Electronic Transactions on Numerical Analysis.
Volume 18, pp. 137-152, 2004.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

ON THE SHIFTED QR ITERATION APPLIED TO COMPANION MATRICES*

DARIO A. BINI †, FRANCESCO DADDI ‡, AND LUCA GEMIGNANI §

Abstract. We show that the shifted QR iteration applied to a companion matrix F maintains the weakly semiseparable structure of F. More precisely, if $A_i - \alpha_i I = Q_i R_i$, $A_{i+1} := R_i Q_i + \alpha_i I$, $i = 0, 1, \ldots$, where $A_0 = F$, then we prove that Q_i, R_i and A_i are semiseparable matrices having semiseparability rank at most 1, 4 and 3, respectively. This structural property is used to design an algorithm for performing a single step of the QR iteration in just $O(n)$ flops. The robustness and reliability of this algorithm is discussed. Applications to approximating polynomial roots are shown.

Key words. companion matrices, QR factorization, QR iteration, semiseparable matrices, eigenvalues, polynomial roots.

AMS subject classifications. 65F15, 15A18, 65H17.

1. Introduction. Let $p(x) = \sum_{i=0}^{n} a_i x^i$, be a polynomial of degree n with leading coefficient $a_n = 1$, let $\boldsymbol{f} = (f_i) \neq \boldsymbol{0}$ in $\mathbb{C}^n$ be such that $f_i = -a_{i-1}$, $i = 1 : n$ and consider the associated companion matrix

$$F = F(\boldsymbol{f}) = \begin{bmatrix} 0 & \dots & 0 & f_1 \\ 1 & \ddots & \vdots & f_2 \\ & \ddots & 0 & \vdots \\ O & & 1 & f_n \end{bmatrix}. \tag{1.1}$$

Since the eigenvalues of F coincide with the zeros of $p(x)$, algorithms for computing matrix eigenvalues can be applied for approximating the zeros of $p(x)$. In fact, the Matlab function `roots`, which provides approximations to the zeros of a polynomial, is based on the shifted QR iteration

$$\begin{aligned} A_k - \alpha_k I &= Q_k R_k \\ A_{k+1} &= R_k Q_k + \alpha_k I = Q_k^H A_k Q_k \end{aligned} \tag{1.2}$$

$k = 0, 1, 2, \ldots$, applied with $A_0 = F$, suitably balanced by means of a diagonal scaling. Here and hereafter, Q^H denotes the transpose conjugate of Q.

Despite F being sparse and structured, after a few steps of (1.2), the matrices A_k are dense and apparently with no structure, except that they are in upper Hessenberg form. In this way, the arithmetic cost of each iteration is $O(n^2)$ arithmetic floating point operations (flops) and the QR algorithm does not seem to take advantage of the initial structure of F. In [4] an attempt to overcome this drawback has been performed by using a restarting technique. In [1], [18], the related problem of computing the eigenvalues of a unitary matrix in Hessenberg form is investigated.

Observe that the matrix F is such that any submatrix contained in the lower triangular part of F has rank at most 1. The same rank property holds for the submatrices contained in the upper triangular part (diagonal included). In general, we call a matrix A *weakly semiseparable of rank* (h, k) if all the submatrices in the strictly lower triangular part have rank at most h, and the submatrices in the strictly upper triangular part have rank at most k, where the values h and k are achieved by some submatrix. We say that A is *semiseparable*

*Received October 31, 2003. Accepted for publication August 30, 2004. Recommended by L. Reichel.
†Dipartimento di Matematica, Università di Pisa, via Buonarroti 2, 56127 Pisa. E-mail: bini@dm.unipi.it. This work was partially supported by MIUR, grant number 2002014121 and by GNCS-INDAM grant “Metodi numerici innovativi per il trattamento di matrici strutturate e sparse.”
‡Dipartimento di Matematica, Università di Pisa, via Buonarroti 2, 56127 Pisa.
§Dipartimento di Matematica, Università di Pisa, via Buonarroti 2, 56127 Pisa. E-mail: gemignan@dm.unipi.it.

ETNA
Kent State University
etna@mcs.kent.edu

of rank (h, k) if there exist matrices L and U of rank h and k, respectively, such that the strictly lower triangular part Tril(A) of A coincides with the strictly lower triangular part Tril(L) of L and the strict upper triangular part Triu(A) of A coincides with the strict upper triangular part Triu(U) of U. According to these definitions, F is weakly semiseparable of rank $(1, 1)$. In particular, Tril(F) is weakly semiseparable of rank $(1, 0)$ while Triu(F) is semiseparable of rank $(0, 1)$ since Triu(F) =Triu$(\boldsymbol{f}\boldsymbol{e}_n^T)$.

Semiseparable matrices are closely related to the inverses of banded matrices and have further nice properties which relate them to their inverses and to their LU and QR factorizations (see [5], [6], [7], [9], [11], [12], [13], [14], [16], [17] and the references therein). In particular, the matrices L_i and U_i, generated by the LU iteration applied to a semiseparable matrix for eigenvalue computation, are semiseparable [11]. Based on the well-know relations between the QR and LR iteration algorithms [19], it can be shown that a similar property holds for the matrices Q_i and R_i generated by the QR iteration provided that A is real symmetric [8]. Counterexamples easily can be given where the semiseparability property is not maintained by the QR iteration if A is not symmetric. Algorithms for computing the LU factorization and for solving linear systems with a semiseparable matrix in $O(n)$ ops have been designed in [11], [5]. Weakly semiseparable matrices have been introduced in [15].

In this paper we prove that the matrices A_k, Q_k and R_k generated by (1.2) for $k = 1, 2, \ldots$, with $A_0 = F$, are weakly semiseparable of rank $(1, 3)$, $(1, 2)$ and $(1, 4)$, respectively. Moreover, Triu(A_k) and Triu(Q_k) are semiseparable of rank $(0, 3)$ and $(0, 1)$ respectively. These facts enable us to represent Q_k, A_k and R_k with $O(n)$ memory space and to design algorithms for computing A_{k+1}, given A_k, with complexity $O(n)$.

The paper is organized in the following way. In section 2 we prove the structural properties of A_k, Q_k and R_k by providing an algorithm for computing the QR factorization of $A_k - \alpha_k I$ in $O(n)$ ops. In section 3 we describe different methods for performing the RQ step, still in $O(n)$ ops. Section 4 contains some remarks concerning the implementation of the algorithms. In section 5 we report the results of some numerical experiments which show that the provided algorithms are promising but still lack the robustness properties needed for a reliable use inside a numerical package. Future research is also discussed.

2. Structural properties of the matrix sequences generated by the QR iteration. In this section we analyze the structural properties of the matrix sequences $\{A_k\}_{k\in\mathbb{N}}$, $\{Q_k\}_{k\in\mathbb{N}}$, $\{R_k\}_{k\in\mathbb{N}}$, generated by the shifted QR iteration (1.2) applied with $A_0 = F$, where F is the companion matrix (1.1) associated with the vector $\boldsymbol{f} = (f_i) \in \mathbb{C}^n$, and we assume that $\det F \neq 0$, i.e., $f_1 \neq 0$.

Let us recall that from (1.2) it follows that

$$
\begin{aligned}
A_k &= P_k^H F P_k, \\
P_k &= Q_{k-1} Q_{k-2} \cdots Q_1,
\end{aligned}
\tag{2.1}
$$

that is, A_k is unitarily similar to F.

2.1. Structure of A_k. We note that, since F is in upper Hessenberg form, all the matrices A_k, $k = 0, 1, \ldots$, are in upper Hessenberg form, that is, $a_{i,j}^{(k)} = 0$ if $i > j + 1$, where $A_k = (a_{i,j}^{(k)})$.

The following simple observation plays a substantial role for the derivation of our results.

