Operator Theory:
Advances and Applications, Vol. 179, 111–143

A Fast QR Algorithm for Companion Matrices

Shiv Chandrasekaran, Ming Gu, Jianlin Xia and Jiang Zhu

Abstract. It has been shown in [4, 5, 6, 31] that the Hessenberg iterates of a companion matrix under the QR iterations have low off-diagonal rank structures. Such invariant rank structures were exploited therein to design fast QR iteration algorithms for finding eigenvalues of companion matrices. These algorithms require only $O(n)$ storage and run in $O(n^2)$ time where n is the dimension of the matrix. In this paper, we propose a new $O(n^2)$ complexity QR algorithm for real companion matrices by representing the matrices in the iterations in their sequentially semi-separable (SSS) forms [9, 10]. The bulge chasing is done on the SSS form QR factors of the Hessenberg iterates. Both double shift and single shift versions are provided. Deflation and balancing are also discussed. Numerical results are presented to illustrate both high efficiency and numerical robustness of the new QR algorithm.

Mathematics Subject Classification (2000). 65F15, 65H17.

Keywords. Companion matrices, sequentially semi-separable matrices, structured QR iterations, structured bulge chasing, Givens rotation swaps.

1. Introduction

After nearly forty years since its introduction [18, 19], the QR algorithm is still the method of choice for small or moderately large nonsymmetric eigenvalue problems $Ax = \lambda x$ where A is an $n \times n$ matrix. At the moment of this writing, moderately large eigenvalue problems refer to matrices of order $1,000$ or perhaps a bit higher. The main reason for such a limitation in problem size is because the algorithm runs in $O(n^3)$ time and uses $O(n^2)$ storage.

The success of the algorithm lies on doing QR iterations repeatedly, which under mild conditions [29] leads to Schur form convergence. However, for a general nonsymmetric dense matrix A, one QR decomposition itself already takes $O(n^3)$ operations, so even if we are lucky enough to do only one iteration per eigenvalue, the cost would still be $O(n^4)$. To make the algorithm practical, it is necessary to first reduce A into an upper Hessenberg matrix H and then carry out QR iterations on H accordingly. It is also important to incorporate a suitable shift strategy (since

QR iteration is implicitly doing inverse iteration), which can dramatically reduce the number of QR iterations needed for convergence.

The rationale for reducing A to H is that the Hessenberg form is invariant under QR iterations. Such Hessenberg invariance structure enables us to implement QR iterations implicitly and efficiently by means of structured bulge chasing. In practice, with the use of shifts, convergence to the Schur form occurs in $O(n)$ bulge chasing passes, each pass consists of $O(n)$ local orthogonal similarity transformations, and each local similarity transformation takes $O(n)$ operations. Therefore the total cost of the algorithm is $O(n^3)$ operations. This new algorithm has been tested for many different types of examples and is stable in practice.

In this paper we consider the eigenvalue computation of a real companion matrix of the form

$$C = \begin{pmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \in \mathbb{R}^{n\times n}. \tag{1}$$

Since the eigenvalues of C coincide with the zeros of a real univariate polynomial

$$p(x) = x^n - a_1 x^{n-1} - \cdots - a_{n-1}x - a_n, \tag{2}$$

algorithms for computing matrix eigenvalues can be used to approximate the zeros of $p(x)$. In fact, the Matlab function `roots` finds the zeros of $p(x)$ by applying the implicit shift QR algorithm to C_0, a suitably balanced version of C by means of a diagonal scaling (note that C_0 is not necessarily a companion matrix). The algorithm costs $O(n^3)$ operations as we mentioned.

The $O(n^3)$ cost and $O(n^2)$ storage are still expensive for a large n. In fact, it is possible to improve the performance of QR iterations by exploiting additional invariance structures of the Hessenberg iterates of C under QR iterations. It has been shown independently in [4] and in [5, 6] that the Hessenberg iterates of a companion matrix preserve an off-diagonal low-rank structure, called sequentially semi-separable structure and semi-separable structure, respectively. This fact was then exploited to design companion eigensolvers which require only $O(n^2)$ time and $O(n)$ storage.

In this paper, we present a new $O(n^2)$ QR variant algorithm for the real companion matrix, with experiments showing numerical stability. We implement both the single shift and double shift QR iterations with compact sequentially semi-separable structures. Instead of working on the similarity transformations of C, we work on the QR factors of these matrices. A swapping strategy for Givens rotation matrices is used to efficiently conduct structured bulge chasing. To maintain compact structured forms of those QR factors we introduce a structure recovery technique. We also provide a structured balancing strategy.

The paper is organized as follows. In Section 2, we describe the sequentially semi-separable representation and some related operations including matrix additions and matrix-matrix multiplications. In Section 3, we adopt the approach in [4] to prove why all Hessenberg iterates of C have off-diagonal blocks with ranks never exceeding 3. Similar off-diagonal rank results can be easily extended to the QR factors Q and R in the QR iterations. Thus Section 4 shows the representations of Q and R in compact SSS forms. In Section 5, we describe the deflation technique and the convergence criterion of the new QR algorithm, and then by using a concrete 5×5 matrix example, we demonstrate how to implicitly do both single and double shift QR iterations based on the compact representations of Q and R. Balancing strategy, which preserves the semi-separable structure, is discussed in Section 6. In Section 7, we present numerical results to demonstrate the performance. Finally, Section 8 draws some concluding remarks.

2. SSS representation

In this section we lay out some necessary background information about sequentially semi-separable (SSS) representations [9, 10]. Closely related matrix structures include quasiseparable matrices (e.g., [14, 15]), hierarchically semi-separable matrices [8], etc. Both the name "SSS" and "quasiseparable" refer to the same type of matrices. Related matrix properties and operations are discussed in the above references. Here we use SSS representations and some associated operations in [9, 10]. Similar results also appear in [14]. They will be used in our fast structured QR iterations.

2.1. SSS notations

We say that $A \in \mathbb{R}^{n\times n}$ is in *SSS* form if it is represented as

$$A = (A_{ij}), \quad \text{where } A_{ij} \in \mathbb{R}^{m_i \times m_j},\ A_{ij} = \begin{cases} \mathcal{D}_i & \text{if } i = j, \\ \mathcal{U}_i \mathcal{W}_{i+1} \cdots \mathcal{W}_{j-1} \mathcal{V}_j^T & \text{if } i < j, \\ \mathcal{P}_i \mathcal{R}_{i-1} \cdots \mathcal{R}_{j+1} \mathcal{Q}_j^T & \text{if } i > j. \end{cases} \tag{3}$$

Here the empty products are treated as the identity matrices, and the *partitioning sequence* $\{m_i\}_{i=1}^r$ satisfies $\sum_{i=1}^r m_i = n$, with r being the number of block rows (or columns) of the partitioning scheme. The SSS *generators* $\{\mathcal{D}_i\}_{i=1}^r$, $\{\mathcal{U}_i\}_{i=1}^{r-1}$, $\{\mathcal{V}_i\}_{i=2}^r$, $\{\mathcal{W}_i\}_{i=2}^{r-1}$, $\{\mathcal{P}_i\}_{i=2}^r$, $\{\mathcal{Q}_i\}_{i=1}^{r-1}$ and $\{\mathcal{R}_i\}_{i=2}^{r-1}$ are real matrices with dimensions specified in Table 2.1.

$\mathcal{D}_i$	$\mathcal{U}_i$	$\mathcal{V}_i$	$\mathcal{W}_i$	$\mathcal{P}_i$	$\mathcal{Q}_i$	$\mathcal{R}_i$
$m_i \times m_i$	$m_i \times k_i$	$m_i \times k_{i-1}$	$k_{i-1} \times k_i$	$m_i \times l_i$	$m_i \times l_{i+1}$	$l_{i+1} \times l_i$

TABLE 1. *Dimensions of matrices in* (3).

To illustrate the compactness of this SSS representation when the off-diagonal blocks of A have small ranks, assume $m_i = k_i = l_i = p \ll n$, then we only need to

store the SSS generators of A with about $7rp^2 (= 7pn)$ working precision numbers instead of storing every entry of A with n^2 numbers.

It should be noted that the SSS structure of a given matrix A depends on the partitioning sequence $\{m_i\}_{i=1}^r$. Different sequences will lead to different representations.

The power of SSS representation for matrices with low-rank off-diagonal blocks has been shown in [9, 10, 11, 30], where fast and stable linear system solvers based on SSS representation were designed with applications to many relevant engineering problems. In [9, 10], algorithms for SSS matrix operations have been systematically introduced, including constructions of the SSS representations, (LU-like) factorizations of SSS matrices, fast SSS matrix additions and fast matrix-matrix multiplications, etc. For our purpose of designing a new QR iteration method for companion matrices, we need to use two important SSS matrix operations, SSS addition and SSS multiplication. We present the results from [9, 10] without proofs.

2.2. SSS addition

Let A and B be two SSS matrices that are conformally partitioned, that is, $m_i(A) = m_i(B)$ for $i = 1, \ldots, r$. Then their sum $A + B$ is an SSS matrix with representation given by the following SSS generators [9, 10]:

$$
\begin{aligned}
&\mathcal{D}_i(A+B) = \mathcal{D}_i(A) + \mathcal{D}_i(B),\\
&\mathcal{U}_i(A+B) = \begin{pmatrix} \mathcal{U}_i(A) & \mathcal{U}_i(B) \end{pmatrix}, \qquad \mathcal{V}_i(A+B) = \begin{pmatrix} \mathcal{V}_i(A) & \mathcal{V}_i(B) \end{pmatrix},\\
&\mathcal{W}_i(A+B) = \begin{pmatrix} \mathcal{W}_i(A) & 0 \\ 0 & \mathcal{W}_i(B) \end{pmatrix},\\
&\mathcal{P}_i(A+B) = \begin{pmatrix} \mathcal{P}_i(A) & \mathcal{P}_i(B) \end{pmatrix}, \qquad \mathcal{Q}_i(A+B) = \begin{pmatrix} \mathcal{Q}_i(A) & \mathcal{Q}_i(B) \end{pmatrix},\\
&\mathcal{R}_i(A+B) = \begin{pmatrix} \mathcal{R}_i(A) & 0 \\ 0 & \mathcal{R}_i(B) \end{pmatrix}.
\end{aligned}
$$

Remark 2.1. Note that the computed SSS representation of the sum might be inefficient in the sense that the dimensions of the SSS generators are increasing additively, whereas in some cases the real ranks of the off-diagonal blocks might be far smaller. Ideally, these formulas should be followed by some sort of rank-reduction or compression step [9, 10].

2.3. SSS multiplication

Let A and B be two SSS matrices that are conformally partitioned. Define forward and backward recursions

$$
\begin{aligned}
&S_1 = 0, \quad S_{i+1} = \mathcal{Q}_i^T(A)\mathcal{U}_i(B) + \mathcal{R}_i(A)S_i\mathcal{W}_i(B), \quad \text{for } i = 1, 2, \ldots, r-1,\\
&T_n = 0, \quad T_{i-1} = \mathcal{V}_i^T(A)\mathcal{P}_i(B) + \mathcal{W}_i(A)T_i\mathcal{R}_i(B), \quad \text{for } i = r, r-1, \ldots, 2.
\end{aligned}
$$

Then the SSS generators of the matrix $A \cdot B$ can be computed through the following formulas [9, 10]:

$$\begin{aligned}
\mathcal{D}_i(A \cdot B) &= \mathcal{D}_i(A)\mathcal{D}_i(B) + \mathcal{P}_i(A)S_i\mathcal{V}_i^T(B) + \mathcal{U}_i(A)T_i\mathcal{Q}_i^T(B),\\
\mathcal{U}_i(A \cdot B) &= \begin{pmatrix} \mathcal{D}_i(A)\mathcal{U}_i(B) + \mathcal{P}_i(A)S_i\mathcal{W}_i(B) & \mathcal{U}_i(A) \end{pmatrix},\\
\mathcal{V}_i(A \cdot B) &= \begin{pmatrix} \mathcal{V}_i(B) & \mathcal{D}_i^T(B)\mathcal{V}_i(A) + \mathcal{Q}_i(B)T_i^T\mathcal{W}_i^T(A) \end{pmatrix},\\
\mathcal{W}_i(A \cdot B) &= \begin{pmatrix} \mathcal{W}_i(B) & 0 \\ \mathcal{V}_i^T(A)\mathcal{U}_i(B) & \mathcal{W}_i(A) \end{pmatrix},\\
\mathcal{P}_i(A \cdot B) &= \begin{pmatrix} \mathcal{D}_i(A)\mathcal{P}_i(B) + \mathcal{U}_i(A)T_i\mathcal{R}_i(B) & \mathcal{P}_i(A) \end{pmatrix},\\
\mathcal{Q}_i(A \cdot B) &= \begin{pmatrix} \mathcal{Q}_i(B) & \mathcal{D}_i^T(B)\mathcal{Q}_i(A) + \mathcal{V}_i(B)S_i^T\mathcal{R}_i^T(A) \end{pmatrix},\\
\mathcal{R}_i(A \cdot B) &= \begin{pmatrix} \mathcal{R}_i(B) & 0 \\ \mathcal{Q}_i^T(A)\mathcal{P}_i(B) & \mathcal{R}_i(A) \end{pmatrix}.
\end{aligned}$$

Remark 2.2. In the case where $m_i = k_i = l_i = p$, the total operation count of this fast multiplication algorithm is at most $40p^3n$, contrasting with $2n^3$ flops for doing ordinary matrix-matrix multiplication.

