Fast QR Eigenvalue Algorithms for Hessenberg Matrices Which Are Rank-One Perturbations of Unitary Matrices

Article *in* SIAM Journal on Matrix Analysis and Applications · January 2007

DOI: 10.1137/050627563 · Source: DBLP

Dario A. Bini
Università di Pisa
216 PUBLICATIONS **4,055** CITATIONS
SEE PROFILE

Yuli Eidelman
Tel Aviv University
95 PUBLICATIONS **975** CITATIONS
SEE PROFILE

Luca Gemignani
Università di Pisa
102 PUBLICATIONS **1,080** CITATIONS
SEE PROFILE

FAST QR EIGENVALUE ALGORITHMS FOR HESSENBERG MATRICES WHICH ARE RANK-ONE PERTURBATIONS OF UNITARY MATRICES

D. A. BINI[†¶], Y. EIDELMAN[‡], L. GEMIGNANI[†¶], AND I. GOHBERG[‡]

Abstract. Let $\mathcal{H}_n \subset \mathbb{C}^{n\times n}$ be the class of $n \times n$ Hessenberg matrices which are rank-one modifications of a unitary matrix, that is, $A \in \mathcal{H}_n$ then $A = H + \boldsymbol{u}\boldsymbol{w}^H$, where $A \in \mathbb{C}^{n\times n}$ is in Hessenberg form, H is unitary and $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{C}^n$. We consider the problem of computing the eigenvalues of $A \in \mathcal{H}_n$ in a fast way by means of the shifted QR eigenvalue algorithm. The class $\mathcal{H}_n$ includes at least three well known sub-classes: unitary Hessenberg matrices, companion (Frobenius) matrices and fellow matrices, that is, Hessenberg matrices which are rank-one perturbations of unitary Hessenberg matrices. The paper describes some novel, fast adaptations of the QR eigenvalue algorithm for a matrix $A \in \mathcal{H}_n$ which can be performed with $O(n)$ flops per iteration and $O(n)$ memory space only. Numerical experiments confirm the effectiveness and the robustness of these methods.

Key words. Hessenberg matrices, rank-one perturbations, unitary matrices, companion matrices, quasiseparable matrices, QR iteration, eigenvalue computation, complexity

AMS subject classifications. 65F15, 65H17

1. Introduction.

The subject of this paper is the efficient computation of the eigenvalues of certain Hessenberg matrices $A \in \mathbb{C}^{n\times n}$ of the form

$$A = H + \boldsymbol{u}\boldsymbol{w}^H, \tag{1.1}$$

where $H \in \mathbb{C}^{n\times n}$ is unitary and $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{C}^n$. Let $\mathcal{H}_n$ denote the class of such $n \times n$ matrices. If $A \in \mathcal{H}_n$ is invertible, then from the Sherman-Morrison formula one has

$$A^{-1} = H^H - \sigma H^H \boldsymbol{u}\boldsymbol{w}^H H^H, \quad \sigma = (1 + \boldsymbol{w}^H H^H \boldsymbol{u})^{-1} \in \mathbb{C},$$

which can be combined with (1.1) to give

$$A - A^{-H} = \boldsymbol{u}\boldsymbol{w}^H + \bar{\sigma} H \boldsymbol{w}\boldsymbol{u}^H H. \tag{1.2}$$

The shifted QR algorithm

$$\begin{aligned} A_0 &= A \\ A_k - \alpha_k I_n &= Q_k R_k \\ A_{k+1} &:= R_k Q_k + \alpha_k I_n, \end{aligned} \tag{1.3}$$

has been known as a standard method for computing the eigenvalues of a dense matrix [13, 17, 23]. A careful exploitation of (1.2) says that all the matrices A_k, $k = 0, 1, \ldots$, generated by the shifted QR algorithm (1.3) applied for the computation of the eigenvalues of the matrix $A = A_0 \in \mathcal{H}_n$ can virtually be represented by a linear number of parameters. Therefore, our main goal is to find out fast and numerically robust adaptations of the customary QR method (1.3) for an input matrix $A_0 \in \mathcal{H}_n$ which require linear time per iteration and linear memory space.

[†]Dipartimento di Matematica, Università di Pisa, Largo Bruno Pontecorvo 5, 56127 Pisa, Italy, {bini, gemignan}@dm.unipi.it

[‡]School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Ramat-Aviv, 69978, Israel, {eideyu, gohberg}@post.tau.ac.il

[¶]This work was partially supported by MIUR, grant number 2002014121, and by GNCS-INDAM.

1.1. The Class $\mathcal{H}_n$. The class $\mathcal{H}_n$ includes at least three well-known subclasses: unitary Hessenberg matrices, companion (Frobenius) matrices and *fellow* matrices. Direct and inverse unitary Hessenberg eigenvalue problems arise in signal processing [16] and in least squares approximation by trigonometric polynomials [3, 18]. A fast QR-algorithm for eigenvalue computation of unitary Hessenberg matrices was first presented in [14]. The algorithm relies upon the Schur representation of a unitary Hessenberg matrix as product of (modified) Givens rotations [12]. Suitable variants of the algorithm of [14] are developed in [15, 22, 21], whereas the potential stability problems encountered with these algorithms are analyzed in [19]. The companion (Frobenius) matrix associated with the $n-$degree polynomial $f(z) = \sum_{i=0}^{n} f_i z^i$, $f_n \neq 0$, is defined by $F = C + \boldsymbol{f}\boldsymbol{e}_n^T$, where C is the generator of the *circulant* matrix algebra, $\boldsymbol{e}_j$ stands for the j-th column of the $n \times n$ identity matrix I_n and $\boldsymbol{f}^T = [-f_0/f_n - 1, -f_1/f_n, \ldots, -f_{n-1}/f_n]$. Matrix methods based on the QR algorithm applied to a companion matrix are customary for polynomial root-finding: in fact, they are proposed to the users of MATLAB*. Matrices of the more general form $F = H + \boldsymbol{u}\boldsymbol{e}_n^T$, where H is unitary Hessenberg, are referred to as fellow matrices in [6]. The root-finding problem for a polynomial expressed as a linear combination of Szegö polynomials reduces to the solution of an eigenvalue problem for a fellow matrix (see [1] and also [2]).

From the Schur parametrization it immediately follows that any unitary Hessenberg matrix H has low rank submatrices in its off-diagonal blocks. The property extends to companion and fellow matrices and is a manifestation of their *quasiseparable* structure. A quasiseparable matrix generally admits several different representations using a small (linear) number of parameters (see also the discussion in Sect. 2). By using (1.2), it is also found that an invertible matrix $A \in \mathcal{H}_n$ can be described in a compact way by means of about $8n$ parameters and, moreover, a similar representation is inherited by each matrix A_k generated by the QR iteration (1.3) applied to $A_0 = A$. Generally speaking, a matrix $B \in \mathbb{C}^{n \times n}$ is called quasiseparable of order (r, s) if $\operatorname{rank} B[k+1\colon n, 1\colon k] \leq r$ and $\operatorname{rank} B[1\colon k, k+1\colon n] \leq s$ for $k = 1, \ldots, n-1$. The class of (block) quasiseparable matrices was introduced and studied in the papers [8, 9, 10, 11], in the monograph [7] and in [20]. The paper [11] describes a fast $O(n)$ scheme for computing a QR factorization of a quasiseparable matrix of order (r, s) such that $r, s << n$. An efficient QR eigenvalue algorithm for nonsingular companion matrices based on the formula (1.2) has been recently designed in [5]. The algorithm is computationally appealing but numerically unstable due to the appearance of the inverse matrix in the formula (1.2).

1.2. Summary of the Results. In this paper we present an unified approach to the solution of the eigenvalue problem for unitary Hessenberg, companion and fellow matrices by means of the QR eigenvalue algorithm. Our results extend previously established techniques, improve the numerical performances of existing methods and lead to novel QR algorithms for eigenvalue computation of an input matrix $A = A_0 \in \mathcal{H}_n$. The algorithms are fast and numerically reliable by providing an acceptable compromise between speed and accuracy. More specifically, we underline the following achievements:

1. We devise a fast adaptation of the QR algorithm for eigenvalue computation of fellow matrices which circumvents the computational problems describe in [6] but, at the same time, the method remains robust and converges as fast

*MATLAB is a registered trademark of The Mathworks, Inc..

as the customary QR algorithm.

2. We design a fast and robust adaptation of the QR algorithm for companion matrices. The algorithm achieves stability without deterioration of the computational performances with respect to the fast variant described in [5].
3. We propose an alternative fast implementation of the QR method for the solution of unitary Hessenberg eigenvalue problems. Our scheme works directly with the matrix entries without requiring computations with characteristic polynomials and determinants. We feel reasonably safe in concluding that this could improve the numerical accuracy of the algorithm as compared with the versions in [15, 22, 21] at least in the case where shifts are not restricted to the unit circle.

The key idea of our approach is to find out a compact representation of the matrices A_k, $k = 0, 1, \ldots$, generated by the shifted QR algorithm (1.3) which neither explicitly nor implicitly involves quantities determined from the entries of A_k^{-1}. To do this we elaborate on the relation (1.1). From $A_{k+1} = Q_k^H A_k Q_k$, one obtains that

$$A_{k+1} = H_{k+1} + \boldsymbol{u}_{k+1}\boldsymbol{w}_{k+1}^H, \quad k \geq 0, \tag{1.4}$$

where H_{k+1} is unitary and, moreover, $H_{k+1} = Q_k^H H_k Q_k$, $\boldsymbol{u}_{k+1} = Q_k^H \boldsymbol{u}_k$ and $\boldsymbol{w}_{k+1} = Q_k^H \boldsymbol{w}_k$ subjected to the initializations $H_0 = H$, $\boldsymbol{u}_0 = u$ and $\boldsymbol{w}_0 = w$. Since A_{k+1} is Hessenberg, from (1.4) we deduce that H_{k+1} has a quasiseparable structure in its strictly lower triangular part. By exploiting this property, we derive a structural representation for the whole matrix H_{k+1} which turns out to be quasiseparable of order $(2, 2)$. Combining this representation with the formula (1.4) yields the sought compact description of the matrix A_{k+1} in terms of a linear number of parameters.

The structure of the matrix H_{k+1} is obtained by computing an additional QR factorization of H_{k+1}, say $H_{k+1} = U_{k+1}S_{k+1}$, where U_{k+1} is unitary and S_{k+1} is upper triangular with real positive diagonal entries. Theoretically we have $H_{k+1} = U_{k+1}$ and $S_{k+1} = I_n$, as H_{k+1} is assumed to be unitary. In floating point computation, it is found that $S_{k+1} = I_n + \Delta_{k+1}$ for a small perturbation Δ_{k+1}. So, in order to avoid the possible accumulation of errors in the matrices H_{k+1} and A_{k+1}, which can deteriorate their structural properties, in our implementations the new iterate A_{k+1} is defined by $A_{k+1} = U_{k+1} + \boldsymbol{u}_{k+1}\tilde{\boldsymbol{w}}_{k+1}^H$, where $\tilde{\boldsymbol{w}}_{k+1}^H = \boldsymbol{w}_{k+1}^H S_{k+1}^{-1}$. The entries of the corrected vector $\tilde{\boldsymbol{w}}_{k+1}$ are determined easily without explicitly computing the inverse of S_{k+1}. The matrix A_{k+1} is still upper Hessenberg whereas, at the same time, U_{k+1} is numerically unitary. Differently speaking, at each QR iteration the eigenvalue problem is slightly perturbed in such a way to preserve all the structures involved. The overall computational cost is $O(n)$ flops and $O(n)$ storage locations per iteration. Numerical experiments show that the algorithm exhibits a stable behavior.

1.3. Paper Organization. The paper is organized as follows. In Sect. 2 we recall some basic properties and algorithms for quasiseparable matrices. In Sect. 3 we first characterize the structure of a unitary matrix H such that $H + \boldsymbol{u}\boldsymbol{w}^H \in \mathcal{H}_n$ for two suitable vectors $\boldsymbol{u}$ and $\boldsymbol{w}$; then we apply this result to determine a numerically suited quasiseparable representation for the matrices A_k generated by the QR iteration (1.3) applied to an input matrix $A = A_0 \in \mathcal{H}_n$. In Sect. 4, we develop fast algorithms to carry out one single QR iteration in a fast and robust way using linear time and linear memory space. In Sect. 5 we discuss practical implementations of our QR-algorithm and present the results of extensive numerical experiments. Finally, conclusion and discussion are the subjects of Sect. 6.

2. Quasiseparable Structures: Definitions and Fast Algorithms. For any $n \times n$ matrix $B = (b_{i,j})$ we adopt the MATLAB notation $B[i\colon j, k\colon l]$ for the principal submatrix of B with entries having row and column indices in the ranges i through j and k through l, respectively. We also denote by $T = (t_{i,j}) = \mathrm{triu}(B, p)$ the upper triangular portion of B such that $t_{i,j} = b_{i,j}$ for $j - i \geq p$, and $t_{i,j} = 0$ elsewhere. Analogously, the $n \times n$ matrix $T = \mathrm{tril}(B, p)$ is formed by the lower triangular portion of B such that $t_{i,j} = b_{i,j}$ for $j - i \leq p$, and $t_{i,j} = 0$ elsewhere.

