CS 542G: Proving Finite Differences, Variational Form of Poisson

Robert Bridson

November 12, 2008

1 A Model Problem

Last time we introduced the finite difference method for solving the Poisson problem. This lecture we're going to analyze how well this works: do we actually get a good approximation to the true solution? Even before that, is our discretized problem well-posed?

To simplify matters, we'll just work on a 1D model problem, solving the Poisson problem on the interval $[0,1]$ with Dirichlet boundary conditions:

$$\begin{aligned} u''(x) &= f(x) \qquad 0 < x < 1 \\ u(0) &= u_l \\ u(1) &= u_r \end{aligned}$$

The analysis today extends without much effort to higher dimensions at least on rectangular domains (more on that at the end) so this simplification is more for pedagogical reasons than making the problem easier.

Define the grid to have $n+1$ intervals of length $\Delta x = \frac{1}{n+1}$, with grid point $x_i = i\Delta x$. The value $u_0 = u(0) = u_l$ is already known, as is $u_{n+1} = u(1) = u_r$, from the boundary conditions so it's just $u_1, \dots,$ u_n that we need to find. The finite difference equations are:

$$\begin{aligned} \frac{u_0 - 2u_1 + u_2}{\Delta x^2} &= f_1 \\ \frac{u_1 - 2u_2 + u_3}{\Delta x^2} &= f_2 \\ &\vdots \\ \frac{u_{n-2} - 2u_{n-1} + u_n}{\Delta x^2} &= f_{n-1} \\ \frac{u_{n-1} - 2u_n + u_{n+1}}{\Delta x^2} &= f_{n-1} \end{aligned}$$

Substituting in the boundary conditions and putting them on the right-hand-side changes the first and last equations to:

$$\begin{aligned}\frac{-2u_1+u_2}{\Delta x^2} &= f_1 - \frac{u_l}{\Delta x^2}\\ \frac{u_{n-1}-2u_n}{\Delta x^2} &= f_{n-1} - \frac{u_r}{\Delta x^2}\end{aligned}$$

We can assemble this linear system of n equations in n unknowns as:

$$Au = f$$

where the u contains the unknown solution values, f contains the function values from the Poisson problem with first and last entry modified by the boundary conditions as above, and the matrix A is:

$$A = -\frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0\\ -1 & 2 & -1 & \cdots & 0 & 0 & 0\\ 0 & -1 & 2 & \cdots & 0 & 0 & 0\\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & \cdots & 2 & -1 & 0\\ 0 & 0 & 0 & \cdots & -1 & 2 & -1\\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

This is an example of a **sparse** matrix, since most of the entries are zero. More specifically, it is a **tridiagonal** matrix since the only nonzero entries are on the main diagonal and the first sub- and superdiagonal.

2 Looking at the Finite Difference Matrix

Our first real question is if the system of equations we have derived is actually solvable. We'll do better, showing that the matrix above is in fact symmetric negative definite—or equivalently that if we take out the $\frac{-1}{\Delta x^2}$ factor, we'll prove it's SPD.

The symmetry is obvious. Let's look at positive definiteness: can we guarantee anything about the sign of $u^T\bar{A}u$ where $\bar{A} = \text{tridiag}(-1, 2, -1)$? Let's write it out:

$$u^T\bar{A}u = u_1(2u_1 - u_2) + u_2(-u_1 + 2u_2 - u_3) + \ldots + u_{n-1}(-u_{n-2} + 2u_{n-1} - u_n) + u_n(-u_{n-1} + 2u_n)$$

This expression can be rearranged as:

$$\begin{aligned}u^T\bar{A}u &= u_1^2 + u_1(u_1 - u_2) + u_2(-u_1 + u_2) + u_2(u_2 - u_3) + \ldots + u_n(-u_{n-1} + u_n) + u_n^2\\ &= u_1^2 + (u_2 - u_1)^2 + \ldots + (u_n - u_{n-1})^2 + u_n^2\end{aligned}$$

This is a sum of squares![1] So clearly $u^T \bar{A} u \geq 0$. If it was zero, then clearly every term needs to be zero—so $u_1 = 0$ and $u_2 - u_1 = 0$ which implies $u_2 = 0$, and $u_3 - u_2 = 0$ which implies $u_3 = 0$, and so on showing u as a whole must be all zero. Therefore $\bar{A}$ is SPD.