3. Invariant off-diagonal low-rank structure

The classical Hessenberg QR algorithm for finding eigenvalues computes a series of Hessenberg matrices H_k which are orthogonally similar to C in (1):

$$\begin{aligned}
&H^{(0)} = C,\\
&H^{(k)} = Q^{(k)}R^{(k)},\ H^{(k+1)} = R^{(k)}Q^{(k)},\ k = 0, 1, 2, \ldots
\end{aligned}$$

Generally, shifts are used in the iterations. It has been shown independently in [4] and [5] that each such Hessenberg matrix H_k (real or complex) maintains off-diagonal low-rank structures. More precisely, the following result holds.

Theorem 3.1. [4, 5] $\max_{1 \le j < n} \operatorname{rank}(H^{(k)}(1 : j, j + 1 : n)) \le 3$.

In what follows, we concentrate on real companion matrices. The proof of the theorem relies on the results in the following two lemmas [4].

Lemma 3.2. *For any Hessenberg matrix $H^{(k)}$ in the Hessenberg QR iterations, there exist an orthogonal matrix $Z^{(k)} \in \mathbb{R}^{n \times n}$ and two vectors $x^{(k)}$, $y^{(k)} \in \mathbb{R}^n$ so that*

$$H^{(k)} = Z^{(k)} + x^{(k)}y^{(k)T}. \tag{4}$$

($H^{(k)}$ is an orthogonal-plus-rank-one structure.)

It suffices to establish the equation for $H^{(0)}$ since the structure of a low-rank modification to an orthogonal matrix is preserved under orthogonal similarity

transformations. For $H^{(0)} = C$, we can write

$$C = \begin{pmatrix} 0 & 0 & \dots & 0 & \pm 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \mp 1 \end{pmatrix}$$
$$\equiv Z^{(0)} + x^{(0)} y^{(0)T}.$$

For convenience, we choose the sign of the $(1, n)$-entry of $Z^{(0)}$ so that $\det(Z^{(0)}) = 1$.

Lemma 3.3. *An orthogonal matrix Z is rank-symmetric* [4]*, in the sense that for any* 2*-by-*2 *block partitioning*

$$Z = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix},$$

where Z_{11} and Z_{22} are square, we have $\operatorname{rank}(Z_{12}) = \operatorname{rank}(Z_{21})$.

This is a direct outcome of the CS decomposition (see [17]). Actually not only $\operatorname{rank}(Z_{12}) = \operatorname{rank}(Z_{21})$, Z_{12} and Z_{21} have the same singular values as well. Therefore, we can expect that a slightly perturbed orthogonal matrix is still numerically rank-symmetric.

Now let us prove Theorem 3.1. For simplicity of the notation, we drop the superscript (k) from (4) in the rest of this section.

Proof of Theorem 3.1. Write $L = xy^T$. According to Lemma 3.2, we have $H = Z + L$. Partition H as

$$H = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix},$$

where H_{11} and H_{22} are square, and partition Z and L conformally. Then H_{21} has rank at most 1, since there is only one possible nonzero in its upper right corner. In addition,

$$\begin{aligned} |\operatorname{rank}(H_{12}) - \operatorname{rank}(H_{21})| &= |(\operatorname{rank}(H_{12}) - \operatorname{rank}(Z_{12})) - (\operatorname{rank}(H_{21}) - \operatorname{rank}(Z_{12}))| \\ &\le |\operatorname{rank}(H_{12}) - \operatorname{rank}(Z_{12})| + |\operatorname{rank}(H_{21}) - \operatorname{rank}(Z_{21})| \\ &\qquad \text{(since } Z \text{ is rank-symmetric)} \\ &\le \operatorname{rank}(L_{12}) + \operatorname{rank}(L_{21}) \\ &\le 2 \cdot \operatorname{rank}(L) = 2. \end{aligned}$$

Thus

$$\operatorname{rank}(H_{12}) \le \operatorname{rank}(H_{21}) + 2 \le 3. \qquad \square$$

Theorem 3.1 indicates that all H in the QR iterations have low-rank off-diagonal blocks. Such a low-rank structure admits a compact representation for H.

Bini, Eidelman, et al. [6] take advantage of this property and represent each H in a quasiseparable form which can be represented by a linear number of parameters. Similarly, the new QR algorithm proposed by Bindel, Chandresekaran, et al. in [4] exploits this structure by writing the Hessenberg iterate H in terms of its SSS representation. Both type of schemes provide explicit formulas for QR iterations with single shifts.

Because during the structured bulge chasing passes only linear memory space and only local updating for the quasiseparable or SSS generators of H are required, those new QR algorithms are able to achieve $O(n^2)$ complexity and $O(n)$ storage. To maintain the compact quasiseparable or SSS representations for H, the algorithm in [6] involves some compression schemes, and the algorithm in [4] incurs merging and splitting SSS representations repeatedly during each bulge chasing pass.

In this paper we propose a different approach for QR iterations: instead of working explicitly on the compact representations of H, we choose to work on Q and R directly, and in the meantime, to maintain compact representations for them, where Q and R are QR factors of H. This allows more flexibility in handling the structured QR iterations. Partly because of this reason we are able to provide both single shift and double shift QR iterations, whereas [4] and [6] only provide single shift versions.

We use the following theorem to characterize the similar low-rank off-diagonal structures of Q and R.

Theorem 3.4. *Suppose that a nonsingular upper Hessenberg matrix H can be expressed as $H = Z + xy^T$, with Z being orthogonal and $x, y \in \mathbb{R}^n$, and suppose that it has QR factorization: $H = QR$. Then*

1. *Q has the form: $Q = Q_1 Q_2 \cdots Q_{n-1}$, where each Q_i is a Givens rotation;*
2. *R can be written as: $R = \widetilde{Z} + \widetilde{x}y^H$, with $\widetilde{Z}$ being orthogonal. Furthermore, if we partition R as*

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

where R_{11} and R_{22} are square, then

$$\operatorname{rank}(R_{12}) \leq 2.$$

Proof. For any Hessenberg matrix H, its QR decomposition can be obtained by applying a sequence of Givens rotations $\{Q_i\}_{i=1}^{n-1}$ to zero out its subdiagonal entries from the top to bottom. Specifically, we will have $Q = Q_1 Q_2 \cdots Q_{n-1}$ and

$$R = Q_{n-1}^T \cdots Q_2^T Q_1^T \cdot H = Q^T(Z + xy^T) =: \widetilde{Z} + \widetilde{x}y^T$$

where $\widetilde{Z} := Q^T Z$ and $\widetilde{x} := Q^T x$. We can then finish the proof by using inequalities similar to those in the proof of Theorem 3.1. □

4. Compact Representations of Q and R

Theorem 3.4 implies that it is possible to represent Q and R in compact forms. We dedicate this section to the detailed description of such compact representations.

4.1. Compact representations of Q

Consider an orthogonal matrix Q which can be expressed in the form

$$Q = Q_1 Q_2 \cdots Q_{n-1} \tag{5}$$

where Q_k is a Givens rotation matrix

$$Q_k = \operatorname{diag}\left(I_{k-1}, \begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix}, I_{n-k-1}\right), \ c_k, s_k \in \mathbb{R}, \ c_k^2 + s_k^2 = 1. \tag{6}$$

For convenience we call Q_k the k-th Givens (rotation) matrix. Multiplying out the product (5), it is straightforward to verify that Q takes the following form (assuming $c_0 = c_n = 1$):

$$Q = Q_1 Q_2 \cdots Q_{n-1} = \begin{pmatrix} c_0c_1 & c_0s_1c_2 & c_0s_1s_2c_3 & \cdots & \cdots & c_0s_1\cdots s_{n-1}c_n \\ -s_1 & c_1c_2 & c_1s_2c_3 & \cdots & \cdots & c_1s_2\cdots s_{n-1}c_n \\ & -s_2 & c_2c_3 & \cdots & \cdots & c_2s_3\cdots s_{n-1}c_n \\ & & \ddots & \ddots & \vdots & \vdots \\ & & & -s_{n-2} & c_{n-2}c_{n-1} & c_{n-2}s_{n-1}c_n \\ & & & & -s_{n-1} & c_{n-1}c_n \end{pmatrix}.$$

It is evident that the maximum off-diagonal rank of Q is at most one. Hence an SSS representation for Q will come in handy when we need to conduct SSS matrix-matrix additions or multiplications. With the partitioning sequence $\{m_i = 1\}_{i=1}^n$, the SSS generators of Q are given by Table 2.

$\mathcal{D}_i(Q)$	$\mathcal{U}_i(Q)$	$\mathcal{V}_i(Q)$	$\mathcal{W}_i(Q)$	$\mathcal{P}_i(Q)$	$\mathcal{Q}_i(Q)$	$\mathcal{R}_i(Q)$
$c_{i-1}c_i$	$c_{i-1}s_i$	c_i	s_i	1	$-s_i$	0

TABLE 2. *SSS generators of Q.*

4.2. Compact representations of R

The off-diagonal low-rank structure of R in Theorem 3.4 admits a compact SSS representation. Using the partitioning sequence $\{m_i = 1\}_{i=1}^n$ and taking into account that R is upper triangular, we have

$$R = (R_{ij})_{N\times N}, \quad \text{where } R_{ij} = \begin{cases} d_i, & \text{if } i = j, \\ u_i w_{i+1} \cdots w_{j-1} v_j^T, & \text{if } i < j, \\ 0, & \text{if } i > j. \end{cases} \tag{7}$$

Again, the empty products above are treated as identity matrices. The dimensions of the (nonzero) SSS generators of R are specified in Table 3.

Generator matrix	$\mathcal{D}_i(R)$	$\mathcal{U}_i(R)$	$\mathcal{V}_i(R)$	$\mathcal{W}_i(R)$
	d_i	u_i	v_i	w_i
Size	1×1	$1 \times p$	$1 \times p$	$p \times p$

TABLE 3. *Dimensions of the SSS generators of* R.

According to Theorem 3.4, a compact SSS representation of R will have p not exceeding 2. During our new QR algorithm, however, we will allow not-so-compact (redundant) intermediate SSS generators of R but will compress them back to compact representations at the end of each QR iteration step.

Remark 4.1. As the SSS generators can be simply represented by a small number of vectors or parameters, later in most places of this paper for convenience we directly provide those vectors or parameters instead of writing the SSS forms.

5. A new QR algorithm for C

Consider the $n \times n$ companion matrix (1). Let

$$Z = \begin{pmatrix} 0 & \cdots & 0 & \pm 1 \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix}, \quad e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \text{and} \quad y = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \mp 1 \end{pmatrix},$$

and choose the sign of the $(1, n)$-entry of Z so that $\det(Z) = 1$. Clearly,

$$C = Z + e_1 y^T.$$

Instead of updating the Hessenberg iterates H in the standard QR algorithm, our new algorithm will carry out the implicit shift QR iterations based on the compact representations of Q and R mentioned in the previous section. The structured representations of Q and R will lead to a more delicate deflation scheme and a more convenient bulge chasing procedure, which are to be discussed in detail in the following subsections.

5.1. Swapping real Givens matrices

Before presenting the detailed QR iterations we first consider an important technique which swaps two or three Givens matrices and will be used in the structured bulge chasing. The notion of "swap" will become evident in a moment. Similar techniques can also be found in other places (e.g., [28]).

First consider the product $Q_i \cdot Q_j$, $1 \le i, j < n$, where Q_i and Q_j are two real Givens matrices as specified in (6).

- If $i = j$, then multiplying the product out we get $\widehat{Q}_i \equiv Q_i \cdot Q_j$, which is another Givens matrix, and

$$\begin{pmatrix} \widehat{c}_i & \widehat{s}_i \\ -\widehat{s}_i & \widehat{c}_i \end{pmatrix}, \ \widehat{c}_i = c_i c_j - s_i s_j, \ \widehat{s}_i = c_i s_j + s_i c_j. \tag{8}$$

- If $|i - j| \geq 2$, then

$$Q_i \cdot Q_j = Q_j \cdot Q_i, \tag{9}$$

which is literally swapping the two Givens matrices.

