
Lecture Notes 11, Math/Comp 128, Math 250

Misha Kilmer
Tufts University

November 7, 2007

QR Algorithm, No Shifts

This “QR Algorithm” is **not** the one that produces the QR factorization of a matrix!
But it does use the QR factorization of a matrix.

Recall this is an algorithm we’re going to use in Phase II of the process! So the A here has already been reduced to upper Hessenberg (triangular) form.

Alg 28.1

$$A^*(0) = A$$

for $k = 1 : \text{maxits}$

- Find the QR factorization of $A^{(k-1)}$: $Q^{(k)}R^{(k)} = A^{(k-1)}$
- Define $A^{(k)}$ by mult. in reverse order: $A^{(k)} = R^{(k)}Q^{(k)}$

Convergence?

Claim: Under certain conditions, this converges to the Schur form (i.e. $A^{(k)}$ starts to look upper triangular, or diagonal if it was Hermitian to begin with.).

Prove (on board) that this is a similarity transform, so that $A^{(k)}$ and $A = A^{(0)}$ have the same eigenvalues.

Show example in Matlab.

The Practical Version for Symmetric Matrices

- As mentioned, A is already in tridiagonal form.
- Instead of computing the QR factorization of $A^{(k-1)}$, compute the QR factorization of a shifted version $A^{(k-1)} - \mu^{(k)}I$. Then put it back together as $A^{(k)} = R^{(k)}Q^{(k)} + \mu^{(k)}I$.
- Whenever possible (e.g. whenever an eigenvalue is found), **deflate** the problem by breaking $A^{(k)}$ into submatrices.

Practical QR Iteration

First, we need to show that by using the QR factorization of the shifted matrix, we still maintain a similarity transform (the eigenvalues of A are still the same as $A^{(k)}$).

What should the shift be? One choice is $\mu^{(k)} = A(m, m)^{(k-1)}$, where m is the size of the matrix. This choice is known as the **Raleigh Quotient shift**.

What is deflation?

This is a symmetric, tridiagonal matrix. So, if an element $A^{(k)}(j, j + 1) \approx 0$, we have $A^{(k)}(j + 1, j) \approx 0$, and so $A^{(k)}$ looks (almost) like a block-diagonal matrix

$$\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

But note that the eigenvalues of a block diagonal matrix are the eigenvalues of the blocks on the diagonal. (Proof on board)

Therefore, we should apply the practical QR iteration to blocks A_1 and A_2 **recursively**.

Practical QR

Assume $A^{(0)}$ is tridiagonal (output of Phase I process). For $k = 1, 2, \dots$ until convergence do

- Pick a shift $\mu^{(k)}$ (like $\mu^{(k)} = A_{mm}^{(k-1)}$)
- $Q^{(k)}R^{(k)} = A^{(k-1)} - \mu^{(k)}I$
- $A^{(k)} = R^{(k)}Q^{(k)} + \mu^{(k)}I$
- If any off-diagonal element $A^{(k)}(j, j + 1)$ is less than tolerance, set it to zero, to get

$$A^{(k)} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

and recurse on A_1 and A_2 .

One other detail...

$A^{(k)}$ (and therefore its shifted cousin) is tridiagonal. Is this preserved? And if so, what's the most efficient way to compute the QR factorizations in either variant of this algorithm...?

Convergence and History

This converges very quickly (cubically) to the eigenvalues, like the Raleigh-Quotient iteration, **depending on the shift**. In the worst case, it may not converge at all!!

This was one of the gold standards for computing eigenvalues from early 1960's to the early 1990's. In the 1990's, another competitor was introduced (divide and conquer).

Difficulties in Practice

Consider what happens to $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

- unshifted QR won't converge at all (why?)
- using the Rq-shift won't help either because it is zero.

The problem is the symmetry in the eigenvalues (± 1 in this case), both of the same magnitude. Need to “break” the symmetry.

Let the current $A^{(k)}$ have lower rightmost 2×2 submatrix look like

$$\begin{bmatrix} a_{m-1} & b_{m-1} \\ b_{m-1} & a_m \end{bmatrix}.$$

The **Wilkinson shift** is the eigenvalue of this little matrix that is closer to a_m (and when there's a tie, break it arbitrarily).

We can compute this value exactly (see (29.8) in text).

Can show that the QR iteration with this shift is **at least quadratically convergent in the worst case**, but cubic in the generic case.

Stability

Since we keep using orthogonal matrices, the QR iteration is backward stable. Note that the eigenvectors (of the tridiagonal matrix) are given by the columns of the *product* of the Q matrices. (Think about how much work to form this product, given the fact we'd use Givens rotations to do the QR factorization of the tridiagonal matrix.)

Theorem 29.1 Let a real, symmetric, tridiagonal matrix $A \in \mathbb{R}^{m \times m}$ be diagonalized by the QR algorithm (Alg. 28.2) on a computer satisfying (13.5) and (13.7). Then

$$\tilde{Q}\tilde{\lambda}\tilde{Q}^* = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = O(\epsilon_{mach}).$$

Accuracy

It can also be shown that

$$\frac{|\tilde{\lambda}_j - \lambda_j|}{\|A\|} = O(\epsilon_{mach}),$$

meaning that, relative to the size of A , the eigenvalues are computed quite accurately.

