THE RESTARTED QR-ALGORITHM FOR EIGENVALUE COMPUTATION OF STRUCTURED MATRICES

DANIELA CALVETTI*, SUN-MI KIM†, AND LOTHAR REICHEL‡

Dedicated to Bill Gragg on the occasion of his 65th birthday.

Abstract. The QR-algorithm is a popular numerical method for the computation of eigenvalues of matrices. All eigenvalues of a general $n \times n$ upper Hessenberg matrix typically can be computed in $\mathcal{O}(n^3)$ arithmetic floating point operations using $\mathcal{O}(n^2)$ storage locations. When the upper Hessenberg matrix is Hermitian or unitary, then it can be represented by $\mathcal{O}(n)$ parameters, and there are variants of the QR-algorithm that reduce the operation count for computing all eigenvalues to $\mathcal{O}(n^2)$ arithmetic floating point operations and the storage requirement to $\mathcal{O}(n)$ locations. However, for many structured matrices that can be represented with $\mathcal{O}(n)$ storage locations, available implementations of the QR-algorithm require $\mathcal{O}(n^3)$ arithmetic floating point operations and $\mathcal{O}(n^2)$ storage locations to determine all eigenvalues. This note shows that for some of the latter matrices, the operation count can be reduced to $\mathcal{O}(n^2)$ arithmetic floating point operations and the memory requirement to $\mathcal{O}(n)$ storage locations by periodically restarting the QR-algorithm.

1. Introduction.

The QR-algorithm is one of the most popular numerical methods for the computation of eigenvalues of matrices of small to moderate size; see, e.g., Golub and Van Loan [7, Chapters 7 and 8] and Gates and Gragg [6] for recent discussions and references. The computation of all eigenvalues of a general $n \times n$ upper Hessenberg matrix H typically requires $\mathcal{O}(n^3)$ arithmetic floating point operations (flops) and $\mathcal{O}(n^2)$ storage locations. The operation count and storage requirement can be reduced to $\mathcal{O}(n^2)$ flops and $\mathcal{O}(n)$ storage locations when the upper Hessenberg matrix H has a special structure, such as when H is Hermitian and tridiagonal or unitary; see [6, 7, 9, 10, 15]. However, small changes in the structure of H, e.g., a rank-one modification, may increase the operation count to $\mathcal{O}(n^3)$ flops and the storage requirement to $\mathcal{O}(n^2)$ locations.

This note discusses the application of periodically restarted QR-algorithms for the computation of all eigenvalues of comrade and fellow matrices. The former are rank-one modifications of Hermitian tridiagonal matrices and the eigenvalues are zeros of a linear combination of polynomials that satisfy a three-term recurrence relation. The latter matrices are rank-one modifications of unitary upper Hessenberg matrices. Their eigenvalues are zeros of a linear combination of Szegő polynomials. Comrade and fellow matrices can be represented with $\mathcal{O}(n)$ parameters; however, available implementations of the QR-algorithm cannot use the structure of these matrices and typically require $\mathcal{O}(n^3)$ flops and $\mathcal{O}(n^2)$ storage locations to compute all eigenvalues.

Periodic restarts of the QR-algorithm reduces the storage requirement when applied to comrade and fellow matrices to $\mathcal{O}(n)$ locations, and we illustrate with numerical examples that the number of iterations typically is smaller than twice the number of iterations required by the QR-algorithm without restarts. The operation count is generally $\mathcal{O}(n^2)$ flops. We refer to the QR-algorithm with restarts every k iterations as the QR(k)-algorithm. The QR-algorithm without restarts is referred to as the standard QR-algorithm.

* Department of Mathematics, Case Western Reserve University, Cleveland, OH 44106. E-mail: dxc57@po.cwru.edu. Research supported in part by NSF grant DMS-0107841.

† Department of Mathematical Sciences, Kent State University, Kent, OH 44242. E-mail: sukim@math.kent.edu. Research supported in part by NSF grant DMS-0107858

‡ Department of Mathematical Sciences, Kent State University, Kent, OH 44242. E-mail: reichel@math.kent.edu. Research supported in part by NSF grant DMS-0107858.

