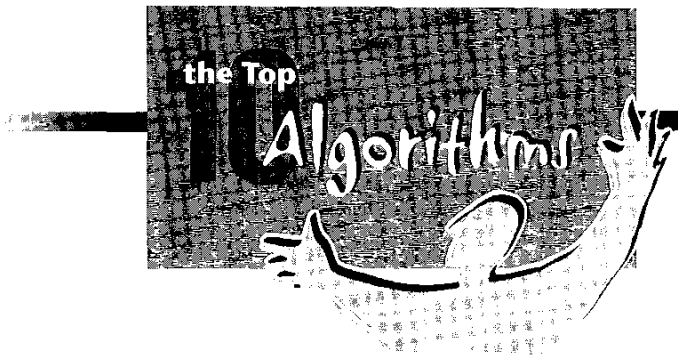

THE QR ALGORITHM

After a brief sketch of the early days of eigenvalue hunting, the author describes the QR Algorithm and its major virtues. The symmetric case brings with it guaranteed convergence and an elegant implementation. An account of the impressive discovery of the algorithm brings the article to a close.

I assume you share the view that the rapid computation of a square matrix's eigenvalues is a valuable tool for engineers and scientists.[1] The QR Algorithm solves the eigenvalue problem in a very satisfactory way, but this success does not mean the QR Algorithm is necessarily the last word on the subject. Machines change and problems specialize. What makes the experts in matrix computations happy is that this algorithm is a genuinely new contribution to the field of numerical analysis and not just a refinement of ideas given by Newton, Gauss, Hadamard, or Schur.

Early history of eigenvalue computations

Matrix theory dramatically increased in importance with the arrival of matrix mechanics and quantum theory in the 1920s and 1930s. In the late 1940s, some people asked themselves how the digital computer might be employed to solve the matrix eigenvalue problem. The obvious approach was to use a two-stage method. First, compute the coefficients of the characteristic polynomial, and then compute the zeros of the characteristic polynomial.

There are several ingenious ways to accomplish the first stage in a number of arithmetic operations proportional to n^3 or n^4, where n is the order of the matrix.[2] The second stage was a hot topic of research during this same time. Except for very small values of n, $n \leq 10$, this two-stage approach is a disaster on a computer with fixed word length. The reason is that the zeros of a polynomial are, in general, incredibly sensitive to tiny changes in the coefficients, whereas a matrix's eigenvalues are often, but not always, insensitive to small uncertainties in the n^2 entries of the matrix. In other words, the replacement of those n^2 entries by the characteristic polynomial's n coefficients is too great a condensation of the data.

A radical alternative to the characteristic polynomial is the use of similarity transformations to obtain a nicer matrix with the same eigenvalues. The more entries that are zero, the nicer the matrix. Diagonal matrices are perfect, but a triangular matrix is good enough for our purposes be-

1521-9615/00/$10.00 © 2000 IEEE

BERESFORD N. PARLETT
University of California, Berkeley

cause the eigenvalues lie on the main diagonal. For deep theoretical reasons, a triangular matrix is generally not attainable in a finite number of arithmetic operations. Fortunately, one can get close to triangular form with only $O(n^3)$ arithmetic operations. More precisely, a matrix is said to be *upper Hessenberg* if it is upper triangular with an extra set of nonzero entries just below the diagonal in positions $(i+1, i)$, $i = 1, 2, \ldots, n-1$, and these are called the *subdiagonal entries*. What is more, the similarity transformations needed to obtain a Hessenberg form (it is not unique) can be chosen so that no computed matrix entry ever exceeds the norm of the original matrix. Any proper norm, such as the square root of the sum of the squares of the entries, will do.

This property of keeping intermediate quantities from becoming much bigger than the original data is important for computation with fixed-length numbers and is called *stability*. The hard task is to find similarity transformations that both preserve Hessenberg form and eliminate those subdiagonal entries. This is where the QR Algorithm comes to the rescue. The subroutine DHSEQR in the Lapack library embodies the latest implementation.[3]

Let us try to put the improvement based on QR in perspective. It has reduced the time for standard eigenvalue computations to the time required for a few matrix multiplies.