REMARK 1. If $f_1 \neq 0$ then a direct inspection shows that

$$
F(\boldsymbol{f})^{-1} = \begin{bmatrix}
-f_2/f_1 & 1 & & & \\
-f_3/f_1 & 0 & 1 & & \\
\vdots & & \ddots & \ddots & \\
-f_n/f_1 & & & & 1 \\
1/f_1 & & & & 0
\end{bmatrix}.
\tag{2.2}
$$

Moreover,

$$F = F^{-H} + UV^H, \quad U, V \in \mathbb{C}^{n\times 2}, \tag{2.3}$$

where

$$U = \begin{bmatrix} -1 & f_1 \\ 0 & f_2 \\ \vdots & \vdots \\ 0 & f_n \end{bmatrix}, \quad V = \begin{bmatrix} f_2/f_1 & 1 \\ f_3/f_1 & 0 \\ \vdots & \vdots \\ f_n/f_1 & 0 \\ -1/f_1 & 0 \end{bmatrix}. \tag{2.4}$$

Pre-multiplying equation (2.3) by P_k^H and post-multiplying it by P_k, from (2.1) we immediately obtain the following identity

$$A_k = A_k^{-H} + U_k V_k^H, \quad k = 0, 1, \ldots, \tag{2.5}$$

where

$$U_k = Q_k^H U_{k-1}, \quad V_k = Q_k^H V_{k-1}. \tag{2.6}$$

Since A_k is in upper Hessenberg form, the entries in the upper triangular part of A_k^{-H} are given by

$$(A_k^{-H})_{i,j} = \frac{1}{y_n^{(k)}} x_i^{(k)} y_j^{(k)}, \quad i \le j, \tag{2.7}$$

where $\boldsymbol{x} = (x_i^{(k)}) = A_k^{-H}\boldsymbol{e}_n$ and $\boldsymbol{y}^{(k)T} = (y_i^{(k)}) = \boldsymbol{e}_1^T A_k^{-H}$ are the last column and the first row of A_k^{-H}, respectively. This property is a direct consequence of the following

LEMMA 2.1. *Let $B = (b_{i,j})$ be a matrix in lower Hessenberg form, and define $\beta_i = b_{i,i+1}$, $\hat{d}_i = \det \widehat{B}_i$, $\tilde{d}_i = \det \widetilde{B}_i$, where $\widehat{B}_i$ and $\widetilde{B}_i$ are the leading and trailing $i \times i$ principal submatrices of B, respectively. Then*

$$(B^{-1})_{i,j} = \frac{(-1)^{i+j}}{\det B} \hat{d}_{i-1} \tilde{d}_{n-j} \prod_{\ell=i}^{j-1} \beta_\ell, \quad i \le j,$$

where, if $i = j$, the product in the right hand side is 1, and $\hat{d}_0 = \tilde{d}_n = 1$.

Proof. The result follows from the relation $(B^{-1})_{i,j} = (-1)^{i+j} \det B_{i,j} / \det B$, where $B_{i,j}$ is the submatrix of B obtained by removing the j-th row and the i-th column. □

From this lemma we may also represent the upper triangular part of A_k^{-H}, by means of a pair of vectors $\boldsymbol{w}^{(k)} = (w_i^{(k)}) = ((-1)^{i-1}\hat{d}_i^{(k)})$, $\boldsymbol{z}^{(k)} = (z_i^{(k)}) = ((-1)^{n-i}\tilde{d}_{n-i}^{(k)})$, as

$$(A_k^{-H})_{i,j} = w_i^{(k)} z_j^{(k)} \frac{1}{\bar{f}_1} \prod_{\ell=i}^{j-1} \bar{b}_\ell^{(k)}, \quad i \le j, \tag{2.8}$$

where $b_\ell^{(k)} = a_{\ell+1,\ell}^{(k)}$, $\ell = 1 : n-1$, are the lower diagonal entries of A_k, $\hat{d}_i^{(k)}$ and $\tilde{d}_i^{(k)}$ are the determinants of the leading and trailing $i \times i$ principal submatrices of A_k^H, respectively, and we use the fact that $\det A_k = \det F^H = (-1)^{n+1}\bar{f}_1$. Here and below, we denote with $\bar{a}$ the complex conjugate of the complex number a.

We have the following representation result for A_k.

THEOREM 2.2. *The matrix $A_k = (a_{i,j}^{(k)})$, generated at the k-th step of the QR iteration, is such that*

$$a_{i,j}^{(k)} = \begin{cases} 0 & \text{for } i > j+1, \\ \dfrac{1}{y_n^{(k)}} x_i^{(k)} y_j^{(k)} + u_{i,1}^{(k)} \bar{v}_{j,1}^{(k)} + u_{i,2}^{(k)} \bar{v}_{j,2}^{(k)} & \text{for } i \le j, \end{cases}$$

ETNA
Kent State University
etna@mcs.kent.edu

where

$$\frac{1}{y_n^{(k)}} x_i^{(k)} y_j^{(k)} = w_i^{(k)} z_j^{(k)} \frac{1}{\bar{f}_1} \prod_{\ell=i}^{j-1} \bar{b}_\ell^{(k)},$$

and $b_\ell^{(k)} = a_{\ell+1,\ell}^{(k)}$, $\ell = 1 : n-1$.

Summing up, all the matrices A_k generated by the QR iteration are determined by $7n-1$ parameters, namely:

(i) The subdiagonal entries $\boldsymbol{b}^{(k)} = (b_1^{(k)}, \dots, b_{n-1}^{(k)})$.

(ii) The last column $\boldsymbol{x}^{(k)}$ and the first row $\boldsymbol{y}^{(k)T}$ of A_k^{-H}, respectively, defining the upper triangular part of A_k^{-H}.

(iii) The columns $\boldsymbol{u}_1^{(k)}, \boldsymbol{u}_2^{(k)}$ of U_k,

(iv) The columns $\boldsymbol{v}_1^{(k)}, \boldsymbol{v}_2^{(k)}$ of V_k.

In particular, the matrices A_k are weakly semiseparable of rank $(1,\nu)$, with $\nu \le 3$, and Triu(A_k) is semiseparable of rank at most 3.

2.2. Structure of R_k. The structure of the sequence $\{R_k\}_{k\in\mathbb{N}}$ easily can be described from the QR factorization of $A_k - \alpha_k I$. Therefore in this section we describe the QR step applied to A_k and, as a byproduct, we obtain the structure of R_k.

Let us recall that A_k is of upper Hessenberg form with lower diagonal entries $b_1^{(k)}, \dots, b_{n-1}^{(k)}$, and with entries $a_{i,j}^{(k)} = \frac{1}{y_n^{(k)}} x_i^{(k)} y_j^{(k)} + u_{i,1}^{(k)} \bar{v}_{j,1}^{(k)} + u_{i,2}^{(k)} \bar{v}_{j,2}^{(k)}$ for $i \le j$, i.e., in the upper triangular part. Let us also denote, with $\boldsymbol{a}^{(k)} = (a_i^{(k)})$, the vector formed by the diagonal entries of A_k and, with $\boldsymbol{d}^{(k)} = (d_i^{(k)})$, $\boldsymbol{g}^{(k)} = (g_i^{(k)})$, the vectors formed by the diagonal and superdiagonal entries of R_k, respectively. The reduction to upper triangular form of $A_k - \alpha_k I$ can be achieved by means of a sequence of Givens rotations

$$G_i^{(k)} = \begin{bmatrix} I_{i-1} & & & \\ & c_i^{(k)} & s_i^{(k)} & \\ & -\bar{s}_i^{(k)} & c_i^{(k)} & \\ & & & I_{n-i-1} \end{bmatrix} = I_{i-1} \oplus \mathcal{G}_i^{(k)} \oplus I_{n-i-1}, \quad i = 1 : n-1,$$

where $c_i^{(k)} \in \mathbb{R}$, $|c_i^{(k)}|^2 + |s_i^{(k)}|^2 = 1$ and I_i denotes the $i \times i$ identity matrix. At the first step $G_1^{(k)}$ is chosen so that the entry in position $(2,1)$ of $G_1^{(k)}(A_k - \alpha_k I)$ is zero. In this way, only the entries in the first two lines of $G_1^{(k)} A_k$ differ from the corresponding entries of $A_k - \alpha_k I$. Moreover, for $j > i+1$, the former entries are given by $\frac{1}{y_n^{(k)}} \hat{x}_i^{(k)} y_j^{(k)} + \hat{u}_{i,1}^{(k)} \bar{v}_{j,1}^{(k)} + \hat{u}_{i,2}^{(k)} \bar{v}_{j,2}^{(k)}$, $i = 1, 2$, $j > i$, where

$$\begin{bmatrix} \hat{x}_1^{(k)} \\ \hat{x}_2^{(k)} \end{bmatrix} = \mathcal{G}_1^{(k)} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix}, \quad \begin{bmatrix} \hat{u}_{1,1}^{(k)} \\ \hat{u}_{2,1}^{(k)} \end{bmatrix} = \mathcal{G}_1^{(k)} \begin{bmatrix} u_{1,1}^{(k)} \\ u_{2,1}^{(k)} \end{bmatrix}, \quad \begin{bmatrix} \hat{u}_{1,2}^{(k)} \\ \hat{u}_{2,2}^{(k)} \end{bmatrix} = \mathcal{G}_1^{(k)} \begin{bmatrix} u_{1,2}^{(k)} \\ u_{2,2}^{(k)} \end{bmatrix},$$

while the remaining entries in the 2×2 leading principal submatrix of $G_1^{(k)}(A_k - \alpha_k I)$ are given by

$$\begin{bmatrix} d_1^{(k)} & g_1^{(k)} \\ 0 & d_2^{(k)} \end{bmatrix} = \mathcal{G}_1^{(k)} \begin{bmatrix} a_1^{(k)} - \alpha_k & \frac{1}{y_n^{(k)}} \hat{x}_1^{(k)} y_2^{(k)} + \hat{u}_{1,1}^{(k)} \bar{v}_{2,1}^{(k)} + \hat{u}_{1,2}^{(k)} \bar{v}_{2,2}^{(k)} \\ b_1^{(k)} & a_2^{(k)} - \alpha_k \end{bmatrix}.$$

The values of $d_1^{(k)}$, $\hat{x}_1^{(k)}$, $\hat{u}_{1,1}^{(k)}$, and $\hat{u}_{1,2}^{(k)}$ are not modified by the subsequent Givens rotations, while the values of $\hat{x}_2^{(k)}$, $\hat{u}_{2,1}^{(k)}$ and $\hat{u}_{2,2}^{(k)}$ and $d_2^{(k)}$ are modified only at the second step where $G_1^{(k)}(A_k - \alpha_k I)$ is pre-multiplied by $G_2^{(k)}$. At the i-th step, $G_i^{(k)}$ is chosen so that the entry in position $(i+1, i)$ of $G_{i-1}^{(k)} \cdots G_1^{(k)}(A_k - \alpha_k I)$ is zero.

