11 The QR Algorithm

11.1 QR Algorithm without Shifts

In the previous chapter (in the Maple worksheet `473_Hessenberg.mws`) we investigated two different attempts to tackling the eigenvalue problem. In the first attempt (which we discarded) the matrix A was multiplied from the left and right by a unitary Householder matrix Q.

We now consider the following

Algorithm ("Pure" QR)

Initialize $A^{(0)} = A$

for $k = 1, 2, \ldots$

Compute the QR factors $Q^{(k)}R^{(k)} = A^{(k-1)}$

Reassemble the factors $A^{(k)} = R^{(k)}Q^{(k)}$

end

Note that $R^{(k)} = \left[Q^{(k)}\right]^T A^{(k-1)}$ and so

$$A^{(k)} = \left[Q^{(k)}\right]^T A^{(k-1)} Q^{(k)},$$

the similarity transform from "approach 1" mentioned above.

We recall that this algorithm will not produce a triangular (or, in our special real symmetric case, a diagonal) matrix in one step. However, it will succeed in an iterative setting. In fact, we will see below that the "pure" QR algorithm is equivalent to the orthogonal simultaneous iteration algorithm presented earlier.

Remark In the next section we will discuss a "practical" QR algorithm that will use shifts and converge cubically like the Rayleigh quotient iteration.

We now show the equivalence of the "pure" QR algorithm and orthogonal simultaneous iteration. Recall the orthogonal simultaneous iteration algorithm:

Initialize $\hat{Q}^{(0)}$ with an arbitrary $m \times n$ matrix with orthonormal columns

for $k = 1, 2, \ldots$

$Z = A\hat{Q}^{(k-1)}$

Compute the QR factorization $\hat{Q}^{(k)}\hat{R}^{(k)} = Z$

end

Now consider the special case of square (real symmetric) matrices, which is what we will be working with in eigenvalue problems. We start the orthogonal simultaneous iteration with

$$\underline{Q}^{(0)} = I \in \mathbb{R}^{m \times m}, \tag{S1}$$

where we no longer need the hat (since the matrices are square), i.e., we will be performing full QR factorizations. The underline is used to distinguish between the Q-matrices appearing in this algorithm and those in the "pure" QR below.

Next, we compute

$$Z = A\underline{Q}^{(k-1)} \tag{S2}$$

$$\underline{Q}^{(k)} R^{(k)} = Z \tag{S3}$$

so that

$$A^{(k)} = \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}. \tag{S4}$$

For the "pure" QR algorithm, on the other hand, we have

$$A^{(0)} = A \tag{Q1}$$

$$Q^{(k)} R^{(k)} = A^{(k-1)} \tag{Q2}$$

$$A^{(k)} = R^{(k)} Q^{(k)} \tag{Q3}$$

For this algorithm we also define

$$\underline{Q}^{(k)} = Q^{(1)} Q^{(2)} \dots Q^{(k)}, \tag{Q4}$$

and for both algorithms

$$\underline{R}^{(k)} = R^{(k)} R^{(k-1)} \dots R^{(1)}. \tag{R5}$$

Here we make the convention that empty products yield the identity.

Theorem 11.1 *Orthogonal simultaneous iteration* (S1)–(S4) *and the "pure" QR algorithm* (Q1)–(Q4) *are equivalent, i.e., both algorithms produce identical sequences* $\underline{R}^{(k)}$, $\underline{Q}^{(k)}$ *and* $A^{(k)}$. *In fact, the* k*-th power of* A *is given by*

$$A^k = \underline{Q}^{(k)} \underline{R}^{(k)}, \tag{28}$$

and the k*-th iterate by*

$$A^{(k)} = \left[\underline{Q}^{(k)}\right]^T A\underline{Q}^{(k)}. \tag{29}$$

Proof The proof uses induction on k.

For $k = 0$ simultaneous iteration begins with $\underline{Q}^{(0)} = I$ so that (28) (here $A^0 = I = \underline{Q}^{(0)}$) and (29) (here $A^{(0)} = A$) are trivially satisfied with $\underline{R}^{(0)} = I$.

For "pure" QR the induction start is also true since we set $A^{(0)} = A$, and then $\underline{Q}^{(0)} = \underline{R}^{(0)} = I$.

Now let's look at simultaneous iteration for $k \geq 1$. Obviously, (29) always holds due to (S4). We need to show that (28) also holds.

