The QR Algorithm

Beresford N. Parlett [*]

April 22, 2002

Contents

[*]Mathematics Department and Computer Science Division, EECS Department, University of California, Berkeley, CA 94720, USA.

Abstract

A brief sketch of the early days of eigenvalue hunting is followed by a description of 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 essay to a close.

1 Early History of Eigenvalue Computations

We assume that the reader shares the view that the rapid computation of the eigenvalues of a square matrix is a valuable tool for engineers and scientists. See [12] for eigenproblems in Engineering.

The QR algorithm solves the eigenvalue problem in a very satisfactory way but this success does not mean that 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.

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 is to use a two stage method. First compute the coefficients of the characteristic polynomial, more on this in Sections 2, 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. See [5], [8], [9]. The second stage was a hot topic of research during this same time period. Except for very small values of n, $n \leq 10$, this two stage approach is a disaster on a computer with fixed word length because the zeros of a polynomial are, in general, incredibly sensitive to tiny changes in the coefficients whereas the eigenvalues of a matrix 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 n coefficients of the characteristic polynomial 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. More on similarities in Sections 2. The more entries that are zero the nicer the matrix. Diagonal matrices are perfect but a triangular matrix is good enough for our purposes because the eigenvalues lie on the main diagonal. For deep theoretical reasons a triangular matrix is not attainable in a finite number of arithmetic operations, in general. 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 will do such as the square root of the sum of the squares of the entries. 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 get rid of those subdiagonal entries. This is where the QR algorithm comes to the rescue. For its discovery see Sections 4. The subroutine DHSEQR in the LAPACK library embodies the latest implementation. See [1].

Let us try to put the improvement based on QR in perspective. The cost of forming the product of two $n \times n$ dense matrices in a conventional way is $2n^3$ arithmetic operations. In 1950 there were no reliable ways to solve the eigenvalue problem. Fifty years later standard software computes all the eigenvalues of a dense symmetric matrix in about the same time as 1 (one!) conventional matrix multiply. To compute a triangular (Schur) decomposition of a real nonsymmetric $n \times n$ dense matrix ($B = QTQ^*$, T triangular, $Q^* = Q^{-1}$) costs about $25n^3$ arithmetic operations. Up until the 1960's floating point arithmetic operations dominated execution time but now the operations $+, -, *$ take little more time than data movement (cache access). So performance comparison is a complicated business. Nevertheless the QR algorithm has reduced the time for standard eigenvalue computations to the time required for a few matrix multiplies.

A feature that distinguishes numerical analysis from other branches of computer science is that the basic arithmetic operations are carried out on numbers with variable exponents but held to a fixed precision, to either eight or sixteen decimal digits approximately. This constriction permits incredibly fast execution but the price for lighting speed is that almost every arithmetic operation produces a very slightly wrong result. The effect of these little errors can be disastrous if intermediate quantities in a calculation are allowed to grow much larger than the original data.

2 The LU and QR Algorithms

Let us recall that if B is a $n \times n$ matrix with real or complex entries then the eigenvalue/eigenvector equation is

$$B\boldsymbol{v} = \boldsymbol{v}\lambda, \qquad \boldsymbol{v} \neq \boldsymbol{o}, \quad \boldsymbol{v} \text{ an } n\text{-vector.}$$

The characteristic polynomial of B is $\det(tI - B)$ and has degree n. Thus

$$\det(tI - B) = t^n + \sum_{i=0}^{n-1} c_i t^i$$

and $(1, c_{n-1}, \ldots, c_0)$ is the sequence of coefficients. The eigenvalues λ of B are the zeros of the characteristic polynomial. Since $\det(XY) = \det(X)\det(Y)$ it follows that similarity transformations, $B \longrightarrow X^{-1}BX =: C$, preserve eigenvalues; $\det(tI - C) = \det(tI - B)$.

We describe the algorithms here in a straightforward way. For their discovery see Sections 4.

Suppose that $B = XY$ with X invertible. Then the new matrix $C := YX$ is similar to B since $C = YX = X^{-1}BX$. This observation is intriguing but it is not clear that it is of much use to eigenvalue hunters. Let us recall two well known ways of factoring a matrix.