At the end of the entire procedure the matrix R_k turns out to be represented in terms of its diagonal entries $d_1^{(k)}, \ldots, d_n^{(k)}$, its superdiagonal entries $g_1^{(k)}, \ldots, g_{n-1}^{(k)}$, by the vectors $\hat{\boldsymbol{x}}^{(k)}$, $\boldsymbol{y}^{(k)}$ and by the matrices $\widehat{U}_k = (\hat{u}_{i,j}^{(k)})$, $V_k = (v_{i,j}^{(k)})$ as

$$r_{i,j}^{(k)} = \begin{cases} d_i^{(k)} & \text{for } i = j, \\ g_i^{(k)} & \text{for } i = j-1, \\ \frac{1}{y_n^{(k)}} \hat{x}_i^{(k)} y_j^{(k)} + \hat{u}_{i,1}^{(k)} \bar{v}_{j,1}^{(k)} + \hat{u}_{i,2}^{(k)} \bar{v}_{j,2}^{(k)} & \text{for } i < j-1, \\ 0 & \text{for } i > j. \end{cases} \tag{2.9}$$

In this way the matrix R_k can be stored by using only $8n-1$ parameters. The entire process for computing the above representation of R_k is synthesized below.

ALGORITHM 1.
INPUT: The matrices U_k, V_k and the vectors $\boldsymbol{x}^{(k)}$, $\boldsymbol{y}^{(k)}$, and $\boldsymbol{b}^{(k)}$ which define the entries of A_k by means of Theorem 2.2. (For a certain use of the algorithm, which we will describe later on, we might give as input the vector $\boldsymbol{a}^{(k)}$ with the diagonal entries of A_k). The shift parameter α_k.
OUTPUT: The Givens parameters $s_i^{(k)}$ and $c_i^{(k)}$, $i = 1 : n-1$, together with the vectors $\boldsymbol{d}_i^{(k)}$, $\hat{\boldsymbol{x}}^{(k)}$, $\boldsymbol{y}^{(k)}$ and the matrices $\widehat{U}_k = (\hat{u}_{i,j}^{(k)})$, $V_k = (v_{i,j}^{(k)})$ which define the entries of R_k through (2.9), where $Q_k R_k = A_k - \alpha_k I$.
COMPUTATION:

1. Let $\boldsymbol{a}^{(k)} = (\frac{1}{y_n^{(k)}} x_i^{(k)} y_i^{(k)} + u_{i,1}^{(k)} \bar{v}_{i,1}^{(k)} + u_{i,2}^{(k)} \bar{v}_{i,2}^{(k)})_{i=1:n}$, $\hat{\boldsymbol{b}}^{(k)} = \boldsymbol{b}^{(k)}$, $\widehat{U}_k = U_k$, $\hat{\boldsymbol{x}}^{(k)} = \boldsymbol{x}^{(k)}$.
2. Set $\boldsymbol{a}^{(k)} = \boldsymbol{a}^{(k)} - \alpha_k (1, \ldots, 1)$.
3. For $i = 1 : n-1$ do
   (a) (Compute $\mathcal{G}_i$)
      i. $\gamma_i = 1/\sqrt{|a_i^{(k)}|^2 + |b_i^{(k)}|^2}$, $\nu_i = a_i^{(k)}/|a_i^{(k)}|$, ($\nu_i = 1$ if $a_i^{(k)} = 0$), $\theta_i = \gamma_i/\nu_i$,
      ii. $c_i^{(k)} = a_i^{(k)} \theta_i$, $s_i^{(k)} = b_i^{(k)} \theta_i$.
   (b) (Update R_k)
      i. $d_i^{(k)} = \gamma_i/\nu_i$, $t = \frac{1}{y_n^{(k)}} \hat{x}_i^{(k)} y_{i+1}^{(k)} + \hat{u}_{i,1}^{(k)} \bar{v}_{i+1,1}^{(k)} + \hat{u}_{i,2}^{(k)} \bar{v}_{i+1,2}^{(k)}$,
      ii. $g_i^{(k)} = c_i^{(k)} t + s_i^{(k)} a_{i+1}^{(k)}$,
      iii. $d_{i+1}^{(k)} = -\bar{s}_i^{(k)} t + c_i^{(k)} a_{i+1}^{(k)}$.
   (c) (Update $\hat{U}_k$)
      i. $t = c_i^{(k)} \hat{u}_{i,1}^{(k)} + s_i^{(k)} \hat{u}_{i+1,1}^{(k)}$, $\hat{u}_{i+1,1}^{(k)} = -\bar{s}_i^{(k)} \hat{u}_{i,1}^{(k)} + c_i^{(k)} \hat{u}_{i+1,1}^{(k)}$, $\hat{u}_{i,1}^{(k)} = t$,
      ii. $t = c_i^{(k)} \hat{u}_{i,2}^{(k)} + s_i^{(k)} \hat{u}_{i+1,2}^{(k)}$, $\hat{u}_{i+1,2}^{(k)} = -\bar{s}_i^{(k)} \hat{u}_{i,2}^{(k)} + c_i^{(k)} \hat{u}_{i+1,2}^{(k)}$, $\hat{u}_{i,2}^{(k)} = t$.
   (d) (Update $\hat{\boldsymbol{x}}^{(k)}$)
      i. $t = c_i^{(k)} \hat{x}_i^{(k)} + s_i^{(k)} \hat{x}_{i+1}^{(k)}$,
      ii. $\hat{x}_{i+1}^{(k)} = -\bar{s}_i^{(k)} \hat{x}_i^{(k)} + c_i^{(k)} \hat{x}_{i+1}^{(k)}$,
      iii. $\hat{x}_i^{(k)} = t$.
4. End do

2.3. Structure of Q_k. By construction, the matrix Q_k in the QR factorization $A_k = Q_k R_k$ is the product of $n-1$ Givens rotations

$$Q_k = {G_1^{(k)}}^H \cdots {G_{n-1}^{(k)}}^H.$$

We recall, from [10], [3], [16], the following lemma about the structure of Q_k, adjusted to the complex field.

LEMMA 2.3. *Let $c_i^{(k)} \in \mathbb{R}$ and $s_i^{(k)} \in \mathbb{C}$, $i = 1 : n-1$, be the parameters defining the Givens rotations $G_i^{(k)}$, $i = 1 : n-1$. Define*

$$
\begin{aligned}
D_{\boldsymbol{s}^{(k)}} &= \operatorname{diag}(1, -s_1^{(k)}, s_1^{(k)} s_2^{(k)}, \ldots, (-1)^{n-1} s_1^{(k)} s_2^{(k)} \cdots s_{n-1}^{(k)}),\\
\boldsymbol{p}^{(k)} &= D_{\boldsymbol{s}^{(k)}}^{-1} [1, c_1^{(k)}, c_2^{(k)}, \ldots, c_{n-1}^{(k)}]^T,\\
\boldsymbol{q}^{(k)} &= D_{\boldsymbol{s}^{(k)}} [c_1^{(k)}, c_2^{(k)}, c_3^{(k)}, \ldots, c_{n-1}^{(k)}, 1]^T.
\end{aligned} \tag{2.10}
$$

Then

$$
Q_k = {G_1^{(k)}}^H \cdots {G_{n-1}^{(k)}}^H = \begin{bmatrix}
q_1^{(k)} p_1^{(k)} & q_2^{(k)} p_1^{(k)} & \cdots & \cdots & q_n^{(k)} p_1^{(k)} \\
\bar{s}_1^{(k)} & q_2^{(k)} p_2^{(k)} & \cdots & & q_n^{(k)} p_2^{(k)} \\
 & \bar{s}_2^{(k)} & \ddots & & \vdots \\
 & & \ddots & q_{n-1}^{(k)} p_{n-1}^{(k)} & q_n^{(k)} p_{n-1}^{(k)} \\
O & & & \bar{s}_{n-1}^{(k)} & q_n^{(k)} p_n^{(k)}
\end{bmatrix}.
$$

Thus, the entries $q_{i,j}^{(k)}$ of Q_k are given by

$$
q_{i,j}^{(k)} = \begin{cases}
0 & \text{if } i > j+1 \\
\bar{s}_i^{(k)} & \text{if } i = j+1 \\
(-1)^{i+j} c_{i-1}^{(k)} c_j^{(k)} \prod_{\ell=i}^{j-1} s_\ell^{(k)} & \text{if } i \le j
\end{cases}
$$

where we assume $c_0^{(k)} = c_n^{(k)} = 1$.

