

A Multiple Shift QR -step for Structured Rank Matrices

Raf Vandebril

Marc Van Barel

Nicola Mastronardi

Report TW 489, April 2007



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Multiple Shift QR -step for Structured Rank Matrices

Raf Vandebril
Marc Van Barel
Nicola Mastronardi

Report TW 489, April 2007

Department of Computer Science, K.U.Leuven

Abstract

Eigenvalue computations for structured rank matrices are the subject of many investigations nowadays. There exist methods for transforming matrices into structured rank form, QR -algorithms for semiseparable and semiseparable plus diagonal form, methods for reducing structured rank matrices efficiently to Hessenberg form and so forth.

Eigenvalue computations for the symmetric case, involving semiseparable and semiseparable plus diagonal matrices have been thoroughly explored.

A first attempt for computing the eigenvalues of nonsymmetric matrices via intermediate Hessenberg-like matrices (i.e. a matrix having all subblocks in the lower triangular part of rank at most one) was restricted to the single shift strategy. Unfortunately this leads in general to the use of complex shifts switching thereby from real to complex operations.

This paper will explain a general multishift implementation for Hessenberg-like matrices (semiseparable matrices are a special case and hence also admit this approach). Besides a general multishift QR -step, this will also admit restriction to real computations when computing the eigenvalues of arbitrary real matrices.

Details on the implementation are provided as well as numerical experiments proving the viability of the presented approach.

Keywords : Multishift, QR -algorithm, structured rank matrices, implicit QR -algorithms

AMS(MOS) Classification : Primary : 65F15, Secondary : 15A18.

A Multiple Shift QR -step for Structured Rank Matrices ^{*}

Raf Vandebril¹, Marc Van Barel¹, Nicola Mastronardi²

¹ KULeuven, Dept. of Computer Science, 3001 Leuven(Heverlee)
e-mail: {raf.vandebril, marc.vanbarel}@cs.kuleuven.ac.be

² Istituto per le Applicazioni del Calcolo M. Picone, sez. Bari,
e-mail: irmanm21@area.ba.cnr.it

11th April 2007

Abstract Eigenvalue computations for structured rank matrices are the subject of many investigations nowadays. There exist methods for transforming matrices into structured rank form, QR -algorithms for semiseparable and semiseparable plus diagonal form, methods for reducing structured rank matrices efficiently to Hessenberg form and so forth.

Eigenvalue computations for the symmetric case, involving semiseparable and semiseparable plus diagonal matrices have been thoroughly explored.

A first attempt for computing the eigenvalues of nonsymmetric matrices via intermediate Hessenberg-like matrices (i.e. a matrix having all subblocks in the lower triangular part of rank at most one) was restricted to the single shift strategy. Unfortunately this leads in general to the use of complex shifts switching thereby from real to complex operations.

This paper will explain a general multishift implementation for Hessenberg-like matrices (semiseparable matrices are a special case and hence also admit this approach). Besides a general multishift QR -step, this will also admit restriction to real computations when computing the eigenvalues of arbitrary real matrices.

Details on the implementation are provided as well as numerical experiments proving the viability of the presented approach.

Key words Multishift, QR -algorithm, structured rank matrices, implicit QR -algorithms

1 Introduction

Eigenvalue computations related to structured rank matrices are the subject of research of many authors nowadays. The investigations concern translations of standard eigenvalue problems, higher order structured rank eigenvalue computations, polynomial rootfinding problems, generalized eigenvalue problems and so forth. Let us briefly discuss these investigations in more detail.

