Implicit double shift QR-algorithm for companion matrices

marc Van Barel, Raf Vandebril, Paul Van Dooren en Katrijn Frederix

Marc Van Barel

Department of Computer Science

KU Leuven, Belgium

Marc.VanBarel@cs.kuleuven.be

Raf Vandebril

Department of Computer Science

KU Leuven, Belgium

Raf.Vandebril@cs.kuleuven.be

Katrijn Frederix

Department of Computer Science

KU Leuven, Belgium

Katrijn.Frederix@cs.kuleuven.be

Paul Van Dooren

Department of Mathematical Engineering

Université Catholique de Louvain, Belgium

Paul.VanDooren@uclouvain.be

Abstract

In this paper an implicit (double) shifted QR-method for computing the eigenvalues of companion and fellow matrices will be presented. Companion and fellow matrices are Hessenberg matrices, that can be decomposed into the sum of a unitary and a rank 1 matrix. The Hessenberg, the unitary as well as the rank 1 structures are preserved under a step of the QR-method. This makes these matrices suitable for the design of a fast QR-method. Several techniques already exist for performing a QR-step. The implementation of these methods is highly dependent on the representation used. Unfortunately for most of the methods compression is needed since one is not able to maintain all three, unitary, Hessenberg and rank 1 structures. In this manuscript an implicit algorithm will be designed for performing a step of the QR-method on the companion or fellow matrix based on a new representation consisting of Givens transformations. Moreover, no compression is needed as the specific representation of the involved matrices is maintained. Finally, also a double shift version of the implicit method is presented.

Article information

KU LEUVEN

Implicit double shift QR-algorithm for companion matrices *

Marc Van Barel[1], Raf Vandebril[1], Paul Van Dooren[2], Katrijn Frederix[1]

[1] Katholieke Universiteit Leuven, Dept. of Computer Science, Celestijnenlaan 200A, B-3001 Leuven (Heverlee), Belgium
e-mail: {raf.vandebril,marc.vanbarel,katrijn.frederix}@cs.kuleuven.be

[2] Université Catholique de Louvain, Dept. of Mathematical Engineering, Avenue Georges Lemaitre 4, B-1348 Louvain-la-Neuve, Belgium
e-mail: paul.vandooren@uclouvain.be

March 16, 2010

Abstract In this paper an implicit (double) shifted QR-method for computing the eigenvalues of companion and fellow matrices will be presented. Companion and fellow matrices are Hessenberg matrices, that can be decomposed into the sum of a unitary and a rank 1 matrix. The Hessenberg, the unitary as well as the rank 1 structures are preserved under a step of the QR-method. This makes these matrices suitable for the design of a fast QR-method.

Several techniques already exist for performing a QR-step. The implementation of these methods is highly dependent on the representation used. Unfortunately for most of the methods compression is needed since one is not able to maintain all three, unitary, Hessenberg and rank 1 structures.

In this manuscript an implicit algorithm will be designed for performing a step of the QR-method on the companion or fellow matrix based on a new representation consisting of Givens transformations. Moreover, no compression is needed as the specific representation of the involved matrices is maintained. Finally, also a double shift version of the implicit method is presented.

Key words Companion matrices, Fellow matrices, QR-algorithm, implicit method, multishift

AMS-Classification 65F15, 11C20

1 Introduction

Computing roots of polynomials is an interesting problem with many challenges. Many methods act directly on the polynomials, e.g., bisection, Sturm sequences and so forth [17]. In actual implementations, however, the problem often is formulated in a matrix setting and the corresponding matrix problem is solved. When working with the standard monomial basis, computing roots of a polynomial coincides with computing eigenvalues of the associated matrix. The eigenvalues of a companion matrix equal the roots of the associated polynomial.

Definition 1 *Given a monic polynomial $p(z) = p_0 + p_1 z + p_2 z^2 + \ldots + p_{n-1} z^{n-1} + z^n$ the associated companion matrix is of the following form:*

$$C_p = \begin{bmatrix} 0 & & & -p_0 \\ 1 & \ddots & & -p_1 \\ & \ddots & & \vdots \\ & & 1 & -p_{n-1} \end{bmatrix}. \tag{1}$$

When changing the basis the idea remains the same, but the associated matrix changes, one obtains e.g., confederate, congenial, comrade matrices and so forth [1]. All these problems are worth investigating, but in this manuscript we will restrict ourselves to the class of companion and fellow matrices because it is the most widespread.

In the current LAPACK implementation the eigenvalues of the companion matrix are computed by the standard $O(n^3)$ Hessenberg eigenvalue solver. This is an unfortunate choice, since a companion matrix and its *QR*-iterates are highly structured as we will see later on.

Recently, many authors have been intrigued by hidden rank properties in dense matrices. Basic theoretical results on rank structured matrices [20, 16], quickly evolved to system solvers for these matrices [14, 12] and to eigenvalue methods based on these matrix structures [19, 15, 11].[1]

A companion matrix, which is Hessenberg, can be written as the sum of a unitary and a rank 1 matrix. Since both the unitary and the rank 1 matrix are structured rank matrices, one wishes to exploit these structures to come to an efficient method. The brute-force Hessenberg eigenvalue solver uses $O(n^3)$ operations for computing the whole spectrum, whereas an efficient method exploiting all involved matrix structures would only use $O(n^2)$ operations. The problem has already been studied extensively. Let us make a summary of the existing methods crediting the authors who have contributed to the development of these methods.

Independently of each other Bindel et al. and Bini et al. [5, 2] proved that the *QR*-iterates H_k have $\{1,3\}$-quasiseparable structure when $H_0 = U + \mathbf{u}\mathbf{v}^H$ is a Hessenberg matrix that is unitary plus rank one. Hence, each Hessenberg matrix H_k can be represented using $O(n)$ parameters. Based on this fact, several algorithms where developed that performed *QR*-iteration steps on the Hessenberg matrices H_k. These algorithms differ in the way the Hessenberg matrix is represented, in the way the Hessenberg as well as the unitary plus rank one structure is maintained and in the explicit or implicit way of performing each *QR*-iteration step.

In the paper by Bini et al., the following relationship between C_p of (1) and its inverse C_p^{-1} is used (when p_0 is different from zero):

$$C_p = C_p^{-H} + UV^H,$$

with $U_k, V_k \in \mathbb{C}^{n \times 2}$. The *QR*-iteration step with shift is implemented in an explicit way. The experiments described in the paper show that the implementation of the algorithm has numerical difficulties (overflow/underflow, no convergence).

In the manuscript of Bindel et al. a larger class of matrices is considered, more precisely, the symmetric, skew symmetric or orthogonal matrices plus a low rank modification. The algorithm uses a compression technique to maintain the structure.

In Bini et al. [6] an alternative algorithm was developed to solve the Hessenberg, unitary plus rank one case $H_k = U_k + \mathbf{u}_k \mathbf{v}_k^H$. The authors showed that the matrix U_k can be written as a product of three sequences of 2×2 unitary transformations as represented in (9). The presented method is an explicit *QR*-algorithm, in which special compression is needed to maintain the unitary structure.

As in the paper of Bindel et al., a bulge chasing procedure is designed in [9]. In contrast to the previous papers this is done for a single as well as for a double shift. The representation that is used for the Hessenberg matrix H_k in each step is its *QR*-factorization, i.e., $H_k = Q_k R_k$. A chasing technique is used to chase the implicitly created bulge. Unfortunately this increases the rank of a certain involved factor. Compression is therefore needed to maintain a compact representation.

---

[1] The literature related to this subject is much more extended than these few references (see [18]).

All the previous papers handled the case $H = U + \mathbf{u}\mathbf{v}^H$, i.e., when the Hessenberg matrix H is the sum of a unitary U and rank one matrix $\mathbf{u}\mathbf{v}^H$. Only the manuscript of Bindel et al. considered the more general class of symmetric, skew symmetric, or unitary plus rank one case. In the paper [7] by Bini et al. of 2005, the matrices $A = (a_{i,j})_{i,j}$ considered have a quasiseparable representation of the following form:

$$a_{ii} = d_i + z_i \overline{w}_i \tag{2}$$
$$a_{ij} = u_i t_{ij}^{\times} \overline{v}_j \qquad \text{when } i > j$$
$$a_{ij} = \overline{u}_j \overline{t_{ji}^{\times}} v_i + z_i \overline{w}_j - \overline{z}_j w_i \qquad \text{when } i < j,$$

with $t_{ij}^{\times} = t_{i-1}t_{i-2}\cdots t_{j+1}$ and given vectors $\mathbf{u}, \mathbf{v}, \mathbf{t}, \mathbf{z}, \mathbf{w}, \mathbf{d}$. This set is called the set of generalized companion matrices. It includes the arrowhead matrices, comrade matrices (symmetric tridiagonal plus rank one), diagonal plus rank one matrices, ... Note that each matrix of this set is $\{1,3\}$-quasiseparable. The authors prove that this set is closed under the application of each step of the *QR*-algorithm. Each step of the *QR*-algorithm is performed in an explicit way, i.e., the *QR*-factorization of $A_k - \mu_k I$ is computed and then the two factors multiplied in reverse order are added to $\mu_k I$.

When finishing this manuscript, we became aware of another similar algorithm to compute the eigenvalues of a companion matrix. This method was explained by Boito at a workshop in Cortona [4].

In this manuscript we will present a new manner of representing the companion matrix, which uses less parameters than the ones presented in the manuscripts above. Moreover, during the algorithm no compression is needed to maintain a low complexity algorithm. The representation is based on Givens transformations and easily admits an implicit *QR*-algorithm. The implicit version is based on structure restoring manipulations of the involved matrix, in order to maintain the same representation in each step of the *QR*-method. Once the single shift version is known, the double shift or multishift version is a straightforward extension.

The manuscript is organized as follows. In Section 2 some preliminary results are discussed. The relation between roots of polynomials and companion matrices is investigated and some comments on working with Givens transformations are given. In the following sections the different aspects to the development of the *QR*-methods are discussed. Section 3 discusses unitary plus rank one matrices, both the preservation of the structure and the representation are investigated. Section 4 and Section 5 discuss both the single and double shift techniques. Due to the special representation of the involved matrices the deflation is uncommon, Section 6 discusses how to perform deflation. Finally some details on the implementation and numerical experiments are presented.

2 Preliminary results

In this section we will firstly discuss the relation between polynomials and matrices. Secondly we will present some tools for working with graphical interpretations of Givens transformations.

2.1 Roots of polynomials

Suppose we are working with a polynomial $p(z)$ either in $\mathbb{R}_n[z]$ or $\mathbb{C}_n[z]$. The subscript n in $\mathbb{R}_n[z]$ and $\mathbb{C}_n[z]$ means that the considered polynomials are of degree less than or equal to n.

Without loss of generality we can assume the coefficient p_n of z^n to be different from zero. Clearly the zeros of $p(z)$ equal the zeros of $p(z)/p_n$, hence we will assume in the remainder the polynomial $p(z)$ to be monic, i.e., with $p_n = 1$.

The companion matrix associated with a monic polynomial of degree n is given in Equation (1).

Note 1 One can fairly easy see that the eigenvalues of the companion matrix coincide with the zeros of the associated polynomial $p(z)$, because $p(z) = \det(zI - C_p)$.

A theoretical, more broad context, not restricted to the standard monomial basis exists for proving the equality between eigenvalues of the associated matrix and roots of the polynomial. More information can be found for example in [1,21].

2.2 Working with Givens transformations

Givens transformations are a powerful tool for working with structured matrices. E.g., the implicit QR-algorithm for Hessenberg matrices makes extensive use of Givens transformations [17] and especially for structured rank matrices [11] they are a valuable tool.

Also for structured rank matrices, Givens transformations play an important role. In this subsection we will discuss a graphical way of working with Givens transformations. Also interactions such as the "shift through" and "the fusion" of Givens transformations will be discussed.

The following graphical scheme represents the QR-factorization of a 6×6 Hessenberg matrix, based on Givens transformations. Often Givens transformations of higher dimensions, say n, are considered. This means that the corresponding 2×2 Givens transformation is embedded in the identity matrix of dimension n, still changing only two rows when applied to the left. The scheme corresponds to the matrix product $G_1 G_2 \ldots G_5 R$. The matrix R is shown on the right, depicted by $\times$, and is clearly upper triangular. The Givens transformation G_5 is located above 1, G_4 in position 2 and so forth. One might consider this misleading, but one should think of the bottom line as a sort of timeline depicting which transformation needs to be applied first. Rewriting the formula we have $G_5^H \ldots G_1^H H = R$. Hence G_1^H annihilates the first subdiagonal element, G_2^H the second and so forth.
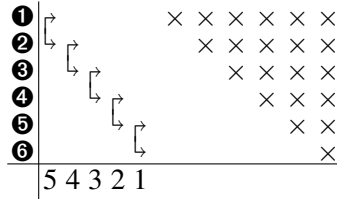
In the remainder of the manuscript we will make extensive use of such schemes, with many more Givens transformations.