2. Eigenvalue computation of comrade matrices. Introduce the Hermitian tridiagonal matrix

$$T := \begin{bmatrix} \alpha_1 & \bar{\beta}_1 & & & & & 0 \\ \beta_1 & \alpha_2 & \bar{\beta}_2 & & & & \\ & \beta_2 & \alpha_3 & & & & \\ & & & \ddots & & & \\ & & \ddots & \ddots & \bar{\beta}_{n-2} & & \\ & & & \beta_{n-2} & \alpha_{n-1} & \bar{\beta}_{n-1} \\ 0 & & & & \beta_{n-1} & \alpha_n \end{bmatrix} \in \mathbb{C}^{n\times n}, \tag{2.1}$$

where the bar denotes complex conjugation. This section discusses the computation of the eigenvalues of comrade matrices, i.e., of matrices of the form

$$C = T + de_n^T, \tag{2.2}$$

where

$$d = [\delta_0, \delta_1, \ldots, \delta_{n-1}]^T \in \mathbb{C}^n \tag{2.3}$$

and e_j denotes the jth axis vector in $\mathbb{C}^n$. The eigenvalues of C are zeros of a linear combination of polynomials that are defined by the tridiagonal matrix T.

Example 2.1. Let the polynomials ψ_j satisfy the three-term recurrence relation

$$\psi_j(\tau) = (\tau - \alpha_j)\psi_{j-1}(\tau) - |\beta_{j-1}|^2\psi_{j-2}(\tau), \qquad j = 1, 2, \ldots, n, \tag{2.4}$$

where $\psi_{-1}(\tau) := 0$, $\psi_0(\tau) := 1$ and $\beta_0 := 0$. The eigenvalues of the comrade matrix (2.2) are the zeros of the polynomial

$$\Psi(\tau) := \sum_{j=0}^{n} \eta_j \psi_j(\tau),$$

where $\eta_j := -\delta_j\beta_{j+1}\cdots\beta_n$ for $0 \le j \le n-2$, $\eta_{n-1} := -\delta_{n-1}$, $\eta_n := 1$ and the δ_j are the entries of the vector d. $\Box$

Eigenvalue problems for comrade matrices arise, e.g., in the evaluation of computer graphics illumination models; see Borges [4] for a discussion.

Application of one step of the QR-algorithm to the comrade matrix C with shift σ_1 yields the new, similar, matrix

$$C_1 := \bar{Q}_1^T C Q_1,$$

where $Q_1 \in \mathbb{C}^{n\times n}$ is the unitary factor in a QR-factorization of $C - \sigma_1 I$ and I denotes the $n \times n$ identity matrix. The matrix C_1 can be expressed as

$$C_1 = T_1 + d_1^{(1)} e_{n-1}^T + d_1^{(0)} e_n^T, \tag{2.5}$$

where T_1 is a Hermitian tridiagonal $n \times n$ matrix and the $d_1^{(j)}$ are n-vectors. After k steps of the QR-algorithm with shifts $\sigma_1, \sigma_2, \ldots, \sigma_k$ applied to the comrade matrix C, we obtain the matrix C_k, which is unitarily similar to C and allows the representation

$$C_k = T_k + \sum_{j=0}^{k} d_k^{(j)} e_{n-j}^T, \tag{2.6}$$

where T_k is a Hermitian tridiagonal $n \times n$ matrix and the $d_k^{(j)}$ are n-vectors, such that C_k is of upper Hessenberg form.

The rate of convergence of the QR(k)-algorithm depends on the choice of shifts σ_j. In our numerical examples, we employ Wilkinson shifts. Assume that during the first k steps of the QR(k)-algorithm no eigenvalue has been determined with sufficient accuracy. We then compute the Wilkinson shift associated with the matrix C_k given by (2.6) and apply this shift to the comrade matrix C given by (2.2). This yields a matrix, denoted by C_1, which is unitarily similar to C and has a representation of the form (2.5). The next shift is applied to this matrix C_1. We continue to apply shifts until an eigenvalue has been determined to sufficient accuracy or a matrix of the form (2.6) has been computed. In the latter case the Wilkinson shift is computed and applied to the original comrade matrix C. This defines the restarted QR(k)-algorithm. Each step of the QR(k)-algorithm requires $\mathcal{O}(n)$ flops for any fixed positive integer k.