The traditional eigenvalue computation of a dense matrix, consists of a preliminary reduction of the matrix to tridiagonal or Hessenberg form. This matrix form is then exploited by the QR -algorithm, which computes the eigenvalues and/or eigenvectors [19,21]. The authors Van Barel, Vandebril and Mastronardi proposed a similar method but instead of working with tridiagonal

^{*} The research of the first two authors, was partially supported by the Research Council K.U.Leuven, project OT/05/40 (Large rank structured matrix computations), CoE EF/05/006 Optimization in Engineering (OPTEC), by the Fund for Scientific Research–Flanders (Belgium), G.0455.0 (RHPH: Riemann-Hilbert problems, random matrices and Padé-Hermite approximation), G.0423.05 (RAM: Rational modelling: optimal conditioning and stable algorithms), and by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization). The first author has a grant as “Postdoctoraal Onderzoeker” from the Fund for Scientific Research–Flanders (Belgium). The work of the third author was partially supported by MIUR, grant number 2004015437. The scientific responsibility rests with the authors.

matrices they considered the class of semiseparable matrices (inverses of tridiagonal matrices are of semiseparable form). An initial reduction to semiseparable, or to Hessenberg-like form was proposed in [23]. Based on this form an implicit QR -algorithm for semiseparable matrices was developed [28].

The authors Delvaux and Van Barel, focused attention more towards the higher order structured rank case. Explicit QR -algorithms for general structured rank matrices were developed [9], as well as algorithms for computing the eigenvalues of unitary structured rank matrices [8].

The root finding problem of polynomials can easily be transformed towards an eigenvalue problem involving a companion matrix. A companion matrix can be considered as a special type of structured rank matrix. The authors Bini, Gemignani, Pan, Eidelman and Gohberg derived several methods for solving this problem [2,3,4,5].

Of course there are many other methods of interest. Divide and conquer methods for computing the eigenvalues of semiseparable plus diagonal matrices were proposed by Chandrasekaran and Gu and by Van Barel, Mastronardi and Van Camp [6,20], reductions of structured rank matrices to Hessenberg form [11,15], eigenvalue computations of quasiseparable matrices [18,17], QZ -algorithms for structured rank matrices [25], and more.

In this manuscript we will consider the first mentioned algorithm. Namely the QR -algorithm for semiseparable and/or Hessenberg-like matrices (the inverse of a Hessenberg matrix is a Hessenberg-like matrix). The strategy developed in the manuscripts mentioned above was restricted to the single shift case. Unfortunately this means that for the Hessenberg-like case, one often has to switch to complex matrices, and one cannot stick to real matrix arithmetic. In the Hessenberg case this problem is solved by using multishift techniques. In this manuscript we will develop a technique for performing multishift iteration steps onto the Hessenberg-like or semiseparable matrix in an efficient manner.

The manuscript is organized as follows. The first preliminary section recapitulates some essential concepts for understanding the manuscript. The section covers the definition of Hessenberg-like matrices, their representation and the computation of the QR -factorization for Hessenberg-like plus diagonal matrices. Also a graphical scheme representing the application of Givens transformations onto a matrix is studied, as well as briefly the multishift setting. Section 3 discusses the problem setting. Details are presented on the approach we will follow and important results related to Hessenberg-like matrices are given. Section 4 and Section 5 discuss the implicit multishift approach for Hessenberg-like matrices. Section 4 focusses on the introduction of the disturbing Givens transformations, whereas Section 5 is dedicated to the structure restoring chasing technique. Section 6 presents some details concerning the implementation and some numerical experiments.

2 Preliminaries

In this section three important concepts will be discussed, essential for the remainder of the manuscript. We will discuss in the following order: a formal definition of Hessenberg-like matrices, an efficient representation for these matrices, interactions between Givens transformations, the QR -factorization of Hessenberg-like matrices and briefly the multishift setting.

It might seem at first that the introduced concepts are somewhat independent of each other, but pieces will fall together in the upcoming sections.

2.1 The definition

A Hessenberg-like matrix is often named a lower semiseparable matrix as the lower triangular part is in fact of semiseparable form.

Definition 1 A matrix $Z \in \mathbb{R}^{n \times n}$ is called *Hessenberg-like*, if and only if the following relation is satisfied¹:

$$\text{rank}(Z(i:n, 1:i)) \leq 1 \text{ with } i = 1, \dots, n.$$

This means that Z is called a Hessenberg-like matrix if and only if all submatrices taken out of the lower triangular part of the matrix, including the diagonal, have rank at most 1.