3. Constructive issues and algorithms. The results of the above section allow us to design a fast algorithm for implementing the single QR step in $O(n)$ ops provided that A_0 is a companion matrix. In this section we discuss some related computational issues and describe in details the new algorithm.

For the sake of clarity, we first provide the description of an algorithm for the QR and RQ steps by ignoring numerical issues like overflow/underflow problems. Numerical drawbacks and the way to overcome them will be discussed later on in section 3.3

3.1. The QR step. The QR step can be carried out by using Algorithm 1. For real data, the arithmetic cost of this algorithm is $5n$ ops for computing the Givens rotations, $11n$ ops for computing R_k, $12n$ ops for updating $\widehat{U}_k$ and $6n$ ops for updating $\hat{\boldsymbol{x}}^{(k)}$. The overall cost is $34n$ ops.

3.2. The RQ step. In order to complete the QR iteration we have to compute the vectors $\boldsymbol{b}^{(k+1)}$, $\boldsymbol{x}^{(k+1)}$, $\boldsymbol{y}^{(k+1)}$ and the matrices U_{k+1}, V_{k+1}, which define the matrix $A_{k+1} = R_k Q_k + \alpha_k I$ by means of Theorem 2.2, given the matrices Q_k and R_k of the factorization $A_k - \alpha_k I = Q_k R_k$. By allowing redundancy of representation, we use as input variable also the vector $\boldsymbol{a}^{(k)}$ with the diagonal entries of A_k. More precisely we have to perform the following task:

GIVEN:

(i) the vectors $\boldsymbol{s}^{(k)}$ and $\boldsymbol{c}^{(k)}$ defining Q_k by means of the Givens rotations,

(ii) the last column $\boldsymbol{x}^{(k)}$ and the first row $\boldsymbol{y}^{(k)T}$ of A_k^{-H}, respectively, defining the upper triangular part of A_k^{-H} by means of (2.7),

(iii) the matrices U_k, V_k, such that (2.5) holds,

(iv) the vectors $\boldsymbol{d}^{(k)}$, $\boldsymbol{g}^{(k)}$, $\hat{\boldsymbol{x}}^{(k)}$ and the matrix $\widehat{U}_k$ defining (together with $\boldsymbol{y}^{(k)}$ and V_k) the matrix R_k through (2.9),

(v) the shift parameter α_k.

COMPUTE:

(j) the vectors $\boldsymbol{a}^{(k+1)}$, $\boldsymbol{b}^{(k+1)}$ defining the diagonal and subdiagonal entries of A_{k+1},

(jj) the vectors $\boldsymbol{x}^{(k+1)}$, $\boldsymbol{y}^{(k+1)}$ defining the upper triangular part of A_{k+1}^{-H},
(jjj) the matrices U_{k+1} and V_{k+1} such that (2.5) holds for A_{k+1}.

The computation of $\boldsymbol{a}^{(k+1)}$ and $\boldsymbol{b}^{(k+1)}$ is straightforward and follows from the relation $A_{k+1} = R_k Q_k + \alpha_k I$ in the light of the representations of Q_k and R_k, given in Lemma 2.3 and in (2.9), respectively:

$$\begin{aligned} a_i^{(k+1)} &= d_i^{(k)} c_{i-1}^{(k)} c_i^{(k)} + g_i \bar{s}_i^{(k)} + \alpha_k \quad && i = 1:n, \ \text{ where } c_0 = c_n = 1, \\ b_i^{(k+1)} &= d_{i+1}^{(k)} \bar{s}_i^{(k)}, && i = 1:n-1. \end{aligned}$$

Observe also that from the equations $U_{k+1} = Q_k U_k$ and $V_{k+1} = Q_k V_k$ we may easily compute U_{k+1} and V_{k+1} by just applying $n-1$ Givens rotations to the two columns of U_k and V_k at the cost of $24(n-1)$ ops. For the orthogonality of Givens rotations, this computation is robust and stable.

The computation of $\boldsymbol{x}^{(k+1)}$ and $\boldsymbol{y}^{(k+1)}$ deserves special attention. We propose different algorithms for this subtask.

Method 1 Rewrite (2.5) as

$$A_{k+1}^{-H} = Q_k^H A_k Q_k - U_{k+1} V_{k+1}^H$$

and from $\boldsymbol{x}^{(k+1)} = A_{k+1}^{-H} \boldsymbol{e}_n$, $\boldsymbol{y}^{(k+1)T} = \boldsymbol{e}_1^T A_{k+1}^{-H}$ obtain that

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= Q_k^H A_k Q_k \boldsymbol{e}_n - U_{k+1} V_{k+1}^H \boldsymbol{e}_n, \\ \boldsymbol{y}^{(k+1)T} &= \boldsymbol{e}_1 Q_k^H A_k Q_k - \boldsymbol{e}_1^T U_{k+1} V_{k+1}^H. \end{aligned}$$

The computational cost of the above expression is $O(n)$ ops. In fact, multiplication of a vector by Q_k is reduced to applying $n-1$ Givens rotations and the product of a vector by the matrix A_k still has a linear cost due to the semiseparability of A_k.

Method 2 From the equation $A_{k+1} = Q_k^H A_k Q_k$ obtain that

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= Q_k^H A_k^{-H} Q_k \boldsymbol{e}_n, \\ \boldsymbol{y}^{(k+1)T} &= \boldsymbol{e}_1^T Q_k^H A_k^{-H} Q_k. \end{aligned}$$

In this way the computation is reduced to computing products of Givens rotations with vectors and to solving systems having a semiseparable coefficient matrix in Hessenberg form. The latter computation can be accomplished by computing the QR factorization of the matrix A_k and by solving two triangular semiseparable systems. Both the computations clearly have linear cost. In fact, the QR factorization of A_k can be computed by applying Algorithm 1, while the solution of triangular semiseparable systems can be computed by means of Algorithm 2 in Section 4.

Method 3 From the equation $A_{k+1} = R_k A_k R_k^{-1}$ we deduce that

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= A_{k+1}^{-H} \boldsymbol{e}_n = R_k^{-H} A_k^{-H} R_k^H \boldsymbol{e}_n = \bar{r}_{n,n}^{(k)} R_k^{-H} \boldsymbol{x}^{(k)}, \\ \boldsymbol{y}^{(k+1)T} &= \boldsymbol{e}_1^T A_{k+1}^{-H} = \boldsymbol{e}_1^T R_k^{-H} A_k^{-H} R_k^H = \frac{1}{\bar{r}_{1,1}^{(k)}} \boldsymbol{y}^{(k)^T} R_k^H. \end{aligned}$$

The computation of $\boldsymbol{x}^{(k+1)}$ and $\boldsymbol{y}^{(k+1)}$ based on the latter formulas is reduced to solving a triangular semiseparable system and to multiplying a triangular semiseparable matrix with a vector, respectively. Both computations can be performed in $O(n)$ ops (compare with Algorithm 2 in Section 4).

Method 4 From the equation $A_{k+1}^{-H} = Q_k^H A_k^{-H} Q_k$ deduce that $(\boldsymbol{y}^{(k+1)})^T = \boldsymbol{h}^T Q_k$, where $\boldsymbol{h}^T$ is the first row of $Q_k^H A_k$. Now, since $Q_k^H = G_{n-1}^{(k)H} \cdots G_1^{(k)H}$, it follows that $\boldsymbol{h}$ is a linear combination of the first two rows of A_k^{-H}. More precisely

$$\boldsymbol{h} = c_1^{(k)}(\boldsymbol{e}_1^T A_k^{-H}) + s_1^{(k)}(\boldsymbol{e}_2^T A_k^{-H}),$$

that is, from the structure of A^{-H} we find that

$$\begin{aligned}\boldsymbol{e}_1^T A_k^{-H} &= \frac{x_1^{(k)}}{y_n^{(k)}}(y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}) = (y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}),\\ \boldsymbol{e}_2^T A_k^{-H} &= \frac{x_2^{(k)}}{y_n^{(k)}}(\xi^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}),\end{aligned}$$

where $\xi^{(k)} = \dfrac{y_n^{(k)}}{x_2^{(k)}}(A_k^{-H})_{2,1}$, i.e., from (2.5)

$$\xi^{(k)} = \frac{y_n^{(k)}}{x_2^{(k)}}(\bar{b}_1^{(k)} + u_{2,1}^{(k)}\bar{v}_{1,1}^{(k)} + u_{2,2}^{(k)}\bar{v}_{1,2}^{(k)}).$$

