A Fast Implicit QR Eigenvalue Algorithm for Companion Matrices

D. A. Bini [a,1] P. Boito [a,1] Y. Eidelman [b] L. Gemignani [a,1]
I. Gohberg [b]

[a] *Dipartimento di Matematica, Università di Pisa, Largo Bruno Pontecorvo 5, 56127 Pisa, Italy*

[b] *School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Ramat-Aviv, 69978, Israel*

Abstract

An implicit version of the shifted QR eigenvalue algorithm given in [D. A. Bini, Y. Eidelman, I. Gohberg, L. Gemignani, SIAM J. Matrix Anal. Appl. 29 (2007), no. 2, 566–585] is presented for computing the eigenvalues of an $n \times n$ companion matrix using $O(n^2)$ flops and $O(n)$ memory storage. Numerical experiments and comparisons confirm the effectiveness and the stability of the proposed method.

Key words: Companion matrix, quasiseparable structure, QR iteration, eigenvalue computation, complexity
2000 MSC: 65F15, 65H17

1 Introduction

The paper concerns the efficient computation of the eigenvalues of companion matrices. A fast and numerically robust algorithm for this task, devised in [4], is based on the use of the explicit QR iteration applied to an input companion matrix $A \in \mathbb{C}^{n\times n}$ represented as a rank-one perturbation of a unitary matrix, namely, $A = U - \boldsymbol{z}\boldsymbol{w}^T$, where $U \in \mathbb{C}^{n\times n}$ is unitary and $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{C}^n$. The computational improvement with respect to the customary methods is

Email addresses: bini@dm.unipi.it (D. A. Bini), boito@mail.dm.unipi.it (P. Boito), eideyu@post.tau.ac.il (Y. Eidelman), gemignan@dm.unipi.it (L. Gemignani), gohberg@post.tau.ac.il (I. Gohberg).

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

achieved by exploiting the quasiseparable Hessenberg structure in the QR iterates inherited from the rank properties of the input companion matrix A. More specifically, in [4] it is shown that each iterate $A^{(k)}$ is a Hessenberg matrix expressed as the sum of a unitary matrix $U^{(k)}$ plus a rank-one correction. The fast implementation of the explicit QR iteration takes in input a complete set of *generators* for the quasiseparable structure of $A^{(k)}$ and returns as output a complete set of generators for the quasiseparable structure of $A^{(k+1)}$. The reduction of matrix operations to manipulating a set of $O(n)$ parameters enables one to perform each QR step in $O(n)$ floating point operations (flops) with $O(n)$ memory storage. In [4] it is also pointed out that in practice, due to rounding errors, some additional computations must be carried out in order to maintain both the quasiseparable structure of $A^{(k)}$ and the unitary property of $U^{(k)}$.

In the classical numerical linear algebra literature [12,17,2] it is generally claimed that the *implicit QR method* should be preferred to its explicit counterpart since it can be faster in the case of multiple shifts, more stable numerically and, moreover, admits suitable variants for the case of real inputs. The (multishift) QR iteration proceeds as follows:

$$\begin{aligned} &A_0 = A \\ &q_k(A^{(k)}) = Q^{(k)} R^{(k)}, \ \text{(QR factorization)} \\ &A^{(k+1)} := {Q^{(k)}}^H A^{(k)} Q^{(k)}, \end{aligned} \tag{1}$$

where $q_k(z)$ is a monic polynomial of degree one (*single-shift step*) or two (*double-shift step*) suitably chosen to accelerate the convergence.

The *bulge-chasing implicit QR techniques* manage to perform the transformation $A^{(k)} \to A^{(k+1)}$ without explicitly forming the matrix $q_k(A^{(k)})$. The implicit determination of $A^{(k+1)}$ from $A^{(k)}$ was first described by Francis [11,10] (see also [12] and [17] and the references given therein). Let Q_1 be a Householder matrix chosen to annihilate the subdiagonal entries in the first column of $q_k(A^{(k)})$. The transformation $A^{(k)} \to Q_1^H A^{(k)} Q_1$ corrupts the upper Hessenberg form of $A^{(k)}$ by creating a *bulge* of size $\deg(q_k(z))$ at the top left corner of the matrix. It is shown that the computation of $A^{(k+1)}$ essentially consists of chasing the bulge down and to the right of the matrix until it disappears. The task can be accomplished by a standard Hessenberg reduction employing a sequence $Q_2, \ldots, Q_{n-1}$ of Householder matrices. The resulting algorithm requires $O(n^2)$ flops and it is provably backward stable [15].

The first implicit fast and accurate QR eigenvalue algorithm for real companion matrices has been presented in [6]. The algorithm employs a factored representation of $A^{(k)}$ as the product of a unitary Hessenberg by a quasiseparable upper triangular matrix. A similar representation was previously considered in [1] for the efficient eigenvalue computation of certain rank-one corrections

of unitary matrices. The bulge-chasing procedure is performed in linear time by taking advantage of the special form of $A^{(k)}$. Specifically, the multiplication on the left by the elementary unitary matrix Q_j^H amounts to a suitable rearrangement of the Schur factorization of the unitary factor by performing a sequence of swapping operations. Moreover, the multiplication on the right by Q_j involves the updating of the quasiseparable structure of the upper triangular factor via manipulations of its generators. At the end of this updating process an auxiliary compression step is still required to recover a minimal-order quasiseparable parametrization for the triangular factor.

In this paper we modify the algorithm given in [4] to incorporate single-shift and double-shift implicit techniques, thus obtaining a fast adaptation of the implicit QR method for the case where the initial matrix $A = A^{(0)} \in \mathbb{C}^{n\times n}$ is in companion form. Our algorithm basically differs from the method in [6] in that we use a different compact way to represent the matrices involved. Specifically, the novel scheme still relies on the representation of each iterate $A^{(k)}$ as a rank-one correction of a unitary matrix, namely, $A^{(k)} = U^{(k)} - \boldsymbol{z}^{(k)}\boldsymbol{w}^{(k)^T}$. Our eigenvalue algorithm takes in input a condensed representation of $U^{(k)}$, the perturbation vectors $\boldsymbol{z}^{(k)}$ and $\boldsymbol{w}^{(k)}$ as well as the coefficients of the shift polynomial $q_k(z)$ and returns as output the generators of $A^{(k+1)}$ computed by means of (1). Differently from the approach pursued in [3] and [4], here the use of a suitable factored representation of $U^{(k)}$ makes it possible the updating of the decomposition during the bulge-chasing process in a stable way without any additional compression and/or re-orthogonalization step. A minimal quasiseparable representation of $U^{(k)}$ is easily computed from its factored form and then used for finding the unitary matrices Q_j involved in the bulge-chasing process. The proposed algorithm is therefore logically simple, efficient and numerically stable. It turns out that the QR iteration can be performed at the cost of $O(n)$ flops using $O(n)$ storage. Numerical experiments confirm that the algorithm is stable. Experimental comparisons are also included by showing that in the considered cases our algorithm outperforms the one in [6].

Algorithms for explicit and implicit QR iterations with Hermitian and small rank perturbations of Hermitian matrices may be found in [5,7–9]. An implicit QR algorithm for companion matrices was also discussed in [16].

The paper is organized as follows. In Sect. 2 we recall the structural properties and introduce condensed representations for the matrices generated by the QR process applied to an input companion matrix. Fast algorithms that carry out both the single-shift and the double-shift implicit QR iteration applied to such matrices are described in Sect. 3. In Sect. 4 we address some complementary issues concerning deflation and stopping techniques while in Sect. 5 the results of extensive numerical experiments are reported. Finally, the conclusion and a discussion are the subjects of Sect. 6.

2 Matrix structures under the QR iteration applied to a companion matrix: The classes $\mathcal{H}_n$ and $\mathcal{U}_n$

In this section we analyze some matrix structures which play a fundamental role in the design of a fast adaptation of the shifted QR eigenvalue algorithm (1) applied to an input matrix $A = A^{(0)} \in \mathbb{C}^{n\times n}$ in companion form.

For a given monic polynomial $p(z)$ of degree n,

$$p(z) = p_0 + p_1 z + \dots p_{n-1} z^{n-1} + z^n = \prod_{i=1}^{n} (z - \xi_i),$$

the associated companion matrix A is defined by

$$A = \begin{pmatrix} 0 & & & -p_0 \\ 1 & 0 & & -p_1 \\ & 1 & \ddots & \vdots \\ & & \ddots\ 0 & \vdots \\ & & & 1\ -p_{n-1} \end{pmatrix}.$$

It is well known that the set of eigenvalues of A coincides with the set of zeros $\xi_1, \dots, \xi_n$ of $p(z)$ and this property provides the classical reduction between the computation of polynomial zeros and eigenvalues of companion matrices. Matrix methods based on the QR iteration (1) applied to a companion matrix are customary for polynomial root-finding: in fact, the MATLAB [2] command `roots` relies on this approach.

The general routines for Hessenberg matrices require $O(n^2)$ flops and $O(n^2)$ memory space per iteration. Fast adaptations of the QR eigenvalue algorithm applied to the companion matrix A can achieve better (linear) estimates both for the cost and for the storage. The computational improvement is due to the exploitation of certain additional matrix structures in the iterates $A^{(k)}$, $k \geq 0$, that follow from the companion form of $A = A^{(0)}$. Specifically, it is worth noting that $A = A^{(0)}$ is an upper Hessenberg matrix which can be expressed as a rank-one perturbation of the unitary companion matrix $U = U^{(0)}$ associated

[2] MATLAB is a registered trademark of The Mathworks, Inc..

with the polynomial $z^n - 1$, i.e.,

$$A = \begin{pmatrix} 0 & \dots & 0 & 1 \\ 1 & \ddots & & 0 \\ & \ddots & \ddots & \vdots \\ & & 1 & 0 \end{pmatrix} - \begin{pmatrix} p_0 + 1 \\ p_1 \\ \vdots \\ p_{n-1} \end{pmatrix} \begin{pmatrix} 0 & 0 & \dots & 1 \end{pmatrix}$$

The Hessenberg shape is preserved under the QR iteration and, moreover, from (1) it follows that $A^{(k+1)}$ is a rank-one perturbation of a unitary matrix whenever $A^{(k)}$ fulfills the same property. Therefore each matrix $A^{(k)}$, $k \geq 0$, generated by the QR iteration (1) applied to the initial companion matrix $A = A^{(0)}$ can be recognized as a member of a larger class of structured matrices.

Definition 2.1 *Let $\mathcal{H}_n \subset \mathbb{C}^{n\times n}$ be the class of $n \times n$ upper Hessenberg matrices defined as rank-one perturbations of unitary matrices. That is, $B \in \mathcal{H}_n$ if there exist $U \in \mathbb{C}^{n\times n}$ unitary and $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{C}^n$ such that*

$$B = U - \boldsymbol{z}\boldsymbol{w}^T. \tag{2}$$

The vectors $\boldsymbol{z} = (z_i)_{i=1}^n$, $\boldsymbol{w} = (w_i)_{i=1}^n$ are called the perturbation vectors *of the matrix B.*

The Hessenberg form of B implies some additional properties of the unitary matrix U.

Definition 2.2 *We define the class $\mathcal{U}_n$ to be the set of $n \times n$ unitary matrices $U = (u_{i,j})_{i,j=1}^n$ such that*

$$u_{i,j} = z_i \cdot w_j, \quad 1 \leq j \leq i-2,\ 3 \leq i \leq n, \tag{3}$$

for suitable complex numbers z_i and w_j referred to as lower generators *of the matrix U.*

Let B be a matrix from the class $\mathcal{H}_n$ represented in the form (2) with the unitary matrix U and the vectors $\boldsymbol{z} = (z_i)_{i=1}^n$ and $\boldsymbol{w} = (w_i)_{i=1}^n$. Then the matrix U belongs to the class $\mathcal{U}_n$ and the numbers $z_i,\ i = 1, \dots, n$ and $w_i,\ i = 1, \dots, n-2$ are lower generators of the matrix U. In the reverse direction let U be a unitary matrix from the class $\mathcal{U}_n$ with lower generators z_i $(i = 3, \dots, n)$ and w_i $(i = 1, \dots, n-2)$. Take arbitrary numbers z_1, z_2 and w_{n-1}, w_n and set $\boldsymbol{z} = (z_i)_{i=1}^n, \boldsymbol{w} = (w_i)_{i=1}^n$, the matrix $B = U - \boldsymbol{z}\boldsymbol{w}^T$ belongs to the class $\mathcal{H}_n$.

In this paper we use essentially representations of the matrices from the class $\mathcal{U}_n$ as a product $U = VF$, where V and F are unitary matrices with zero entries above some superdiagonal and below some subdiagonal respectively. We consider in more details properties of such matrices. Denote by $\oplus$ the

direct sum of two matrices such that

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}.$$

Lemma 2.3 *Let $W = (w_{ij})_{i,j=1}^{n}$ be unitary matrix and let m be positive integer.*

The matrix W satisfies the conditions $w_{ij} = 0$ for $i > j + m$ if and only if W admits the factorization

$$W = W_1 W_2 \cdots W_{n-m}, \tag{4}$$

where

$$W_i = I_{i-1} \oplus \mathcal{W}_i \oplus I_{n-i-m}, \quad i = 1, \ldots, n-m \tag{5}$$

with $(m+1) \times (m+1)$ unitary matrices $\mathcal{W}_i$.