We conclude, finally, that the matrix is invertible and therefore the discrete problem is well-posed. On the other hand, we don't have any idea of its condition number, but that can actually be estimated with the technique in the next section with a bit more work (which I'll leave for you, the reader).

3 Von Neumann Analysis

The other issue we'll tackle is checking if the solution to the discretized system is a good approximation to the true solution of the PDE. Last time we saw that the true solution satisfies the discrete system to within an error of $O(\Delta x^2)$; this time we'll try to show it goes the other way too. Our tool of choice is something called **Von Neumann analysis**, which generally applies to understanding linear, constant-coefficient differential equations on regular grids and rectangular domains, using Fourier analysis. (Under the hood, the reason Fourier analysis is useful in this case is that the eigenvectors of this sort of matrix happen to always be Fourier modes.)

Underlying the Fourier transform is the idea of representing a function (or in this case, a vector of discrete points) as a sum of complex exponentials. Thanks to the linearity of the system, we can do this one mode at a time—consider them separately instead of in the full sum at once. We'll also simplify our lives by skipping over the issue of boundary conditions and which frequencies actually show up in the discrete Fourier transform. So let's pick a frequency $\omega \in \mathbb{R}$ and look at the corresponding mode of the solution:

$$u_i = U e^{\sqrt{-1}\omega i}$$

Here U is the amplitude of this mode, and I've used $\sqrt{-1}$ for the simplest imaginary number (what more commonly is denoted by i in math and j in engineering—here i and j are just too useful as integer indices to be wasted on imaginary numbers). Similarly this frequency mode of the right-hand-side is:

$$f_i = F e^{\sqrt{-1}\omega i}$$

Putting them together, the i'th finite difference equation gives:

$$\begin{aligned} \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} &= f_i \\ \Rightarrow \quad \frac{U}{\Delta x^2} e^{\sqrt{-1}\omega(i-1)} - 2\frac{U}{\Delta x^2} e^{\sqrt{-1}\omega i} + \frac{U}{\Delta x^2} e^{\sqrt{-1}\omega(i+1)} &= F e^{\sqrt{-1}\omega i} \end{aligned}$$

[1] In fact, if we brought back the Δx^2 factor, we could view this as a sum of the first derivative of u (approximated by finite difference) squared. This is no accident: at the end today we'll show the PDE itself is closely related to something like this.

Taking out the common factor of $e^{\sqrt{-1}\omega i}$ reduces this to:

$$\frac{U}{\Delta x^2}\left(e^{-\sqrt{-1}\omega} - 2 + e^{\sqrt{-1}\omega}\right) = F$$

Simplifying the remaining complex exponentials to $2\cos\omega$ and solving for U gives:

$$U = -\frac{F\Delta x^2}{2(1-\cos\omega)}$$

That's the discrete solution right there.[2]

Let's compare that to the true solution $\bar{u}(x)$ of the original PDE. The Fourier transform works for that as well. Replacing i with $x/\Delta x$, the continuous Fourier mode of the same frequency is

$$\bar{u}(x) = \bar{U}e^{\sqrt{-1}\omega x/\Delta x}$$

and for the right-hand-side is

$$f(x) = Fe^{\sqrt{-1}\omega x/\Delta x}$$

Plugging these into the PDE gives, after differentiation:

$$\begin{array}{rrcl} & \bar{u}'' & = & f \\ \Rightarrow & \bar{U}\left(\sqrt{-1}\dfrac{\omega}{\Delta x}\right)^2 e^{\sqrt{-1}\omega x/\Delta x} & = & Fe^{\sqrt{-1}\omega x/\Delta x} \end{array}$$

Expanding the square term to $-\omega^2/\Delta x^2$, taking out the common factor of $e^{\sqrt{-1}\omega x/\Delta x}$ and solving for $\bar{U}$ gives

$$\bar{U} = -\frac{F\Delta x^2}{\omega^2}$$

This is the true solution!

