

Mat1062: Introductory Numerical Methods for PDE

Mary Pugh

March 31, 2009

1 Ownership

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

2 Further Sources

For more information on spectral and pseudospectral methods, have a look at “Spectral Methods: Fundamentals in Single Domains” by C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zhang. This is QA377 S677 in the Math/Stat library. Also, “Spectral methods in MATLAB” by L. M. Trefethen is a wonderfully readable, short introduction. This is QA377 T65 in the Math/Stat library.

3 The Heat Equation

Note: from now on, I will assume that I’ve resolved things well enough in space so that $\hat{u}_\ell^d = \hat{u}_\ell^c$ up to round off error. For this reason, I’ll stop carrying around the d and c superscripts and will simply refer to \hat{u}_ℓ .

We are now in a position to compute some initial value problems. Consider the heat equation with periodic boundary conditions:

$$\begin{cases} u_t = Du_{xx} & \forall t > 0, \forall x \in (0, L) \\ u(x, 0) = u_0(x) & \forall x \in [0, L] \\ u(0) = u(L) & \forall t > 0 \end{cases}$$

Applying the Fourier transform, this becomes an infinite family of uncoupled ODE:

$$\begin{cases} \frac{d}{dt}\hat{u}_\ell = -D\ell^2 (2\pi/L)^2 \hat{u}_\ell & \forall t > 0, \forall \ell \in \mathbb{Z} \\ \hat{u}_\ell(0) = \widehat{u_0}_\ell & \forall \ell \in \mathbb{Z} \end{cases}$$

We explicitly solve these ODE

$$\hat{u}_\ell(t) = e^{-D\ell^2 (2\pi/L)^2 t} \widehat{u_0}_\ell$$

and then apply the inverse fourier transform to find the true solution:

$$u(x, t) = \sum_{\ell=-\infty}^{\infty} e^{-D\ell^2 (2\pi/L)^2 t} \widehat{u_0}_\ell e^{i\ell 2\pi x/L}.$$

Note that I'm using ℓ for the wave number. This is because I will be using k for the time-step size shortly. If we wish to do this on the computer, we fix N and sample the initial data u_0 at the N points x_0, x_1, \dots, x_{N-1} and then take the discrete Fourier transform. In spectral space, we then define

$$\hat{v}_\ell^d = \begin{cases} e^{-D\ell^2 (2\pi/L)^2 t} \widehat{u_0}_\ell & 0 \leq \ell < N/2 \\ 0 & \ell = N/2 \\ e^{-D(N-\ell)^2 (2\pi/L)^2 t} \widehat{u_0}_\ell & N/2 < \ell \leq N-1 \end{cases} \quad (1)$$

and then apply the inverse discrete Fourier transform to recover $u(x, t)$ at the desired time t . *Note that I have switched to using N for the number of sample points. This is because I will shortly be talking about time-stepping from t_n to t_{n+1} .* Also note that I'm not worrying about whether N is even or odd. If N is even then I'm zeroing out the $\ell = \lfloor N/2 \rfloor$ mode. And if N is odd then the zeroing-out in (1) doesn't apply. Either way, (1) is correct.

The above approach solves the problem fully in frequency space. We don't need to do any time-stepping at all — we only transform back to physical space at those times for which we want a solution. However, we will soon see that there are times when we would like to take time steps of size k , rather than computing exactly up to the output time. This means we need to numerically approximate solutions of each ODE

$$\frac{d}{dt}\hat{u}_\ell = -D\ell^2 (2\pi/L)^2 \hat{u}_\ell \quad \hat{u}_\ell(0) = \widehat{u_0}_\ell.$$

We can do this in many ways, including fully explicit, fully implicit, and Crank-Nicolson. Fully implicit timestepping would lead to the algorithm:

1. At time $t_n = nk$, apply the discrete Fourier transform to the values $u_0^n, u_1^n, \dots, u_{N-1}^n$, creating the values $\widehat{u}^n_0, \widehat{u}^n_1, \dots, \widehat{u}^n_{N-1}$.
2. Create the \widehat{v}^d_ℓ values

$$\widehat{v}^d_\ell = \begin{cases} (1 - D\ell^2(2\pi/L)^2k) \widehat{u}^n_\ell & 0 \leq \ell < N/2 \\ 0 & \ell = N/2 \\ (1 - D(N-\ell)^2(2\pi/L)^2k) \widehat{u}^n_\ell & N/2 < \ell \leq N-1 \end{cases} \quad (2)$$