When one of the matrices C_j of the form (2.6) with $1 \leq j \leq k$ generated during the iterations with the QR-algorithm has a "tiny" last subdiagonal entry, the last diagonal entry is accepted as an eigenvalue of C. This eigenvalue is then used as shift in a step of the QR-algorithm applied to C and we obtain a matrix C_1 of the form (2.5). If a subdiagonal entry of C_1 is tiny, then the eigenvalue problem splits. In particular, if the last subdiagonal entry of C_1 is tiny, then the eigenvalues of the leading $(n-1) \times (n-1)$ principal submatrix remain to be computed. This submatrix is a comrade matrix, and its eigenvalues can be determined by the restarted QR(k)-algorithm. If none of the subdiagonal entries are tiny, then the same shift is applied again. The matrix C_2 obtained in this manner typically has a tiny subdiagonal entry and the eigenvalue problem splits.

The convergence of the QR(k)-algorithm is the same as for the standard QR-algorithm between restarts. We have found that when the last subdiagonal element of the matrix C_k is of fairly small magnitude ρ, the Wilkinson shift associated with C_k applied to the comrade matrix C generally yields a matrix of the form (2.5) with the last subdiagonal entry of magnitude about ρ.

Example 2.2. Let the last subdiagonal entry of C be of magnitude 10^{-1}. Then typically the restarted QR(2)-algorithm yields a sequence of matrices whose last subdiagonal entry are roughly of magnitude 10^{-2}, 10^{-4}, 10^{-4}, 10^{-8}, 10^{-8}, 10^{-16}. $\square$

We conclude this section with a computed example that illustrates the performance of the QR(k)-algorithm for a few values of k. All computations were carried out using Matlab with about 16 significant decimal digits. For each $n \in \{10, 20, 35, 50, 100\}$, we generated 100 comrade matrices (2.2) of order n. The entries α_j and β_j were chosen to be uniformly distributed random numbers in the open interval $(-1, 1)$, and the entries δ_j were uniformly distributed random numbers in the interval $(-100, 100)$. Let C denote one of these comrade matrices and let $\lambda(C)$ denote the set of eigenvalues computed by the Matlab function eig. Let $\tilde{\lambda}(C)$ denote the set of eigenvalues computed by the QR(k)-algorithm, and define the distance between the sets $\lambda(C)$ and $\tilde{\lambda}(C)$ by

$$\text{distance}(\tilde{\lambda}(C), \lambda(C)) := \max\{ \max_{\tilde{\zeta} \in \tilde{\lambda}(C)} ||\tilde{\zeta} - \lambda(C)||, \max_{\zeta \in \lambda(C)} ||\zeta - \tilde{\lambda}(C)|| \},$$

where $||\tilde{\zeta} - \lambda(C)|| := \min_{\zeta \in \lambda(C)} |\tilde{\zeta} - \zeta|$. We refer to this distance as the error in the eigenvalues computed by the QR(k)-algorithm. Thus, we tacitly assume that the Matlab function eig computes the eigenvalues exactly.

Table 2.1 displays the average number of steps required by the restarted and nonrestarted QR-algorithms to compute all the eigenvalues of all the matrices of a

n	QR	QR(2)	QR(3)	QR(4)
10	26.40	51.20	45.52	43.02
20	52.33	103.39	94.84	91.11
35	89.24	176.89	166.49	162.63
50	123.03	245.07	234.65	230.09
100	239.95	478.58	461.53	459.12

TABLE 2.1
Average number of QR-steps.

n	QR	QR(2)	QR(3)	QR(4)
10	0.1764	0.1185	0.1169	0.1162
20	0.1955	0.1320	0.1332	0.1318
35	0.2487	0.1677	0.1712	0.1685
50	0.2517	0.1627	0.1699	0.1715
100	0.3838	0.3443	0.3436	0.3423

TABLE 2.2
Average error $\times 10^{13}$.

particular order n. The last diagonal entry of a matrix C_j is accepted as an eigenvalue of C when the last subdiagonal entry of C_j is of magnitude smaller than or equal to $1 \cdot 10^{-16}$. The average number of steps required by the QR(k)-algorithm is seen to decrease as k increases. The number of steps required by the QR(k)-algorithm is within a factor two of the number of steps required by the unrestarted algorithm. The error in the computed eigenvalues is shown by Table 2.2. The table indicates that all algorithms gave about the same accuracy.

