Week 10: Wednesday and Friday, Oct 24 and 26

Orthogonal iteration to QR

On Monday, we went through a somewhat roundabout algbraic path from orthogonal subspace iteration to the QR iteration. Let me start this lecture with a much more concise version:

1. The orthogonal iteration $\underline{Q}^{(k+1)} R^{(k)} = A\underline{Q}^{(k)}$ is a generalization of the power method. In fact, the first column of this iteration is *exactly* the power iteration. In general, the first p columns of $\underline{Q}^{(k)}$ are converging to an orthonormal basis for a p-dimensional invariant subspace associated with the p eigenvalues of A with largest modulus (assuming that there aren't several eigenvalues with the same modulus to make this ambiguous).

2. If all the eigenvalues have different modulus, orthogonal iteration ultimately converges to the orthogonal factor in a Schur form

   $$AU = UT$$

   What about the T factor? Note that $T = U^* AU$, so a natural approximation to T at step k would be

   $$A^{(k)} = (\underline{Q}^{(k)})^* A \underline{Q}^{(k)},$$

   and from the definition of the subspace iteration, we have

   $$A^{(k)} = (\underline{Q}^{(k)})^* \underline{Q}^{(k+1)} R^{(k)} = Q^{(k)} R^{(k)},$$

   where $Q^{(k)} \equiv (\underline{Q}^{(k)})^* \underline{Q}^{(k+1)}$ is unitary.

3. Note that

   $$A^{(k+1)} = (\underline{Q}^{(k+1)})^* A^{(k)} \underline{Q}^{(k+1)} = (Q^{(k)})^* A^{(k)} Q^{(k)} = R^{(k)} Q^{(k)}.$$

   Thus, we can go from $A^{(k)}$ to $A^{(k+1)}$ directly without the orthogonal factors from subspace iteration, simply by computing

   $$A^{(k)} = Q^{(k)} R^{(k)}$$
   $$A^{(k+1)} = R^{(k)} Q^{(k)}.$$

   This is the QR iteration.

Hessenberg matrices and QR steps in $O(n^2)$

A matrix H is said to be *upper Hessenberg* if it has nonzeros only in the upper triangle and the first subdiagonal. For example, the nonzero structure of a 5-by-5 Hessenberg matrix is

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}.$$

For any square matrix A, we can find a unitarily similar Hessenberg matrix $H = Q^*AQ$ by the following algorithm (see for comparison the Householder QR code in lecture 18):

```
function [H,Q] = lec27hess(A)
  % Compute the Hessenberg decomposition H = Q'*A*Q using
  % Householder transformations.

  n = length(A);
  Q = eye(n);      % Orthogonal transform so far
  H = A;           % Transformed matrix so far

  for j = 1:n-2

    % -- Find W = I-2vv' to put zeros below H(j+1,j)
    u    = H(j+1:end,j);
    u(1) = u(1) + sign(u(1))*norm(u);
    v    = u/norm(u);

    % -- H := WHW', Q := QW
    H(j+1:end,:) = H(j+1:end,:)-2*v*(v'*H(j+1:end,:));
    H(:,j+1:end) = H(:,j+1:end)-(H(:,j+1:end)*(2*v))*v';
    Q(:,j+1:end) = Q(:,j+1:end)-(Q(:,j+1:end)*(2*v))*v';

  end
```

A Hessenberg matrix H is very nearly upper triangular, and is an interesting object in its own right for many applications. For example, in control

theory, one sometimes would like to evaluate a *transfer function*

$$h(s) = c^T(sI - A)^{-1}b + d$$

for many different values of s. Done naively, it looks like each each evaluation would require $O(n^3)$ time in order to get a factorization of $sI - A$; but if $H = Q^*AQ$ is upper Hessenberg, we can write

$$h(s) = (Qc)^*(sI - H)^{-1}(Qb) + d,$$

and the Hessenberg structure of $sI - H$ allows us to do Gaussian elimination on it in $O(n^2)$ time.