Next consider the product of the form: $Q_i Q_{i+1} G_i$, where Q_i and G_i are two i-th Givens matrices and Q_{i+1} is the $(i+1)$-st Givens matrix, with $1 \leq i \leq n-2$. Without loss of generality, we use $Q_1 Q_2 G_1$ as an example. Given the three Givens matrices in $\mathbb{R}^{3\times 3}$

$$Q_1 = \begin{pmatrix} c_1 & s_1 & \\ -s_1 & c_1 & \\ & & 1 \end{pmatrix}, \quad Q_2 = \begin{pmatrix} 1 & & \\ & c_2 & s_2 \\ & -s_2 & c_2 \end{pmatrix}, \quad G_1 = \begin{pmatrix} \alpha_1 & \beta_1 & \\ -\beta_1 & \alpha_1 & \\ & & 1 \end{pmatrix}, \tag{10}$$

we want to find another three Givens matrices in $\mathbb{R}^{3\times 3}$

$$\widehat{G}_2 = \begin{pmatrix} 1 & & \\ & \widehat{\alpha}_2 & \widehat{\beta}_2 \\ & -\widehat{\beta}_2 & \widehat{\alpha}_2 \end{pmatrix}, \quad \widehat{Q}_1 = \begin{pmatrix} \widehat{c}_1 & \widehat{s}_1 & \\ -\widehat{s}_1 & \widehat{c}_1 & \\ & & 1 \end{pmatrix}, \quad \widehat{Q}_2 = \begin{pmatrix} 1 & & \\ & \widehat{c}_2 & \widehat{s}_2 \\ & -\widehat{s}_2 & \widehat{c}_2 \end{pmatrix}, \tag{11}$$

so that

$$Q_1 Q_2 G_1 = \widehat{G}_2 \widehat{Q}_1 \widehat{Q}_2. \tag{12}$$

We present Algorithm 1 (next page) for the computation of $\widehat{G}_2$, $\widehat{Q}_1$ and $\widehat{Q}_2$.

Note that both approaches in the third step of Algorithm 1 for computing $\widehat{Q}_1$ and $\widehat{Q}_2$ (in exact arithmetic) yield $Q_1 Q_2 G_1 = \widehat{G}_2 \widehat{Q}_1 \widehat{Q}_2$. In a similar fashion, given three Givens matrices G_2, Q_1 and $Q_2 \in \mathbb{R}^{3\times 3}$, we can compute another three Givens matrices $\widehat{Q}_1$, $\widehat{Q}_2$ and $\widehat{G}_1 \in \mathbb{R}^{3\times 3}$ so that

$$G_2 Q_1 Q_2 = \widehat{Q}_1 \widehat{Q}_2 \widehat{G}_1, \tag{13}$$

where G_2 has a similar form as $\widehat{G}_2$ in (11) but without the hats in the notations, and the same situation holds for $\widehat{G}_1$ and G_1.

For the convenience of future reference, we call (12) a *backward Givens swap*, and (13) a *forward Givens swap*, according to the direction of G_1 (or G_2) being pushed. It is not hard to prove the backward stability of such swapping formulas.

Lastly, consider a special case of a backward Givens swap: $Q_{n-1} Q_n G_{n-1}$ with $Q_n = \mathrm{diag}\,[I_{n-1}, -1]$. We want to find another Givens matrix $\widehat{Q}_{n-1}$ so that

$$Q_{n-1} Q_n G_{n-1} = \widehat{Q}_{n-1} Q_n. \tag{14}$$

This boils down to inspect the products of their trailing 2×2 blocks:

$$\begin{pmatrix} c_{n-1} & s_{n-1} \\ -s_{n-1} & c_{n-1} \end{pmatrix} \begin{pmatrix} 1 & \\ & -1 \end{pmatrix} \begin{pmatrix} \alpha_{n-1} & \beta_{n-1} \\ -\beta_{n-1} & \alpha_{n-1} \end{pmatrix} = \begin{pmatrix} \widehat{c}_{n-1} & \widehat{s}_{n-1} \\ -\widehat{s}_{n-1} & \widehat{c}_{n-1} \end{pmatrix} \begin{pmatrix} 1 & \\ & -1 \end{pmatrix}$$

Algorithm 1 Givens swap of type I

(1) Compute

$$A := Q_1 Q_2 G_1 = \begin{pmatrix} c_1\alpha_1 - s_1c_2\beta_1 & c_1\beta_1 + s_1c_2\alpha_1 & s_1s_2 \\ -s_1\alpha_1 - c_1c_2\beta_1 & -s_1\beta_1 + c_1c_2\alpha_1 & c_1s_2 \\ -s_2\beta_1 & -s_2\alpha_1 & c_2 \end{pmatrix} = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix}.$$

(2) Compute a Givens matrix $\widehat{G}_2$ so that

$$A_1 := \widehat{G}_2^T A = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ & \times & \times \end{pmatrix}.$$

(3) *We have two different approaches to get* $\widehat{Q}_1$ *and* $\widehat{Q}_2$.

- *Either*, let

$$\begin{cases} \widehat{c}_1 = & A_1(1,1), \\ \widehat{s}_1 = & -A_1(2,1), \end{cases} \quad \text{and} \quad \begin{cases} \widehat{c}_2 = & A_1(3,3), \\ \widehat{s}_2 = & -A_1(3,2), \end{cases}$$

since if there holds $A_1 = \widehat{Q}_1\widehat{Q}_2$, A_1 must also have the form

$$A_1 = \begin{pmatrix} \widehat{c}_1 & \widehat{s}_1\widehat{c}_2 & \widehat{s}_1\widehat{s}_2 \\ -\widehat{s}_1 & \widehat{c}_1\widehat{c}_2 & \widehat{c}_1\widehat{s}_2 \\ & -\widehat{s}_2 & \widehat{c}_2 \end{pmatrix}.$$

- *Or*, continue to find $\widehat{Q}_1$ so that

$$A_2 := \widehat{Q}_1^T A_1 = \begin{pmatrix} \times & \times & \times \\ & \times & \times \\ & \times & \times \end{pmatrix};$$

and then find $\widehat{Q}_2$ so that

$$A_3 := \widehat{Q}_2^T A_2 = \begin{pmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \end{pmatrix}.$$

Since A_3 is triangular and orthogonal, it must be an identity matrix.

which leads to

$$\begin{cases} \widehat{c}_{n-1} &= & c_{n-1}\alpha_{n-1} + s_{n-1}\beta_{n-1}, \\ \widehat{s}_{n-1} &= & -c_{n-1}\beta_{n-1} + s_{n-1}\alpha_{n-1}. \end{cases} \tag{15}$$

5.2. Initial QR factorization of C

We start the new QR algorithm by first finding the initial QR factorization of $C \equiv C^{(0)}$. This can be easily done by applying a sequence of (transposes of) Givens rotations $\{Q_i^T\}_{i=1}^{n-1}$ to C from the left side to zero out its subdiagonal entries (which are 1's) from top to bottom. The process can be expressed as

$$Q_{n-1}^T(Q_{n-2}^T(\cdots(Q_2^T(Q_1^T C))\cdots)) \Longrightarrow R^{(0)}, \tag{16}$$

where R is an upper triangular matrix and Q_k is the k-th Givens rotation matrix of the form (6).

Thus from equation (16), we can write

$$C = Q_1 Q_2 \cdots Q_{n-1} \cdot R^{(0)}.$$

Let $Q^{(0)} \equiv Q_1 Q_2 \ldots Q_{n-1}$. Then $Q^{(0)}$ is completely represented in terms of its cosine and sine parameters: $\{c_i,\ s_i\}_{i=1}^n$ (with the assumptions $c_n = 1$ and $s_n = 0$). As for $R^{(0)}$, it is straightforward to check that in terms of $\{c_i,\ s_i\}_{i=1}^n$ and $\{a_i\}_{i=1}^n$ we have:

$$R^{(0)} = (R_{ij}^{(0)}), \quad \text{where } R_{ij}^{(0)} = \begin{cases} c_i s_{i-1} \cdots s_1 a_i - s_i & \text{if } i = j, \\ c_i s_{i-1} \cdots s_1 a_j & \text{if } i < j, \\ 0 & \text{if } i > j. \end{cases}$$

Or equivalently, we can use the following SSS generators to completely describe $R^{(0)}$:

$$\begin{cases} \mathcal{D}(R^{(0)}) & \equiv d_i = c_i s_{i-1} \cdots s_1 a_i - s_i, \quad \text{if } 1 \le i \le n, \\ \mathcal{U}(R^{(0)}) & \equiv u_i = c_i s_{i-1} \cdots s_1, \quad \text{if } 1 \le i \le n-1, \\ \mathcal{V}(R^{(0)}) & \equiv v_i = a_i, \quad \text{if } 2 \le i \le n, \\ \mathcal{W}(R^{(0)}) & \equiv w_i = 1, \quad \text{if } 2 \le i \le n-1. \end{cases}$$

Note that for now, p, the common column dimension of SSS generators, is 1.

5.3. Structured QR iteration: single shift case

In this section, by using a concrete 5×5 example, we describe in detail how to implement the following implicit single shift QR iteration on an H as in Theorem 3.4, where $\sigma \in \mathbb{R}$ is a shift.

$$\begin{aligned} H - \sigma I &= QR, \\ \widehat{H} &= RQ + \sigma I = Q^T H Q. \end{aligned}$$

Contrasting with the standard QR algorithm, where we chase a bulge along the second subdiagonal of the Hessenberg iterate H, in our new QR algorithm, we create and chase a bulge along the subdiagonal of R.

Before we start, we make two notations clear:

$\widehat{G}_k$	:	the Givens rotation used to generate a bulge at $R(k+1,k)$,
$\widetilde{G}_k$	:	the Givens rotation used to eliminate the bulge at $R(k+1,k)$,

where $R(i,j)$ denotes the (i,j) entry of R.

Suppose that at the beginning of the QR iteration, we have

$$H = Q_1 Q_2 Q_3 Q_4 \cdot R = Z + xy^T,$$

where Z is orthogonal but not explicitly stored.

(1) **Initiate bulge chasing.** Let $H_0 = H$. Choose a Givens rotation $\bar{G}_1$ of the form

$$\bar{G}_1 = \operatorname{diag}\left(\begin{pmatrix} \bar{c}_1 & \bar{s}_1 \\ -\bar{s}_1 & \bar{c}_1 \end{pmatrix}, I_3\right), \quad \text{where } \bar{c}_1^2 + \bar{s}_1^2 = 1,$$

so that the first column of $\bar{G}_1$, that is, the vector $\begin{pmatrix} \bar{c}_1 & -\bar{s}_1 & 0 & 0 & 0 \end{pmatrix}^T$, is proportional to, $\begin{pmatrix} h_{11} - \sigma & h_{21} & 0 & 0 & 0 \end{pmatrix}^T$, the first column of $H_0 - \sigma I$. Let

$$H_1 \equiv \bar{G}_1^T H_0 \bar{G}_1 = (\bar{G}_1^T Q_1) Q_2 Q_3 Q_4 \cdot R\bar{G}_1.$$

Then a bulge is created at the $(2,1)$ entry of $R\bar{G}_1$. In fact, if we formed $R\bar{G}_1$ explicitly, we should expect

$$R\bar{G}_1 = \begin{pmatrix} \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{pmatrix},$$

where the bulge is indicted by a plus sign. Next choose

$$\widetilde{G}_1 = \operatorname{diag}\left(\begin{pmatrix} \widetilde{c}_1 & \widetilde{s}_1 \\ -\widetilde{s}_1 & \widetilde{c}_1 \end{pmatrix}, I_3\right), \quad \text{where } \widetilde{c}_1^2 + \widetilde{s}_1^2 = 1$$

so that $R_1 \equiv \widetilde{G}_1^T (R\bar{G}_1)$ is upper triangular again. Let $\bar{Q}_1 = \bar{G}_1^T Q_1$, then

$$\begin{aligned} H_1 &= (\bar{G}_1^T Q_1) Q_2 Q_3 Q_4 \widetilde{G}_1 \cdot \widetilde{G}_1^T R_0 \bar{G}_1 \\ &= \bar{Q}_1 Q_2 Q_3 Q_4 \widetilde{G}_1 \cdot R_1 \\ &= (\bar{Q}_1 Q_2 \widetilde{G}_1) Q_3 Q_4 \cdot R_1 \qquad (\widetilde{G}_1 \text{ pushed forward}) \\ &= (\bar{G}_2 \widehat{Q}_1 \bar{Q}_2) Q_3 Q_4 \cdot R_1 \qquad (\text{backward Givens swap}) \\ &= \bar{G}_2 \cdot \widehat{Q}_1 \bar{Q}_2 Q_3 Q_4 \cdot R_1. \end{aligned}$$

(2) **Second chasing.** Let

$$H_2 \equiv \bar{G}_2^T H_1 \bar{G}_2 = \widehat{Q}_1 Q_2 Q_3 Q_4 \cdot R_1 \bar{G}_2,$$

where if explicitly formed,

$$R_1 \bar{G}_2 = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & + & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

Thus the bulge has been "chased" from the $(2,1)$ position to the $(3,2)$ position. To eliminate this bulge, we choose a Givens rotation $\widetilde{G}_2$ so that

