Class Notes, Math 554, Autumn 2012

Lecture XXIV: QR Algorithm

1 The QR algorithm

So far, we have seen ways to solve $Ax = b$, approximate best $||b - Ax||_2^2$, and (theoretically, but in practice not stably!) find spectral/SVD decompositions.

The induction-based proof of the existence of the Schur form we presented (which is also the basis for the SVD algorithm we presented) cannot be translated into a practical algorithm (although you may think that we know how to obtain the largest eigenvalue and an eigenvector for it through the power method, in practice the power method is not stable. Not to mention that if the largest eigenvalue is not simple or you have multiple "largest" eigenvalues which have the same absolute value, the power method will not work.)

We will now see (as our last algorithm this quarter) how these latter decompositions are found *in practice*.

The algorithm is iterative. Starting with a matrix $A \in \mathcal{M}_n$, the first step is to use unitary similarity transformations to reduce it to a *upper Hessenberg* matrix: $A = UHU^*$. The matrix H has all entries under the first *lower subdiagonal* 0; i.e., $H_{ij} = 0$, for all $i - j > 1$.

This can be achieved via successive multiplication by Householder reflectors (recall P4 from Homework 7). The reason why one cannot triangularize to find the Schur form is because, unlike with the QR decomposition, we need to find a unitary *similarity* transformation; thus, we act on the matrix both at the left and at the right, and we cannot ensure that the zeros created in the first column by the left multiplication will remain in place, unless we leave the first row unchanged. The picture below should hopefully clarify this. If we try to use the full first column to find the reflector we end up doing this

$$H_z \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} H_z^* = \begin{bmatrix} x & \circ & \circ & \circ \\ 0 & \circ & \circ & \circ \\ 0 & \circ & \circ & \circ \\ 0 & \circ & \circ & \circ \end{bmatrix} H_z^* = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix},$$

where we have changed symbols to indicate the fact that previous entries change, and x is real. Whereas if we attempt to zero out only the 3rd through nth entries of the first column, this translates as

$$\begin{bmatrix} 1 & 0 \\ 0 & H_z \end{bmatrix} \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & H_z^* \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ x & \circ & \circ & \circ \\ 0 & \circ & \circ & \circ \\ 0 & \circ & \circ & \circ \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & H_z^* \end{bmatrix} = \begin{bmatrix} * & \square & \square & \square \\ x & \square & \square & \square \\ 0 & \square & \square & \square \\ 0 & \square & \square & \square \end{bmatrix}.$$

Essentially, this proves the following claim:

Proposition 1. *Any $n \times n$ matrix has an upper Hessenberg form, which can be computed using $n - 1$ Householder reflectors (the last of which is scalar).*

Remark 1. *Note that, in finding the upper Hessenberg form of the matrix, the entries under the subdiagonal can be made to be always real.*

We present now the QR algorithm. The way the algorithm works is that, through unitary transformations, it iteratively decreases the first subdiagonal entries of H to numerical 0 (or some user-prescribed tolerance). Then it declares the diagonal entries of the obtained matrix to be the eigenvalues of A.

Without further ado, here a basic version of the algorithm. Generally, the matrix A is upper Hessenberg.

$A_0 = A$
$i = 0$
while $\max\{(A_i)_{j,j+1},\ j = 1, \ldots, n\} > tol$ **do**
  $[Q_i, R_i] = qr(A_i)$
  $A_{i+1} = R_i Q_i$
  $i = i + 1$
end while

This is only one (and by no means the best) way of stopping a QR iteration. In general, the algorithm proceeds until *one* of the entries is very small (so that the matrix is almost divided into two diagonal blocks. The union of the two eigenvalue sets of the two blocks is very close in some sense to the set of eigenvalues of the initial matrix–under certain assumptions. So the algorithm proceeds then by breaking the problem into two subproblems, each dealing with its respective block.

Proposition 2. *If A is upper Hessenberg, then all the matrices A_i are upper Hessenberg.*

Proof. This is not hard to see. Assume we do QR with a Gram-Schmidt variant. If A_i is upper Hessenberg, then Gram-Schmidt will produce an upper Hessenberg Q_i and a triangular R_i, which when multiplied in the opposite order, will once again yield an upper Hessenberg A_{i+1}. □

So at every step, the algorithm produces a new Hessenberg matrix, which eventually converges to a triangular one. We will not give the full proof of this fact, but will assign it as a reading (for example, under assumptions like $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, the algorithm converges). However, we will prove two simple facts that offer a bit of intuition toward this.

Lemma 1. *A_{k+1} is unitarily similar to A.*

Proof. Note that $A_{k+1} = R_k Q_k = Q_k^* Q_k R_k Q_k = Q_k^* A_k Q_k$. Thus, if we denote by $\tilde{Q}_k = Q_0 Q_1 \ldots Q_k$, $A_{k+1} = \tilde{Q}_k^* A \tilde{Q}_k$. □

Lemma 2. *Let $\tilde{R}_k = R_k \ldots R_1 R_0$. Then $\tilde{Q}_k \tilde{R}_k = A^{k+1}$.*

Proof. We prove this by induction; it is immediate for $k = 0$. Assume now it is true for $k - 1$, and let's prove it for k.

Note that

$$\tilde{R}_k = R_k \tilde{R}_{k-1} = Q_k^* A_{k-1} \tilde{R}_{k-1} = Q_k^* \tilde{Q}_{k-1} A \tilde{Q}_{k-1} \tilde{R}_{k-1} = \tilde{Q}_k A A^k = \tilde{Q}_k A^{k+1} \ .$$

□

This should suggest that the QR iteration converges under mild assumptions, for the same reason that the power method converges (on similar assumptions). One can actually do the same for an appropriately chosen set of eigenvalues (not just the first, and not all of them).

In practice, the QR iteration has many flavors and many additions that speed up convergence (adding shifts, splittig the matrix as explained above, etc.), but there will always be "problematic" matrices on which the QR algorithm will NOT converge.

As part of your homework, you will have to read the actual proof.