Incidentally, one may ask why not just use the Fourier transform and this formula for the true solution instead of bothering with finite difference approximations? In fact, this is a popular numerical scheme called the **spectral method**. With the FFT to provide $O(n\log n)$ Fourier transforms, this is both efficient and exquisitely accurate. If the right-hand-side f is band-limited (only finitely many nonzero Fourier coefficients) we get the exact answer for n large enough; more generally the convergence rate is only limited by the smoothness of f. In particular, if f is infinitely differentiable, convergence is faster than any polynomial in Δx—spectacularly good. The real problem limiting its general use is that it only works

[2] In fact, if we did properly take care of boundary conditions and which frequencies are allowed, this leads to a very efficient algorithm for solving the linear system: use the Fast Fourier Transform (FFT) on the right-hand-side, scale it by the right factors as shown, and then use inverse FFT to get the solution u back. This is an $O(n\log n)$ procedure, close to optimal and very hard to beat in 2D or 3D. Interestingly, just simple Cholesky is $O(n)$ in 1D, if the tridiagonal structure is exploited.

when Fourier transforms are applicable: rectangular domains, simple-enough boundary conditions, just the simple Laplacian (no generalizations such as $\nabla \cdot a(x)\nabla u = f$). In these cases finite differences may still work, though of course the von Neumann analysis also breaks down and other methods will be needed to prove convergence.

Back to the problem at hand, we can now look at the error between the finite difference solution and the true solution:

$$U = -\frac{F\Delta x^2}{2(1-\cos\omega)} \qquad \text{vs.} \qquad \bar{U} = -\frac{F\Delta x^2}{\omega^2}$$

Note that Taylor series tells us $\cos\omega \approx 1 - \frac{1}{2}\omega^2 + O(\omega^4)$ (for small ω), so the discrete solution is:

$$\begin{aligned} U &= -\frac{F\Delta x^2}{\omega^2 + O(\omega^4)} \\ &= \bar{U}\frac{1}{1+O(\omega^2)} \\ &= \bar{U}(1+O(\omega^2)) \\ &= \bar{U} + O(F\Delta x^2) \end{aligned}$$

That is, at least the low frequencies of the discrete solution will converge as expected to the true solution, with second order accuracy—the lower the frequency, the better. The high frequencies (where the Taylor series for $\cos\omega$ isn't as useful) aren't covered by the above, but as long as f is reasonably smooth then F will be very small for the large ω and the error bound also ends up small.

Our conclusion: the finite difference method we proposed does in fact work. It produces a well-posed linear system (that can be scaled to be SPD), the discrete solution converges to the true solution with second order accuracy, and the lower frequency components are more accurate than the higher frequencies. However, to prove all this required some creative manipulation of expressions (for $u^T\bar{A}u$) that doesn't obviously extend to other finite difference problems, and a Fourier analysis which only applies for rectangular domains with simple boundary conditions etc. This is somewhat unsatisfying. Finite differences certainly are a useful numerical method, and in some cases provide the most efficient method available for a given problem, but a sound theoretical framework to show when they work in general is not one of their strengths. We'll soon look at an alternative method of discretization which provides an elegant framework for all of this; it also has the advantage of working not just on regular grids but much more general meshes or more, allowing the possibility of adapting the resolution of the method to the solution (putting more "samples" in regions where the solution is interesting and less where it's relatively flat).

4 Numerically Studying Convergence

Before looking at an alternative, it's worth pointing out that finite differences—and many other numerical methods—are applied to problems without a full proof that they work. In fact, sometimes numerical methods are applied to find solutions to problems that aren't even known to be well-posed (for example, three dimensional viscous incompressible fluid flow). Even when we do have a proof of convergence, as above, it usually isn't quite specific enough to actually give us a rigourous upper bound on the error of a particular solution: usually there's an unknown constant lurking in $O()$ notation, with a condition that n has to be "large enough". And finally, there are almost always other errors we haven't explicitly noted: for example, the PDE itself might just be an approximation to reality, and the measured data (such as f or the geometry of the domain) might have errors in it too. This means that in practice, running a single simulation really can't give us much confidence in the results: we just don't know how big the error might be.

However, there are ways we can use multiple simulations to build confidence, even if in the end nothing will be proven rigourously. In some cases, we may have a simpler but related experiment that can be performed and measured in real life: if the result of the numerical method agrees well with real data, we gain a measure of confidence it will similarly give good results for the problem we really care about. Similarly there may be special model problems (such as the Poisson problem on a rectangle) where we do have the exact solution, and can evaluate the true error to get a feel for how the method really performs.