$R_2 \equiv \widetilde{G}_2^T(R_1\bar{G}_2)$ becomes upper triangular again. Thus

$$\begin{aligned} H_2 &= \widehat{Q}_1\bar{Q}_2Q_3Q_4\widetilde{G}_2 \cdot \widetilde{G}_2^T R_1\bar{G}_2 \\ &= \widehat{Q}_1(\bar{Q}_2Q_3\widetilde{G}_2)Q_4 \cdot R_2 \qquad (\widetilde{G}_2 \text{ pushed forward}) \\ &= \widehat{Q}_1(\bar{G}_3\widehat{Q}_2\bar{Q}_3)Q_4 \cdot R_2 \qquad (\text{backward Givens swap}) \\ &= \bar{G}_3 \cdot \widehat{Q}_1\widehat{Q}_2\bar{Q}_3Q_4 \cdot R_2. \qquad (\bar{G}_3 \text{ pushed forward}). \end{aligned}$$

(3) **Third chasing.** Similarly, let

$$H_3 \equiv \bar{G}_3^T H_2\bar{G}_3 = \widehat{Q}_1\widehat{Q}_2\bar{Q}_3Q_4 \cdot R_2\bar{G}_3,$$

where if explicitly formed,

$$R_2\bar{G}_3 = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & + & \times & \times \\ & & & & \times \end{pmatrix}.$$

Thus the bulge has been chased from the $(3,2)$ position to the $(4,3)$ position. To eliminate this bulge, we choose a Givens rotation $\widetilde{G}_3$ so that $R_3 \equiv \widetilde{G}_3^T(R_2\bar{G}_2)$ becomes upper triangular again. Thus

$$\begin{aligned} H_3 &= \widehat{Q}_1\widehat{Q}_2(\bar{Q}_3Q_4\widetilde{G}_3) \cdot \widetilde{G}_3^T R_2\bar{G}_3 \\ &= \widehat{Q}_1\widehat{Q}_2(\bar{G}_4\widehat{Q}_3\bar{Q}_4) \cdot R_3 \qquad (\text{backward Givens swap}) \\ &= \bar{G}_4 \cdot \widehat{Q}_1\widehat{Q}_2\widehat{Q}_3\bar{Q}_4 \cdot R_3. \qquad (\widetilde{G}_4 \text{ pushed forward}). \end{aligned}$$

(4) **Final chasing.** Let

$$H_4 \equiv \bar{G}_4^T H_3\bar{G}_4 = \widehat{Q}_1\widehat{Q}_2\widehat{Q}_3\bar{Q}_4 \cdot R_3\bar{G}_4,$$

where if explicitly formed,

$$R_3\bar{G}_4 = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & + & \times \end{pmatrix}.$$

Thus the bulge has been chased from $(4,3)$ to $(5,4)$. This leads us to choose a Givens rotation $\widetilde{G}_4$ such that $R_4 \equiv \widetilde{G}_4^T(R_3\bar{G}_4)$ becomes upper triangular again. Let $\widehat{Q}_4 \equiv \bar{Q}_4\widetilde{G}_4$, then

$$\begin{aligned} H_4 &= \widehat{Q}_1\widehat{Q}_2\widehat{Q}_3(\bar{Q}_4\widetilde{G}_4) \cdot (\widetilde{G}_4^T R_3\bar{G}_4). \\ &= \widehat{Q}_1\widehat{Q}_2\widehat{Q}_3\widehat{Q}_4 \cdot R_4. \end{aligned}$$

Let $\widehat{H} = H_4$. A cycle of QR iteration with single shift is then completed.

Write $\bar{G} \equiv \bar{G}_1\bar{G}_2\bar{G}_3\bar{G}_4$, $\widetilde{G} \equiv \widetilde{G}_1\widetilde{G}_2\widetilde{G}_3\widetilde{G}_4$ and $\widehat{Q} = \widehat{Q}_1\widehat{Q}_2\widehat{Q}_3\widehat{Q}_4$. Then the structured single shift bulge chasing procedure presented above tells us

$$\begin{aligned} H_4 &= \bar{G}_4^T\bar{G}_3^T\bar{G}_2^T\bar{G}_1^T \cdot H_0 \cdot \bar{G}_1\bar{G}_2\bar{G}_3\bar{G}_4 = \bar{G}^T \cdot H_0 \cdot \bar{G}, \\ R_4 &= \widetilde{G}_4^T\widetilde{G}_3^T\widetilde{G}_2^T\widetilde{G}_1^T \cdot R_0 \cdot \bar{G}_1\bar{G}_2\bar{G}_3\bar{G}_4 = \widetilde{G}^T \cdot R_0 \cdot \bar{G}, \\ H_4 &= \widehat{Q} \cdot R_4. \end{aligned} \tag{17}$$

Remark 5.1. Since the first column of $\bar{G}$ is proportional to that of $H_0 - \sigma I$, according to the well known implicit Q theorem, $\bar{G}$ will be the same (up to sign differences in each column) as the Q-factor of the QR decomposition of $H_0 - \sigma I$.

Next we discuss the computation and elimination of the bulges in terms of the structured representations. Note that none of the R_k's are formed explicitly except certain entries. The explanation is as follows. R_k is represented via its SSS generators, $\{d_i, u_i, v_i, w_i\}$. Not all these generators are updated during the intermediate steps of a bulge chasing cycle. We need to form explicitly the main diagonal vector (d_i generators) and the first superdiagonal vector of R_k in order to compute the bulges. To simplify the notations we temporarily write R_k as R, in a general SSS form:

$$R = \begin{pmatrix} \ddots & & \cdots & & \vdots \\ & d_i & u_iv_{i+1}^T & u_iw_{i+1}v_{i+2}^T & \cdots & u_iw_{i+1}\cdots w_{n-1}v_n^T \\ & & d_{i+1} & u_{i+1}v_{i+2}^T & \cdots & u_{i+1}w_{i+2}\cdots w_{n-1}v_n^T \\ & & & d_{i+2} & \cdots & u_{i+2}w_{i+3}\cdots w_{n-1}v_n^T \\ & & & & \ddots & \vdots \end{pmatrix},$$

where the i-th through $(i+2)$-nd rows are shown. Let h be the first superdiagonal vector. That is, $h_i \equiv R_{i,i+1} = u_iv_{i+1}^T$. During the bulge chasing, a bulge b_i is created by right multiplying a Givens matrix $\bar{G}_j = \begin{pmatrix} c_i & -s_i \\ s_i & c_i \end{pmatrix}$ to a 2-by-2 upper triangular diagonal block:

$$\begin{pmatrix} \widehat{d}_i & \widehat{h}_i \\ b_i & \widehat{d}_{i+1} \end{pmatrix} = \begin{pmatrix} d_i & h_i \\ 0 & d_{i+1} \end{pmatrix} \begin{pmatrix} c_i & -s_i \\ s_i & c_i \end{pmatrix}. \tag{18}$$

A new Givens matrix $\widetilde{G}_j = \begin{pmatrix} \widetilde{c}_i & -\widetilde{s}_i \\ \widetilde{s}_i & \widetilde{c}_i \end{pmatrix}$ is now computed based on $\begin{pmatrix} \widehat{d}_i \\ b_i \end{pmatrix}$ so as to eliminate the bulge b_i:

$$\begin{pmatrix} \widetilde{d}_i & \widetilde{h}_i \\ 0 & \widetilde{d}_{i+1} \end{pmatrix} = \begin{pmatrix} \widetilde{c}_i & -\widetilde{s}_i \\ \widetilde{s}_i & \widetilde{c}_i \end{pmatrix} \begin{pmatrix} \widehat{d}_i & \widehat{h}_i \\ b_i & \widehat{d}_{i+1} \end{pmatrix}. \tag{19}$$

Then the i-th and $(i+1)$-st rows of R should be updated, which is done as follows:

$$\begin{aligned}
&\begin{pmatrix} \widetilde{c}_i & -\widetilde{s}_i \\ \widetilde{s}_i & \widetilde{c}_i \end{pmatrix} \left(\begin{array}{cc|ccc} \widehat{d}_i & \widehat{h}_i & u_i w_{i+1} v_{i+2}^T & \cdots & u_i w_{i+1} \cdots w_{n-1} v_n^T \\ b_i & \widehat{d}_{i+1} & u_{i+1} v_{i+2}^T & \cdots & u_{i+1} w_{i+2} \cdots w_{n-1} v_n^T \end{array}\right) \\
&= \left(\begin{pmatrix} \widetilde{d}_i & \widetilde{h}_i \\ 0 & \widetilde{d}_{i+1} \end{pmatrix} \begin{pmatrix} \widetilde{c}_i & -\widetilde{s}_i \\ \widetilde{s}_i & \widetilde{c}_i \end{pmatrix} \begin{pmatrix} u_i w_{i+1} \\ u_{i+1} \end{pmatrix} \begin{pmatrix} v_{i+2}^T & w_{i+2} v_{i+2}^T & \cdots & w_{i+2} \cdots w_{n-1} v_n^T \end{pmatrix} \right) \\
&= \left(\begin{pmatrix} \widetilde{d}_i & \widetilde{h}_i \\ 0 & \widetilde{d}_{i+1} \end{pmatrix} \begin{pmatrix} \widehat{u}_i \\ \widehat{u}_{i+1} \end{pmatrix} \begin{pmatrix} v_{i+2}^T & w_{i+2} v_{i+2}^T & \cdots & w_{i+2} \cdots w_{n-1} v_n^T \end{pmatrix} \right) \\
&= \begin{pmatrix} \widetilde{d}_i & \widetilde{h}_i & \widehat{u}_i v_{i+2}^T & \cdots & \widehat{u}_i w_{i+2} \cdots w_{n-1} v_n^T \\ 0 & \widetilde{d}_{i+1} & \widehat{u}_{i+1} v_{i+2}^T & \cdots & \widehat{u}_{i+1} w_{i+2} \cdots w_{n-1} v_n^T \end{pmatrix}. \qquad (20)
\end{aligned}$$

That is, we only need to find the updated $\widetilde{d}_i, \widetilde{d}_{i+1}, \widetilde{h}_i$, $\widehat{u}_i$, and $\widehat{u}_{i+1}$. After this step, the new superdiagonal entry $h_{i+1} = \widehat{u}_{i+1} v_{i+1}^T$ is formed. The next bulge will be generated with another Givens matrix applied on the right to the next 2-by-2 diagonal block

$$\begin{pmatrix} \widetilde{d}_{i+1} & h_{i+1} \\ 0 & d_{i+2} \end{pmatrix},$$

and the above process repeats. Therefore, during the bulge chasing cycle, $\{d_i, u_i\}$ are updated, and $\{h_i\}$ are formed. Clearly, we use each h_i once a time and do not need to store the entire h.

Equation (20) is sufficient for deriving h_{i+1} and thus further computing and eliminating the bulges. However, the $\widehat{u}_i$ it provides may not be an SSS generator of the final R. As an example, the updated value of $R_{i,i+1}$ is $\widetilde{h}_i$, which is generally not $\widehat{u}_i v_{i+1}^T$. Therefore, to get a final updated SSS form for R, we update all $\{u_i, v_i, w_i\}$ at the end of the bulge chasing cycle. For example, in the process (17) above, the SSS generators of R_4 are obtained by multiplying three SSS matrices $\widetilde{G}^T$, R_0, and $\bar{G}$ using the fast SSS matrix-matrix multiplication formulas in Subsection 2.3.

Remark 5.2. An outcome of using those multiplication formulas is that the column dimensions of R_4's SSS generators will grow additively by 2 (in case of single shift bulge chasing), since both $\bar{G}$ and $\widetilde{G}$ have the maximum off-diagonal rank 1. In Subsection 5.5 we will show how to recover a compact representation for R_4.

5.4. Structured QR iteration: double shift case

This section describes how to maintain real arithmetic by employing two shifts σ and $\bar{\sigma}$ at the same time, where $\bar{\sigma}$ is the complex conjugate of σ (although in this paper notations with bars do not necessarily mean complex conjugates). The

process of shifting σ and $\bar{\sigma}$ successively is like

$$\begin{aligned} H - \sigma I &= Q^{(1)} R^{(1)}, \\ H^{(1)} &= R^{(1)} Q^{(1)} + \sigma I = \left(Q^{(1)}\right)^T H \left(Q^{(1)}\right), \\ H^{(1)} - \bar{\sigma} I &= Q^{(2)} R^{(2)}, \\ \widehat{H} = H^{(2)} &= R^{(2)} Q^{(2)} + \bar{\sigma} I = \left(Q^{(1)} Q^{(2)}\right)^T H \left(Q^{(1)} Q^{(2)}\right), \end{aligned}$$

which leads to

$$M \equiv \left(Q^{(1)} Q^{(2)}\right) \left(R^{(2)} R^{(1)}\right) = (H - \sigma I)(H - \bar{\sigma} I) = H^2 - sH + tI, \tag{21}$$

with $s = 2\,\mathrm{Re}(\sigma)$, $t = |\sigma|^2$. Thus $\left(Q^{(1)} Q^{(2)}\right) \left(R^{(2)} R^{(1)}\right)$ is the QR decomposition of the real matrix M, and therefore $Q^{(1)} Q^{(2)}$, as well as $R^{(2)} R^{(1)}$, can be chosen real, which means that $\widehat{H} = \left(Q^{(1)} Q^{(2)}\right)^T H \left(Q^{(1)} Q^{(2)}\right)$ is also real.