Once $\boldsymbol{y}^{(k+1)}$ has been computed, the vector $\boldsymbol{x}^{(k+1)}$ can be recovered from the nonlinear system

$$A_{k+1}^H \boldsymbol{x}^{(k+1)} = \boldsymbol{e}_n,$$

which can be solved by means of substitution in the following way. Rewrite the system as (where for the sake of simplicity we omit the index k and we consider $n = 5$)

$$\begin{aligned}&\left(\begin{bmatrix} \bar{a}_1 & \bar{b}_1 & & & \\ \bar{x}_1\bar{y}_2 & \bar{a}_2 & \bar{b}_2 & & \\ \bar{x}_1\bar{y}_3 & \bar{x}_2\bar{y}_3 & \bar{a}_3 & \bar{b}_3 & \\ \bar{x}_1\bar{y}_4 & \bar{x}_2\bar{y}_4 & \bar{x}_3\bar{y}_4 & \bar{a}_4 & \bar{b}_4 \\ \bar{x}_1\bar{y}_5 & \bar{x}_2\bar{y}_5 & \bar{x}_3\bar{y}_5 & \bar{x}_4\bar{y}_5 & \bar{a}_5 \end{bmatrix} + \right.\\ &\left.\sum_{i=1}^{2}\begin{bmatrix} 0 & & & & \\ \bar{u}_{1,i}v_{2,i} & 0 & & & \\ \bar{u}_{1,i}v_{3,i} & \bar{u}_{2,i}v_{3,i} & 0 & & \\ \bar{u}_{1,i}v_{4,i} & \bar{u}_{2,i}v_{4,i} & \bar{u}_{3,i}v_{4,i} & 0 & \\ \bar{u}_{1,i}v_{5,i} & \bar{u}_{2,i}v_{5,i} & \bar{u}_{3,i}v_{5,i} & \bar{u}_{4,i}v_{5,i} & 0 \end{bmatrix}\right)\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.\end{aligned} \tag{3.1}$$

Set $x_1 = y_n$ and compute x_2 from the first equation, compute x_3 from the second equation, and so forth until x_n is computed from the $(n-1)$-st equation. Once again the semiseparable structure of A_k enables us to compute $\boldsymbol{x}^{(k+1)}$ in $O(n)$ ops.

A similar substitution technique can be applied to compute $\boldsymbol{y}^{(k+1)}$ once the vector $\boldsymbol{x}^{(k+1)}$ has been computed.

Method 5 This method is valid if $\alpha_k = 0$, i.e., if no shift is performed in the QR iteration. In fact, in this case we have $A_{k+1} = R_k Q_k$ so that we obtain

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= R_k^{-H} Q_k \boldsymbol{e}_n\\ \boldsymbol{y}^{(k+1)} &= \frac{1}{\bar{r}_{1,1}^{(k)}} Q_k^H \boldsymbol{e}_1\end{aligned} \tag{3.2}$$

3.3. Numerical issues. In this section we are concerned with numerical issues of stability and robustness of the different algorithms for performing the QR iteration described in the previous section.

An important observation related to these issues concerns the growth of the vectors $\boldsymbol{u}_1^{(k)}, \boldsymbol{u}_2^{(k)}, \boldsymbol{v}_1^{(k)}, \boldsymbol{v}_2^{(k)}$.

REMARK 2. Since Q_k is unitary, it follows from the relations $\boldsymbol{u}_1^{(k+1)} = Q_k \boldsymbol{u}_1^{(k)}$, $\boldsymbol{u}_2^{(k+1)} = Q_k \boldsymbol{u}_2^{(k)}$, $\boldsymbol{v}_1^{(k+1)} = Q_k \boldsymbol{v}_1^{(k)}$, $\boldsymbol{v}_2^{(k+1)} = Q_k \boldsymbol{v}_2^{(k)}$ that

$$\begin{aligned}
\|\boldsymbol{u}_1^{(k)}\| &= \|\boldsymbol{u}_1^{(0)}\| = 1,\\
\|\boldsymbol{u}_2^{(k)}\| &= \|\boldsymbol{u}_2^{(0)}\| = (\sum_{i=1}^{n} |f_i|^2)^{\frac{1}{2}},\\
\|\boldsymbol{v}_1^{(k)}\| &= \|\boldsymbol{v}_1^{(0)}\| = \frac{1}{|f_1|}(\sum_{i=1}^{n} |f_i|^2)^{\frac{1}{2}},\\
\|\boldsymbol{v}_2^{(k)}\| &= \|\boldsymbol{v}_2^{(0)}\| = 1,
\end{aligned}$$

where $\|\cdot\|$ denotes the Euclidean norm. Therefore, in the computation of U_k and V_k, we do not have to expect numerical problems such as overflow or breakdowns.

A different situation holds for the vectors $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$; their computation is numerically much more delicate.

We may easily obtain uniform bounds for $\|\boldsymbol{x}^{(k)}\|$ and $\|\boldsymbol{y}^{(k)}\|$. In fact, since $\boldsymbol{x}^{(k)} = A_k^{-H}\boldsymbol{e}_1$, we have

$$\|\boldsymbol{x}^{(k)}\| \le \|A_k^{-H}\| = \|F^{-H}\| \le \|F^{-H}\|_F = \sqrt{n-1+\frac{1}{|f_1|^2}(1+\sum_{i=1}^{n-1}|f_i|^2)},$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Similarly we have

$$\|\boldsymbol{y}^{(k)}\| \le \|F^{-1}\| \le \sqrt{n-1+\frac{1}{|f_1|^2}(1+\sum_{i=1}^{n-1}|f_i|^2)}.$$

Moreover, from the relation $|a_{i,j}| \le \|A\|_F$, valid for the Frobenius norm $\|\cdot\|_F$, we deduce that

$$x_i^{(k)} y_j^{(k)}/y_n^{(k)} = x_i^{(k)} y_j^{(k)}/x_1^{(k)} \le \sqrt{n-1+\frac{1}{|f_1|^2}(1+\sum_{i=1}^{n-1}|f_i|^2)}, \quad i \le j.$$

Indeed these relations are useful to keep under control the rounding errors generated by a floating point computation of the algorithms of the previous section. However, unlike the case of the matrices U_k and V_k, the uniform boundedness of $\boldsymbol{x}^{(k)}$ and of $\boldsymbol{y}^{(k)}$ does not guarantee that numerical breakdown is avoided. The following example clarifies the situation.

Consider the matrix $A = (a_{i,j})$ such that $a_{i,j} = \epsilon^{j-i}$, $j \ge i$, where $0 < \epsilon < 1$, say $\epsilon = 1/10$. The first row $\boldsymbol{y}^T$ and the last column $\boldsymbol{x}$ of A are such that $\boldsymbol{y}^T = (1, \epsilon, \epsilon^2, \ldots, \epsilon^{n-1})$ and $\boldsymbol{x} = (\epsilon^{n-1}, \ldots, \epsilon^2, \epsilon, 1)^T$. Clearly it holds $a_{i,j} = x_i y_j / x_1$ for $j \ge i$. Moreover, the moduli of the components of $\boldsymbol{x}$, $\boldsymbol{y}$ and of $x_i y_j/x_1$ are bounded from above by 1. If $n > 309$, working with the IEEE floating point arithmetic, we encounter underflow in the computation of $x_1 = 10^{-n+1}$ so that the expression $x_i y_j / x_1$ would generate a breakdown due to a division by zero.

The situation illustrated by this example is not artificial at all as it could seem at first glance. In fact, during convergence of the QR iteration, the matrix A_k tends to an upper

ETNA
Kent State University
etna@mcs.kent.edu

triangular matrix, and, if the shift strategy is applied for approximating a simple eigenvalue, the entry in position $(n, n-1)$ of A_k converges superlinearly to zero. This implies that the last column of A_k^{-H} converges superlinearly to a multiple of $\boldsymbol{e}_n$, that is, $x_1^{(k)}$ converges to zero.

An attempt to keep handle this difficulty is to use a suitable representation of the vectors $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$. To this regard, it is important to point out that the representation of Q_k given in Lemma 2.3 shows that the last column as well as the first row of Q_k suffer from the same problem as $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$. In fact, from the equations

$$
\begin{aligned}
Q_k \boldsymbol{e}_n &= (-1)^{n-1}(s_1^{(k)} \cdots s_{n-1}^{(k)}) D_{\boldsymbol{s}^{(k)}}^{-1} \begin{bmatrix} 1 \\ c_1^{(k)} \\ \vdots \\ c_{n-1}^{(k)} \end{bmatrix}, \\
Q_k^H \boldsymbol{e}_1 &= \overline{D}_{\boldsymbol{s}^{(k)}} \begin{bmatrix} c_1^{(k)} \\ \vdots \\ c_{n-1}^{(k)} \\ 1 \end{bmatrix}, \\
D_{\boldsymbol{s}^{(k)}} &= \mathrm{diag}(1, -s_1^{(k)}, s_1^{(k)} s_2^{(k)}, \ldots, (-1)^{n-1} s_1^{(k)} \cdots s_{n-1}^{(k)}),
\end{aligned} \tag{3.3}
$$

which we deduce from Lemma 2.3, we find that computing the components of $\boldsymbol{p}^{(k)}$ and $\boldsymbol{q}^{(k)}$ may generate overflow and underflow even for moderate values of n due to the property $|s_i^{(k)}|, |c_i^{(k)}| \le 1$, $i = 1 : n-1$. In fact, the matrix Q_k is more conveniently represented by means of the Givens parameters $s_i^{(k)}, c_i^{(k)}$, $i = 1 : n-1$, rather than by means of the vectors $\boldsymbol{q}^{(k)}$ and $\boldsymbol{p}^{(k)}$. Moreover, this latter representation of Q_k allows one to perform numerical computations with Q_k, such as the evaluation of matrix-vector products, without incurring in any numerical breakdown.