The increase in the number of auxiliary vectors $d_k^{(j)}$ in the representation (2.6) with the number of steps k of the standard QR-algorithm leads to a linear increase with k in computational work required for each step as k increases. The operation count required to compute all eigenvalues of a comrade matrix, using the structure (2.6), therefore generally is $\mathcal{O}(n^3)$ flops.

The computational work required for each step of the restarted QR(k)-algorithm applied to a comrade matrix is $\mathcal{O}(n)$ flops independently of the number of steps already carried out. This operation count assumes that the unitary matrices, such as Q_1, required to carry out the similarity transformations in the QR(k)-algorithm are represented by a product of Givens matrices. For instance, Q_1 is represented by $n-1$ Givens matrices.

Counting the number of Givens matrices required to carry out each step of the standard QR- and restarted QR(k)-algorithms for the matrices of Table 2.1 shows that the standard QR-algorithm determined the first two eigenvalues with fewer arithmetic floating point operations than the restarted QR(k)-algorithm for $k = 2, 3, 4$. However, from the third eigenvalue and onwards, until due to deflation only an eigenvalue problem for a small matrix remained, the QR(k)-algorithm computed each eigenvalue with less arithmetic work than the standard QR-algorithm for $k = 2, 3, 4$. Despite the fact that the QR(k)-algorithm required more steps, it demanded less computational work for the computation of all eigenvalues than the standard QR-algorithm; only $\mathcal{O}(n^2)$ flops were required for $k = 2, 3, 4$. The operation count for the standard QR-algorithm was $\mathcal{O}(n^3)$ flops.

3. Eigenvalue computation of fellow matrices. This section discusses the computation of the eigenvalues of matrices of the form

$$F = H + de_n^T, \tag{3.1}$$

where H is a unitary upper Hessenberg matrix with positive subdiagonal entries and the vector d is given by (2.3). We refer to matrices of the form (3.1) as fellow matrices.

The matrix H can be expressed as a product of n elementary unitary matrices

$$H = G_1(\gamma_1)G_2(\gamma_2)\cdots G_{n-1}(\gamma_{n-1})G_n(\tilde{\gamma}_n), \tag{3.2}$$

where, for $1 \leq j < n$, $G_j(\gamma_j)$ denotes the Givens matrix

$$G_j(\gamma_j) := \begin{bmatrix} I_{j-1} & & & \\ & -\gamma_j & \sigma_j & \\ & \sigma_j & \overline{\gamma}_j & \\ & & & I_{n-j-1} \end{bmatrix} \in \mathbb{C}^{n\times n}. \tag{3.3}$$

The entries $\gamma_j \in \mathbb{C}$ are often referred to as Schur parameters; they lie in the open unit disk. We define the auxiliary parameters $\sigma_j := (1 - |\gamma_j|^2)^{1/2}$. The last Schur parameter $\tilde{\gamma}_n \in \mathbb{C}$ lies on the unit circle and determines the unitary matrix

$$G_n(\tilde{\gamma}_n) := \mathrm{diag}[1, 1, \ldots, 1, -\tilde{\gamma}_n] \in \mathbb{C}^{n\times n}.$$

The representation (3.2) of unitary upper Hessenberg matrices with positive subdiagonal entries was first exploited by Gragg [8], and has lead to the development of several fast algorithms for the eigenvalue problem for unitary upper Hessenberg matrices, among them a QR-algorithm [9, 10, 15] and a divide-and-conquer method [11, 12, 3].

The eigenvalues of fellow matrices are zeros of a linear combination of Szegő polynomials. The latter are determined by recursion formulas with the Schur parameters as recursion coefficients. For notational convenience, we introduce the Schur parameter $\gamma_n \in \mathbb{C}$ in the open unit disk.

Example 3.1. Let the polynomials ϕ_j satisfy the recurrence relation

$$\sigma_j\phi_j(\zeta) = \zeta\phi_{j-1}(\zeta) + \gamma_j\phi_{j-1}^*(\zeta), \qquad j = 1, 2, \ldots, n, \tag{3.4}$$

where $\phi_0(\zeta) := 1$ and the $\phi_j^*(\zeta) := \zeta^j\bar{\phi}_j(\zeta^{-1})$, $0 \leq j < n$, are auxiliary polynomials. The ϕ_j are known as Szegő polynomials. Many of their properties are discussed by Grenander and Szegő [13]. In particular, they are orthogonal with respect to an inner product on the unit circle.