While the rationale for maintaining real arithmetic is exactly the same, the difference of our new algorithm from the standard one lies in the use of the compact representations for Q and R. Contrasting with the standard implicit double shift QR algorithm where a 2-by-2 bulge is chased along the subdiagonal of the Hessenberg iterate H, in our new algorithm the 2-by-2 bulge is chased along the subdiagonal of R. Before we start, we make the following notations clear

F_{k+1}	:	the 1st Givens used to generate a nonzero at $R(k+2,k)$,
$\bar{G}_k$	:	the 2nd Givens used to generate nonzeros at $R(k+1:k+2,k)$,
$\widetilde{F}_{k+1}$	:	the 1st Givens used to eliminate the nonzero at $R(k+2,k)$,
$\widetilde{G}_k$	:	the 2nd Givens used to eliminate the nonzero at $R(k+1,k)$.

Let us use the same 5-by-5 example from the last subsection. Suppose that at the beginning of the QR iteration, we have

$$H_0 \equiv H = Q_1 Q_2 Q_3 Q_4 \cdot R = Z + xy^T,$$

where Z is orthogonal but not explicitly stored.

(1) **Initiate bulge chasing.** Given a pair of complex conjugate shifts σ and $\bar{\sigma}$, we compute the first column of M in (21):

$$Me_1 = (H^2 - sH + tI)e_1 = \begin{pmatrix} x_1 & x_2 & x_3 & 0 & \cdots & 0 \end{pmatrix}^T,$$

where

$$\begin{cases} x_1 &= h_{11}^2 + h_{12}h_{21} - sh_{11} + t, \\ x_2 &= h_{21}(h_{11} + h_{22} - s), \\ x_3 &= h_{21}h_{32}. \end{cases} \tag{22}$$

Then find two Givens rotations $\bar{G}_1$ and $\bar{F}_2$ such that

$$\left(\bar{G}_1\right)^T \left(\bar{F}_2\right)^T \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \times \\ 0 \\ 0 \end{pmatrix}.$$

In other words, the first column of $\left(\bar{F}_2\bar{G}_1\right)$ should be made proportional to Me_1. Let

$$\begin{aligned}
H_1 &\equiv \left(\bar{F}_2\bar{G}_1\right)^T \cdot H_0 \cdot \left(\bar{F}_2\bar{G}_1\right) \\
&= \left(\bar{G}_1\right)^T \cdot \left((\bar{F}_2)^T Q_1 Q_2\right) Q_3 Q_4 \cdot R_0 \bar{F}_2 \bar{G}_1 \\
&= \left(\bar{G}_1\right)^T \cdot \left(\bar{Q}_1 \bar{Q}_2 \widetilde{F}_1\right) Q_3 Q_4 \cdot R_0 \bar{F}_2 \bar{G}_1 \qquad \text{(forward Givens swap)} \\
&= \left(\left(\bar{G}_1\right)^T \bar{Q}_1\right) \bar{Q}_2 Q_3 Q_4 \cdot \left(\widetilde{F}_1 R_0\right) \bar{F}_2 \bar{G}_1 \\
&= \widehat{Q}_1 \bar{Q}_2 Q_3 Q_4 \cdot \widetilde{R}_0 \bar{F}_2 \bar{G}_1,
\end{aligned}$$

where $\widehat{Q}_1 \equiv (\bar{G}_1)^T \bar{Q}_1$, $\widetilde{R}_0 \equiv \widetilde{F}_1 R_0$, and if formed explicitly,

$$\widetilde{R}_0 \bar{F}_2 \bar{G}_1 = \begin{pmatrix} \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times \\ + & + & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

We see that there is a 2-by-2 bulge, indicted by plus signs. Next choose two Givens rotations $\widetilde{F}_2$ and $\widetilde{G}_1$ to zero out entries $(3,1)$ and $(2,1)$ of $\widetilde{R}_0 \bar{F}_2 \bar{G}_1$ in order. Let $\widetilde{R}_1 \equiv \left(\widetilde{G}_1\right)^T \left(\widetilde{F}_2\right)^T \cdot \left(\widetilde{R}_0 \bar{F}_2 \bar{G}_1\right)$, then we may write

$$\begin{aligned}
H_1 &= \widehat{Q}_1 \bar{Q}_2 Q_3 Q_4 \left(\widetilde{F}_2 \widetilde{G}_1\right) \cdot \widetilde{R}_1 \\
&= \widehat{Q}_1 \left(\bar{Q}_2 Q_3 \widetilde{F}_2\right) Q_4 \widetilde{G}_1 \cdot \widetilde{R}_1 \qquad (\widetilde{F}_2 \text{ pushed forward}) \\
&= \widehat{Q}_1 \left(\bar{F}_3 \widetilde{Q}_2 \bar{Q}_3\right) Q_4 \widetilde{G}_1 \cdot \widetilde{R}_1 \qquad \text{(backward Givens swap)} \\
&= \bar{F}_3 \left(\widehat{Q}_1 \widetilde{Q}_2 \widetilde{G}_1\right) \bar{Q}_3 Q_4 \cdot \widetilde{R}_1 \qquad (\bar{F}_3 \text{ and } \widetilde{G}_1 \text{ pushed forward.}) \\
&= \bar{F}_3 \left(\bar{G}_2 \widehat{\widetilde{Q}}_1 \widehat{Q}_2\right) \bar{Q}_3 Q_4 \cdot \widetilde{R}_1 \qquad \text{(backward Givens swap)} \\
&= \bar{F}_3 \bar{G}_2 \cdot \widehat{\widetilde{Q}}_1 \widehat{Q}_2 \bar{Q}_3 Q_4 \cdot \widetilde{R}_1.
\end{aligned}$$

(2) **Second chasing.** Let

$$H_2 \equiv \left(\bar{F}_3\bar{G}_2\right)^T \cdot H_1 \cdot \left(\bar{F}_3\bar{G}_2\right) = \widehat{\widetilde{Q}}_1 \widehat{Q}_2 \bar{Q}_3 Q_4 \cdot \left(\widetilde{R}_1 \bar{F}_3 \bar{G}_2\right),$$

where if explicitly formed,

$$\widetilde{R}_1 \bar{F}_3 \bar{G}_2 = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & + & \times & \times & \times \\ & + & + & \times & \times \\ & & & & \times \end{pmatrix}.$$

Thus compared with $\widetilde{R}_0\bar{F}_2\bar{G}_1$, the 2-by-2 bulge has been chased to the right for one column. Next choose two Givens rotations $\widetilde{F}_3$ and $\widetilde{G}_2$ to zero out $(4,2)$ and $(3,2)$ entries in order. Let $\widetilde{R}_2 \equiv \left(\widetilde{G}_2\right)^T \left(\widetilde{F}_3\right)^T \cdot \left(\widetilde{R}_1\bar{F}_3\bar{G}_2\right)$, then we may write

$$\begin{aligned}
H_2 &= \widehat{\widehat{Q}}_1\widehat{Q}_2\bar{Q}_3Q_4\left(\widetilde{F}_3\widetilde{G}_2\right)\cdot\widetilde{R}_2 \\
&= \widehat{\widehat{Q}}_1\widehat{Q}_2\left(\bar{Q}_3Q_4\widetilde{F}_3\right)\widetilde{G}_2\cdot\widetilde{R}_2 \\
&= \widehat{\widehat{Q}}_1\widehat{Q}_2\left(\bar{F}_4\widetilde{Q}_3\bar{Q}_4\right)\widetilde{G}_2\cdot\widetilde{R}_2 \qquad \text{(backward Givens swap)} \\
&= \bar{F}_4\widehat{\widehat{Q}}_1\left(\widehat{Q}_2\widetilde{Q}_3\widetilde{G}_2\right)\bar{Q}_4\cdot\widetilde{R}_2 \qquad (\bar{F}_4 \text{ and } \widetilde{G}_2 \text{ pushed forward}) \\
&= \bar{F}_4\widehat{\widehat{Q}}_1\left(\bar{G}_3\widehat{\widehat{Q}}_2\widehat{Q}_3\right)\bar{Q}_4\cdot\widetilde{R}_2 \qquad \text{(backward Givens swap)} \\
&= \bar{F}_4\bar{G}_3\cdot\widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\widehat{Q}_3\bar{Q}_4\cdot\widetilde{R}_2. \qquad (\bar{G}_3 \text{ pushed forward}).
\end{aligned}$$

(3) **Final two steps of bulge chasing.** Let

$$H_3 \equiv \left(\bar{F}_4\bar{G}_3\right)^T\cdot H_2\cdot\left(\bar{F}_4\bar{G}_3\right) = \widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\widehat{Q}_3\bar{Q}_4\cdot\left(\widetilde{R}_2\bar{F}_4\bar{G}_3\right),$$

where if explicitly formed,

$$\widetilde{R}_2\bar{F}_4\bar{G}_3 = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & + & \times & \times \\ & & + & + & \times \end{pmatrix}.$$

Thus compared with $\widetilde{R}_1\bar{F}_3\bar{G}_2$, the 2-by-2 bulge has been chased by one column to the lower right. Next choose two Givens rotations $\widetilde{F}_4$ and $\widetilde{G}_3$ to zero out the $(5,3)$ and $(4,3)$ entries in order. Let $\widetilde{R}_3 \equiv \left(\widetilde{G}_3\right)^T\left(\widetilde{F}_4\right)^T\cdot\left(\widetilde{R}_2\bar{F}_4\bar{G}_3\right)$, then we may write

$$\begin{aligned}
H_3 &= \widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\widehat{Q}_3\bar{Q}_4\left(\widetilde{F}_4\widetilde{G}_3\right)\cdot\widetilde{R}_3 \\
&= \widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\left(\widehat{Q}_3\widetilde{Q}_4\widetilde{G}_3\right)\cdot\widetilde{R}_3 \qquad (\widetilde{Q}_4 \equiv \bar{Q}_4\widetilde{F}_4) \\
&= \widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\left(\bar{G}_4\widehat{\widehat{Q}}_3\widehat{Q}_4\right)\cdot\widetilde{R}_3 \qquad \text{(backward Givens swap)} \\
&= \bar{G}_4\cdot\widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\widehat{\widehat{Q}}_3\widehat{Q}_4\cdot\widetilde{R}_3. \qquad (\bar{G}_4 \text{ pushed forward}).
\end{aligned}$$

Lastly, let

$$H_4 \equiv \left(\bar{G}_4\right)^T\cdot H_3\cdot\left(\bar{G}_4\right) = \widehat{\widehat{Q}}_1\widehat{\widehat{Q}}_2\widehat{\widehat{Q}}_3\widehat{Q}_4\cdot\left(\widetilde{R}_3\bar{G}_4\right),$$

where if explicitly formed,

$$\widetilde{R}_3\bar{G}_4 = \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & + & \times \end{pmatrix}.$$

Next choose a Givens rotation $\widetilde{G}_4$ to zero out the $(5,4)$ entry above to get an upper triangular matrix $R_4 \equiv \left(\widetilde{G}_4\right)^T \cdot \widetilde{R}_3\bar{G}_4$. Now we may write

$$\begin{aligned} H_4 &= \widehat{\widetilde{Q}}_1\widehat{\widetilde{Q}}_2\widehat{\widetilde{Q}}_3\widehat{Q}_4\widetilde{G}_4 \cdot R_4 \\ &= \widehat{\widetilde{Q}}_1\widehat{\widetilde{Q}}_2\widehat{\widetilde{Q}}_3\widehat{\widetilde{Q}}_4 \cdot R_4. \qquad (\widehat{\widetilde{Q}}_4 \equiv \widehat{Q}_4\widetilde{G}_4). \end{aligned}$$

Let $\widehat{H} = H_4$. A cycle of QR iteration with a pair of complex conjugate shifts $\{\sigma, \bar{\sigma}\}$ is then completed. Define $\widehat{Q} \equiv \widehat{\widetilde{Q}}_1\widehat{\widetilde{Q}}_2\widehat{\widetilde{Q}}_3\widehat{\widetilde{Q}}_4$, and

$$\begin{aligned} \bar{W} &\equiv \bar{F}_2\bar{G}_1\bar{F}_3\bar{G}_2\bar{F}_4\bar{G}_3\bar{G}_4 \\ &= \left(\bar{F}_2\bar{F}_3\bar{F}_4\right) \cdot \left(\bar{G}_1\bar{G}_2\bar{G}_3\bar{G}_4\right) \\ &\equiv \bar{F} \cdot \bar{G}, \\ \widetilde{W} &\equiv \widetilde{F}_1\widetilde{F}_2\widetilde{G}_1\widetilde{F}_3\widetilde{G}_2\widetilde{F}_4\widetilde{G}_3\widetilde{G}_4 \\ &= \left(\widetilde{F}_1\widetilde{F}_2\widetilde{F}_3\widetilde{F}_4\right) \cdot \left(\widetilde{G}_1\widetilde{G}_2\widetilde{G}_3\widetilde{G}_4\right) \\ &\equiv \widetilde{F} \cdot \widetilde{G}. \end{aligned}$$

We can then summarize the structured double shift bulge chasing procedure as:

$$\begin{aligned} H_4 &= \bar{W}^T \cdot H_0 \cdot \bar{W} = (\bar{F}\bar{G}) \cdot H_0 \cdot (\bar{F}\bar{G}), \\ R_4 &= \widetilde{W}^T \cdot R_0 \cdot \bar{W} = (\widetilde{F}\widetilde{G}) \cdot H_0 \cdot (\bar{F}\bar{G}), \\ H_4 &= \widehat{Q} \cdot R_4. \end{aligned}$$

Remark 5.3. Since the first column of $\bar{W}$ is proportional to that of $H^2 - sH + tI$ (with $s = 2\,\mathrm{Re}(\sigma)$, $t = |\sigma|^2$), according to the well known implicit Q theorem, $\bar{W}$ will be the same (up to sign differences in each column) as the Q-factor of the QR decomposition of $H^2 - sH + tI$.