A matrix $A \in \mathbb{C}^n$ is called order (n_L, n_U)-*quasiseparable* [8] if

$$n_L = \max_{1 \leq k \leq n-1} \operatorname{rank} A[k+1\colon n, 1\colon k], \quad n_U = \max_{1 \leq k \leq n-1} \operatorname{rank} A[1\colon k, k+1\colon n].$$

In case $n_U = n_L = r$ one refers to A as an order-r-quasiseparable matrix. A computationally important property of $n \times n$ quasiseparable matrices is that they can be represented by only $O((n_L + n_U)n)$ parameters via *generators*. Given the set of $m \times m$ matrices $\{B_j\}_{j=2}^{n-1}$ and two positive integers i, j such that $1 < i + 1 \leq j \leq n$, we define the matrix $B_{i,j}^{\times}$ as follows: $B_{i,j}^{\times} = B_{i+1} \cdots B_{j-1}$ for $n \geq j > i + 1$ and $B_{j,j+1}^{\times} = I_m$ for $1 \leq j \leq n - 1$. Analogously, if $j + 1 \leq i \leq n$ then ${}^{\times}B_{i,j} = B_{i-1} \cdots B_{j+1}$ for $n \geq i > j + 1$ and ${}^{\times}B_{i,i-1} = I_m$ for $2 \leq i \leq n$. Then an order (n_L, n_U)-*quasiseparable* matrix $A = (a_{i,j}) \in \mathbb{C}^n$ can be represented as follows (see [7] and also [8]):

$$a_{i,j} = \begin{cases} \boldsymbol{p}_i^T \, {}^{\times}B_{i,j} \boldsymbol{q}_j, & 1 \leq j < i \leq n, \\ \boldsymbol{g}_i^T C_{i,j}^{\times} \boldsymbol{h}_j, & 1 \leq i < j \leq n. \end{cases} \tag{2.1}$$

Here $\boldsymbol{p}_i \in \mathbb{C}^{n_L}$, $2 \leq i \leq n$, $\boldsymbol{q}_i \in \mathbb{C}^{n_L}$, $1 \leq i \leq n-1$ and $B_i \in \mathbb{C}^{n_L \times n_L}$, $2 \leq i \leq n-1$; these elements are said to be *lower generators* of the matrix A. Similarly, the elements $\boldsymbol{g}_i \in \mathbb{C}^{n_U}$, $1 \leq i \leq n-1$, $\boldsymbol{h}_i \in \mathbb{C}^{n_U}$, $2 \leq i \leq n$ and $C_i \in \mathbb{C}^{n_U \times n_U}$, $2 \leq i \leq n-1$ are said to be *upper generators* of the matrix A. If the elements of a matrix are given via generators by the formula (2.1) we say that the matrix has a *quasiseparable structure.*

The class of quasiseparable matrices turns out to be closed under arithmetic operations, inversion, LU and QR factorization. More precisely the matrices generated are still quasiseparable with generally a different order of quasiseparability. Fast $O(n)$ algorithms for performing these operations which are based upon generator manipulations have already been devised in [8, 11, 10]. For our purposes a special mention is due to the algorithm for computing a QR factorization of a (block) quasiseparable matrix $A = QR$ presented in [11] (derived there via applying the more general Dewilde-Van der Veen method [7] to finite quasiseparable matrices). The algorithm first reduces the matrix A into a block upper Hessenberg form $S = V^H A$ and then transforms S into a triangular matrix $R = U^H S$. It is worth observing that in certain cases, where A has a very special quasiseparable structure in its lower triangular part, the triangularization process above can be modified in such a way that $Q = VU$ can be constructed in the usual manner as product of Givens rotations. This latter representation of Q is computationally equivalent to the representation via generators obtained by multiplying the quasiseparable matrices U and V using the multiplication algorithm in [8]. Nevertheless, the approach based on the use of Givens rotations is much more familiar in the numerical analysis community and will be pursued in this paper whenever it is possible.

3. The Quasiseparable Structure of QR Iterates. In this section we show the quasiseparable structure of the matrices A_k, $k = 0, 1, \ldots$, generated by the shifted QR algorithm (1.3) applied for the computation of the eigenvalues of the input Hessenberg matrix $A = A_0 \in \mathcal{H}_n$, $A = H + \boldsymbol{u}\boldsymbol{w}^H$, where $H \in \mathbb{C}^{n \times n}$ is unitary and

$\boldsymbol{u}, \boldsymbol{w} \in \mathbb{C}^n$. The main result follows from a basic property of the unitary matrix H; more precisely we show that H must be an order-2-quasiseparable matrix.

3.1. The Quasiseparable Structure of Certain Unitary Matrices. Let $P \in \mathbb{C}^{n\times n}$ be a unitary matrix such that the matrix $P - \boldsymbol{v}\boldsymbol{z}^H$ is a Hessenberg matrix for two suitable vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{C}^n$. Here we address the question of "What can be said about the (quasiseparable) structure of the whole matrix P?"

At first we consider in detail the case of a unitary Hessenberg matrix. A unitary Hessenberg matrix P with real nonnegative subdiagonal entries can be represented as a product of (modified) Givens rotations [14], i.e.,

$$\begin{aligned} &P = G_1(a_1)G_2(a_2)\cdots G_n(a_n) \\ &G_j(a_j) = I_{j-1} \oplus \begin{bmatrix} -a_j & b_j \\ b_j & \bar{a}_j \end{bmatrix} \oplus I_{n-j-1}, \quad 1 \le j \le n-1, \\ &G_n(a_n) = I_{n-1} \oplus (-a_n), \quad b_j \ge 0, |a_j|^2 + b_j^2 = 1, |a_n| = 1. \end{aligned} \tag{3.1}$$

The decomposition is usually referred to as the *Schur parametrization* of P. The a_j's are the *Schur parameters* of H and the b_j's are the *complementary parameters.* From (3.1) it follows [12] that

$$P = \begin{bmatrix} -\bar{a}_0 a_1 & -\bar{a}_0 b_1 a_2 & -\bar{a}_0 b_1 b_2 a_3 & \dots & -\bar{a}_0 b_1 \cdots b_{n-1} a_n \\ b_1 & -\bar{a}_1 a_2 & -\bar{a}_1 b_2 a_3 & \dots & -\bar{a}_1 b_2 \cdots b_{n-1} a_n \\ & b_2 & -\bar{a}_2 a_3 & & \vdots \\ & & \ddots & \ddots & \vdots \\ \bigcirc & & & b_{n-1} & -\bar{a}_{n-1} a_n \end{bmatrix}, \ (a_0 = 1),$$

which yields the representation of P via generators. For a general unitary Hessenberg matrix $U \in \mathbb{C}^{n\times n}$ we can determine a unitary diagonal matrix $D = \mathrm{diag}[\mathrm{e}^{\mathrm{i}\theta_1}, \dots, \mathrm{e}^{\mathrm{i}\theta_n}]$ such that $D^H U D$ has nonnegative subdiagonal entries. In this way it is found that U also admits the following representation

$$U = \begin{bmatrix} -\bar{\phi}_0 \phi_1 & -\bar{\phi}_0 \psi_1 \phi_2 & \dots & \dots & -\bar{\phi}_0 \psi_1 \cdots \psi_{n-1} \phi_n \\ \bar{\psi}_1 & -\bar{\phi}_1 \phi_2 & \dots & \dots & -\bar{\phi}_1 \psi_2 \cdots \psi_{n-1} \phi_n \\ & \bar{\psi}_2 & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ \bigcirc & & & \bar{\psi}_{n-1} & -\bar{\phi}_{n-1} \phi_n \end{bmatrix}, \ (\phi_0 = 1), \tag{3.2}$$

for suitable elements ϕ_j and ψ_j of modulus less than or equal to 1.

REMARK 3.1. *Given the entries of the matrix U, we can compute its parametrization or, equivalently, its generators by factoring U in the QR form. If $U = QR$, where Q is unitary and R is upper triangular with nonnegative diagonal entries, then we may conclude that $R = I_n$ and $Q = U$. Therefore, a careful exploitation of the structure of Q, expressed as the product of a linear number of (modified) Givens rotations, yields the desired characterization for the entries of U.*

Relation (3.2) says that any unitary Hessenberg matrix U is $(1,1)$ quasiseparable. The following results generalize this property to the case where the unitary matrix P is not assumed to be in Hessenberg form.

THEOREM 3.2. *Let $P = (p_{i,j}) \in \mathbb{C}^{n\times n}$ be a unitary matrix such that* $\mathrm{tril}(P, -2) = \mathrm{tril}(\boldsymbol{v}\boldsymbol{z}^H, -2)$ *for two suitable n-vectors $\boldsymbol{v} = [v_1, \dots, v_n]^T$ and $\boldsymbol{z} = [z_1, \dots, z_n]^T$. Then*

P admits the factorization $P = VU_1U$, where V, U_1 and U are unitary Hessenberg matrices. Moreover, V and U_1 have a Schur parametrization of the form $V^H = G_3(\tilde{a}_3)\cdots G_{n-1}(\tilde{a}_{n-1})$ and $U_1^H = G_{n-1}(\hat{a}_{n-1})\cdots G_2(\hat{a}_2)$, respectively.

Proof. The reduction of the matrix P into an upper Hessenberg form can be split in two phases. In the first stage the matrix P is transformed into a matrix $\tilde{P}$ of lower bandwidth 2, i.e., $\tilde{p}_{i,j} = 0$ whenever $i > j + 2$. The task is accomplished as follows. Choose the first rotation $G_{n-1}(\tilde{a}_{n-1}) = I_{n-2} \oplus \mathcal{G}(\tilde{a}_{n-1})$ to yield

$$\mathcal{G}(\tilde{a}_{n-1}) \begin{bmatrix} v_{n-1} \\ v_n \end{bmatrix} = \begin{bmatrix} \hat{v}_{n-1} \\ 0 \end{bmatrix}.$$

Similarly, choose the successive rotations $G_j(\tilde{a}_j) = I_{j-1} \oplus \mathcal{G}(\tilde{a}_j) \oplus I_{n-j+1}$, $3 \le j \le n-2$, to yield

$$\mathcal{G}(\tilde{a}_j) \begin{bmatrix} v_j \\ \hat{v}_{j+1} \end{bmatrix} = \begin{bmatrix} \hat{v}_j \\ 0 \end{bmatrix}, \quad j = n-2, \ldots, 3.$$

In the second phase the matrix

$$\tilde{P} = G_3(\tilde{a}_3)\cdots G_{n-1}(\tilde{a}_{n-1})P = V^H P$$

is reduced to an upper Hessenberg form by pre-multiplication by the matrices $G_2(\hat{a}_2)$, $\ldots$, $G_{n-1}(\hat{a}_{n-1})$ suitably chosen to annihilate the entries along the second subdiagonal. At the end of the process we find that the matrix

$$U = G_{n-1}(\hat{a}_{n-1})\cdots G_2(\hat{a}_2)\tilde{P} = U_1^H \tilde{P}$$

is unitary Hessenberg. □

REMARK 3.3. *The matrix V in the previous theorem is similar to the matrix V determined at the first stage of the algorithm in [11] applied for computing a QR factorization of the quasiseparable matrix P. At the second stage the cited algorithm reduces $\tilde{P}$ into a triangular form by annihilating its subdiagonal entries column-by-column. Differently, in the scheme above the annihilation of the subdiagonal entries of $\tilde{P}$ is performed one subdiagonal at the time.*

The factorization of P stated in the previous theorem leads to a characterization of the entries in the strictly upper triangular part of the matrix P. More specifically, let $U = U^{(0)}$, where U is defined as in Theorem 3.2. Recall that U can be represented in the form (3.2) for suitable elements ϕ_j and ψ_j of modulus less than or equal to 1. The composite scheme described in the proof of Theorem 3.2 for the reduction from $P = P^{(0)}$ to $U = U^{(0)}$ proceeds as follows:

$$\begin{aligned} P = P^{(0)} &\overset{\mathcal{G}(\tilde{a}_{n-1})}{\rightarrow} P^{(1)} \overset{\mathcal{G}(\tilde{a}_{n-2})}{\rightarrow} \ldots \overset{\mathcal{G}(\tilde{a}_3)}{\rightarrow} P^{(n-3)} \\ &= \tilde{P} \overset{\mathcal{G}(\hat{a}_2)}{\rightarrow} \ldots \overset{\mathcal{G}(\hat{a}_{n-1})}{\rightarrow} P^{(2n-5)} = U^{(0)}, \end{aligned} \tag{3.3}$$

where at the jth step $P^{(j-1)}$ is pre-multiplied by the corresponding Givens rotation to obtain $P^{(j)}$. Let us define $U^{(s)} = P^{(2n-5-s)} = (u_{i,j}^{(s)})$ for $0 \le s \le 2n-5$. The desired characterization of the entries of P is found by carrying out the scheme (3.3) in the reverse order, i.e.,

$$\begin{aligned} U = U^{(0)} &\overset{\mathcal{G}(\hat{a}_{n-1})^H}{\rightarrow} U^{(1)} \overset{\mathcal{G}(\hat{a}_{n-2})^H}{\rightarrow} \ldots \overset{\mathcal{G}(\hat{a}_2)^H}{\rightarrow} U^{(n-2)} \\ &= \tilde{P} \overset{\mathcal{G}(\tilde{a}_3)^H}{\rightarrow} \ldots \overset{\mathcal{G}(\tilde{a}_{n-1})^H}{\rightarrow} U^{(2n-5)} = P. \end{aligned} \tag{3.4}$$

We have

LEMMA 3.4. *Under the assumption of Theorem 3.2, let V and U_1 have a Schur parametrization of the form*

$$V^H = G_3(\tilde{a}_3)\cdots G_{n-1}(\tilde{a}_{n-1}), \quad U_1^H = G_{n-1}(\hat{a}_{n-1})\cdots G_2(\hat{a}_2),$$

respectively and let the matrix U parametrized in the form (3.2). Then there exist vectors $\boldsymbol{q}_j \in \mathbb{C}^2$, $1 \le j \le n$, vectors $\boldsymbol{t}_j \in \mathbb{C}^2$, $1 \le j \le n$, and lower triangular matrices $B_j \in \mathbb{C}^{2\times 2}$, $1 \le j \le n-1$, such that

$$\text{(3.5)} \quad \operatorname{triu}(\tilde{P}, 0) = \operatorname{triu}(U^{(n-2)}, 0) = \begin{bmatrix} \boldsymbol{q}_1^T \boldsymbol{t}_1 & \boldsymbol{q}_1^T B_1 \boldsymbol{t}_2 & \dots & \boldsymbol{q}_1^T B_1 \cdots B_{n-1} \boldsymbol{t}_n \\ & \boldsymbol{q}_2^T \boldsymbol{t}_2 & \dots & \boldsymbol{q}_2^T B_2 \cdots B_{n-1} \boldsymbol{t}_n \\ & & \ddots & \vdots \\ \bigcirc & & & \boldsymbol{q}_n^T \boldsymbol{t}_n \end{bmatrix}.$$

The matrices B_j are defined by

$$\text{(3.6)} \qquad B_1 = I_2, \quad B_j = \begin{bmatrix} \psi_{j-1} & 0 \\ \bar{\hat{a}}_j \bar{\phi}_{j-1} & \hat{b}_j \end{bmatrix}, \; 2 \le j \le n-1.$$

The vectors $\boldsymbol{q}_j$ and $\boldsymbol{t}_j$ are determined according to the following equations:

$$\text{(3.7)} \qquad \boldsymbol{q}_1^T = [-1, 0], \; \boldsymbol{q}_2^T = [0, 1], \quad \boldsymbol{q}_j^T = \left[-\hat{b}_{j-1} \bar{\phi}_{j-2}, \hat{a}_{j-1}\right], \quad 3 \le j \le n,$$

$$\text{(3.8)} \qquad \boldsymbol{t}_1 = \begin{bmatrix} \phi_1 \\ 0 \end{bmatrix}, \quad \boldsymbol{t}_j = \begin{bmatrix} \psi_{j-1} \phi_j \\ u_{j,j}^{(n-j)} \end{bmatrix}, \quad 2 \le j \le n,$$

where

$$u_{j,j}^{(n-j)} = -\bar{\hat{a}}_j u_{j,j}^{(0)} - \hat{b}_j \bar{\hat{a}}_{j+1} \bar{\psi}_j, \quad 2 \le j \le n-1.$$

