Week 15: Wednesday, Dec 2

Logistics

1. The final exam is Monday, Dec 14, 2-4:30 in Olin 218. A practice final has been posted online.
2. Homework 6 is due next Friday, Dec 11.

Reminder: 1D model problem

In the last lecture, we introduced a model problem: a discretized version of the one-dimensional Poisson equation with homogeneous Dirichlet boundary conditions:

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f \text{ for } x \in (0,1) \\ u(0) &= 0 \\ u(1) &= 0 \end{aligned}$$

Let $x_j = j/(n+1)$ for $j = 0, 1, \ldots, n$ be a set of mesh points. We can approximate the second derivative of u at a point by a finite difference method:

$$-\frac{d^2u}{dx^2}(x_j) \approx \frac{-u(x_{j-1}) + 2u(x_j) - u(x_{j+1})}{h^2}$$

where $h = 1/(n+1)$ is the mesh spacing. If we replace the second derivative in the Poisson equation with this finite-difference approximation, we have a scheme for computing $u_j \approx u(x_j)$:

$$\begin{aligned} -u_{j-1} + 2u_j - u_{j-1} &= hf_j \text{ for } 1 \leq j \leq n \\ u_0 &= 0 \\ u_n &= 0 \end{aligned}$$

We can write this approximation as a matrix equation $Tu = h^2 f$, where

$$T = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}.$$

This problem has a known eigendecomposition $T = Z\Lambda Z^T$ where $Z_{ij} = \sqrt{2/(n+1)}\sin(ij\pi/(n+1))$ and $\lambda_j = 2(1-\cos(j\pi h))$. Note that the eigenvectors come from sampling sampled eigenfunctions for the continuous problem (i.e. $Z_{ij} \propto \sin(j\pi x_i)$).

Jacobi iteration

Recall that stationary iterative methods for solving $Ax = b$ have the form

$$Mx^{(k+1)} = Nx^{(k)} + b$$

where $A = M - N$. The error $e^{(k)} = x^{(k)} - x$ is governed by the recurrence

$$e^{(k+1)} = Re^{(k)} = R^k e^{(0)}.$$

Eventual convergence is guaranteed if the spectral radius $\rho(R)$ is less than one (the spectral radius is the largest eigenvalue magnitude). A sufficient, though not necessary, condition for this to hold is that $\|R\| < 1$ in some operator norm.

The Jacobi iteration is one of the simpler examples of a stationary method. In terms of the mesh, we compute a new approximate solution $u^{(k+1)}$ from an old approximate solution $u^{(k)}$ by the formula

$$-u_{j-1}^{(k)} + 2u_j^{(k+1)} - u_{j+1}^{(k)} = h^2 f_j$$

for each j. That is, we sweep through the mesh and update each solution component by solving a local equation under the assumption that all the other components were right in the previous step. This “sweeping” viewpoint is one we might often take when coding the iteration.

In matrix terms, the Jacobi iteration takes M to be the diagonal part of A. In the case of the model problem, this gives us $M = 2I$, $N = 2I - T$, and

$$R = I - T/2 = Z(I - \Lambda/2)Z^T.$$

That is, the eigenvalues of R are $1 - \lambda_j/2 = \cos(j\pi h)$. Thus,

$$\rho(R) = \|R\|_2 = \cos(\pi h) = 1 - \frac{1}{2}(\pi h)^2 + O(h^4).$$

The number of iterations required to cut the error by a fixed constant (say e) is therefore

$$k_* \approx -1/\log\left(\cos(\pi h)\right) = O(h^{-2}) = O(n^2).$$

Therefore, we say the cost to solve the problem is $O(n^2)$ iterations, since it costs $O(n^2)$ steps to compute the error to a given accuracy (where the constant depends on the accuracy). This is not so great for the one-dimensional Poisson problem, since it leads to an $O(n^3)$ overall cost, as each iteration costs $O(n)$. For a d-dimensional Poisson problem with a mesh that is n on a side, though, convergence still takes $O(n^2)$ steps even though there are $N = n^d$ unknowns. For two-dimensional Poisson problems, then, solution by Jacobi iteration costs $O(N^2)$; for three-dimensional problems, solution by Jacobi iteration costs $O(N^{5/3})$.

It's illuminating to look at the behavior of the Jacobi iteration expressed in the eigenbasis. That is, let $\hat{e}^{(k)} = Z^T e^{(k)}$; then

$$\hat{e}_j^{(k)} = \cos(j\pi h)^k \hat{e}_j^{(0)}.$$

For j close to 0 or $n+1$, $\cos(j\pi h)$ is very near one in magnitude; but for values of j in between, the magnitude of $\cos(j\pi h)$ is significantly less than one. Those intermediate components of the error therefore are damped out very quickly by the Jacobi iteration. Noting that increasing values of j correspond to increasing frequency oscillations in the eigenvector, we say that Jacobi is very effective at damping out *intermediate* frequency errors.