Another useful remark is that the representation of $\boldsymbol{y}^{(k+1)}$ given by (3.2), in the case where no shift strategy is applied, is given in terms of the first row of Q_k, i.e. (compare with (3.3)),

$$
\boldsymbol{y}^{(k+1)} = \frac{1}{\bar{r}_{1,1}^{(k)}} Q_k^H \boldsymbol{e}_1 = \frac{1}{\bar{r}_{1,1}^{(k)}} \overline{D}_{\boldsymbol{s}^{(k)}} \begin{bmatrix} c_1^{(k)} \\ \vdots \\ c_{n-1}^{(k)} \\ 1 \end{bmatrix}.
$$

These considerations suggest to represent the vectors $\boldsymbol{x}^{(k+1)}$ and $\boldsymbol{y}^{(k+1)}$ by means of the Givens parameters $s_i^{(k)}$ and $c_i^{(k)}$ and by auxiliary vectors $\boldsymbol{w}^{(k+1)}$ and $\boldsymbol{z}^{(k+1)}$, such that

$$
\begin{aligned}
\boldsymbol{x}^{(k+1)} &= (-1)^{n-1}(s_1^{(k)} \cdots s_{n-1}^{(k)}) D_{\boldsymbol{s}^{(k)}}^{-1} \boldsymbol{z}^{(k+1)}, \\
\boldsymbol{y}^{(k+1)} &= \overline{D}_{\boldsymbol{s}^{(k)}} \boldsymbol{w}^{(k+1)}.
\end{aligned} \tag{3.4}
$$

In this way, if no shift strategy is applied, then from (3.2) and (3.4) we find that

$$
\begin{aligned}
\boldsymbol{w}^{(k+1)} &= \frac{1}{\bar{r}_{1,1}^{(k)}} (c_1^{(k)}, \ldots, c_{n-1}^{(k)}, 1)^T, \\
\boldsymbol{z}^{(k+1)} &= D_{\boldsymbol{s}^{(k)}} R_k^{-H} D_{\boldsymbol{s}^{(k)}}^{-1} (1, c_1^{(k)}, \ldots, c_{n-1}^{(k)})^T.
\end{aligned} \tag{3.5}
$$

Thus, $\boldsymbol{w}^{(k+1)}$ is readily available from the Givens rotations and from $r_{1,1}^{(k)}$ without numerical problems, while $\boldsymbol{z}^{(k+1)}$ is obtained by solving a triangular semiseparable system. For the computational issues related to the latter problem we refer the reader to section 4 and to

Algorithm 2. With this representation we expect more robustness even in the computation of $\boldsymbol{z}^{(k+1)}$.

A different way to represent $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$ is derived from equation (2.8), where the upper triangular part of A_k^{-H} is given in terms of the lower diagonal entries of A_k by means of two auxiliary vectors. Let us rewrite this representation below, where this time $\boldsymbol{w}^{(k+1)}$ and $\boldsymbol{z}^{(k+1)}$ denote the vectors with components $((-1)^{i-1}\hat{d}_i^{(k)})$ and $((-1)^{n-i}\tilde{d}_{n-i}^{(k)})$, respectively (compare with Lemma 2.1),

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= (b_1^{(k+1)} \cdots b_{n-1}^{(k+1)}) D_{\boldsymbol{b}^{(k+1)}}^{-1} \boldsymbol{z}^{(k+1)}, \\ \boldsymbol{y}^{(k+1)} &= D_{\boldsymbol{b}^{(k+1)}} \boldsymbol{w}^{(k+1)}. \end{aligned} \tag{3.6}$$

4. Implementation. We are ready to describe implementations of the QR and RQ steps which aim to remove the breakdown situations encountered in the original computations of section 3. They are based on the representation of $\boldsymbol{x}^{(k)}$ and of $\boldsymbol{y}^{(k)}$ provided in (3.4). Similar implementations can be based on (3.6).

Let us consider the QR step as described in Algorithm 1 and replace the vectors $\boldsymbol{x}^{(k)}$, $\hat{\boldsymbol{x}}^{(k)}$, $\boldsymbol{y}^{(k)}$ by $\boldsymbol{z}^{(k)}$, $\hat{\boldsymbol{z}}^{(k)}$ and $\boldsymbol{w}^{(k)}$, respectively, such that

$$\begin{aligned} \boldsymbol{x}^{(k)} &= (-1)^{n-1}(s_1^{(k-1)} \cdots s_{n-1}^{(k-1)}) D_{\boldsymbol{s}^{(k-1)}}^{-1} \boldsymbol{z}^{(k)}, \\ \hat{\boldsymbol{x}}^{(k)} &= (-1)^{n-1}(s_1^{(k-1)} \cdots s_{n-1}^{(k-1)}) D_{\boldsymbol{s}^{(k-1)}}^{-1} \hat{\boldsymbol{z}}^{(k)}, \\ \boldsymbol{y}^{(k)} &= D_{\boldsymbol{s}^{(k-1)}} \boldsymbol{w}^{(k)}. \end{aligned}$$

These replacements modify the computational scheme of Algorithm 1 only in the stages 1, (b) and (d), which now become

1. $\boldsymbol{a}^{(k)} = (\frac{1}{w_n^{(k)}} z_i^{(k)} w_i^{(k)} + u_{i,1}^{(k)} \bar{v}_{i,1}^{(k)} + u_{i,2}^{(k)} \bar{v}_{i,2}^{(k)})_{i,1,n}$, $\hat{\boldsymbol{b}}^{(k)} = \boldsymbol{b}^{(k)}$, $\widehat{U}_k = U_k$, $\hat{\boldsymbol{z}}^{(k)} = \boldsymbol{z}^{(k)}$.

(b) (Update R_k)

(i) $d_i^{(k)} = \gamma_i^{(k)}$, $t = \frac{1}{w_n^{(k)}} \hat{z}_i^{(k)} w_{i+1}^{(k)} s_i^{(k-1)} + \hat{u}_{i,1}^{(k)} \bar{v}_{i+1,1}^{(k)} + \hat{u}_{i,2}^{(k)} \bar{v}_{i+1,2}^{(k)}$,

(ii) $g_i^{(k)} = c_i^{(k)} t + s_i^{(k)} a_{i+1}^{(k)}$,

(iii) $d_{i+1}^{(k)} = -s_i^{(k)} t + c_i^{(k)} a_{i+1}^{(k)}$.

(d) (Update $\hat{\boldsymbol{z}}^{(k)}$)

(i) $t = c_i^{(k)} \hat{z}_i^{(k)} - \frac{s_i^{(k)}}{s_i^{(k-1)}} \hat{z}_{i+1}^{(k)}$,

(ii) $\hat{z}_{i+1}^{(k)} = s_i^{(k)} s_i^{(k-1)} \hat{z}_i^{(k)} + c_i^{(k)} \hat{z}_{i+1}^{(k)}$,

(iii) $\hat{z}_i^{(k)} = t$.

The computational cost of this modification is slightly higher than the one of Algorithm 1 but still remains linear in n. Possible numerical troubles in this modification might be encountered in the computation of the ratio $\frac{s_i^{(k)}}{s_i^{(k-1)}}$. It is worth observing that if $s_i^{(k-1)}$ is numerically zero, then convergence has taken place and the algorithm stops (or continues after deflating the approximated eigenvalue). Moreover, in the case of linear convergence it holds that $\lim_{k\to\infty} \frac{s_i^{(k)}}{s_i^{(k-1)}}$ is a nonzero constant, whereas in the case of superlinear convergence it holds $\lim_{k\to\infty} \frac{s_i^{(k)}}{s_i^{(k-1)}} = 0$.

Concerning the implementation of the RQ step, consider the most simple case where no shift strategy is applied so that we may rely on method 5. Indeed, from (3.5) we may recover $\boldsymbol{w}^{(k+1)}$ at no additional cost, whereas for computing $\boldsymbol{z}^{(k+1)}$ we have to solve the

ETNA
Kent State University
etna@mcs.kent.edu

linear system

$$D_{\boldsymbol{s}^{(k)}} R_k^H D_{\boldsymbol{s}^{(k)}}^{-1} \boldsymbol{z}^{(k+1)} = \boldsymbol{b}, \qquad (4.1)$$
$$\boldsymbol{b} = (1, c_1^{(k)}, \ldots, c_{n-1}^{(k)})^T,$$

where $R_k = (r_{i,j}^{(k)})$ is defined by the vectors $\boldsymbol{d}^{(k)}$, $\boldsymbol{g}^{(k)}$, $\hat{\boldsymbol{x}}^{(k)}$, $\boldsymbol{y}^{(k)}$ and by the matrices $\widehat{U}_k$, V_k as in (2.9).