Proof. The proof is by induction. We find that

$$U^{(1)}[n-1:n, n-1:n] = \begin{bmatrix} u_{n-1,n-1}^{(1)} & u_{n,n}^{(0)} \hat{b}_{n-1} + \bar{\hat{a}}_{n-1} \bar{\phi}_{n-2} \psi_{n-1} \phi_n \\ * & \hat{a}_{n-1} u_{n,n}^{(0)} - \hat{b}_{n-1} \bar{\phi}_{n-2} \psi_{n-1} \phi_n \end{bmatrix}.$$

Whence, we can write

$$u_{n,n}^{(1)} = \left[-\hat{b}_{n-1} \bar{\phi}_{n-2}, \hat{a}_{n-1}\right] \begin{bmatrix} \psi_{n-1} \phi_n \\ u_{n,n}^{(0)} \end{bmatrix} = \boldsymbol{q}_n^T \boldsymbol{t}_n.$$

Observe that

$$u_{n-1,n-2}^{(1)} = -\bar{\hat{a}}_{n-1} \bar{\psi}_{n-2}, \; u_{n-1,n-1}^{(1)} = [0, 1] \begin{bmatrix} \psi_{n-2} \phi_{n-1} \\ u_{n-1,n-1}^{(1)} \end{bmatrix} = [0, 1]\, \boldsymbol{t}_{n-1},$$

and

$$u_{n-1,n}^{(1)} = [0, 1] \begin{bmatrix} \psi_{n-2} & 0 \\ \bar{\hat{a}}_{n-1} \bar{\phi}_{n-2} & \hat{b}_{n-1} \end{bmatrix} \begin{bmatrix} \psi_{n-1} \phi_n \\ u_{n,n}^{(0)} \end{bmatrix} = [0, 1]\, B_{n-1} \boldsymbol{t}_n.$$

Moreover, we have

$$u^{(1)}_{n-2,n-1} = \left[-\bar{\phi}_{n-3}, 0\right] \left[\begin{array}{c} \psi_{n-2}\phi_{n-1} \\ u^{(1)}_{n-1,n-1} \end{array}\right] = \left[-\bar{\phi}_{n-3}, 0\right] \boldsymbol{t}_{n-1},$$

and

$$u^{(1)}_{n-2,n} = \left[-\bar{\phi}_{n-3}, 0\right] \left[\begin{array}{cc} \psi_{n-2} & 0 \\ \bar{\tilde{a}}_{n-1}\phi_{n-2} & \hat{b}_{n-1} \end{array}\right] \left[\begin{array}{c} \psi_{n-1}\phi_n \\ u^{(0)}_{n,n} \end{array}\right] = \left[-\bar{\phi}_{n-3}, 0\right] B_{n-1}\boldsymbol{t}_n.$$

Now, suppose that

$$u^{(k)}_{n-k,n-k-1} = -\bar{\tilde{a}}_{n-k}\bar{\psi}_{n-k-1},\ u^{(k)}_{n-k,j} = [0,1]\, B_{n-k}\cdots B_{j-1}\boldsymbol{t}_j, \quad n-k \le j \le n.$$

Since

$$u^{(k)}_{n-k-1,j} = \left[-\bar{\phi}_{n-k-2}, 0\right] B_{n-k}\cdots B_{j-1}\boldsymbol{t}_j, \quad n-k \le j \le n,$$

it follows that

$$u^{(k+1)}_{n-k,j} = \boldsymbol{q}^T_{n-k} B_{n-k}\cdots B_{j-1}\boldsymbol{t}_j, \quad n-k \le j \le n,$$

and, moreover, for $n-k \le j \le n$,

$$u^{(k+1)}_{n-k-1,j} = \left[\bar{\tilde{a}}_{n-k-1}\bar{\phi}_{n-k-2}, \hat{b}_{n-k-1}\right] B_{n-k}\cdots B_{j-1}\boldsymbol{t}_j = [0,1]\, B_{n-k-1}\cdots B_{j-1}\boldsymbol{t}_j,$$

whereas the diagonal entry is determined by

$$u^{(k+1)}_{n-k-1,n-k-1} = -\bar{\tilde{a}}_{n-k-1}u^{(0)}_{n-k-1,n-k-1} - \hat{b}_{n-k-1}\bar{\tilde{a}}_{n-k}\bar{\psi}_{n-k-1}.$$

□

Now notice that each of the remaining step $U^{(j)} \to U^{(j+1)}$ in (3.4), $n-2 \le j \le 2n-6$, amounts to perform a linear combination between two consecutive rows of $U^{(j)}$ starting from the third row without modifying the general structure of the matrix. In this way, we arrive at the following:

Theorem 3.5. *Under the assumption of Theorem 3.2, let V and U_1 have a Schur parametrization of the form*

$$V^H = G_3(\tilde{a}_3)\cdots G_{n-1}(\tilde{a}_{n-1}), \quad U_1^H = G_{n-1}(\hat{a}_{n-1})\cdots G_2(\hat{a}_2),$$

respectively and let the matrix U parametrized in the form (3.2).

Then there exist vectors $\tilde{\boldsymbol{q}}_j \in \mathbb{C}^2$, $1 \le j \le n-1$, vectors $\boldsymbol{t}_j \in \mathbb{C}^2$, $2 \le j \le n$, and lower triangular matrices $B_j \in \mathbb{C}^{2\times 2}$, $2 \le j \le n-1$, such that the strictly upper triangular part of the unitary matrix P admits the following representation:

$$p_{i,j} = \tilde{\boldsymbol{q}}_i^T B^{\times}_{i,j}\boldsymbol{t}_j, \ \text{for } i+1 \le j \le n. \tag{3.9}$$

The matrices B_j and the vectors $\boldsymbol{t}_j$ are defined as in (3.6) *and* (3.8), *respectively. The vectors $\tilde{\boldsymbol{q}}_j$ are generated by the following two-step procedure subjected to the initializations $\tilde{\boldsymbol{q}}_1^T = \boldsymbol{q}_1^T B_1$, $\tilde{\boldsymbol{q}}_2^T = \boldsymbol{q}_2^T B_2$, $\hat{\boldsymbol{q}}_3^T = \boldsymbol{q}_3^T$, where the vectors $\boldsymbol{q}_j$ are given by (3.7).*

```
**for** $j = 3 \ : \ n-1$
  *1.* $\tilde{\boldsymbol{q}}_j^T = -\bar{\tilde{a}}_j\hat{\boldsymbol{q}}_j^T B_j + \tilde{b}_j\boldsymbol{q}_{j+1}^T$;
```

2. $\hat{\boldsymbol{q}}_{j+1}^T = \tilde{b}_j \hat{\boldsymbol{q}}_j^T B_j + \tilde{a}_j \boldsymbol{q}_{j+1}^T$;
end

Proof.

$$\boldsymbol{T}_N = \boldsymbol{t}_n, \quad \boldsymbol{T}_i = \begin{bmatrix} \boldsymbol{t}_i & B_i \boldsymbol{t}_{i+1} & \dots & B_i \cdots B_{n-1} \boldsymbol{t}_n \end{bmatrix}, \; i = n-1, \dots, 1. \tag{3.10}$$

By Lemma 3.4 the upper triangular part of the matrix $\tilde{P}$ has the form (3.5) and hence we have

$$\tilde{P}[i, i+1:n] = \boldsymbol{q}_i^T B_i \boldsymbol{T}_{i+1}, \; \tilde{P}[i+1, i+1:n] = \boldsymbol{q}_{i+1}^T \boldsymbol{T}_{i+1}, \quad i = 1, \dots, n-1.$$

The transformation

$$P = G_{n-1}^H(\tilde{a}_{n-1}) \cdots G_3^H(\tilde{a}_3) \tilde{P}$$

does not change the first two rows of the matrix $\tilde{P}$. Hence setting $\tilde{\boldsymbol{q}}_1^T = \boldsymbol{q}_1^T B_1$, $\tilde{\boldsymbol{q}}_2^T = \boldsymbol{q}_2^T B_2$ we obtain (3.9) for $i = 1, 2$. Next we prove by induction that

$$(G_i^H(\tilde{a}_{n-1}) \cdots G_3^H(\tilde{a}_3) \tilde{P})[i : i+1, i+1 : n] = \begin{bmatrix} \tilde{\boldsymbol{q}}_i^T \\ \hat{\boldsymbol{q}}_{i+1}^T \end{bmatrix} \boldsymbol{T}_{i+1}, \quad i = 3, \dots, n-1. \tag{3.11}$$

From here using

$$(G_i^H(\tilde{a}_i) \cdots G_3^H(\tilde{a}_3) \tilde{P})[i,] = P[i, :], \quad i = 3, \dots, n-1$$

we will get (3.9). Notice that by the definition of the vectors $\tilde{\boldsymbol{q}}_j$, $\hat{\boldsymbol{q}}_j$ we have

$$\mathcal{G}^H(\tilde{a}_j) \begin{bmatrix} \hat{\boldsymbol{q}}_j^T B_j \\ \boldsymbol{q}_{j+1}^T \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{q}}_j^T \\ \hat{\boldsymbol{q}}_{j+1}^T \end{bmatrix}.$$

So setting $\hat{\boldsymbol{q}}_3^T = \boldsymbol{q}_3^T$ we get

$$(G_3^H(\tilde{a}_3) \tilde{P})[3:4, 4:n] = \mathcal{G}^H(\tilde{a}_3) \begin{bmatrix} \hat{\boldsymbol{q}}_3^T B_3 \\ \boldsymbol{q}_4^T \end{bmatrix} \boldsymbol{T}_4 = \begin{bmatrix} \tilde{\boldsymbol{q}}_3^T \\ \hat{\boldsymbol{q}}_4^T \end{bmatrix} \boldsymbol{T}_4.$$

Assume that (3.11) holds for some $i: \; 3 \le i \le N-2$. We obtain

$$(G_{i+1}^H(\tilde{a}_{i+1}) \cdots G_3^H(\tilde{a}_3) \tilde{P})[i+1 : i+2, i+2 : N] = \mathcal{G}^H(\tilde{a}_{i+1}) \begin{bmatrix} \hat{\boldsymbol{q}}_{i+1}^T \boldsymbol{T}_{i+1} \\ \boldsymbol{q}_{i+2}^T \boldsymbol{T}_{i+2} \end{bmatrix}$$

and using the relation $\boldsymbol{T}_{i+1} = \begin{bmatrix} \boldsymbol{t}_{i+1} & B_{i+1} \boldsymbol{T}_{i+1} \end{bmatrix}$ we get

$$(G_{i+1}^H(\tilde{a}_{n-1}) \cdots G_3^H(\tilde{a}_3) \tilde{P})[i+1 : i+2, i+2 : n] = \begin{bmatrix} \tilde{\boldsymbol{q}}_{i+1}^T \\ \hat{\boldsymbol{q}}_{i+2}^T \end{bmatrix} \boldsymbol{T}_{i+2}.$$

□

REMARK 3.6. *A more detailed analysis also reveals that the parameters defining the representation of the strictly upper triangular part of P can be bounded from above in magnitude by $O(n)$. The existence of such an uniform bound is highly important from a numerical point of view since it implies the robustness of the representation. Differently saying, absolute perturbations in the parameters can not magnify to give a large absolute perturbation of the considered matrix.*

3.2. Main Result. We are now in position to prove the main result of this section concerning the quasiseparable structure of the matrices A_k, $k = 0, 1, \ldots$, generated by the shifted QR algorithm (1.3) applied for the computation of the eigenvalues of the input Hessenberg matrix $A = A_0 \in \mathcal{H}_n$, $A = H + \boldsymbol{u}\boldsymbol{w}^H$, where $H \in \mathbb{C}^{n\times n}$ is unitary and $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{C}^n$. Since $A = A_0$ is upper Hessenberg, it is well known that each iterate A_k has the same Hessenberg shape as A_0, i.e.,

$$A_k = \begin{bmatrix} a_{1,1}^{(k)} & a_{1,2}^{(k)} & \ldots & a_{1,n}^{(k)} \\ a_{2,1}^{(k)} & a_{2,2}^{(k)} & \ldots & a_{2,n}^{(k)} \\ & \ddots & \ddots & \vdots \\ \bigcirc & & a_{n,n-1}^{(k)} & a_{n,n}^{(k)} \end{bmatrix}.$$

For notational convenience we also denote as $\beta_j^{(k)} = a_{j+1,j}^{(k)}$, $1 \le j \le n-1$, the subdiagonal entries of A_k. Since $A_{k+1} = Q_k^H A_k Q_k$ for any $k \ge 0$, from (1.1) we obtain

$$\begin{aligned} & A_{k+1} = H_{k+1} + \boldsymbol{u}_{k+1}\boldsymbol{w}_{k+1}^H \\ & H_{k+1} = Q_k^H H_k Q_k \\ & \boldsymbol{u}_{k+1} = Q_k^H \boldsymbol{u}_k, \ \boldsymbol{w}_{k+1} = Q_k^H \boldsymbol{w}_k, \end{aligned} \tag{3.12}$$

subjected to the initializations $H = H_0 \in \mathbb{C}^{n\times n}$, $\boldsymbol{u}_0 = \boldsymbol{u} = \left[u_1^{(0)}, \ldots, u_n^{(0)}\right]^T$ and $\boldsymbol{w}_0 = \boldsymbol{w} = \left[w_1^{(0)}, \ldots, w_n^{(0)}\right]^T$.