The matrix W satisfies the conditions $w_{ij} = 0$ for $j > i + m$ if and only if W admits the factorization

$$W = W_{n-m} W_{n-m-1} \cdots W_2 W_1 \tag{6}$$

with the unitary matrices $\mathcal{W}_i$ $(i = 1, \ldots, n-2)$ of the form (4).

PROOF. Assume that

$$w_{i,j} = 0, \quad i > j + m. \tag{7}$$

The first column of the matrix W has the form

$$W(:,1) = \begin{pmatrix} w_1 \\ 0 \end{pmatrix},$$

where w_1 is an $m+1$-dimensional column with the unit norm. We take a unitary $(m+1) \times (m+1)$ matrix $\mathcal{W}_1$ such that $\mathcal{W}_1^H f_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T$ and then determine the matrix W_1 by the formula (5). We get $W_1^H W(:,1) = e_1$, where e_1 is the first vector from the standard basis in $\mathbb{C}^n$ and since the matrix $W_1^* W$ is unitary we conclude that

$$W = W_1 \begin{pmatrix} 1 & 0 \\ 0 & \hat{W}_2 \end{pmatrix}$$

with a unitary $(n-1)\times(n-1)$ matrix $\hat{W}_2$ which satisfies the condition (7). Next we apply the same procedure to the matrix $\hat{F}_2$ and so on and on the $(n-m-1)$-th step we obtain the factorization (16).

Assume that the matrix W has the form (4) with the matrices W_i $(i = 1,\ldots,n-m)$ of the form (5). We prove by induction in n that the condition (7) holds. For $n = m+2$ we have obviously

$$W = W_1 W_2 = \begin{pmatrix} \mathcal{W}_1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{W}_2 \end{pmatrix}$$

and therefore $w_{n,1} = w_{m+2,1} = 0$. Let for some $n \geq 2$ the $(n-1)\times(n-1)$ matrix

$$\tilde{W} = \tilde{W}_1 \cdots \tilde{W}_{n-m-1},$$

where

$$\tilde{W}_i = I_{i-1} \oplus \mathcal{W}_i \oplus I_{n-i-m-1}, \quad i = 1,\ldots,n-m-1$$

with $(m+1)\times(m+1)$ matrices $\mathcal{W}_i$, satisfies the condition (12). The matrix W defined via (4), (5) may be done in the form

$$W = W_1 \cdots W_{n-m-1} W_{n-m} = \begin{pmatrix} \tilde{W} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} I_{n-m-1} & 0 \\ 0 & \mathcal{W}_{n-m} \end{pmatrix}.$$

Hence it follows that

$$W(m+i+1:n, 1:i) = \begin{pmatrix} \tilde{W}(m+i+1:n-1, 1:i) \\ 0 \end{pmatrix}, \quad i = 1,\ldots,n-m-1$$

which completes the proof of the first part of the lemma.

Applying the first part of the lemma to the transposed matrix W^T we prove the second part. □

Remark 2.4 *Every matrix $\mathcal{W}_i$ $(i = 1,\ldots,n-2)$ may be taken either as a $(m+1)\times(m+1)$ Householder matrix or as a product of complex Givens rotations.*

For a matrix $U \in \mathcal{U}_n$ we derive a condensed representation as a product of elementary unitary matrices.

Definition 2.5 *Let $\boldsymbol{z} = (z_i)_{i=1}^n$ be a vector and let $\mathcal{V}_i$, $i = 2,\ldots,n$ be 2×2 unitary matrices. We say that the vector $\boldsymbol{z}$ is reduced by the matrices $\mathcal{V}_i$ $(i =$*

$2, \ldots, n)$ *if there exist complex numbers* $\beta_i,\ i = 2, \ldots, n$ *such that the relations*

$$\beta_n = z_n, \quad \mathcal{V}_i^H \begin{pmatrix} z_i \\ \beta_{i+1} \end{pmatrix} = \begin{pmatrix} \beta_i \\ 0 \end{pmatrix},\ i = n-1, \ldots, 2, \tag{8}$$

hold. Define the unitary matrix $V \in \mathbb{C}^{n\times n}$ *as the product*

$$V = V_{n-1} \cdots V_2, \tag{9}$$

where

$$V_i = I_{i-1} \oplus \mathcal{V}_i \oplus I_{n-i-1}, \quad 2 \le i \le n-1. \tag{10}$$

We say also that the vector $\boldsymbol{z}$ *is reduced by the matrix* V*.*

From (9) by Lemma 2.3 it follows that $V = (v_{i,j})$ is a unitary lower Hessenberg matrix, i.e.,

$$v_{i,j} = 0, \quad j > i+1. \tag{11}$$

Lemma 2.6 *Let* $U \in \mathbb{C}^{n\times n}$ *be a unitary matrix from the class* $\mathcal{U}_n$ *with lower generators* $z_i\ (i = 3, \ldots, n),\ w_j\ (j = 1, \ldots, n-2)$ *and let* z_1, z_2, w_{n-1}, w_n *be arbitrary numbers. Assume that the vector* $\boldsymbol{z} = (z_i)_{i=1}^n$ *is reduced by* 2×2 *unitary matrices* $\mathcal{V}_i,\ i = 2, \ldots, n-1$*, define the unitary matrix* V *via relations (9), (10). Then the unitary matrix* $F = V^H \cdot U = (f_{i,j})$ *has null entries below the second subdiagonal, i.e.,*

$$f_{i,j} = 0, \quad i > j+2. \tag{12}$$

PROOF. Set $\boldsymbol{w} = (w_1, \ldots, w_n)^T$, $\boldsymbol{z} = (z_1, \ldots, z_n)^T$ and define the matrix $B \in \mathbb{C}^{n\times n}$ as in (2). It is clear that B is an upper Hessenberg matrix. Furthermore the relations (9)-(8) imply that the vector $g = V^H \boldsymbol{z} = (g_i)$ has null components g_i for $i > 3$. By Lemma 2.3 V is a lower Hessenberg matrix. Hence using the fact that B and V^H are upper Hessenberg matrices we conclude that the matrix

$$F = V^H \cdot U = V^H \cdot B + (V^H \boldsymbol{z}) \cdot \boldsymbol{w}^T$$

satisfies the relations (12). $\square$

Lemma 2.7 *Every matrix* U *from the class* $\mathcal{U}_n$ *admits the decomposition*

$$U = V \cdot F, \tag{13}$$

where V *is a unitary lower Hessenberg matrix and* $F = (f_{ij})$ *is a unitary matrix satisfying the condition* $f_{i,j} = 0, \quad i > j+2$*. Moreover one can take the matrix* V *in the form*

$$V = V_{n-1} \cdots V_2, \tag{14}$$

where

$$V_i = I_{i-1} \oplus \mathcal{V}_i \oplus I_{n-i-1}, \quad 2 \le i \le n-1 \tag{15}$$

with 2×2 *unitary matrices* $\mathcal{V}_i$ *and the matrix* F *in the form*

$$F = F_1 F_2 \cdots F_{n-2}, \tag{16}$$

where

$$F_i = I_{i-1} \oplus \mathcal{F}_i \oplus I_{n-i-2}, \qquad 1 \le i \le n-2. \tag{17}$$

with 3×3 *unitary matrices* $\mathcal{F}_i$

PROOF. Let z_i $(i = 3, \ldots, n)$, w_j $(j = 1, \ldots, n-2)$ be lower generators of the matrix U and let z_1, z_2, w_{n-1}, w_n be arbitrary numbers. Determine 2×2 unitary matrices $\mathcal{V}_i$ such that the relations (8) hold, and define the unitary matrix V via (14), (15). By Lemma 2.3 the matrix V is lower Hessenberg. Moreover by Lemma 2.6 the matrix $F = V^H \cdot U$ is lower banded with bandwidth 2. $\square$

Summing up, we obtain that any matrix B from the class $\mathcal{H}_n$ can be represented as

$$B = U - \boldsymbol{z}\boldsymbol{w}^T = V \cdot F - \boldsymbol{z}\boldsymbol{w}^T,$$

where V and F are unitary matrices represented in the factored form as specified by (14), (15) and (16), (17), and $\boldsymbol{z}$ and $\boldsymbol{w}$ are the perturbation vectors. Hence, the matrix $B \in \mathcal{H}_n$ is completely specified by the following parameters:

(1) the unitary matrices $\mathcal{V}_k$, $k = 2, \ldots, n-1$ defining the matrix V;
(2) the unitary matrix $\mathcal{F}_k$, $k = 1, \ldots, n-2$ defining the matrix F;
(3) the perturbation vectors $\boldsymbol{z}$ and $\boldsymbol{w}$.

These parameters are also called *the generating elements* of the matrix B.

The decomposition of $U \in \mathcal{U}_n$ by means of the elementary matrices $\mathcal{V}_k$ and $\mathcal{F}_k$ is numerically robust but not easy to be manipulated under the QR iteration applied to B. In this respect a more suited condensed form of U is its quasiseparable parametrization introduced in [4]. The representation gives an explicit description of each entry of U as the product of certain vectors and matrices of small size. We will show that for every matrix from the class $\mathcal{U}_n$ there exist vectors $\boldsymbol{g}_j \in \mathbb{C}^2$, $1 \le j \le n$, $\boldsymbol{h}_j \in \mathbb{C}^2$, $1 \le j \le n$, and matrices $B_j \in \mathbb{C}^{2\times 2}$, $2 \le j \le n$, such that

$$u_{i,j} = \boldsymbol{g}_i^T B_{i,j}^{\times} \boldsymbol{h}_j, \text{ for } j - i \ge 0, \tag{18}$$

where $B_{i,j}^{\times} = B_{i+1} \cdots B_j$ for $n \ge j \ge i+1$ and $B_{i,j}^{\times} = I_2$ if $i = j$. The elements $\boldsymbol{g}_j, \boldsymbol{h}_j$ $(j = 1, \ldots, n)$, B_j $(j = 2, \ldots, n)$ are called *upper generators* of the matrix U. The next result shows that generators of U can be easily reconstructed

from the two sequences of elementary matrices $\mathcal{V}_k$ and $\mathcal{F}_k$ defining the unitary factors V and F, respectively. The recurrence relations for the upper generators can also be used to compute some elements in the lower triangular part of U. In particular we provide the formulas for the subdiagonal elements of U involved in the QR iteration.

Theorem 2.8 *Let $U = (u_{i,j})$ be a unitary matrix from the class $\mathcal{U}_n$ with the given factorization $U = VF$, where the factors V and F defined by the formulas (14), (15) and (16), (17) using the given unitary 2×2 matrices $\mathcal{V}_i$ $(i = 2, \ldots, n-1)$ and unitary 3×3 matrices $\mathcal{F}_i$ $(i = 1, \ldots, n-2)$.*

The entries $u_{i,j}$, $\max\{1, i-2\} \le j \le n$, $1 \le i \le n$, satisfy the following relations:

$$u_{i,j} = \boldsymbol{g}_i^T B_{i,j}^{\times} \boldsymbol{h}_j \text{ for } j - i \ge 0; \tag{19}$$

$$u_{i,j} = \sigma_j \text{ for } 2 \le i = j + 1 \le n, \tag{20}$$

where the vectors $\boldsymbol{h}_k$ and the matrices B_k are determined by the formulas

$$\boldsymbol{h}_k = \mathcal{F}_k(1\colon 2, 1), \quad B_{k+1} = \mathcal{F}_k(1\colon 2, 2\colon 3), \quad 1 \le k \le n-2, \tag{21}$$

and the vectors $\boldsymbol{g}_k$ and the numbers σ_k are computed recursively

$$\Gamma_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}, \quad \boldsymbol{g}_1^T = \begin{pmatrix} 1 & 0 \end{pmatrix}; \tag{22}$$

$$\begin{pmatrix} \sigma_k & \boldsymbol{g}_{k+1}^T \\ * & \Gamma_{k+1} \end{pmatrix} = \mathcal{V}_{k+1} \begin{pmatrix} \Gamma_k & 0 \\ 0 & 1 \end{pmatrix} \mathcal{F}_k, \quad k = 1, \ldots, n-2, \tag{23}$$

$$\sigma_{n-1} = \Gamma_{n-1} \boldsymbol{h}_{n-1}, \quad \boldsymbol{g}_n^T = \Gamma_{n-1} B_{n-1} \tag{24}$$

with the auxiliary variables $\Gamma_k \in \mathbb{C}^{1\times 2}$.