I. Triangular factorization (or Gaussian elimination),

$$B = LU$$

where L is lower triangular with ones 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.

II. The 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 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* $B_1 = B$,

$$\begin{array}{rrcl} \textit{Factor} & B_j & = & Q_j R_j, \\ \textit{Form} & B_{j+1} & = & R_j Q_j, \quad j = 1, 2, \ldots \end{array}$$

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

This result is not obvious and proofs may be found in [20]. The procedure that generates $\{B_j\}$ is called the *basic* QR algorithm.

A similar theorem holds for the basic LU algorithm provided that all the transforms exist. Let $C_1 := B$ and

$$\begin{array}{rrcl} \text{Factor} & C_j & = & L_j U_j, \\ \text{Form} & C_{j+1} & := & U_j L_j, \quad j = 1, 2, \ldots \end{array}$$

then $\{C_j\}$ converges to an upper triangular matrix with the eigenvalues in monotonic order down the diagonal. However this limit is *not* a Schur form of B because the transforms do not use unitary similarities that preserve the norm.

From a purely theoretical point of view there is not much more to be said. However 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 all its LU iterates. This useful result is easy to see. If B_j is Hessenberg then so is Q_j, since R_J^{-1} is triangular. Consequently $R_j Q_j = (B_{j+1})$ 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 $n \times n$. See [11]. A similar result holds for LU iterates.

Fortunately any matrix may be reduced by similarities to Hessenberg form in $O(n^3)$ arithmetic operations in a stable way and from this point onwards we shall assume that 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. See [15].

• 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. The situation can be improved by making a subtle change in our goal. Instead of looking at the matrix sequence $\{B_j\}$ we may focus on the $(n, n-1)$ entry of each matrix, the last subdiagonal entry. When the $(n, n-1)$ entry is negligible then the (n, n) entry is an eigenvalue, to within working precision, and column n does not influence the remaining eigenvalues. Consequently the variable n may be reduced by 1 and computation continues on the smaller matrix. We say that the nth eigenvalue has been *deflated.* Note that the top of the matrix need not be close to triangular form. Thus, in practice, 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 the simple device of translation, or shift of origin.

The mechanism behind this improvement is not hard to understand. In exact arithmetic if B is singular then $R_1 = Q_1^* B = Q_1^* B$ must be singular since Q_1 is unitary. The Hessenberg form dictates that it is the (n, n) entry of R_1 that must vanish. Hence the nth row of $B_2 = R_1^* Q_1$ vanishes and we may deflate, $n \longrightarrow n-1$, after one step. The implication of the observation is that we should apply the QR transform to $B - sI$ where s is our best guess at the smallest eigenvalue of B. Thus we arrive at 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 there is a (different) convergence theory for each shift strategy.

This is not the place to discuss shift strategy but in the current implementations (see [1]) it is rare that more than $2n$ QR iterations are needed to compute all the eigenvalues of B. The average number of iterations per eigenvalue is less than 2 over a huge class of test matrices.

• The Double Shift Implementation for Real Matrices

There is a clever variation on the shifted QR algorithm that should be mentioned. In many applications the initial matrix is real but some of the eigenvalues are complex. The shifted algorithm presented above must then be implemented in complex arithmetic in order to achieve quadratic convergence. The man who first presented the QR algorithm, J. G. F Francis, see [10], 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 that 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' which is a significant component of the QR success story. We will describe it briefly without detailed justification.

• 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 that $s_1 = s$, $s_2 = \bar{s} \neq s$. If B_1 is Hessenberg then so are all the B_i. Suppose that B_3 has positive subdiagonal entries. By the theorem both B_3 and Q_1Q_2 are determined by column 1 of Q_1Q_2, call it $\boldsymbol{q}$. Since R_2R_1 is upper triangular $\boldsymbol{q}$ is a multiple of the first column of

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

Since B_1 is Hessenberg the vector $\boldsymbol{q}$ is zero except in its top 3 entries.

The following 3 stage algorithm computes B_3. It makes use of orthogonal matrices H_j, $j = 1, 2, \dots, n-1$, 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.