Based on the relations (3.12) we are able to derive a quasiseparable representation for the matrices A_k, $k \ge 0$. Because of the Hessenberg form of A_k from (3.12) we have

$$\begin{aligned} & h_{i,j}^{(k)} = -u_i^{(k)}\bar{w}_j^{(k)}, \text{ for } i - j \ge 2 \\ & h_{j+1,j}^{(k)} = \beta_j^{(k)} - u_{j+1}^{(k)}\bar{w}_j^{(k)}. \end{aligned} \tag{3.13}$$

This means that each matrix H_k meets the assumption of Theorem 3.2. Therefore, by substituting the representation provided by Theorem 3.5 into the equations (3.12) we obtain

THEOREM 3.7. *For each upper Hessenberg matrix $A_k = (a_{i,j}^{(k)})$ generated by (1.3) applied to the generalized companion matrix $A_0 = H_0 + \boldsymbol{u}_0\boldsymbol{w}_0^H$, there exist vectors $\boldsymbol{q}_j^{(k)} \in \mathbb{C}^2$, $1 \le j \le n-1$, vectors $\boldsymbol{t}_j^{(k)} \in \mathbb{C}^2$, $2 \le j \le n$ and lower triangular matrices $B_j^{(k)} \in \mathbb{C}^{2\times 2}$, $2 \le j \le n-1$, such that*

$$a_{i,j}^{(k)} = \boldsymbol{q}_i^{(k)T} B_{i,j}^{(k)\times} \boldsymbol{t}_j^{(k)} + u_i^{(k)}\bar{w}_j^{(k)}, \text{ for } i+1 \le j \le n. \tag{3.14}$$

REMARK 3.8. *It is instructive to compare (3.14) with the representation of A_k used in the paper [5] and based on the formula (1.2). Suppose that $A = A_0$ is ill conditioned; more precisely, assume that A has a fixed norm but A^{-1} has an arbitrarily large norm. Then $\| A_k \|_2 = \| A \|_2$ and, moreover, from (1.2) we find that the strictly upper triangular part of A_k is the sum of the strictly triangular parts of three rank-one matrices. The first one is $\boldsymbol{u}_k\boldsymbol{w}_k^H$ that is bounded. On the contrary,* *the other two remaining rank-one matrices can have arbitrarily large entries which must combine together so that the cumulative contribution has bounded norm. Hence, in the considered situation the formula* (1.2) *would generate large cancellation errors and would not be robust. Differently, the representation* ${\boldsymbol{q}_i^{(k)}}^T {B_{i,j}^{(k)}}^{\times} \boldsymbol{t}_j^{(k)}$ *used for the entries of the upper triangular matrix* $\mathrm{triu}(A_k - \boldsymbol{u}_k \boldsymbol{w}_k^H, 1)$ *enables the contribution to be expressed in a numerically robust way by using bounded quantities and thus by taking under control potential cancellation errors.*

4. The Structured QR Iteration. In this section we describe a fast adaptation of the shifted QR eigenvalue algorithm (1.3) for an input matrix $A = A_0 \in \mathcal{H}_n$. Exploiting the quasiseparable structure (3.14) of each iterate A_k enables us to yield linear time and linear memory space per iteration. Given a condensed representation of the form (3.14) for the matrix A_k our algorithm computes a similar representation for the matrix A_{k+1} defined as in (1.3). The resulting process is referred to as the *structured QR iteration.* The computation proceeds as follows:

1. Compute Q_k and R_k such that $A_k - \alpha_k I_n = Q_k R_k$ provides a QR factorization of the left-hand side matrix.
2. Determine the vectors $\boldsymbol{u}_{k+1} := Q_k^H \boldsymbol{u}_k$, $\boldsymbol{w}_{k+1} := Q_k^H \boldsymbol{w}_k$ and the unitary matrix $H_{k+1} := Q_k^H H_k Q_k = R_k Q_k + \alpha_k I_n - \boldsymbol{u}_{k+1} \boldsymbol{w}_{k+1}^H$.
3. Set $A_{k+1} := H_{k+1} + \boldsymbol{u}_{k+1} \boldsymbol{w}_{k+1}^H$.

In the view of (3.12), we see that the computed matrix A_{k+1} coincides with the one obtained after having performed one step of the shifted QR process (1.3) starting from A_k. From Theorem 3.7 it follows that both A_k and A_{k+1} can be represented in the condensed form (3.14) by means of $O(n)$ parameters. In the sequel, we will show that the same holds for all the data structures involved in the computational process above. In this way matrix operations can be reduced to manipulating $O(n)$ rather than $O(n^2)$ elements with a dramatic reduction of the operation count.

Let us first specify the parametrization used for the matrices A_k and H_k. According to the notations introduced in the previous section, the input data at iteration $k+1$ are given by:

1. the vector $\boldsymbol{\beta}^{(k)} = \left[\beta_1^{(k)}, \dots, \beta_{n-1}^{(k)}\right]^T \in \mathbb{C}^{n-1}$ of the subdiagonal entries of A_k;
2. the vector $\boldsymbol{a}^{(k)} = \left[a_{1,1}^{(k)}, \dots, a_{n,n}^{(k)}\right]^T \in \mathbb{C}^n$ of the diagonal entries of A_k;
3. the vector $\boldsymbol{h}^{(k)} = \left[h_{1,1}^{(k)}, \dots, h_{n,n}^{(k)}\right]^T \in \mathbb{C}^n$ of the diagonal entries of H_k;
4. the vectors $\boldsymbol{q}_j^{(k)} \in \mathbb{C}^2$, $1 \le j \le n-1$;
5. the vectors $\boldsymbol{t}_j^{(k)} \in \mathbb{C}^2$, $2 \le j \le n$;
6. the lower triangular matrices $B_j^{(k)} = \begin{bmatrix} \hat{\psi}_j^{(k)} & 0 \\ \hat{\phi}_j^{(k)} & \hat{\rho}_j^{(k)} \end{bmatrix} \in \mathbb{C}^{2\times 2}$, $2 \le j \le n-1$;
7. the vector $\boldsymbol{u}_k = \left[u_1^{(k)}, \dots, u_n^{(k)}\right]^T \in \mathbb{C}^n$;
8. the vector $\boldsymbol{w}_k = \left[w_1^{(k)}, \dots, w_n^{(k)}\right]^T \in \mathbb{C}^n$;
9. the shift parameter α_k.

The entries of the unitary matrix $H_k = (h_{i,j}^{(k)})$ are given by

$$\begin{aligned} h_{i,j}^{(k)} &= -u_i^{(k)}\bar{w}_j^{(k)}, \text{ for } i-j \geq 2 \\ h_{j+1,j}^{(k)} &= \beta_j^{(k)} - u_{j+1}^{(k)}\bar{w}_j^{(k)} \\ h_{j,j}^{(k)} &= (\boldsymbol{h}^{(k)})_j \\ h_{i,j}^{(k)} &= \boldsymbol{q}_i^{(k)^T} B_{i+1}^{(k)\,\times} \boldsymbol{t}_j^{(k)}, \text{ for } j-i \geq 1. \end{aligned} \tag{4.1}$$

Analogously, the entries of the upper Hessenberg matrix A_k are defined by

$$\begin{aligned} a_{i,j}^{(k)} &= 0, \text{ for } i-j \geq 2 \\ a_{j+1,j}^{(k)} &= \beta_j^{(k)} \\ a_{j,j}^{(k)} &= (\boldsymbol{a}^{(k)})_j \\ a_{i,j}^{(k)} &= \boldsymbol{q}_i^{(k)^T} B_{i+1}^{(k)\,\times} \boldsymbol{t}_j^{(k)} + u_i^{(k)}\bar{w}_j^{(k)}, \text{ for } j-i \geq 1. \end{aligned} \tag{4.2}$$

If H_0 is unitary Hessenberg, then at the very beginning for $k=0$ we may set $\boldsymbol{q}_j^{(0)} = \left[-\bar{\phi}_{j-1}^{(0)}\psi_j^{(0)}, 0\right]$, $\boldsymbol{t}_j^{(0)} = \left[\phi_j^{(0)}, 0\right]^T$ and $B_j^{(0)} = \begin{bmatrix} \psi_j^{(0)} & 0 \\ 0 & 0 \end{bmatrix}$, where the $\phi_j^{(0)}$'s and the $\psi_j^{(0)}$'s define the (modified) Schur parametrization for H_0. Otherwise, if H_0 is not in Hessenberg form, the entries of $\boldsymbol{q}_j^{(0)}$, $\boldsymbol{t}_j^{(0)}$ and $B_j^{(0)}$ are to be determined according to Theorem 3.5 and Remark 3.1 by performing a preliminary QR factorization of H_0.

In the next subsection the structure of the matrix A_k is used in order to prove that the unitary factor Q_k and the upper triangular factor R_k can be determined in linear time. The design of an efficient method for computing the parametrization of H_{k+1} given a structural representation for A_k, Q_k and R_k is the subject in Subsections 4.2, 4.3 and 4.4. Finally, in Subsection 4.5 we give the details about the deflation strategy.

4.1. The QR Step. In this section we describe the QR step applied to A_k and as a byproduct we obtain the structure of R_k and Q_k. For the sake of notational simplicity we omit the superscript (k) and denote by α the shift parameter α_k. Since $A_k - \alpha I_n$ is upper Hessenberg, the reduction to upper triangular form can be achieved by means of a sequence of Givens rotations $G_1(\hat{a}_1), \ldots, G_{n-1}(\hat{a}_{n-1})$ suitably chosen to annihilate the subdiagonal entries. Recall that

$$G_j(\hat{a}_j) = \begin{bmatrix} I_{j-1} & & & \\ & -\hat{a}_j & \hat{b}_j & \\ & \hat{b}_j & \bar{\hat{a}}_j & \\ & & & I_{n-j-1} \end{bmatrix} = I_{j-1} \oplus \mathcal{G}_j(\hat{a}_j) \oplus I_{n-j-1}, \; 1 \leq j \leq n-1,$$

where $\hat{b}_j \geq 0$ and $|\hat{a}_j|^2 + \hat{b}_j^2 = 1$. At the first step $G_1(\hat{a}_1)$ is chosen so that the entry in position $(2,1)$ of $G_1(\hat{a}_1)(A_k - \alpha I_n) = (\tilde{a}_{i,j})$ is made zero. Observe that only the entries in the first two lines of $G_1(\hat{a}_1)(A_k - \alpha I_n)$ differ from the entries of $A_k - \alpha I_n$. These entries satisfy

$$\tilde{a}_{1,j} = \tilde{\boldsymbol{q}}_{\boldsymbol{1}}^T B_{2,j}^{\times} \boldsymbol{t}_j + \tilde{u}_1 \bar{w}_j, \quad \tilde{a}_{2,j} = \hat{\boldsymbol{q}}_{\boldsymbol{2}}^T B_{2,j}^{\times} \boldsymbol{t}_j + \hat{u}_2 \bar{w}_j, \quad j > 2,$$

where

$$\begin{bmatrix} \tilde{\boldsymbol{q}}_{\boldsymbol{1}}^T \\ \hat{\boldsymbol{q}}_{\boldsymbol{2}}^T \end{bmatrix} = \mathcal{G}_1(\hat{a}_1) \begin{bmatrix} \boldsymbol{q}_{\boldsymbol{1}}^T B_2 \\ \boldsymbol{q}_{\boldsymbol{2}}^T \end{bmatrix}, \quad \begin{bmatrix} \tilde{u}_1 \\ \hat{u}_2 \end{bmatrix} = \mathcal{G}_1(\hat{a}_1) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{4.3}$$

The remaining entries in the 2×2 leading principal submatrix of $G_1(\hat{a}_1)(A_k - \alpha I_n)$ are given by

$$\begin{bmatrix} d_1 & g_1 \\ 0 & d_2 \end{bmatrix} = \mathcal{G}_1(\hat{a}_1) \begin{bmatrix} a_{1,1} - \alpha & \boldsymbol{q_1}^T \boldsymbol{t}_2 + u_1 \bar{w}_2 \\ \beta_1 & a_{2,2} - \alpha \end{bmatrix}.$$

The values of $\tilde{\boldsymbol{q}}_{\boldsymbol{1}}{}^T$, $\tilde{u}_1$, d_1 and g_1 are not modified by the subsequent Givens rotations, while the values of $\hat{\boldsymbol{q}}_{\boldsymbol{2}}{}^T$, $\hat{u}_2$ and d_2 only change at the second step where the matrix $G_1(\hat{a}_1)(A_k - \alpha I_n)$ is pre-multiplied by $G_2(\hat{a}_2)$. At the j-th step, $G_j(\hat{a}_j)$ is chosen so that the entry in position $(j+1, j)$ of $G_{j-1}(\hat{a}_{j-1}) \cdots G_1(\hat{a}_1)(A_k - \alpha I_n)$ is made zero. The entire process is synthesized below. The function *Givens* constructs $\hat{a}_j$ and $\hat{b}_j$ and guards against the risk of overflow. We refer to Bindel et al. [4] for a detailed explanation on how this function should be implemented.

```
for $j = 1 : n$; $d_j = a_{j,j} - \alpha$; end
for $j = 1 : n-1$; $\hat{\boldsymbol{q}}_j = \boldsymbol{q}_j$; $\hat{u}_j = u_j$; end
for $j = 1 : n-2$
        $(\hat{a}_j, \hat{b}_j) = Givens(d_j, \beta_j)$;
        $d_j = -\hat{a}_j d_j + \hat{b}_j \beta_j$;
        $g = \hat{\boldsymbol{q}}_j^T \boldsymbol{t}_{j+1} + \hat{u}_j \bar{w}_{j+1}$;
        $g_j = -\hat{a}_j g + \hat{b}_j d_{j+1}$;
        $d_{j+1} = \hat{b}_j g + \bar{\hat{a}}_j d_{j+1}$;
        $\tilde{\boldsymbol{q}}_j^T = -\hat{a}_j \hat{\boldsymbol{q}}_j^T B_{j+1} + \hat{b}_j \hat{\boldsymbol{q}}_{j+1}^T$;
        $\hat{\boldsymbol{q}}_{j+1}^T = \hat{b}_j \hat{\boldsymbol{q}}_j^T B_{j+1} + \bar{\hat{a}}_j \hat{\boldsymbol{q}}_{j+1}^T$;
        $\tilde{u}_j = -\hat{a}_j \hat{u}_j + \hat{b}_j \hat{u}_{j+1}$;
        $\hat{u}_{j+1} = \hat{b}_j \hat{u}_j + \bar{\hat{a}}_j \hat{u}_{j+1}$;
end
$(\hat{a}_{n-1}, \hat{b}_{n-1}) = Givens(d_{n-1}, \beta_{n-1})$;
$d_{n-1} = -\hat{a}_{n-1} d_{n-1} + \hat{b}_{n-1} \beta_{n-1}$;
$g = \hat{\boldsymbol{q}}_{n-1}^T \boldsymbol{t}_n + \hat{u}_{n-1} \bar{w}_n$;
$g_{n-1} = -\hat{a}_{n-1} g + \hat{b}_{n-1} d_n$;
$d_n = \hat{b}_{n-1} g + \bar{\hat{a}}_{n-1} d_n$;
$\tilde{u}_{n-1} = -\hat{a}_{n-1} \hat{u}_{n-1} + \hat{b}_{n-1} \hat{u}_n$;
$\tilde{u}_n = \hat{b}_{n-1} \hat{u}_{n-1} + \bar{\hat{a}}_{n-1} \hat{u}_n$;
```

The cost is about $40n$ flops. At the end of the entire process the upper triangular matrix $R_k = (r_{i,j})$ turns out to be represented as follows:

$$\begin{aligned} r_{i,i} &= d_i \\ r_{i,i+1} &= g_i \\ r_{i,j} &= \tilde{\boldsymbol{q}}_i^T B_{i+1,j}^{\times} \boldsymbol{t}_j + \tilde{u}_i \bar{w}_j \text{ for } j - i \geq 2. \end{aligned} \tag{4.4}$$

This representation involves approximately $9n$ parameters.

