CME 302: NUMERICAL LINEAR ALGEBRA
FALL 2005/06
LECTURE 13

GENE H. GOLUB

1. Iterative Methods

Very large problems (naturally sparse, from applications): *iterative methods*. Structured matrices (even sometimes dense, but use iterative techniques).

Given $A\mathbf{x} = \mathbf{b}$, write $M\mathbf{x} = N\mathbf{x} + \mathbf{b}$ and construct the iteration

$$M\mathbf{x}^{(k+1)} = N\mathbf{x}^{(k)} + \mathbf{b}.$$

Subtracting these equations, we obtain

$$M(\mathbf{x} - \mathbf{x}^{(k+1)}) = N(\mathbf{x} - \mathbf{x}^{(k)}).$$

Therefore if we denote the error in $\mathbf{x}^{(k)}$ by $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$, then

$$\mathbf{e}^{(k+1)} = M^{-1}N\mathbf{e}^{(k)} \equiv B\mathbf{e}^{(k)}.$$

Thus $\mathbf{e}^{(k)} = B^k\mathbf{e}^{(0)}$, which suggests the following theorem:

Theorem $\mathbf{e}^{(k)} \to 0$ as $k \to \infty$ for all $\mathbf{e}^{(0)}$ if and only if $\rho(B) < 1$.

Convergence can still occur if $\rho(B) = 1$, but in that case we must be careful in how we choose $\mathbf{x}^{(0)}$.

Note that from $\mathbf{e}^{(k)} = B^k\mathbf{e}^{(0)}$, it follows that

$$\frac{\|\mathbf{e}^{(k)}\|}{\|\mathbf{e}^{(0)}\|} \leq \|B\|^k.$$

2. The Jacobi Method

We now develop a simple iterative method. If we rewrite $A\mathbf{x} = \mathbf{b}$ as

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, \ldots, n,$$

then

$$a_{ii}x_i = b_i - \sum_{i \neq j} a_{ij}x_j,$$

or

$$x_i = \frac{1}{a_{ii}}\Big[b_i - \sum_{j \neq i} a_{ij}x_j\Big].$$

Date: November 29, 2005, version 1.0.

Notes originally due to James Lambers. Minor editing by Lek-Heng Lim.

In other words,

$$M = \begin{bmatrix} a_{11} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix}, \quad N = -\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}.$$

Our iteration is therefore

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\Big[b_i - \sum_{j \neq i} a_{ij} x^{(k)}\Big],$$

known as the *Jacobi method*, with

$$M^{-1}N = \begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \cdots & \frac{a_{n,n-1}}{a_{nn}} & 0 \end{bmatrix} \equiv B_J.$$

So, if

$$\|M^{-1}N\|_\infty = \max_{1 \leq i \leq n} \sum_{j \neq i} \left| \frac{a_{ij}}{a_{ii}} \right| < 1,$$

i.e. if B_J is *strictly diagonally dominant*, then the iteration converges.

For example, suppose

$$A = \begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix}.$$

Then $\|B_J\|_\infty = \frac{1}{2}$, so the Jacobi method converges rapidly. On the other hand, if

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix},$$

which arises from discretizing the Laplacian, then $\|B_J\|_\infty = 1$. A more subtle analysis can be used to show convergence, but it is slow.

Note that for these two examples, $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(1)}$ when *all* elements of $\mathbf{x}^{(1)}$ have been computed. This is a waste of storage; we need only $n + 2$ elements of storage of A above. This shows that the ordering of equations is *very* important. If we reorder the equations in such a way that odd-numbered equations and even-numbered equations are grouped separately, then we obtain, for the

latter example,

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

Then, we can solve for all odd indices, then all even indicies, independently of each other. Not only does this approach save storage space but it also lends itself to parallelism.

3. The Gauss-Seidel Method

In the Jacobi method, we compute $x_i^{(k+1)}$ using the elements of $\mathbf{x}^{(k)}$, even though $x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ are already known. The *Gauss-Seidel* method is designed to take advantage of the latest information available about $\mathbf{x}$:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right].$$

To derive this method, we write $A = L + D + U$ where

$$L = \begin{bmatrix} 0 & & & \\ a_{21} & \ddots & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}, \quad D = \begin{bmatrix} a_{11} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix}.$$

Thus the Gauss-Seidel iteration can be written as

$$D\mathbf{x}^{(k+1)} = \mathbf{b} - L\mathbf{x}^{(k+1)} - U\mathbf{x}^{(k)},$$

or

$$(D + L)\mathbf{x}^{(k+1)} = \mathbf{b} - U\mathbf{x}^{(k)}$$

which yields

$$\mathbf{x}^{(k+1)} = -(D + L)^{-1} U\mathbf{x}^{(k)} + (D + L)^{-1}\mathbf{b}.$$

Thus the iteration matrix for the Gauss-Seidel method is $B_{GS} = -(D + L)^{-1}U$, as opposed to the iteration matrix for the Jacobi method, $B_J = -D^{-1}(L + U)$. In some cases, $\rho(B_{GS}) = (\rho(B_J))^2$, so the Gauss-Seidel method converges twice as fast. On the other hand, note that Gauss-Seidel is very sequential; i.e. it does not lend itself to parallelism.

4. Poisson's Equation

Consider the standard problem of solving Poisson's equation on a domain R in two dimensions,

$$-\Delta u = f, \quad (x, y) \in R, \quad \Delta u = u_{xx} + u_{yy},$$

$$u = g, \quad (x, y) \in \partial R.$$

We take R to be the unit rectangle $[0, 1] \times [0, 1]$ and discretize the problem using a uniform grid with spacing $h = 1/(N + 1)$ in the x and y directions, and gridpoints $x_i = ih$, $i = 0, \ldots, N + 1$,

and $y_j = jh$, $j = 0, \ldots, N+1$. Then, for $i, j = 1, \ldots, N$, we replace the differential equation by a difference approximation

$$\frac{-u_{i-1,j} + 2u_{ij} - u_{i+1,j}}{h^2} + \frac{-u_{i,j+1} + 2u_{ij} - u_{i,j+1}}{h^2} = f_{ij},$$

where $u_{ij} = u(x_i, y_j)$ and $f_{ij} = f(x_i, y_j)$. From the boundary conditions, we have

$$u_{0j} = g(x_0, y_j), \quad j = 1, 2, \ldots, N,$$

and similar conditions for the other gridpoints along the boundary.

Let $\mathbf{u}_j = [u_{1j}, \ldots, u_{Nj}]^\top$. Then

$$-\mathbf{u}_{j-1} + T\mathbf{u}_j - \mathbf{u}_{j+1} = \tilde{\mathbf{f}}_j$$

where

$$T = \begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -14 & \end{bmatrix}, \quad [\tilde{\mathbf{f}}_j]_i = \begin{cases} h^2 f_{1j} + g(x_0, j) & i = 1 \\ h^2 f_{ij} & i = 2, \ldots, N-1 \\ h^2 f_{Nj} + g(x_N, j) & i = N \end{cases}$$

Thus we can solve the problem on the entire domain by solving $A\mathbf{u} = \tilde{\mathbf{f}}$ where

$$A = \begin{bmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & T \end{bmatrix}.$$

We say that A is a *block tridiagonal matrix*. A is also a *band* matrix, but the band is sparse and Gaussian elimination may fill-in the whole band. However, the equations can be re-ordered to avoid fill-in.

Department of Computer Science, Gates Building 2B, Room 280, Stanford, CA 94305-9025
E-mail address: golub@stanford.edu