Remark 5.4. Similar to the single shift case, none of the R_k's are formed explicitly, except few diagonal vectors which are needed for computing the bulges. The SSS generators $\{d_i, u_i\}$ of R_k are updated during the process. At the end of a bulge chasing cycle, $\{v_i, w_i\}$, are updated (also $\{u_i\}$, in fact), and this can be done efficiently by applying the fast SSS matrix-matrix multiplication formulas. However, an outcome of using those multiplication formulas is that the column dimensions of R's SSS generators will grow by 4 in case of double shift bulge chasing, since

both $\bar{W}$ and $\widetilde{W}$ have the maximum off-diagonal rank to be 2. In the next subsection, we will show how to recover a compact representation for R_4, or in general R_{n-1}.

5.5. Recovery of the compact SSS representation of R

In both single and double shift cases, we computed the SSS representation of R_{n-1} ($n = 5$ for the 5-by-5 example we considered) through the formula

$$R_{n-1} = \widetilde{G}^T \cdot R_0 \cdot \bar{G},$$

where for simplicity of notation, we have written in case of double shift iteration: $\widetilde{W} = \widetilde{F}\widetilde{G}$ as $\widetilde{G}$, $\bar{W} = \bar{F}\bar{G}$ as $\bar{G}$. As pointed out in Remarks 5.2 and 5.4, the column dimensions of the SSS generators of R_{n-1} increase by 2 and 4 in single and double shift cases, respectively. However, the mathematical ranks of the off-diagonal blocks of R_{n-1} do not increase starting from $n = 2$. The reason is that given $H_0 = Z + xy^T$, where Z is orthogonal but never explicitly stored, we can represent R_{n-1} as a rank-one modification to an orthogonal matrix:

$$R_{n-1} = \widehat{Q}^T H_{n-1} = \widehat{Q}^T \bar{G}^T H_0 \bar{G} = \left(\widehat{Q}^T \bar{G}^T Z \bar{G}\right) + \left(\widehat{Q}^T \bar{G}^T x\right) \cdot \left(\bar{G}^T y\right)^T.$$

According to Theorem 3.4, $\operatorname{rank}(R_{12}) \leq 2$ for any 2-by-2 blocking partitioning.

To recover a compact representation of R_{n-1}, we do the following.

(1) Compute $\widehat{x} = \widehat{Q}^T \bar{G}^T x$ and $\widehat{y} = \bar{G}^T y$. As just shown, the computed R_{n-1} in a redundant SSS form can be viewed as a rank-one perturbation to an orthogonal matrix, that is,

$$R_{n-1} - \widehat{x}\widehat{y}^T \text{ is an orthogonal matrix.}$$

(2) Find a sequence of Givens rotations $\{X_1, X_2, \ldots, X_{n-1}\}$, and let

$$X \equiv X_1 X_2 \cdots X_{n-1},$$

so that

$$X\widehat{x} = e_1.$$

Apply X to $R_{n-1} - \widehat{x}\widehat{y}^T$ from the left-hand side. Now $XR_{n-1} - e_1\widehat{y}^T$ remains orthogonal. On the other hand, since R_{n-1} is upper triangular and X is upper Hessenberg, $XR_{n-1} - e_1\widehat{y}^T$ is also upper Hessenberg.

(3) Thus we can find another sequence of Givens rotations $\{Y_{n-1}, Y_{n-2}, \ldots, Y_1\}$, let $Y \equiv Y_1 Y_2 \cdots Y_{n-1}$, so that

$$\left(XR_{n-1} - e_1\widehat{y}^T\right) Y^T = I.$$

(4) The last equation provides an alternative way to express R_{n-1}, that is,

$$R_{n-1} = X^T Y + X^T e_1 \widehat{y}^T = X^T Y + \widehat{x}\widehat{y}^T.$$

Both X are Y have orthogonal upper Hessenberg matrices with similar structure as that of Q, so that they can be written as SSS matrices with the maximum off-diagonal rank to be 1. The rank-one matrix $\widehat{x}\widehat{y}^T$ can also be written in SSS form with off-diagonal rank to be 1. By applying the fast SSS matrix-matrix multiplication in Subsection 2.3 to $X^T Y$ we obtain an SSS form for

X^TY with generator sizes bounded by 2 (the sizes increase additively). Then another fast SSS addition (Subsection 2.2) makes $R_{n-1} = (X^TY) + (\widehat{x}\widehat{y}^T)$ a new SSS matrix with generator sizes bounded by 3. That means, we get a new compact representation for R_{n-1}. Here although theoretically, according to Theorem 3.4 it is possible to further make the generator sizes no larger than 2, it does not make a significant difference in practice. We allow the sizes to be 3 for the sake of convenience in the programming. The above recovery process also applies to all subsequent QR iterations and it guarantees the generators sizes to be bounded by 3. Another implication of the equation above is that in exact arithmetics, $X^TY + \widehat{x}\widehat{y}^T$ is an upper triangular matrix.

5.6. Deflation and Convergence Criterion

After showing the details of the fast structured bulge chasing schemes we provide the deflation technique and the convergence criterion in terms of SSS representations.

Deflation is an important concept in the practical implementation of the QR iteration method. It amounts to setting small subdiagonal elements of the Hessenberg matrix to zero. After deflation, it splits the Hessenberg matrix into two smaller subproblems which may be independently refined further. Theoretically, assume that deflation occurs to an intermediate Hessenberg matrix

$$H = Q_1 \cdots Q_{n-1} \cdot R,$$

and a subdiagonal entry $h_{i,i-1}$ of H becomes 0. This corresponds to the fact that the Givens matrix Q_{i-1} in the Q-factor sequence of H becomes an identity matrix:

$$\begin{aligned} H &= (Q_1 \cdots Q_{i-2}) \cdot Q_{i-1} \cdot (Q_i \cdots Q_{n-1}) \cdot R \\ &= (Q_1 \cdots Q_{i-2}) \cdot \quad I \quad \cdot (Q_i \cdots Q_{n-1}) \cdot R. \end{aligned} \tag{23}$$

In traditional deflation schemes H will be treated as two subproblems individually. That means here we have to look for a new orthogonal-plus-rank-one representation such as (4) for each subproblem. It is not obvious so far how we can quickly get those representations based on the original orthogonal-plus-rank-one representation. However, instead of seeking new representations, we will keep the original orthogonal-plus-rank-one representation, reuse the original Q- and R-factors, and in the meantime, keep track of the identity matrices such as Q_{i-1}. The identity matrix Q_{i-1} in (23) splits the Q_j factors into two subgroups (corresponding to the two subproblems in traditional deflation schemes). In later bulge chasing steps, operations will be done within each subgroup. That is, we maintain global representations for Q- and R-factors, but keep the actual structured operations locally within subgroups.

We also need to take care of deflation criteria based on the low-rank structures. In traditional computations there are various deflation criteria, such as the one proposed by Wilkinson which is used in LAPACK [2] and a new one proposed by Ahues and Tisseur [1]. For our new QR algorithm, we can adopt similar criteria. The difference is that since the Hessenberg iterate H is not explicitly formed, we

need to compute relevant elements of H on the fly through compact representations of Q and R. For example, Wilkinson's deflation criterion will set $h_{i,i-1}$ to zero if

$$|h_{i,i-1}| \leq \tau \cdot (|h_{i-1,i-1}| + |h_{i,i}|), \tag{24}$$

where τ is a given tolerance. In terms of the elements of Q and R we have

$$\begin{pmatrix} h_{i-1,i-1} & \times \\ h_{i,i-1} & h_{i,i} \end{pmatrix} = \begin{pmatrix} -s_{i-2} & c_{i-2}c_{i-1} & \times \\ & -s_{i-1} & c_{i-1}c_i \end{pmatrix} \begin{pmatrix} u_{i-2}v_{i-1}^T & \times \\ d_{i-1} & u_{i-1}v_i^T \\ & d_i \end{pmatrix},$$

where $\times$ denotes certain element in the corresponding matrix. This gives us

$$\begin{cases} h_{i,i-1} &= -s_{i-1}d_{i-1}, \\ h_{i-1,i-1} &= -s_{i-2}(u_{i-2}v_{i-1}^T) + c_{i-2}c_{i-1}d_{i-1}, \\ h_{i,i} &= -s_{i-1}(u_{i-1}v_i^T) + c_{i-1}c_i d_i. \end{cases}$$

When the criterion (24) is satisfied, we want to set $h_{i,i-1}$ to zero. However, since H is not explicitly stored, we choose to do this by making s_{i-1} zero. There are two possible scenarios:

1. If $|s_{i-1}| \leq O(\epsilon)$, with ϵ being the machine precision, it's straightforward: we will just set $s_{i-1} \equiv 0$ and $c_{i-1} \equiv \text{sign}(c_{i-1})$ without changing anything else.
2. If $|s_{i-1}| > O(\epsilon)$, things become tricky. We first multiply $(Q_{i-1}Q_i \cdots Q_{n-1})$ to R to get $H(i-1:n, i-1:n)$ in its SSS form. We then find another sequence of Givens rotation matrices $(\widehat{Q}_{i-1}\widehat{Q}_i \cdots \widehat{Q}_{n-1})$, whose transpose applied to the left side of $H(i-1:n, i-1:n)$ will yield a new upper triangular matrix $\widehat{R}$. Note that:
   (a) $\widehat{Q}_{i-1}$ is automatically an identity matrix, since $h_{i,i-1}$ is small enough to be ignored;
   (b) all matrix-matrix multiplications are done quickly by updating SSS generators.

In the standard QR algorithm, we say that the algorithm converges if the Hessenberg iterate H_k eventually becomes a real quasi triangular matrix (called the Schur form). In our new QR algorithm for real companion matrices, we say that the algorithm converges if the Q-factor in its trigonometric parametrization form $Q = Q_1Q_2 \cdots Q_{n-1}$ satisfies the following *convergence criterion*: for any two consecutive Givens rotations $\{Q_k, Q_{k+1}\}$ $(k = 1, 2, \ldots, n-2)$, one of them must be an identity matrix.

5.7. Summary of the new QR algorithm for C

The gist of our new QR algorithm for companion matrices is the usage of compact representations for Q (as a product of a sequence of Givens rotations) and for R (in terms of its SSS form) during the QR iteration process. The feasibility of such compact representations for Q and R is guaranteed by the fact that the Hessenberg iterates of the companion matrix during QR iteration process have low-rank off-diagonal blocks (the maximum off diagonal rank of H never exceeds 3). Similar low-rank properties extend to the Q- and R-factors of H.

In terms of compact representations of Q and R, rather than explicitly forming and updating structured matrices for the Hessenberg iterates H as done in [4] and [6], we may summarize our new QR iteration method in Algorithm 2.

6. Balancing Strategy

We also briefly mention the balancing strategy. Before QR iterations for the eigenvalues of a matrix A we usually apply a diagonal similarity transformation to A for the purpose of better accuracy and efficiency. That is, we compute the eigenvalues of DAD^{-1} where D is a diagonal matrix. The matrix D is often chosen such that the norms of each row and the corresponding column of DAD^{-1} are close.

