Notes for 2015-03-06

Direct to Iterative

For the past few weeks, we have discussed *direct* methods for solving linear systems and least squares problems. These methods typically involve a factorization, such as LU or QR, that reduces the problem to a triangular solve using forward or backward substitution. These methods run to completion in a fixed amount of time, and are backed by reliable software in packages like LAPACK or UMFPACK.

There are a few things you need to know to be an informed *user* (not developer) of direct methods:

- You need some facility with matrix algebra, so that you know how to manipulate matrix factorizations and "push parens" in order to compute efficiently.
- You need to understand the complexity of different factorizations, and a little about how to take advantage of common matrix structures (e.g. low-rank structure, symmetry, orthogonality, or sparsity) in order to effectively choose between factorizations and algorithms.
- You need to understand a little about conditioning and the relationship between forward and backward error. This is important not only for understanding rounding errors, but also for understanding how other errors (such as measurement errors) can affect a result.

It's also immensely helpful to understand a bit about how the methods work in practice. On the other hand, you are unlikely to have to build your own dense Gaussian elimination code with blocking for efficiency; you'll probably use a library routine instead. It's more important that you understand the ideas behind the factorizations, and how to apply those ideas to use the factorizations effectively in applications.

Iterative methods are different, in that they provide more room for clever, application-specific twists and turns. An iterative method for solving the linear system $Ax = b$ produces a series of guesses

$$\hat{x}^1, \hat{x}^2, \ldots \to x.$$

The goal of the iteration is not always to get the exact answer as fast as possible; it is to get a good enough answer, fast enough to be useful. The rate at which the iteration converges to the solution depends not only on the nature of the iterative method, but also on the structure in the problem. The picture is complicated by the fact that different iterations cost different amounts per step, so a "slowly convergent" iteration may in practice get an adequate solution more quickly than a "rapidly convergent" iteration, just because each step in the slowly convergent iteration is so cheap.

As with direct methods, though, sophisticated iterative methods are constructed from simpler building blocks. In this lecture, we set up one such building block: stationary iterations.

Stationary Iterations

A stationary iteration for the equation $Ax = b$ is typically associated with a *splitting* $A = M - N$, where M is a matrix that is easy to solve (i.e. a triangular or diagonal matrix) and N is everything else. In terms of the splitting, we can rewrite $Ax = b$ as

$$Mx = Nx + b,$$

which is the fixed point equation for the iteration

$$Mx^{k+1} = Nx^k + b.$$

If we subtract the fixed point equation from the iteration equation, we have the error iteration

$$Me^{k+1} = Ne^k$$

or

$$e^{k+1} = Re^k, \quad R = M^{-1}N.$$

We've already seen one example of such an iteration (iterative refinement with an approximate factorization); in other cases, we might choose M to be the diagonal part of A (Jacobi iteration) or the upper or lower triangle of A (Gauss-Seidel iteration). We will see in the next lecture that there is an alternate "matrix-free" picture of these iterations that makes sense in the context of some specific examples, but for analysis it is often best to think about the splitting picture.

Convergence: Norms and Eigenvalues

We consider two standard approaches to analyzing the convergence of a stationary iteration, both of which revolve around the error iteration matrix $R = M^{-1}N$. These approaches involve taking a norm inequality or using an eigenvalue decomposition. The first approach is often easier to reason about in practice, but the second is arguably more informative.

For the norm inequality, note that if $\|R\| < 1$ for some operator norm, then the error satisfies

$$\|e^{k+1}\| \leq \|R\|\|e^k\| \leq \|R\|^k \|e^0\|.$$

Because $\|R\|^k$ converges to zero, the iteration eventually converges. As an example, consider the case where A is strictly row diagonally dominant and M is the diagonal. In that case, $\|R\|_\infty = \|M^{-1}N\|_\infty < 1$. Therefore, the infinity norm of the error is monontonically decreasing[1]

Bounding by one the infinity norm (or two norm, or one norm) of the iteration matrix R is *sufficient* to guarantee convergence, but not *necessary*. In order to completely characterize when stationary iterations converge, we need to turn to an eigenvalue decomposition. Suppose R is diagonalizable, and write the eigendecomposition as

$$R = V\Lambda V^{-1}.$$

Now, note that $R^k = V\Lambda^k V^{-1}$, and therefore

$$\|e^k\| = \|R^k e^0\| = \|V\Lambda^k V^{-1} e^0\| \leq \kappa(V)\rho(R)^k \|e^0\|,$$

where $\rho(R)$ is the *spectral radius* of R, i.e.

$$\rho(R) = \max_{\lambda \text{ an eig}} |\lambda|,$$

and $\kappa(V) = \|V\|\|V^{-1}\|$. For a diagonalizable matrix, convergence of the iteration happens if and only if the spectral radius of R is less than one. *But* that statement ignores the condition number of the eigenvector matrix! For

[1] In finite-dimensional spaces, there is a property of "equivalence of norms" that says that convergence in one norm implies convergence in any other norm; however, this does *not* mean that monotone convergence in one norm implies monotone convergence in any other norm.

highly "non-normal" matrices in which the condition number is large, the iteration may appear to make virtually no progress for many steps before eventually it begins to converge at the rate predicted by the spectral radius. This is consistent with the bounds that we can prove, but often surprises people who have not seen it before.