Givens transformations also interact with each other.

Lemma 1 *Suppose two Givens transformations[2] G_1 and G_2 are given:*

$$G_1 = \begin{bmatrix} c_1 & -\bar{s}_1 \\ s_1 & \bar{c}_1 \end{bmatrix} \text{ and } G_2 = \begin{bmatrix} c_2 & -\bar{s}_2 \\ s_2 & \bar{c}_2 \end{bmatrix}.$$

Then we have that $G_1 G_2 = G_3$ is again a Givens transformation. We will call this the fusion of Givens transformations in the remainder of the text.

The proof is trivial. In our graphical schemes, we will depict this as follows:
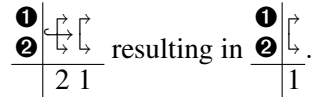
The following lemma is very powerful and will give the possibility to interchange the order of Givens transformations and to obtain different patterns.

Lemma 2 (Shift through lemma) *Suppose three 3×3 Givens transformations $\check{G}_1, \check{G}_2$ and $\check{G}_3$ are given, such that the Givens transformations $\check{G}_1$ and $\check{G}_3$ act on the first two rows of a matrix, and $\check{G}_2$ acts on the second and third row (when applied on the left to a matrix).*

Then there exist 3 Givens transformations $\hat{G}_1, \hat{G}_2$ and $\hat{G}_3$ such that

$$\check{G}_1 \check{G}_2 \check{G}_3 = \hat{G}_1 \hat{G}_2 \hat{G}_3,$$

where $\hat{G}_1$ and $\hat{G}_3$ work on the second and third row and $\hat{G}_2$, works on the first two rows.

---

[2] In fact these transformations are rotations. Throughout the manuscript we assume to be working with rotations. More information on Givens rotations can be found in [3].

This result is well-known. The proof can be found in [22] and is simply based on the fact that one can factorize a 3×3 unitary matrix in different ways. Graphically we will depict this rearrangement as follows.

Of course there is also a variant in the other direction (from the right to the left scheme depicted by $\curvearrowright$).

Important to remark is that the fusion of Givens transformations can be considered as a special case of the shift through lemma. For our purposes we also need an additional operation, which is in fact also a special case of the shift through lemma, namely the scaled fusion.

Lemma 3 *Suppose two Givens transformations G_1, G_2 and I_α, with $|\alpha| = 1$ are given,*

$$G_1 = \begin{bmatrix} c_1 & -\bar{s}_1 \\ s_1 & \bar{c}_1 \end{bmatrix}, \ G_2 = \begin{bmatrix} c_2 & -\bar{s}_2 \\ s_2 & \bar{c}_2 \end{bmatrix} \ and \ I_\alpha = \begin{bmatrix} 1 & 0 \\ 0 & \alpha \end{bmatrix}.$$

Then we have that $G_1 I_\alpha G_2 = G_3 I_\alpha = I_\alpha G_4$, with G_3 and G_4 again a Givens transformation. We will call this the scaled fusion of Givens transformations in the remainder of the text.

The proof involves straightforward computations, by embedding I_α in a 3×3 Givens transformation matrix, applying the shift through lemma and exploiting the fact that the matrix in which I_α is embedded is a diagonal matrix.

Graphically we will formulate this as:

Note 2 We note that the matrix I_α can be replaced by a matrix J_α of the following form:

$$J_\alpha = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix}.$$

leading to the following equations: $G_1 J_\alpha G_2 = G_3 J_\alpha = J_\alpha G_4$.

This scaled fusion is necessary since in the forthcoming *QR*-algorithm Givens transformations will tend to become diagonal. In the real case these Givens transformations equal the identity and hence there are no problems, but in the general complex setting they have a diagonal consisting of α and $\bar{\alpha}$ with $|\alpha| = 1$. Since these transformations will be surrounded by other Givens transformations some special operations are needed in order to manipulate them in an elegant way.

A final important operation is the shift through operation of length ℓ.

Lemma 4 *Suppose we have the following matrix product GWX, in which G denotes a Givens transformation acting on row 1 and 2. The matrices W and X are both unitary matrices consisting of a descending sequence of ℓ Givens transformations. This means that both W and X consist of ℓ successive Givens transformations. The ith Givens transformation G_i^W of W acts on row $i+1$ and $i+2$. The ith Givens transformation G_i^X of X acts on row i and $i+1$.*

The matrix product GWX can then be rewritten as

$$GWX = \hat{W}\hat{X}\hat{G},$$

where $\hat{G}$ is now a Givens transformation acting on row $\ell+1$ and $\ell+2$. The unitary matrices $\hat{W}$ and $\hat{X}$ are again descending sequences of ℓ Givens transformations.

Before proving the statement we will depict this graphically. Initially on the left we have the following scheme, in which G is marked by a $\times$. The unitary matrices W and X are both sequences of 4 Givens transformations, W ranges from position 8 up to 5 and X ranges from position 4 up to 1. The $\overset{\ell}{\curvearrowright}$ indicates that a shift through operation of length ℓ will be performed.

$$
\tag{3}
$$
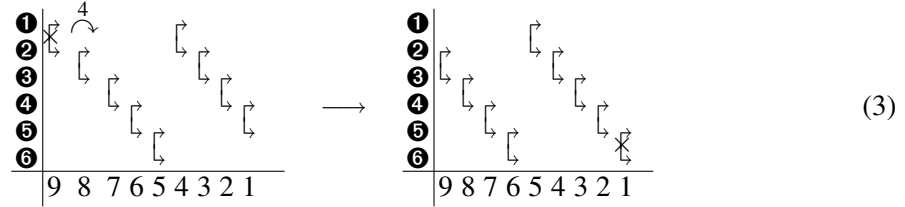
On the right we see the resulting scheme which is equivalent to the first one. The Givens transformation $\hat{G}$ is again denoted by a $\times$. The new sequence $\hat{W}$ ranges from 9 upto 6 and $\hat{X}$ ranges from 5 up to 2.

In the text we will refer to this as a shift through operation of length ℓ. One can also perform a shift through operation from right to left. This will be referred to as a shift through operation of length ℓ to the left.

Proof The proof consists of nothing else than a successive application of the shift through operation. Without loss of generality we assume $\ell = 4$. Hence we will prove the transition from left to right in Scheme (3).

Reordering permits the application of the shift through operation. This leads to the right scheme. We have marked another Givens transformation now with $\times$.
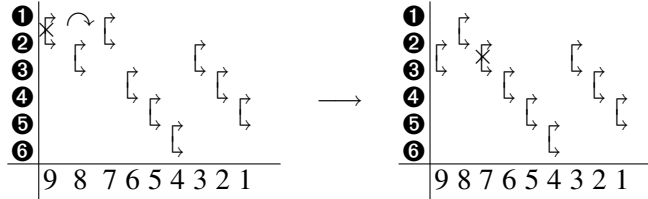
Rewriting gives us the possibility to apply again the shift through operation, resulting after rewriting in the right scheme.
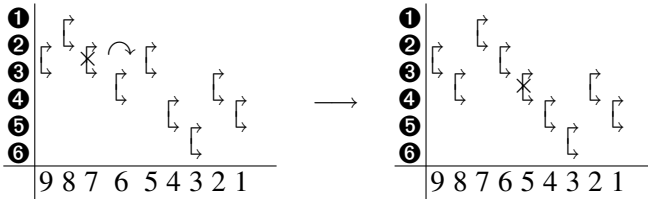
Clearly the undesired Givens transformation moves down, creating two new sequences, starting in positions 9 and 8 and removing the sequences in positions 4 up to 1. Continuing this procedure, and applying two more times the shift through operation gives us Scheme (4).

$$
\tag{4}
$$

This corresponds to the desired situation. Mathematically we obtain $\hat{W}\hat{X}\hat{G}$, where $\hat{W}$ denotes the sequence of Givens transformations ranging from position 9 up to position 6 and $\hat{X}$ denotes the sequence ranging from position 5 up to position 2. The resulting Givens transformation in position 1 is $\hat{G}$.

Note 3 One can easily see that the shift through operation is a special case of Lemma 4, where the two sequences W and X are of length 1.

3 Unitary plus rank one matrices

In this manuscript we will develop *QR*-methods for computing eigenvalues of Hessenberg, unitary plus rank one matrices. This is an important class of matrices as the companion and the fellow matrix are both of this form. Hence, the problem of computing zeros of polynomials can be translated to the eigenproblem of a Hessenberg, unitary plus rank one matrix. Let us divide this section into small parts discussing some ingredients for developing an implicit *QR*-method.

3.1 Structure under a QR-step

Let us denote the Hessenberg matrix we are working with as follows

$$H = U + \mathbf{u}\mathbf{v}^H, \tag{5}$$

with H Hessenberg, U unitary and $\mathbf{u}$ and $\mathbf{v}$ two vectors.

Suppose we have a shift μ and we would like to perform a step of the *QR*-iteration on the matrix H. We have the following formulas

$$H - \mu I = QR$$
$$\hat{H} = RQ + \mu I = Q^H H Q.$$

Applying now the similarity transformation on the terms of Equation (5) we obtain the relations $\hat{H} = Q^H H Q = Q^H U Q + Q^H \mathbf{u}\mathbf{v}^H Q = \hat{U} + \hat{\mathbf{u}}\hat{\mathbf{v}}^H$, with $\hat{H}$ Hessenberg, $\hat{U}$ unitary and $Q^H \mathbf{u}$ and $Q^H \mathbf{v}$ two vectors.

Hence the Hessenberg as well as the unitary plus rank 1 structure are preserved under a step of the *QR*-iteration. This is essential for developing an efficient implicit *QR*-method exploiting the matrix structure.

3.2 A representation for the unitary matrix

Since the unitary matrix in the splitting of the Hessenberg matrix is in general a dense matrix, we want to represent this matrix as efficiently as possible. Even though the matrix is full, it has a structured rank part.

Consider $U = H - \mathbf{u}\mathbf{v}^H$, since the matrix H is Hessenberg and has zeros below the subdiagonal, the matrix U needs to be of rank 1 below the subdiagonal. The matrix U is therefore unitary.

We will construct the representation of this matrix in a graphical form for a 6×6 example. We annihilate the vector $\mathbf{u}$ completely, except the first element, which will contain the weight of the entire vector $\mathbf{u}$.
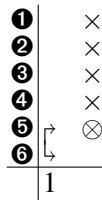
Let us briefly explain the scheme above as similar ones will be used throughout the entire manuscript. The right part of the scheme depicts a matrix (in this case it is a vector). The bracket above 1 indicates a single Givens transformation acting on rows ❺ and ❻ of the matrix. Mathematically, the scheme depicts

$$\mathbf{u} = G_1 \mathbf{u}_1, \tag{6}$$

where G_1 corresponds to the Givens transformation in position 1, and $\mathbf{u}_1$ is the matrix on the right having an extra zero in the last row. In fact the scheme here presents a sort of factorization of the original vector $\mathbf{u}$. The element to be annihilated by the second Givens transformation is marked in the scheme with $\otimes$. Applying a second transformation gives us $\mathbf{u} = G_1 G_2 \mathbf{u}_2$. In the following scheme we see the vector $\mathbf{u}_2$ on the right, Givens transformation G_1 above 2 and G_2

above 1. Recall that the bottom line depicts a time line, indicating in which order the Givens transformation have to be applied on the vector $\mathbf{u}_2$ (see Subsection 2.2).

$$
\begin{array}{c|ccc}
\mathbf{1} & & & \times \\
\mathbf{2} & & & \times \\
\mathbf{3} & & & \times \\
\mathbf{4} & & \nearrow\,\searrow & \otimes \\
\mathbf{5} & \nearrow\,\searrow & & \\
\mathbf{6} & \searrow & & \\
\hline
& 2 & 1 &
\end{array}
$$
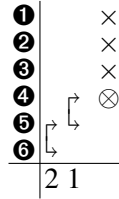
After the Givens transformations acting on row ❸ and ❹, ❷ and ❸ and ❶ and ❷ are performed we have completely removed the vector except for the top element.

$$
\begin{array}{c|cccccc}
\mathbf{1} & & & & & \nearrow\,\searrow & \times \\
\mathbf{2} & & & & \nearrow\,\searrow & & \\
\mathbf{3} & & & \nearrow\,\searrow & & & \\
\mathbf{4} & & \nearrow\,\searrow & & & & \\
\mathbf{5} & \nearrow\,\searrow & & & & & \\
\mathbf{6} & \searrow & & & & & \\
\hline
& 5 & 4 & 3 & 2 & 1 &
\end{array}
\tag{7}
$$