The following algorithm solves the triangular semiseparable system (4.1). The algorithm relies on the structure of the matrix $D_{\boldsymbol{s}^{(k)}} R_k^H D_{\boldsymbol{s}^{(k)}}^{-1}$ which is obtained from (2.9) and whose entries are reported below

$$\begin{cases} \bar{d}_i^{(k)} & \text{if } i = j, \\ -s_i^{(k)} \bar{g}_i^{(k)} & \text{if } i = j+1, \\ (s_j^{(k)} \cdots s_{i-1}^{(k)}) \Big(\dfrac{1}{\bar{w}_n^{(k)}} \bar{w}_i^{(k)} \overline{\hat{z}}_j^{(k)} (\bar{s}_j^{(k-1)} \cdots \bar{s}_{i-1}^{(k-1)}) + & \\ \qquad (-1)^{i+j} (v_{i,1}^{(k)} \overline{\hat{u}}_{j,1}^{(k)} + v_{i,2}^{(k)} \overline{\hat{u}}_{j,2}^{(k)}) \Big) & \text{if } i > j+1, \\ 0 & \text{if } i < j. \end{cases}$$

ALGORITHM 2.
INPUT: The vectors $\boldsymbol{d}^{(k)}$, $\boldsymbol{g}^{(k)}$, $\hat{\boldsymbol{z}}^{(k)}$, $\boldsymbol{w}^{(k)}$, and the matrices $\widehat{U}_k = (\hat{u}_{i,j}^{(k)})$, $V_k = (v_{i,j}^{(k)})$, together with the Givens parameters $s_i^{(k-1)}$ and $c_i^{(k-1)}$ which define the entries of R_k by means of (2.9). The right-hand side vector $\boldsymbol{b}$ and the components of $\boldsymbol{s}^{(k)}$.
OUTPUT: The solution $\boldsymbol{z}^{(k+1)}$ of the system (4.1).
COMPUTATION:
Set $z_1^{(k+1)} = b_1/\bar{d}_1^{(k)}$, $z_2^{(k+1)} = (b_2 + \bar{g}_1^{(k)} s_1^{(k)} z_1^{(k+1)})/\bar{d}_2^{(k)}$, $\gamma_{2,1} = \gamma_{2,2} = \phi_2 = 0$.
For $i = 3 : n$ do
1. $\gamma_{i,j} = s_{i-1}^{(k)}(-\gamma_{i-1,j} + \overline{\hat{u}}_{i-2,j}^{(k)} s_{i-2}^{(k)} z_{i-2}^{(k+1)})$, $j = 1, 2$,
2. $\phi_i = s_{i-1}^{(k)} \bar{s}_{i-1}^{(k-1)} (\phi_{i-1} - \overline{\hat{z}}_{i-2}^{(k)} \bar{s}_{i-2}^{(k-1)} s_{i-2}^{(k)} z_{i-2}^{(k+1)})$,
3. $z_i^{(k+1)} = (b_i + \bar{g}_{i-1}^{(k)} s_{i-1}^{(k)} z_{i-1}^{(k+1)} - v_{i,1}^{(k)} \gamma_{i,1} - v_{i,2}^{(k)} \gamma_{i,2} + \bar{w}_i^{(k)} \phi_i / w_n^{(k)})/\bar{d}_i^{(k)}$,

End do

The more general case when the QR iteration is applied with a shift can be treated similarly. For instance, applying method 3 with the replacement (3.4) yields

$$\boldsymbol{w}^{(k+1)} = \frac{\sigma_n}{\bar{r}_{1,1}^{(k)}} \overline{D}_{\boldsymbol{s}^{(k)}}^{-1} \overline{R}_k \overline{D}_{\boldsymbol{s}^{(k)}} \mathrm{Diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_n^{-1}) \boldsymbol{w}^{(k)},$$
$$\boldsymbol{z}^{(k+1)} = \bar{r}_{n,n}^{(k)} D_{\boldsymbol{s}^{(k)}} R_k^{-H} D_{\boldsymbol{s}^{(k)}}^{-1} \mathrm{Diag}(\bar{\sigma}_1, \bar{\sigma}_2, \ldots, \bar{\sigma}_n) \boldsymbol{z}^{(k)},$$
$$\sigma_1 = 1, \quad \sigma_i = \prod_{j=1}^{i-1} \theta_j, \quad i = 2:n, \quad \theta_i = s_i^{(k-1)}/s_i^{(k)}, \quad i = 1:n-1.$$

The computation of $\boldsymbol{z}^{(k+1)}$ involves the solution of a triangular semiseparable system of the same kind as (4.1) where $\boldsymbol{b} = \frac{1}{\bar{r}_{n,n}^{(k)}} \mathrm{Diag}(\bar{\sigma}_1, \ldots, \bar{\sigma}_n) \boldsymbol{z}^{(k)}$, so that we may apply Algorithm 2. The computation of $\boldsymbol{w}^{(k+1)}$ requires the multiplication of a triangular semiseparable matrix and a vector. Also this task can be carried out in $O(n)$ ops.

5. Numerical experiments. We have implemented the QR iteration for computing a given number m of eigenvalues of the matrix F based on the method 3 and 5 (if shift is not performed) and relying on the representation (3.4). For the sake of simplicity the implementation and the numerical experiments have been performed in the real field. The program has been written in Fortran 90 and tested under the Linux system on a computer with an Athlon 1600 cpu.

test 1: Wilkinson's polynomial $p(x) = \prod_{i=1}^{n}(x-i)$ for $n = 10, 20$.
test 2: Reversed Wilkinson's polynomial. $p(x) = \prod_{i=1}^{n}(x-1/i)$ for $n = 10, 20$.
test 3: Polynomial with well separated roots, $p(x) = \prod_{i=1}^{n}(x-1/2^i)$, $n = 40$.
test 4: $p(x) = (x^{n-m}+1)\prod_{i=2}^{m+1}(x-1/i)$ for $m = 20$ and $n \le 10^6$.
test 5: $p(x) = (x^{n-m}+1)\prod_{i=1}^{m}(x-1/2^i)$ for $m = 40$ and $n \le 10^6$.

The shift technique is applied if the difference between two subsequent approximations $a_{n,n}^{(k+1)} - a_{n,n}^{(k)}$ has modulus less than 1/10. As stopping condition for the QR iteration we chose $|a_{n,n-1}^{(k)}| < \epsilon |a_{n,n}^{(k)}|$ with $\epsilon = 10^{-11}$. Once an eigenvalue λ has been approximated, we deflated the matrix A_k by removing the last row and the last column and continued to apply the algorithm to the submatrix obtained in this way.

In the case of the Wilkinson polynomial of degree $n = 20$ the algorithm failed to converge if starting with no shift, i.e., with $\alpha_0 = 0$. Starting with $\alpha_0 = 22$, i.e., by approximating the eigenvalues in decreasing order, the algorithm provided the approximations shown in Table 2. In Tables 1,2,3, we report the values i, λ_i and the number of required iterations. In Tables 4 and 5, we report the value of the cpu time, in seconds, required for computing the first m zeros of the polynomial of degree n where n takes values up to 10^6. We do not report the approximation errors since they seem to be almost independent of the degree n.

As we can see from the tables, our algorithm has a cost which grows linearly with n, and allows us to handle polynomials of very large degree with no problems of memory storage. In certain cases the approximations are reasonably precise, in other cases (see the Wilkinson polynomial of degree 20 with zeros approximated in increasing order) the algorithm fails to converge. Other cases of breakdown due to overflow/underflow have been encountered. This means that the algorithm, even where implemented with the representation based on (3.4), is not robust and needs more investigation. One reason for this weakness is the fact that the representation of A_k somehow involves the expression of the inverse A_k^{-1}, and consequently requires performing divisions. Algorithms for the QR iteration which perform unitary transformations and do not require divisions (by numbers of small modulus) have shown to be stable and robust. More precisely, we refer the reader to the paper [2] where the QR iteration is specifically designed for computing eigenvalues of a special class of matrices including arrowhead matrices and matrices of the kind diagonal+rank 1.

A way to overcome this difficulty is to apply the QR iteration directly to the matrix $A_0 = F + F^{-1}$ which generates the sequence A_k such that $A_k = A_k^H + U^{(k)}V^{(k)H}$ for suitable $n \times 2$ matrices $U^{(k)}$ and $V^{(k)}$. Even in this case the weak semiseparable structure of A_0 is maintained; moreover, no inversion of A_k is needed anymore in the representation formulas, except that of $A_0 = F$. When the eigenvalues $\mu_1, \ldots, \mu_n$ have been computed, we may find the eigenvalues λ_i of F in the set of the solutions of the equations $\lambda + \lambda^{-1} = \mu_i$, $i = 1, \ldots, n$. Alternatively, the representation of F as a unitary Hessenberg plus a rank one matrix could be another way for designing inversion-free algorithms for performing the QR step. These will be the subjects of our next research.

