Week 12: Friday, Nov 9

Iterative methods for the 2D model problem

On Wednesday, we got as far as discussing the cost of various direct methods for the model problem. This time, let's discuss iterative methods.

By nature, an iterative method produces a (hopefully convergent) sequence of approximations to the true answer to a problem. So what do we mean when, for example, we write that Jacobi iteration "solves" the 2D model problem in $O(N^2)$ time? When we talk about solution time, by convention, we mean the time to solution as the time to reduce the initial error by some constant factor. For convenience, that constant factor is generally $1/e$. A stationary method which multiplies the error by R at each step will asymptotically multiply the error at each step by $\rho(R)$, so after k steps of the iteration, the error typically behaves like

$$\|e_k\| \leq C\|e_0\|\rho(R)^k$$

and the time to reduce the error to e^{-1} behaves like

$$k_* = \frac{-1 - \log(C)}{\log(\rho(R))} = O(\log(\rho(R))^{-1})$$

and if $\delta = 1 - \rho(R)$ is small, then

$$-\log(\rho(R))^{-1} = \delta^{-1} + O(1).$$

Thus, we will generally say the "number of iterations to convergence" for an iterative method is $O(\delta^{-1})$.

In the case of the 2D model problem, recall that the eigenvalues are

$$\lambda_{i,j} = 2\left(2 - \cos(\pi i h) - \cos(\pi j h)\right)$$

The extreme eigenvalues are

$$\lambda_{1,1} = 2h^2\pi^2 + O(h^4)$$

and

$$\lambda_{n,n} = 4 - 2h^2\pi^2 + O(h^4).$$

The diagonal of $T_{n\times n}$ is simply $4I$, so the Jacobi iteration matrix looks like

$$R = \frac{1}{4}(4I - T_{n\times n}),$$

for which the eigenvalues are

$$\lambda_{i,j}(R) = -(\cos(\pi ih) + \cos(\pi jh))/2,$$

and the spectral radius is

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

Thus, the number of iterations to reduce the error by $1/e$ scales like

$$\frac{2}{\pi^2 h^2} = \frac{2}{\pi^2}(n+1)^2 = O(N);$$

and since each step takes $O(N)$ time, the total time to reduce the error by a constant factor scales like $O(N^2)$.

The successive overrelaxation iteration uses a splitting

$$M = \omega^{-1}(D - \omega\tilde{L}) = \omega^{-1}D^{-1}(I - \omega L),$$

which yields an iteration matrix

$$R_{SOR} = (I - \omega L)^{-1}((1-\omega)I + \omega U).$$

In general, this is rather awkward to deal with, since it is a nonsymmetric matrix. However, for the model problem with a particular ordering of unknowns (red-black ordering), one has that the eigenvalues μ of R_J correspond to the eigenvalues λ of R_{SOR} via

$$(\lambda + \omega - 1)^2 = \lambda\omega^2\mu^2.$$

For the case $\omega = 1$ (Gauss-Seidel), this degenerates to

$$\lambda = \mu^2,$$

and so $\rho(R_{GS}) = \rho(R_J)^2$. Consequently, each Gauss-Seidel iteration reduces the error by the same amount as two Jacobi iterations, i.e. Gauss-Seidel converges twice as fast on the model problem. This tends to be true for other problems similar to the model problem, too. However, going from Jacobi to Gauss-Seidel only improves the convergence rate by a constant factor; it doesn't improve the asymptotic complexity at all. However optimal ω (about $2-O(h)$) gives us a spectral radius of $1-O(h)$ rather than $1-O(h^2)$, allowing us to accelerate convergence to $O(N^{3/2})$.

Red-black ordering

For Gauss-Seidel and SOR methods, the order in which the variables are processed matters. One order, the *red-black* order, turns out to be particularly convenient for both analysis and implementation. To motivate the red-black order, think of a checkerboard. Each red square only has black squares to its north, south, east, and west; similarly, each black square is only adjacent to red squares. If we are thinking about doing Gauss-Seidel, then, it is convenient to similarly color nodes red or black depending on whether $i + j$ is even or odd, then process all the red squares first followed by all the black squares. Because no red node depends on information from any other red node, the red nodes can be processed in any order without changing results; they just have to be processed before the black nodes. Similarly, the black nodes can be processed in any order, followed by the red nodes.