To achieve the representation of U, the Hessenberg matrix H has to undergo the same operations, $G_1, \ldots, G_5$, as the vector $\mathbf{u}$:

$$
H = \begin{bmatrix}
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times \\
 & \times & \times & \times & \times & \times \\
 & & \times & \times & \times & \times \\
 & & & \times & \times & \times \\
 & & & & \times & \times
\end{bmatrix}.
$$

Applying the first Givens transformation, G_1, will create a fill in element at position $(6,5)$, this is denoted by $\otimes$ in the scheme.

$$
\begin{array}{c|cccccc}
\mathbf{1} & \times & \times & \times & \times & \times & \times \\
\mathbf{2} & \times & \times & \times & \times & \times & \times \\
\mathbf{3} & & \times & \times & \times & \times & \times \\
\mathbf{4} & & & \times & \times & \times & \times \\
\mathbf{5} & \nearrow & & & \times & \times & \times \\
\mathbf{6} & \searrow & & & \otimes & \times & \times \\
\hline
& 1 & & & & &
\end{array}
$$
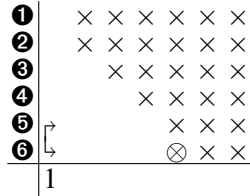
The other four Givens transformations, $G_2, \ldots, G_5$ will also create fill in elements, and the matrix remaining on the right side is now a generalized Hessenberg matrix having two subdiagonals.

$$
\begin{array}{c|cccccccccccc}
\mathbf{1} & & & & & \nearrow\,\searrow & \times & \times & \times & \times & \times & \times \\
\mathbf{2} & & & & \nearrow\,\searrow & & \times & \times & \times & \times & \times & \times \\
\mathbf{3} & & & \nearrow\,\searrow & & & \otimes & \times & \times & \times & \times & \times \\
\mathbf{4} & & \nearrow\,\searrow & & & & & \otimes & \times & \times & \times & \times \\
\mathbf{5} & \nearrow\,\searrow & & & & & & & \otimes & \times & \times & \times \\
\mathbf{6} & \searrow & & & & & & & & \otimes & \times & \times \\
\hline
& 5 & 4 & 3 & 2 & 1 & & & & & &
\end{array}
\tag{8}
$$
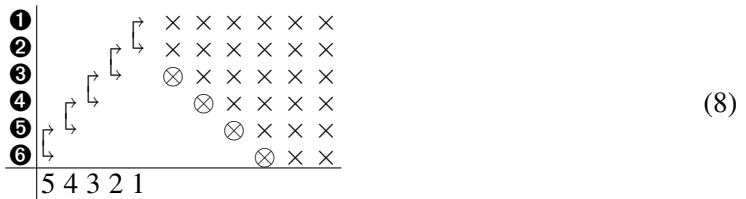
By construction the Givens transformations in position 5 to 1 in Scheme (8) are exactly the same as the ones in position 5 to 1 of Scheme (7). Subtraction of the factorization $\mathbf{u}\mathbf{v}^H$ will not affect the above structure, only the elements on the top row of the right side will change. This means that the unitary matrix U can be represented as in Scheme (8).

Pealing of the second subdiagonal, removing successively all elements marked with $\otimes$ in Scheme (8), will give us an extra descending sequence of Givens transformations. These Givens transformations can be found in positions 1 up to 4.

$$
\begin{array}{c|ccccccccccccc}
\mathbf{1} & & & & & & \nearrow\,\searrow & & \times & \times & \times & \times & \times & \times \\
\mathbf{2} & & & & \nearrow\,\searrow & \nearrow\,\searrow & & & \otimes & \times & \times & \times & \times & \times \\
\mathbf{3} & & & \nearrow\,\searrow & & & \nearrow\,\searrow & & & \otimes & \times & \times & \times & \times \\
\mathbf{4} & & \nearrow\,\searrow & & & \nearrow\,\searrow & & \nearrow\,\searrow & & & \otimes & \times & \times & \times \\
\mathbf{5} & \nearrow\,\searrow & & & & & & \nearrow\,\searrow & & & & \otimes & \times & \times \\
\mathbf{6} & \searrow & & & & & & & \searrow & & & & \otimes & \times \\
\hline
& 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & & & &
\end{array}
$$
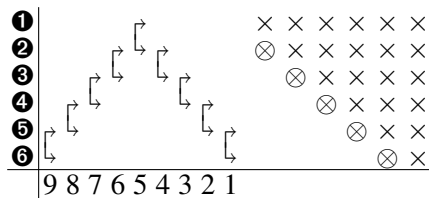
In the final step the remaining subdiagonal from the matrix on the right is removed. This will be done by a descending sequence of 5 Givens transformations. We obtain the following scheme.

$$\tag{9}$$
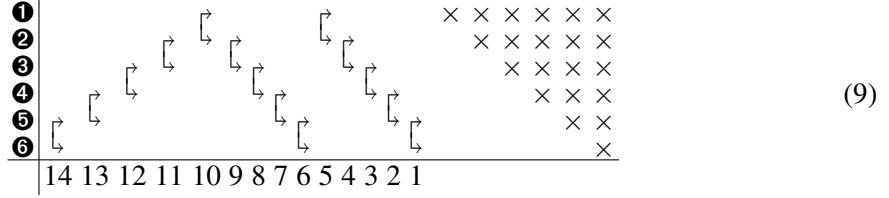
In fact we have computed now a QR-factorization of the unitary matrix consisting of three sequences of Givens transformations: $U = QR$. Since the matrix R is upper triangular and also unitary, the matrix will be diagonal. Moreover, the final sequence of Givens transformations can be chosen such that the diagonal matrix R has all diagonal elements except the last one equal to 1, hence $R = I_\alpha$, with $|\alpha| = 1$. Graphically this is depicted as follows (we do not depict the ones on the diagonal of I_α):

$$\tag{10}$$
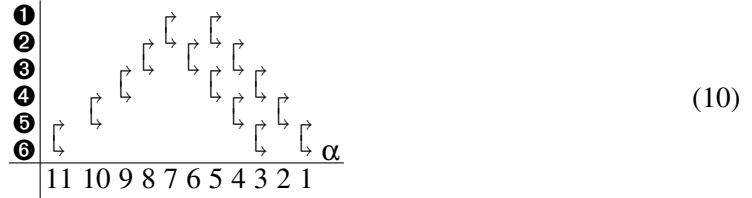
Denoting in Scheme (9) the product of the Givens transformations 14 to 10 by V, those from 9 to 6 by W and of those from 5 to 1 by X, we have factored the unitary matrix U as the product of three unitary matrices $U = VWXI_\alpha$. We use three different symbols for the unitary matrices, because the use of too many sub and superscripts would make the forthcoming implicit method difficult to read.

This will be the representation of the unitary matrix in the sum $H = U + \mathbf{u}\mathbf{v}^H$. As stated before the Givens transformations in position 11 upto 7 in Scheme (10) are identical to the ones in position 5 to 1 of Scheme (7).

This means that we get the following equalities:

$$\begin{aligned} H = U + \mathbf{u}\mathbf{v}^H &= VWXI_\alpha + \mathbf{u}\mathbf{v}^H \\ &= V\left(WXI_\alpha + V^H\mathbf{u}\mathbf{v}^H\right) = V\left(WXI_\alpha + \hat{\mathbf{u}}\mathbf{v}^H\right), \end{aligned}$$

where the vector $\hat{\mathbf{u}}$ has only the first element different from zero (this is the vector on the right in Scheme (7)).

In the next section we will discuss the implementation of the implicit single shifted QR-method based on this representation.

4 The single shift method

There exist several variants for performing an implicit QR-method on unitary plus low rank matrices. Unfortunately most of these algorithms are not capable of preserving both the unitary and the low rank structure. Numerical roundoff creates loss of the low rank structure and hence a form of rank compression is needed. This compression can create undesired results such as loss of accuracy. Often also an increase of parameters for representing the unitary plus low rank matrix is needed to deal with the loss of structure.

In this section we will discuss how to perform an implicit QR-step on the companion matrix, exploiting thereby the developed representation.

4.1 Implicit QR-methods

Assume we have $A - \mu I = QR$, with A, e.g., of tridiagonal, Hessenberg or semiseparable form. Perform now a unitary similarity transformation $\hat{A} = \hat{Q}^H A \hat{Q}$ with $\hat{Q}$ having $\hat{Q}\mathbf{e}_1 = Q\mathbf{e}_1$ such that $\hat{A}$ has the same structural constraints as assumed for A. Then one can prove that the matrix $\hat{A}$ is essentially the same as the matrix coming from an explicit *QR*-step with shift μ performed on the matrix A.

An implicit *QR*-method consists of performing the transformation $\hat{Q}^H A \hat{Q}$ in such a manner that the full matrix Q is not needed beforehand, but is determined on the fly.

The global flow of an implicit method is the following.

– Compute an orthogonal transformation $\hat{Q}_1$, such that $\hat{Q}_1^H (A - \mu I)\mathbf{e}_1 = \beta \mathbf{e}_1$.
– Perform a similarity transformation with $\hat{Q}_1$ on the matrix A: $\hat{Q}_1^H A \hat{Q}_1$.
– Perform now a new similarity transformation with the matrix $\hat{Q}_2$, with $\hat{Q}_2\mathbf{e}_1 = \mathbf{e}_1$, such that $\hat{Q}_2^H \hat{Q}_1^H A \hat{Q}_1 \hat{Q}_2$ has the same structural constraints as the original matrix A.

The first two items are often named the initialization step, whereas the third step is often called the chasing step, since in the Hessenberg and tridiagonal case, bulges are chased away to restore the Hessenberg or tridiagonal structure. This is also the subdivision used for the next two subsections.

4.2 Initialization

The implicit *QR*-step performed on a Hessenberg matrix is determined by the first Givens transformation G_1 (corresponds to $\hat{Q}_1$ from the previous subsection), such that

$$G_1^H (H - \mu I)\mathbf{e}_1 = \pm \|(H - \mu I)\mathbf{e}_1\| \mathbf{e}_1.$$

We will now perform the similarity transformation $G_1^H H G_1$, exploiting the factorization of the matrix H (for simplicity we omit the hat and write $\mathbf{u}$ instead of $\hat{\mathbf{u}}$)

$$H = V \left(WXI_\alpha + \mathbf{u}\mathbf{v}^H \right). \tag{11}$$

The idea is to obtain again a Hessenberg matrix, after the complete *QR*-step is finished. Moreover, we want the resulting Hessenberg matrix to satisfy the same representation as shown in (11). Of course after applying the first similarity transformation based on G_1, the structure will not be exactly as desired. Hence extra Givens transformations will be constructed to restore the structure. This will be the chasing. Throughout the entire procedure we want the representation to remain as close as possible to the original representation of H.

Let us perform the first similarity transformation on the matrix $H = H_0 = V_0 \left(W_0 X_0 I_\alpha + \mathbf{u}\mathbf{v}_0^H \right)$, we do not add a subscript to $\mathbf{u}$ since this vector will not alter. We obtain

$$\begin{aligned} H_1 &= G_1^H V_0 \left(W_0 X_0 I_\alpha + \mathbf{u}\mathbf{v}_0^H \right) G_1 \\ &= G_1^H V_0 \left(W_0 X_0 I_\alpha G_1 + \mathbf{u}\mathbf{v}_0^H G_1 \right) \\ &= G_1^H V_0 \left(W_0 X_0 I_\alpha G_1 + \mathbf{u}\mathbf{v}_1^H \right), \end{aligned}$$

with $\mathbf{v}_1^H = \mathbf{v}_0^H G_1$. In the remainder of the derivations, all intermediate variables will be depicted with a tilde or a hat and all final values for representing H_1 will be $V_1, W_1, X_1, \mathbf{u}$ and $\mathbf{v}_1$.

Since the Givens transformation G_1^H acts on the first two rows, and V_0 acts on the rows 1 up to 6 we can apply the shift through lemma $G_1^H V_0 = \tilde{V}_1 \tilde{G}_1^H$:

$$H_1 = \qquad \left(\qquad + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

The Givens transformation $\tilde{G}_1^H$ can be brought between the brackets and will not affect the second term. It does act on the first term, but can be removed by a fusion:

$$H_1 = \tilde{V}_1 \tilde{G}_1^H \left(W_0 X_0 I_\alpha G_1 + \mathbf{u} \mathbf{v}_1^H \right)$$

$$= \left(+ \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right)$$

$$= \tilde{V}_1 \left(\tilde{G}_1^H W_0 X_0 I_\alpha G_1 + \mathbf{u} \mathbf{v}_1^H \right)$$

$$= \left(+ \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right) .$$

$$= \tilde{V}_1 \left(\hat{W}_1 X_0 I_\alpha G_1 + \mathbf{u} \mathbf{v}_1^H \right) .$$

The only remaining undesired Givens transformation is G_1 in the middle of the formula. The idea is now to drag this Givens transformation completely through the other matrices such that it appears before the matrix V_0 and moreover acts on row 2 and row 3. It has to act on row 2 and row 3, so that the next Givens transformation G_2 can be chosen to annihilate this transformation and such that it does not interfere with row 1 anymore.