PROOF. Let the elements $\boldsymbol{g}_i, \boldsymbol{h}_i$ $(i = 1, \ldots, N)$, B_k $(k = 2, \ldots, n)$, σ_k $(k = 1, \ldots, n-1)$ be given via (21)-(24). Using the elements $\boldsymbol{g}_k, B_k$ we define the matrices G_k $(k = 1, \ldots, N)$ of sizes $k \times 2$ via relations

$$G_k = \operatorname{col}(\boldsymbol{g}_i^T B_{i,k}^{\times})_{i=1}^{k}, \quad k = 1, \ldots, n. \tag{25}$$

It is easy to see that the relations (19), (20) are equivalent to

$$U(1:k+1, k) = \begin{pmatrix} G_k \boldsymbol{h}_k \\ \sigma_k \end{pmatrix}, \quad k = 1, \ldots, n-1; \quad U(1:n, n) = G_n \boldsymbol{h}_n. \tag{26}$$

Next we set

$$C_k = V_{k+1} \cdots V_2 F_1 \cdots F_k, \quad k = 1, \ldots, n-2.$$

Using the formulas (14), (16) we get

$$C_k = \begin{pmatrix} C_k(1:k+2,1:k+2) & 0 \\ 0 & I_{n-k-2} \end{pmatrix}, \quad k = 1, \ldots, n-2. \tag{27}$$

Moreover using (13), (14), (16) we have

$$U = (V_{N-1} \cdots V_{k+2}) C_k (F_{k+1} \cdots F_{N-2}), \quad k = 1, \ldots, n-3; \quad U = C_{n-2}.$$

Furthermore from (14), (17) we get

$$V_{n-1} \cdots V_{k+2} = \begin{pmatrix} I_{k+1} & 0 \\ 0 & * \end{pmatrix}, \quad F_{k+1} \cdots F_{n-2} = \begin{pmatrix} I_k & 0 \\ 0 & * \end{pmatrix}$$

and therefore

$$U(1:k+1,1:k) = C_k(1:k+1,1:k), \quad k = 1, \ldots, n-2. \tag{28}$$

Next we prove by induction that

$$C_k(1:k+2,1:k+2) = \begin{pmatrix} C_{k-1}(1:k,1:k-1) & G_k \boldsymbol{h}_k & G_k B_{k+1} \\ * & \sigma_k & \boldsymbol{g}_{k+1}^T \\ * & * & \Gamma_{k+1} \end{pmatrix}, \; k = 1, \ldots, n-2. \tag{29}$$

For $k = 1$ using the formula $C_1 = V_2 F_1$ and the formulas (14), (17) we have therefore

$$C_1(1:3,1:3) = \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_2 \end{pmatrix} \mathcal{F}_1.$$

From here using (22), $\boldsymbol{g}_1^T = G_1$ and (21), (23) with $k = 1$ we get

$$C_1(1,1:3) = \mathcal{F}_1(1,1:3) = \boldsymbol{g}_1 \begin{pmatrix} \boldsymbol{h}_1 & B_2 \end{pmatrix} = \begin{pmatrix} G_1 \boldsymbol{h}_1 & G_1 B_2 \end{pmatrix},$$

$$C_1(2:3,1:3) = \mathcal{V}_2 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathcal{F}_1 = \begin{pmatrix} \sigma_1 & \boldsymbol{g}_2^T \\ * & \Gamma_2 \end{pmatrix}$$

which implies (29) with $k = 1$.

Let for some k with $1 \le k \le n-3$ the relation (29) holds. Using the equality

$G_k = \begin{pmatrix} G_k B_{k+1} \\ \boldsymbol{g}_{k+1}^T \end{pmatrix}$ one can rewrite (29) in the form

$$C_k(1:k+2, 1:k+2) = \begin{pmatrix} C_k(1:k+1, 1:k) & G_{k+1} \\ * & \Gamma_{k+1} \end{pmatrix}.$$

Using the formula $C_{k+1} = V_{k+2} C_k F_{k+1}$ and the formulas (27), (14), (17) we obtain

$$C_{k+1}(1:k+3, k+1:k+3) =$$
$$\begin{pmatrix} I_{k+1} & 0 \\ 0 & \mathcal{V}_{k+2} \end{pmatrix} \begin{pmatrix} C_k(1:k+1, 1:k) & G_{k+1} & 0 \\ * & \Gamma_{k+1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} I_k & 0 \\ 0 & \mathcal{F}_{k+1} \end{pmatrix}.$$

From here using the relation $\mathcal{F}_{k+1}(1:2, 1:3) = \begin{pmatrix} \boldsymbol{h}_{k+1} & B_{k+2} \end{pmatrix}$ and the formula (23) we get

$$C_{k+1}(1:k+1, k+1:k+3) = \begin{pmatrix} G_{k+1}\boldsymbol{h}_{k+1} & G_{k+1} B_{k+2} \end{pmatrix},$$

$$C_{k+1}(k+2:k+3, k+1:k+3) = \mathcal{V}_{k+2} \begin{pmatrix} \Gamma_{k+1} & 0 \\ 0 & 1 \end{pmatrix} \mathcal{F}_{k+1} = \begin{pmatrix} \sigma_{k+1} & \boldsymbol{g}_{k+2}^T \\ * & \Gamma_{k+2} \end{pmatrix}$$

which completes the proof of (29).

Now combining the relations (28), (29) together we obtain the equalities (26) for $k = 1, \ldots, n-2$. Moreover using the equality $C_{n-2} = U$ and the relation (29) with $k = n-2$ we have

$$U(1:n, n-1:n) = \begin{pmatrix} G_{n-2} B_{n-1} \\ \boldsymbol{g}_{n-1}^T \\ \Gamma_{n-1} \end{pmatrix}.$$

Taking here $B_n, \boldsymbol{h}_{n-1}, \boldsymbol{h}_n, \sigma_{n-1}, \boldsymbol{g}_n^T$ as in (21), (24) we get

$$U(1:n, n-1:n) = \begin{pmatrix} G_{n-1} h_{n-1} & G_{n-1} B_n \boldsymbol{h}_n \\ \sigma_{N-1} & \boldsymbol{g}_n^T \boldsymbol{h}_n \end{pmatrix}$$

which completes the proof of the theorem. $\square$

Remark 2.9 *One can check easily that the relations (18) are equivalent to the equalities*

$$U(k, k:n) = \boldsymbol{g}_k^T H_k, \quad k = 1, \ldots, n \tag{30}$$

with $2 \times (n-k+1)$ *matrices* H_k *defined via relations*

$$H_k = \mathrm{row}(B_{k,i}^{\times} h_i)_{i=k}^{n}, \quad k = 1, \ldots, n. \tag{31}$$

In the next sections we provide a fast adaptation of the implicit QR iteration (1) applied for computing the complete set of eigenvalues of the input companion matrix $A = A_0 \in \mathcal{H}_n$. It takes in input the generating elements of the matrix $A_k \in \mathcal{H}_n$ together with the coefficients of the shift polynomial $q_k(z)$ and returns as output the generating elements of the matrix $A_{k+1} \in \mathcal{H}_n$. The computation is carried out at the total cost of $O(n)$ flops using $O(n)$ memory space.

3 The Structured QR Iteration

In this section we present a fast modification of the implicit QR iteration (1) applied to an input matrix $A \in \mathcal{H}_n$ expressed in terms of its generating elements, that is, given in the form

$$A = U - \boldsymbol{z}\boldsymbol{w}^T = V \cdot F - \boldsymbol{z}\boldsymbol{w}^T = (V_{n-1} \cdots V_2) \cdot (F_1 \cdots F_{n-2}) - \boldsymbol{z}\boldsymbol{w}^T,$$

where $U = V \cdot F$ is the factored representation of $U \in \mathcal{U}_n$ and $\boldsymbol{z}$ and $\boldsymbol{w}$ are the perturbation vectors of A. For the sake of notational simplicity we omit the superscript k which labels the QR steps.

Let $q(z)$ be a monic polynomial of degree $\ell \in \{1, 2\}$ determined to speed up the convergence of the QR iteration. The cases $\ell = 1$ and $\ell = 2$ are referred to as the *single-shift* and the *double-shift* iteration, respectively. Moreover, let $Q_1 = \mathcal{G}_1 \oplus I_{n-\ell-1}$ be a unitary matrix suitably chosen to annihilate the subdiagonal entries in the first column of $q(A)$. The transformation $A \to Q_1^H A Q_1$ corrupts the upper Hessenberg form of A by creating a *bulge* of size ℓ at the top left corner of the matrix. It can be shown that the computation of $A^{(1)}$ essentially consists of chasing the bulge down and to the right of the matrix until it disappears. The task can be accomplished by a standard Hessenberg reduction employing a sequence $Q_2, \ldots, Q_{n-1}$ of unitary matrices.

The cumulative unitary factor $Q^{(1)}$ such that $A^{(1)} = {Q^{(1)}}^H A Q^{(1)}$ is given by the product $Q^{(1)} := Q_1 \cdot Q_2 \cdots Q_{n-1}$. The updating of the matrices $A \to A^{(1)}$ and $U \to U^{(1)}$ and of the vectors $\boldsymbol{z} \to \boldsymbol{z}^{(1)}$ and $\boldsymbol{w} \to \boldsymbol{w}^{(1)}$ can be carried out in $n-1$ steps according to the following rules:

$$A_1 = Q_1^H A; \quad A'_k = A_k Q_k,\ A_{k+1} = Q_{k+1}^H A'_k,\ k = 1, \ldots, n-1; \quad A^{(1)} := A_{n-1} Q_{n-1}. \tag{32}$$

Here every $A_k,\ k = 1, \ldots, n-1$ is an upper Hessenberg matrix. Moreover using the formula (2) and setting

$$U_1 = Q_1^H U; \quad U'_k = U_k Q_k,\ U_{k+1} = Q_{k+1}^H U'_k,\ k = 1, \ldots, n-1; \qquad U^{(1)} := U_{n-1} Q_{n-1}, \tag{33}$$

$$\boldsymbol{z}_1 = Q_1^H \boldsymbol{z}; \quad \boldsymbol{z}_{k+1} = Q_{k+1}^H \boldsymbol{z}_k,\ k = 1, \ldots, n-2; \quad \boldsymbol{z}^{(1)} := \boldsymbol{z}_{n-1} \tag{34}$$

$$\boldsymbol{w}_1 = \boldsymbol{w}; \quad \boldsymbol{w}_{k+1}^T = \boldsymbol{w}_k^T Q_k^T,\ k = 1, \ldots, N-1; \quad \boldsymbol{w}^{(1)} := \boldsymbol{w}_n. \tag{35}$$

we get

$$A_k = U_k - \boldsymbol{z}_k \boldsymbol{w}_k^T, \quad k = 1, \ldots, n-1. \tag{36}$$

Here U_k are unitary matrices and $\boldsymbol{z}_k, \boldsymbol{w}_k$ are vectors. Moreover the matrices Q_{k+1}^H are chosen to remove the bulge in the matrix A'_k and therefore A_k is an upper Hessenberg matrix. Hence it follows that every matrix A_k belongs to the class $\mathcal{H}_n$.

The *structured QR iteration* essentially consists of an efficient procedure for evaluating the unitary matrices Q_j, $1 \leq j \leq n-1$, combined with an efficient scheme for updating the factorized representation of U under the multiplication on the right and on the left by the matrices Q_j. In the next two subsections we describe the structured QR iteration in the single-shift and in the double-shift case. Specifically, in Subsection 3.1 we present a detailed description of the fast single-shift iteration and give a formal proof of its correctness. Then in Subsection 3.2 we sketch the fast double shift iteration putting in evidence the basic differences with the single-shift step. A proof of the correctness of the double-shift iteration proceeds in exactly the same way as for the single-shift case.

3.1 The Structured QR Iteration: The Single-Shift Case

The procedure FastQR_ss for the fast computation of the generating elements of $A^{(1)}$ such that

$$\begin{aligned} &A - \alpha I = QR, \text{ (QR factorization)} \\ &A^{(1)} := Q^H A Q \end{aligned} \tag{37}$$

is outlined below. Given the generating elements $\mathcal{V}_k$ $(k = 2, \ldots, n-1)$, $\mathcal{F}_k$ $(k = 1, \ldots, n-2)$, $\boldsymbol{z} = (z_i)_{i=1}^n$, $\boldsymbol{w} = (w_i)_{i=1}^n$ of A together with the shift parameter α, the algorithm calculates the generating elements $\mathcal{V}_k^{(1)}$ $(k = 2, \ldots, N-1)$, $\mathcal{F}_k^{(1)}$ $(k = 1, \ldots, N-2)$ $\boldsymbol{z}^{(1)} = (z_i^{(1)})_{i=1}^n$, $\boldsymbol{w}^{(1)} = (w_i^{(1)})_{i=1}^n$ of $A^{(1)}$. The

condensed representations of $A^{(1)}$ and $U^{(1)}$ are computed by the scheme (32)–(35), where Q_j are Givens rotation matrices determined in the bulge-chasing process.

Procedure FastQR_ss

(1) Using algorithm from Theorem 2.8 compute upper generators $\boldsymbol{g}_i, \boldsymbol{h}_i$ $(i = 1, \ldots, n)$, B_k $(k = 2, \ldots, n)$ and subdiagonal entries σ_k $(k = 1, \ldots, n-1)$ of the matrix U.