A related iteration, *weighted* Jacobi, sets M to be w times the diagonal part of A. The matrix $R_w = M^{-1}N$ for weighted Jacobi is $R_w = Z(1 - w\Lambda/2)Z$, i.e. for the model problem,

$$\hat{e}_j^{(k)} = \left(1 - w + w\cos(j\pi h)\right)^k \hat{e}_j^{(0)}.$$

If we set $w = 2/3$, we have that $|1 - w + \cos(j\pi h)| \leq 1/3$ for $j > (n+1)/2$. That is, weighted Jacobi iteration can significantly reduce all the *high* frequency components of the error.

Coarse grid approximation

Now, consider the case where f is a fairly smooth function, so that we expect the vector u to also correspond to a fairly smooth function. If u is very smooth, we expect to be able to do a good job of locally approximating u by linear interpolants; i.e., $u_j \approx (u_{j-1} + u_{j+1})/2$. This suggests that we could reduce the size of the problem by taking as unknowns only the values u_j for j even, and getting the other values by interpolation; that is, compute an approximation $\hat{u}_j$ of the form

$$\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \\ \hat{u}_4 \\ \hat{u}_5 \\ \hat{u}_6 \vdots \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & & \\ 1 & & \\ \frac{1}{2} & \frac{1}{2} & \\ & 1 & \\ & \frac{1}{2} & \frac{1}{2} \\ & & 1 \\ & & \ddots \end{bmatrix} \begin{bmatrix} \hat{u}_2 \\ \hat{u}_4 \\ \hat{u}_6 \\ \vdots \end{bmatrix}.$$

We can write this more concisely as $\hat{u} = Pv$, where v consists of the even-index entries of $\hat{u}$. Following the ideas that we described when we discussed conjugate gradients, we can choose $\hat{u} = Pv$ to minimize the energy norm of the error by making the residual orthogonal to the span of P; that is,

$$0 = P^T(T\hat{u} - h^2 f) = P^T T P v - h^2 P^T f.$$

In other words, a "coarse" approximate solution is

$$\hat{u} = \left(P(P^T T P)^{-1} P^T\right) (h^2 f).$$

Because the first several eigenvectors of T are well approximated in the range of P, we expect $\hat{u}$ to have small error at *low* frequencies — which is complementary to the behavior of the Jacobi iteration.

Two-grid iteration

Now, let us consider combining the high-frequency error reduction of Jacobi with the low-frequency error reduction of a coarse grid approximation. We can do this by alternating between the two. Starting from an old solution $u^{(\text{old})}$, we update as follows:

1. Reduce the high-frequency error with a weighted Jacobi:

$$u^{(\mathrm{WJ})} = u^{(\mathrm{old})} + \frac{1}{2w}\left(h^2 f - T u^{(\mathrm{old})}\right).$$

2. Reduce low-frequency error with a correction based on the coarse grid:

$$u^{(\mathrm{corrected})} = u^{(\mathrm{WJ})} + \left(P(P^T T P)^{-1} P^T\right)\left(h^2 f - T u^{(\mathrm{WJ})}\right).$$

3. Reduce the high-frequency error again with weighted Jacobi:

$$u^{(\mathrm{new})} = u^{(\mathrm{corrected})} + \frac{1}{2w}\left(h^2 f - T u^{(\mathrm{corrected})}\right).$$

For the model problem and closely related problems, it turns out that this procedure reduces the error by a constant amount *independent of* n! Thus, we expect convergence in a small number of iterations if we use this two-grid procedure alone, and even better convergence if we use it as a preconditioner in a Krylov subspace method (as is done in HW 6).

Multigrid

The most expensive part of this update procedure in the two-grid refinement step is the solution of the coarse grid problem (i.e. the solve with $P^T T P$). The idea of multigrid is to replace this coarse grid solve with an approximate solve using the same two-grid strategy as before. That is, we solve by smoothing the error (killing high frequencies with weighted Jacobi), then projecting to a coarser grid where we again smooth the error, then projecting to a yet coarser grid where we again smooth the error, until we reach a small enough grid that it is cheap to solve the projected problem directly. We then work our way back up to the finer grids, at each step making a correction and doing another pass of smoothing. This progression from fine grid to coarse grid and back is a *multigrid V cycle.*

The multigrid V-cycle is sufficiently close to the two-grid procedure that it still reduces the error at each step by a fraction that does not depend on n. But unlike the two-grid procedure, the full multigrid cycle only takes $O(n)$ time per step. For the one-dimensional model problem, for example, we would require $O(n)$ work at the finest mesh; $O(n/2)$ work at the next finest mesh; $O(n/4)$ work next; and so on. But $\sum_{j=0}^{\infty} n2^{-j} = 2n$, so the overall work per iteration is only $O(n)$. A similar cost analysis holds in the higher dimensions cases. Therefore, we expect multigrid to converge in a constant number of iterations, each requiring $O(n)$ work; so multigrid converges in $O(n)$ time, and is thus asymptotically optimal... at least for problems that look enough like the model problem.