Let us continue, by trying to remove also the second undesired Givens transformation denoted with $\times$. One can fairly easily apply two times the shift through operation to obtain the following scheme. In fact one applies a shift through operation of length 2 to the left ($\hat{W}_1 X_0 I_\alpha G_1 = \hat{G}_1 W_1 X_1 I_\alpha$).

Now there is an unwanted transformation in position 10. Denote this transformation with $\hat{G}_1$. In formulas we obtain now $H_1 = \tilde{V}_1 \left(\hat{G}_1 W_1 X_1 I_\alpha + \mathbf{u} \mathbf{v}_1^H \right)$.

This gives us the following relations:

$$H_1 = \tilde{V}_1 \hat{G}_1 \left(W_1 X_1 I_\alpha + \hat{G}_1^H \mathbf{u} \mathbf{v}_1^H \right) ,$$
$$= \tilde{V}_1 \hat{G}_1 \left(W_1 X_1 I_\alpha + \mathbf{u} \mathbf{v}_1^H \right) ,$$

$$= \left(+ \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right)$$

Two terms are of importance here: $\hat{G}_1^H \mathbf{u}$ and $\tilde{V}_1 \hat{G}_1$. Clearly the vector $\mathbf{u}$, will not change due to the Givens transformation $\hat{G}_1^H$. On $\tilde{V}_1 \hat{G}_1$ the shift through lemma can be applied, as already indicated

in the above scheme. The final step results in:

$$H_1 = G_2 V_1 \left(W_1 X_1 I_\alpha + \mathbf{u}\mathbf{v}_1^H \right),$$

$$= \quad \cdots \quad \left(\cdots + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right)$$

$$= G_2 V_1 \left(W_1 X_1 I_\alpha + \mathbf{u}\mathbf{v}_1^H \right).$$

The equation depicted here is almost of the desired structure, the matrix H_1 is almost in Hessenberg form, except for a bulge in position $(2,1)$. The representation is also almost in the correct form, only the Givens transformation G_2 is undesired. We will remove this transformation (removing thereby also the bulge in H_1), by applying a similarity transformation with G_2.

4.3 The chasing

We have now performed the initial step and we will continue now the algorithm, in an implicit way. We will only depict how to remove the Givens transformation G_2, the procedure can be continued in a similar fashion.

Since we want our matrix H to become again of unitary plus low rank form obeying the designed representation, we want to remove the disturbance G_2. Performing the unitary similarity transformation with this Givens transformation removes the transformation on the left, but creates an extra transformation on the right. We obtain the following:

$$\begin{aligned} H_2 &= G_2^H H_1 G_2 \\ &= G_2^H G_2 V_1 \left(W_1 X_1 I_\alpha + \mathbf{u}\mathbf{v}_1^H \right) G_2 \\ &= V_1 \left(W_1 X_1 I_\alpha G_2 + \mathbf{u}\mathbf{v}_1^H G_2 \right) = V_1 \left(W_1 X_1 I_\alpha G_2 + \mathbf{u}\mathbf{v}_2^H \right), \end{aligned}$$

where $\mathbf{v}_2^H = \mathbf{v}_1^H G_2$.

Similarly as in the initial step we can drag G_2 through W_1 and X_1, by a shift through operation of length 2. We obtain $W_1 X_1 G_2 = \tilde{G}_2 W_2 X_2$. The reader can create the schemes to see on which rows the transformation is performed. This gives us

$$\begin{aligned} H_2 &= V_1 \left(\tilde{G}_2 W_2 X_2 I_\alpha + \mathbf{u}\mathbf{v}_2^H \right), \\ &= V_1 \tilde{G}_2 \left(W_2 X_2 I_\alpha + \tilde{G}_2^H \mathbf{u}\mathbf{v}_2^H \right). \end{aligned}$$

Since the Givens transformation $\tilde{G}_2^H$ acts on rows 4 and 5, $\tilde{G}_2^H \mathbf{u} = \mathbf{u}$ ($\mathbf{u}$ has only the first element different from zero). Applying a final shift through operation for $V_1 \tilde{G}_2$ we obtain $G_3 V_2 = V_1 \tilde{G}_2$ giving us $H_2 = G_3 V_2 \left(W_2 X_2 I_\alpha + \mathbf{u}\mathbf{v}_2^H \right)$. We have performed now a step of the chasing method since the Givens transformation G_3 has shifted down one position w.r.t. the Givens transformation G_2.

The chasing procedure can be continued in a straightforward manner to go from G_i to G_{i+1}. Unfortunately this approach does not allow us to determine the last Givens transformation G_{n-1}. Let us show what goes wrong and how to solve this problem in the next subsection.

4.4 The last Givens transformation

Suppose we have performed step $n-3$ and we have the following situation:

$$H_{n-3} = G_{n-2} V_{n-3} \left(W_{n-3} X_{n-3} I_\alpha + \mathbf{u}\mathbf{v}_{n-3}^H \right).$$

Performing the similarity transformation determined by G_{n-2} results in

$$\begin{aligned} H_{n-2} &= V_{n-3} \left(W_{n-3} X_{n-3} I_\alpha G_{n-2} + \mathbf{u}\mathbf{v}_{n-3}^H G_{n-2} \right) \\ &= V_{n-3} \left(W_{n-3} X_{n-3} I_\alpha G_{n-2} + \mathbf{u}\mathbf{v}_{n-2}^H \right). \end{aligned}$$

The Givens transformation G_{n-2} works on rows $n-2$ and $n-1$, hence we obtain the left graphical schemes for the first term of $W_{n-3}X_{n-3}I_\alpha G_{n-2}^H$. Applying once the shift through operation moves the Givens transformation in position 0 to position 6, depicted on the right.

$$\text{(12)}$$

Unfortunately we cannot shift the Givens transformation through anymore, a single fusion can be applied and the undesired transformation vanishes. We obtain the following equation $H_{n-2} = V_{n-2}\left(W_{n-2}X_{n-2}I_\alpha + \mathbf{u}\mathbf{v}_{n-2}^H\right)$.

The representation is of the desired form, but we know that the matrix H_{n-2} is not yet of the correct form, this matrix is of Hessenberg form with an extra bulge in position $(n, n-1)$. The final Givens transformation can hence not be determined implicitly anymore. We determine this Givens transformation explicitly by computing the matrix vector product $H_{n-2}\mathbf{e}_{n-2}$. Compute now G_{n-1} such that $\mathbf{e}_n^H G_{n-1}^H H_{n-2}\mathbf{e}_{n-2} = 0$.

The final similarity transformation results in $H_{n-1} = G_{n-1}^H V_{n-2}\left(W_{n-2}X_{n-2}I_\alpha G_{n-1} + \mathbf{u}\mathbf{v}_{n-2}^H G_{n-1}\right)$. Applying a fusion for the product $G_{n-1}^H V_{n-2}$ and a scaled fusion for the product $X_{n-2}I_\alpha G_{n-1}$ results in $H_{n-1} = V_{n-1}\left(W_{n-1}X_{n-1}I_\alpha + \mathbf{u}\mathbf{v}_{n-1}^H\right)$, which is a new Hessenberg matrix, i.e., a sum of a unitary and a rank 1 matrix, represented using the desired representation. This completes the single shifted *QR*-step.

5 The double shift strategy

For our purpose the double shift strategy is enough. Nevertheless the multishift setting can be implemented as an almost trivial extension of the results presented here. First the single shift strategy was proposed since the double shift strategy is slightly more complicated, but makes extensive use of the results from the single shift strategy.

Note 4 In order not to overload the notation in the following subsections, we omit the factor α, which is normally present in the bottom right corner. Its presence is not so influential, hence it is not so troublesome to omit it in the description.

5.1 Initial step

Suppose the initial Hessenberg matrix H, with its representation is given. Two shifts μ_1 and μ_2 are given. Determine the Givens transformations G_1 and G_2 such that:

$$G_2^H G_1^H (H - \mu_1 I)(H - \mu_2 I)\mathbf{e}_1 = \beta\mathbf{e}_1,$$

with β a constant.

Let us perform now the initial similarity transformation $G_2^H G_1^H H G_1 G_2$, exploiting the representation of the unitary plus low rank matrix.

Again we consider the matrix H of the following form:

$$H = H_0 = V_0\left(W_0 X_0 I_\alpha + \mathbf{u}\mathbf{v}_0^H\right). \tag{13}$$

Let us compute $H_1 = G_2^H G_1^H H_0 G_1 G_2$. Schematically we obtain (the unwanted transformations are marked again with a $\times$):

$$H_1 = \qquad\left(\qquad + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}\mathbf{v}_0^H\right).$$

In the next scheme two things have happened, first we will bring the disturbing Givens transformations G_1 and G_2 on the right inside the brackets. The vector $\mathbf{v}_0^H$ will be transformed into $\mathbf{v}_1^H$. Secondly we will apply the shift through lemma on the Givens transformations depicted at the left:

$$H_1 = \left(\; + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

The shift through is already depicted, performing it, one obtains the following scheme:

$$H_1 = \left(\; + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

Bringing two Givens transformations inside the brackets, and performing the indicated shift through with the first term of the second factor gives us the following scheme. Remark that the vector $\mathbf{u}$ is not affected by this operation:

$$H_1 = \left(\; + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

After the shift through operation one applies a fusion such that one Givens transformation disappears. We obtain the following:

$$H_1 = \left(\; + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

$$= \tilde{V}_1 \left(\tilde{G}_1 \tilde{W}_1 X_1 I_\alpha G_1 G_2 + \mathbf{u}\mathbf{v}_1^H \right)$$

There are three more undesired Givens transformations G_1 and G_2 at the right and $\tilde{G}_1$ at the left.

Applying twice a shift through operation of length 2 to the left moves the undesired Givens transformations, G_1 and G_2, from right to left. Secondly one can take the three Givens transformations out of the brackets. We don not create fill in in the rank 1 part:

$$H_1 = \left(\; + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

$$= \left(\; + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

Performing the indicated shift through we obtain the following scheme:

$$H_1 = \qquad \left(\qquad\qquad + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

Rewriting this formula for the last time by another shift through operation gives us the desired form:

$$H_1 = \qquad \left(\qquad\qquad + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right). \tag{14}$$

Now we have finished the first step of the double shift algorithm. Clearly there are three undesired Givens transformations. The first two ones are used for the next chasing step. One Givens, the third one, will remain in that position, and is used in all consecutive steps.

Note 5 One might be surprised by the fact that the matrix is now disturbed by three Givens transformations, from which we will only take two for determining the next chasing step. This is completely logical when comparing this situation with the double shift algorithm for Hessenberg matrices. The double shift algorithm also has bulges of three undesired elements, from which only two will be annihilated. The remaining third element of the bulge will shift down every iteration step, just like the Givens transformation will move down here.

5.2 Chasing

The chasing proceeds similarly as to the single shift case. We will simply depict a single step. We start with Equation (14). The chasing, since this is a double shift method, consists now of performing two Givens at the same time. The next two Givens determining the similarity transformation are chosen as the inverses of the two leftmost Givens transformations. Applying this similarity transformation results in the matrix H_2, which will be of the following form:

$$H_2 = \qquad \left(\qquad\qquad + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_1^H \right).$$

With respect to H_1, the first two Givens transformations are removed, and two other ones appear at the right of the matrix. To continue, one applies the last two Givens transformations on both terms of the summation. This transforms $\mathbf{v}_1^H$ into $\mathbf{v}_2^H$. Applying twice a shift through operation of length 2 to the left moves the undesired Givens transformations on the right through the first term in the summation. This leads to the following graphical representation:

$$H_2 = \qquad \left(\qquad\qquad + \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{v}_2^H \right).$$

Bringing now the Givens transformations inside the brackets, marked with $\times$ outside of the brackets does not affect the column vector since these Givens transformations act on the zeros. Outside the brackets we can again bring these transformations to the front, perform a final shift through operation on the three leftmost Givens and complete this step.