(2) Set $\beta_n = z_n$ and for $k = n-1, \ldots, 3$ compute

$$\beta_k = (\mathcal{V}_k^H)(1, 1:2) \begin{pmatrix} z_k \\ \beta_{k+1} \end{pmatrix}. \tag{38}$$

(3) Compute the Givens rotation matrices $\mathcal{G}_k,\ k = 1, \ldots, n-1$ and the updated perturbation vectors $\boldsymbol{z}^{(1)}, \boldsymbol{w}^{(1)}$

(a) Set $w_1^{(\tau)} = w_1$. Determine the complex Givens rotation matrix $\mathcal{G}_1$ such that

$$\mathcal{G}_1^H \begin{pmatrix} \boldsymbol{g}_1^T \boldsymbol{h}_1 - z_1 w_1 - \alpha \\ \sigma_1 - z_2 w_1 \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix} \tag{39}$$

and compute

$$\begin{pmatrix} z_1^{(1)} & * & * \\ z_2^{(\tau)} & \sigma_1^{(\tau)} & (g_2^{(\tau)})^T \end{pmatrix} = \mathcal{G}_1^H \begin{pmatrix} z_1 & \boldsymbol{g}_1^T \boldsymbol{h}_1 & \boldsymbol{g}_1^T B_1 \\ z_2 & \sigma_1 & \boldsymbol{g}_2^T \end{pmatrix}. \tag{40}$$

(b) For $k = 1, \ldots, n-2$ perform the following.
Compute

$$\begin{pmatrix} \rho_k' & \rho_k'' \\ \rho_k''' & \rho_k'''' \\ w_k^{(1)} & w_{k+1}^{(\tau)} \end{pmatrix} = \begin{pmatrix} \sigma_k^{(\tau)} & (\boldsymbol{g}_{k+1}^{(\tau)})^T \boldsymbol{h}_{k+1} \\ z_{k+2} w_k^{(\tau)} & \sigma_{k+1} \\ w_k^{(\tau)} & w_{k+1} \end{pmatrix} \mathcal{G}_k. \tag{41}$$

Determine the complex Givens rotation matrix $\mathcal{G}_{k+1}$ from the condition

$$\mathcal{G}_{k+1}^H \begin{pmatrix} \rho_k' - z_{k+1}^{(\tau)} w_k^{(1)} \\ \rho_k''' - z_{k+2} w_k^{(1)} \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}. \tag{42}$$

Compute

$$\begin{pmatrix} z_{k+1}^{(1)} \\ z_{k+2}^{(\tau)} \end{pmatrix} = \mathcal{G}_{k+1}^H \begin{pmatrix} z_{k+1}^{(\tau)} \\ z_{k+2} \end{pmatrix} \tag{43}$$

$$\begin{pmatrix} * & * \\ \sigma_{k+1}^{(\tau)} & (\boldsymbol{g}_{k+2}^{(\tau)})^T \end{pmatrix} = \mathcal{G}_{k+1}^H \begin{pmatrix} \rho_k'' & (\boldsymbol{g}_{k+1}^{(\tau)})^T B_{k+1} \\ \rho_k'''' & \boldsymbol{g}_{k+2} \end{pmatrix}. \tag{44}$$

(c) Compute

$$\begin{pmatrix} w_{n-1}^{(1)} & w_n^{(1)} \end{pmatrix} = \begin{pmatrix} w_{n-1}^{(\tau)} & w_n \end{pmatrix} \mathcal{G}_{n-1} \tag{45}$$

and set

$$z_n^{(1)} = z_n^{(\tau)}. \tag{46}$$

(4) The fourth stage of the algorithm is to compute the generating elements $\mathcal{V}_k^{(1)}$ $(k = 2, \ldots, n-1)$, $\mathcal{F}_k^{(1)}$ $(k = 1, \ldots, n-2)$ of the matrix $A^{(1)}$.

(a) Determine the complex Givens rotation matrix $\mathcal{V}_2^{(\tau)}$ such that

$$(\mathcal{V}_2^{(\tau)})^H \begin{pmatrix} z_2^{(\tau)} \\ \beta_3 \end{pmatrix} = \begin{pmatrix} \beta_2^{(1)} \\ 0 \end{pmatrix} \tag{47}$$

with some number $\beta_2^{(1)}$ and compute the 3×3 matrix

$$\mathcal{F}_1^{(\tau)} = \begin{pmatrix} 1 & 0 \\ 0 & (\mathcal{V}_2^{(\tau)})^H \end{pmatrix} \begin{pmatrix} \mathcal{G}_1^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_2 \end{pmatrix} \mathcal{F}_1. \tag{48}$$

(b) For $k = 1, \ldots, n-3$ perform the following.
Compute the 3×3 matrix

$$X_{k+1} = \begin{pmatrix} \mathcal{G}_{k+1}^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_{k+2} \end{pmatrix} \begin{pmatrix} \mathcal{V}_{k+1}^{(\tau)} & 0 \\ 0 & 1 \end{pmatrix} \tag{49}$$

and the 4×4 matrix

$$Y_k = \begin{pmatrix} \mathcal{F}_k^{(\tau)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{F}_{k+1} \end{pmatrix} \begin{pmatrix} \mathcal{G}_k & 0 \\ 0 & I_2 \end{pmatrix}. \tag{50}$$

Determine the complex Givens rotation matrix $\mathcal{Z}_k$ from the condition

$$X_{k+1}(1, 2:3)\mathcal{Z}_k = \begin{pmatrix} * & 0 \end{pmatrix}. \tag{51}$$

Compute the matrix

$$W_k = X_{k+1} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{Z}_k \end{pmatrix} \tag{52}$$

and the matrix

$$D_k = \begin{pmatrix} I_2 & 0 \\ 0 & \mathcal{Z}_k^H \end{pmatrix} Y_k. \tag{53}$$

The unitary matrix W_k has the zero entry in the position $(1,3)$ and the unitary matrix D_k has the zero entry in the position $(4,1)$.

Compute the factorization

$$W_k = \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_{k+2}^{(\tau)} \end{pmatrix} \begin{pmatrix} \mathcal{V}_{k+1}^{(1)} & 0 \\ 0 & 1 \end{pmatrix} \tag{54}$$

with unitary 2×2 matrices $\mathcal{V}_{k+2}^{(\tau)}, \mathcal{V}_{k+1}^{(1)}$ and the factorization

$$D_k = \begin{pmatrix} \mathcal{F}_k^{(1)} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{F}_{k+1}^{(\tau)} \end{pmatrix} \tag{55}$$

with 3×3 unitary matrices $\mathcal{F}_k^{(1)}, \mathcal{F}_{k+1}^{(\tau)}$.

(c) Compute

$$\mathcal{V}_{n-1}^{(1)} = \mathcal{G}_{n-1}^H \mathcal{V}_{n-1}^{(\tau)}, \tag{56}$$

$$\mathcal{F}_{n-2}^{(1)} = \mathcal{F}_{n-2}^{(\tau)} \begin{pmatrix} \mathcal{G}_{n-2} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{G}_{n-1} \end{pmatrix}. \tag{57}$$

Theorem 3.1 *Let $A = U - \boldsymbol{z}\boldsymbol{w}^T$ be a matrix from the class $\mathcal{H}_n$ with generating elements $\mathcal{V}_k$ $(k = 2, \ldots, n-1)$, $\mathcal{F}_k$ $(k = 1, \ldots, n-2)$, $\boldsymbol{z} = (z_i)_{i=1}^n$, $\boldsymbol{w} = (w_i)_{i=1}^n$ such that the vector $\boldsymbol{z}$ is reduced by the matrix*

$$V = V_{n-1} \cdots V_2, \quad V_i = I_{i-1} \oplus \mathcal{V}_i \oplus I_{n-i-1}, \quad 2 \le i \le n-1.$$

Then the output data $\mathcal{V}_k^{(1)}$ $(k = 2, \ldots, N-1)$, $\mathcal{F}_k^{(1)}$ $(k = 1, \ldots, N-2)$, $\boldsymbol{z}^{(1)} = (z_i^{(1)})_{i=1}^n$, $\boldsymbol{w}^{(1)} = (w_i^{(1)})_{i=1}^n$ returned by FastQR_ss *are the generating elements of the matrix $A^{(1)}$ satisfying* (37) *and, moreover, $\boldsymbol{z}^{(1)}$ is reduced by the matrix*

$$V^{(1)} = V_{n-1}^{(1)} \cdots V_2^{(1)}, \quad V_i^{(1)} = I_{i-1} \oplus \mathcal{V}_i^{(1)} \oplus I_{n-i-1}, \quad 2 \le i \le n-1.$$

PROOF. Using Stage 1 of the algorithm we obtain upper generators and subdiagonal entries of the matrix U.

On Stage 2 of the algorithm we use the fact that the vector z is reduced by the matrix V and compute the parameters β_k, $k = N, \ldots, 3$ using the formulas (8).

To justify Stage 3 of the algorithm we prove by induction that the formulas (39), (42) determine the unitary matrices $\mathcal{G}_k$, $k = 1, \ldots, n-1$ such that the

unitary matrices $Q_k = I_{k-1} \oplus \mathcal{G}_k \oplus I_{n-k-1}$ make the transformations (32)-(35) and moreover for $k = 1, \ldots, n-1$ the relations

$$U_k(k+1, k:n) = \begin{pmatrix} \sigma_k^{(\tau)} & (\boldsymbol{g}_{k+1}^{(\tau)})^T H_{k+1} \end{pmatrix} \tag{58}$$

hold and the vectors $\boldsymbol{z}_k, \boldsymbol{w}_k$ have the form

$$\boldsymbol{z}_k = (z_1^{(1)}, \ldots, z_k^{(1)}, z_{k+1}^{(\tau)}, z_{k+2}, \ldots, z_n)^T, \tag{59}$$

$$\boldsymbol{w}_k = (w_1^{(1)}, \ldots, w_{k-1}^{(1)}, w_k^{(\tau)}, z_{k+1}, \ldots, z_n)^T. \tag{60}$$

where the matrices U_k and the vectors $\boldsymbol{z}_k, \boldsymbol{w}_k$ are defined in (33)-(35).

On the first step we deal with the matrix

$$A_1 = Q_1^H A = U_1 - \boldsymbol{z}_1 \boldsymbol{w}_1^T$$

with $U_1 = Q_1^H U,\ \boldsymbol{z}_1 = Q_1^H \boldsymbol{z},\ \boldsymbol{w}_1 = \boldsymbol{w}$. We have

$$(A - \alpha I)(1:2, 1) = \begin{pmatrix} \boldsymbol{g}_1^T \boldsymbol{h}_1 - z_1 w_1 - \alpha \\ \sigma_1 - z_2 w_1 \end{pmatrix}$$

and determine the complex Givens rotation matrix $\mathcal{G}_1$ from the condition (39). Next using the formula (30) with $k = 1, 2$, the formula (20) with $k = 1$ and the relation $H_1 = \begin{pmatrix} \boldsymbol{h}_1 & B_2 H_2 \end{pmatrix}$ we get

$$U(1:2, 1:N) = \begin{pmatrix} \boldsymbol{g}_1^T h_1 & \boldsymbol{g}_1^T B_2 H_2 \\ \sigma_1 & \boldsymbol{g}_2^T H_2 \end{pmatrix}.$$

Using the formula (40) we obtain (58), (59), (60) with $k = 1$. Let for some k with $1 \le k \le n-2$ the statement of induction holds. From the formulas (33) it follows that

$$U_k(k+2, k:n) = U(k+2, k:n).$$

Furthermore using the formulas (30), (20) we obtain

$$U_k(k+2, k+1:N) = \begin{pmatrix} \sigma_{k+1} & \boldsymbol{g}_{k+2}^T H_{k+2} \end{pmatrix}$$

and from the formulas (36), $A_k(k+2, k) = 0$ and (59), (60) it follows that $U_k(k+2, k) = z_{k+2} w_k^{(\tau)}$. Moreover using the formula (58) and the equality

$$H_{k+1} = \begin{pmatrix} \boldsymbol{h}_{k+1} & B_{k+2} H_{k+2} \end{pmatrix}$$

we obtain

$$U_k(k+1:k+2,k:N)=\begin{pmatrix}\sigma_k^{(\tau)} & (\boldsymbol{g}_{k+1}^{(\tau)})^T\boldsymbol{h}_{k+1} & (\boldsymbol{g}_{k+1}^{(\tau)})^T B_{k+2}H_{k+2}\\ z_{k+2}w_k^{(\tau)} & \sigma_{k+1} & \boldsymbol{g}_{k+2}^T H_{k+2}\end{pmatrix}. \quad (61)$$

We have

$$A_k'=A_kQ_k=U_kQ_k-\boldsymbol{z}_k(\boldsymbol{w}_k^TQ_k)=U_k'-\boldsymbol{z}_k\boldsymbol{w}_{k+1}^T.$$

Postmultiplication of a matrix C by the matrix Q_k means only postmultiplication of the columns $k,k+1$ of C by the matrix $\mathcal{G}_k$. Hence applying the formula (41) we obtain

$$U_k'(k+1:k+2,k:n)=\begin{pmatrix}\rho_k' & \rho_k'' & (\boldsymbol{g}_{k+2}^{(\tau)})^T B_{k+2}H_{k+2}\\ \rho_k''' & \rho_k'''' & \boldsymbol{g}_{k+2}^T H_{k+2}\end{pmatrix}$$

and

$$\boldsymbol{w}_{k+1}=(w_1^{(1)},\dots,w_k^{(1)},w_{k+1}^{(\tau)},w_{k+2},\dots,w_n)^T.$$

Now the elements in the $(k+1,k),(k+2,k)$ positions in the matrix A_k' are

$$\rho_k'-z_{k+1}^{(\tau)}w_k^{(1)},\ \rho_k'''-z_{k+2}w_k^{(1)}$$

and we determine the matrix of Givens rotation $\mathcal{G}_{k+1}^H$ from the condition (42).