The zeros of ϕ_n are the eigenvalues of the matrix

$$\hat{H} := G_1(\gamma_1)G_2(\gamma_2)\cdots G_{n-1}(\gamma_{n-1})\tilde{G}_n(\gamma_n).$$

This is a fellow matrix; it can be expressed as $\hat{H} = H + ce_n^T$, where H is the unitary upper Hessenberg matrix (3.2) and $c := (-\gamma_n + \tilde{\gamma}_n)G_1(\gamma_1)G_2(\gamma_2)\cdots G_{n-1}(\gamma_{n-1})e_n$.

Similarly, the zeros of a linear combination of Szegő polynomials

$$\Phi(\zeta) := \sum_{j=0}^{n} \eta_j\phi_j(\zeta)$$

are the eigenvalues of a fellow matrix. Specifically, they are eigenvalues of

$$F = \hat{H} + xe_n^T = H + de_n^T, \tag{3.5}$$

where $x := [\xi_0, \xi_1, \ldots, \xi_{n-1}]^T$ is determined by $\eta_j := -\xi_j \sigma_{j+1} \cdots \sigma_{n-1}$, $0 \le j \le n-2$, $\eta_{n-1} := -\xi_{n-1}$, and $\eta_n := 1$. The vector d in (3.5) therefore is $d = c + x$. □

The need to compute eigenvalues of fellow matrices arises in many applications. For instance, Szegő polynomials arise in time series analysis and knowledge of the location of their zeros is important for frequency analysis of time series and for filter implementation. A short review of this application is presented in [2]; more details can be found in [5, 14]. Moreover, the determination of eigenvalues of a companion matrix can be transformed into the problem of computing eigenvalues of a fellow matrix. This observation forms the basis of a new accurate polynomial zero-finder described in [1].

Application of one step of the QR-algorithm with shift σ_1 to the fellow matrix (3.1) yields the new, similar, matrix

$$F_1 := \bar{Q}_1^T F Q_1,$$

where $Q_1 \in \mathbb{C}^{n\times n}$ is the unitary factor in a QR-factorization of $F - \sigma_1 I$. The matrix F_1 has the representation

$$F_1 = H_1 + d_1^{(1)} e_{n-1}^T + d_1^{(0)} e_n^T, \tag{3.6}$$

where H_1 is a unitary upper Hessenberg $n \times n$ matrix and the $d_1^{(j)}$ are n-vectors. After k steps of the QR-algorithm applied to the fellow matrix (3.1), we obtain a matrix of the form

$$F_k = H_k + \sum_{j=0}^{k} d_k^{(j)} e_{n-j}^T, \tag{3.7}$$

where H_k is a unitary upper Hessenberg $n \times n$ matrix and the $d_k^{(j)}$ are n-vectors, such that F_k is of upper Hessenberg form. Each iteration requires the selection of a shift σ_j. In our numerical examples, we apply Wilkinson shifts.

The restarted QR(k)-algorithm for fellow matrices is analogous to the restarted QR(k)-algorithm for comrade matrices. One step of the QR-algorithm for unitary upper Hessenberg $n \times n$ matrices introduced by Gragg [9, 10] requires $\mathcal{O}(n)$ flops. This algorithm uses the representation (3.2). Using this representation of the matrix H_k in (3.7), one step of the QR-algorithm applied to the matrix F_k can be carried out in $\mathcal{O}(n)$ flops provided that k is bounded independently of n.

We conclude this section with a computed example. For each $n \in \{10, 20, 35, 50, 100\}$, we generated 100 fellow matrices (3.1) of order n. The Schur parameters γ_j, $1 \le j < n$, in the open unit disk, the Schur parameter $\tilde{\gamma}_n$ on the unit circle and the coefficients δ_j, $0 \le j \le n$, in the disk $\{\zeta : |\zeta| < 100\}$ are all randomly generated. We accepted the last diagonal entry of a matrix F_k as an eigenvalue under the same condition as in Section 2.

Table 3.1 is analogous to Table 2.1 and shows that the QR(k)-algorithm for $k = 2, 3, 4$ requires at most twice the number of steps as the nonrestarted algorithm. Table 3.2 is similar to Table 2.2 and displays that all methods give about the same accuracy.