Just as it makes it cheap to do Gaussian elimination, the special structure of the Hessenberg matrix also makes the Householder QR routine very economical. The Householder reflection computed in order to introduce a zero in the $(j+1, j)$ entry needs only to operate on rows j and $j+1$. Therefore, we have

$$Q^*H = W_{n-1}W_{n-2}\ldots W_1H = R,$$

where W_j is a Householder reflection that operates only on rows j and $j+1$. Computing R costs $O(n^2)$ time, since each W_j only affects two rows ($O(n)$ data). Now, note that

$$RQ = R(W_1W_2\ldots W_{n-1});$$

that is, RQ is computed by an operation that first mixes the first two columns, then the second two columns, and so on. The only subdiagonal entries that can be introduced in this process lie on the first subdiagonal, and so RQ is again a Hessenberg matrix. Therefore, one step of QR iteration on a Hessenberg matrix results in another Hessenberg matrix, and a Hessenberg QR step can be performed in $O(n^2)$ time.

Putting these ideas in concrete form, we have the following code

```
function H = lec27hessqr(H)
  % Basic Hessenberg QR step via Householder transformations.

  n = length(H);
  V = zeros(2,n-1);

  % Compute the QR factorization
```

```
for j = 1:n-1

  % -- Find W_j = I-2vv' to put zero into H(j+1,j)
  u      = H(j:j+1,j);
  u(1)   = u(1) + sign(u(1))*norm(u);
  v      = u/norm(u);
  V(:,j) = v;

  % -- H := W_j H
  H(j:j+1,:) = H(j:j+1,:)-2*v*(v'*H(j:j+1,:));

end

% Compute RQ
for j = 1:n-1

  % -- H := WHW', Q := QW
  v = V(:,j);
  H(:,j:j+1) = H(:,j:j+1)-(H(:,j:j+1)*(2*v))*v';

end
```

Inverse iteration and the QR method

When we discussed the power method, we found that we could improve convergence by a spectral transformation that mapped the eigenvalue we wanted to something with large magnitude (preferably much larger than the other eigenvalues). This was the *shift-invert* strategy. We already know there is a connection leading from the power method to orthogonal iteration to the QR method, which we can summarize with a small number of formulas. Let us see if we can follow the same path to uncover a connection from inverse iteration (the power method with A^{-1}, a special case of shift-invert in which the shift is zero) to QR. If we call the orthogonal factors in orthogonal iteration $\underline{Q}^{(k)}$ ($\underline{Q}^{(0)} = I$) and the iterates in QR iteration $A^{(k)}$, we have

$$A^k = \underline{Q}^{(k)} \underline{R}^{(k)} \tag{1}$$

$$A^{(k)} = (\underline{Q}^{(k)})^* A (\underline{Q}^{(k)}). \tag{2}$$

In particular, note that because $R^{(k)}$ are upper triangular,

$$A^k e_1 = (\underline{Q}^{(k)} e_1) r_{11}^{(k)};$$

that is, the first column of $\underline{Q}^{(k)}$ corresponds to the kth step of power iteration starting at e_1. What happens when we consider negative powers of A? Inverting (1), we find

$$A^{-k} = (\underline{R}^{(k)})^{-1} (\underline{Q}^{(k)})^*$$

The matrix $\tilde{R}^{(k)} = (\underline{R}^{(k)})^{-1}$ is again upper triangular; and if we look carefully, we can see in this fact another power iteration:

$$e_n^* A^{-k} = e_n^* \tilde{R}^{(k)} (\underline{Q}^{(k)})^* = \tilde{r}_{nn}^{(k)} (\underline{Q}^{(k)} e_n)^*.$$

That is, the last column of $\underline{Q}^{(k)}$ corresponds to a power iteration converging to a *row* eigenvector of A^{-1}.

Shifting gears

The connection from inverse iteration to orthogonal iteration (and thus to QR iteration) gives us a way to incorporate the shift-invert strategy into QR iteration: simply run QR on the matrix $A - \sigma I$, and the (n, n) entry of the iterates (which corresponds to a Rayleigh quotient with an increasingly-good approximate row eigenvector) should start to converge to $\lambda - \sigma$, where λ is the eigenvalue nearest σ. Put differently, we can run the iteration:

$$\begin{aligned} Q^{(k)} R^{(k)} &= A^{(k-1)} - \sigma I \\ A^{(k)} &= R^{(k)} Q^{(k)} + \sigma I. \end{aligned}$$

If we choose a good shift, then the lower right corner entry of $A^{(k)}$ should converge to the eigenvalue closest to σ in fairly short order, and the rest of the elements in the last row should converge to zero.