By the induction hypothesis (for (28)) we have

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

so that

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

Using (S2), (S3), and (R5) we get

$$\begin{aligned} A^k &\overset{(S2)}{=} Z\underline{R}^{(k-1)} \\ &\overset{(S3)}{=} \underline{Q}^{(k)}R^{(k)}\underline{R}^{(k-1)} \\ &\overset{(R5)}{=} \underline{Q}^{(k)}\underline{R}^{(k)}, \end{aligned}$$

which establishes this claim.

Finally, we need to establish the induction step for "pure" QR. First we show that (28) is true. As above we have

$$A^k = A\underline{Q}^{(k-1)}\underline{R}^{(k-1)}. \tag{30}$$

Now the induction hypothesis (for (29)) can be rephrased as follows:

$$A^{(k-1)} = \left[\underline{Q}^{(k-1)}\right]^T A\underline{Q}^{(k-1)} \quad \Longleftrightarrow \quad A = \underline{Q}^{(k-1)}A^{(k-1)}\left[\underline{Q}^{(k-1)}\right]^T. \tag{31}$$

Combining (30) and (31) together with (Q2), (Q4) and (R5) lets us establish this part:

$$\begin{aligned} A^k &\overset{(30)}{=} A\underline{Q}^{(k-1)}\underline{R}^{(k-1)} \\ &\overset{(31)}{=} \underline{Q}^{(k-1)}A^{(k-1)}\underbrace{\left[\underline{Q}^{(k-1)}\right]^T\underline{Q}^{(k-1)}}_{=I}\underline{R}^{(k-1)} \\ &= \underline{Q}^{(k-1)}A^{(k-1)}\underline{R}^{(k-1)} \\ &\overset{(Q2)}{=} \underline{Q}^{(k-1)}Q^{(k)}R^{(k)}\underline{R}^{(k-1)} \\ &\overset{(Q4),(R5)}{=} \underline{Q}^{(k)}\underline{R}^{(k)}. \end{aligned}$$

The last part of the proof consists of showing that (29) holds for the "pure" QR algorithm. First,

$$\begin{aligned} A^{(k)} &\overset{(Q3)}{=} R^{(k)}Q^{(k)} \\ &\overset{(Q2)}{=} \left[Q^{(k)}\right]^T A^{(k-1)}Q^{(k)}. \end{aligned}$$

To finish we use the induction hypothesis for (29), i.e., $A^{(k-1)} = \left[\underline{Q}^{(k-1)}\right]^T A\underline{Q}^{(k-1)}$. Then

$$A^{(k)} = \underbrace{\left[Q^{(k)}\right]^T\left[\underline{Q}^{(k-1)}\right]^T}_{\overset{(Q4)}{=}\left[\underline{Q}^{(k)}\right]^T} A\underbrace{\underline{Q}^{(k-1)}Q^{(k)}}_{\overset{(Q4)}{=}\underline{Q}^{(k)}}$$

and the proof is complete. ■

Let us now think about why "pure" QR iteration works. First, for power iteration we showed that $A^k v^{(0)}$ converges to the eigenvector corresponding to $\lambda_{\max}$. Now, (28)

says that $A^k = \underline{Q}^{(k)}\underline{R}^{(k)}$, and therefore we have an orthonormal basis for A^k given by the columns of $\underline{Q}^{(k)}$, and the eigenvectors (to all eigenvalues) are found.

To understand this, note that we know from one of the homework problems that Rayleigh quotients are the diagonal entries of Q^TAQ. In fact, (29) implies that the Rayleigh quotients of A corresponding to $\underline{Q}^{(k)}$ are on the diagonal of $A^{(k)}$. Since the columns of $\underline{Q}^{(k)}$ converge to the eigenvectors of A, the diagonal entries of $A^{(k)}$ converge to the eigenvalues of A. More precisely we have

Theorem 11.2 *For real symmetric matrices with distinct eigenvalues and for which the leading principal minors of the eigenvector matrix Q are nonsingular we have*

$$|a_{jj}^{(k)} - \lambda_j| = \mathcal{O}(C^k), \qquad j = 1, \ldots, m,$$

where $C = \max_{1\le\ell\le m-1} \frac{|\lambda_{\ell+1}|}{|\lambda_\ell|}$. Moreover,

$$\|\boldsymbol{q}_j^{(k)} - (\pm\boldsymbol{q}_j)\| = \mathcal{O}(C^k), \qquad j = 1, \ldots, m.$$

Thus, convergence of the "pure" (unshifted) QR algorithm is linear for both the eigenvalues and eigenvectors. We now look at the "practical" QR algorithm that will yield cubic convergence.