The red-black ordering can be convenient for parallel implementation, because allowing the red nodes (or black nodes) to be processed in any orer gives more flexibility for different scheduling choices. But it is also a useful choice for analysis. For example, in the red-black ordering, the model problem looks like

$$A = \begin{bmatrix} 4I & B \\ B^T & 4I \end{bmatrix}$$

The preconditioner based on Jacobi iteration is

$$M_J = \begin{bmatrix} 4I & 0 \\ 0 & 4I \end{bmatrix},$$

which results in the iteration matrix

$$R_J = M_J^{-1}(M_J - A) = \frac{1}{4} \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}.$$

The eigenvalues of R_J are thus plus or minus one quarter the singular values of B. Note that this much would have been the same for more general problems with the same structure!

I did not drag you in class through the rest of the analysis, and I would not expect you to repeat it on an exam. Nonetheless, it may be worth writing it out in order to satisfy the curious. The preconditioner for Gauss-Seidel is

$$M_{GS} = \begin{bmatrix} 4I & 0 \\ B^T & 4I \end{bmatrix};$$

and because of the relatively simple form of this matrix, we have

$$M_{GS}^{-1} = \frac{1}{4}\begin{bmatrix} I & 0 \\ B^T/4 & I \end{bmatrix}.$$

The iteration matrix for Gauss-Seidel is

$$R_{GS} = M_{GS}^{-1}(M_{GS} - A) = \begin{bmatrix} 0 & B/4 \\ 0 & -\frac{1}{16}B^TB \end{bmatrix},$$

which has several zero eigenvalues together with some eigenvalues that are minus 1/16 times the squared singular values of B^TB. Thus, as indicated earlier, the spectral radius of R_{GS} is the square of the spectral radius of R_J (for the model problem).

The analysis for the general SOR case is slightly messier, but I'll include it here for completeness. The preconditioner is

$$M_{SOR} = \frac{1}{\omega}\begin{bmatrix} 4I & 0 \\ \omega B^T & 4I \end{bmatrix},$$

and the inverse is

$$M_{SOR}^{-1} = \frac{\omega}{4}\begin{bmatrix} I & 0 \\ -\omega B^T/4 & I \end{bmatrix},$$

The iteration matrix is

$$\begin{aligned} R_{SOR} &= \frac{1}{4}\begin{bmatrix} I & 0 \\ -\omega B^T/4 & I \end{bmatrix}\begin{bmatrix} 4(1-\omega)I & -\omega B \\ 0 & 4(1-\omega)I \end{bmatrix} \\ &= \begin{bmatrix} (1-\omega)I & -\omega B/4 \\ -(1-\omega)\omega B^T/4 & \omega^2 B^TB/16 + (1-\omega)I \end{bmatrix}. \end{aligned}$$

If λ is any eigenvalue of R_{SOR} except $1-\omega$, we can do partial Gaussian elimination on the eigenvalue equation

$$(R_{SOR} - \mu I)v = 0;$$

after eliminating the first block of variables, we have the residual system

$$\left(\frac{\omega^2}{16}B^TB - (\lambda+\omega-1)I - \frac{(1-\omega)\omega^2}{16}B^T((1-\omega-\lambda)I)^{-1}B\right)v_2 = 0,$$

Refactoring, we have

$$\left[\left(\frac{1-\omega}{\lambda+\omega-1}+1\right)\frac{\omega^2}{16}B^TB - (\lambda+\omega-1)I\right]v_2 = 0.$$

From our earlier arguments, letting μ be an eigenvalue of the Jacobi matrix, we know that μ^2 is an eigenvalue of $B^T B/16$. The corresponding eigenvalues λ of R_{SOR} must therefore satisfy

$$\left(\frac{1-\omega}{\lambda+\omega-1}+1\right)\omega^2\mu^2-(\lambda-\omega-1)=0.$$

Multiplying through by $\lambda-\omega-1$, we have

$$(1-\omega+\lambda+\omega-1)\omega^2\mu^2-(\lambda-\omega-1)^2=0$$

or

$$\lambda\omega^2\mu^2=(\lambda-\omega-1)^2,$$

which is the formula noted before.