It is well-known that invertible Hessenberg-like matrices are the inverses of Hessenberg matrices. Moreover if the lower triangular part of the Hessenberg-like matrix can be written as coming from a rank 1 matrix $\mathbf{u}\mathbf{v}^T$, its inverse will be an irreducible Hessenberg matrix.

2.2 The representation

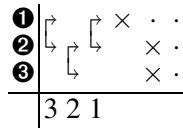
It was proven in [29, 10], that an efficient way to represent structured rank matrices consists of a (some) sequence(s) of Givens transformations and a vector (weights). Due to the flexible nature of this representation, we will restrict ourselves to Hessenberg-like matrices represented by the Givens-vector representation². Of course there are other also efficient representations such as the generator representation [1] or the quasiseparable representation [16, 14]. We choose however to work with the Givens-vector representation due to the close relation between this representation and the QR -factorization of the involved matrix. This will lead to efficient methods when considering the multishift technique.

The Givens-vector representation for this type of matrices was extensively discussed in previous manuscripts (see e.g. [29, 10]). In this article we will work with a graphical representation of this representation for clarifying the flow and ideas of the proposed method.

In the manuscript we will use figures to represent the interaction between Givens transformations and matrices [13]. Let us introduce this representation. Suppose we have the following product of matrices. Three Givens transformations are applied onto the matrix on the right:

$$\begin{bmatrix} c_3 & -\bar{s}_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & -\bar{s}_2 \\ 0 & s_2 & c_2 \end{bmatrix} \begin{bmatrix} c_1 & -\bar{s}_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix}. \quad (1)$$

The application of these three Givens transformations onto the matrix, is graphically depicted as follows:



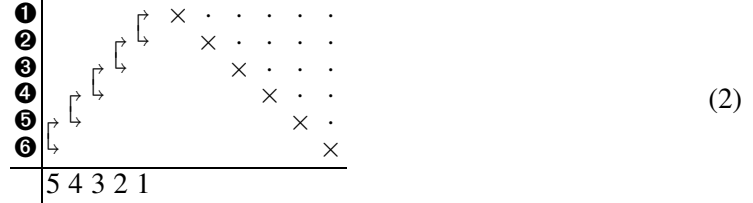
On the right of the figure we see a matrix with the same structure as the right matrix in Equation (1). The important elements, for our purposes, are marked by \times , the less important elements are marked with \cdot . The elements not shown (in the first column) are assumed to be zero. The vertical axis with elements $\textcircled{1}, \textcircled{2}, \dots$ depicts the rows of the matrix. The horizontal axis with numbers $1, 2, 3, \dots$ depicts a time axis of the Givens transformations in the columns above. It denotes in which order the Givens transformations have to be applied onto the matrix on the right. The long arrow-pointed brackets denote Givens transformations. The first Givens transformation (corresponds to the transformation with parameters c_1, s_1), which is shown in column 1, affects only the first two rows of the matrix, this is illustrated by the arrows on the bracket. The second Givens transformation (corresponds to the one with parameters c_2, s_2), acts on the second and third row, and finally the last Givens transformation acts again on the first two rows. Even though the Givens transformations are not specified in terms of their cosines and sines, and also the matrix is not specified by its elements, the figure clearly denotes the interaction of arbitrary Givens transformations onto the matrix.³

¹ With $Z(i:j, k:l)$ the submatrix with rows from i up to j and k up to l is meant.

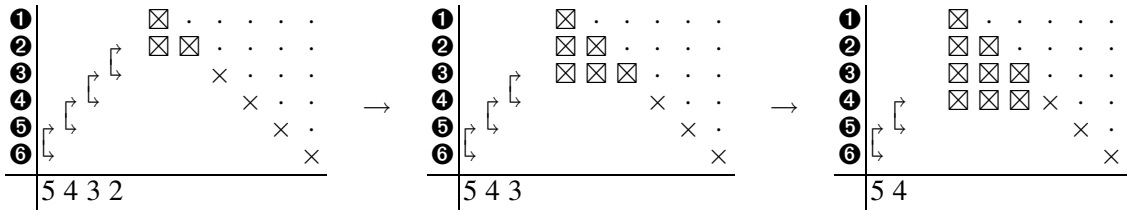
² There is of course no loss in generality, as this representation covers the complete set.