From (3.1) it follows that the unitary matrix $Q_k = (G_{n-1}(\hat{a}_{n-1}) \cdots G_1(\hat{a}_1))^H$ is upper Hessenberg with Schur parameters $\{\bar{\hat{a}}_j\}$, $\hat{a}_n = -1$, and complementary parameters $\{\bar{b}_j\}$. The computation of the diagonal and subdiagonal entries of A_{k+1} essentially reduces to evaluating the corresponding entries of the matrix $R_k Q_k$. The next simple procedure performs this latter task at the cost of $5n$ flops.

```
for $j = 1 : n-1$
        $d_j = -\bar{\hat{a}}_j d_j + \hat{b}_j g_j$;
        $\beta_j = \hat{b}_j d_{j+1}$;
        $d_{j+1} = \hat{a}_j d_{j+1}$;
end
```

The β_j's are the subdiagonal entries of A_{k+1}. The diagonal entries are defined by $a_{j,j} = d_j + \alpha$, $1 \leq j \leq n$. It is worth pointing out that a slight modification of the procedure above makes it possible to compute a quasiseparable representation of A_{k+1} given the representations of R_k and Q_k. However, such an approach has the following noticeable drawback: We know that the strictly upper triangular part of A_{k+1} admits a quasiseparable representation of order 3 but the representation computed by multiplying the structures of R_k and Q_k has order 4. To circumvent this difficulty, we devise a compression strategy based on the computation of an additional QR factorization of the matrix H_{k+1}. More specifically, first we describe two different ways for computing a possibly redundant quasiseparable representation for H_{k+1} and then we show that the computation of a QR factorization of H_{k+1} leads to the desired parametrization of minimal length for H_{k+1} and, a fortiori, for A_{k+1}.

4.2. Computing a Quasiseparable Representation of $\boldsymbol{H_{k+1}}$: The First Method. From (3.12) we obtain that $H_{k+1} = R_k Q_k + \alpha I_n - \boldsymbol{u}_{k+1}\boldsymbol{w}_{k+1}^H$. Since the matrices R_k and Q_k as well as the vectors $\boldsymbol{u}_{k+1}$ and $\boldsymbol{w}_{k+1}$ are already available, this relation yields the simplest approach for the computation of a quasiseparable structure of H_{k+1}. By starting from the representation of R_k given in (4.4), we can easily compute a quasiseparable representation of the matrix $R_k Q_k$. Observe that $\boldsymbol{u}_{k+1} = (\tilde{u}_i)$, where $\tilde{u}_i$ are generated by the procedure for finding the entries of R_k and Q_k. Let $\boldsymbol{w}_{k+1} = (\tilde{w}_i)$. At the first step we have

$$\begin{bmatrix} d_1 & g_1 \\ 0 & d_2 \end{bmatrix} \mathcal{G}_1(\hat{a}_1)^H = \begin{bmatrix} d_1 & \hat{g}_1 \\ \beta_1 & d_2 \end{bmatrix},$$

where

$$\hat{g}_1 = d_1 \hat{b}_1 + g_1 \hat{a}_1 = \tilde{g}_1 + \tilde{u}_1 \hat{w}_2, \quad \hat{w}_2 = \hat{b}_1 \bar{w}_1 + \hat{a}_1 \bar{w}_2.$$

Observe that

$$\tilde{g}_1 = \left[\tilde{\boldsymbol{q}}_1^T, \tilde{g}_1\right] \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix}, \tilde{\boldsymbol{q}}_1^T \boldsymbol{t}_3 = \left[\tilde{\boldsymbol{q}}_1^T, \tilde{g}_1\right] \begin{bmatrix} \boldsymbol{t}_3 \\ 0 \end{bmatrix}.$$

Hence, at the successive step we find

$$\begin{bmatrix} \tilde{g}_1 + \tilde{u}_1 \tilde{w}_2 & \tilde{\boldsymbol{q}}_1^T \boldsymbol{t}_3 + \tilde{u}_i \bar{w}_3 \\ d_2 & g_2 \end{bmatrix} \mathcal{G}_2(\hat{a}_2)^H = \begin{bmatrix} \hat{\boldsymbol{q}}_1^T \tilde{\boldsymbol{z}}_2 + \tilde{u}_1 \tilde{w}_2 & \hat{\boldsymbol{q}}_1^T \hat{\boldsymbol{z}}_2 + \tilde{u}_i \hat{w}_3 \\ d_2 & \hat{g}_2 \end{bmatrix},$$

where

$$\hat{\boldsymbol{q}}_1^T = \left[\tilde{\boldsymbol{q}}_1^T, \tilde{g}_1\right], [\tilde{\boldsymbol{z}}_2, \hat{\boldsymbol{z}}_{\boldsymbol{2}}] = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{t}_3 \\ 1 & 0 \end{bmatrix} \mathcal{G}_2(\hat{a}_2)^H, [\tilde{w}_2, \hat{w}_3] = [\hat{w}_2, \bar{w}_3]\, \mathcal{G}_2(\hat{a}_2)^H.$$

Since

$$\hat{\boldsymbol{q}}_1^T \hat{\boldsymbol{z}}_2 = \hat{\boldsymbol{q}}_1^T \left[\begin{array}{c|c} \dfrac{B_3}{\boldsymbol{0}^T} & \hat{\boldsymbol{z}}_2 \end{array} \right] \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix} = \hat{\boldsymbol{q}}_1^T F_2 \begin{bmatrix} \boldsymbol{0} \\ 1 \end{bmatrix}, \quad F_2 \in \mathbb{C}^{3\times 3},$$

and

$$\tilde{\boldsymbol{q}}_1^T B_3 \boldsymbol{t}_4 = \hat{\boldsymbol{q}}_1^T F_2 \begin{bmatrix} \boldsymbol{t}_4 \\ 0 \end{bmatrix},$$

the process can continue by generating a matrix $S_k = R_k Q_k = (s_{i,j})$ represented as

$$\begin{aligned} s_{i,j} &= \hat{\boldsymbol{q}}_{\boldsymbol{i}}{}^T F_{i,j}^{\times} \tilde{\boldsymbol{z}}_j + \tilde{u}_i \bar{w}_j, \text{ for } j - i \geq 1 \\ s_{i,i} &= d_i \\ s_{i+1,i} &= \beta_i \\ s_{i,j} &= 0, \text{ for } i - j \geq 2. \end{aligned}$$

The following pseudo-code calculates the parameters of this representation at the cost of $20n$ flops. It also incorporates the procedure stated above for the computation of the diagonal and subdiagonal entries of A_{k+1}.

```
$\hat{g}_1 = d_1 \hat{b}_1 + g_1 \hat{a}_1$; $d_1 = -\bar{\hat{a}}_1 d_1 + \hat{b}_1 g_1$; $\beta_1 = \hat{b}_1 d_2$;
$d_2 = \hat{a}_1 d_2$; $\tilde{w}_1 = -\bar{\hat{a}}_1 \bar{w}_1 + \hat{b}_1 \bar{w}_2$; $\hat{w}_2 = \hat{b}_1 \bar{w}_1 + \hat{a}_1 \bar{w}_2$;
for $j = 2 : n-2$
        $\tilde{g}_{j-1} = \hat{g}_{j-1} - \tilde{u}_{j-1} \hat{w}_j$; $\hat{\boldsymbol{q}}_{j-1}^T = \left[\tilde{\boldsymbol{q}}_{j-1}^T, \tilde{g}_{j-1}\right]$;
        $\tilde{\boldsymbol{z}}_j^T = [\hat{b}_j \boldsymbol{t}_{j+1}^T, -\bar{\hat{a}}_j]$;
        $F_j = [B_{j+1}, \hat{a}_j \boldsymbol{t}_{j+1}; 0, 0, \hat{b}_j]$;
        $\tilde{w}_j = -\bar{\hat{a}}_j \hat{w}_j + \hat{b}_j \bar{w}_{j+1}$; $\hat{w}_{j+1} = \hat{b}_j \hat{w}_j + \hat{a}_j \bar{w}_{j+1}$;
        $\hat{g}_j = d_j \hat{b}_j + g_j \hat{a}_j$; $d_j = -\bar{\hat{a}}_j d_j + \hat{b}_j g_j$;
        $\beta_j = \hat{b}_j d_{j+1}$; $d_{j+1} = \hat{a}_j d_{j+1}$; end
$\tilde{g}_{n-2} = \hat{g}_{n-2} - \tilde{u}_{n-2} \hat{w}_{n-1}$; $\hat{\boldsymbol{q}}_{n-2}^T = [\tilde{\boldsymbol{q}}_{n-2}^T, \tilde{g}_{n-2}]$;
$\tilde{\boldsymbol{z}}_{n-1}^T = [\hat{b}_{n-1} \boldsymbol{t}_n^T, -\bar{\hat{a}}_{n-1}]$;
$F_{n-1} = [I_2, \hat{a}_{n-1} \boldsymbol{t}_n; 0, 0, \hat{b}_{n-1}]$; $\tilde{\boldsymbol{z}}_n^T = [0, 0, 1]$;
$\tilde{w}_{n-1} = -\bar{\hat{a}}_{n-1} \hat{w}_{n-1} + \hat{b}_{n-1} \bar{w}_n$; $\tilde{w}_n = \hat{b}_{n-1} \hat{w}_{n-1} + \hat{a}_{n-1} \bar{w}_n$;
$\hat{g}_{n-1} = d_{n-1} \hat{b}_{n-1} + g_{n-1} \hat{a}_{n-1}$; $d_{n-1} = -\bar{\hat{a}}_{n-1} d_{n-1} + \hat{b}_{n-1} g_{n-1}$;
$\beta_{n-1} = \hat{b}_{n-1} d_n$; $d_n = \hat{a}_{n-1} d_n$; $\tilde{g}_{n-1} = \hat{g}_{n-1} - \tilde{u}_{n-1} \tilde{w}_n$;
$\hat{\boldsymbol{q}}_{n-1}^T = [\boldsymbol{0}^T, \tilde{g}_{n-1}]$;
```

Once the quasiseparable representation for the matrix $R_k Q_k$ is available, then we can specify the entries of $H_{k+1} = (h_{i,j})$ as follows:

$$\begin{aligned} h_{i,j} &= \hat{\boldsymbol{q}}_i^T F_{i,j}^{\times} \tilde{\boldsymbol{z}}_j, \text{ for } j - i \geq 1 \\ h_{i,i} &= d_i + \alpha - \tilde{u}_i \tilde{w}_i \\ h_{i+1,i} &= \beta_i - \tilde{u}_{i+1} \tilde{w}_i \\ h_{i,j} &= -\tilde{u}_i \tilde{w}_j, \text{ for } i - j \geq 2. \end{aligned} \tag{4.5}$$

4.3. Computing a Quasiseparable Representation of H_{k+1}: The Second Method. An appropriate parametrization for the matrix $H_{k+1} = (h_{i,j}^{(k+1)})$ can also be found by performing the matrix multiplications $\widehat{H} = (\hat{h}_{i,j}) = Q_k^H H_k$ and $H_{k+1} = \widehat{H} Q_k$ explicitly. As usual, for the sake of notational simplicity we omit the superscript (k). In particular, from (4.1) we have that the entries of the unitary matrix $H_k = (h_{i,j})$ are given by:

$$\begin{aligned} h_{i,j} &= -u_i \bar{w}_j, \text{ for } i - j \geq 2 \\ h_{j+1,j} &= \beta_j - u_{j+1} \bar{w}_j \\ h_{j,j} &= (\boldsymbol{h})_j \\ h_{i,j} &= \boldsymbol{q}_i{}^T B_{i,j}^{\times} \boldsymbol{t}_j, \text{ for } j - i \geq 1, \end{aligned}$$

where β_j, $1 \leq j \leq n-1$, denote the subdiagonal entries of A_k. Recursions for the entries of $\widehat{H}$ and H_{k+1} can easily be derived by using the factored form for Q_k and are presented below without proofs. This form enables each matrix multiplication to be further reduced to a sequence of elementary steps where only two successive rows or columns are linearly combined.

The entries of $\widehat{H}_k$ have the form:

$$\begin{aligned}
\hat{h}_{i,j} &= \tilde{\boldsymbol{q}}_{\boldsymbol{i}}{}^T B^{\times}_{i+1,j} \boldsymbol{t}_j, \text{ for } j - i \geq 2\\
\hat{h}_{i,i+1} &= \hat{r}_i\\
\hat{h}_{i,i} &= \hat{h}_i\\
\hat{h}_{i,j} &= \tilde{\boldsymbol{p}}_{\boldsymbol{i}}{}^{T\times} L_{i,j} \boldsymbol{s}_j, \text{ for } i - j \geq 1,
\end{aligned}$$