3. Apply the discrete inverse Fourier transform to $\widehat{v}^d_0, \widehat{v}^d_1, \dots, \widehat{v}^d_{N-1}$, thus creating u at the next time step: $u_0^{n+1}, u_1^{n+1}, \dots, u_{N-1}^{n+1}$.

We would use the same approach for fully implicit timestepping except that we'd take

$$\widehat{v}^d_\ell = \begin{cases} \frac{1}{1+D\ell^2(2\pi/L)^2k} \widehat{u}^n_\ell & 0 \leq \ell < N/2 \\ 0 & \ell = N/2 \\ \frac{1}{1+D(N-\ell)^2(2\pi/L)^2k} \widehat{u}^n_\ell & N/2 < \ell \leq N-1 \end{cases} \quad (3)$$

For Crank-Nicolson, we'd take

$$\widehat{v}^d_\ell = \begin{cases} \frac{1-D/2\ell^2(2\pi/L)^2k}{1+D/2\ell^2(2\pi/L)^2k} \widehat{u}^n_\ell & 0 \leq \ell < N/2 \\ 0 & \ell = N/2 \\ \frac{1+D/2(N-\ell)^2(2\pi/L)^2k}{1+D(N-\ell)^2(2\pi/L)^2k} \widehat{u}^n_\ell & N/2 < \ell \leq N-1 \end{cases} \quad (4)$$

When we used finite-difference methods for this problem, we had a tri-diagonal matrix that we needed to invert in order to take a time-step with fully implicit or Crank-Nicolson. Here, because the operator ∂_{xx} is a diagonal operator in frequency space, when it comes to inverting the operator it involves inverting a diagonal matrix, which we did by hand in finding (3) and (4). This is where those denominators came from. This shows one of the powers of the spectral method — it's fantastic for operators that diagonalise in frequency space. All constant coefficient differential operators do as do some integral operators.

Note that the three multipliers:

$$\left(1 - D\ell^2(2\pi/L)^2k\right), \quad \frac{1}{1 + D\ell^2(2\pi/L)^2k}, \quad \frac{1 - D/2(N - \ell)^2(2\pi/L)^2k}{1 + D(N - \ell)^2(2\pi/L)^2k} \quad (5)$$

are all different ways of doing a small- k approximation of the true multiplier

$$e^{-D\ell^2(2\pi/L)^2k}. \quad (6)$$

which would result in the exact solution for \hat{u}_ℓ^{n+1} given \hat{u}_ℓ^n . We call the multiplier (6) the integrating factor or propagator.

3.1 Simulations

To test the exact code, one would take simple initial data like $10 \sin(2\pi x/L) + \cos(4\pi x/L)$ and confirm that the resulting solution equals

$$10e^{-D(2\pi/L)^2t} \sin(2\pi x/L) + e^{-4D(2\pi/L)^2t} \cos(4\pi x/L)$$

up to roundoff error.

To test the codes with time-stepping, choose initial data and then choose N sufficiently large to resolve the initial data. At this point, the only source of error is in the time-stepping. This is because all frequencies in the initial data are resolved and because the equation is *linear* no higher frequencies will be needed to describe the resulting solution of the PDE. And so we do the usual thing which is to refine the time-steps and verify that the errors are decreasing by the expected ratios. For example, consider the initial data

$$u_0(x) = \frac{1}{1 + \sin(x/2)^2} \quad (7)$$

on $[0, 2\pi]$. We take $N = 64$, and compute from time $t = 0$ to time $t = 1$ with the initial time-step $h = 1/M = .01$. The following are the errors and ratios for the fully explicit, fully implicit, and Crank-Nicolson schemes.