11.2 Practical QR Algorithm (with shifts)

We start with noting

Theorem 11.3 *Orthogonal simultaneous* inverse *iteration (applied to a permuted matrix) and the "pure" QR algorithm are equivalent.*

Proof In the previous section we saw that

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

with

$$\underline{Q}^{(k)} = Q^{(1)}Q^{(2)}\ldots Q^{(k)},$$

where the $Q^{(j)}$ are same orthogonal matrices as used by the "pure" QR algorithm. Now

$$A^{-k} = \left[\underline{Q}^{(k)}\underline{R}^{(k)}\right]^{-1} = \left[\underline{R}^{(k)}\right]^{-1}\left[\underline{Q}^{(k)}\right]^T.$$

Since A is symmetric, so is A^{-k}, and therefore

$$A^{-k} = \left[A^{-k}\right]^T = \underline{Q}^{(k)}\left[\underline{R}^{(k)}\right]^{-T}. \tag{32}$$

We now let

$$P = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$$

be the permutation that reverses the order of columns (if multiplied from the right), and note that $P^2 = I$.

Therefore (32) can be rewritten as

$$\begin{aligned} A^{-k} &= \underline{Q}^{(k)} \left[\underline{R}^{(k)}\right]^{-T} \\ \iff \quad A^{-k}P &= \underline{Q}^{(k)} P P \left[\underline{R}^{(k)}\right]^{-T} P \\ &= \underbrace{\underline{Q}^{(k)} P}_{\text{orthogonal}} \underbrace{P \left[\underline{R}^{(k)}\right]^{-T} P}_{\text{upper triangular}} \end{aligned} \tag{33}$$

Since (33) consists of the product of an orthogonal with an upper triangular matrix it is the QR factorization of $A^{-k}P$. It also provides a formula for the k-th step of orthogonal simultaneous iteration for the permuted A^{-1}, i.e., orthogonal simultaneous inverse iteration for the permuted A. ■

Now, in the previous chapter we added two features to inverse iteration to obtain cubic convergence:

1. We added a shift.
2. We combined our algorithm with the Rayleigh quotient estimate for eigenvalues.

11.3 Adding a Shift

The main steps of the "pure" QR algorithm were

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

With a shift this becomes

$$Q^{(k)} R^{(k)} = A^{(k-1)} - \mu^{(k)} I \tag{34}$$

$$A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I. \tag{35}$$

From (34) we get

$$R^{(k)} = \left[Q^{(k)}\right]^T \left(A^{(k-1)} - \mu^{(k)} I\right),$$

which we insert into (35):

$$\begin{aligned} A^{(k)} &= \left[Q^{(k)}\right]^T \left(A^{(k-1)} - \mu^{(k)} I\right) Q^{(k)} + \mu^{(k)} I \\ &= \left[Q^{(k)}\right]^T A^{(k-1)} Q^{(k)} - \underbrace{\left[Q^{(k)}\right]^T \mu^{(k)} I Q^{(k)}}_{=\mu^{(k)} I} + \mu^{(k)} I \\ &= \left[Q^{(k)}\right]^T A^{(k-1)} Q^{(k)}. \end{aligned}$$

Recursive application of this identity produces

$$A^{(k)} = \left[\underline{Q}^{(k)}\right]^T A \underline{Q}^{(k)},$$

which is the similarity transform condition (29) from Theorem 11.1.

The other condition (analogous to (28)) now becomes

$$(A - \mu I)^k = \underline{Q}^{(k)} \underline{R}^{(k)}$$

provided we use just one uniform shift μ, or — if we use different shifts —

$$\prod_{j=k}^{1} \left(A - \mu^{(j)} I\right) = \underline{Q}^{(k)} \underline{R}^{(k)}. \tag{36}$$

We now connect this approach to the Rayleigh quotient. To get cubic convergence we need to choose the shifts $\mu^{(k)}$ as Rayleigh quotients, i.e.,

$$\mu^{(k)} = \frac{\left(\boldsymbol{q}_m^{(k)}\right)^T A \boldsymbol{q}_m^{(k)}}{\left(\boldsymbol{q}_m^{(k)}\right)^T \boldsymbol{q}_m^{(k)}} = \left(\boldsymbol{q}_m^{(k)}\right)^T A \boldsymbol{q}_m^{(k)},$$

where $\boldsymbol{q}_m^{(k)}$ is the m-th column of the orthogonal matrix $\underline{Q}^{(k)}$. Note that using (29) this simplifies further:

$$\begin{aligned} \left(\boldsymbol{q}_m^{(k)}\right)^T A \boldsymbol{q}_m^{(k)} &= \boldsymbol{e}_m^T \left[\underline{Q}^{(k)}\right]^T A \underline{Q}^{(k)} \boldsymbol{e}_m \\ &= \boldsymbol{e}_m^T A^{(k)} \boldsymbol{e}_m = A_{mm}^{(k)}. \end{aligned}$$