Next consider the matrix

$$A_{k+1}=Q_{k+1}^HA_k'=Q_{k+1}^HU_k'-Q_{k+1}^H\boldsymbol{z}_k\boldsymbol{w}_{k+1}^T=U_{k+1}-\boldsymbol{z}_{k+1}\boldsymbol{w}_{k+1}^T.$$

Premultiplication of a matrix C by the matrix Q_{k+1}^H means only premultiplication of the rows $k+1,k+2$ of C by the matrix $\mathcal{G}_{k+1}^H$. Hence using the formulas (44), (43) we obtain

$$U_{k+1}(k+2,k+1:N)=\begin{pmatrix}\sigma_{k+1}^{(\tau)} & (\boldsymbol{g}_{k+2}^{(\tau)})^T H_{k+2}\end{pmatrix}$$

and

$$\boldsymbol{z}_{k+1}=(z_1^{(1)},\dots,z_k^{(1)},z_{k+1}^{(1)},z_{k+2}^{(\tau)},z_{k+3},\dots,z_n)^T.$$

Thus we have determined the Givens rotation matrices $\mathcal{G}_k,\ k=1,\dots,n-1$ and the vectors $\boldsymbol{z}_{n-1},\boldsymbol{w}_{n-1}$. To finish the proof of Stage 3 notice that the last coordinates of the vectors $\boldsymbol{z}_{n-1}=\boldsymbol{z}_1$ and $\boldsymbol{w}_n=\boldsymbol{w}^{(1)}$ are given by the formulas (46) and (45).

Now we will justify Stage 4 of the algorithm. By the definition of generating elements we have

$$U=(V_{n-1}\cdots V_2)(F_1\cdots F_{n-2}), \quad (62)$$

where

$$V_i = I_{i-1} \oplus \mathcal{V}_i \oplus I_{n-i-1}, \quad F_i = I_{i-1} \oplus \mathcal{F}_i \oplus I_{n-i-2}.$$

We should prove that the matrix $U^{(1)}$ admits the factorization

$$U^{(1)} = (V_{n-1}^{(1)} \cdots V_2^{(1)})(F_1^{(1)} \cdots F_{n-2}^{(1)}), \tag{63}$$

where

$$V_i^{(1)} = I_{i-1} \oplus \mathcal{V}_i^{(1)} \oplus I_{n-i-1}, \quad F_i^{(1)} = I_{i-1} \oplus \mathcal{F}_i^{(1)} \oplus I_{n-i-2}$$

with the matrices $\mathcal{V}_i^{(1)}, \mathcal{F}_i^{(1)}$ determined in the algorithm, and moreover the vector $\boldsymbol{z}^{(1)}$ is reduced by the matrix $V^{(1)}$. To do this we prove by induction that every unitary matrix U_k from (33) admits the factorization

$$U_k = \hat{V}^{(k)} \cdot \hat{F}^{(k)}, \quad k = 1, \ldots, n-2 \tag{64}$$

with

$$\hat{V}^{(k)} = V_{n-1} \cdots V_{k+2} V_{k+1}^{(\tau)} V_k^{(1)} \cdots V_2^{(1)}, \quad \hat{F}^{(k)} = F_1^{(1)} \cdots F_{k-1}^{(1)} F_k^{(\tau)} F_{k+1} \cdots F_{n-2},$$

where

$$V_{k+1}^{(\tau)} = I_k \oplus \mathcal{V}_{k+1}^{(\tau)} \oplus I_{n-k-2}, \quad F_k^{(\tau)} = I_{k-1} \oplus \mathcal{F}_k^{(\tau)} \oplus I_{n-k-2}$$

and the matrices V_i, F_i and $V_i^{(1)}, F_i^{(1)}$ are defined above. Moreover we will prove that the vector $\boldsymbol{z}_k$ defined in (34) is reduced by the matrix $\hat{V}^{(k)}$, i.e.,

$$\beta_n = z_n, \quad \mathcal{V}_i^H \begin{pmatrix} z_i \\ \beta_{i+1} \end{pmatrix} = \begin{pmatrix} \beta_i \\ 0 \end{pmatrix}, \ i = n-1, \ldots, k+2; \tag{65}$$

$$(\mathcal{V}_{k+1}^{(\tau)})^H \begin{pmatrix} z_{k+1}^{(\tau)} \\ \beta_{k+2} \end{pmatrix} = \begin{pmatrix} \beta_{k+1}^{(1)} \\ 0 \end{pmatrix}, \tag{66}$$

$$(\mathcal{V}_i^{(1)})^H \begin{pmatrix} z_i^{(1)} \\ \beta_{i+1}^{(1)} \end{pmatrix} = \begin{pmatrix} \beta_i^{(1)} \\ 0 \end{pmatrix}, \quad i = k, \ldots, 2. \tag{67}$$

From (62) it follows that the matrix $U_1 = Q_1^H U$ has the form

$$U_1 = (V_{n-1} \cdots \tilde{V}_3)(Q_1^H V_2 F_1)(F_2 \cdots F_{n-2}).$$

Taking the matrix $\mathcal{V}_2^{(\tau)}$ from (47) and the matrix $\mathcal{F}_1^{(\tau)}$ from (48) we obtain

$$U_1 = \hat{V}^{(1)} \cdot \hat{F}^{(1)}$$

with

$$\hat{V}^{(1)} = V_{n-1} \cdots V_3 V_2^{(\tau)}, \quad \hat{F}^{(1)} = F_1^{(\tau)} F_2 \cdots \tilde{F}_{n-2}.$$

Furthermore using the relations (8) with $i = n, \ldots, 3$ and (47) we conclude that the vector $\boldsymbol{z}_1$ is reduced by the matrix $\hat{V}^{(1)} = V_{n-1} \cdots V_3 V_2^{(\tau)}$.

Assume that for some k with $1 \le k \le n-3$ the induction statement holds. Hence the matrix $U_{k+1} = Q_{k+1}^H U_k Q_k$ has the form

$$U_{k+1} = T_{k+1} \cdot S_{k+1}$$

with

$$T_{k+1} = (V_{n-1} \cdots \tilde{V}_{k+3})(Q_{k+1}^H V_{k+2} V_{k+1}^{(\tau)})(V_k^{(1)} \cdots V_2^{(1)}),$$
$$S_{k+1} = (F_1^{(1)} \cdots F_{k-1}^{(1)})(F_k^{(\tau)} F_{k+1} Q_k)(F_{k+2} \cdots F_{n-2}).$$

We have

$$Q_{k+1}^H V_{k+2} V_{k+1}^{(\tau)} = I_k \oplus X_{k+1} \oplus I_{n-k-3}$$

and

$$F_k^{(\tau)} F_{k+1} Q_k = I_{k-1} \oplus Y_k \oplus I_{n-k-3}$$

with the 3×3 unitary matrix X_{k+1} from (49) and the 4×4 unitary matrix Y_k from (50). Next we determine a 2×2 unitary matrix $\mathcal{Z}_k$ from the condition (51) and compute the matrix W_k by the formula (52). The unitary 3×3 matrix W_k has the zero entry in the position $(1, 3)$. Hence this matrix can be factorized in the form (54) with unitary 2×2 matrices $\mathcal{V}_{k+2}^{(\tau)}, \mathcal{V}_{k+1}^{(1)}$. Set

$$Z_k = I_{k+1} \oplus \mathcal{Z}_k \oplus I_{n-k-3}$$

and furthermore $\hat{V}^{(k+1)} = T_{k+1} Z_k,\ \hat{F}^{(k+1)} = Z_k^H S_{k+1}$. We have

$$U_{k+1} = \hat{V}^{(k+1)} \cdot \hat{F}^{(k+1)}, \tag{68}$$

where

$$\hat{V}^{(k+1)} = (V_{n-1} \cdots V_{k+3})(Q_{k+1}^H V_{k+2} V_{k+1}^{(\tau)} Z_k)(V_k^{(1)} \cdots V_2^{(1)}) =$$
$$(V_{n-1} \cdots V_{k+3})(V_{k+2}^{(\tau)} V_{k+1}^{(1)})(V_k^{(1)} \cdots \tilde{V}_2^{(1)})$$

which yields the desired representation for the matrix $\hat{V}^{(k+1)}$.

Let us check that the vector $\boldsymbol{z}_{k+1}$ is reduced by the matrix $\hat{V}^{(k+1)}$. Indeed using the relations (49), (52), (54) we have

$$\begin{pmatrix} (\mathcal{V}_{k+1}^{(1)})^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & (\mathcal{V}_{k+2}^{(\tau)})^H \end{pmatrix} =$$
$$\begin{pmatrix} 1 & 0 \\ 0 & \mathcal{Z}_k^H \end{pmatrix} \begin{pmatrix} (\mathcal{V}_{k+1}^{(\tau)})^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_{k+2}^H \end{pmatrix} \begin{pmatrix} \mathcal{G}_{k+1} & 0 \\ 0 & 1 \end{pmatrix}.$$

Moreover using the relations (43), (66) and the relation (65) with $i = k+2$ we get

$$\begin{pmatrix} 1 & 0 \\ 0 & \mathcal{Z}_k^H \end{pmatrix} \begin{pmatrix} (\mathcal{V}_{k+1}^{(\tau)})^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_{k+2}^H \end{pmatrix} \begin{pmatrix} \mathcal{G}_{k+1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} z_{k+1}^{(1)} \\ z_{k+2}^{(\tau)} \\ \beta_{k+3} \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ 0 & \mathcal{Z}_k^H \end{pmatrix} \begin{pmatrix} (\mathcal{V}_{k+1}^{(\tau)})^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{V}_{k+2}^H \end{pmatrix} \begin{pmatrix} z_{k+1}^{(\tau)} \\ z_{k+2} \\ \beta_{k+3} \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0 \\ 0 & \mathcal{Z}_k^H \end{pmatrix} \begin{pmatrix} (\mathcal{V}_{k+1}^{(\tau)})^H & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} z_{k+1}^{(\tau)} \\ \beta_{k+2} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \mathcal{Z}_k^H \end{pmatrix} \begin{pmatrix} \beta_{k+1}^{(1)} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \beta_{k+1}^{(1)} \\ 0 \\ 0 \end{pmatrix}.$$

Hence it follows that

$$(\mathcal{V}_{k+2}^{(\tau)})^H \begin{pmatrix} z_{k+2}^{(\tau)} \\ \beta_{k+3} \end{pmatrix} = \begin{pmatrix} \beta_{k+2}^{(1)} \\ 0 \end{pmatrix}, \quad (\mathcal{V}_{k+1}^{(1)})^H \begin{pmatrix} z_{k+1}^{(1)} \\ \beta_{k+2}^{(1)} \end{pmatrix} = \begin{pmatrix} \beta_{k+1}^{(1)} \\ 0 \end{pmatrix}$$

which together with the relations (65) with $i = N-1, \ldots, k+3$ and (67) implies that the vector $\boldsymbol{z}_{k+1}$ is reduced by the matrix $\hat{V}^{(k+1)}$.

Thus applying Lemma 2.6 to the matrix U_{k+1} we get

$$\hat{F}^{(k+1)}(k+3, k) = 0. \tag{69}$$

Furthermore we get

$$\hat{F}^{(k+1)} = (F_1^{(1)} \cdots F_{k-1}^{(1)})(Z_k^H F_k^{(\tau)} F_{k+1} Q_k)(F_{k+2} \cdots F_{n-2}).$$

Here we have

$$Z_k^H F_k^{(\tau)} F_{k+1} Q_k = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & D_k & 0 \\ 0 & 0 & I_{N-k-3} \end{pmatrix}$$

with the 4×4 matrix D_k from (53). This implies

$$\hat{F}^{(k+1)} = (F_1^{(1)} \cdots F_{k-1}^{(1)}) \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & D_k & 0 \\ 0 & 0 & I_{n-k-3} \end{pmatrix} (F_{k+2} \cdots F_{N-2}).$$

We have $D_k(4,1) = \hat{F}^{k+1}(k+3,k)$ and using (69) we conclude that the 4×4 unitary matrix D_k has the zero entry in the $(4,1)$ position. Hence the matrix D_k admits the factorization (55) with 3×3 unitary matrices $\mathcal{F}_k^{(1)}, \mathcal{F}_{k+1}^{(\tau)}$. Hence it follows that

$$\hat{F}^{(k+1)} = (F_1^{(1)} \cdots F_{k-1}^{(1)})(F_k^{(1)} F_{k+1}^{(\tau)})(F_{k+2} \cdots F_{N-2}),$$

which completes the proof of (64).

