Notes for 2015-03-09

Splittings and Sweeps

In the last lecture, we discussed stationary iterative methods using matrix notation. Specifically, we said a stationary method for the equation $Ax = b$ is associated with a *splitting* $A = M - N$, and convergence (or lack thereof) is associated with the iteration matrix $R = M^{-1}N$. This is the right linear algebraic framework for discussing convergence of these methods, but it is not the way they are usually programmed. The connection between a matrix splitting and a "sweep" of a stationary iteration like Gauss-Seidel or Jacobi iteration is not always immediately obvious, and so it is probably worth spending a moment or two explaining in more detail.

For the sake of concreteness, let's consider a standard model problem: a discretization of a Poisson equation on a line. That is, we approximate

$$-\frac{d^2u}{dx^2} = f, \quad u(0) = u(1) = 0$$

using the second-order finite difference approximation

$$\frac{d^2u}{dx^2} \approx \frac{u(x-h) - 2u(x) + u(x+h)}{h^2}$$

where h is a small step size. We discretize the problem by meshing $[0, 1]$ with evenly spaced points $x_j = jh$ for $j = 0$ to $N + 1$ where $h = 1/(N + 1)$, then apply this approximation at each point. This procedure yields the equations

$$-u_{j-1} + 2u_j - u_{j+1} = h^2 f_j, \quad j = 1, \dots, N$$

or, in matrix notation

$$Tu = h^2 f$$

where u and f are vectors in $\mathbb{R}^N$ representing the sampled (approximate) solution and the sampled forcing function. The matrix T is a frequently-recurring model matrix, the tridiagonal

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

Suppose we forgot about how cheap Gaussian elimination is for tridiagonal matrices. How might we solve this system of equations? A natural thought is that we could make an initial guess at the solution, then refine the solution by "sweeping" over each node j and adjusting the value at that node (u_j) to be consistent with the values at neighboring nodes. In one sweep, we might compute a new set of values u^{new} from the old values u^{old}:

```
for j = 1:N
  unew(j) = (h^2*f(j) + uold(j−1) + uold(j+1))/2;
end
```

or we might update the values for each node in turn, using the most recent estimate for each update, i.e.

```
for j = 1:N
  u(j) = (h^2*f(j) + u(j−1) + u(j+2))/2;
end
```

These are, respectively, a step of *Jacobi* iteration and a step of *Gauss-Seidel* iteration, which are two standard stationary methods.

How should we relate the "sweep" picture to a matrix splitting? The update equation from step k to step $k+1$ in Jacobi is

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

while the Gauss-Seidel update is

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

In terms of splittings, this means that Jacobi corresponds to taking M to be the diagonal part of the matrix,

$$M = \begin{bmatrix} 2 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & \ddots & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix}, \quad N = \begin{bmatrix} 0 & 1 & & & & \\ 1 & 0 & 1 & & & \\ & 1 & 0 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 0 & 1 \\ & & & & 1 & 0 \end{bmatrix},$$

while Gauss-Seidel corresponds to taking M to be the lower triangle of the matrix,

$$M = \begin{bmatrix} 2 & & & & & \\ -1 & 2 & & & & \\ & -1 & 2 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 2 & \\ & & & & -1 & 2 \end{bmatrix}, \quad N = \begin{bmatrix} 0 & 1 & & & & \\ & 0 & 1 & & & \\ & & 0 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & 0 & 1 \\ & & & & & 0 \end{bmatrix}.$$

The point of this exercise is that *programming* stationary iterative methods and *analyzing* the same methods may lead naturally to different ways of thinking about the iterations. It's worthwhile practicing mapping back and forth between these two modes of thought.

Linear Solves and Quadratic Minimization

Last time, we briefly described an argument that Jacobi iteration converges for strictly row diagonally dominant matrices. We now discuss an argument that Gauss-Seidel converges (or at least part of such an argument). In the process, we will see a useful way of reformulating the solution of symmetric positive definite linear systems that will prepare us for our upcoming discussion of conjugate gradient methods.

Let A be a symmetric positive definite matrix, and consider the "energy" function

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b.$$

The stationary point for this function is the point at which the derivative in any direction is zero. That is, for any direction vector u,

$$\begin{aligned} 0 &= \left.\frac{d}{d\epsilon}\right|_{\epsilon=0} \phi(x + \epsilon u) \\ &= \frac{1}{2}u^T Ax + \frac{1}{2}x^T Au - u^T b \\ &= u^T(Ax - b) \end{aligned}$$

Except in pathological instances, a directional derivative can be written as the dot product of a direction vector and a gradient; in this case, we have

$$\nabla\phi = Ax - b.$$

Hence, minimizing ϕ is equivalent to solving $Ax = b$ [1].

Now that we have a characterization of the solution of $Ax = b$ in terms of an optimization problem, what can we do with it? One simple approach is to think of a sweep through all the unknowns, adjusting each variable in term to minimize the energy; that is, we compute a correction Δx_j to node j such that

$$\Delta x_j = \operatorname{argmin}_z \phi(x + z e_j)$$

Note that

$$\frac{d}{dz}\phi(x + z e_j) = e_j^T (A(x + z e_j) - b),$$

and the update $x_j := x_j + \Delta x_j$ is equivalent to choosing a new x_j to set this derivative equal to zero. But this is exactly what the Gauss-Seidel update does! Hence, we can see Gauss-Seidel in two different ways: as a stationary method for solving a linear system, or as an optimization method that constantly makes progress toward a solution that minimizes the energy [2]. The latter perspective can be turned (with a little work) into a convergence proof for Gauss-Seidel on positive-definite linear systems.

Extrapolation: A Hint of Things to Come

Stationary iterations are simple. Methods like Jacobi or Gauss-Seidel are easy to program, and it's (relatively) easy to analyze their convergence. But these methods are also often slow. We'll talk next time about more powerful *Krylov subspace* methods that use stationary iterations as a building block.

There are many ways to motivate Krylov subspace methods. We'll end this lecture by picking one motivating idea that extends beyond the land of linear solvers and into other applications as well. The key to this idea is the observation that the error in our iteration follows a simple pattern:

$$x^{(k)} - x = e^{(k)} = R^k e^{(0)}, \quad R = M^{-1}N.$$

For large k, the behavior of the error is dominated by the largest eigenvalue

[1] If you are unconvinced that this is a minimum, work through the algebra to show that $\phi(A^{-1}b + w) = \frac{1}{2} w^T A w$ for any w.

[2] Later in the class, we'll see this as coordinate-descent with exact line search.

and the associated eigenvector[3], i.e.

$$e^{(k+1)} \approx \lambda_1 e^{(k)}.$$

Note that this means

$$x^{(k)} - x^{(k+1)} = e^{(k)} - e^{(k+1)} \approx (1 - \lambda_1) e^{(k)}.$$

If we have an estimate for λ_1, we can write

$$x = x^{(k)} - e^{(k)} \approx x^{(k)} - \frac{x^{(k)} - x^{(k+1)}}{1 - \lambda_1}.$$

That is, we might hope to get a better estimate of x than is provided by $x^{(k)}$ or $x^{(k+1)}$ individually by taking an appropriate linear combination of $x^{(k)}$ and $x^{(k+1)}$. This idea generalizes: if we have a sequence of approximations $x^{(0)}, \ldots, x^{(k)}$, why not ask for the "best" approximation that can be written as a linear combination of the $x^{(j)}$? This is the notion underlying methods such as the conjugate iteration iteration, which will be the subject (at least in part) of our next lecture.

[3]If you don't understand this now, you will when we talk about the power method in a week or so!