Especially in the absence of real data or known solutions, but recommended in every scenario, we can turn to a **numerical convergence study**. Here we run the simulation with a particular grid spacingΔx, then re-run it with grid-spacing $\frac{1}{2}\Delta x$, then with $\frac{1}{4}\Delta x$, and continue halving until we run out of computing power or think we've done enough. (Other geometric sequences can work, of course, but powers of one half are generally favoured.) If we don't have an exact solution or well-trusted approximate solution available, we make the assumption that the discrete solution from the finest grid is close enough. We can then measure the error or an approximation thereof at each grid size. Generally we expect on very coarse grids that the error will be large and vary somewhat erratically, but at some point—when Δx is smaller than some threshold value—we will enter the "convergence regime" and expect to see the error decrease at a polynomial rate. If the measured error values do match a polynomial—e.g. if they are reduced by roughly a factor of four when we halve the grid size, that would match $C\Delta x^2$—then we make the leap of faith that we are seeing convergence and there will be no surprises at even smaller grid sizes. Nothing guarantees this, but experience suggests the world usually isn't out to get us, and we can trust these results—particularly if we have some sort of theoretical argument justifying the observed convergence rate.

5 The Variational Form of the Poisson Problem

As a prelude to an alternative discretization, let's rework the Poisson problem into another form, one we have actually brushed up against long ago when looking at RBF interpolation. We'll show it's equivalent to finding the function which minimizes a particular integral; this is called a **variational form**.

We'll work with the following simplified Poisson problem:

$$\begin{aligned} \nabla \cdot \nabla u(x) &= f(x) \qquad \text{for } x \in \Omega \\ u(x) &= 0 \qquad \text{for } x \in \partial\Omega \end{aligned}$$

The domain Ω isn't limited to rectangles anymore. We are now specifying a zero Dirichlet condition on the boundary $\partial\Omega$; nonzero Dirichlet conditions aren't much harder, and Neumann conditions are actually easier in the variational form, but we'll skip those for the purposes of this course.

An equivalent minimization problem is:

$$\min_{u : u|_{\partial\Omega}=0} \int_\Omega \|\nabla u\|^2 + 2fu$$

That is, of all functions which are zero on the boundary, find the one which minimizes an integral involving the norm of the gradient squared. Refer back to our early work on RBFs, where we tried to define an interpolant which is as smooth as possible, for comparison. We'll use the same arguments to show the solution of the minimization solves the PDE.

Assume u *is* the minimum; now take any v that's also zero on the boundary and look at the integral for $u + \epsilon v$ (which also satisfies the boundary condition) as a function of ϵ:

$$h(\epsilon) = \int_\Omega \|\nabla(u + \epsilon v)\|^2 + 2f(u + \epsilon v)$$

Since u is the minimum, it must be that $h'(0) = 0$ no matter what v is. Let's find $h'(\epsilon)$:

$$\begin{aligned} h'(\epsilon) &= \int_\Omega \frac{\partial}{\partial \epsilon}\|\nabla(u + \epsilon v)\|^2 + \frac{\partial}{\partial \epsilon} 2f(u + \epsilon v) \\ &= \int_\Omega 2\nabla(u + \epsilon v) \cdot \nabla v + 2fv \end{aligned}$$

Evaluating this at $\epsilon = 0$ and setting $h'(0) = 0$ gives:

$$\int_\Omega 2\nabla u \cdot \nabla v + 2fv = 0$$

Now use integration by parts on the first term, to get:

$$\int_\Omega -2(\nabla \cdot \nabla u)v + 2fv - \int_{\partial\Omega} 2\nabla u \cdot \hat{n} v = 0$$

Since $v = 0$ on the boundary $\partial\Omega$, the boundary integral vanishes and, dividing by -2, we are left with:

$$\int_{\Omega} (\nabla \cdot \nabla u - f)v = 0$$

This is true for almost arbitrary v, so we conclude u satisfies the PDE $\nabla \cdot \nabla u = f$.

In a nutshell the **Finite Element Method** (FEM), which we'll start looking at next time, takes this variational form but instead of minimizing over all functions satisfying the boundary condition restricts the minimization to a carefully chosen finite-dimensional function space—making it solvable on the computer. We've seen this idea several times before: approximate the solution of a hard problem by restricting it to a small dimensional subspace where it's easier to solve, which will give a good approximation as long as the subspace can approximate the answer. For example, in Rayleigh-Ritz we went from a big eigenvalue problem (for a large $n \times n$ matrix) to a small $k \times k$ eigenvalue problem by setting up the eigenproblem as a minimization and then restricting to a small k-dimensional space we're interested in. FEM does the same thing, only now "n" is infinite.