Thus the unitary matrix U_{n-2} admits the factorization

$$U_{n-2} = \hat{V}^{(n-2)} \cdot \hat{F}^{(N-2)}$$

and therefore the matrix $U^{(1)} = Q_{n-1}^H U_{n-2} Q_{n-2} Q_{n-1}$ has the form

$$U^{(1)} = (Q_{n-1}^H V_{n-1}^{(\tau)})(V_{n-2}^{(1)} V_2^{(1)})(F_1^{(1)} \cdots F_{n-3}^{(1)})(\tilde{F}_{n-2}^{(\tau)} Q_{n-2} Q_{n-1}).$$

From here using (56), (57) we obtain the desired factorization (63) for the unitary matrix $U^{(1)}$.

Next using (43) with $k = n-1$ and (56) we get

$$(\mathcal{V}_{n-1}^{(1)})^H \begin{pmatrix} z^{(1)} n-1 \\ \beta_n^{(1)} \end{pmatrix} = (\mathcal{V}_{n-1}^{(1)})^H \mathcal{G}_{n-1} \mathcal{G}_{n-1}^H \begin{pmatrix} p_{n-1}^{(1)} \\ p^{(1)}(N) \end{pmatrix} = \begin{pmatrix} \beta_{n-1}^{(1)} \\ 0 \end{pmatrix}$$

which together with (67) with $k = n-2$ means that the vector $\boldsymbol{z}^{(1)}$ is reduced by the matrix $V^{(1)} = V_{n-1}^{(1)} \cdots V_2^{(1)}$. $\square$

Remark 3.2 *Notice that in the subsequent iteration the parameter β_3 may be updated by the formula (66) with $k = 2$, i.e.,*

$$(\mathcal{V}_3^{(\tau)})^* \begin{pmatrix} z_3^{(\tau)} \\ \beta_4 \end{pmatrix} = \begin{pmatrix} \beta_3^{(1)} \\ 0 \end{pmatrix}.$$

In this case the Stage 2 of FastQR_ss *can be skipped.*

3.2 The Structured QR Iteration: The Double Shift Case

The double shift technique follows again the scheme (32)–(35). In this case, however, the matrices Q_k are of the type $Q_k = I_{k-1} \oplus \mathcal{G}_k \oplus I_{n-k-2}$, where $\mathcal{G}_k$ is a 3×3 unitary matrix and $1 \le k \le n-2$. The goal is to reduce the matrix $q(A) = (A - \alpha_1 I_n)(A - \alpha_2 I_n)$ in upper Hessenberg form, where α_1 and α_2 are

the eigenvalues of the 2×2 trailing block in the matrix A. At the beginning, Q_1 is chosen in such a way that it reduces the first column of $q(A)$.

The lower triangular part of the matrix A'_{k-1} obtained after $k-1$ steps of (32)–(35) is given by

$$\mathrm{tril}(A'_{k-1},0) = \begin{pmatrix} a_1^{(1)} & & & & & & & & \\ \gamma_1^{(1)} & a_2^{(1)} & & & & & & & \\ & \ddots & \ddots & & & & & & \\ & & \gamma_{k-2}^{(1)} & a_{k-1}^{(1)} & & & & & \\ & & & \delta_{k-1} & \hat{a}_k & & & & \\ & & & \theta_{k-1} & \hat{\gamma}_k & \hat{a}_{k+1} & & & \\ & & & \xi_{k-1} & \eta_k & \hat{\gamma}_{k+1} & a_{k+2} & & \\ & & & & & & \ddots & \ddots & \\ & & & & & & & \gamma_{n-1} & a_n \end{pmatrix}.$$

In general, the matrix $\mathcal{G}_k$ is chosen so that

$$\mathcal{G}_k^H \begin{pmatrix} \delta_{k-1} & \hat{a}_k & * \\ \theta_{k-1} & \hat{\gamma}_k & \hat{a}_{k+1} \\ \xi_{k-1} & \eta_k & \hat{\gamma}_{k+1} \end{pmatrix} = \begin{pmatrix} \beta_{k-1}^{(1)} & a_k^{(1)} & * \\ 0 & \hat{\gamma}_k^{(1)} & a_{k+1}^{(1)} \\ 0 & 0 & \gamma_{k+1}^{(1)} \end{pmatrix}.$$

Observe that, while in classical implicit QR algorithm with double shift the matrix $\mathcal{G}_k$ is only required to reduce the first column of the bulge, in this case the whole bulge should be eliminated in order to ensure that A_k attains again an upper Hessenberg form.

The bulge chasing step to be applied to the matrix U_k can be described as follows. Set $R_k = U_k(k+1:k+4, k:k+3)$, we have

$$R_k = \begin{pmatrix} \sigma_k^{(t)} & \rho_{k+1} & (\boldsymbol{g}_{k+1}^{(t)})^T \boldsymbol{h}_{k+2} & (\boldsymbol{g}_{k+1}^{(t)})^T B_{k+3} \boldsymbol{h}_{k+3} \\ z_{k+2}^{(t)} w_k^{(t)} & \sigma_{k+1}^{(\tau)} & (\boldsymbol{g}_{k+2}^{(\tau)})^T \boldsymbol{h}_{k+2} & (\boldsymbol{g}_{k+2}^{(\tau)})^T B_{k+3} \boldsymbol{h}_{k+3} \\ z_{k+3} w_k^{(t)} & z_{k+3} w_{k+1}^{(\tau)} & \sigma_{k+2} & (\boldsymbol{g}_{k+3})^T \boldsymbol{h}_{k+3} \\ z_{k+4} w_k^{(t)} & z_{k+4} w_{k+1}^{(\tau)} & z_{k+4} w_{k+2} & \sigma_{k+4} \end{pmatrix}$$

and next

$$\left(\begin{array}{c|c} \mathcal{G}_{k+1}^H & \\ \hline & 1 \end{array}\right) R_k \left(\begin{array}{c|c} \mathcal{G}_k & \\ \hline & 1 \end{array}\right) =$$

$$\begin{pmatrix} \sigma_k^{(1)} & * & * & * \\ z_{k+2}^{(t)} w_k^{(1)} & \sigma_{k+1}^{(t)} & \rho_{k+2} & (\boldsymbol{g}_{k+2}^{(t)})^T \boldsymbol{h}_{k+3} \\ z_{k+3}^{(\tau)} w_k^{(1)} & z_{k+3}^{(\tau)} w_{k+1}^{(t)} & \sigma_{k+2}^{(\tau)} & (\boldsymbol{g}_{k+3}^{(\tau)})^T \boldsymbol{h}_{k+3} \\ z_{k+4} w_k^{(1)} & z_{k+4} w_{k+1}^{(t)} & z_{k+4} w_{k+2}^{(\tau)} & \sigma_{k+4} \end{pmatrix}.$$

The updated vectors $\boldsymbol{z}^{(k+1)}$ and $\boldsymbol{w}^{(k+1)}$ are defined so that

$$\begin{pmatrix} w_k^{(1)} \\ w_{k+1}^{(t)} \\ w_{k+2}^{(\tau)} \end{pmatrix} = \mathcal{G}_k^H \begin{pmatrix} w_k^{(t)} \\ w_{k+1}^{(\tau)} \\ w_{k+2} \end{pmatrix}$$

and

$$\begin{pmatrix} z_k^{(1)} \\ z_{k+1}^{(t)} \\ z_{k+2}^{(\tau)} \end{pmatrix} = \mathcal{G}_{k+1}^H \begin{pmatrix} z_k^{(t)} \\ z_{k+1}^{(\tau)} \\ z_{k+2} \end{pmatrix}.$$

Once determined through the bulge chasing process, the matrices Q_j are used to compute the product representation for the updated U. Let us describe the general update step, using for the product representation the same notation as in the previous section. In order to exploit the bulge chasing properties of the matrices Q_j, consider an update step where the current unitary term U_k is multiplied by Q_k on the right and by Q_{k+1}^H on the left. We have:

$$\begin{aligned} & Q_{k+1}^H \cdot U_k \cdot Q_k \\ &= Q_{k+1}^H \cdot V_{n-1} \cdots V_{k+3} V_{k+2}^{(\tau)} V_{k+1}^{(t)} \cdots V_2^{(1)} \cdot F_1^{(1)} \cdots F_k^{(t)} F_{k+1}^{(\tau)} F_{k+2} F_{n-2} \cdot Q_k \\ &= V_{n-1}^{(k)} \cdots Q_{k+1}^H \cdot V_{k+3} \cdot V_{k+2}^{(\tau)} \cdot V_{k+1}^{(t)} \cdots V_2^{(1)} \cdot F_1^{(1)} \cdots F_{(t)}^{(k)} \cdot F_{k+1}^{(\tau)} \cdot F_{k+2} \cdot Q_k \cdots F_{n-2}. \end{aligned}$$

Let us examine more closely the product $P_{k+1} = Q_{k+1}^H \cdot V_{k+3} \cdot V_{k+2}^{(\tau)} \cdot V_{k+1}^{(t)}$. We have:

$$\left(\begin{array}{c|c} \mathcal{G}_{k+1}^H & \\ \hline & 1 \end{array}\right) \cdot \left(\begin{array}{c|c} I_2 & \\ \hline & \mathcal{V}_{k+3} \end{array}\right) \cdot \left(\begin{array}{c|c|c} 1 & & \\ \hline & \mathcal{V}_{k+2}^{(\tau)} & \\ \hline & & 1 \end{array}\right) \cdot \left(\begin{array}{c|c} \mathcal{V}_{k+1}^{(t)} & \\ \hline & I_2 \end{array}\right)$$

$$= \left(\begin{array}{c|c} \mathcal{G}_{k+1}^H & \\ \hline & 1 \end{array}\right) \cdot \begin{pmatrix} * & * & 0 & 0 \\ * & * & * & 0 \\ * & * & * & * \\ * & * & * & * \end{pmatrix} = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix},$$

that is, multiplying on the left by Q_{k+1}^H spoils the lower Hessenberg structure of $V_{k+3} \cdot V_{k+2}^{(\tau)} \cdot V_{k+1}^{(t)}$. Analogously, multiplying on the right by Q_k spoils the structure of the product $F_k^{(t)} \cdot F_{k+1}^{(\tau)} \cdot F_{k+2}$:

$$\left(\begin{array}{c|c} \mathcal{F}_k^{(t)} & \\ \hline & I_2 \end{array}\right) \cdot \left(\begin{array}{c|c|c} 1 & & \\ \hline & \mathcal{F}_{k+1}^{(\tau)} & \\ \hline & & 1 \end{array}\right) \cdot \left(\begin{array}{c|c} I_2 & \\ \hline & \mathcal{F}_{k+2} \end{array}\right) \cdot \left(\begin{array}{c|c} \mathcal{G}_k & \\ \hline & I_2 \end{array}\right)$$

$$\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \end{pmatrix} \cdot \left(\begin{array}{c|c} \mathcal{G}_k & \\ \hline & I_2 \end{array}\right) = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}.$$

In order to restore these structures, define a 3×3 unitary matrix $\mathcal{Z}_{k+2}$ such that $P_{k+1} \cdot Z_{k+2}^H$ is again lower Hessenberg, where $Z_{k+2} = I_{k+1} \oplus \mathcal{Z}_{k+2} \oplus I_{n-k-4}$, and consider the updated U_k in the form

$$\begin{aligned} & Q_{k+1}^H \cdot U_k \cdot Q_k \\ & = V_{n-1} \cdots Q_{k+1}^H \cdot V_{k+3} \cdot V_{k+2}^{(\tau)} \cdot V_{k+1}^{(t)} \cdot Z_{k+2}^H \cdots V_2^{(1)} \cdot \\ & \qquad \cdot F_1^{(1)} \cdots Z_{k+2} \cdot F_k^{(t)} \cdot F_{k+1}^{(\tau)} \cdot F_{k+2} \cdot Q_k \cdots F_{n-2}. \end{aligned}$$

Since $P_{k+1} \cdot Z_{k+2}^H$ is lower Hessenberg, it can be re-factorized as

$$P_{k+1} \cdot Z_{k+2}^H = V_{k+3}^{(\tau)} \cdot V_{k+2}^{(t)} \cdot V_{k+1}^{(1)},$$

where

$$V_{k+3}^{(\tau)} = I_{k+2} \oplus \mathcal{V}_{k+3}^{(\tau)} \oplus I_{n-k-4},$$
$$V_{k+2}^{(t)} = I_{k+1} \oplus \mathcal{V}_{k+2}^{(t)} \oplus I_{n-k-3},$$
$$V_{k+1}^{(1)} = I_k \oplus \mathcal{V}_{k+1}^{(1)} \oplus I_{n-k-2},$$

for suitable 2×2 unitary matrices $\mathcal{V}_{k+3}^{(\tau)}$, $\mathcal{V}_{k+2}^{(t)}$ and $\mathcal{V}_{k+1}^{(1)}$. Because of the way in which the matrices Q_j have been chosen, the matrix Z_{k+2} must also have the property of restoring the structure of the product $F_k^{(t)} \cdot F_{k+1}^{(\tau)} \cdot F_{k+2}$. It is therefore possible to compute 3×3 unitary matrices $\mathcal{F}_k^{(1)}$, $\mathcal{F}_{k+1}^{(t)}$ and $\mathcal{F}_{k+2}^{(\tau)}$ such that

$$Z_{k+2} \cdot F_k^{(t)} \cdot F_{k+1}^{(\tau)} \cdot F_{k+2} \cdot Q_k = F_k^{(1)} \cdot F_{k+1}^{(t)} \cdot F_{k+2}^{(\tau)},$$

where

$$F_k^{(1)} = I_{k-1} \oplus \mathcal{F}_k^{(1)} \oplus I_{n-k-2},$$
$$F_{k+1}^{(t)} = I_k \oplus \mathcal{F}_{k+1}^{(t)} \oplus I_{n-k-3},$$
$$F_{k+2}^{(\tau)} = I_{k+1} \oplus \mathcal{F}_{k+2}^{(\tau)} \oplus I_{n-k-1}.$$

4 Convergence Criteria and Deflation Techniques

Deflation is an important concept in the practical implementation of the QR iteration. Deflation amounts to setting a small subdiagonal element of the Hessenberg matrix to zero. This is called deflation because it splits the Hessenberg matrix into two smaller subproblems which may be independently refined further.

