

Mat1062: Computational Methods for PDE

Mary Pugh

February 12, 2008

1 Ownership

These notes are built upon those of Rob Almgren who taught an analogous course in 2003. Whatever you learn of value from them is due to him. All mistakes and sources of confusion are to be blamed on me.

2 Iterative solution methods

If you can't avoid the linear system, then you have to solve it. The "classical" methods apply to the matrix equality problem $Au = b$. For this standard problem, there are many methods available, both direct and iterative: for example, LU decomposition if A is tridiagonal.

We are now going to talk about "classical iterative methods." In practice, these methods have largely been superseded by more modern ones, such as conjugate gradient for symmetric problems, GMRES for asymmetric problems, or multigrid for problems arising from finite difference simulation.

The main idea is to choose some other matrix B , and write $Au = b$ as

$$Bu + (A - B)u = b.$$

We construct an sequence of approximate solutions $u^{(0)}, u^{(1)}, \dots$ by

$$Bu^{(k+1)} + (A - B)u^{(k)} = b,$$

or

$$u^{(k+1)} = Mu^{(k)} + B^{-1}b, \quad \text{with } M = I - B^{-1}A.$$

If this sequence converges to a limit u_* then we know

$$u^{(k)} \rightarrow u_* \quad \implies \quad Mu^{(k)} \rightarrow Mu_* \text{ and } u^{(k+1)} \rightarrow u_*.$$

And therefore

$$\mathbf{u}_* \leftarrow \mathbf{u}^{(k+1)} = M\mathbf{u}^{(k)} + B^{-1}\mathbf{b} \rightarrow M\mathbf{u}_* + B^{-1}\mathbf{b}.$$

This shows that \mathbf{u}_* satisfies $\mathbf{u}_* - M\mathbf{u}_* = B^{-1}\mathbf{b}$ which is equivalent to satisfying $A\mathbf{u}_* = \mathbf{b}$.

Iterative methods never give the exact solution of $A\mathbf{u} = \mathbf{b}$ because the iteration cannot continue to infinity. What iterative methods can give you is something which is ϵ -close to the true solution. Of course, if you're using matlab (or fortran or C) then you always have round-off error anyway and so you can't get the exact solution even if you used a direct method like Gaussian elimination or an LU decomposition.

In constructing an iterative method, we hope that the sequence will be easy to compute, and that it will converge rapidly. For this to be true, B must have two properties:

- B must be easily invertible; and
- B must be close to A , in the sense that M has small eigenvalues.

The problem of approximating a given matrix A by a “preconditioner” B which is easily invertible comes up in many situations. Of course, you still have to be able to apply A in the forward direction. (This may sound silly but for large, full matrices just doing matrix multiplication to compute $A\mathbf{u}^{(k)}$ can be very slow.)

If $\|M\| < 1$ then the iteration will certainly converge. If $\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \mathbf{u}_*$ denotes the error, then $\mathbf{e}^{(k+1)} = M\mathbf{e}^{(k)}$, and so $\|\mathbf{e}^{(k+1)}\| \leq \|M\| \|\mathbf{e}^{(k)}\|$. It follows that $\|\mathbf{e}^{(k)}\| \leq \|M\|^k \|\mathbf{e}^{(0)}\|$. This shows that $\|\mathbf{e}^{(k)}\|$ goes to zero as $k \rightarrow \infty$, proving convergence.

The iteration will converge if and only if the *spectral radius* of M , $\rho(M)$ is less than 1

$$\rho(M) = \max_{1 \leq i \leq n} |\lambda_i|$$

where M is an $n \times n$ matrix with real or complex entries. Having a spectral radius less than 1 implies convergence because as $k \rightarrow \infty$, $\mathbf{e}^{(k)} \sim C\lambda_{\max}^k \mathbf{v}_{\max}$, where λ_{\max} is the dominant eigenvalue (the eigenvalue of largest magnitude) and \mathbf{v}_{\max} is the corresponding eigenvector.

The actual rate of convergence is thus given by $\rho(M)$ rather than $\|M\|$. Since $\rho(M) \leq \|M\|$, this is a sharper and more informative condition —

there are matrices for which $\rho(M) < 1 < \|M\|$. For example,

$$\begin{pmatrix} \frac{1}{2} & 2 \\ 0 & \frac{1}{2} \end{pmatrix}$$

has $\rho(M) = 1/2$ and $\|M\| = 9/4 + \sqrt{5}$. Here I took the L^2 norm of M :

$$\|M\| = \sup_{\vec{x} \in \mathbb{R}^2} \frac{\|M\vec{x}\|}{\|\vec{x}\|} = \sup_{\|\vec{x}\|=1} \|M\vec{x}\|$$

For matrices which arise from finite-difference approximation, you can sometimes get an acceptable approximate inverse B by splitting A into diagonal, lower-triangular, and upper-triangular parts:

$$A = D - L - U,$$

with D diagonal, L strictly lower-triangular, and U strictly upper-triangular (L and U have zeros on the diagonal).

Jacobi method Take $B = D$, so $M_J = D^{-1}(L+U)$. Of course D is trivially invertible since it is diagonal. The iteration may be written

$$u_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} u_j^{(k)} - \sum_{j=i+1}^m A_{ij} u_j^{(k)} \right).$$

It converges if A is *strictly* diagonally dominant: each diagonal element $|A_{ii}|$ is larger than either the sum of all the other elements in the same row or the sum in the same column: $\sum_{j \neq i} |A_{ij}|$ or $|A_{ii}| > \sum_{j \neq i} |A_{ji}|$. The Jacobi iteration does not change if you permute rows and columns of A and u , as long as you keep the same elements on the diagonal. (You can't gain anything by reordering the equations or the unknowns.)

Gauss-Seidel method Take $B = D - L$, so $M_{GS} = (D - L)^{-1}U$. Note that $D - L$ is lower-triangular, hence easily invertible by forward substitution. This is the same as the Jacobi formula, except that you replace elements of $u^{(k)}$ with elements of $u^{(k+1)}$ as soon as you have them, so you can use the same storage for both $u^{(k)}$ and $u^{(k+1)}$. It is written

$$u_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} u_j^{(k+1)} - \sum_{j=i+1}^m A_{ij} u_j^{(k)} \right). \quad (1)$$

where you compute $u_1^{(k+1)}, \dots, u_m^{(k+1)}$ in order. The formula changes if you permute equations and variables, and it can sometimes be advantageous to explore rearrangements of the problem (*e.g.*, “red-black” ordering in 2 or 3 dimensions). The Gauss-Seidel method converges under the same conditions (diagonal dominance) as the Jacobi method, but when they both converge, Gauss-Seidel always converges *faster*, typically twice as fast.