A careful operation count shows that the restarted QR(k)-algorithm determines all but the first few eigenvalues and the very last few eigenvalues faster than the

n	QR	QR(2)	QR(3)	QR(4)
10	28.92	52.59	47.84	45.44
20	58.21	113.29	104.72	100.44
35	98.82	202.01	187.96	183.26
50	138.50	286.07	268.57	264.79
100	268.55	565.40	533.46	529.65

TABLE 3.1
Average number of QR-steps.

n	QR	QR(2)	QR(3)	QR(4)
10	0.2102	0.1271	0.1346	0.1344
20	0.2034	0.1372	0.1395	0.1401
35	0.2373	0.1606	0.1623	0.1614
50	0.2465	0.1483	0.1527	0.1518
100	0.4805	0.4504	0.4438	0.4441

TABLE 3.2
Average error $\times 10^{13}$.

standard QR-algorithm for $k = 2, 3, 4$ if the matrix is not very small. The operation count for the restarted QR(k)-algorithm was $\mathcal{O}(n^2)$ flops for $k = 2, 3, 4$, while the standard QR-algorithm required $\mathcal{O}(n^3)$ flops.

Acknowledgement. We would like to thank Bill Gragg for valuable comments.

REFERENCES

[1] G. S. Ammar, D. Calvetti, W. B. Gragg and L. Reichel, Polynomial zerofinders based on Szegő polynomials, J. Comput. Appl. Math., 127 (2001), pp. 1–16.
[2] G. S. Ammar, D. Calvetti and L. Reichel, Continuation methods for the computation of zeros of Szegő polynomials, Linear Algebra Appl., 249 (1996), pp. 125–155.
[3] G. S. Ammar, L. Reichel and D. C. Sorensen, Algorithm 730: An implementation of a divide and conquer algorithm for the unitary eigenproblem, ACM Trans. Math. Software, 18 (1992), pp. 292–307 and 20 (1994), p. 161.
[4] C. F. Borges, On a class of Gauss-like quadrature rules, Numer. Math., 67 (1994), pp. 271–288.
[5] J. A. Cadzow, Foundations of Digital Signal Processing and Data Analysis, MacMillan, New York, 1987.
[6] K. Gates and W. B. Gragg, Notes on TQR algorithms, J. Comput. Appl. Math., 86 (1997), pp. 195–203.
[7] G. H. Golub and C. F. Van Loan, Matrix Computations, 3rd ed., John Hopkins Univ. Press, Baltimore, 1996.
[8] W. B. Gragg, Positive definite Toeplitz matrices, the Arnoldi process for isometric operators, and Gaussian quadrature on the unit circle (in Russian), in Numerical Methods in Linear Algebra, ed. E. S. Nikolaev, Moscow U. P., Moscow, 1982, pp. 16–32. Published in English in revised form in J. Comput. Appl. Math., 46 (1993), pp. 183–198.
[9] W. B. Gragg, The QR algorithm for unitary Hessenberg matrices, J. Comput. Appl. Math., 16 (1986), pp. 1–8.
[10] W. B. Gragg, Stabilization of the UHQR algorithm, in Advances in Computational Mathematics, eds. Z. Chen, Y. Li, C. Micchelli and Y. Xu, Lecture Notes in Pure and Applied Mathematics, vol. 202, Marcel Dekker, Hong Kong, 1998, pp. 139–154.
[11] W. B. Gragg and L. Reichel, A divide and conquer method for the unitary eigenproblem, in Hypercube Multiprocessors 1987, ed. M. T. Heath, SIAM, Philadelphia 1987, pp. 639–647.
[12] W. B. Gragg and L. Reichel, A divide and conquer method for the unitary and orthogonal eigenproblems, Numer. Math., 57 (1990), pp. 695–718.
[13] U. Grenander and G. Szegő, Toeplitz Forms and Their Applications, 2nd ed., Chelsea, New

York, 1984.
[14] T. Kailath, Linear estimation for stationary and near-stationary processes, in Modern Signal Processing, ed. T. Kailath, Hemisphere, New York, 1985, pp. 59–128.
[15] T.-L. Wang and W. B. Gragg, Convergence of the shifted QR algorithm for unitary Hessenberg matrices, Math. Comp., in press.