where $\tilde{\boldsymbol{p}}_{\boldsymbol{i}} \in \mathbb{C}^2$, $\boldsymbol{s}_j \in \mathbb{C}^2$ and L_k, $2 \leq k \leq n-1$, are 2×2 matrices in lower triangular form. The vectors $\tilde{\boldsymbol{q}}_{\boldsymbol{i}}$ are returned as output by the procedure for computing the entries of R_k and Q_k and therefore must not be computed again. We denote this by using the symbol %. Recursions for the remaining elements are provided in the next procedure which requires approximately $20n$ flops.

```
for $j = 1 : n$; $\hat{h}_j = h_j = h_{j,j}$; end
for $j = 1 : n-1$; $\hat{\boldsymbol{q}}_j = \boldsymbol{q}_j$; end
$\hat{g}_1 = \hat{b}_1 \hat{h}_1 + \bar{\hat{a}}_1(\beta_1 - u_2 \bar{w}_1)$;
$\hat{h}_1 = -\hat{a}_1 \hat{h}_1 + \hat{b}_1(\beta_1 - u_2 \bar{w}_1)$;
$\hat{r}_1 = -\hat{a}_1 \hat{\boldsymbol{q}}_1^T \boldsymbol{t}_2 + \hat{b}_1 \hat{h}_2$;
$\hat{h}_2 = \hat{b}_1 \hat{\boldsymbol{q}}_1^T \boldsymbol{t}_2 + \bar{\hat{a}}_1 \hat{h}_2$;
% $\tilde{\boldsymbol{q}}_1^T = -\hat{a}_1 \hat{\boldsymbol{q}}_1^T B_2 + \hat{b}_1 \hat{\boldsymbol{q}}_2^T$;
% $\hat{\boldsymbol{q}}_2^T = \hat{b}_1 \hat{\boldsymbol{q}}_1^T B_2 + \bar{\hat{a}}_1 \hat{\boldsymbol{q}}_2^T$;
for $j = 2 : n-2$
        $\hat{g}_j = \hat{b}_j \hat{h}_j + \bar{\hat{a}}_j(\beta_j - u_{j+1} \bar{w}_j)$;
        $\hat{h}_j = -\hat{a}_j \hat{h}_j + \hat{b}_j(\beta_j - u_{j+1} \bar{w}_j)$;
        $\hat{r}_j = -\hat{a}_j \hat{\boldsymbol{q}}_j^T \boldsymbol{t}_{j+1} + \hat{b}_j \hat{h}_{j+1}$;
        $\hat{h}_{j+1} = \hat{b}_j \hat{\boldsymbol{q}}_j^T \boldsymbol{t}_{j+1} + \bar{\hat{a}}_j \hat{h}_{j+1}$;
%       $\tilde{\boldsymbol{q}}_j^T = -\hat{a}_j \hat{\boldsymbol{q}}_j^T B_{j+1} + \hat{b}_j \hat{\boldsymbol{q}}_{j+1}^T$;
%       $\hat{\boldsymbol{q}}_{j+1}^T = \hat{b}_j \hat{\boldsymbol{q}}_j^T B_{j+1} + \bar{\hat{a}}_j \hat{\boldsymbol{q}}_{j+1}^T$;
        $\boldsymbol{s}_{j-1}^T = [-\bar{w}_{j-1}, \hat{g}_{j-1}]$;
        $\tilde{\boldsymbol{p}}_j^T = [\hat{b}_j u_{j+1}, -\hat{a}_j]$;
        $L_j = [1, 0; \bar{\hat{a}}_j u_{j+1}, \hat{b}_j]$;
end
$\hat{g}_{n-1} = \hat{b}_{n-1} \hat{h}_{n-1} + \bar{\hat{a}}_{n-1}(\beta_{n-1} - u_n \bar{w}_{n-1})$;
$\hat{h}_{n-1} = -\hat{a}_{n-1} \hat{h}_{n-1} + \hat{b}_{n-1}(\beta_{n-1} - u_n \bar{w}_{n-1})$;
$\hat{r}_{n-1} = -\hat{a}_{n-1} \hat{\boldsymbol{q}}_{n-1}^T \boldsymbol{t}_n + \hat{b}_{n-1} \hat{h}_n$;
$\hat{h}_n = \hat{b}_{n-1} \hat{\boldsymbol{q}}_{n-1}^T \boldsymbol{t}_n + \bar{\hat{a}}_{n-1} \hat{h}_n$;
$\boldsymbol{s}_{n-2}^T = [-\bar{w}_{n-2}, \hat{g}_{n-2}]$;
$\tilde{\boldsymbol{p}}_{n-1}^T = [\hat{b}_{n-1} u_n, -\hat{a}_{n-1}]$;
$L_{n-1} = [1, 0; \bar{\hat{a}}_{n-1} u_n, \hat{b}_{n-1}]$;
$\tilde{\boldsymbol{p}}_n^T = [0, 1]$;
```

The computation of $H_{k+1} = \widehat{H} Q_k$ proceeds in a very similar way. The entries of $H_{k+1} = (h_{i,j})$ are specified by:

$$\begin{aligned}
h_{i,j} &= \boldsymbol{r}_{\boldsymbol{i}}{}^T F^{\times}_{i,j} \tilde{\boldsymbol{z}}_j, \text{ for } j - i \geq 1\\
h_{i,i} &= \tilde{h}_i\\
h_{i+1,i} &= \tilde{g}_i\\
h_{i,j} &= \tilde{\boldsymbol{p}}_{\boldsymbol{i}}{}^{T\times} L_{i,j+1} \tilde{\boldsymbol{s}}_j, \text{ for } i - j \geq 2,
\end{aligned} \tag{4.6}$$

where $\tilde{\boldsymbol{p}}_{\boldsymbol{i}}, \boldsymbol{s}_i \in \mathbb{C}^2$, $\boldsymbol{r}_i, \tilde{\boldsymbol{z}}_i \in \mathbb{C}^3$, $L_i \in \mathbb{C}^{2\times 2}$ and $F_i \in \mathbb{C}^{3\times 3}$. The next pseudo-code calculates the parameters of this representation of H_{k+1} at the cost of about $35n$ flops .

```
for $j = 1 : n$; $\tilde{h}_j = \hat{h}_j$; end
for $j = 1 : n-1$; $\hat{\boldsymbol{s}}_j = \boldsymbol{s}_j$; end
$\tilde{u}_1 = \hat{b}_1 \tilde{h}_1 + \hat{a}_1 \hat{r}_1$;
$\tilde{h}_1 = -\bar{\hat{a}}_1 \tilde{h}_1 + \hat{b}_1 \hat{r}_1$;
$\tilde{g}_2 = -\bar{\hat{a}}_1 \tilde{\boldsymbol{p}}_2^T \hat{\boldsymbol{s}}_1 + \hat{b}_1 \tilde{h}_2$;
$\tilde{h}_2 = \hat{b}_1 \tilde{\boldsymbol{p}}_2^T \hat{\boldsymbol{s}}_2 + \hat{a}_1 \tilde{h}_2$;
$\tilde{\boldsymbol{s}}_1 = -\bar{\hat{a}}_1 L_2 \hat{\boldsymbol{s}}_1 + \hat{b}_1 \hat{\boldsymbol{s}}_2$;
$\hat{\boldsymbol{s}}_2 = \hat{b}_1 L_2 \hat{\boldsymbol{s}}_1 + \hat{a}_1 \hat{\boldsymbol{s}}_2$;
for $j = 2 : n-2$
        $\tilde{u}_j = \hat{b}_j \tilde{h}_j + \hat{a}_j \hat{r}_j$;
        $\tilde{h}_j = -\bar{\hat{a}}_j \tilde{h}_j + \hat{b}_j \hat{r}_j$;
        $\tilde{g}_{j+1} = -\bar{\hat{a}}_j \tilde{\boldsymbol{p}}_{j+1}^T \hat{\boldsymbol{s}}_j + \hat{b}_j \tilde{h}_{j+1}$;
        $\tilde{h}_{j+1} = \hat{b}_j \tilde{\boldsymbol{p}}_{j+1}^T \hat{\boldsymbol{s}}_{j+1} + \hat{a}_j \tilde{h}_{j+1}$;
        $\tilde{\boldsymbol{s}}_j = -\bar{\hat{a}}_j L_{j+1} \hat{\boldsymbol{s}}_j + \hat{b}_j \hat{\boldsymbol{s}}_{j+1}$;
        $\hat{\boldsymbol{s}}_{j+1} = \hat{b}_j L_{j+1} \hat{\boldsymbol{s}}_j + \hat{a}_j \hat{\boldsymbol{s}}_{j+1}$;
        $\boldsymbol{r}_{j-1}^T = [\tilde{\boldsymbol{q}}_{j-1}^T, \tilde{u}_{j-1}]$;
        $\tilde{\boldsymbol{z}}_j^T = [\hat{b}_j \boldsymbol{t}_{j+1}^T, -\bar{\hat{a}}_j]$;
        $F_j = [B_{j+1}, \hat{a}_j \boldsymbol{t}_{j+1}; 0, 0, \hat{b}_j]$;
end
$\tilde{u}_{n-1} = \hat{b}_{n-1} \tilde{h}_{n-1} + \hat{a}_{n-1} \hat{r}_{n-1}$;
$\tilde{h}_{n-1} = -\bar{\hat{a}}_{n-1} \tilde{h}_{n-1} + \hat{b}_{n-1} \hat{r}_{n-1}$;
$\tilde{g}_n = -\bar{\hat{a}}_{n-1} \tilde{\boldsymbol{p}}_n^T \hat{\boldsymbol{s}}_{n-1} + \hat{b}_{n-1} \tilde{h}_n$;
$\tilde{h}_n = \hat{b}_{n-1} \tilde{\boldsymbol{p}}_n^T \hat{\boldsymbol{s}}_n + \hat{a}_{n-1} \tilde{h}_n$;
$\boldsymbol{r}_{n-2}^T = [\tilde{\boldsymbol{q}}_{n-2}^T, \tilde{u}_{n-2}]$;
$\tilde{\boldsymbol{z}}_{n-1}^T = [\hat{b}_{n-1} \boldsymbol{t}_n^T, -\bar{\hat{a}}_{n-1}]$;
$F_{n-1} = [I_2, \hat{a}_{n-1} \boldsymbol{t}_n; 0, 0, \hat{b}_{n-1}]$;
$\tilde{\boldsymbol{z}}_n^T = [0, 0, 1]$;
```

It is worth noting that the arithmetic cost of the second method is about twice the one of the first method. We have performed numerical experiments to compare the stability features of the two methods. The results of our experience are discussed in the next section.

4.4. Compressing the Quasiseparable Representation of $\boldsymbol{H_{k+1}}$. Both the first and the second method provide a representation for the matrix H_{k+1} which is generally redundant and must be further compressed. In this subsection we describe how the compression can be carried out in a fast way. Due to the upper Hessenberg form of A_{k+1}, from (3.12) it follows that $h_{i,j}^{(k+1)} = -u_i^{(k+1)} \bar{w}_j^{(k+1)}$ for $i \geq j+2$. Then the unitary matrix H_{k+1} satisfies the hypotheses of Theorem 3.2 and, therefore, its strictly upper triangular part can be reconstructed as in Theorem 3.5 from the coefficients of the Givens rotation matrices employed in the process of computing a QR factorization of H_{k+1}. More specifically, suppose that the entries of H_{k+1} are specified by:

$$
\begin{aligned}
&h_{i,j} = \boldsymbol{r_i}^T F_{i,j}^{\times} \tilde{\boldsymbol{z}}_j, \text{ for } j - i \geq 1 \\
&h_{i,i} = \tilde{h}_i \\
&h_{i+1,i} = \tilde{g}_i \\
&h_{i,j} = -u_i \bar{w}_j, \text{ for } i - j \geq 2,
\end{aligned}
\tag{4.7}
$$

where u_i and w_j denote the entries of $\boldsymbol{u}^{(k+1)}$ and $\boldsymbol{w}^{(k+1)}$, respectively. According to the approach followed in the proof of Theorem 3.2, H_{k+1} is first reduced to upper

Hessenberg form by applying a sequence of $2n-5$ Givens rotations suitably chosen to annihilate the entries in the left-bottom corner. The transformed matrix is still unitary and, therefore, its QR factorization gives the Schur and the complementary parameters of its structured representation (3.2) (compare with Remark 3.1). As described in Theorem 3.5, these parameters can finally be combined with the coefficients of the previously determined Given rotations in order to define the structure of the upper triangular part of H_{k+1}.

At the beginning the matrix H_{k+1} is transformed into a matrix $\tilde{P}$ of lower bandwidth 2 by applying a sequence of Givens rotations $G_{n-1}(\tilde{a}_{n-1}),\dots,G_3(\tilde{a}_3)$, $G_2(\tilde{a}_2), G_1(\tilde{a}_1)$. Observe that $G_2(\tilde{a}_2)=G_1(\tilde{a}_1)=\mathrm{diag}[1,-1,1\dots,1]$ are only introduced for notational convenience since $G_1(\tilde{a}_1)G_1(\tilde{a}_2)=I_n$. The nonzero entries of the matrix $\tilde{P}=(\tilde{p}_{i,j})$ are specified by:

$$\begin{aligned}\tilde{p}_{i,j} &= \tilde{\boldsymbol{p}}_{\boldsymbol{i}}{}^T E^{\times}_{i-1,j}\widehat{\boldsymbol{z}}_j, \text{ for } j-i\geq 0\\ \tilde{p}_{i+1,i} &= q_i\\ \tilde{p}_{i+2,i} &= r_i.\end{aligned}$$

The first Givens rotation $G_{n-1}(\tilde{a}_{n-1})$ is such that

$$\mathcal{G}_{n-1}(\tilde{a}_{n-1})\begin{bmatrix} u_{n-1}\\ u_n\end{bmatrix}=\begin{bmatrix}\hat{u}_{n-1}\\ 0\end{bmatrix}.$$

Similarly, choose the successive rotations $\mathcal{G}(\tilde{a}_j)$, $3\leq j\leq n-2$, to yield

$$\mathcal{G}(\tilde{a}_j)\begin{bmatrix} u_j\\ \hat{u}_{j+1}\end{bmatrix}=\begin{bmatrix}\hat{u}_j\\ 0\end{bmatrix}, j=n-2,\dots,3.$$

Since

$$\tilde{h}_n=[\mathbf{0}^T,1]\begin{bmatrix}\tilde{\boldsymbol{z}}_n\\ \tilde{h}_n\end{bmatrix},\quad \boldsymbol{r_{n-1}}^T\tilde{\boldsymbol{z}}_n=[\boldsymbol{r_{n-1}}^T,0]\begin{bmatrix}\tilde{\boldsymbol{z}}_n\\ \tilde{h}_n\end{bmatrix},$$

we have

$$\mathcal{G}_{n-1}(\tilde{a}_{n-1})H_{k+1}[n-1:n,n-2:n]=\begin{bmatrix}\widehat{g}_{n-2} & \widehat{h}_{n-1} & \widehat{\boldsymbol{p}}_{\boldsymbol{n-1}}{}^T\widehat{\boldsymbol{z}}_n\\ r_{n-2} & q_{n-1} & \tilde{\boldsymbol{p}}_{\boldsymbol{n}}{}^T\widehat{\boldsymbol{z}}_n\end{bmatrix},$$

where

$$\mathcal{G}_{n-1}(\tilde{a}_{n-1})\begin{bmatrix}\boldsymbol{r_{n-1}}^T & 0\\ \mathbf{0}^T & 1\end{bmatrix}=\begin{bmatrix}\widehat{\boldsymbol{p}}_{\boldsymbol{n-1}}{}^T\\ \tilde{\boldsymbol{p}}_{\boldsymbol{n}}{}^T\end{bmatrix},\quad \widehat{\boldsymbol{z}}_n=\begin{bmatrix}\tilde{\boldsymbol{z}}_n\\ \tilde{h}_n\end{bmatrix}.$$

Observe that

$$\widehat{\boldsymbol{p}}_{\boldsymbol{n-1}}{}^T\widehat{\boldsymbol{z}}_n=[\mathbf{0}^T,1]\left[\begin{array}{c|c} F_{n-1} & \mathbf{0}\\ \hline \multicolumn{2}{c}{\widehat{\boldsymbol{p}}_{\boldsymbol{n-1}}{}^T}\end{array}\right]\widehat{\boldsymbol{z}}_n=[\mathbf{0}^T,1]\,E_{n-1}\widehat{\boldsymbol{z}}_n$$

and

$$h_{n-2,n}=\boldsymbol{r_{n-2}}^T F_{n-1}\tilde{\boldsymbol{z}}_n=[\boldsymbol{r_{n-2}}^T,0]\,E_{n-1}\widehat{\boldsymbol{z}}_n.$$