We are now ready to perform the next similarity transformation based on the first two Givens transformations:

$$
H_2 = \qquad\qquad
$$

5.3 Last step

Similarly as in the single shift case, we are not able to continue using the implicit approach until the end. The last two steps cannot be completed in an implicit manner. Again one has to determine these columns of the Hessenberg matrix with its bulge explicitly to determine these final Givens transformations.

6 Deflation

The goal of repeatedly applying a QR-step is obtaining convergence to a quasi upper triangular matrix, which has the eigenvalues of the original matrix on its diagonal.

In the standard Hessenberg case deflation is of the following form. Consider the Hessenberg matrix H, having a zero subdiagonal element

$$
H = \left[\begin{array}{c|c} H_1 & \times \\ \hline 0 & H_2 \end{array} \right], \tag{15}
$$

then the matrix is subdivided into two other Hessenberg matrices H_1 and H_2 on which one continues working separately.

Suppose, however, that we are working with a Hessenberg matrix, written as the sum of a unitary and a rank 1 matrix. In case of deflation, it is not true in general that both matrices H_1 and H_2 are also of unitary plus rank 1 form.

The idea exploited in this manuscript differs significantly from the standard subdivision scheme as mentioned above. Since the original matrix H is written as the sum of a unitary plus low rank matrix, we will not subdivide the matrix as a combination of smaller unitary plus low rank matrices. Instead we will simply continue working on the global representation.

6.1 Structure of the representation

The occurrence of a zero on the subdiagonal must have its impact on the representation. This will be used for detecting subdiagonal zeros during the algorithm. Let us reconsider the Hessenberg matrix and its representation in case of a zero subdiagonal element. Suppose the matrix H satisfies Equation (15), assume the matrix to be of size 7×7 and $H(4,3) = 0$ (cases $H(3,2), H(5,4), H(6,5)$ can be dealt with similarly). The special cases $H(2,1)$ and $H(n, n-1) = H(7,6)$ are considered afterwards.

We have the standard representation $H = V \left(W X I_\alpha + \mathbf{u}\mathbf{v}^H \right)$. Consider now $V^H H = W X I_\alpha + \mathbf{u}\mathbf{v}^H$. Due to the zero in position $H(4,3)$ the matrix $V^H H$, which is a generalized Hessenberg matrix having two subdiagonals, has a zero in position $(5,3)$. Since the rank one matrix $\mathbf{u}\mathbf{v}^H$ has only the top row different from zero, also the matrix WX needs to have a zero in position $(5,3)$. The zero in position $(5,3)$ corresponds to a diagonal Givens transformation present in the descending sequence of W, this Givens transformation will act on row 4 and row 5.

This results in the following representation for the unitary matrix U in $H = U + \mathbf{u}\mathbf{v}^H$. The dashed line coincides with the block division in the original Hessenberg matrix. In this representation, the diagonal Givens transformation is indicated by the parameters δ_i and $\bar{\delta}_i$ (in fact $\delta_i = c_i$

and $\bar{\delta}_i = \bar{c}_i$, but we use this notation to indicate that the Givens transformation is almost diagonal, i.e., $|c_i| \approx 1$). They simply indicate row scalings, just like α does.

$$\tag{16}$$
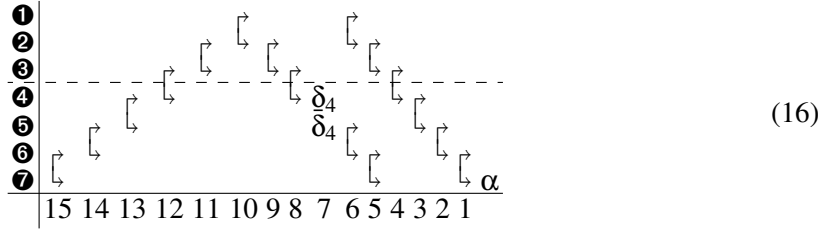
The zero subdiagonal element in the Hessenberg matrix appears as a diagonal Givens transformation, in position 7 on the graphical scheme. Note that the diagonal Givens is positioned lower than the actual dashed line indicating the cutting. This diagonal Givens transformation is also the criterion that will be used for deciding whether a block can be deflated or not. Unfortunately we cannot monitor convergence in position $H(n, n-1)$ in this fashion, we will come back to this in Subsection 6.5.

Instead of subdividing the problem and trying to design two separate representations, one for the upper left and one for the lower right part, we will simply continue working with this representation. We know that the line indicates the splitting in the Hessenberg matrix, hence we want to work on the part above and below the line separately. This will be depicted in forthcoming subsections.

Continuing performing steps of the *QR*-algorithm on this representation, will result in a quasi upper triangular matrix R, which is represented as the sum of a unitary and a rank one matrix. This quasi upper triangular matrix R has the eigenvalues of the original matrix on its block diagonal.

The Givens pattern of the final unitary matrix (omitting the row scaling factors) has the following form. We see that during the reducing procedure we will transform the Givens transformations from the inner sequence into unitary diagonal matrices.
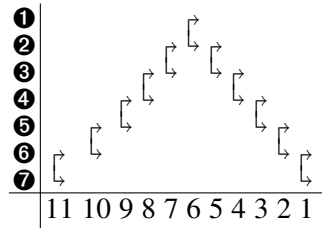
*Let us illustrate how to continue working on both parts of the Hessenberg matrix. The part above and the part below the dashed line. The shift and the initial Givens transformation can be computed by explicitly computing some columns of the Hessenberg matrix. This will cost at most $O(n)$ operations.

6.2 Upper part

Let us see how the algorithm changes in case of the following scheme (only the unitary factor is represented). The original Hessenberg matrix has a zero in position $H(6,5)$ and we want to perform a *QR*-step on the top part, i.e., the first 5 rows. Since the standard *QR* procedure applied on a reducible Hessenberg matrix breaks down when reaching the zero subdiagonal element, we would also expect a certain break during the chasing procedure in this case.
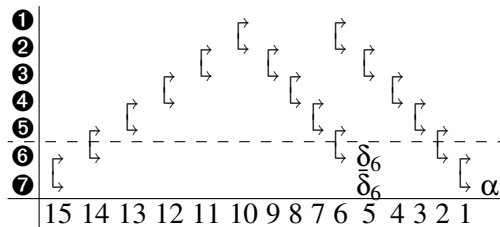
The first initial disturbing Givens transformation is applied in an identical way as before the deflation. Nothing new here. The next Givens similarity transformation acting on rows and columns 2 and 3 can be performed as in the standard case, just like the transformation on rows and columns 3 and 4. The next similarity transformation acting on row and columns 4 and 5 is a special one. In the standard case, this Givens transformation does not create another bulge. Hence, the matrix is again of Hessenberg form and the *QR* procedure breaks down. One cannot compute the following Givens transformation acting on row 5 and 6 anymore.

Let us see how this translates to our case. Suppose we have the following situation, just before applying the Givens similarity transformation. Since only the unitary matrix is essentially involved in this chasing procedure we will not depict any operations on the rank one structure.
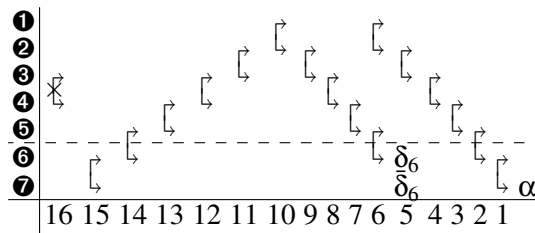
Performing the Givens similarity transformation will remove the undesired Givens transformation in position 16 and create another one on the right. In the left scheme this situation is depicted. A shift through operation is depicted and the result is shown in the right scheme.
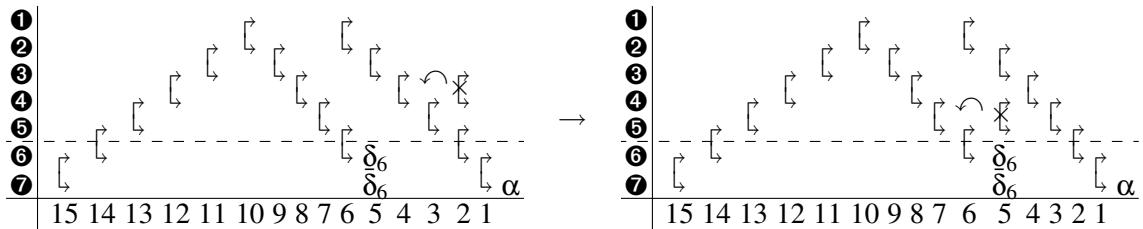
In the right scheme we see that we can apply another shift through operation, due to the fact that the Givens transformation in position 6 crosses the dashed line. In Section 4.4 this Givens transformation fused with another one and vanished. Hence, in this case, we do not need to determine the last Givens transformation by explicitly computing the fourth column. The final step is therefore essentially different from the final step in Section 4.4.

So when dragging the remaining undesired Givens transformation completely to the left we obtain the following scheme.
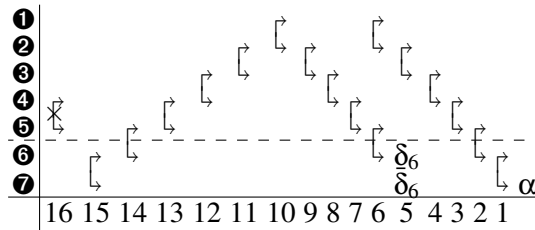
When performing the next transformation, the algorithm should stop since the element $H(6,5)$ in the Hessenberg matrix is zero. This means that at the end of this similarity transformation, all undesired Givens transformations should be removed. Let us see how this translates towards the representation. After the similarity transformation we obtain the left scheme. A scaled fusion is

depicted in the right scheme.
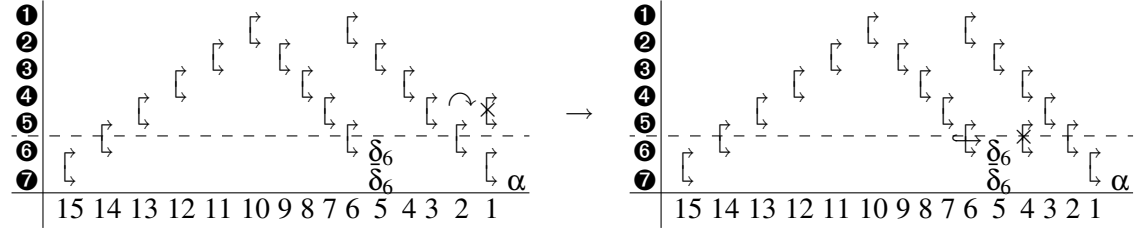
As a result we obtain again a similar represented Hessenberg matrix. This process can easily be repeated until another Givens transformation in the second sequence is close enough to diagonal form.

6.3 Bottom part

Applying a *QR*-step on the bottom part only differs from Section 4 in the way the first Givens transformation is performed. The final Givens transformation again has to be determined by computing the penultimate column.

Assume we have the situation as depicted in Scheme (16). Since we would like to perform a *QR*-step on the bottom part, the initial similarity transformation will act immediately on rows and columns 4 and 5. Suppose the transformation is determined and the similarity transformation is performed. This results in the following scheme (the Givens transformations are applied directly below the dashed line).
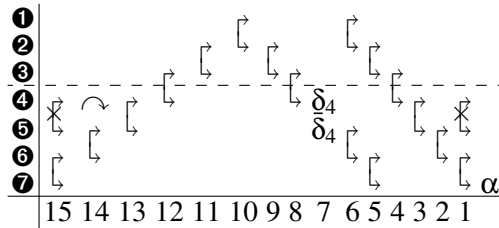
The Givens transformation on the left marked with $\times$ is shifted through such that it appears in the middle. The next scheme clearly shows that only one scaled fusion needs to be applied to remove the undesired Givens transformation from the left.
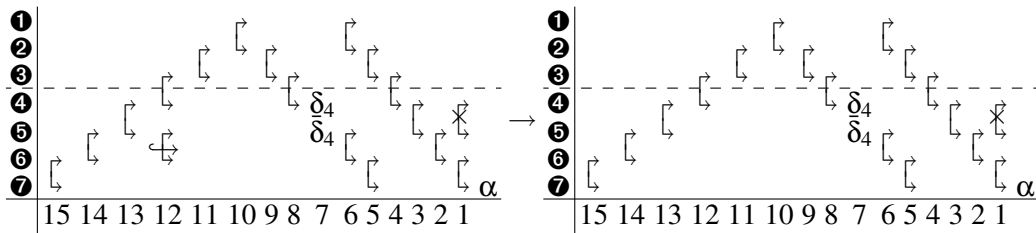
As a result we obtain sort of the standard situation. The Givens transformation on the right is not troublesome, dragging it through completely to the left as done before, it will come out acting on rows 5 and 6. Hence it can be used for determining the next Givens transformation.