The LU and QR Algorithms

Suppose $B = XY$, with X invertible. Then the new matrix $C := YX$ is similar to B, because $C = YX = X^{-1}BX$. Although intriguing, this observation does not appear to be of much use to eigenvalue hunters. Let us recall two well-known ways of factoring a matrix:

- Triangular factorization (or Gaussian elimination), $B = LU$, where L is lower triangular with 1's along the main diagonal and U is upper triangular. This factorization is not always possible. The multipliers in the reduction are stored in L, the reduced matrix in U.
- QR (or orthogonal triangular) factorization, $B = QR$, where Q is unitary, $Q^{-1} = Q^*$ (conjugate transpose of Q), and R is upper triangular with nonnegative diagonal entries. This factorization always exists. Indeed, the columns of Q are the outputs of the Gram-Schmidt orthonormalizing process when it is executed in exact arithmetic on the columns of $B = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$.

Each factorization leads to an algorithm by iteration.

The LU transform of B is $UL = L^{-1}BL$, and the QR transform of B is $RQ = Q^{-1}BQ = Q^*BQ$. In general, the LU or QR transform of B will not have more zero entries than B. The rewards of using this transform come only by repetition.

Theorem 1 (Fundamental Theorem). If B's eigenvalues have distinct absolute values and the QR transform is iterated indefinitely starting from $B_1 = B$,

$$\begin{array}{lcll} \text{Factor} & B_j & = & Q_jR_j, \\ \text{Form} & B_{j+1} & = & R_jQ_j, \qquad j = 1, 2, \ldots. \end{array}$$

then, under mild conditions on the eigenvector matrix of B, the sequence $\{B_j\}$ converges to the upper triangular form (commonly called the Schur form) of B with eigenvalues in monotone decreasing order of absolute value down the diagonal.

This result is not obvious; see James Hardy Wilkinson's article for proofs.[4] The procedure that generates $\{B_j\}$ is called the *basic* QR Algorithm.

A similar theorem holds for the basic LU algorithm, provided all the transforms exist. It takes several clever observations to turn this simple theory into the highly successful QR Algorithm of today.

Invariance of the Hessenberg form

If B is an upper Hessenberg matrix (entry (i, j) vanishes if $i > j + 1$), then so are all its QR iterates and LU iterates. This useful result is easy to see because it depends only on the pattern of zeros in the matrices and not on the values of the nonzero entries. If B_j is Hessenberg, then so is Q_j, because R_j^{-1} is triangular. Consequently, $R_jQ_j = (B_{j+1})$ is also Hessenberg. The cost of computing the QR factorization falls from $O(n^3)$ to $O(n^2)$ when the matrix is Hessenberg and $n \times n$.[5] A similar result holds for LU iterates.

Fortunately, any matrix can be reduced by similarities to Hessenberg form in $O(n^3)$ arithmetic operations in a stable way, and from this point on we will assume this reduction has been performed as an initial phase. The mild conditions mentioned in the Fundamental Theorem are satisfied by upper Hessenberg matrices with nonzero subdiagonals.[6]

Accelerating convergence

The Hessenberg sequences $\{B_j\}$ and $\{C_j\}$ produced by the basic algorithms converge linearly, and that is too slow for impatient customers. We can improve the situation by making a subtle

change in our goal. Instead of looking at the matrix sequence $\{B_j\}$, we can focus on the $(n, n-1)$ entry of each matrix, the last subdiagonal entry. When the $(n, n-1)$ entry is negligible, the (n, n) entry is an eigenvalue—to within working precision—and column n does not influence the remaining eigenvalues. Consequently, the variable n can be reduced by 1, and computation continues on the smaller matrix. We say that the nth eigenvalue has been *deflated*. The top of the matrix need not be close to triangular form. Thus, convergence refers to the scalar sequence of $(n, n-1)$ entries. The rate of convergence of this sequence can be vastly improved, from linear to quadratic, by using the shifted QR Algorithm: Let $B_1 = B$. For $i = 1, 2, \ldots$ until convergence,

Select a shift s_i
Factor $B_i - s_i I = Q_i R_i$
Form $B_{i+1} = R_i Q_i + s_i I = Q_i^* B_i Q_i$.

In principle, each shift strategy has a different convergence theory. It is rare that more than $2n$ QR iterations are needed to compute all the eigenvalues of B.

The double shift implementation for real matrices

There is a clever variation on the shifted QR Algorithm that I should mention. In many applications, the initial matrix is real, but some of the eigenvalues are complex. The shifted algorithm must then be implemented in complex arithmetic to achieve quadratic convergence. The man who first presented the QR Algorithm, J.G.F Francis,[7] showed how to keep all arithmetic in the real field and still retain quadratic convergence.