The reduction process can thus continue until the matrix is converted into the desired form. The last two rotations are just used to make the representation of the whole matrix uniform. The following pseudo-code describes the process which requires approximately $20n$ flops.

```
for $j = 1 : n$; $\hat{h}_j = \tilde{h}_j$; $\hat{u}_j = u_j$ end
for $j = 1 : n-1$; $\hat{g}_j = \tilde{g}_j$; end
for $j = n-1 : -1 : 3$
        $(\tilde{a}_j, \tilde{b}_j) = Givens(\hat{u}_j, \hat{u}_{j+1})$;
        $\hat{u}_j = -\tilde{a}_j \hat{u}_j + \tilde{b}_j \hat{u}_{j+1}$;
        $q_j = \tilde{b}_j \hat{h}_j + \bar{\tilde{a}}_j \hat{g}_j$;
        $r_{j-1} = \tilde{b}_j \hat{g}_{j-1} - \bar{\tilde{a}}_j \hat{u}_{j+1} \bar{w}_{j-1}$;
        $\hat{g}_{j-1} = -\tilde{a}_j \hat{g}_{j-1} - \tilde{b}_j \hat{u}_{j+1} \bar{w}_{j-1}$;
        $\hat{h}_j = -\tilde{a}_j \hat{h}_j + \tilde{b}_j \hat{g}_j$;
        $\widehat{\boldsymbol{z}}_{j+1}^T = \left[\tilde{\boldsymbol{z}}_{j+1}^T, \hat{h}_{j+1}\right]$;
        $\tilde{\boldsymbol{p}}_{j+1}^T = \left[\tilde{b}_j \boldsymbol{r}_j^T, \bar{\tilde{a}}_j\right]$;
        $\widehat{\boldsymbol{p}}_j^T = \left[-\tilde{a}_j \boldsymbol{r}_j^T, \tilde{b}_j\right]$;
        $E_j = [F_j, \mathbf{0}; \widehat{\boldsymbol{p}}_j^T]$;
end
$q_2 = \hat{g}_2$; $r_1 = -\hat{u}_3 \bar{w}_1$; $\hat{g}_1 = -\hat{g}_1$; $\hat{h}_2 = -\hat{h}_2$;
$\widehat{\boldsymbol{z}}_3^T = \left[\tilde{\boldsymbol{z}}_3^T, \hat{h}_3\right]$; $\tilde{\boldsymbol{p}}_3^T = [\mathbf{0}^T, 1]$;
$\widehat{\boldsymbol{p}}_2^T = [-\boldsymbol{r}_2^T, 0]$; $E_2 = [F_2, \mathbf{0}; \widehat{\boldsymbol{p}}_2^T]$;
$q_1 = -\hat{g}_1$; $\widehat{\boldsymbol{z}}_2^T = \left[\tilde{\boldsymbol{z}}_2^T, \hat{h}_2\right]$; $\tilde{\boldsymbol{p}}_2^T = [\mathbf{0}^T, -1]$;
$\widehat{\boldsymbol{p}}_1^T = [\boldsymbol{r}_1^T, 0]$; $E_1 = [I_3, \mathbf{0}; \widehat{\boldsymbol{p}}_1^T]$;
$\widehat{\boldsymbol{z}}_1^T = \left[\mathbf{0}^T, \hat{h}_1\right]$; $\tilde{\boldsymbol{p}}_1^T = [\mathbf{0}^T, 1]$;
```

The matrix $\tilde{P}$ can be easily reduced to a Hessenberg form $\widehat{P}$ by means of a sequence of $n-2$ Givens rotations $G_2(\hat{a}_2), \ldots, G_{n-1}(\hat{a}_{n-1})$ suitably chosen to annihilate the second lower subdiagonal. We find that $\widehat{P} = (\widehat{p}_{i,j}) = G_{n-1}(\hat{a}_{n-1}) \cdots G_2(\hat{a}_2)\tilde{P}$ can be represented as:

$$\begin{aligned}
&\widehat{p}_{i,j} = \widehat{\boldsymbol{p}}_{\boldsymbol{i}}{}^T E_{i,j}^{\times} \widehat{\boldsymbol{z}}_j, \text{ for } j - i \geq 1 \\
&\widehat{p}_{i,i} = a_i \\
&\widehat{p}_{i+1,i} = b_i.
\end{aligned}$$

Recursions for the elements of this representation are given below. The cost is about $35n$ flops.

```
$\widehat{\boldsymbol{p}}_{\boldsymbol{1}}{}^T = \tilde{\boldsymbol{p}}_1^T E_1$ $a_1 = \tilde{\boldsymbol{p}}_1^T \widehat{\boldsymbol{z}}_1$;
for $j = 2 : n-1$
        $(\hat{a}_j, \hat{b}_j) = Givens(q_{j-1}, r_{j-1})$;
        $b_{j-1} = -\hat{a}_j q_{j-1} + \hat{b}_j r_{j-1}$;
        $a_j = -\hat{a}_j \tilde{\boldsymbol{p}}_j^T \widehat{\boldsymbol{z}}_j + \hat{b}_j q_j$;
        $q_j = \hat{b}_j \tilde{\boldsymbol{p}}_j^T \widehat{\boldsymbol{z}}_j + \bar{\hat{a}}_j q_j$;
        $\widehat{\boldsymbol{p}}_{\boldsymbol{j}}{}^T = -\hat{a}_j \tilde{\boldsymbol{p}}_j^T E_j + \hat{b}_j \tilde{\boldsymbol{p}}_{j+1}^T$;
        $\tilde{\boldsymbol{p}}_{j+1}^T = \hat{b}_j \tilde{\boldsymbol{p}}_j^T E_j + \bar{\hat{a}}_j \tilde{\boldsymbol{p}}_{j+1}^T$;
end
$\widehat{\boldsymbol{p}}_{\boldsymbol{n}}{}^T = \tilde{\boldsymbol{p}}_n^T$; $a_n = \tilde{\boldsymbol{p}}_n^T \widehat{\boldsymbol{z}}_n$; $b_{n-1} = q_{n-1}$;
```

Finally, the matrix $\widehat{P}$ can be converted into the identity matrix of order n by applying n modified rotations of the form

$$\begin{aligned}
&G_j(a'_j) = I_{j-1} \oplus \begin{bmatrix} -a'_j & b'_j \\ \bar{b}'_j & \bar{a}'_j \end{bmatrix} \oplus I_{n-j-1}, \quad 1 \leq j \leq n-1, \quad |a'_j|^2 + |b'_j|^2 = 1, \\
&G_n(a'_n) = I_{n-1} \oplus (-a'_n), \quad |a'_n| = 1,
\end{aligned}$$

chosen in such a way that $G_n(a'_n)G_{n-1}(a'_{n-1})\cdots G_1(a'_1)\widehat{P}$ is an upper triangular matrix with nonnegative diagonal entries. The following pseudo-code computes the coefficients a'_j and b'_j, $j = 1, \dots, n-1$, at the cost of $20n$ flops.

```
for $j = 1 : n - 2$
        $r = \sqrt{|a_j|^2 + |b_j|^2}$;
        $a'_j = -\bar{a}_j/r$; $b'_j = \bar{b}_j/r$;
        $a_{j+1} = \bar{b}'_j \widehat{\boldsymbol{p}}_j^T \widehat{z}_{j+1} + \bar{a}'_j a_{j+1}$;
        $\widehat{\boldsymbol{p}}_{j+1}^T = \bar{b}'_j \widehat{\boldsymbol{p}}_j^T E_{j+1} + \bar{a}'_j \widehat{\boldsymbol{p}}_{j+1}^T$;
end
$r = \sqrt{|a_{n-1}|^2 + |b_{n-1}|^2}$; $a'_{n-1} = -\bar{a}_{n-1}/r$; $b'_{n-1} = \bar{b}_{n-1}/r$;
$a_n = \bar{b}'_{n-1} \widehat{\boldsymbol{p}}_{n-1}^T \widehat{z}_n + \bar{a}'_{n-1} a_n$; $a'_n = -\bar{a}_n/|a_n|$;
```

The coefficients a'_j and b'_j define the parameters ϕ_j and ψ_j used in Theorem 3.5 to describe the structure in the strictly upper triangular part of the matrix P such that

$$P^H = H_{k+1}^H = G_{n-1}(a'_{n-1})\cdots G_1(a'_1)G_{n-1}(\hat{a}_{n-1})\cdots G_2(\hat{a}_2)G_3(\tilde{a}_1)\cdots G_{n-1}(\tilde{a}_{n-1}).$$

More specifically, we have $\phi_j = \bar{a}'_j$ and $\psi_j = b'_j$ The matrix H_{k+1} can thus be reconstructed in the desired form using the procedure given in Theorem 3.5 at the cost of approximately $20n$ flops.

4.5. Deflation After a Step of the QR-algorithm. When an eigenvalue has been approximated with sufficiently high precision, a deflation technique is generally employed before proceeding with the computation of the remaining eigenvalues. Suppose that after k iterations of the QR-algorithm the matrix A_k has the form

$$A_k = \begin{bmatrix} A_k^{(1)} & \star \\ \bigcirc & A_k^{(2)} \end{bmatrix},$$

where $A_k^{(1)} \in \mathbb{C}^{(n-s)\times(n-s)}$ is upper Hessenberg whereas numerically $A_k^{(2)} \in \mathbb{C}^{s\times s}$ can be considered upper triangular so that its diagonal entries provide approximations for the first s eigenvalues of the matrix $A = A_0$. The remaining eigenvalues of A are eigenvalues of $A_k^{(1)}$, hence we might economize by performing subsequent iterations on the smaller matrix. The process is called *deflation* and continuing in this fashion, by operating on smaller and smaller matrices, we may approximate all the eigenvalues of A.

The matrix H_k is partitioned accordingly with A_k as follows:

$$H_k = \begin{bmatrix} H_k^{(1)} & \star \\ -\boldsymbol{u}_k^{(2)}\boldsymbol{w}_k^{(1)^H} + \beta_{n-s}^{(k)}\boldsymbol{e}_1\boldsymbol{e}_s^T & H_k^{(2)} \end{bmatrix},$$

where $\boldsymbol{u}_k^{(2)}$ is formed from the last s components of the vector $\boldsymbol{u}_k$, $\boldsymbol{w}_k^{(1)}$ is defined by the first $n-s$ elements of $\boldsymbol{w}_k$ and $\boldsymbol{e}_1$ and $\boldsymbol{e}_s$ are the first and the last column of the matrices I_{n-s} and I_s, respectively. It is worth noting that $\boldsymbol{u}_k^{(2)} = \boldsymbol{u}_{k+j}^{(2)}$ and $\beta_{n-s}^{(k)} = \beta_{n-s}^{(k+j)}$ for $j = 0, 1, \dots$, and, therefore, we can determine a unitary upper Hessenberg matrix $U_s \in \mathbb{C}^{s\times s}$ such that $U_s\boldsymbol{u}_{k+j}^{(2)} = \left[\tilde{u}_{n-s+1}, \boldsymbol{0}^T\right]^T$, $j = 0, 1, \dots$. The computation of U_s given the matrix U_{s-1} requires $0(1)$ flops only. We find that

$$(I_{n-s} \oplus U_s)H_k(I_{n-s} \oplus U_s)^H = \begin{bmatrix} H_k^{(1)} & \star \\ -u_{n-s+1}\boldsymbol{e}_1\boldsymbol{w}_k^{(1)^H} + \beta_{n-s}^{(k)}U_s\boldsymbol{e}_1\boldsymbol{e}_s^T & \star \end{bmatrix}.$$

Then the procedure stated in Theorem 3.5 allows one to reconstruct the quasiseparable structure in the strictly upper triangular part of $A_k^{(1)}$ by means of the coefficients of the (modified) Givens rotations applied for the computation of a QR factorization of the $(n-s+1)\times(n-s)$ matrix

$$\left[\begin{array}{c} H_k^{(1)} \\ \hline -u_{n-s+1}\boldsymbol{w}_k^{(1)^H} + \tilde{\beta}_{n-s}^{(k)}\boldsymbol{e}_s^T \end{array}\right].$$

5. Experimental Issues. In this section we design a practical implementation of the algorithm described above and present the results of extensive numerical experiments.

5.1. The Practical Algorithm. A set of preliminary tests has been carried out to verify the numerical behavior of the *structured QR iteration* in the basic form stated at the beginning of the previous section. The choice of the method used to compute the structural representation of H_{k+1} at step 2 seems to play no role for the accuracy of the results. On the contrary, the quality of the computed approximations strongly depends on the properties of the matrix H_{k+1}. If this matrix remains numerically unitary as the iterative process goes on, then the computed eigenvalues are generally accurate. Otherwise, whenever there is a deterioration of the property of being unitary a loss of accuracy is observed and this loss propagates to the subsequent steps. To avoid the possible accumulation of errors in the computation of the structural representation of H_{k+1} we proceed as follows. Assume that H_{k+1} is a small perturbation of a unitary matrix . Its QR factorization is $H_{k+1} = U_{k+1}S_{k+1}$, where U_{k+1} is unitary and $S_{k+1} = I_n + \Delta$ is upper triangular. Hence, from $A_{k+1} = H_{k+1} + \boldsymbol{u}_{k+1}\boldsymbol{w}_{k+1}^H$ we obtain $A_{k+1}S_{k+1}^{-1} = U_{k+1} + \boldsymbol{u}_{k+1}\boldsymbol{w}_{k+1}^H S_{k+1}^{-1}$. The matrix $A_{k+1}S_{k+1}^{-1}$ is still upper Hessenberg whereas U_{k+1} is numerically unitary. The entries of the corrected vector $\boldsymbol{w}_{k+1}S_{k+1}^{-1}$ are determined easily from the entries in the second subdiagonal of U_{k+1} without explicitly computing the matrix S_{k+1}. Therefore, $A_{k+1}S_{k+1}^{-1}$ has two very important features: It is a Hessenberg matrix which is a rank-one modification of a unitary matrix and, at the same time, it is a small perturbation of A_{k+1}. Whence, we define $A_{k+1} := U_{k+1} + \boldsymbol{u}_{k+1}\tilde{\boldsymbol{w}}_{k+1}^H$, where $\tilde{\boldsymbol{w}}_{k+1}^H = \boldsymbol{w}_{k+1}^H S_{k+1}^{-1}$. Roughly speaking, this means that at each QR iteration the current iterate is slightly perturbed in such a way to preserve all the structures of the considered eigenproblem. The process is summarized below.