A similar balancing strategy as in [4] can be used. In our new fast eigensolver for the companion matrix C, we have exploited the fact that the Hessenberg iterates under the QR iteration have low-rank off-diagonal blocks, so we are able to use compact representations for the Q- and and R-factors. However, after balancing these rank structures for the iterates of DCD^{-1} may be destroyed, where $D = \mathrm{diag}(d_1, \ldots, d_n)$. That is, The Hessenberg iterates for DCD^{-1} may no longer have low-rank off-diagonal blocks. However, notice

$$DCD^{-1} = \begin{pmatrix} a_1 & \frac{d_1}{d_2}a_2 & \cdots & \frac{d_1}{d_{n-1}}a_{n-1} & \frac{d_1}{d_n}a_n \\ \frac{d_2}{d_1} & 0 & \cdots & 0 & 0 \\ 0 & \frac{d_3}{d_2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{d_n}{d_{n-1}} & 0 \end{pmatrix}. \tag{25}$$

If we can select D such that

$$\frac{d_2}{d_1} = \frac{d_3}{d_2} = \cdots = \frac{d_n}{d_{n-1}} \equiv \alpha$$

for certain α, then DCD^{-1} becomes the multiple of a new companion matrix:

$$DCD^{-1} = \alpha \cdot \begin{pmatrix} \frac{a_1}{\alpha} & \frac{a_2}{\alpha^2} & \cdots & \frac{a_{n-1}}{\alpha^{n-1}} & \frac{a_n}{\alpha^n} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \equiv \alpha \cdot \widehat{C},$$

where $\widehat{C}$ is the companion matrix corresponding to the polynomial $p(\alpha x)/\alpha^n$, with $p(x)$ being the polynomial (2) corresponding to the original companion matrix C. This means that we can choose a geometric scaling ($d_i = \alpha^i$), and apply the fast QR iterations to $\widehat{C}$ so as to preserve the low-rank structures. After the eigenvalues of the new companion matrix $\widehat{C}$ are obtained we can multiply them by α to get those of C.

Some efficient balancing algorithms for a given matrix A based on the approximation of Perron vectors of $|A|$ are developed in [12]. It was also shown

Algorithm 2 New structured QR algorithm for a real companion matrix C

Input: the first row of C: $\begin{pmatrix} a_1 & a_2 \dots & a_{n-1} & a_n \end{pmatrix}$
Output: Q: in terms of $\{c(Q), s(Q)\}$;
R: in terms of $\{d(R), u(R), v(R), w(R)\}$.

(1) **Initialization**
  (a) Compute QR factorization of C: $C = Q_1 Q_2 \dots Q_{n-1} \cdot R$.
  (b) Find x and y such that $C = Z + xy^T$. [*Note that only* $\{c_i(Q), s_i(Q)\}$, $\{d_i(R), u_i(R), v_i(R), w_i(R)\}$, x *and* y *are explicitly stored.*]
(2) **Repeat**
  (a) **Modified Bulge Chasing** with shift(s)
    (i) Determine what shift to use (Francis single or double shift or exceptional shift).
    (ii) For $i = 1$, find $\bar{G}_i$ to create a bulge on subdiagonal of R and then find $\widetilde{G}_i$ to eliminate it.
    (iii) For $i = 2, \dots, n-1$:
    (iv) Update Q by Givens swaps: $Q_{i-1} Q_j \widetilde{G}_{i-1} \Rightarrow \bar{G}_i \widehat{Q}_{i-1} \widehat{Q}_i$. Store $\bar{G}_i$.
    (v) Update R by bulge elimination: find $\widetilde{G}_i$ to eliminate the bulge in $R\bar{G}_i$. For example, for single shift:
      Update $d_i(R), d_{i+1}(R)$, form the bulge b_i in $R\bar{G}_i$, and update h_i, as in (18).
      Compute $\widetilde{G}_i$ and update $d_i(R), d_{i+1}(R)$ as in (19).
      Update $u_i(R), u_{i+1}(R)$ as in (20).
    (vi) Endfor
    (vii) Merge $\widetilde{G}_{n-1}$ into Q_{n-1}: $\widehat{Q}_{n-1} := Q_{n-1} \widetilde{G}_{n-1}$. Each $\widehat{Q}_i$ becomes the new Q_i.
    (viii) Get updated SSS representation for R by two SSS matrix multiplications (see, e.g. (17)).
  (b) **Deflation**:
    (i) If $H_{i+1,i}$ is small enough to be thrown away **and** if Q_i is not an identity matrix, update $Q_i, \dots, Q_{n-1}$ and the corresponding parts of SSS generators of $\widehat{R}$.
  (c) **Restore Compact Representation of** R
    (i) $\widehat{Q}$ and $\bar{G}$ are available through the parametric representations of $\widehat{Q}_i$ and $\bar{G}_i$, respectively. Let $\widehat{x} := \widehat{Q}^T \bar{G}^T x$, $\widehat{y} = \bar{G}^T y$, then $\widehat{R}$ satisfies: $\widehat{R} = \widehat{Z} + \widehat{x}\widehat{y}^T$ for some orthogonal $\widehat{Z}$.
    (ii) Find X so $X\widehat{x} = e_1 \implies X\widehat{R} - e_1 \widehat{y}^T$ is orthogonal and upper Hessenberg.
    (iii) Find Y so that $(X\widehat{R} - e_1 \widehat{y}^T) Y^T = I$.
    (iv) Compute SSS generators $\{d_i(R), u_i(R), v_i(R), w_i(R)\}$ of $R := X^T Y + \widehat{x}\widehat{y}^T$: first use SSS multiplications to obtain an SSS form for $X^T Y$ with generator sizes no larger than 2. Then use SSS additions to obtain an SSS form for R with generator sizes no larger than 3.
Until convergent

that if A is irreducible and x and y are the right and left Perron vectors of $|A|$, respectively, then $D = \mathrm{diag}(1/x_1, \ldots, 1/x_n)$ minimizes $\|DAD^{-1}\|_\infty$, and $D = \mathrm{diag}(\sqrt{y_1/x_1}, \ldots, \sqrt{y_n/x_n})$ minimizes $\|DAD^{-1}\|_2$. Here C is a companion matrix, and so is $|C|$. The matrix C has a right Perron vector with entries $x_i = \alpha^{n-i}$, where α is the maximum positive eigenvalue of $|C|$, or equivalently the largest positive root of $x^n - |a_1|x^{n-1} - \cdots - |a_{n-1}|x - |a_n|$. Therefore, a geometric scaling with such an α minimizes the infinity-norm of DCD^{-1}. In our algorithm, however, only orthogonal transformations are applied. Ideally, we should look for a geometric scaling strategy such that $\|DCD^{-1}\|_2$ is minimized. Empirically, we find the following criterion for choosing α to be useful: choosing α to make

$$\mathrm{Range}\{|\widehat{c}_1|, |\widehat{c}_2|, \ldots, |\widehat{c}_n|, 1\} \equiv \frac{\max\{|\widehat{c}_1|, \ldots, |\widehat{c}_n|, 1\}}{\min\{|\widehat{c}_1|, \ldots, |\widehat{c}_n|, 1\}}$$

as small as possible, where $\widehat{c}_i = \frac{a_i}{\alpha^i}$.

In practice, α is often selected to be a power of the machine radix so as to avoid errors in computing DCD^{-1}. In our numerical experiments we have tried different powers of 2 as α (see the next section), although more work needs to be done on a systematic way of choosing α.

7. Numerical Experiments

We have tested our new structured QR algorithm on many different examples and it is stable in practice, although it is still an open problem to show whether the new algorithm is stable or not. We implemented the new QR-iteration method in FORTRAN 90 for computing the eigenvalues of real companion matrices. The codes are available online.[1] Numerical experiments are run on a laptop with an Intel Pentium M 1.7GHz CPU and 512MB RAM. Results are summarized in the following two subsections to illustrate both the performance, i.e., $O(n^2)$ complexity and the stability in practice.

We first point out that among all our numerical tests, the program runs stably and we did not observe any significant failure or corruption of the orthogonal-plus-rank-one structures by using the compact SSS QR factors. The low-rank Hessenberg structures are well preserved in the experiments.

7.1. $O(n^2)$ complexity Tests

We use real polynomials with uniformly random coefficients as test polynomials. The degree of the polynomials doubles from 25 up to 102,400. We also show the relative backward error

$$\frac{\|\bar{G}^T \cdot C_0 \cdot \bar{G} - Q^{(m)}R^{(m)}\|_\infty}{\|C_0\|_\infty},$$

where C_0 denotes the initial companion matrix, m is the number of iterations needed for convergence, $Q^{(m)}$ and $R^{(m)}$ are explicitly formed Q- and R-factors

[1] http://www.math.ucla.edu/~jxia/work/companion/

n (size)	DGEEV(sec)	New SSS(sec)	iter. #	rel. BkErr
25	0.01	0.01	83	1×10^{-15}
50	0.03	0.03	161	2×10^{-15}
100	0.12	0.09	309	3×10^{-15}
200	0.33	0.22	584	7×10^{-15}
400	1.70	0.51	1200	2×10^{-14}
800	12.33	1.98	2165	3×10^{-14}
1,600	95.82	7.43	4170	1×10^{-13}
3,200	865.22	56.11	8125	
6,400	-	296.21	15569	
12,800	-	1,302.22	30551	
25,600	-	5,465.76	62080	
51,200	-	21,080.34	116708	
102,400	-	83,583.64	252822	

TABLE 4. *Numerical results on new $O(n^2)$ companion eigensolver.*

of the final convergent Schur form of C_0, and $\bar{G}$ is the accumulated orthogonal similarity transformation.

Remark 7.1. The break-even size of the current new companion eigensolver implementation versus LAPACK is about $n = 50$. For the test problem of size $102,400$, it took the new companion eigensolver about 23 hours to converge all the roots; on the other hand, the LAPACK routine `DGEEV` can't even run for problems of size about $8,000$ since it uses $O(n^2)$ storage; even if the memory was not an issue, it would take `DGEEV` more than 300 days to converge on the same machine since it's an $O(n^3)$ method.

Remark 7.2. From Table 4 and Figure 1, we clearly see the quadratic (i.e. $O(n^2)$) complexity of the new QR iteration Algorithm 1, see Figure 1 (a). The average iteration number needed per eigenvalue is less than 3, see Figure 1 (b). In the mean time, we observe nearly linear growth in both backward and forward errors.

Note that Figure 1 (a) reports the ratio between the running time for matrices of sizes $n = 25 \times 2^k$ and $n = 25 \times 2^{k-1}$. Since the new companion eigensolver is an $O(n^2)$ algorithm, we expect the ratio to be close to 4 for large n.

7.2. Backward Stability Tests

If the new QR algorithm for companion matrix is backward stable in eigenproblem sense, then according to error analysis by Van Dooren and Dewilde [13], and further by Edelman and Murakami [16], the new algorithm is also backward stable in polynomial sense, more precisely, the "calculus" definition holds: "the first order perturbations of the matrix lead to first order perturbations of the coefficients", see [16] for details.

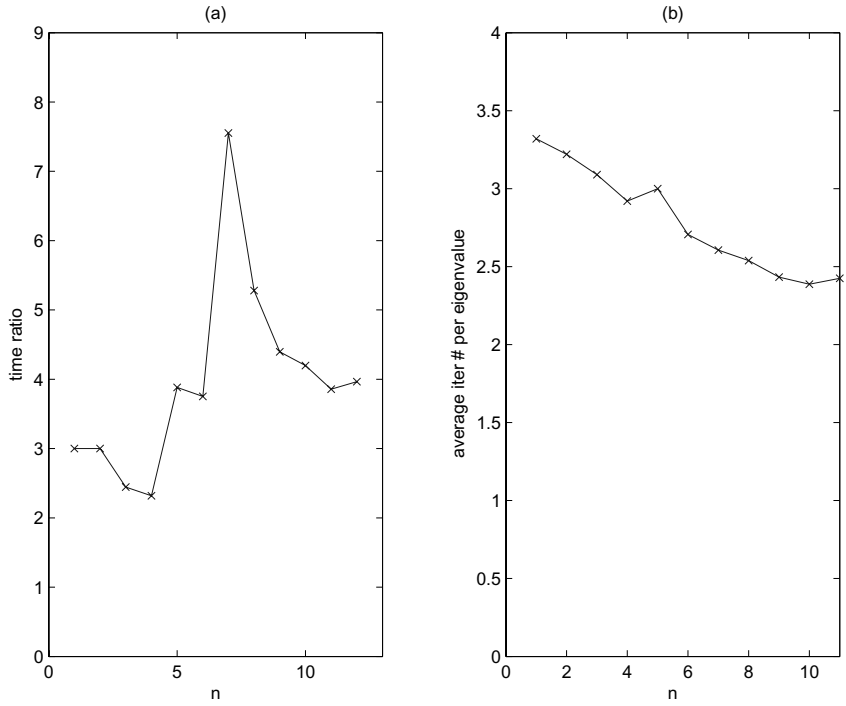

FIGURE 1. *New companion eigensolver, with test matrices of size* $25 \times 2^{n-1}$, $1 \le n \le 13$.