Let us see how it is done. Without loss, take $j = 1$. Consider two successive steps:

$$\begin{aligned} B_1 - s_1 I &= Q_1 R_1 \\ B_2 &= R_1 Q_1 + s_1 I \\ B_2 - s_2 I &= Q_2 R_2 \\ B_3 &= R_2 Q_2 + s_2 I. \end{aligned}$$

It turns out, after some manipulation, that

$$(Q_1 Q_2)(R_2 R_1) = (B_1 - s_1 I)(B_1 - s_2 I)$$

and

$$B_3 = (Q_1 Q_2)^* B_1 (Q_1 Q_2).$$

Suppose s_1 and s_2 are either both real or a complex conjugate pair. Then $(B_1 - s_1 I)(B_1 - s_2 I)$ is real. By the uniqueness of the QR factorization, $Q_1 Q_2$ is the Q factor of $(B_1 - s_1 I)(B_1 - s_2 I)$ and so is real orthogonal, not just unitary. Hence, B_3 is a product of three real matrices and thus real.

The next challenge is to compute B_3 from B_1 and s (complex) without constructing B_2. The solution is far from obvious and brings us to the concept of *bulge chasing*, a significant component of the QR success story.

Bulge chasing

The theoretical justification comes from the Implicit Q or Uniqueness of Reduction property.

Theorem 2 (Uniqueness). If Q is orthogonal, B is real, and $H = Q^* B Q$ is a Hessenberg matrix in which each subdiagonal entry $h_{i+1,i} > 0$, then H and Q are determined uniquely by B and q_1, the first column of Q.

Now return to the equations above and suppose $s_1 = s$, $s_2 = \bar{s} \neq s$. If B_1 is Hessenberg, then so are all the B_i's. Suppose that B_3 has positive subdiagonal entries. By Theorem 2, both B_3 and $Q_1 Q_2$ are determined by column 1 of $Q_1 Q_2$, which we'll call $\boldsymbol{q}$. Because $R_2 R_1$ is upper triangular, $\boldsymbol{q}$ is a multiple of the first column of

$$B_1^2 - 2(\mathrm{Re}\, s) B_1 + |s|^2 I.$$

Because B_1 is Hessenberg, the vector $\boldsymbol{q}$ is zero except in its top three entries.

The following three-stage algorithm computes B_3. It uses orthogonal matrices $H_j, j = 1, 2, \ldots, n-1$, such that H_j is the identity except for a 3×3 submatrix in rows and columns $j, j+1, j+2$. H_{n-1} differs from I only in its trailing 2×2 matrix.

1. Compute the first three entries of $\boldsymbol{q}$.
2. Compute a matrix H_1 such that $H_1^t \boldsymbol{q}$ is a multiple of $\boldsymbol{e}_1$, the first column of I. Form $C_1 = H_1^t B_1 H_1$. It turns out that C_1 is upper Hessenberg except for nonzeros in positions (3, 1), (4, 1), and (4, 2). This little submatrix is called the *bulge*.
3. Compute a sequence of matrices $H_2, \ldots, H_{n-1}$ and $C_j = H_j^t C_{j-1} H_j, j = 2, \ldots, n-1$ such that $C_{n-1} = H_{n-1}^t \ldots H_2^t C_1 H_2 \ldots H_{n-1}$ is a Hessenberg matrix with positive subdiagonal entries. More on H_j below.

We claim that $C_{n-1} = B_3$. Recall that column 1 of H_j is $\mathbf{e}_1$ for $j > 1$. Thus, $H_1 H_2 \ldots H_{n-1} \mathbf{e}_1 = H_1 \mathbf{e}_1 = \mathbf{q}/\|\mathbf{q}\| = (Q_1 Q_2)\mathbf{e}_1$. Now $C_{n-1} = (H_1 \ldots H_{n-1})^t B_1 (H_1 \ldots H_{n-1})$, and $B_3 = (Q_1 Q_2)^t B_1 (Q_1 Q_2)$. The Implicit Q Theorem ensures that B_3 and C_{n-1} are the same. Moreover, if s is not an eigen-

value, then C_{n-1} must have positive subdiagonal entries in exact arithmetic.