M	expl. error	ratios	impl. error	ratios	C.-N. error	ratios
100	3.0819e+76		5.0667e-04	1.9980	1.1559e-06	4.0000
200	1.9131e+99		2.5359e-04	1.9990	2.8897e-07	4.0000
400	3.2266e+42		1.2685e-04	1.9995	7.2242e-08	4.0000
800	6.3474e-05	2.0002	6.3443e-05	1.9998	1.8061e-08	4.0000
1600	3.1733e-05	2.0001	3.1725e-05	1.9999	4.5151e-09	4.0000
3200	1.5866e-05		1.5864e-05		1.1288e-09	

Note that for the first three values of N , the explicit scheme is clearly unstable; for this reason the ratios aren't presented.

3.2 Time-step Constraints

As noted above, the explicit scheme is unstable if the timestep is too large. We knew this would happen because the von Neumann stability analysis applies directly to this spectral simulation. (For what size timesteps k do the multipliers (5) have magnitude less than or equal to 1 for all $|\ell| < n/2$?) However, it is instructive to look at the solutions and their power spectra at a sequence of times.

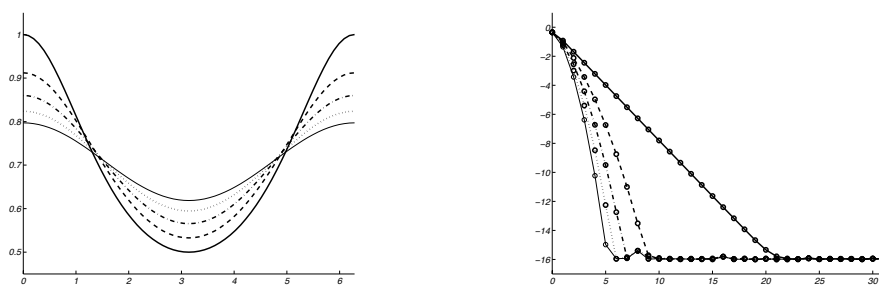


Figure 1: The initial data is given in (7). The approximate solution is computed using fully explicit timestepping. The initial time is $t = 0$. The time step is $h = 1/800$. The solution is on $[0, 2\pi]$ and the space step is $h = 2\pi/64$. Left plot: The approximate solutions are shown at times 0 (heavy solid), .25 (heavy dashed), .5 (heavy dot-dashed), .75 (heavy dotted), and 1 (solid). The maxima and minima are decreasing monotonically in time, as expected. The solution is converging to a constant as time goes to infinity. Right plot: Plotted here is the power spectrum, $(k, \log_{10} |\hat{u}_k(t)|)$, at the same times with the same line types. The active spectrum is decreasing in time. There is no numerical instability.

In Figure 1, we consider the explicit scheme with a time step that is small enough for the scheme to be stable. The initial data is (7). The left plot shows the solution at a sequence of times. Note that the Fourier coefficients that were initially at the level of round-off remain at the level of round-off. We call those modes whose Fourier coefficients are above the level of round-off the “active” spectrum. As time passes, the active spectrum becomes smaller and smaller; in the infinite time limit the active spectrum will simply be $\{0\}$ — the solutions are converging to a constant solution. Indeed, since the equation is linear and we know the precise decay rate of each mode we can calculate the time at which any particular mode will leave

the active spectrum:

$$|\hat{u}_k(t_c)| = |\hat{u}_k(0)|e^{-Dk^2(2\pi/L)t} \sim 10^{-16} \implies t_c \sim \frac{1}{Dk^2} \left(\frac{L}{2\pi} \right) \ln \left(\frac{|\hat{u}_k(0)|}{10^{-16}} \right)$$

For Figure 2, the time-step has been taken to be large enough that the scheme is numerically unstable. In the left plot, the approximate solutions are shown at a sequence of times. As expected, spurious oscillations eventually become visible to the eye. In the right plot, we see the power spectrum. This spectrum shows the instability immediately: with the very first time step, the Fourier coefficients at high frequencies start to grow. This spectrum shows the computation is unstable, and therefore the solution is inaccurate. It's *not* that the solution was accurate up to time .6 and then became "bad". It was always bad. The simulation must be repeated with a smaller timestep.