Acknowledgments. We thank an anonymous referee for valuable suggestions that improved the presentation.

ETNA
Kent State University
etna@mcs.kent.edu

i	λ_i	iter
1	**1.0000000000**81050	17
2	**1.99999999**3547419	6
3	**3.00000004**3477431	6
4	**3.9999999**80710313	5
5	**4.999999**447874695	5
6	**6.00000**1980752747	5
7	**6.99999**6815436769	5
8	**8.00000**2744205705	5
9	**8.99999**8765156054	4
10	**10.0000000**22875783	1

$\alpha_0 = 0$

i	λ_i	iter
1	**10.000000000**04859	10
2	**8.999999999**765816	6
3	**8.000000000**480380	5
4	**6.999999999**455802	5
5	**6.000000000**369495	5
6	**4.9999999998**46834	5
7	**4.0000000000**37894	5
8	**2.99999999999**4979	5
9	**2.000000000000**302	5
10	**0.999999999999**9289	1

$\alpha_0 = 12$

TABLE 5.1
Wilkinson's polynomial $n = 10$.

i	λ_i	iter	i	λ_i	iter
1	**20.0000**4391793787	10	11	**10.00**101029035203	5
2	**18.999**58361338783	6	12	**8.999**690499995966	5
3	**18.00**179081927481	5	13	**8.0000**65781753833	5
4	**16.99**529972600241	5	14	**6.99999**2485609691	5
5	**16.00**827840794409	5	15	**5.999999**762404186	5
6	**14.9**8930172974257	5	16	**5.000000**226762443	5
7	**14.0**1045173655527	5	17	**3.9999999**66695113	5
8	**12.99**200882355489	5	18	**3.00000000**2193996	5
9	**12.00**498158865593	5	19	**1.9999999999**23354	5
10	**10.99**750062124958	5	20	**1.00000000000**3987	1

TABLE 5.2
Wilkinson's polynomial, $n = 20$, $\alpha_0 = 22$.

REFERENCES

[1] G. S. AMMAR AND W. B. GRAGG, *Schur flows for orthogonal Hessenberg matrices*, in Hamiltonian and gradient flows, algorithms and control, Fields Inst. Commun., 3, Amer. Math. Soc., Providence, RI, 1994, pp. 27–34.

[2] D. A. BINI, L. GEMIGNANI, AND V. Y. PAN, *QR-like algorithms for generalized semiseparable matrices*, Technical Report 1470, Dipartimento di Matematica, Università di Pisa, 2003.

[3] D.A. BINI, L. GEMIGNANI, AND F. TISSEUR, *The Ehrlich-Aberth method for the nonsymmetric tridiagonal eigenvalue problem*, Technical Report 428, Manchester Centre for Computational Mathematics, University of Manchester, 2003.

[4] D. CALVETTI, S. KIM, AND L. REICHEL, *The restarted QR-algorithm for eigenvalue computation of structured matrices*, J. Comput. Appl. Math., 149 (2002), no. 2, pp. 415–422.

[5] S. CHANDRASEKARAN AND M. GU, *A fast and stable solver for recursively semi-separable systems of linear equations*, in Structured matrices in mathematics, computer science, and engineering, II, Boulder, CO, 1999, Contemp. Math., 281, Amer. Math. Soc., Providence, RI, 2001, pp. 39–53.

[6] Y. EIDELMAN AND I. GOHBERG, *Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices*, Comput. Math. Appl., 33 (1997), pp. 69–79.

[7] Y. EIDELMAN AND I. GOHBERG, *On a new class of structured matrices*, Integral Equations Operator Theory, 34 (1999), pp. 293–324.

[8] D. FASINO, *Personal communication*.

[9] D. FASINO, N. MASTRONARDI, AND M. VAN BAREL, *Fast and stable algorithms for reducing diagonal plus semiseparable matrices to tridiagonal and bidiagonal form*, in Fast Algorithms for Structured Matrices: Theory and Applications, Vadim Olshevsky, ed., Contemp. Math., 323, Amer. Math. Soc., Providence, RI, 2003, pp. 105–118.

[10] P. E. GILL, G. H. GOLUB, W. MURRAY, AND M. A. SAUNDERS, *Methods for modifying matrix factorizations*, Math. Comp., 28 (1974), pp. 505–535.

[11] I. GOHBERG, T. KAILATH, AND I. KOLTRACHT, *Linear complexity algorithms for semiseparable matrices*, Integral Equations Operator Theory, 8 (1985), no. 6, pp. 780–804.

[12] I. KOLTRACHT, *Linear complexity algorithm for semiseparable matrices*, Integral Equations Operator

i	λ_i	iter	i	λ_i	iter
1	**9.09**0699628083369E-13	6	21	**9.5367431640625**61E-07	3
2	**1.819**421487856309E-12	4	22	**1.90734863281249**4E-06	3
3	**3.637**691425913883E-12	3	23	**3.8146972656249**70E-06	3
4	**7.2759**60482738636E-12	3	24	**7.6293945312500**17E-06	3
5	**1.4552**01315732692E-11	3	25	**1.52587890625000**5E-05	3
6	**2.9103**90935284155E-11	3	26	**3.05175781249999**0E-05	3
7	**5.82077**0837885015E-11	3	27	**6.10351562500000**1E-05	3
8	**1.164153**475948272E-10	3	28	**1.2207031249999**84E-04	3
9	**2.328306**570434075E-10	3	29	**2.4414062500000**34E-04	3
10	**4.6566129**41269948E-10	3	30	**4.882812499999**877E-04	3
11	**9.3132257**80556513E-10	3	31	**9.765625000000**256E-04	3
12	**1.86264515**0958509E-09	3	32	**1.9531249999999**59E-03	3
13	**3.72529029**9327524E-09	3	33	**3.9062500000000**27E-03	3
14	**7.45058059**7357062E-09	3	34	**7.8124999999999**41E-03	3
15	**1.490116119**406440E-08	3	35	**1.56249999999999**7E-02	3
16	**2.9802322387**80377E-08	3	36	**3.1250000000000**10E-02	3
17	**5.96046447754**4437E-08	3	37	**6.2499999999999**58E-02	3
18	**1.19209289550**8090E-07	3	38	**0.124999999999999**7	3
19	**2.384185791015**752E-07	3	39	**0.250000000000000**2	3
20	**4.7683715820312**78E-07	3	40	**0.499999999999999**6	1

TABLE 5.3
Polynomial $\prod_{i=1}^{40}(x-1/2^i)$, $\alpha_0 = 0$.

n	cpu	n	cpu
100	0.01	16000	1.59
200	0.02	32000	3.12
400	0.04	64000	6.47
800	0.07	125000	12.63
1600	0.11	250000	25.1
3200	0.26	500000	52.3
6400	0.51	1000000	108.8

$p(x) = (x^{n-m}+1)\prod_{i=2}^{m+1}(x-1/i)$,
$m = 20$

n	cpu	n	cpu
100	0.00	16000	1.56
200	0.01	32000	3.38
400	0.02	64000	6.78
800	0.04	125000	13.02
1600	0.08	250000	28.02
3200	0.16	500000	52.51
6400	0.36	1000000	116.02

$p(x) = (x^{n-m}+1)\prod_{i=1}^{m+1}(x-1/2^i)$,
$m = 40$

TABLE 5.4
Computing m *zeros of* $p(x)$*: cpu time in seconds.*

Theory, 29 (1997), no. 3, pp. 313–319.

[13] P. FAVATI P. RÓZSA, R. BEVILACQUA AND F. ROMANI, *On the inverse of block tridiagonal matrices with applications to the inverses of band matrices and block band matrices*, Oper. Theory Adv. Appl., 40 (1989), pp. 447–469.

[14] P. RÓZSA, R. BEVILACQUA, F. ROMANI, AND P. FAVATI, *On band matrices and their inverses*, in Proceedings of the First Conference of the International Linear Algebra Society, Provo, UT, 1989,, Linear Algebra Appl., 150 (1991), pp. 287–295.

[15] E. E. TYRTYSHNIKOV, *Mosaic ranks for weakly semiseparable matrices*, in Large-scale scientific computations of engineering and environmental problems, II, Sozopol, 1999, Notes Numer. Fluid Mech., 73, Vieweg, Braunschweig, 2000, pp. 36–41.

[16] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *An implicit qr algorithm for semiseparable matrices, to compute the eigendecomposition of symmetric matrices*, Technical Report TW367, Department of Computer Science, Katholieke Universiteit, Leuven, August 2003.

[17] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *An orthogonal similarity reduction of a matrix to semiseparable form*, Technical Report TW355, Department of Computer Science, Katholieke Universiteit, Leuven, February 2003.

[18] T. WANG AND W. B. GRAGG, *Convergence of the shifted* QR *algorithm for unitary Hessenberg matrices*, Math. Comp., 71 (2002), no. 240, pp. 1473–1496.

[19] H. XU, *The relation between the QR and LR algorithms*, SIAM J. Matrix Anal. Appl., 19 (1998), no. 2, pp. 551–555.