Therefore, we choose the shifts as

$$\mu^{(k)} = A_{mm}^{(k)}.$$

11.4 Deflation

In order to efficiently isolate the eigenvalues one frequently uses a procedure known as *deflation*. Remember that we are going to use a 2-phase algorithm such that (for real symmetric matrices) we

1. Convert A to tridiagonal form.
2. Produce a sequence of tridiagonal matrices that converges to a diagonal T.

Consider the tridiagonal matrix $A^{(k)}$ (obtained in phase 1 of our eigenvalue algorithm and maintained during phase 2).

If some sub-diagonal entry $A_{j,j+1}^{(k)}$ is less than a specified tolerance, then we set it and its corresponding symmetric counterpart to zero, i.e., $A_{j,j+1}^{(k)} = A_{j+1,j}^{(k)} = 0$. This

allows us to break $A^{(k)}$ into two tridiagonal blocks. Schematically,

$$A^{(k)} = \left[\begin{array}{ccc|ccc} x & x & 0 & | & 0 & 0 \\ x & x & x & | & 0 & 0 \\ 0 & x & x & | & \mathbf{0} & 0 \\ - & - & - & + & - & - \\ 0 & 0 & \mathbf{0} & | & x & x \\ 0 & 0 & 0 & | & x & x \end{array}\right],$$

where the **0** entries are those set to zero in the deflation procedure.

One then applies the algorithm recursively on the two tridiagonal blocks. We will implement this idea in one of the computer assignments.

Here is a high-level algorithm that will find all eigenvalues of a real symmetric matrix A, and (usually) do so with cubic convergence.

Algorithm ("Practical" QR)

$A^{(0)}$ = tridiagonalization of A (obtained using Hessenberg reduction)

for $k = 1, 2, \ldots$

$\mu^{(k)} = A^{(k-1)}_{mm}$
$Q^{(k)}R^{(k)} = A^{(k-1)} - \mu^{(k)}I$
$A^{(k)} = R^{(k)}Q^{(k)} + \mu^{(k)}I$
If deflation is possible
apply the algorithm recursively
end

end

11.5 Problems with the Rayleigh Quotient Shift

There are certain special configurations for which QR iteration with the Rayleigh quotient shift *does not* converge.

Example Consider the matrix

$$A = \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right].$$

In this case $A_{mm} = 0$, and so the shifted QR algorithm is the same as the unshifted algorithm. Thus,

$$A^{(0)} = Q^{(1)}R^{(1)} = \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right]\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right],$$

and

$$A^{(1)} = R^{(1)}Q^{(1)} = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right]\left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right] = A^{(0)}.$$

We see that nothing happens — the algorithm stalls.

The reason for this failure is that the eigenvalues of A, $\lambda_{1,2} = \pm 1$, are located symmetric about the origin, and the estimate $\mu = 0$ cannot decide which way to go.

A remedy is provided by a different shift which breaks the symmetry. One such example is the *Wilkinson shift*. It is discussed, e.g., in the book [Trefethen/Bau].

In summary, a "practical" QR algorithm with Wilkinson shift (instead of the Rayleigh quotient shift used above) for real symmetric matrices

- is backward stable,
- converges cubically for most initial guesses (otherwise it converges quadratically),
- has a computational cost of $\mathcal{O}(\frac{4}{3}m^3)$ flops (with most of the computing time spent in the tridiagonalization phase 1; phase 2 requires only $\mathcal{O}(m^2)$ operations).

11.6 Other Eigenvalue Algorithms

Fairly recently, two competitors for the QR iteration algorithm have emerged:

1. In the 1980s so-called *divide-and-conquer* algorithms were proposed. They use ideas similar to deflation and are about twice as fast as the QR algorithm if *both* eigenvalues and eigenvectors are to be computed.
2. *Relatively robust representation* methods from the 1990s can find the eigenvalues of a real symmetric matrix in $\mathcal{O}(m^2)$ operations.

Both approaches are implemented in the LAPACK software library.