Step 1. Compute the first three entries of $\boldsymbol{q}$.

Step 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'.

Step 3. Compute a sequence of matrices $H_2, \dots, H_{n-1}$ and $C_j = H_j^t C_{j-1} H_j$, $j = 2, \dots, n-1$ such that $C_{n-1} = H_{n-1}^t \cdots H_2^t C_1 H_2 \cdots H_{n-1}$ is a Hessenberg matrix with positive subdiagonal entries. More on the H_j below.

We claim that $C_{n-1} = B_3$. Recall that column 1 of H_j is $\boldsymbol{e}_1$ for $j > 1$. Thus $H_1 H_2 \cdots H_{n-1} \boldsymbol{e}_1 = H_1 \boldsymbol{e}_1 = \boldsymbol{q}/\|\boldsymbol{q}\| = (Q_1Q_2)\boldsymbol{e}_1$. Now $C_{n-1} = (H_1 \cdots H_{n-1})^t B_1 (H_1 \cdots H_{n-1})$ and $B_3 = (Q_1Q_2)^t B_1 (Q_1Q_2)$. The Implicit Q Theorem ensures us that B_3 and C_{n-1} are the same. It can be shown that if s is not an eigenvalue then C_{n-1} must have positive subdiagonal entries in exact arithmetic.

Step 3 involves $n-2$ minor steps at each of which only 3 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 $C_2 \longrightarrow H_2^t C_1 H_2$ 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 down until it falls off the bottom of the matrix and the Hessenberg form is restored. The transformation $B_1 \longrightarrow B_2$ 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 it is the $(n-1, n-2)$ entry that becomes negligible, not $(n, n-1)$.

There is an analogous procedure for computing a double step with the LU algorithm. Although the LU algorithm is somewhat faster than the QR algorithm it is not guaranteed to keep all intermediate quantities nicely bounded and for this reason it has been abandoned.

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. It is possible that a more complicated choice of shifts would produce a nicer theory but practical performance is excellent. See [2], [4], [3] and [6] for blemishes in the simple shift strategy.

3 The Symmetric Case

The QR transform preserves symmetry for real matrices and preserves the Hermitian property for complex matrices: $B \longrightarrow Q^*BQ$. It also preserves Hessenberg form. Since a symmetric Hessenberg matrix is tridiagonal, 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 $10\,n^2$ arithmetic operations. Recall that all the eigenvalues are real.

One reason that this case is worthy of a section to itself is its convergence theory. Recall 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 suggested by J. H. Wilkinson in the 1960's consists of computing both eigenvalues of the trailing 2×2 submatrix, the one in rows and columns $n-1$ and n, and choosing as shift the eigenvalue that is closer to the (n, n) entry. This choice is used from the very beginning and the beautiful fact is that the $(n-1, n)$ and $(n, n-1)$ entries *always* converge to zero. Moreover convergence is rapid. Everyone believes that the rate is cubic (very fast) although our proofs only guarantee a quadratic rate (like Newton's iteration for polynomial zeros). For more details see [16, Chap. 8].

The implementation for symmetric tridiagonals is particularly elegant and we devote a few lines to evoke the procedure. The bulge chasing method described in Sections 2 simplifies because the bulge consists of a single entry on each side of the diagonal. Moreover the orthogonal matrices H_i in Sections 2 may be replaced by what are called 'plane rotations'. A plane rotation equals the identity matrix except in a pair of adjacent rows and columns where it has the form

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

and θ is the angle of rotation. We show the changing part of the matrix before and after a properly chosen plane rotation. The bulge is denoted by the Greek letter β, new values by $'$ and an x does not change. Thus

$$\begin{pmatrix} x & a & \beta & 0 \\ a & b & e & 0 \\ \beta & e & d & f \\ 0 & 0 & f & x \end{pmatrix} \longrightarrow \begin{pmatrix} x & a' & 0 & 0 \\ a' & b' & e' & \beta' \\ 0 & e' & d' & f' \\ 0 & \beta' & f' & x \end{pmatrix}$$

The value of θ is given by

$$a \sin\theta = \beta \cos\theta$$

which dictates that $\cos\theta = a/\sqrt{a^2 + \beta^2}$.