6.4 More general cases

Of course when running the algorithm cases like the following will occur:

$$H = \left[\begin{array}{c|c|c} H_1 & \times & \times \\ \hline 0 & H_2 & \times \\ \hline 0 & 0 & H_3 \end{array} \right]. \tag{17}$$

To apply a *QR*-step on the matrix H_2, a combination of techniques from the previous two sections needs to be made. Performing the initial step is described in Subsection 6.3, whereas performing the last Givens is done like in Subsection 6.2. *QR*-steps performed on these middle blocks are cheaper than the ones on H_1 and H_3.

6.5 Special case

Two cases were not treated before, namely the possibilities of $H(n, n-1)$ and $H(2,1)$ being zero. For $H(n, n-1)$, the appearance of a zero element in this position is not reflected in the representation. Hence one has to compute this element explicitly and monitor its behavior. It can be computed by computing the vector $H\mathbf{e}_{n-1}$. In case $H(n, n-1)$ equals zero, nothing special occurs, one can apply similar techniques as those described in Section 6.2.

7 Numerical experiments

7.1 Balancing strategy

Because of the possible variation in magnitude of coefficients of the polynomial, normally a pre-processing step is applied to the matrix as, e.g., in [9]. In fact this is a balancing strategy, meaning that the variation in magnitude of the elements of the matrix is reduced. Generally DCD^{-1} is computed with D a diagonal matrix. To obtain again a unitary plus low rank matrix, the diagonal elements have to be chosen equal to $d_i = \beta^i$ for some β. This results in

$$DCD^{-1} = \beta \begin{bmatrix} 0 & 0 & \ldots & 0 & -\frac{p_0}{\beta^n} \\ 1 & 0 & \ldots & 0 & -\frac{p_1}{\beta^{n-1}} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ddots & \ddots & 0 & -\frac{p_{n-2}}{\beta^2} \\ 0 & 0 & \ldots & 1 & -\frac{p_{n-1}}{\beta} \end{bmatrix} = \beta\hat{C}.$$

For an appropriate β, the eigenvalues of matrix C are then obtained by computing the eigenvalues of matrix $\hat{C}$ and then multiplying them by the factor β. Empirically, they found the following criterion for choosing β to be useful. Choose β such that

$$\text{Range}\{|\hat{c}_1|, |\hat{c}_2|, \ldots, |\hat{c}_n|, 1\} := \frac{\max\{|\hat{c}_1|, |\hat{c}_2|, \ldots, |\hat{c}_n|, 1\}}{\min\{|\hat{c}_1|, \ldots, |\hat{c}_n|, 1\}}$$

is as small as possible, where $\hat{c}_i = p_{n-i}/\beta^i$.

7.2 The implementation

The algorithm using a double shift as explained in Section 5 was implemented in Matlab[3] Version 7.7 (R2008b). To avoid the fact that the Givens transformations for the last two columns have to be computed explicitly, we do one iteration step with the two shifts equal to zero, applied to the polynomial $z^2 p(z)$ which has clearly two additional zeros for $z = 0$. This iteration step brings these two zeros as eigenvalues of the bottom 2×2 block of the (extended) companion matrix. So, after this iteration step, one can deflate these two artificial roots and no explicit computation of the Givens transformations based on the last two columns has to be computed again. We will refer to our implementation as VanBarel. This software can be requested from the authors. We wanted to compare this implementation with all other methods for which we were able to obtain an implementation. We had access to the following implementations: Delvaux corresponds to a Matlab-implementation of the method proposed in [10], Bini denotes the Matlab-implementation for the method from [6], Chandrasekaran denotes a Fortran-implementation for the method proposed in [8] and eig no bal and eig bal correspond to the eig function of Matlab without and with balancing, respectively (with balancing is meant an extra balancing inside the function eig).

---

[3] Matlab is a registered trademark of The MathWorks, Inc.

7.3 The experiments

Two different errors will be considered in the experiments. The maximum eigenvalue error (Max Eig) is defined as the distance between the sets $\lambda(C)$ and $\tilde{\lambda}(C)$:

$$\operatorname{dist}(\lambda(C),\tilde{\lambda}(C)) = \max \left\{ \max_{\tilde{\lambda}\in\tilde{\lambda}(C)} ||\tilde{\lambda}-\lambda(C)||, \max_{\lambda\in\lambda(C)} ||\lambda-\tilde{\lambda}(C)|| \right\},$$

where $\tilde{\lambda}(C)$ are the eigenvalues computed with the algorithms and where $\lambda(C)$ are the exact eigenvalues, when they are known; otherwise we assume that the eigenvalues computed by the Matlab function **eig** are exact. The norm $||\lambda-\tilde{\lambda}(C)||$ is defined as $\min_{\tilde{\lambda}\in\tilde{\lambda}(C)}|\lambda-\tilde{\lambda}|$ and $\min_{\tilde{\lambda}\in\tilde{\lambda}(C)}(|\lambda-\tilde{\lambda}|/|\lambda|)$ for the absolute respectively relative maximum eigenvalue error (Max Eig Abs/ Max Eig Rel). The second errors are the maximal componentwise absolute and relative error in the coefficients of the polynomials (Max Coeff Abs/Max Coeff Rel):

$$\max_i |p_i-\tilde{p}_i|, \qquad \max_i \left(\frac{|p_i-\tilde{p}_i|}{|p_i|}\right),$$

where $\tilde{p}_i, p_i$ are the coefficients of the polynomial obtained by the computed and exact eigenvalues, respectively. The coefficients $\tilde{p}_i$ are computed in multiple precision.

Experiment 1: In this experiment, we investigate the robustness of the different methods with respect to the choice of the scaling factor β described in Subsection 7.1.

To do so, we consider the following monic polynomials each of degree 20 where almost all of these polynomials have a large variation in the magnitude of their coefficients [13,9]:

1. Wilkinson polynomial with roots: k with $k=1,\ldots,20$.
2. Monic polynomial with roots: $[-2.1:0.2:1.7]$.
3. Monic polynomial with roots: 2^k with $k=-10,\ldots,9$.
4. Reversed Wilkinson polynomial with roots: $\frac{1}{k}$ with $k=1,\ldots,20$.
5. Polynomial $p(z) = (20!)\sum_{k=0}^{20}\frac{z^k}{k!}$.
6. Polynomial $p(z) = z^{20}+z^{19}+\cdots+z+1$.

For the polynomials $1.-4.$ the roots are known, for $5)-6)$ this is not the case. Because of the wide range of the coefficients of the polynomial, e.g. for the Wilkinson polynomial $|p_i|\in[1,10^{20}]$, balancing, which is discussed in Subsection 7.1, is very important. Therefore different balancing factors β are applied to the matrix C. Figure 1 contains the result of varying the balancing factor $\beta=2^j, j=-10,\ldots,10$ for the six polynomials of degree 20. The result of the maximal componentwise relative error on the coefficients is shown on a logarithmic scale for the six different methods: VanBarel, Delvaux, Bini, Chandrasekaran, eig no bal and eig bal. In fact, a relative error bigger than one does not give much information. However, we included them in the figures to show compare the behaviour of the errors.

The figure shows that the methods Delvaux, Bini, and Chandrasekaran give a (rather) accurate solution, for some specific scaling factors β. For the method VanBarel the choice of the scaling parameter β is not so crucial to obtain an accurate solution. As long as the scaling factor is not too large, the method works. This shows that this method is very robust with respect to the other $O(n^2)$ methods that we are aware of. The function eig bal of Matlab always works well. Note that this is an $O(n^3)$ method as well as eig with no balancing. Figure 2 shows the condition number of the eigenvectors for the different scaling parameters β, giving an indication of the ill-conditioning of the problems.

In Table 1-6, the maximal eigenvalue error (Max Eig Abs/Rel) and the maximal componentwise error on the coefficients (Max Coeff Abs/Rel) for the optimal choice of the parameter β are shown for the methods Bini, Delvaux, Chandrasekaran, VanBarel, Eig no bal and Eig. These tables show that the method Chandrasekaran gives accurate results followed by the method

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$1.6374e-02$	$7.2081e-02$	$4.4348e-02$	$2.4255e-03$	$6.8288e-02$	$6.6512e-01$
Max Eig Rel	$1.0328e-03$	$5.0278e-03$	$2.9636e-03$	$1.8658e-04$	$4.9016e-03$	$4.8500e-02$
Max Coeff Abs	$4.3777e+07$	$2.6171e+07$	$4.5311e+08$	$3.9322e+05$	$5.5520e+08$	$2.1709e+05$
Max Coeff Rel	$1.0588e-11$	$2.9900e-12$	$1.8624e-10$	$7.1132e-14$	$2.2821e-10$	$1.5727e-14$

Table 1. Experiment 1: Polynomial 1 with balancing factor $\beta = 2^3$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$3.0608e-10$	$3.2560e-10$	$4.8209e-09$	$1.0378e-12$	$3.3662e-10$	$3.3314e-10$
Max Eig Rel	$2.0405e-10$	$2.1706e-10$	$2.8358e-09$	$6.1049e-13$	$2.2441e-10$	$2.2209e-10$
Max Coeff Abs	$1.0118e-11$	$6.6237e-12$	$8.5017e-11$	$1.5859e-11$	$3.9790e-12$	$3.0127e-12$
Max Coeff Rel	$2.4399e-11$	$3.4542e-11$	$4.2490e-10$	$1.7465e-12$	$3.0063e-11$	$9.0659e-13$

Table 2. Experiment 1: Polynomial 2 with balancing factor $\beta = 1$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$1.0183e-07$	$4.3739e-04$	$3.8646e+14$	$8.7539e-11$	$3.4902e-07$	$5.7270e-12$
Max Eig Rel	$5.5336e-05$	$3.9111e-06$	$7.5480e+11$	$3.8769e-13$	$1.7927e-04$	$1.0374e-12$
Max Coeff Abs	$8.6250e+00$	$3.0235e+07$	$3.8646e+14$	$1.8531e+01$	$3.9062e+00$	$9.1719e+00$
Max Coeff Rel	$1.6926e-05$	$3.4559e-07$	$3.7740e+11$	$4.2721e-13$	$5.1920e-05$	$9.2531e-14$

Table 3. Experiment 1: Polynomial 3 with balancing factor $\beta = 2^{-2}$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$2.6257e-03$	$2.6713e-03$	$5.6555e-03$	$9.1241e-05$	$6.7330e-03$	$9.5857e-04$
Max Eig Rel	$4.3290e-02$	$4.2742e-02$	$8.2913e-02$	$1.2360e-03$	$9.6674e-02$	$1.2015e-02$
Max Coeff Abs	$2.7622e-13$	$4.5785e-12$	$3.5232e-12$	$1.7764e-15$	$5.3291e-15$	$6.2172e-15$
Max Coeff Rel	$1.2944e-11$	$1.0349e-11$	$1.2226e-10$	$7.9069e-14$	$3.6591e-11$	$1.2709e-14$

Table 4. Experiment 1: Polynomial 4 with balancing factor $\beta = 2^{-3}$.

VanBarel and Delvaux when scaling is required. The selected balancing parameter correspond to the minimum of the cond(V) as shown in Figure 2.

In Table 7-12, the maximal eigenvalue error (Max Eig Abs/Rel) and the maximal componentwise error on the coefficients (Max Coeff Abs/Rel) for parameter $\beta = 2^{-7}$ are shown for the methods Bini, Delvaux, Chandrasekaran, VanBarel, Eig no bal and Eig. These tables show that the method VanBarel gives accurate results in contrast to the results of Bini, Delvaux, Chandrasekaran for a small balancing factor, except for polynomial 6 .

