crank-Nicolson is no longer cheap to execute. If you're going to go through the hard work of coding up that nonlinear solver, you may wish to use it on a time-stepping scheme with higher accuracy. There is the third-order Adams-Moulton method:

$$u^{n+1} = u^n + \frac{1}{12}k \left(5f(u^{n+1}) + 8f(u^n) - f(u^{n-1}) \right)$$

and the fourth-order Adams Moulton method:

$$u^{n+1} = u^n + \frac{1}{24}k \left(9f(u^{n+1}) + 19f(u^n) - 5f(u^{n-1}) + f(u^{n-2}) \right).$$

To implement these schemes we need to invert one of $I - k f$ or $I - k/2 f$ or $I - 5k/12 f$ or $I - 9k/24 f$. Inverting any one of these will involve the same amount of labour coding and testing the code.

Because of the bother involved in programming up the nonlinear solver, and the slow-down caused by having to do a nonlinear solve with each time step, we may also wish to consider higher-order explicit schemes. The good news is that we can find such schemes and so there's no hard work in solving for u^{n+1} . The bad news is that these explicit schemes all have stability constraints on the time-step k , just as the explicit Euler scheme does. Here is the second-order Adams-Bashforth method:

$$u^{n+1} = u^n + \frac{1}{2}k \left(3f(u^n) - f(u^{n-1}) \right),$$

the third-order Adams-Bashforth method:

$$u^{n+1} = u^n + \frac{1}{12}k \left(23f(u^n) - 16f(u^{n-1}) + 5f(u^{n-2}) \right),$$

and the fourth-order Adams-Bashforth method:

$$u^{n+1} = u^n + \frac{1}{24}k \left(55f(u^n) - 59f(u^{n-1}) + 37f(u^{n-2}) - 9f(u^{n-3}) \right).$$

Note that all the higher-order Adams-Moulton and Adams-Bashforth schemes require memory of past values. This does not cause implementation problems. If you want to implement the third-order scheme then you would use explicit Euler to generate u^1 from u^0 . You'd then use the second-order scheme to generate u^2 from u^1 and u^0 . You'd then be in a position to start using the third-order scheme to generate u^n for all $n \geq 3$. Even though there were a few initial steps of lower-order accuracy, the total scheme would still be third-order accurate.

Alternate higher-order explicit schemes would include Runge-Kutta time-stepping methods. These are also explicit and, unfortunately, have stability constraints on the time-step. Runge-Kutta schemes do not use past values of u , they compute intermediate values; sort of like how the ADI Peaceman-Rachford scheme had to find an intermediate u in the process of finding u^{n+1} .

And so, given a particular problem that you want to compute you have to decide whether you are willing to spend the time programming up the nonlinear solver and have a code that runs a little more slowly because it has to execute a nonlinear solve at each time step or if you want to code up an explicit scheme which is easy to code and runs more quickly per time step except that because of the stability constraint you'll have to take more time steps to reach your desired end time.

For some problems, the stability constraint is prohibitive. For example, consider the Kuramoto-Sivashinsky equation: $u_t = -u_{xxxx} - u_{xx} - uu_x$ on $[0, L]$. This equation is a classic model problem for weak turbulence and the study of how PDE are related to dynamical systems. The solutions have very interesting behaviour, strongly influenced by the domain size. You need to be cunning about how you do the time-stepping for such a high-order equation: a naive explicit method will have a time step constraint of $k < ch^4$.

For dispersive problems, how one chooses the time-stepping is also influenced by questions of wanting to minimize the phase error introduced by the scheme.

2.2 Computing the Convolution Sum

In the system (1) for each $\ell \in \mathbb{Z}$ we need to compute

$$\sum_{m=-\infty}^{\infty} \hat{f}_m \hat{g}_{\ell-m} \quad (2)$$

where $\hat{f}_m = \text{im}(2\pi/L)\hat{u}_m$ and $\hat{g}_{\ell-m} = \hat{u}_{\ell-m}$. At first sight, this looks prohibitive: for a given ℓ we need to find

$$\cdots + \hat{f}_{-2} \hat{g}_{\ell+2} + \hat{f}_{-1} \hat{g}_{\ell+1} + \hat{f}_0 \hat{g}_{\ell} + \hat{f}_1 \hat{g}_{\ell-1} + \hat{f}_2 \hat{g}_{\ell-2} \cdots \quad (3)$$