For the sake of clarity, suppose that after k iterations of the QR algorithm the subdiagonal entry $\beta_{n-s}^{(k)}$ of $A^{(k)}$ becomes small enough to be considered *negligible*. A customary criterion requires that such an entry be small compared with its diagonal neighbors

$$|\beta_{n-s}^{(k)}| \leq \boldsymbol{u} \cdot (|a_{n-s,n-s}^{(k)}| + |a_{n-s+1,n-s+1}^{(k)}|).$$

If this condition is satisfied, then $\beta_{n-s}^{(k)}$ is set to zero which makes $A^{(k)}$ a block upper triangular matrix:

$$A^{(k)} = \begin{pmatrix} A_1^{(k)} & \star \\ \bigcirc & A_2^{(k)} \end{pmatrix},$$

where $A_1^{(k)} \in \mathbb{C}^{(n-s)\times(n-s)}$ and $A_2^{(k)} \in \mathbb{C}^{s\times s}$. Now $A_1^{(k)}$ and $A_2^{(k)}$ can be reduced into upper triangular form separately. 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. Specifically, if $s = 1$ or $s = 2$ then the computed eigenvalues of $A_2^{(k)}$ may be taken to be eigenvalues of the initial matrix A.

The matrix $U^{(k)}$ can be partitioned accordingly with $A^{(k)}$:

$$U^{(k)} = \begin{pmatrix} U_1^{(k)} & \star \\ -\boldsymbol{u}_2^{(k)} \boldsymbol{w}_1^{(k)H} & U_2^{(k)} \end{pmatrix},$$

where

$$\boldsymbol{u}^{(k)} = \begin{pmatrix} \boldsymbol{u}_1^{(k)} \\ \boldsymbol{u}_2^{(k)} \end{pmatrix}, \ \boldsymbol{w}^{(k)} = \begin{pmatrix} \boldsymbol{w}_1^{(k)} \\ \boldsymbol{w}_2^{(k)} \end{pmatrix}, \quad \boldsymbol{u}_1^{(k)}, \boldsymbol{w}_1^{(k)} \in \mathbb{C}^{n-s}, \ \boldsymbol{u}_2^{(k)}, \boldsymbol{w}_2^{(k)} \in \mathbb{C}^s.$$

Observe that $U_1^{(k)}$ and $U_2^{(k)}$ are not generally unitary matrices and, therefore, the matrices $A_1^{(k)}$ and $A_2^{(k)}$ are specified in terms of the whole matrix $U^{(k)}$ as follows:

$$A_1^{(k)} = \begin{pmatrix} I_{n-s} & 0 \end{pmatrix} U^{(k)} \begin{pmatrix} I_{n-s} \\ 0 \end{pmatrix} + \boldsymbol{u}_1^{(k)} \boldsymbol{w}_1^{(k)H},$$

and

$$A_2^{(k)} = \begin{pmatrix} 0 & I_s \end{pmatrix} U^{(k)} \begin{pmatrix} 0 \\ I_s \end{pmatrix} + \boldsymbol{u}_2^{(k)} \boldsymbol{w}_2^{(k)H}.$$

These representations enable the fast QR iteration to be applied to the input matrices $A_1^{(k)}$ and $A_2^{(k)}$. Observe that, even though the whole matrix $U^{(k)}$ is required for representation purposes, working only on $A_1^{(k)}$ or $A_2^{(k)}$ means that only part of the generators are actually involved in the computation.

Remark 4.1 *The implicit QR method for eigenvalue computation relies heavily on the implicit Q theorem (see e.g. [12]), which applies to irreducible Hessenberg matrices. As a consequence, deflation is a crucial step in implicit QR and it is necessary to perform it whenever possible.*

A more subtle form of deflation should also be applied when two consecutive subdiagonal entries of $A^{(k)}$ become small (typically, small enough for their product to be less than or equal to the machine epsilon). In this case, the next bulge chasing process should only be applied to the submatrix $A^{(k)}(r+1 : n, r+1 : n)$, where r is the index such that the small subdiagonal entries are $A^{(k)}(r+1, r)$ and $A^{(k)}(r+2, r+1)$. Indeed, it turns out that beginning the bulge chasing process from the $(r+1)$-st row introduces negligible perturbations of the Hessenberg structure in the entries of indices $(r+2, r)$ (and $(r+3, r)$ in the double shift case).

This is an idea that dates back to Francis and is explained in detail in Wilkinson's classical book [18], Chapter 8 Section 38. It is however often overlooked in textbooks, so we recall it briefly here. Let A be an upper Hessenberg matrix having small elements ϵ_1 and ϵ_2 in positions $(r+1, r)$ and $(r+2, r+1)$, for a certain index r, and consider the following partition:

$$A = \left(\begin{array}{cc|ccc} * & * & * & * & * \\ * & * & * & * & * \\ \hline 0 & \epsilon_1 & * & * & * \\ 0 & 0 & \epsilon_2 & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * \end{array}\right) = \left(\begin{array}{c|c} X & Y \\ \hline E & W \end{array}\right). \tag{70}$$

With this notation we have:

Proposition 4.2 *Let μ be any eigenvalue of W. Then μ is also an eigenvalue of a matrix A', of the same size as A, which differs from A only by an element $\epsilon_1\epsilon_2/(A(r+1, r+1) - \mu)$ in position $(r+2, r)$.*

As a consequence of Proposition 4.2, if the product $\epsilon_1\epsilon_2$ is small enough then the next QR iteration can be applied to the submatrix W, ignoring the fact that E is not numerically zero and performing in fact a sort of deflation.

In practical implementations, the search for a suitable partition is generally performed from bottom up, so that one always works on the smallest possible trailing submatrix. More precisely, before starting each QR iteration on the current iterate $A^{(k)}$, the following procedure is performed:

- look for the smallest index s such that $A^{(k)}(n-s+1, n-s)$ is negligible;
- if $s = 1$ (or $s = 2$ for the double shift case), then one or two eigenvalues have been found; exit procedure;
- else look for the largest index r, with $n-s \leq r \leq n-2$ (or $n-s \leq r \leq n-3$ for the double shift case), such that beginning the bulge chasing process from the $(r+1)$-st row introduces negligible perturbations of the Hessenberg structure in the entries of indices $(r+2, r)$ (and $(r+3, r)$ for the double shift case);
- perform bulge chasing on the trailing submatrix $A^{(k)}(r+1 : n, r+1 : n)$.

See also [14] for practical criteria and suggestions in the classical (non-structured) case.

5 Numerical Experiments

The algorithms for eigenvalue computation described in the previous sections have been implemented in Fortran 95, for the single and double shift versions. These programs deal with the particular case of the companion matrix A associated with a given polynomial $P(x) = \sum_{k=0}^{n} c_k x^k$ and have been used to test the efficiency and stability of the proposed algorithms on several examples. The software is available upon request to the authors.

The following definitions of errors are used in tests for stability:

- Absolute forward error (a.f.e.): this is the absolute distance between the eigenvalues computed by the structured method (which will be called here FASTQR for reference purposes) and the eigenvalues computed by LAPACK (routines ZGEEV and DGEEV), which are assumed to be correct. The absolute forward error is computed as the ∞-norm of the difference between the two output vectors obtained from FASTQR and LAPACK and sorted by decreasing absolute value. An estimate for this error, based on theoretical arguments, is given by the quantity $\delta = \epsilon \cdot \|A\|_2 \cdot \max\{\texttt{condeig(A)}\}$, where ϵ is the machine epsilon and, using Matlab terminology, `condeig(A)` is the vector of eigenvalue condition numbers for the matrix A.
- Matrix relative backward error: it is defined as

$$\text{m.b.e.} = \frac{\|Q_a^* A_0 Q_a - (V_f F_f - p_f q_f^T)\|_\infty}{\|A_0\|_\infty},$$

  where $A_0 = A$ is the initial matrix in the QR iteration process, Q_a is the accumulated unitary similarity transformation and V_f, F_f, p_f, q_f are generators for the product structure of the final iterate A_f.
- Coefficient relative backward error: it is a vector whose entries are defined as

$$(\text{c.b.e})_k = \frac{|\tilde{c}_k - c_k|}{|c_k|} \qquad \text{if} \quad c_k \neq 0,$$

  where $\{\tilde{c}_k\}_{k=1,\ldots n}$ are the coefficients, computed in high precision, of the polynomial whose roots are the eigenvalues of A given by our structured method.

We start with some examples of the behavior of the single shift version of FASTQR applied to complex polynomials. Most examples in this section are taken from [4] and [6].

Example 5.1 *Given a positive integer n, define $P(x) = \sum_{k=0}^{n}(a_k + ib_k)x^k$, where a_k and b_k are uniformly distributed random numbers in $[-1, 1]$.*

As discussed in [13] randomly generated polynomials can be useful for gathering statistics on the performance of the program with respect to computational

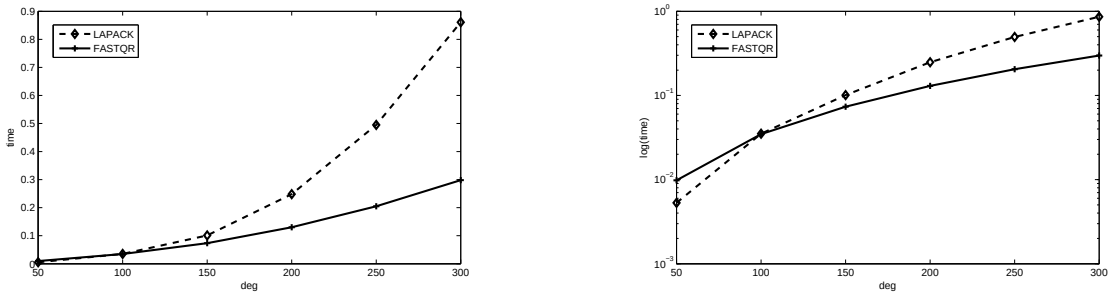

Fig. 1. Comparison of running times (Example 5.1).

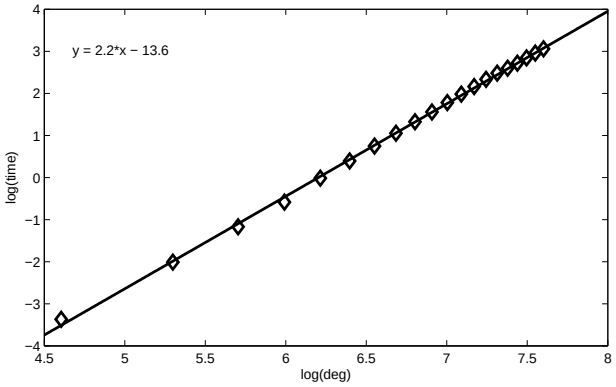

Fig. 2. Experimental estimate of time growth (Example 5.1).

speed and accuracy. Figure 1 shows, on the left, running times for FASTQR and LAPACK on random polynomials generated as in Example 5.1, for degrees ranging between 50 and 350. On the right we report the log-scale plot of the same data, where it can be seen that FASTQR becomes faster than LAPACK for polynomials of degree about 100.