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal
Max Eig Abs	$5.9114e-11$	$1.6175e-10$	$7.0761e-10$	$6.1573e-11$	$4.6811e-11$
Max Eig Rel	$9.1362e-12$	$2.4998e-11$	$1.0936e-10$	$9.5162e-12$	$7.2347e-12$
Max Coeff Abs	$2.3962e+05$	$2.6881e+05$	$2.3500e+07$	$1.7459e+05$	$1.0547e+05$
Max Coeff Rel	$9.8490e-14$	$1.1049e-13$	$9.6591e-12$	$7.1763e-14$	$4.3352e-14$

Table 5. Experiment 1: Polynomial 5 with balancing factor $\beta = 2^3$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal
Max Eig Abs	$3.3643e-15$	$2.2888e-15$	$3.6367e-15$	$1.6653e-15$	$2.4980e-15$
Max Eig Rel	$3.3643e-15$	$2.2888e-15$	$3.6367e-15$	$1.6653e-15$	$2.4980e-15$
Max Coeff Abs	$4.5963e-14$	$1.8886e-14$	$2.7124e-14$	$1.1324e-14$	$1.9984e-14$
Max Coeff Rel	$4.5963e-14$	$1.8886e-14$	$2.7124e-14$	$1.1324e-14$	$1.9984e-14$

Table 6. Experiment 1: Polynomial 6 with balancing factor $\beta = 1$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$4.0024e-02$	$5.6829e+15$	$1.0311e+13$	$1.1943e+01$	$5.1384e-03$	$1.1387e+00$
Max Eig Rel	$2.5078e-03$	$2.8415e+14$	$5.1557e+11$	$8.0482e-01$	$3.6703e-04$	$8.6411e-02$
Max Coeff Abs	$2.2323e+05$	$1.1975e+32$	$1.3804e+19$	$1.3679e+18$	$4.7923e+05$	$1.1674e+05$
Max Coeff Rel	$2.4903e-14$	$9.5275e+25$	$4.9102e+10$	$1.0981e-01$	$3.7881e-14$	$2.5039e-14$

Table 7. Experiment 1: Polynomial 1 with balancing factor $\beta = 2^{-7}$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$3.9778e-10$	$1.9012e+11$	$4.8435e+12$	$8.3272e-01$	$1.5159e-09$	$2.8514e-10$
Max Eig Rel	$2.6235e-10$	$9.0535e+10$	$2.3064e+12$	$3.9653e-01$	$1.1661e-09$	$1.9009e-10$
Max Coeff Abs	$3.9064e-09$	$7.2437e+17$	$4.8435e+12$	$3.1687e+02$	$3.9671e-09$	$5.0022e-12$
Max Coeff Rel	$1.9956e-10$	$1.2708e+17$	$1.2109e+12$	$5.2838e+00$	$4.3421e-10$	$1.1332e-13$

Table 8. Experiment 1: Polynomial 2 with balancing factor $\beta = 2^{-7}$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$6.3915e-09$	$3.9321e+13$	$4.1811e+12$	$4.1906e-02$	$6.0751e-13$	$5.7270e-12$
Max Eig Rel	$3.8842e-11$	$7.6800e+10$	$8.1662e+09$	$6.5521e-04$	$2.3204e-13$	$1.0374e-12$
Max Coeff Abs	$1.4423e+02$	$1.2160e+14$	$1.2160e+14$	$4.5075e+10$	$5.3594e+00$	$9.1719e+00$
Max Coeff Rel	$1.2391e-11$	$3.8400e+10$	$4.0831e+09$	$3.7750e-04$	$4.8907e-14$	$9.2531e-14$

Table 9. Experiment 1: Polynomial 3 with balancing factor $\beta = 2^{-7}$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal	Eig
Max Eig Abs	$1.3461e-04$	$1.6513e+00$	$3.1757e+12$	$3.6395e-01$	$2.1203e-05$	$3.7008e-03$
Max Eig Rel	$2.0192e-03$	$2.1695e+00$	$3.1757e+12$	$7.4831e-01$	$2.5450e-04$	$4.9816e-02$
Max Coeff Abs	$4.6274e-13$	$7.7114e+00$	$3.1757e+12$	$1.2757e-01$	$1.7764e-15$	$2.3981e-14$
Max Coeff Rel	$4.9316e-13$	$1.3591e+00$	$8.8269e+11$	$1.6896e-01$	$6.1913e-15$	$1.2475e-14$

Table 10. Experiment 1: Polynomial 4 with balancing factor $\beta = 2^{-7}$.

Experiment 2: To check that the computational complexity is of order $O(n^2)$, we constructed random polynomials of degree n, by computing the coefficients $p_\ell \in \mathbb{C}$, $\ell = 0, \ldots, n$ by using randomly distributed complex numbers. The matrix sizes were chosen as $n = 25 \cdot 2^{i-1}$, with $i = 1, \ldots, 6$. We assume that no balancing ($\beta = 1$) is necessary. For each of these sizes n, 5 samples were considered.

Figure 3(a) shows the ratio T_{i+1}/T_i averaged over the 5 samples for the method Delvaux, Bini, Chandrasekaran and VanBarel. As you can see this is of the order $O(n^2)$. Figure 3(b) shows the average number of *QR* iterations per eigenvalue. For the method Delvaux the average is between

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal
Max Eig Abs	$1.1566e-10$	$6.1650e+13$	$1.6160e+13$	$3.6562e+00$	$4.5040e-08$
Max Eig Rel	$1.7875e-11$	$4.3386e+12$	$1.1372e+12$	$4.9664e-01$	$6.9610e-09$
Max Coeff Abs	$4.2547e+06$	$2.4329e+18$	$2.4329e+18$	$1.4933e+18$	$3.3156e+08$
Max Coeff Rel	$1.7488e-12$	$3.9944e+12$	$8.0799e+11$	$6.1377e-01$	$1.7974e-09$

Table 11. Experiment 1: Polynomial 5 with balancing factor $\beta = 2^{-7}$.

	EigVanBarel	EigDelvaux	EigBini	EigChan	Eig bal
Max Eig Abs	$1.4138e+03$	$2.5229e+14$	$5.2627e+12$	$3.9242e+09$	$1.2930e-06$
Max Eig Rel	$1.4138e+03$	$2.5229e+14$	$5.2627e+12$	$3.9242e+09$	$1.2930e-06$
Max Coeff Abs	$1.2319e+12$	$2.5229e+14$	$5.2627e+12$	$9.0749e+49$	$1.4591e-05$
Max Coeff Rel	$1.2319e+12$	$2.5229e+14$	$5.2627e+12$	$9.0749e+49$	$1.4591e-05$

Table 12. Experiment 1: Polynomial 6 with balancing factor $\beta = 2^{-7}$.

(a) Polynomial 1

(b) Polynomial 2

(c) Polynomial 3

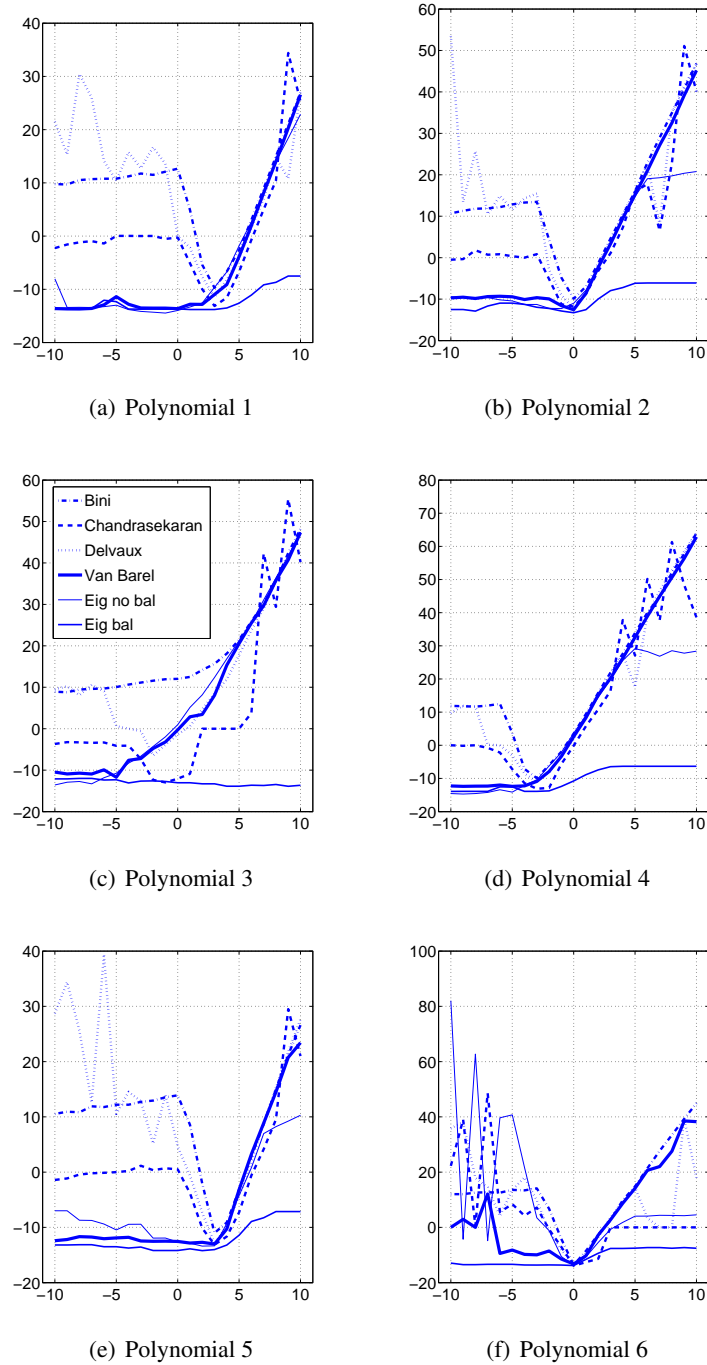(d) Polynomial 4

(e) Polynomial 5

(f) Polynomial 6

Figure 1. Experiment 1: Looking for the optimal balancing factor $\beta = 2^j$ for polynomials $1-6$.

2 and 3. For the method Bini it is a little bit more, between 5 and 6. For the method VanBarel, it is between 1 and 2 and for the method Chandrasekaran is it around three iterations per eigenvalue.

This number is strongly influenced by the policy used for stopping the iteration, the deflation criterion and the type of iteration (double or single shift). Let us briefly comment on these three items for all implementations in the following.

The method Delvaux uses a Wilkinson shift for a single-shift QR-iteration. Deflation is allowed anywhere in the matrix and occurs when the non-relative criterion is satisfied $|h_{i,i-1}| < \varepsilon$ ($h_{i,i-1}$ represents a subdiagonal element in the Hessenberg matrix). In practice they take ε as the machine precision. All subdiagonal elements are computed explicitly in order to check for deflation. Subblocks are considered sufficiently small when they are of size less then or equal to 8×8,

(a) Polynomial 1

(b) Polynomial 2

(c) Polynomial 3

(d) Polynomial 4

(e) Polynomial 5
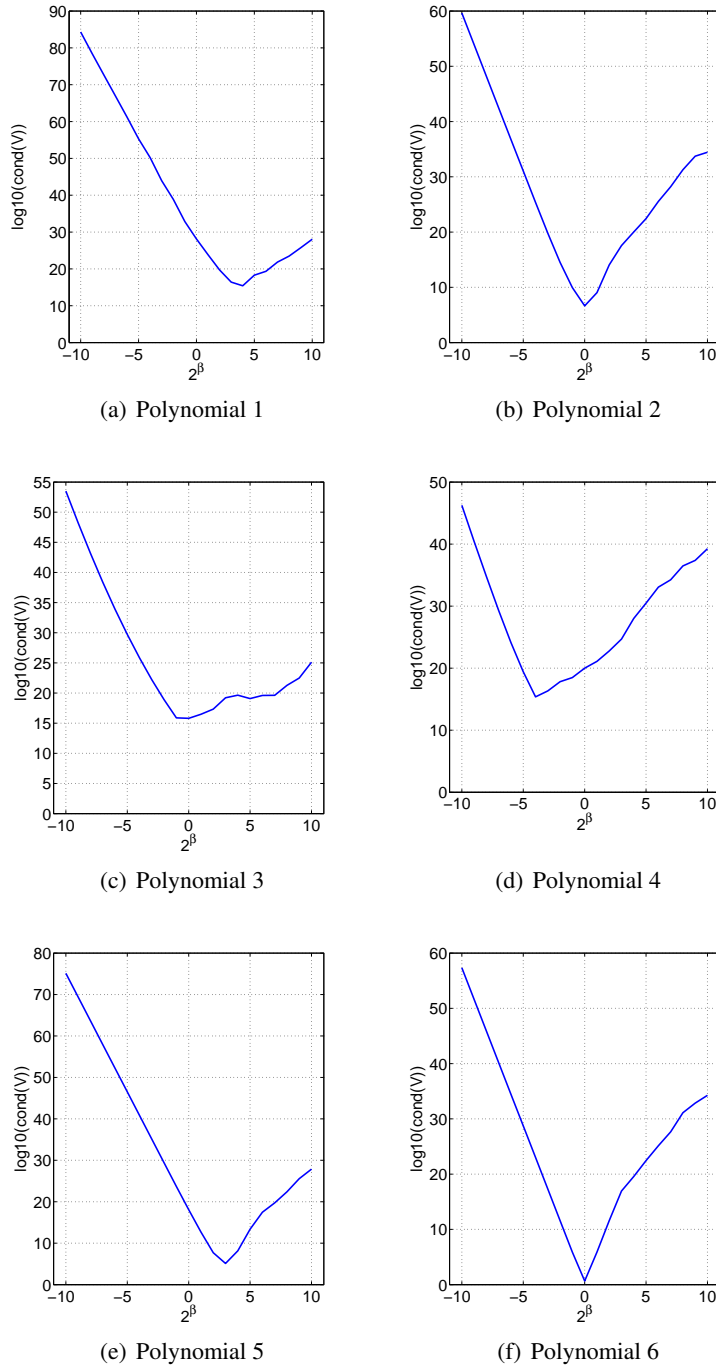
(f) Polynomial 6

Figure 2. Experiment 1: Condition of eigenvectors $cond(V)$ for different balancing factors $\beta = 2^j$ with $j = -10, \ldots, 10$ for polynomials $1 - 6$.

then their eigenvalues are computed by standard matlab routines eig for computing eigenvalues of Hessenberg matrices and not by the presented fast iteration anymore.