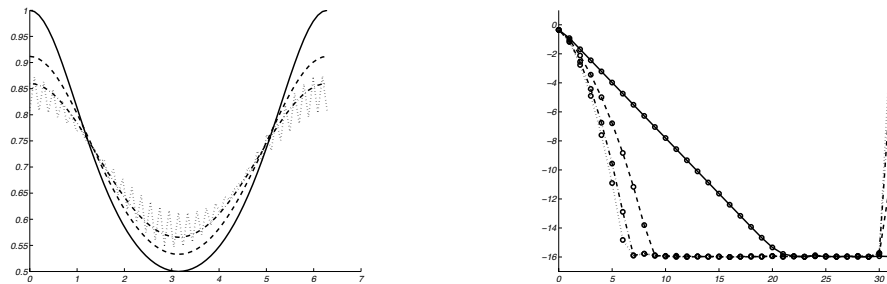


Figure 2: Everything is the same as for Figure 1 except the time step is larger: $h = 1/450$. Left plot: The approximate solutions are shown at times 0 (heavy solid), .2511 (heavy dashed), .5 (heavy dot-dashed), and .62 (heavy dotted). Initially, the maxima and minima are decreasing monotonically in time, as expected. However, shortly before time .6, the maxima stop decreasing and spurious oscillations become visible to the eye. Right plot: Plotted here is the power spectrum ($k, \log_{10}(|\hat{u}_k(t)|)$) at the same times with the same line types. The highest frequency which was initially at the level of roundoff immediately starts to grow. Around time .6, its amplitude is $\mathcal{O}(1)$ and the effect is visible in the solution, as shown. However, the solution was corrupted immediately and the solution was immediately inaccurate. Had the time step been larger than more than one of the high frequencies would have grown due to numerical instability.

The right plot of Figure 2 shows one type of numerical instability. For

some problems, you may observe an instability in which *all* of the coefficients that were at round-off start to grow. Not just the higher ones. This can happen when computing the motion of a vortex sheet in 2-d Euler flow, for example. There is no cunning choice of time-stepping that will avoid this — the physical problem is unstable. In such a case, a “Krasny filter” or a spectral filter¹ of some sort may be of help in controlling the spurious growth.

Note: because the initial data u_0 is real-valued, the negative Fourier coefficients will have the same magnitude as the positive Fourier coefficients. This is why I only plot $\log_{10}(|\hat{u}_k|)$ for $0 \leq k < n/2$. If u_0 were complex-valued then I would consider the power spectrum for $-n/2 < k < n/2$.

3.3 Some Comments on Integrating Factors

Let us return to the time-stepping scheme

$$\hat{u}_\ell^{n+1} = e^{-D\ell^2(2\pi/L)^2 k} \hat{u}_\ell^n.$$

Cumulatively, this is the same thing as the exact solution. After n timesteps we have

$$\hat{u}_\ell^n = e^{-D\ell^2(2\pi/L)^2 nk} \hat{u}_\ell^0 = e^{-D\ell^2(2\pi/L)^2 t_n} \hat{u}_\ell^0.$$

For this reason, this time-stepping produces the exact solution; there’s no approximation involved.

This is the simplest example of an integrating factor approach. These approaches can be very powerful for more difficult equations. First of all, consider the ODE

$$u_t = -D\ell^2 u + f(u, t). \quad (8)$$

This has the exact solution

$$u^{n+1} = u(t_n + k) = e^{-D\ell^2 k} u(t_n) + e^{-D\ell^2 k} \int_{t_n}^{t_{n+1}} e^{D\ell^2(s-t_n)} f(u(s), s) ds.$$

. We now need to approximate the integral

$$\int_{t_n}^{t_{n+1}} e^{D\ell^2(s-t_n)} f(u(s), s) ds.$$

¹For example, see “Computing nearly singular solutions using pseudo-spectral methods” by T. Y. Hou and R. Li in *J. Comput. Phys* 226(2007)379–397.

