Week 13: Monday, Nov 16

Logistics

1. HW 5 is posted. It is due Nov 25 (but feel free to submit early).
2. The rest of the semester will be various flavors of iterative methods.

Tridiagonal reduction redux

Consider the problem of computing eigenvalues of a symmetric matrix A that is large and sparse. Using the "grail" code in LAPACK, we can compute all the eigenvalues of an n-by-n tridiagonal matrix in $O(n)$ time; but the usual Householder-based algorithm to reduce A to tridiagonal form costs $O(n^3)$. Though it's possible to maintain sparsity in some cases, it is not altogether straightforward. So let's see if we can try something different.

What we want is an orthogonal matrix Q such that $AQ = QT$, where

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Writing column j of this matrix equation gives us

$$Aq_j = \beta_{j-1}q_{j-1} + \alpha_j q_j + \beta_j q_{j+1},$$

where $\alpha_j = q_j^T A q_j$ and $\beta_{j-1} = q_{j-1}^T A q_j$. If we rearrange this slightly, we have

$$\begin{aligned} r_{j+1} &= Aq_j - q_j\alpha_j - q_{j-1}\beta_{j-1} \\ &= Aq_j - (q_j q_j^T)Aq_j - (q_{j-1}q_{j-1}^T)Aq_j \\ q_{j+1} &= r_{j+1}/\beta_j. \end{aligned}$$

That is, q_{j+1} is *exactly* what we compute if we orthogonalize Aq_j against the vectors q_j and q_{j-1}. Note that Aq_j is automatically orthogonal to $q_1, \ldots, q_{j-2}$, since $q_k^T A q_j = T_{kj}$ is zero for $k < j-1$.

This process of incrementally producing an orthonormal basis by multiplying the last vector by A and orthogonalizing the result against what came before gives us the basic Lanczos procedure.

```
function [alpha, beta] = lec32lanczos(A,q1,btol,kmax);
%
% Run a basic Lanczos iteration until either beta_k < btol
% or k == kmax.

k  = 0;
qk = 0;
r  = q1;
b  = 1;
while (b > btol) & (k < kmax)
  k        = k+1;
  qkm1     = qk;
  qk       = r/b;
  Aqk      = A*qk;
  alpha(k) = qk'*Aqk;
  r        = Aqk-qk*alpha(k)-qkm1*b;
  b        = norm(r);
  beta(k)  = b;
end
```

If we start with $q_1 = e_1$, the Lanczos procedure will give us the same T matrix[1] as the Householder algorithm, at least in exact arithmetic. But while a step in the Householder algorithm costs $O(n^2)$, the cost of a Lanczos step is a matrix-vector multiply (which may cost less than $O(n^2)$ if A is sparse) followed by $O(n)$ work in dot products and gaxpy operations. Also, note that the Lanczos step requires only a few vectors of intermediate storage.

In exact arithmetic, we could iterate until we found $\beta_k = 0$, which would correspond to having an invariant subspace spanned by $\{q_1, \ldots, q_k\}$. Except for very special choices of starting vectors, though, this would happen only for $k = n$.

[1] Well, almost the same T matrix. Note that the off-diagonal elements β_j might be positive or negative in the Householder tridiagonalization, but they are always positive in the Lanczos algorithm. However, the two T matrices satisfy a similarity relation with a diagonal matrix whose diagonal entries are ± 1.

Partial tridiagonalization and Krylov subspaces

What happens when we run only a few steps of the Lanczos iteration? After m steps, we have computed m columns of Q and m rows of T; and there is some hope that we might be able to use the leading m-by-m subblock of T (i.e. the block Rayleigh quotient with the first m columns of Q) in order to extract some useful information about the spectrum of A. The reason why we might have such a hope is that we notice that the space spanned by $q_1, \ldots, q_m$ is also spanned by the first m iterates of a power method:

$$\mathcal{K}_m(A, q_1) = \operatorname{span}\{q_1, q_2, \ldots, q_m\} = \operatorname{span}\{q_1, Aq_1, \ldots, A^{m-1}q_1\}.$$

The space $\mathcal{K}_m(A, q_1)$ is a sufficiently useful object that it merits a name: it is the m-dimensional *Krylov subspace* generated by A and q_1. And because it contains the mth vector produced by power iteration starting from q_1, we expect that the largest *Ritz value* associated with $\mathcal{K}_m(A, q_1)$ (i.e. the largest magnitude eigenvalue of the block Rayleigh quotient $(Q_{:,1:m})^T AQ_{:,1:m}$) will be at least as large as the Rayleigh quotient estimate produced by m steps of power iteration.

Of course, the Krylov subspace $\mathcal{K}_m(A, q_1)$ has a lot more information than just the vector from step m of a power iteration. For example, notice that if σ is any shift, then $\mathcal{K}_m(A - \sigma I, q_1) = \mathcal{K}_m(A, q_1)$; that is, Krylov subspaces are *shift invariant*. This is relevant because adding a shift to A can change which of the exterior eigenvalues in the spectrum is dominant. So if $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, then we expect to be able to extract good estimates to λ_1 and λ_n from a few steps of Lanczos iteration under the assumption that power iteration with certain shifted matrices converges sufficiently fast.

This is still not the whole story. But in the interest of sketching a broad picture before painting in details, we will move on for now, and we will return to the discussion of the convergence of Lanczos iterations soon.

Lanczos in inexact arithmetic

The observant reader will have noticed something unsettling about the basic Lanczos iteration: at the heart of the Lanczos iteration is a Gram-Schmidt orthogonalization step, which we said in our discussion of QR factorizations could become numerically instable. In particular, when $r_{k+1} = (A - \alpha_k)q_k + q_{k-1}\beta_{k-1}$ is "small" (i.e. when β_k is small relative to $\|A\|$), cancellation may cause the computed vector to consisty mostly of roundoff, so that it is no longer orthogonal to the preceding vectors. But a small value of β_k corresponds exactly to convergence to a good approximation to an invariant subspace! Thus, in floating point arithmetic, the basic Lanczos iteration tends to run reasonably well until convergence, at which point it effectively restarts with a "random" starting vector which is made up primarily of roundoff error. So we get convergence of the Ritz values of the computed T to the first few exterior eigenvalues; and then more "ghost" Ritz values converge to the first few exterior eigenvalues; and we continue on in this fashion for as long as we are willing to let things run. This not state of affairs is not entirely satisfactory, but we can fix it by *re-orthogonalization*: that is, we improve the orthogonality of the computed basis by explicitly orthogonalizing Aq_k against vectors to which it should already be orthonormal in exact arithmetic. We will go into this in more detail in the next lecture.