The shift-invert power iteration converges fastest when we choose a shift that is close to the eigenvalue that we want. We can do even better if we choose a shift *adaptively*, which was the basis for running Rayleigh quotient iteration. The same idea is the basis for the *shifted QR iteration*:

$$Q^{(k)} R^{(k)} = A^{(k-1)} - \sigma_k I \tag{3}$$

$$A^{(k)} = R^{(k)} Q^{(k)} + \sigma_k I. \tag{4}$$

This iteration is equivalent to computing

$$\begin{aligned}
\underline{Q}^{(k)}\underline{R}^{(k)} &= \prod_{j=1}^{n}(A-\sigma_j I) \\
A^{(k)} &= (\underline{Q}^{(k)})^* A(\underline{Q}^{(k)}) \\
\underline{Q}^{(k)} &= Q^{(k)}Q^{(k-1)}\dots Q^{(1)}.
\end{aligned}$$

What should we use for the shift parameters σ_k? A natural choice is to use $\sigma_k = e_n^* A^{(k-1)} e_n$, which is the same as $\sigma_k = (\underline{Q}^{(k)}e_n)^* A(\underline{Q}^{(k)}e_n)$, the Rayleigh quotient based on the last column of $\underline{Q}^{(k)}$. This simple shifted QR iteration is equivalent to running Rayleigh iteration starting from an initial vector of e_n, which we noted before is locally quadratically convergent.

Double trouble

The simple shift strategy we described in the previous section gives *local* quadratic convergence, but it is not *globally* convergent. As a particularly pesky example, consider what happens if we want to compute a complex conjugate pair of eigenvalues of a real matrix. With our simple shifting strategy, the iteration (3)–(4) will never produce a complex iterate, a complex shift, or a complex eigenvalue. The best we can hope for is that our initial shift is closer to both eigenvalues in the conjugate pair than it is to anything else in the spectrum; in this case, we will most likely find that the last two columns of $\underline{Q}^{(k)}$ are converging to a basis for an *invariant row subspace* of A, and the corresponding eigenvalues are the eigenvalues of the trailing 2-by-2 sub-block.

Fortunately, we know how to compute the eigenvalues of a 2-by-2 matrix! This suggests the following shift strategy: let σ_k be one of the eigenvalues of $A^{(k)}(n-1:n, n-1:n)$. Because this 2-by-2 problem can have complex roots even when the matrix is real, this shift strategy allows the possibility that we could converge to complex eigenvalues. On the other hand, if our original matrix is real, perhaps we would like to consider the *real* Schur form, in which U is a real matrix and T is block diagonal with 1-by-1 and 2-by-2 diagonal blocks that correspond, respectively, to real and complex eigenvalues. If we

shift with *both* roots of $A^{(k)}(n-1:n, n-1:n)$, equivalent to computing

$$\begin{aligned} Q^{(k)}R^{(k)} &= (A^{(k-1)} - \sigma_{k+}I)(A^{(k-1)} - \sigma_{k-}) \\ A^{(k)} &= (Q^{(k)})^* A^{(k-1)} Q^{(k)}. \end{aligned}$$

There is one catch here: even if we started with $A^{(0)}$ in Hessenberg form, it is unclear how to do this double-shift step in $O(n^2)$ time!

The following fact will prove our salvation: if we Q and V are both orthogonal matrices and $Q^T AQ$ and $V^T AV$ are both (unreduced) Hessenberg[1]) and the first column of Q is the same as the first column of V, then all successive columns of Q are unit scalar multiples of the corresponding columns of V. This is the *implicit Q theorem*. Practically, it means that we can do any sort of shifted QR step we would like in the following way:

1. Apply as a similarity any transformations in the QR decomposition that affect the leading submatrix (1-by-1 or 2-by-2).

2. Restore the resulting matrix to Hessenberg form without further transformations to the leading submatrix.

In the first step, we effectively compute the first column of Q; in the second step, we effectively compute the remaining columns. Certainly we compute *some* transformation with the right leading column; and the implicit Q theorem tells us that any such transformation is basically the one we would have computed with an ordinary QR step.

[1] An unreduced Hessenberg matrix has no zeros on the first subdiagonal.