Week 6: Monday, Mar 5

Iterative and Direct Methods

So far, we have discussed *direct* methods for solving linear systems and least squares problems. These methods have several advantages:

- They are general purpose. It helps to recognize some basic structural properties (sparsity, symmetry, etc), and you need to understand conditioning. Otherwise, you can often trust that MATLAB's backslash operation is doing something reasonable.
- They are robust. More specifically, direct methods are generally backward stable.
- There are good, fast standard libraries.

The main challenges of direct methods involve scaling. Forming and factoring a large matrix can be expensive.

Iterative methods for solving linear systems have a lot of knobs to twiddle, and they often have to be tailored for specific types of systems in order to converge well. But when they are tailored, and when the parameters are set right, they can be very efficient.

A Model Problem

There is a standard model problem for introducing iterative methods for linear systems: a discretized Poisson equation. In lecture, I talked about the two-dimensional case (which is the same case that is in the book); but in order to present the ideas in a simple way, let me write in these notes about the 1D case.

In order to set up this model problem, we need the following approximation: if $u(x)$ is twice differentiable, then

$$u''(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} + O(h^2).$$

We can use this *finite difference approximation* to solve differential equations. For example, suppose we want to approximate the solution to

$$\begin{aligned} -u''(x) &= f(x) \text{ for } 0 \leq x \leq 1 \\ u(0) &= u(1) = 0. \end{aligned}$$

The standard approach would be to sample the interval $[0, 1]$ with a mesh of points ih for $i = 0, 1, 2, \ldots, N+1$ (so $h = 1/(N+1)$), and let $u_i \approx u(ih)$ and $g_i = h^2 f(ih)$. Then

$$\begin{aligned} -u_{i-1} + 2u_i - u_{i+1} &= -g_i \text{ for } i = 1, 2, \ldots, N \\ u_0 &= u_{N+1} = 0. \end{aligned}$$

Listing the equations in order, we have

$$Tu = -g,$$

where T is a tridiagonal matrix with 2 on the main diagonal and -1 on the first sub and superdiagonals. For example, for $N = 5$, we have $T \in \mathbb{R}^{5\times 5}$ given by

$$T = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}.$$

Relax!

Suppose we wanted to solve a system like $Tu = -g$ using an iterative method. That is, we are willing to put aside the machinery we've built for directly solving the system through a factorization, and instead we will construct a sequence of guesses $u^{(k)}$ that will converge to the true solution as $k \to \infty$. How should we do this?

The key point here is that we don't necessarily care that $u^{(k+1)}$ should be the true answer – it should just be more right than $u^{(k)}$. So it is natural to try to *relax* the problem so that we can "fix up" the solution by a little bit at each step. For example, if we believe that $u^{(k)}$ is a good guess, then we might try to fix up $u^{(k+1)}$ by making sure that the variable at each point in the new steps satisfies the balance equation at that same point (assuming

that the neighbor data comes from the old step). That is, for each i we would compute a new approximate solution value using

$$-u_{i-1}^{(k)} + 2u_i^{(k+1)} - u_{i+1}^{(k)} = -g_i.$$

This is *Jacobi* iteration.

Suppose we programmed Jacobi iteration sweeping from $i = 1$ up to $i = N$[1]:

```
% Perform a single Jacobi iteration, computing unew from u
unew(1) = ( u(2)-g(1) )/2;
for i = 2:N-1
  unew(i) = ( u(i-1)+u(i+1)-g(i) )/2;
end
unew(N) = ( u(N-1)-g(1) )/2;
```

Notice that at the time we have computed $u_i^{(k+1)}$ in this code, we have also computed $u_{i-1}^{(k+1)}$. Wouldn't it be better to update $u_i^{(k+1)}$ using this new value, instead of the old one? This natural idea is sometimes called *Gauss-Seidel* iteration:

$$-u_{i-1}^{(k+1)} + 2u_i^{(k+1)} - u_{i+1}^{(k)} = -g_i.$$

When we program a Gauss-Seidel iteration, we can get away with just a single vector for the approximate solution that is overwritten during each sweep:

```
% Perform a Gauss-Seidel sweep, overwriting u with updated guesses
u(1) = ( u(2)-g(1) )/2;
for i = 2:N-1
  u(i) = ( u(i-1)+u(i+1)-g(i) )/2;
end
unew(N) = ( u(N-1)-g(1) )/2;
```

Unfortunately, as we have presented them so far, it seems like it would be a mess to analyze the convergence of either Jacobi or Gauss-Seidel. In order to stay sane during such convergence analysis, we would like a clean notation, and it is to this topic we now turn.

[1] I have assumed the MATLAB vector `u` only holds the active variables $u_1, \ldots, u_N$. If I kept a little extra space for the boundary values $u_0 = u_{N+1} = 0$, I could get rid of the special-case updates for u_1 and u_N.

The Matrix Splitting Perspective

Consider the following general approach to constructing fixed-point iterations to solve $Ax = b$:

1. Split A into two pieces: $A = M - N$. The matrix M should ideally "look like" A, but it should be easy to solve linear systems involving M (where it might not be so easy with A).

2. Iterate on
$$Mx^{(k+1)} = Nx^{(k)} + b,$$
or, equivalently,
$$x^{(k+1)} = x^{(k)} - M^{-1}(Ax^{(k)} - b). \tag{1}$$

The fixed point for the iteration (1) is clearly $x_* = A^{-1}b$. Furthermore, both Jacobi and Gauss-Seidel iteration can be written in terms of a matrix splitting: for Jacobi, we take M to be the diagonal part of A, and for Gauss-Seidel, we take M to be the lower triangular part.

Remember now that we have a general strategy for analyzing the convergence of fixed point iterations, which is to subtract the fixed point equation from the iteration equation in order to get an equation for error propogation. In this case,

$$e^{(k+1)} = e^{(k)} - M^{-1}Ae^{(k)} = (I - M^{-1}A)e^{(k)}.$$

Now, notice that for any consistent choice of norms,

$$\|e^{(k+1)}\| = \|(I - M^{-1}A)e^{(k)}\| \leq \|(I - M^{-1}A)\|\|e^{(k)}\|,$$

so that if $\|I - M^{-1}A\| < 1$, the iteration converges. The converse is not quite true, and in order to make a precise statement about convergence we need to reason about the *spectral radius* of $I - M^{-1}A$. But this norm-based bound is good enough for our present purposes, and we will leave the spectral analysis to another time.