³ Givens transformations can equal the identity, without loss of generalization.

The following figure represents in a graphical way the Givens-vector representation. Let us illustrate the action of the Givens transformations, on the vector elements (denoted by the elements \times) and how this constructs the lower triangular low rank part.



Applying the first Givens transformation (this is the transformation in column 1) onto the matrix R on the right, will change only the first two rows. The next Givens transformation acts on row two and three and smears out the information of the first two elements in row two towards row three. This process can be continued and one can clearly see that the resulting upper triangular matrix gets filled up with a lower triangular part in which the elements \boxtimes denote the lower triangular structured rank part. All submatrices taken out of the part marked by \boxtimes are of rank at most 1. Clearly the matrix on the right is of Hessenberg-like form. The following figure depicts the successive application of the first three Givens transformations onto the matrix R on the right.



In some sense this scheme depicts also the QR -factorization of a Hessenberg-like matrix. In practice one represents the lower triangular part with the Givens transformations and the elements \times , but for theoretical considerations one can assume this to be a QR -factorization of the involved matrix. The right part represents the matrix R and the left part the sequence of Givens transformations from bottom to top. This representation is called a representation from top to bottom. Another Givens-vector representation is the representation from right to left. This representation is highly related to the RQ -factorization of the considered matrix. In every step a Givens transformation is applied acting on the columns, going from the right to left thereby gradually filling up the matrix. The transition from a top bottom to a right to left representation is called the swapping and was discussed in e.g. [10,26]. This transition is cheap and can be performed in linear time. In the remaining part of the manuscript, we will often use the connection between the representations and the QR/RQ -factorizations of the matrices.

This representation will be used extensively when discussing the multishift QR -algorithm for Hessenberg-like matrices. Disturbing Givens transformations will be posed onto a matrix of the form (2), the chasing technique will then restore the structure of this matrix.

2.3 Operations with Givens transformations

In order to be able to restore the Hessenberg-like structure, we need some essential operations involving Givens transformations. We will discuss the shift through lemma and the fusion of Givens transformations. Based on these operations we will show how to transform a \vee -pattern into a \wedge -pattern of Givens transformations. This transition is essential for developing the chasing technique.

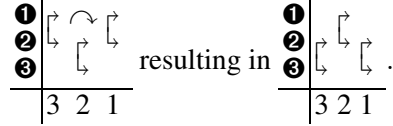
Lemma 1 (Shift through lemma) *Suppose three 3×3 Givens transformations G_1, G_2 and G_3 are given, such that the Givens transformations G_1 and G_3 act on the first two rows of a matrix, and G_2 acts on the second and third row.*

Then we have that

$$G_1 G_2 G_3 = \hat{G}_1 \hat{G}_2 \hat{G}_3,$$

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.

The proof is straightforward, based on the factorization of a unitary 3×3 matrix. Graphically the shift through lemma is depicted as follows.

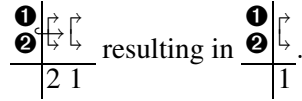


For indicating the transition from the right to the left figure, we use \curvearrowright .

Lemma 2 Suppose two Givens transformations G_1 and G_2 are given.

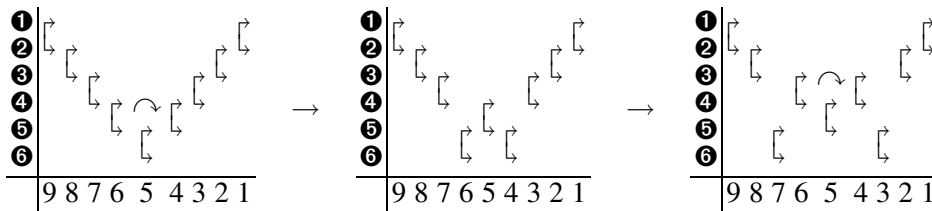
Then we have that $G_1 G_2 = G_3$ is again a Givens transformation. We will call this the fusion of Givens transformations in the remainder of the text.