1. Compute Q_k and R_k such that $A_k - \alpha_k I_n = Q_k R_k$ provides a QR factorization of the left-hand side matrix.
2. Determine the matrix $A_{k+1} :== R_k Q_k + \alpha_k I_n$ and the vectors $\boldsymbol{u}_{k+1} := Q_k^H \boldsymbol{u}_k$, $\boldsymbol{w}_{k+1} := Q_k^H \boldsymbol{w}_k$.
3. Determine the unitary matrix $H_{k+1} = A_{k+1} - \boldsymbol{u}_{k+1}\boldsymbol{w}_{k+1}^H$.
4. Compute the unitary matrix U_{k+1} such that $H_{k+1} = U_{k+1}S_{k+1}$ is a QR factorization of H_{k+1} and S_{k+1} is upper triangular with real positive diagonal entries.
5. Find the vector $\tilde{\boldsymbol{w}}_{k+1}$.
6. Set $A_{k+1} := U_{k+1} + \boldsymbol{u}_{k+1}\tilde{\boldsymbol{w}}_{k+1}^H$.

This algorithm has been implemented in MATLAB and then applied to the computation of the eigenvalues of companion matrices of both small and large size. The results of extensive numerical experiments confirm the robustness and the efficiency of the proposed approach.

Our implementation requires approximately $180\,n+O(1)$ flops per iteration. The main program incorporates the following shifting strategy suggested in [[23], p. 549]. At the beginning the shift parameter σ is equal to zero. If $A_s = (a_{i,j}^{(s)}) \in \mathbb{C}^{n\times n}$ satisfies

$$|a_{n,n}^{(s-1)} - a_{n,n}^{(s)}| \le 0.1|a_{n,n}^{(s-1)}|, \tag{5.1}$$

then we apply non-zero shifts by setting $\sigma_k = a_{n,n}^{(k)}$, $k = s, s+1, \ldots$. We say that $a_{n,n}^{(k)}$ provides a numerical approximation of an eigenvalue λ of A_0 whenever

$$|\beta_n^{(k)}| \le eps\ (|a_{n,n}^{(k)}| + |a_{n-1,n-1}^{(k)}|,$$

where eps is the machine precision, i.e., $eps \simeq 2.2\cdot 10^{-16}$. If this condition is fulfilled, then we set $\lambda = a_{n,n}^{(k)}$ and deflate the matrix.

After non-zero shifting has begun, we check for the convergence of the last diagonal entries of the currently computed iterate A_k. If convergence fails to occur after 15 iterations, then at the 16-th iteration we set $\sigma_k = 1.5\ (|a_{n,n}^{(k)}| + |\beta_n^{(k)}|)$ and continue with non-zero shifting. If $a_{n,n}^{(k)}$ does not converge in the next 15 iterations, then the program reports failure. In our experience such failure has been never encountered.

5.2. Numerical Experiments. We tested companion matrices associated with the following polynomials:

1. the scaled "Wilkinson polynomial": $p(z) = \prod_{k=1}^{n}(z - k/n)$;
2. the monic polynomial with zeros equally spaced on the curve $z = x+\mathtt{i}\sin(\pi x)$, $-1 \le x \le 1$, namely $p(z) = \prod_{k=-n/2}^{n/2-1}(z - \frac{2(k+0.5)}{n-1} - \mathtt{i}\sin(\frac{2(k+0.5)}{n-1}))$;
3. the polynomial $p(z) = z^n - 1$ with zeros equispaced on the unit circle;
4. the random polynomial $p(z) = \sum_{j=0}^{n-1} a_j z^j + z^n$ with $a_j = \text{rand} + \mathtt{i}\ \text{rand}$, $j = 0, \ldots, n-1$, where rand is a pseudo–random number uniformly distributed in $[0,1]$;
5. the random polynomial $p(z) = \sum_{j=0}^{n-1} a_j z^j + z^n$, where $a_j = a_{1,j}\ \times 10^{e_{1,j}} + \mathtt{i}\ a_{2,j}\times 10^{e_{2,j}}$, $a_{i,j} = \text{rand}+\mathtt{i}\ \text{rand}$, $e_{i,j} = 5\times(\text{rand}-0.5)+\mathtt{i}\ 5\times(\text{rand}-0.5)$, $i = 1,2$, $j = 0,\ldots,n-1$.

Let C be the companion matrix associated with the polynomial $p(z)$. An estimate of the maximum error expected in the computation of the eigenvalues of C by using the shifted QR eigenvalue algorithm without balancing is $eps\max(\mathtt{condeig}(C))\cdot \parallel C \parallel$, where the MATLAB function `condeig` is used to approximate the eigenvalue condition numbers. This means that $(rcond)^{-1} = \max(\mathtt{condeig}(C))\cdot \parallel C \parallel$ is such that the base 10 logarithm of $(rcond)^{-1}$ provides an estimate of the number of digits of accuracy which can be lost during the computation (see for instance [23]). Tables 5.1, 5.2 and 5.3 show the results of our numerical experiments for the polynomials from 1) to 3) by reporting the degree n, the value of $rcond$ and the maximum absolute error of the computed roots. Table 5.3 also reports the average number it of QR iterations per eigenvalue. The total number of iterations performed to compute all the eigenvalues is less than $6n$. The quantity $rcond$ is returned by our program as output. Figures 5.1 and 5.2 cover our tests with random polynomials of the form 4–5) of high degree. Each figure shows the $error$ and the value of $rcond$ for polynomials of degree $n(m) = 2^{2+m}$ for $m = 1,\ldots,9$. For each size we carried out 100 numerical experiments and report the average values. The $error$ measures the distance between the set of the computed eigenvalues and the set of the eigenvalues returned by the function `eig` with the same input data. Let $\lambda(A)$ denote the set of eigenvalues computed by the MATLAB

	Scaled Wilkinson polynomial	
n	rcond	err
4	2.7e-03	6.4e-15
8	4.1e-07	2.0e-11
16	3.1e-15	5.4e-05

TABLE 5.1

	$p(z) = \prod_{k=-n/2}^{n/2-1}(z - \frac{2(k+0.5)}{n-1} - \mathtt{i}\sin(\frac{2(k+0.5)}{n-1}))$	
n	rcond	err
8	1.0e-02	8.4e-15
16	4.6e-06	2.8e-12
32	2.8e-13	9.9e-07

TABLE 5.2

function `eig`. Let $\tilde{\lambda}(A)$ denote the set of eigenvalues computed by our algorithm, and define the distance between the sets $\lambda(A)$ and $\tilde{\lambda}(A)$ by

$$\mathrm{dist}(\lambda(A), \tilde{\lambda}(A)) = \max\{\max_{\tilde{\lambda}\in\tilde{\lambda}(A)} \parallel \tilde{\lambda} - \lambda(A) \parallel, \max_{\lambda\in\lambda(A)} \parallel \lambda - \tilde{\lambda}(A) \parallel\},$$

where $\parallel \lambda - \tilde{\lambda}(A) \parallel = \min_{\tilde{\lambda}\in\tilde{\lambda}(A)} |\lambda - \tilde{\lambda}|$. We refer to this distance as the error in the eigenvalues computed by our algorithm. Therefore, we tacitly assume that the MATLAB function `eig` computes the eigenvalues exactly.

6. Conclusion.

In this paper we have presented a novel QR eigenvalue algorithm for a class of Hessenberg matrices which are rank-one perturbations of unitary matrices. The method is appealing because of its low memory requirements and low computational cost. In fact, the exploitation of the quasiseparable structure of the associated eigenvalue problems leads to a $O(n^2)$ complexity algorithm requiring only $O(n)$ memory space. The results of extensive numerical experiments confirm the robustness and the effectiveness of the proposed approach. The accuracy of computed results is generally in accordance with the estimates on the conditioning of the input matrix.

REFERENCES

[1] G. S. Ammar, D. Calvetti, W. B. Gragg, and L. Reichel. Polynomial zerofinders based on Szegő polynomials. *J. Comput. Appl. Math.*, 127(1-2):1–16, 2001. Numerical analysis 2000, Vol. V, Quadrature and orthogonal polynomials.

[2] G. S. Ammar, D. Calvetti, and L. Reichel. Continuation methods for the computation of zeros of Szegő polynomials. *Linear Algebra Appl.*, 249:125–155, 1996.

[3] G. S. Ammar, W. B. Gragg, and L. Reichel. Downdating of Szegő polynomials and data-fitting applications. *Linear Algebra Appl.*, 172:315–336, 1992. Second NIU Conference on Linear Algebra, Numerical Linear Algebra and Applications (DeKalb, IL, 1991).

[4] D. Bindel, J. Demmel, W. Kahan, and O. Marques. On computing Givens rotations reliably and efficiently. *ACM Trans. Math. Software*, 28:206–238, 2002.

[5] D. A. Bini, F. Daddi, and L. Gemignani. On the shifted *QR* iteration applied to Frobenius matrices. *Electron. Trans. Numer. Anal.*, 18:137–152, 2004.

[6] D. Calvetti, S. Kim, and L. Reichel. The restarted *QR*-algorithm for eigenvalue computation of structured matrices. *J. Comput. Appl. Math.*, 149:415–422, 2002.

	$p(z) = z^n - 1$		
n	rcond	err	it
128	1.0	5.2e-15	5.86
256	1.0	9.1e-15	5.91
512	1.0	1.7e-14	5.60

TABLE 5.3

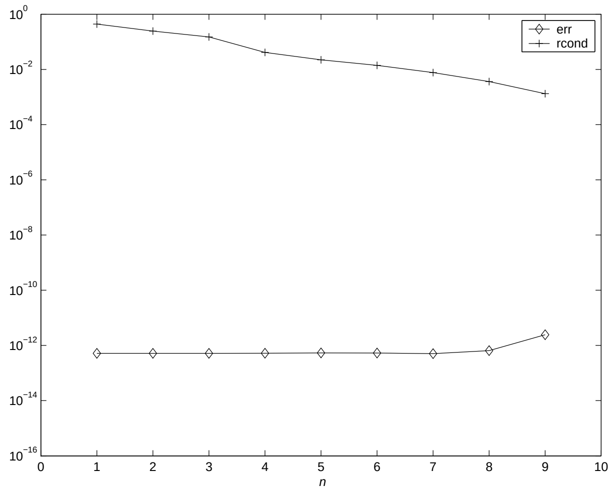

FIG. 5.1. *Random polynomials of kind 4) of degree $n(m) = 2^{2+m}$, $1 \leq m \leq 9$.*

[7] P. Dewilde and A. J. van der Veen. *Time-varying systems and computations.* Kluwer Academic Publishers, Boston, MA, 1998.

[8] Y. Eidelman and I. Gohberg. On a new class of structured matrices. *Integral Equations Operator Theory*, 34:293–324, 1999.

[9] Y. Eidelman and I. Gohberg. Linear complexity inversion algorithms for a class of structured matrices. *Integral Equations Operator Theory*, 35:28-52, 1999.

[10] Y. Eidelman and I. Gohberg. Fast inversion algorithms for a class block structured matrices, *Contemporary Mathematics*, 281:17-38 (2001).

[11] Y. Eidelman and I. Gohberg. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra Appl.*, 343-344:419–450, 2002.

[12] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations.* John Hopkins University Press, 1996.

[14] W. B. Gragg. The QR algorithm for unitary Hessenberg matrices. *J. Comput. Appl. Math.*, 16:1–8, 1986.

[15] W. B. Gragg. Stabilization of the uhqr-algorithm. In *Advances in computational mathematics (Guangzhou, 1997)*, volume 202 of *Lecture Notes in Pure and Appl. Math.*, pages 139–154. Dekker, New York, 1999.

[16] M. S. Moonen, G. H. Golub, and B. L. R. De Moor, editors. *Direct and inverse unitary eigenvalue problems in signal processing: an overview*, volume 232 of *NATO Advanced Science Institutes Series E: Applied Sciences*, Dordrecht, 1993. Kluwer Academic Publishers Group.

[17] B. N. Parlett. *The symmetric eigenvalue problem.* Society for Industrial and Applied Mathematics (SIAM), 1998.

[18] L. Reichel, G. S. Ammar, and W. B. Gragg. Discrete least squares approximation by trigonometric polynomials. *Math. Comp.*, 57(195):273–289, 1991.

[19] M. Stewart. Stability properties of several variants of the unitary Hessenberg QR algorithm. In *Structured matrices in mathematics, computer science, and engineering, II (Boulder, CO, 1999)*, volume 281 of *Contemp. Math.*, pages 57–72. Amer. Math. Soc., Providence,

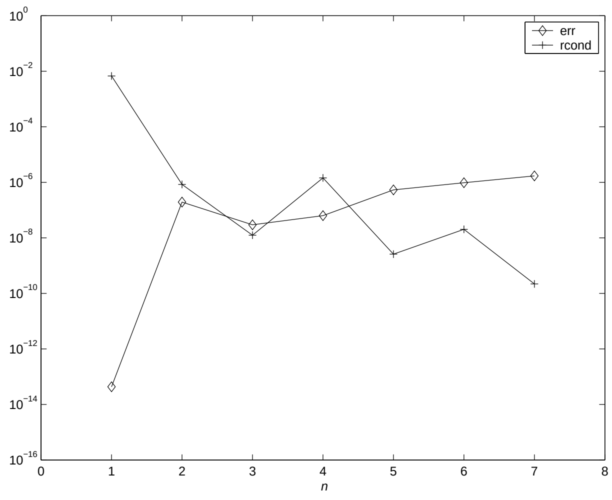

Fig. 5.2. *Random polynomials of kind 5) of degree* $n(m) = 2^{2+m}$, $1 \leq m \leq 7$.

RI, 2001.

[20] E. E. Tyrtyshnikov. Mosaic ranks for weakly semiseparable matrices. In *Large-scale scientific computations of engineering and environmental problems, II (Sozopol, 1999)*, volume 73 of *Notes Numer. Fluid Mech.*, pages 36–41. Vieweg, Braunschweig, 2000.

[21] T. L. Wang and W. B. Gragg. Convergence of the shifted QR algorithm for unitary Hessenberg matrices. *Math. Comp.*, 71(240):1473–1496 (electronic), 2002.

[22] T. L. Wang and W. B. Gragg. Convergence of the unitary QR algorithm with a unimodular Wilkinson shift. *Math. Comp.*, 72(241):375–385 (electronic), 2003.

[23] J. H. Wilkinson. *The algebraic eigenvalue problem.* Monographs on Numerical Analysis. The Clarendon Press Oxford University Press, New York, 1988. Oxford Science Publications.