Assume that ℓ is in the “active spectrum” at time t . Specifically, assume that $|\ell| \leq N_{as}$ where $\hat{f}_m \sim 10^{-16}$ for all $|m| > N_{as}/2$. Then the sum is over

$N_{\text{as}} + 1$ terms:

$$\hat{f}_{-N_{\text{as}}/2} \hat{g}_{\ell+N_{\text{as}}/2} \cdots + \hat{f}_{-1} \hat{g}_{\ell+1} + \hat{f}_0 \hat{g}_\ell + \hat{f}_1 \hat{g}_{\ell-1} \cdots \hat{f}_{N_{\text{as}}/2} \hat{g}_{\ell-N_{\text{as}}/2}$$

The subscript is for “active spectrum” — N_{as} will be a function of time. Similarly, assume $\hat{g}_m \sim 10^{-16}$ for all $|m| > N_{\text{as}}/2$ (this is certainly true for the above choice of f and g). And so the sum is over $N_{\text{as}} - |\ell| + 1$ terms:

$$\widehat{(fg)}_\ell = \sum_{m=\max\{-N_{\text{as}}/2, \ell-N_{\text{as}}/2\}}^{\min\{N_{\text{as}}/2, \ell+N_{\text{as}}/2\}} \hat{f}_m \hat{g}_{\ell-m}.$$

Computing this sum is $\mathcal{O}(N_{\text{as}})$. And so we see that if we were to compute the convolution sum (2) for each ℓ in a direct and naive manner it would take $\mathcal{O}(N_{\text{as}}^2)$ operations. Which is prohibitively slow.

So how do we compute the convolution? We like to be in frequency space when it comes to derivatives — they’re diagonal operators there. And they’re harder to deal with in real space — we get into finite differences. On the other hand, we like to be in real-space when it comes to nonlinearities — if we want to compute u^2 at $x = x_j$ it’s just $(u(x_j))^2$. Nonlinearities are harder to deal with in frequency space because of the convolutions they entail. And so we want a scheme that does derivatives in frequency space and nonlinearities in real space. This is what is meant by a *pseudospectral* schem.

If we want to do explicit Euler time-stepping on Burger’s equation:

$$\hat{u}_\ell^{n+1} = \hat{u}_\ell^n - k \left(-D\ell^2 (2\pi/L)^2 \hat{u}_\ell^n - \widehat{(u^n u_x^n)}_\ell \right) \quad (4)$$

we would do it as follows.

1. At time t_n we are in spectral space and have the discrete Fourier coefficients $\{\hat{u}_0^n, \dots, \hat{u}_{N-1}^n\}$. We use these Fourier coefficients to create

$$\hat{v}_k = \begin{cases} ik \frac{2\pi}{L} \hat{u}_k^n & 0 \leq k \leq \lfloor N/2 \rfloor - 1 \\ 0 & k = \lfloor N/2 \rfloor \\ -i(N-k) \frac{2\pi}{L} \hat{u}_k^n & \lfloor N/2 \rfloor + 1 \leq k \leq N-1 \end{cases}$$

which are a good approximation of the discrete Fourier coefficients of u_x^n (if N is large enough for u to be spectrally resolved).

2. Apply the inverse discrete Fourier transform (ifft) to $\{\hat{u}_\ell^n\}$ and $\{\hat{v}_\ell\}$ to recover u^n and u_x^n in physical space. Now apply the nonlinearity, creating

$$w_j = (\text{ifft}(\hat{u}^n))_j (\text{ifft}(\hat{v}))_j, \quad j = 0, \dots, N-1$$

which is uu_x in physical space.

3. Apply the discrete Fourier transform (fft) to $\{w_j\}$ creating

$$(\widehat{u^n u_x^n})_\ell = \text{fft}(w)_\ell \quad \ell = 0, \dots, N-1$$

Now that we have $(\widehat{u^n u_x^n})_\ell$ we can take a time step via (4)

This method for computing the nonlinearity can introduce aliasing error. And so if you ever find yourself on the verge of using a spectral method in your research I strongly encourage you to look into how to deal with this aliasing error. You can learn more in §3.4 of Canuto et al.

