Lecture 11

(Finished discussion of Neumann from previous lecture: mass flux, conservation of mass in the diffusion equation, and other conservation laws.)

2d finite-difference discretizations: discretized the 2d Laplacian operator ∇^2 in an $L_x \times L_y$ box with $N_x \times N_y$ points, for Dirichlet (0) boundaries, so that $u(m\Delta x, n\Delta y)$ is approximated by $N_x N_y$ degrees of freedom $u_{m,n}$. Showed that simply applying the 1d center-difference rule along x and y results in a (famous) "5-point stencil" approximation for $-\nabla^2$ in which the Laplacian at (n_x,n_y) depends on u at (n_x,n_y) and the 4 nearest-neighbor points.

In order to express this as a matrix A, however, we need to "flatten" the 2d grid of points $u_{nx,ny}$ into a single column vector **u** with $N_x N_y$ components. There are multiple ways to do this, but a standard and easy scheme is the "column-major" approach in which **u** is formed from the contiguous columns (x) $u_{nx,:}$ concatenated in sequence. (This is the approach used internally within Matlab to store matrices.)

Given this, we then see how to form $\partial^2/\partial x^2$ by operating one N_x-column at a time, using the the $N_x \times N_x$ discrete 1d Laplacian A_x ($=-D_x^T D_x$). The $\partial^2/\partial x^2$ matrix is simply a matrix with A_x along the diagonal N_y times, which differentiates each N_x-column block by block. The $\partial^2/\partial y^2$ matrix is a little tricker, but if we think of operating on whole columns then we see that it is just the A_y matrix with the entries "multiplied" by $N_x \times N_x$ identity matrices I_x.

In order to have a convenient way to express this, we use the Kronecker product notation $A \otimes B$ [kron(A,B) in Matlab], which multiplies the *entries* of A by the *matrix* B to create a *matrix of matrices*. In this notation, $A = I_y \otimes A_x + A_y \otimes I_x$.

Using this machinery, constructed A for $N_x=10$ and $N_y=15$ for $L_x=1$ and $L_y=1.5$ in Julia. Visualized the pattern of nonzero entries with spy. Solved for the eigenfunctions, and plotted a few; to convert a column vector **u** back into a 2d matrix, used reshape(**u**,N_x,N_y), and plotted in 3d with the surf command. The first few eigenfunctions can be seen to roughly match the $\sin(n_x \pi x/L_x)\sin(n_y \pi x/L_y)$ functions we expect from separation of variables. However, $N_x=10$, $N_y=15$ is rather coarse, too coarse a discretization to have a really nice (or accurate) picture of the solutions.

In order to increase N_x and N_y, however, we have a problem. If the problem has $N=N_x N_y$ degrees of freedom, we need to store N^2 numbers ($8N^2$ bytes) just to store the matrix A, and even just solving Ax=b by Gaussian elimination takes about N^3 arithmetic operations. Worked through a few numbers to see that even $N_x=N_y=100$ would have us waiting for 20 minutes and needing a GB of storage, while 3d grids (e.g. 100×100×100) seem completely out of reach. The saving grace, however, is sparsity: the matrix is mostly zero (and in fact the 5-point stencil A has $< 5N$ nonzero entries). This means that, first, you can store only the nonzero entries, greatly reducing storage. Second, it turns out there are ways to exploit the sparsity to solve Ax=b much more quickly, and there are also quick ways to find a *few* of the eigenvalues and eigenvectors.

In Julia, you exploit sparsity by using the `sparse` command and friends to create sparse matrice. Once you have a sparse matrix, Matlab automatically uses algorithms to exploit sparsity if you solve Ax=b by x=A\b and use the `eigs` function to find a few eigenvalues (instead of `eig`).

Starting with the ∇^2 operator on a square grid, showed how we can convert to any other Ω shape with Dirichlet boundaries just by taking a subset of the rows/cols (as in problem 2 of pset 5). Recovered the Bessel solutions for a circular domain.

Further reading: Section 3.5 of the Strang book on 2d finite differences, section 7.1 on sparsity. See, for example min-max theorem in Wikipedia, although this presentation is rather formal. Unfortunately, most of the discussion you will find of this principle online and in textbooks is either (a) full of formal functional analysis or (b) specific to quantum mechanics [where the operator is $A=-\nabla^2+V$ for some "potential-energy" function V(x)].

18.303 Linear Partial Differential Equations: Analysis and Numerics

Fall 2014