Figure 2 is a log-log plot of the running times of FASTQR on polynomials of degrees ranging between 50 and 2000. A linear fit of this plot shows that the growth of running time is indeed quadratic in n, since the slope of the line is about 2.2. The number of required QR iteration is usually less than 3 per eigenvalue.

Polynomials in Example 5.1 are usually rather well-conditioned and the behavior of FASTQR is satisfactory both for backward and forward stability (more details are given in Example 5.4 for the double shift version). In the next example, coefficients are still random but chosen with an unbalanced distribution:

Example 5.2 *Given a positive integer n, define $P(x) = x^n + \sum_{j=0}^{n-1} a_j x^j$, with $a_j = u_j \cdot 10^{v_j}$, where $|u_j|$ is a uniformly distributed random number in $[-1, 1]$ and v_j is a uniformly distributed random number in $[-5, 5]$.*

The behavior of FASTQR on such polynomials is shown in the following table:

n	m.b.e.	a.f.e.	δ
50	4.91×10^{-15}	6.24×10^{-10}	2.21×10^{-8}
100	6.63×10^{-15}	6.95×10^{-10}	1.61×10^{-8}
150	7.02×10^{-15}	3.19×10^{-10}	1.12×10^{-7}
500	7.43×10^{-15}	8.30×10^{-10}	8.28×10^{-7}
1000	1.44×10^{-14}	1.47×10^{-9}	1.59×10^{-6}

Backward errors are small, which ensures backward stability. The growth of the forward error mirrors closely the behavior of the quantity δ, therefore the output of our method is consistent with theoretical expectations.

Example 5.3 *Given a positive even integer n, define $P(x) = \prod_{k=-n/2}^{n/2-1}(x - \frac{2(k+0.5)}{n-1} - \boldsymbol{i}\sin(\frac{2(k+0.5)}{n-1}))$, where $\boldsymbol{i}^2 = -1$. This is the monic polynomial with zeros equally spaced on the curve $x = t + \boldsymbol{i}\sin(\pi t)$ with $-1 \leq t \leq 1$.*

Results for this class of test polynomials are shown in the following table:

n	a.f.e.	δ
8	9.84×10^{-15}	2.11×10^{-14}
16	8.31×10^{-13}	4.86×10^{-11}
32	$.76 \times 10^{-8}$	7.83×10^{-4}

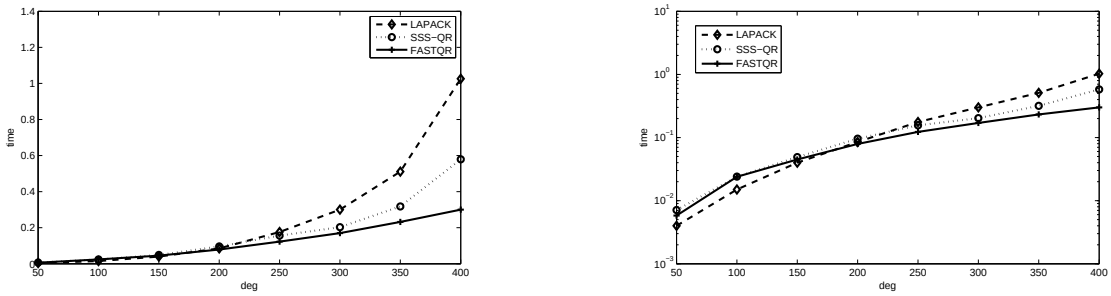

Fig. 3. Comparison of running times (Example 5.4).

Next, let us consider some examples with real polynomials, to which we apply the double shift version of FASTQR.

Example 5.4 *Given a positive integer n, define $P(x) = \sum_{k=0}^{n} c_k x^k$, where c_k is a random number uniformly distributed in $[-1, 1]$.*

As in Example 5.1, we use these polynomials to compare running times and check that the growth of the running time is indeed quadratic. In this case we are also able to introduce a comparison with the structured implicit QR method proposed in [6], denoted as SSS-QR, a Fortran 95 implementation of which is available on-line for the case of real polynomials. Figure 3 shows on the left the running times for FASTQR, LAPACK and SSS-QR for polynomials of degrees ranging between 50 and 400 and on the right the corresponding logarithmic plot. FASTQR becomes faster than LAPACK for polynomials of degrees between 150 and 200.

Figure 4 shows a log-log plot of the running times of FASTQR for polynomials of degrees ranging between 50 and 2000, with a linear fit; the slope here is about 2.02. The number of double shift iterations is generally less than 1.5 per eigenvalue. The logarithms of the absolute forward errors computed for these same polynomials are plotted in Figure 5. It is interesting to notice that, while for degrees up to 1000 there is a growth of the forward errors (which is however expected), for degrees higher than 1000 the errors stay roughly the same. This is also the case for experiments made for degrees 3000 and 5000. This suggests that FASTQR reliably computes eigenvalues even for very high degrees.

Example 5.5 (Roots of 1) *Given a positive integer n, define $P(x) = x^n - 1$.*

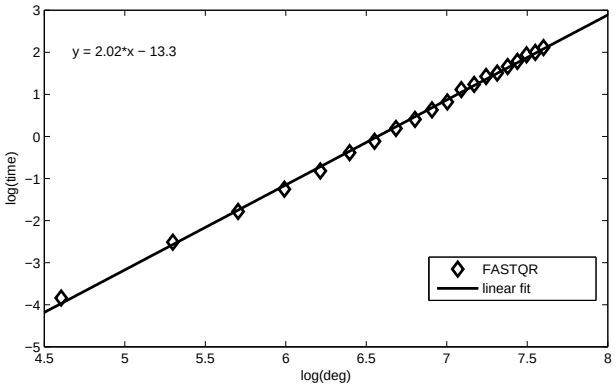

Fig. 4. Experimental estimate of time growth (Example 5.4).

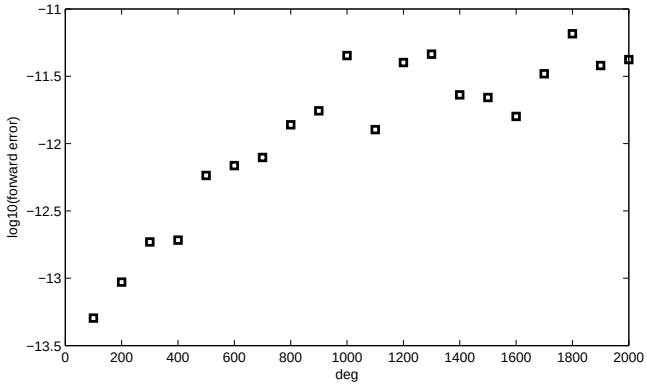

Fig. 5. Absolute forward errors (Example 5.4).

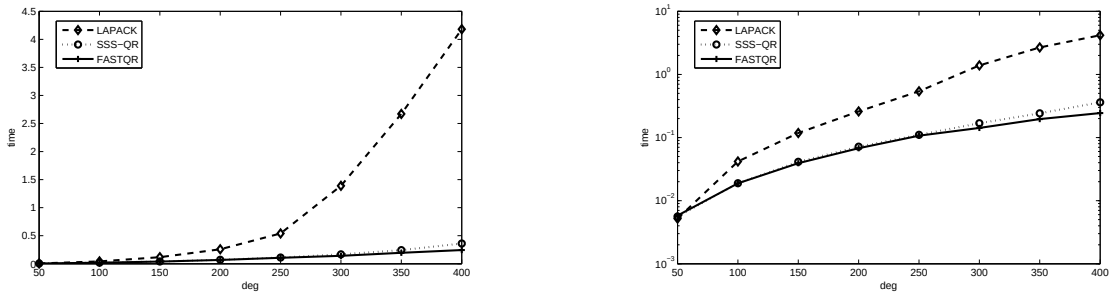

Fig. 6. Comparison of running times (Example 5.5).

We use these polynomials to compare the running times of FASTQR to the running times of LAPACK and SSS-QR; the regular and logarithmic plot for polynomials of degrees ranging from 50 to 400 are found in Figure 6. Here FASTQR becomes faster than LAPACK for $n \geq 50$. The computation of the roots of unity with FASTQR is generally very accurate (forward errors are about 10^{-15} for the polynomials used in this experiment).

Example 5.6 (Multiple roots) *Given a positive integer n, define $P(x) = (\sum_{k=0}^{20} c_k x^k) \cdot (x-1)^n$, where c_k is a random number in $[-1, 1]$.*

In this case, the computation of roots with multiplicity 1 is very accurate, whereas the multiple root 1 suffers from ill conditioning. The following table shows forward and backward errors for several values of n. For the coefficient backward error we give an interval containing all entries of the error vector.

n	a.f.e.	c.b.e.	δ
1	2.81×10^{-15}	$(0, 8 \times 10^{-13})$	3.23×10^{-15}
2	1.61×10^{-7}	$(0, 5 \times 10^{-13})$	8.15×10^{-7}
3	9.33×10^{-7}	$(0, 10^{-11})$	1.04×10^{-4}
4	3.91×10^{-5}	$(0, 3 \times 10^{-11})$	1.34×10^{-2}

Example 5.7 *Define $P(x) = x^{20} + x^{19} + \ldots + x + 1$.*

We obtain here a.f.e.$= 3.58 \times 10^{-15}$; the entries in the coefficient backward

error are in the range $(0, 8 \times 10^{13})$, which is roughly the same result given in [6] and the same as one would obtain using the Matlab function `roots`.

Example 5.8 *Define $P(x)$ as the monic polynomial whose roots are, in Matlab notation,* $[-2.1 : 0.2 : 1.7]$.

For this example we have a forward error of 1.21×10^{-11}, which is consistent with the estimate given by $\delta = 1.76 \times 10^{-8}$. The entries in the coefficient backward error are in the range $(0, 4 \times 10^{12})$, which is again comparable to the result given in [6].

6 Conclusion

The analysis and development of efficient numerical methods for eigenvalue computation of rank structured matrices constitutes one of the major challenges in the field of numerical linear algebra. In this paper we have presented an implicit version of the QR algorithm for companion matrices designed in [4]. The novel method is conceptually simple, computationally fast and numerically stable. The complexity is an order of magnitude less than the customary implementation of the QR algorithm for Hessenberg matrices in LAPACK. Experimental results are also reported to compare the performances of our method with other existing fast eigensolvers for companion matrices.

References

[1] G. S. Ammar, W. B. Gragg, and C. He. An efficient QR algorithm for a Hessenberg submatrix of a unitary matrix. In *New directions and applications in control theory*, volume 321 of *Lecture Notes in Control and Inform. Sci.*, pages 1–14. Springer, Berlin, 2005.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors. *Templates for the solution of algebraic eigenvalue problems*, volume 11 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. A practical guide.

[3] D. A. Bini, Y. Eidelman, L. Gemignani, and I. Gohberg. The unitary completion and QR iterations for a class of structured matrices. *Math. Comp.*, 77(261):353–378, 2008.

[4] D. A. Bini, Y. Eidelman, L. Gemignani, and I. Gohberg. Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices. *SIAM J. Matrix Anal. Appl.*, 29(2):566–585, 2007.

[5] D. A. Bini, L. Gemignani, and V. Y. Pan. Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations. *Numer. Math.*, 100(3):373–408, 2005.

[6] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu. A fast QR algorithm for companion matrices. In *Recent advances in matrix and operator theory*, volume 179 of *Oper. Theory Adv. Appl.*, pages 111–143. Birkhäuser, Basel, 2008.

[7] Y. Eidelman, L. Gemignani, and I. Gohberg. On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms. *Linear Algebra Appl.*, 420:86–101, 2007.

[8] Y. Eidelman, L. Gemignani, and I. Gohberg. Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations. *Numer. Algorithms*, 47(1):253–273, 2008.

[9] Y. Eidelman, I. Gohberg, and V. Olshevsky. The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order. *Linear Algebra Appl.*, 404:305–324, 2005.

[10] J. G. F. Francis. The QR transformation: a unitary analogue to the LR transformation. I. *Comput. J.*, 4:265–271, 1961/1962.

[11] J. G. F. Francis. The QR transformation. II. *Comput. J.*, 4:332–345, 1961/1962.

[12] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[13] M. A. Jenkins and J. F. Traub. Principles for testing polynomial zerofinding programs. *ACM Trans. Math. Software*, 1:26–34, 1975.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes*. Cambridge University Press, Cambridge, third edition, 2007. The art of scientific computing.

[15] F. Tisseur. Backward stability of the QR algorithm. Technical Report 239, UMR 5585 Lyon Saint-Etienne, 1996.

[16] M. Van Barel, P. Van Dooren and R. Vandebril. Computing the eigenvalues of a companion matrix. *Slides of the talk on the conference Structured Linear Algebra Problems: Analysis, Algorithms, and Applications, Cortona, Italy*, September 15-19, 2008. http://bezout.dm.unipi.it//Cortona08/slides/Van Barel.pdf.

[17] D. S. Watkins. *Fundamentals of matrix computations*. Pure and Applied Mathematics (New York). Wiley-Interscience [John Wiley & Sons], New York, 2002. Second editon.

[18] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. The Oxford University Press, 1965.