The method $\mathsf{Van\ Barel}$ uses a double shifted QR-iteration based on Francis shift strategy to retain real arithmetic. For single shift iterations the standard Rayleigh shift is taken. Deflation is allowed anywhere in the matrix and occurs as discussed in Section 6 when a Givens rotation becomes diagonal. Since Givens rotations are unitary a non-relative criterion based on the machine precision is used to see if the off-diagonal elements are small enough. Only the final subdiagonal element of the Hessenberg matrix is computed explicitly and a relative criterion

n	Max Eig Abs Err	Max Eig Rel Err	Max Coeff Abs Err	Max Coeff Rel Err
25	$1.27\ 10^{-14}$	$1.21\ 10^{-14}$	$5.88\ 10^{-14}$	$4.65\ 10^{-14}$
50	$2.88\ 10^{-14}$	$2.53\ 10^{-14}$	$2.48\ 10^{-13}$	$1.73\ 10^{-13}$
100	$6.70\ 10^{-14}$	$7.19\ 10^{-14}$	$1.01\ 10^{-12}$	$7.24\ 10^{-13}$
200	$2.67\ 10^{-13}$	$2.37\ 10^{-13}$	$5.50\ 10^{-12}$	$4.17\ 10^{-12}$
400	$5.58\ 10^{-12}$	$4.90\ 10^{-12}$	$3.26\ 10^{-11}$	$2.41\ 10^{-11}$
800	$7.24\ 10^{-11}$	$6.24\ 10^{-11}$	$1.57\ 10^{-10}$	$1.31\ 10^{-10}$

Table 13. Experiment 2: Results Bini.

n	Max Eig Abs Err	Max Eig Rel Err	Max Coeff Abs Err	Max Coeff Rel Err
25	$5.55\ 10^{-15}$	$5.14\ 10^{-15}$	$4.22\ 10^{-14}$	$3.08\ 10^{-14}$
50	$9.02\ 10^{-15}$	$8.23\ 10^{-15}$	$7.93\ 10^{-14}$	$6.33\ 10^{-14}$
100	$1.40\ 10^{-14}$	$1.28\ 10^{-14}$	$2.82\ 10^{-13}$	$2.45\ 10^{-13}$
200	$6.22\ 10^{-14}$	$5.39\ 10^{-14}$	$9.43\ 10^{-13}$	$7.57\ 10^{-13}$
400	$1.89\ 10^{-13}$	$1.80\ 10^{-13}$	$6.01\ 10^{-12}$	$5.17\ 10^{-12}$
800	$1.31\ 10^{-12}$	$1.22\ 10^{-12}$	$5.13\ 10^{-11}$	$4.71\ 10^{-11}$

Table 14. Experiment 2: Results Delvaux.

n	Max Eig Abs Err	Max Eig Rel Err	Max Coeff Abs Err	Max Coeff Rel Err
25	$2.78\ 10^{-15}$	$8.99\ 10^{-15}$	$1.79\ 10^{-14}$	$1.34\ 10^{-14}$
50	$9.23\ 10^{-15}$	$6.32\ 10^{-14}$	$7.11\ 10^{-14}$	$5.66\ 10^{-14}$
100	$1.19\ 10^{-14}$	$7.52\ 10^{-14}$	$3.31\ 10^{-13}$	$2.97\ 10^{-13}$
200	$2.15\ 10^{-14}$	$4.01\ 10^{-13}$	$2.09\ 10^{-12}$	$1.82\ 10^{-12}$
400	$3.871\ 10^{-13}$	$1.81\ 10^{-12}$	$1.04\ 10^{-11}$	$8.96\ 10^{-12}$
800	$1.98\ 10^{-11}$	$3.97\ 10^{-11}$	$1.73\ 10^{-10}$	$1.30\ 10^{-10}$

Table 15. Experiment 2: Results Chandrasekaran.

$|h_{n,n-1}| < \varepsilon(|h_{n-1,n-1}| + |h_{n,n}|)$ is used (ε depends on the machine precision). A block is considered small enough when it is of size 3×3.

In Chandrasekaran also double shifted QR-iterations with Francis shift strategy are performed and single shift iterations use the Raileigh shift. In this method the traditional deflation criterion $|h_{i,i-1}| < \varepsilon(|h_{i-1,i-1}| + |h_{i,i}|)$ is also used, but translated to the representation of the Hessenberg matrix. Hence the subdiagonal and diagonal elements are not computed explicitly to check for deflation, but a specific procedure acting on its Hierarchical representation is used. Submatrices are deflated when they are of size 3×3.

In the method of Bini a single shifted QR-iteration with Rayleigh shift is considered. Deflation is only checked at the trailing subdiagonal element, which is explicitly computed, and the relative criterion $|h_{n,n-1}| < \varepsilon(|h_{n-1,n-1}| + |h_{n,n}|)$ is also utilized here to check if deflation can be applied. Since deflation is only allowed in the lower right part of the matrix, all deflated blocks have size 1×1.

In Table 13-16, the maximal componentwise error on the coefficients (Max Coeff Abs/Rel) and the maximal eigenvalue error (Max Eig Abs/Rel) are shown for the methods, Bini, Delvaux, Chandrasekaran and VanBarel. For the errors it can be seen that if n increases the errors increase slightly. Looking at the errors the results obtained with method Delvaux are slightly better.

8 Conclusion

In this paper an algorithm to compute the eigenvalues of a monic polynomial is described. This is done by computing the eigenvalues of the corresponding companion matrix. We showed how the QR algorithm can be adjusted such that the algorithm exploits the unitary plus rank one structure of the matrices involved and how the matrices can be represented by $O(n)$ parameters using

n	Max Eig Abs Err	Max Eig Rel Err	Max Coeff Abs Err	Max Coeff Rel Err
25	$1.92\ 10^{-14}$	$1.65\ 10^{-14}$	$1.15\ 10^{-13}$	$8.59\ 10^{-14}$
50	$3.16\ 10^{-14}$	$2.82\ 10^{-14}$	$5.13\ 10^{-13}$	$4.04\ 10^{-13}$
100	$1.20\ 10^{-13}$	$1.19\ 10^{-13}$	$2.69\ 10^{-12}$	$2.21\ 10^{-12}$
200	$3.67\ 10^{-13}$	$3.36\ 10^{-13}$	$2.11\ 10^{-11}$	$1.92\ 10^{-11}$
400	$2.54\ 10^{-12}$	$2.02\ 10^{-12}$	$1.89\ 10^{-10}$	$1.11\ 10^{-10}$
800	$6.98\ 10^{-12}$	$5.74\ 10^{-12}$	$1.81\ 10^{-09}$	$1.74\ 10^{-09}$

Table 16. Experiment 2: Results Van Barel.
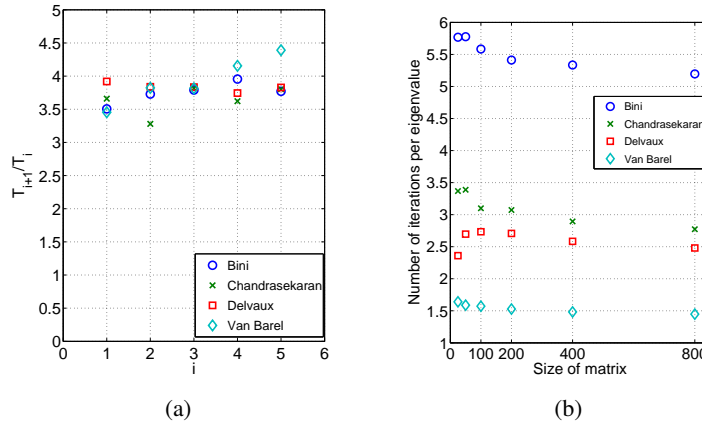
Figure 3. Experiment 2: a) time ratio, b) average number of iterations per eigenvalue.

sequences of Givens transformations. The numerical performance of the algorithm was demonstrated by means of numerical experiments showing the good accuracy and efficiency of the algorithm. Especially the robustness with respect to the scaling parameter β is as far as we know not observed in any other $O(n^2)$ method.

In the near future, we plan to make an implementation in C or C++ to study the efficiency of the method compared to standard eigenvalue solvers, e.g., from LAPACK. We also want to prove the numerical stability of our algorithm.

References

1. S. Barnett. *Polynomials and Linear Control Systems*. Monographs and Textbooks in Pure and Applied Mathematics. Marcel Dekker, Inc., New York, USA, 1983.
2. D. Bindel, S. Chandrasekaran, J. W. Demmel, D. Garmire, and M. Gu. A fast and stable nonsymmetric eigensolver for certain structured matrices. Technical report, Department of Computer Science, University of California, Berkeley, California, USA, May 2005.
3. D. Bindel, J. W. Demmel, W. Kahan, and O. A. Marques. On computing Givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software*, 28(2):206–238, June 2002.
4. D. A. Bini, P. Boito, Y. Eidelman, L. Gemignani, and I. C. Gohberg. A fast QR eigenvalue method for a class of structured matrices. *Linear Algebra and its Applications*, 2009. Accepted for publication, doi: 10.1016/j.laa.2009.08.003.
5. D. A. Bini, F. Daddi, and L. Gemignani. On the shifted *QR* iteration applied to companion matrices. *Electronic Transactions on Numerical Analysis*, 18:137–152, 2004.
6. D. A. Bini, Y. Eidelman, L. Gemignani, and I. C. Gohberg. Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(2):566–585, 2007.
7. D. A. Bini, L. Gemignani, and V. Y. Pan. Fast and stable *QR* eigenvalue algorithms for generalized companion matrices and secular equations. *Numerische Mathematik*, 100(3):373–408, 2005.
8. S. Chandrasekaran, M. Gu, J. Xia, and C.-B. Zhu. A fast eigensolver for companion matrices. *Operator Theory: Advances and Applications*, 179(111-143), 2007.
9. S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu. A fast QR algorithm for companion matrices. *Operator Theory: Advances and Applications*, 179:111–143, 2007.

10. S. Delvaux, K. Frederix, and M. Van Barel. An algorithm for computing the eigenvalues of block companion matrices. Technical Report TW538, Department of Computer Science, Katholieke Universiteit Leuven, Apr 2008.
11. S. Delvaux and M. Van Barel. The explicit QR-algorithm for rank structured matrices. Technical Report TW459, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, May 2006.
12. S. Delvaux and M. Van Barel. A *QR*-based solver for rank structured matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(2):464–490, 2008.
13. A. Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64(210):763–776, Apr. 1995.
14. Y. Eidelman and I. C. Gohberg. On a new class of structured matrices. *Integral Equations and Operator Theory*, 34:293–324, 1999.
15. Y. Eidelman, I. C. Gohberg, and V. Olshevsky. The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order. *Linear Algebra and its Applications*, 404:305–324, July 2005.
16. M. Fiedler and T. L. Markham. Completing a matrix when certain entries of its inverse are specified. *Linear Algebra and its Applications*, 74:225–237, 1986.
17. G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, USA, third edition, 1996.
18. R. Vandebril, M. Van Barel, G. H. Golub, and N. Mastronardi. A bibliography on semiseparable matrices. *Calcolo*, 42(3–4):249–270, 2005.
19. R. Vandebril, M. Van Barel, and N. Mastronardi. An implicit *QR*-algorithm for symmetric semiseparable matrices. *Numerical Linear Algebra with Applications*, 12(7):625–658, 2005.
20. R. Vandebril, M. Van Barel, and N. Mastronardi. A note on the representation and definition of semiseparable matrices. *Numerical Linear Algebra with Applications*, 12(8):839–858, Oct. 2005.
21. R. Vandebril, M. Van Barel, and N. Mastronardi. *Matrix Computations and Semiseparable Matrices, Volume II: Eigenvalue and Singular Value Methods*. Johns Hopkins University Press, 2008.
22. R. Vandebril, M. Van Barel, and N. Mastronardi. A parallel QR-factorization/solver of structured rank matrices. *Electronic Transactions on Numerical Analysis*, 30:144–167, 2008.