Step 3 involves $n-2$ minor steps, and at each one only three rows and columns of the array are altered. The code is elegant, and the cost of forming C_{n-1} is about $5n^2$ operations. The transformation $C_2 \rightarrow H_2{}^t C_1 H_2$ pushes the bulge into positions (4, 2), (5, 2), and (5, 3), while creating zeros in positions (3, 1) and (4, 1). Subsequently, each operation with an H matrix pushes the bulge one row lower until it falls off the bottom of the matrix and the Hessenberg form is restored. The transformation $B_1 \rightarrow B_3$ is called a double step.

It is now necessary to inspect entry $(n-1, n-2)$ as well as $(n, n-1)$ to see whether a deflation is warranted. For complex conjugate pairs, the $(n-1, n-2)$ entry, not $(n, n-1)$, becomes negligible.

The current shift strategies for QR do not guarantee convergence in all cases. Indeed, examples are known where the sequence $\{B_j\}$ can cycle. To guard against such misfortunes, an ad hoc exceptional shift is forced from time to time. A more complicated choice of shifts might produce a nicer theory, but practical performance is excellent.[8]

The symmetric case

The QR transform preserves symmetry for real matrices and preserves the Hermitian property for complex matrices: $B \rightarrow Q^* B Q$. It also preserves Hessenberg form. Because a symmetric Hessenberg matrix is tridiagonal (that means entry (i, j) vanishes if $|i-j| > 1$), the QR Algorithm preserves symmetric tridiagonal form and the cost of a transform plunges from $O(n^2)$ to $O(n)$ operations. In fact, the standard estimate for the cost of computing all the eigenvalues of an $n \times n$ symmetric tridiagonal matrix is $10n^2$ arithmetic operations. Recall that all the eigenvalues are real.

One reason this case is worthy of a section to itself is its convergence theory. Theorem 1 tells us that the basic algorithm (all shifts are zero) pushes the large eigenvalues to the top and the small ones to the bottom. A shift strategy Wilkinson suggested in the 1960s *always* makes the $(n, n-1)$ entry converge to zero. Moreover convergence is rapid. Everyone believes the rate is cubic (very fast), although our proofs only guarantee a quadratic rate (such as Newton's iteration for polynomial zeros).[9]

The implementation for symmetric tridiagonals is particularly elegant, and we devote a few lines to evoke the procedure. The bulge-chasing method described earlier simplifies, because the bulge consists of a single entry on each side of the diagonal.

There is one more twist to the implementation that is worth mentioning. The code can be rearranged so that no square roots need to be computed.[9]

The discovery of the algorithms

There is no obvious benefit in factoring a square matrix B into $B = QR$ and then forming a new matrix $RQ = Q^* B Q$. Indeed, some structure in B might be lost in $Q^* B Q$.

So how did someone come up with the idea of iterating this transformation? Major credit is due to the Swiss mathematician and computer scientist H. Rutishauser. His doctoral thesis was not concerned with eigenvalues but rather with a more general algorithm he invented, which he called the Quotient-Difference (QD) Algorithm.[10] This procedure can be used to find zeros of polynomials or poles of rational functions or to manipulate continued fractions. The algorithm transforms an array of numbers, which Rutishauser writes as

$$Z = (q_1, e_1, q_2, e_2, \ldots, q_{n-1}, e_{n-1}, q_n)$$

into another one, $\hat{Z}$, of the same form.

Let us define two bidiagonal matrices associated with Z. For simplicity, take $n = 5$; then

$$L = \begin{pmatrix} 1 & & & & \\ e_1 & 1 & & & \\ & e_2 & 1 & & \\ & & e_3 & 1 & \\ & & & e_4 & 1 \end{pmatrix}, U = \begin{pmatrix} q_1 & 1 & & & \\ & q_2 & 1 & & \\ & & q_3 & 1 & \\ & & & q_4 & 1 \\ & & & & q_5 \end{pmatrix}.$$

Rutishauser observed that the rhombus rules he discovered for the QD transformation, namely

$$\begin{aligned} \hat{e}_i + \hat{q}_{i+1} &= q_{i+1} + e_{i+1} \\ \hat{e}_i \hat{q}_i &= q_{i+1} e_i \end{aligned},$$

admit the following remarkable interpretation:

$$\hat{L}\hat{U} = UL.$$

Note that UL and LU are tridiagonal matrices with all superdiagonal entries equal to one. In other words, the QD Algorithm is equivalent to the following procedure on tridiagonals $\mathcal{J}$ with unit superdiagonals:

$$\begin{aligned} &\text{Factor} \quad \mathcal{J} = LU, \\ &\text{Form} \quad \hat{\mathcal{J}} = UL. \end{aligned}$$

Thus was the LU transform born. It did not take Rutishauser a moment to see that the idea of reversing factors could be applied to a dense

matrix or a banded matrix. Although the QD Algorithm appeared in 1953 or 1954, Rutishauser did not publish his LU algorithm until 1958.[10,11] He called it LR, but I use LU to avoid confusion with QR.