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. This is possible because the eigenvalues depend on diagonal entries such as b and d above but only on the squares of the off-diagonal entries such as a, e, f. By using β^2, a^2, b, e^2, d, and f^2 it is possible to compute $\sin^2\theta$ and $\cos^2\theta$ and the new entries $(\beta')^2$, $(a')^2$, b', $(e')^2$, d', and $(f')^2$. The details are tricky and may be found in [16, Chap. 8] and [11].

4 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 $RQ = Q^*BQ$. Indeed some structure in B may be lost in Q^*BQ. 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 [17] was not concerned with eigenvalues but with a more general algorithm he invented and called the quotient-difference, or qd, algorithm. This procedure may be used for finding zeros of polynomials or poles of rational functions or manipulating 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 same form.

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

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

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 J with unit superdiagonals:

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

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 1953/54 it was not until 1958 that Rutishauser published his LU algorithm. See [17] and [18]. He called it LR but I use LU to avoid confusion with QR.

Unfortunately the LU transform is not always stable and so the hunt was begun for a stable variant. This was found by a young computer scientist J. G. F Francis, see [10], greatly assisted by his mentor C. Strachey, the first professor of Computation at Oxford University. Independently of Rutishauser and Francis the basic QR algorithm was presented by V. Kublanovskaya in the USSR in 1961, see [13] (in Russian) and [14] (English translation). 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.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide (second edition)*, SIAM, Philadelphia, 1995, 324 pages.

[2] S. Batterson, Convergence of the shifted QR algorithm on 3×3 normal matrices. Numer. Math. 58 (1990), pp. 341–352.

[3] S. Batterson, Convergence of the Francis shifted QR algorithm on normal matrices. Linear Algebra Appl. 207 (1994), pp. 181–195.

[4] S. Batterson and D. Day, Linear convergence in the shifted QR algorithm. Math. Comp. 59 (1992), pp. 141–151.

[5] E. Bodewig, *Matrix calculus.* 2nd revised and enlarged edition. North-Holland Publishing Co., Amsterdam; Interscience Publishers, Inc., New York 1959.

[6] D. Day, How the QR algorithm fails to converge and how to fix it. Sandia National Laboratory, Tech Report, 96-0913J, April, 1996.

[7] J. Demmel, *Applied Numerical Algebra.* SIAM Publications, 1997.

[8] V. N. Faddeeva, *Computational Methods of Linear Algebra.* Dover Publications, London, 1959.

[9] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra.* W. H. Freeman and Co., San Francisco, California, 1963.

[10] J. G. F Francis, 'The QR Transformation, Parts I and II' Computer J., 4 (1961-62), pp. 265–271, 332–345.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations.* The Johns Hopkins University Press, 3rd Edition, 1996, Baltimore, MD.

[12] A. Jennings and J. J. McKeown, *Matrix Computations.* John Wiley and Sons, 2nd Edition, N.Y., 1977.

[13] V. N. Kublanovskaya, On some algorithms for the solution of the complete eigenvalue problem. Zh. Vych. Mat., 1 (1961), pp. 555–570.

[14] V. N. Kublanovskaya, On some algorithms for the solution of the complete eigenvalue problem. USSR Comp. Math. Phys. 3 (1961), pp. 637–657.

[15] B. N. Parlett, Global convergence of the basic QR algorithm on Hessenberg matrices. Math. Comp. 22 (1968), pp. 803–817.

[16] B. N. Parlett, The Symmetric Eigenvalue Problem. SIAM, 2nd Edition, Philadelphia, 1998. 398 pp.

[17] H. Rutishauser, Der Quotienten-Differenzen-Algorithmus. Z. Angew. Math. Phys., 5 (1954), pp. 233–251.

[18] H. Rutishauser, Solution of eigenvalue problems with the LR-transformation. Nat. Bur. Standards Appl. Math. Series, 49 (1958), pp. 47–81.

[19] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[20] J. H. Wilkinson, Convergence of the LR, QR, and related algorithms. Computer J., 4 (1965), pp. 77–84.