The proof is trivial. In our graphical schemes, we will depict this as follows:

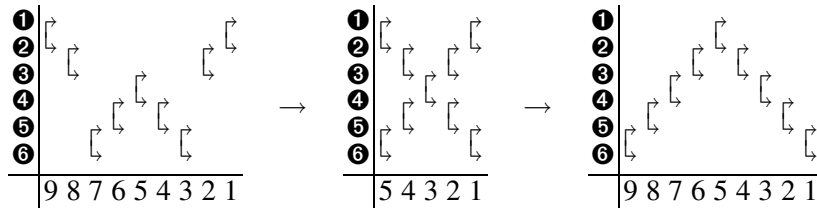


To illustrate the power of the shift through lemma we will show how one can easily transform a \vee -pattern of Givens transformations into a \wedge -pattern. Moreover this transition is needed in the upcoming chasing technique.

Suppose we have a sequence of Givens transformations of the following form. We do not discuss the interaction on the matrix, hence there are no matrix elements shown. This is called the \vee -pattern, shown as the most left figure. In the left figure already the first application of the shift through lemma is depicted. Applying the shift through lemma results in the middle figure, which can be rewritten to obtain the right figure. This rewriting, to go from the middle to the right figure is possible as the Givens transformations in positions 7 and 6 (4 and 3) operate on different rows. The last figure already indicates another application of the shift through lemma.



Applying now again the shift through lemma involving the Givens transformations in position 6, 5 and 4 results in a so-called \times -pattern, shown in the middle figure. This pattern can be used for parallel computing of the QR-factorization of e.g. Hessenberg-like matrices [30]. It is clear that this process of applying the shift through lemma can be continued to obtain the right figure, which denotes the \wedge -pattern.



In each step of the chasing procedure we will need this transformation from a \vee to a \wedge -pattern two times. One can see that this transition involves k times the application of the shift through lemma if there are $2k + 1$ Givens transformations involved. Hence the total complexity of this operation is $O(k)$.

2.4 The QR -factorization

The QR -factorization of a Hessenberg-like matrix is well-known [24]. We will recapitulate the most important facts. Suppose we have a matrix Z of Hessenberg-like form and a diagonal matrix D . The QR -factorization of the matrix $Z - D$ involves $2n$ Givens transformations⁴ in total. The first n transformations are performed from bottom to top, in order to remove the rank structure of the matrix Z . Hence we obtain

$$G_n G_{n-1} \dots G_1 (Z - D) = R - H,$$

where R is an upper triangular matrix and H is a Hessenberg matrix, in fact G_n is not really necessary, see the footnote. Important to know is the fact that these Givens transformations G_1 up to G_n can be chosen as the inverse of the Givens transformations from the Givens-vector representation of the matrix Z . This can clearly be seen in the graphical figure (2). Annihilating these Givens transformations on the left creates the upper triangular matrix R .

The remaining matrix $R - H$ is of Hessenberg form and can easily be brought to upper triangular form by a sequence of n Givens transformations from top to bottom, consecutively annihilating all subdiagonal elements.

We remark that combining all Givens transformations to bring $Z - D$ to upper triangular form, namely n Givens transformations from bottom to top followed by n Givens transformations from top to bottom, leads to a \wedge -pattern of annihilation.

2.5 The multishift setting

Explicitly, the multishift QR -step on an arbitrary matrix A is of the following form. Suppose we want to apply k shifts at the same time, denote them with σ_1 up to σ_k . Assume we computed the following QR -factorization:

$$QR = (A - \sigma_1 I)(A - \sigma_2 I) \dots (A - \sigma_k I).$$

The multishift QR -step consists now of applying the following similarity transformation onto the matrix A :

$$\hat{A} = Q^H A Q.$$

An implicit multishift QR -step consists of performing an initial disturbing unitary similarity transformation onto the matrix A , followed by structure restoring similarity transformations, not involving the first column.