If we do a Taylor expansion on the integrand about $s = t_n$, the integral equals

$$\int_{t_n}^{t_{n+1}} \left(1 + D\ell^2(s - t_n) + \text{H.O.T.} \right) \cdot (f(u(t_n), t_n) + [f_u(u(t_n), t_n)u_t(t_n) + f_t(u(t_n), t_n)](s - t_n) + \text{H.O.T.}) \cdot e^{D\ell^2(s-t_n)} f(u(s), s) ds.$$

To lowest order, this is approximated by

$$\int_{t_n}^{t_{n+1}} f(u(t_n), t_n) ds = kf(u^n, t_n).$$

In this case, the time-stepping would be

$$u^{n+1} = e^{-D\ell^2 k} u^n + e^{-D\ell^2 k} kf(u^n, t_n) = e^{-D\ell^2 k} (u^n + kf(u^n, t_n)). \quad (9)$$

(Note: from the above, you see that the error induced by the timestepping is caused by how I approximated that integral. The local truncation error will be $\mathcal{O}(k^2)$ leading to a global truncation error of $\mathcal{O}(k)$. For a higher-order approximation, I would need a better approximation of that integral.)

How should you understand (9)? If we change variables by introducing

$$v(t) = e^{D\ell^2 t} u(t)$$

then v satisfies the ODE

$$v_t = e^{D\ell^2 t} f(e^{-D\ell^2 t} v, t) = \tilde{f}(v, t)$$

If we timestep this ODE with explicit Euler

$$v^{n+1} = v^n + k\tilde{f}(v^n, t_n) \quad (10)$$

and transform back to u we find that the time-stepping corresponds to

$$u^{n+1} = e^{-D\ell^2 k} (u^n + kf(u^n, t_n))$$

which is precisely the integrating factor time-stepping (9).

And so by reformulating the problem we have removed the stiffness caused by the $-D\ell^2 u$ term in (8) which forced us to take small timesteps when doing an explicit scheme. Indeed, the time-stepping (9) will only require us to take time steps that are small enough to resolve the timescales

involved in the nonlinearity. Alternately, we can do the time-stepping (10) and then recover $u(T)$ from $v(T)$ whenever we want to print out a solution (which may not be at every time step). Either way, we avoid the time-step constraints that would have been forced upon when explicitly time-stepping the ODE for u — we side-stepped the pesky $-D^2u$ term.

How does this work for a PDE? Consider

$$u_t = -\mathcal{L}u + N(u)$$

where $N(u)$, which may be nonlinear, has fewer derivatives in it than the linear operator \mathcal{L} . The linear operator \mathcal{L} may have coefficients that depend on space, making spectral methods inapplicable. We rewrite the PDE as

$$u_t = -\mathcal{L}_0u + (\mathcal{L}_0 - \mathcal{L})u + N(u)$$

where \mathcal{L}_0 is a cunningly-chosen constant-coefficient linear operator (for which spectral methods apply). For example, we would like \mathcal{L}_0 to have the same behaviour as \mathcal{L} when it comes to high frequencies in which case $\mathcal{L}_0 - \mathcal{L}$ would be effectively a 0th order operator — it would act as if it took no derivatives at all, at least as far as the high frequencies are concerned. This would be as if \mathcal{L}_0 were the homogenized or “effective” operator for \mathcal{L} . Finding \mathcal{L}_0 requires careful, problem-specific thought.

In any case, given \mathcal{L}_0 , we introduce

$$v(x, t) = e^{\mathcal{L}_0 t} u(x, t)$$

which solves

$$v_t = (\mathcal{L}_0 - \mathcal{L})v + e^{\mathcal{L}_0 t} N(e^{-\mathcal{L}_0 t} v) = (\mathcal{L}_0 - \mathcal{L})v + \tilde{N}(v). \quad (11)$$

We choose an explicit time-stepping scheme for (11), one which has no time-step constraint (or has a less-severe one) and then transform everything back to a time-stepping scheme for u . This would result in a time-stepping scheme for u which has an integrating factor as well as a discrete approximation part, much as (9) does.

This type of approach is called “stiffness reduction via integrating factors” and has been tremendously successful for a variety of problems. For example, one can now quickly and accurately compute solutions for certain fluid problems that are high-order due to the presence of surface tension. See T. Y. Hou, J. S. Lowengrub, M. J. Shelley, “Removing the stiffness from interfacial flows with surface tension” *J. Comput. Phys.* 114(1994)312–338.