2.3 Let's compute!

We want to test various time-stepping schemes. First, we would like to have an exact solution to compare our numerics to. Such exact solutions are easily created, thanks to the Cole-Hopf transform¹. Specifically, if ϕ is a solution of the heat equation $\phi_t = D\phi_{xx}$ and

$$u = -D \frac{\phi_x}{\phi}$$

then u is a solution of $u_t + uu_x = Du_{xx}$.

Using this, for the periodic problem on $[0, L]$ we take $\phi(x, t)$ to be any linear combination of basic solutions such as

$$e^{-D\ell^2(\frac{2\pi}{L})^2 t} \sin(\ell 2\pi x/L)$$

and use this to generate a solution $u(x, t)$. If we were to take a single basic solution rather than a linear combination, the resulting $u(x, t)$ would be independent of time.

¹J.D. Cole "On a quasilinear parabolic equation occurring in aerodynamics" *Quarterly of Applied Mathematics* 9(1951)225-236, E. Hopf, "The partial differential equation $u_t + uu_x = \mu u_{xx}$ " *Communications in Pure and Applied Mathematics* 3(1950)201-230.

Specifically, we take

$$u_0(x) = -2D \frac{\cos(x)}{3 + \sin(x)} \implies u(x, t) = -2D \frac{e^{-Dt} \cos(x)}{3 + e^{-Dt} \sin(x)}$$

on $[0, 2\pi]$. We consider four time-stepping schemes:

1. An integrating factor scheme:

$$\hat{u}_\ell^{n+1} = e^{-D\ell^2 k} \left(\hat{u}_\ell^n - k (\widehat{u^n u_x^n})_\ell \right)$$

2. Explicit Euler time-stepping:

$$\hat{u}_\ell^{n+1} = (1 - D\ell^2 k) \hat{u}_\ell^n - k (\widehat{u^n u_x^n})_\ell$$

3. A scheme for which the linear term is treated implicitly and the non-linear term is treated explicitly:

$$\hat{u}_\ell^{n+1} = \frac{\hat{u}_\ell^n - k (\widehat{u^n u_x^n})_\ell}{1 + D\ell^2 k}$$

4. Explicit second-order Adams-Bashforth:

$$\hat{u}_\ell^{n+1} = \hat{u}_\ell^n + \frac{k}{2} \left(-D\ell^2 (3\hat{u}_\ell^n - \hat{u}_\ell^{n-1}) - 3(\widehat{u^n u_x^n})_\ell + (\widehat{u^{n-1} u_x^{n-1}})_\ell \right)$$

We test by taking $D = 2$, $N = 128$ and the time step $k = 1/1000$. We compute up to time $T = 1/100$ and then compare the computed solution to the exact solution. We divide the time step by 2 and then repeat. Computing seven solutions in this way, we compare the ratios of the errors in the L^∞ norm and find that the first three schemes have ratios going to 2 and the fourth scheme has ratios going to 4, as expected.

We now take the time step $k = 1/64000$ and compare the schemes. We find

$$\text{integrating factor} \implies \|\text{err}\|_{L^\infty} = 9.49e - 7$$

$$\text{explicit} \implies \|\text{err}\|_{L^\infty} = 7.79e - 7$$

$$\text{mixed} \implies \|\text{err}\|_{L^\infty} = 1.66e - 6$$

$$\text{Adams-Bashforth} \implies \|\text{err}\|_{L^\infty} = 1.34e - 9$$

The second-order Adams-Bashforth scheme has the smallest error, unsurprisingly.

The integrating factor and mixed schemes have no stability constraint on their time-step while the explicit and Adams-Bashforth scheme do. Indeed, when computing the explicit scheme, it was unstable for $k = 1/1000$ and $k = 1/2000$ but stable for $k = 1/4000$. The Adams-Bashforth scheme was unstable for $k = 1/4000$ too — it was stable for $k = 1/8000$. And so we see, in this example at least, that the Adams-Bashforth scheme had a tighter stability constraint than the explicit scheme.

You can find the “stability domains” for various time-stepping schemes in pretty much any graduate numerical analysis book, including the Canuto et al. book. Separately, stability analysis is usually done for *linear* systems which is one reason why it’s important to have some computational way of telling if your nonlinear computation is becoming unstable.