Outline for Remainder

- Bisection and Divide and Conquer
- Application(s)
- Computing the SVD by considering *the correct* eigenvalue problem
- Iterative methods for solutions to linear systems (that also sometimes give approximate eigen-information)
- Application(s)

Bisection

- Real, symmetric tridiagonal matrices
- Very fast at finding eigenvalues in arbitrarily small subintervals
- If only a small number of eigenvalues are needed, it takes $O(m \log(\epsilon_{mach}))$ flops, which is better than the $O(m^2)$ for the QR iteration.
- Algorithm based around an eigenvalue interlacing property for symm. tridiagonal matrices (see next board, picture in book pg. 228).
- Three key observations give rise to this algorithm, as we shall see.

Interlacing and Sturm Sequences

Interlacing means that for the $k \times k$ leading principal submatrix of the symm. tridiagonal matrix A , we have

$$\lambda_j^{(k+1)} < \lambda_j^{(k)} < \lambda_{j+1}^{(k+1)}, \quad k = 1, 2, \dots, m-1; j = 1, 2, \dots, k-1$$

Key Observation 1: Interlacing property makes it possible to count the exact number of eigenvalues of a matrix in a specified interval!

Recall that $\det(A) = \prod_{i=1}^m \lambda_i$. From this fact, we can at least establish if there is a negative eigenvalue (provided A is invertible) among the bunch.

One of the few times computing a determinant is both feasible and useful...!

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

Determinants of the submatrices are easy to compute, because it's a tridiagonal matrix...

$$\det(A^{(1)}) = 1, \det(A^{(2)}) = -1, \det(A^{(3)}) = -3, \det(A^{(4)}) = 4$$

Thus, we know $A^{(1)}$ has no negative e-vals, $A^{(2)}$ has 1, $A^{(3)}$ has only 1 (because of the interlacing property), $A^{(4)}$ has 2 (again, because of interlacing).

In general, for a symm. tridiagonal A , the number of negative eigenvalues is equal to the number of sign changes in the Sturm sequence

$$1, \det(A^{(1)}), \det(A^{(2)}), \det(A^{(3)}), \dots, \det(A^{(m)})$$

Note: transition from + or 0 to - counts, from - or 0 to + counts, but not from + or - to 0.

Key Observation 2: Since the eigenvalues of $A - xI$ are $\lambda_i - x$, by shifting A , we can determine the number of eigenvalues in any interval $[a, b)$: it's the number of eigenvalues in $[-\infty, b)$ minus the number in $(-\infty, a)$.

Bisection, cont.

For a dense matrix A , computing $\det(A)$ is expensive and usually unstable.

However, this is a symmetric tridiagonal matrix, so by expansion by minors we can show

$$\det(A^{(k)}) = a_k \det(A^{(k-1)}) - b_{k-1}^2 \det(A^{(k-2)}).$$

Introducing a shift x , and writing $p^{(k)}(x) = \det(A^{(k)} - xI)$, we get

$$p^{(k)}(x) = (a_k - x)p^{(k-1)}(x) - b_{k-1}^2 p^{(k-2)}(x), p^{(-1)}(x) := 0, p^{(0)}(x) := 1, k = 1, \dots, m.$$

See demo on example.

Divide and Conquer

- Most description on board.
- Works on symmetric, tridiagonal matrix.
- Implementation details not fully discussed – but the devil (and the stability) is in the details!
- Introduced by Cuppen 1981, more than 2x as fast as QR iteration if you want the eigenvectors as well as the eigenvalues. But stable version of the algorithm came in the 1990's.

Key Idea: Rewrite the tridiagonal matrix as a 2×2 block matrix + a rank-1 correction, where the 2×2 blocks are each roughly $m/2$.

Divide and Conquer

Write $T = \hat{T} + \text{rank-one}$ (see board, pg. 230)

Divide and Conquer

If we know

$$\hat{T}_1 = Q_1 D_1 Q_1^T, \quad \hat{T}_2 = Q_2 D_2 Q_2^T$$

then

$$T = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \left(\begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} + \beta z z^T \right) \begin{bmatrix} Q_1^T & \\ & Q_2^T \end{bmatrix}$$

where $z = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$, and q_i is the last row of Q_i , $i=1,2$.

So now we need to find the eigenvalues of a diagonal matrix + a rank-one correction...

Secular Equation

The eigenvalues of $D + ww^T$ are the roots of the **secular equation**:

$$f(\lambda) = 1 + \sum_{j=1}^m \frac{w_j^2}{d_j - \lambda}$$

Since this is not linear in λ , we need a root-finding procedure (i.e. have to solve $f(\lambda) = 0$).

Use a variant of Newton's method. Only $O(1)$ iterations of Newton's method required per root, so we need $O(m)$ flops per root of an $m \times m$ matrix, so $O(m^2)$ altogether.

Since this is a divide and conquer method, we need to account for solving via recursion...

CS folks??

$$O(m^2 + 2(m/2)^2 + 4(m/4)^2 + 8(m/8)^2 + \dots m(m/m)^2)$$

converges to $O(m^2)$.