Unfortunately, the LU transform is not always stable, so the hunt was begun for a stable variant. This variant was found by a young computer scientist J.G.F Francis,[7] greatly assisted by his mentor Christopher Strachey, the first professor of computation at Oxford University. Independent of Rutishauser and Francis, Vera Kublanovskaya in the USSR presented the basic QR Algorithm in 1961.[12] However, Francis not only gave us the basic QR Algorithm, but at the same time exploited the invariance of the Hessenberg form and gave the details of a double step to avoid the use of complex arithmetic.

In 1955, the calculation of the eigenvalues and eigenvectors of a real matrix that was not symmetric was considered a formidable task for which there was no reliable method. By 1965, the task was routine, thanks to the QR Algorithm. However, there is more to be done, because of accuracy issues. The eigenvalues delivered by QR have errors that are tiny, like the errors in rounding a number to 15 decimals, with respect to the size of the numbers in the original matrix. That is good news for the large eigenvalues but disappointing for any that are tiny. In many applications, the tiny eigenvalues are important because they are the dominant eigenvalues of the inverse.

For a long time, this limitation on accuracy was regarded as an intrinsic limitation caused by the limited number of digits for each number. Now we know that there are important cases where the data in the problem do define all the eigenvalues to high relative accuracy, and current work seeks algorithms that can deliver answers to that accuracy.

The QR Algorithm was aimed at matrices with at most a few hundred rows and columns. Its success has raised hopes. Now customers want to find a few eigenpairs of matrices with thousands, even hundreds of thousands, of rows. Most of the entries of these matrices are zero, and they are called sparse. The QR Algorithm is not well suited for such cases. It destroys sparsity and has difficulty finding selected eigenpairs. Since 1970, research has turned toward these challenging cases.[13]

References

1. A. Jennings and J.J. McKeown, *Matrix Computations*, 2nd ed., John Wiley, New York., 1977.
2. D.K. Faddeev and V.N. Faddeeva, *Computational Methods of Linear Algebra*, W.H. Freeman and Co., San Francisco, 1963.
3. E. Anderson et al., *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.
4. J.H. Wilkinson, "Convergence of the LR, QR, and Related Algorithms," *The Computer J.*, No. 4, 1965, pp. 77–84.
5. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins Univ. Press, Baltimore, 1996.
6. B.N. Parlett, "Global Convergence of the Basic QR Algorithm on Hessenberg Matrices," *Mathematics of Computation*, No. 22, 1968, pp. 803–817.
7. J.G.F Francis, "The QR Transformation, Parts I and II," *The Computer J.*, No. 4, 1961–1962, pp. 265–271 and 332–345.
8. S. Batterson and D. Day, "Linear Convergence in the Shifted QR Algorithm," *Mathematics of Computation*, No. 59, 1992, pp. 141–151.
9. B.N. Parlett, *The Symmetric Eigenvalue Problem*, 2nd ed., SIAM, Philadelphia, 1998.
10. H. Rutishauser, "Der Quotienten-Differenzen-Algorithmus," *Zeitung der Angewandte Mathematik und Physik*, No. 5, 1954, pp. 233–251.
11. H. Rutishauser, "Solution of Eigenvalue Problems with the LR-Transformation," *Nat'l Bureau Standards Applied Mathematics Series*, No. 49, 1958, pp. 47–81.
12. V.N. Kublanovskaya, "On Some Algorithms for the Solution of the Complete Eigenvalue Problem," *USSR Computational Mathematics and Physics*, No. 3, 1961, pp. 637–657.
13. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester Univ. Press, Manchester, UK, 1992.

Beresford N. Parlett is an emeritus professor of mathematics and of computer science at the University of California, Berkeley. His professional interest is in matrix computations, especially eigenvalue problems. He received his BA in mathematics from Oxford University and his PhD from Stanford University. He is a member of the AMS and SIAM. Contact him at the Mathematics Dept. and Computer Science Division, EECS Dept., Univ. of California, Berkeley, CA 94720; parlett@math.berkeley.edu.