Following Toh and Trefethen [27] and Edelman and Murakami [16], we explore the following degree 20 monic real coefficient polynomials:

(1) "Wilkinson polynomial": zeros $1, 2, 3, \ldots, 20$.
(2) the monic polynomial with zeros $[-2.1 : 0.2 : 1.7]$.
(3) $p(z) = (20!) \sum_{k=0}^{20} z^k/k!$.
(4) the Bernoulli polynomial of degree 20.
(5) the polynomial $z^{20} + z^{19} + z^{18} + \cdots + z + 1$.
(6) the univariate polynomial with zeros $2^{-10}, 2^{-9}, 2^{-8}, \ldots, 2^9$.
(7) the Chebyshev polynomial of degree 20.

In addition, we tested some random polynomials of degree $100, 200, \ldots, 1600$:

(8) random coefficients with uniform distribution.

Like what Edelman and Murakami did in their paper [16], for each example above, we first computed the coefficients either exactly or with ultra-high precision using MPFUN90 (Multiple Precision package by David Bailey, [3]). Then we rounded these numbers to double precision (in F90). And we took the rounded polynomials stored in F90 to be our official test cases.

For all test cases, we computed two sets of relative backward errors. One is the norm-wise matrix relative backward error:

$$\frac{\|E\|_\infty}{\|\widehat{C}\|_\infty} \equiv \frac{\|\bar{G}^T \cdot \widehat{C} \cdot \bar{G} - Q^{(m)}R^{(m)}\|_\infty}{\|\widehat{C}\|_\infty},$$

where $\widehat{C}$ denotes the scaled companion matrix after balancing in (25), $\bar{G}$ is the accumulated orthogonal similarity transformation, and $Q^{(m)}R^{(m)}$ converges to the Schur form of $\widehat{C}$. The other is the component-wise coefficient relative backward error:

$$\frac{|\delta\,\widehat{c}_i|}{|\widehat{c}_i|} \equiv \frac{|\widetilde{c}_i - \widehat{c}_i|}{|\widehat{c}_i|},$$

where $\widehat{c}$ corresponds to the coefficient of the characteristic polynomial of $\widehat{C}$, and $\widetilde{c}_i$ is the ith coefficient of the polynomial recovered from the computed zeros by using ultra-high precision, e.g. MPFUN90.

Test	(1)	(2)	(3)	(4)	(5)	(6)	(7)
α	8	1	8	2	1	1/4	1/2
$\|\widehat{C}\|_\infty$	7	3	4	2	2	22	4
rel_bkerr	10^{-15}	10^{-15}	10^{-16}	10^{-15}	10^{-15}	10^{-16}	10^{-15}

TABLE 5. *Test* (1–7)*: matrix norm-wise backward errors.*

7.2.1. Test (1-7), degree 20.

Remark 7.3. 1. The last two rows of Table 6 show (1) x_{max}: the maximum positive root of $p_b(x) = x^n - |c_1|x^{n-1} - \cdots - |c_{n-1}|x_1 - |c_n|$, and (2) α: the particular scaling factor chosen so that the maximum coefficient backward error is minimized. As we can see, such α usually doesn't agree well with x_{max}. Although using x_{max} as scaling factor will minimize $\|DCD^{-1}\|_\infty$, the magnitudes of the coefficients of the new polynomial under such scaling could vary wildly.

2. The empty entries for Test 4 and 7 correspond to zero coefficients.

7.2.2. Test(8), random polynomials, degree $100, 200, \ldots 1600$**.**

Remark 7.4. 1. From Table 7, we can see that the new companion eigensolver has small backward error in matrix-norm sense, it also finds roots with small (coefficient) backward errors. In our random polynomial experiments, we choose $\alpha = 1$. When the size of polynomial gets bigger, to balance the corresponding companion matrix with geometric scaling limits our option.

2. Where the "average abs_bkerr" (average absolute backward error) is computed as average of $\{\log_{10}|c_i|\}$, and the "average rel_bkerr" (average relative backward error) is computed as average of $\left\{\log_{10}\frac{|\delta c_i|}{|c_i|}\right\}$.

index/Test	(1)	(2)	(3)	(4)	(5)	(6)	(7)
1	10^{-15}	10^{-14}	10^{-14}	10^{-14}	10^{-14}	10^{-14}	-
2	10^{-15}	10^{-14}	10^{-14}	10^{-14}	10^{-13}	10^{-14}	10^{-15}
3	10^{-14}	10^{-15}	10^{-14}	-	10^{-13}	10^{-13}	-
4	10^{-14}	10^{-12}	10^{-14}	10^{-14}	10^{-14}	10^{-13}	10^{-15}
5	10^{-14}	10^{-14}	10^{-14}	-	10^{-14}	10^{-13}	-
6	10^{-14}	10^{-13}	10^{-14}	10^{-14}	10^{-13}	10^{-13}	10^{-15}
7	10^{-14}	10^{-14}	10^{-14}	-	10^{-13}	10^{-13}	-
8	10^{-14}	10^{-14}	10^{-14}	10^{-14}	10^{-13}	10^{-13}	10^{-15}
9	10^{-14}	10^{-14}	10^{-14}	-	10^{-13}	10^{-13}	-
10	10^{-14}	10^{-15}	10^{-14}	10^{-14}	10^{-13}	10^{-13}	10^{-14}
11	10^{-14}	10^{-13}	10^{-14}	-	10^{-13}	10^{-13}	-
12	10^{-14}	10^{-14}	10^{-14}	10^{-14}	10^{-13}	10^{-13}	10^{-14}
13	10^{-13}	10^{-13}	10^{-14}	-	10^{-13}	10^{-13}	-
14	10^{-13}	10^{-14}	10^{-14}	10^{-14}	10^{-13}	10^{-13}	10^{-14}
15	10^{-13}	10^{-13}	10^{-14}	-	10^{-13}	10^{-13}	-
16	10^{-13}	10^{-13}	10^{-14}	10^{-14}	10^{-13}	10^{-13}	10^{-14}
17	10^{-13}	10^{-14}	10^{-14}	-	10^{-14}	10^{-13}	-
18	10^{-13}	10^{-13}	10^{-14}	10^{-13}	10^{-14}	10^{-13}	10^{-14}
19	10^{-13}	10^{-12}	10^{-14}	-	10^{-14}	10^{-12}	-
20	10^{-13}	10^{-13}	10^{-14}	10^{-14}	10^{-14}	10^{-12}	10^{-14}
max bkerr	10^{-13}	10^{-12}	10^{-14}	10^{-13}	10^{-13}	10^{-12}	10^{-14}
x_{max}	296.2	6.1	38.2	12.6	2.0	1319.8	2.6
α	8	1	8	2	1	1/4	1/2

TABLE 6. *Test* (1–7)*: coefficient-wise backward errors with appropriate* α*.*

size	matrix-wise		polynomial coeff.-wise			
	$\|\widehat{C}\|_\infty$	rel_bkerr	average abs_fwderr	average abs_fwderr		
100	5×10^1	3×10^{-15}	10^{-14}	10^{-13}		
200	9×10^1	7×10^{-15}	10^{-13}	10^{-13}		
400	2×10^2	2×10^{-14}	10^{-12}	10^{-12}		
800	4×10^2	3×10^{-14}	10^{-12}	10^{-11}		
1600	8×10^2	1×10^{-13}	10^{-11}	10^{-11}		

TABLE 7. *Test* (8)*: backward errors in matrix and polynomial coefficients.*

8. Conclusions

In this paper we presented a new fast QR algorithm for computing the eigenvalues of a real companion matrix. The algorithm is backward stable in practice. The success of the new method relies on (i) compact (SSS) representations for Q and

R, (ii) a new technique called Givens rotation swaps to update Q in an efficient fashion, and (iii) exploring the special rank structure of R for the purpose of efficient compression. The overall complexity is $O(n^2)$, though we have not yet derived the counts in detail. Our suspect is that the counts are similar to those in [6].

We also expect to propose a modified version with stability proof in the near future.

Acknowledgements

The authors are grateful to Professor Yuli Eidelman at Tel Aviv University and to the two anonymous referees for their valuable suggestions on this paper. We also thank Professor James Demmel and Doctor David Bindel at the University of California at Berkeley for their kind help in improving and testing the algorithm.

References

[1] Mario Ahues and Francoise Tisseur, *A new deflation criterion for the QR algorithm*, Technical Report CRPC-TR97713-S, Center for Research on Parallel Computation, January 1997.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, *LAPACK Users' Guide, Release 2.0*, SIAM, Philadelphia, PA, USA, second edition, 1995.

[3] D. Bailey, *Software: MPFUN90 (Fortran-90 arbitrary precision package)*, available online at http://crd.lbl.gov/~dhbailey/mpdist/index.html

[4] D. Bindel, S. Chandresekaran, J. Demmel, D. Garmire, and M. Gu, *A fast and stable nonsymmetric eigensolver for certain structured matrices*, Technical report, University of California, Berkeley, CA, 2005.

[5] D. A. Bini, F. Daddi, and L. Gemignani, *On the shifted QR iteration applied to companion matrices, Electronic Transactions on Numerical Analysis* **18** (2004), 137–152.

[6] 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*, Technical Report no.1587, Department of Mathematics, University of Pisa, 2005.

[7] S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semiseparable matrices*, SIAM J. Matrix Anal. Appl. **25** no. 2 (2003), 373–384.

[8] S. Chandrasekaran, M. Gu, and W. Lyons, *A fast and stable adaptive solver for hierarchically semi-separable representations*, Technical Report UCSB Math 2004-20, U.C. Santa Barbara, 2004.

[9] Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Fast stable solvers for sequentially semi-separable linear systems of equations and least squares problems*, Technical report, University of California, Berkeley, CA, 2003.

[10] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Some fast algorithms for sequentially semiseparable representations*, SIAM J. Matrix Anal. Appl **27** (2005), 341–364.

[11] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Mat. Anal. Appl., to appear.

[12] T.-Y. Chen and J.W. Demmel, *Balancing sparse matrices for computing eigenvalues*, Lin. Alg. and Appl. **309** (2000), 261–287.

[13] P. Van Dooren and P. Dewilde, *The eigenstructure of an aribtrary polynomial matrix: Computational aspects*, Lin. Alg. and Appl. **50** (1983), 545–579.

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

[15] Y. Eidelman, I. Gohberg and V. Olshevsky, *The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order*, Lin. Alg. and Appl. **404** (2005), 305–324.

[16] A. Edelman and H. Murakami, *Polynomial roots from companion matrix eigenvalues*, Mathematics of Computation **64** (1995), 763–776.

[17] G. Golub and C. V. Loan, *Matrix Computations*, The John Hopkins University Press, 1989.

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

[19] V. N. Kublanovskaya, *On some algorithms for the solution of the complete eigenvalue problem*, U.S.S.R. Comput. Math. and Math. Phys. **3** (1961), 637–657.

[20] C. Moler, *Roots – of polynomials, that is*, The Mathworks Newsletter **5** (1991), 8–9.

[21] V. Pan, *On computations with dense structured matrices*, Math. Comp. **55** (1990), 179–190.

[22] B. Parlett, *The symmetric eigenvalue problems*, SIAM, 1997.

[23] B. Parlett, *The QR algorithm*, Computing in Science and Engineering **2** (2000), 38–42. Special Issue: Top 10 Algorithms of the Century.

[24] G. Sitton, C. Burrus, J. Fox, and S. Treitel, *Factoring very-high-degree polynomials.* IEEE Signal Processing Mag. **20** no. 6 (2003), 27–42.

[25] M. Stewart, *An error analysis of a unitary Hessenberg QR algorithm*, Tech. Rep. TR-CS-98-11, Department of Computer Science, Australian National University, Canberra 0200 ACT, Australia, 1998.

[26] F. Tisseur, *Backward stability of the QR algorithm*, TR 239, UMR 5585 Lyon Saint-Etienne, October 1996.

[27] K.-C. Toh and L. N. Trefethen. *Pseudozeros of polynomials and pseudospectra of companion matrices.* Numer. Math. **68** (1994), 403–425.

[28] M. Van Barel and A. Bultheel, *Discrete Linearized Least Squares Rational Approximation on the Unit Circle*, J. Comput. Appl. Math. **50** (1994), 545–563.

[29] J. H. Wilkinson, *The algebraic eigenvalue problem*, Oxford University Press, London, 1965.

[30] J. Xia, *Fast Direct Solvers for Structured Linear Systems of Equations*, Ph.D. Thesis, University of California, Berkeley, 2006.

[31] J. Zhu, *Structured Eigenvalue Problems and Quadratic Eigenvalue Problems*, Ph.D. Thesis, University of California, Berkeley, 2005.

Shiv Chandrasekaran
Department of Electrical and Computer Engineering
University of California at Santa Barbara
USA
e-mail: shiv@ece.ucsb.edu

Ming Gu
Department of Mathematics
University of California at Berkeley
USA
e-mail: mgu@math.berkeley.edu

Jianlin Xia
Department of Mathematics
University of California at Los Angeles
USA
e-mail: jxia@math.ucla.edu

Jiang Zhu
Department of Mathematics
University of California at Berkeley
USA
e-mail: zhujiang.cal@gmail.com