Let us explain in more detail the implicit version of the multishift QR -step in case the matrix $A = H$, is a Hessenberg matrix. More precisely, assume

$$\mathbf{v} = (H - \sigma_1 I)(H - \sigma_2 I) \dots (H - \sigma_k I) \mathbf{e}_1, \quad (3)$$

in which $\mathbf{e}_1 = [1, 0, \dots, 0, 0]^T$ denotes the first basis vector of \mathbb{C}^n . Suppose we design now a specific unitary matrix \tilde{Q} such that $\tilde{Q}^H \mathbf{v} = \pm \|\mathbf{v}\| \mathbf{e}_1$. Apply then this transformation onto the matrix H , resulting in

$$\hat{H} = \tilde{Q}^H H \tilde{Q}.$$

Because the matrix H is of Hessenberg form and the matrix \tilde{Q} was constructed in a specific manner, this results in the introduction of a bulge below the subdiagonal.

The implicit approach is then completed by transforming the matrix H back to its original form, e.g. Hessenberg form, by successively applying unitary similarity transformations. The unitary transformation matrix \hat{Q} needs to be chosen such that $\hat{Q} \mathbf{e}_1 = \mathbf{e}_1$ and

$$\hat{H} = \hat{Q}^H \tilde{Q}^H H \tilde{Q} \hat{Q},$$

with \hat{H} again a Hessenberg matrix. The condition $\hat{Q} \mathbf{e}_1 = \mathbf{e}_1$ is necessary to prove that this implicit QR -step is essentially the same as an explicit QR -step. One can prove this via the implicit Q -theorem (see [19]), or via other approaches (see [31]). In the following section we will describe in more detail the problem setting for Hessenberg-like matrices.

⁴ Combining two Givens transformations leads to $2n - 1$ transformations.

3 The problem setting

An algorithm for reducing a nonsymmetric matrix via orthogonal similarity transformations into Hessenberg-like form, was discussed in [23]. The advantage of this reduction w.r.t. the traditional reduction to Hessenberg form lies in the fact that dominant eigenvalues or clusters are already revealed during the reduction phase. This is not the case in the traditional approach.

Following the reduction method QR -methods for semiseparable and Hessenberg-like matrices were proposed. The QR -method for semiseparable matrices was proposed in [28], and was directly applicable to the Hessenberg-like case. To develop this technique a translation of the implicit Q -theorem for Hessenberg-like matrices was developed [27], as well as theorems proving the maintainance of the Hessenberg-like structure under a step of the QR -algorithm [28, 7, 12].

The main steps for performing a QR -step with a single shift on a Hessenberg-like matrix Z are the following ones, assuming we have already a shift κ :

- Perform a step of the QR -algorithm without shift onto the matrix $Z = Z^{(1)}$. We obtain $Z^{(2)} = Q^H Z^{(1)} Q$, which is again a Hessenberg-like matrix. Important to remark is that the orthogonal matrix Q^H is essentially a combination of the sequence of Givens transformations from top to bottom used in the Givens-vector representation of the matrix $Z^{(1)}$. This connection leads to an efficient implementation of this step.
- Based on the previous step and on the shift κ the vector $(Z - \kappa I)\mathbf{e}_1 = \mathbf{v}$ is transformed into a vector $Q^H \mathbf{v} = [\times, \times, 0, \dots, 0]^T$. A Givens transformation \hat{G}^H is determined such that performing it onto this vector creates a multiple of \mathbf{e}_1 .
- Apply this Givens transformation \hat{G}^H onto the matrix $Z^{(2)}$, which results in a matrix which is not of Hessenberg-like form anymore.
- Restore the structure of the matrix $\hat{G}^H Z^{(2)} \hat{G}$, by constructing a matrix \hat{Q} , with $\hat{Q}\mathbf{e}_1 = \mathbf{e}_1$, such that performing an orthogonal similarity transformation with this matrix leads to $\hat{Q}^H \hat{G}^H Z^{(2)} \hat{G} \hat{Q}$, that is again a Hessenberg-like matrix.

Based on the implicit Q -theorem for Hessenberg-like matrices, one can prove that the result is essentially the same result as coming from an explicit step of the QR -algorithm.

Unfortunately only the single shift case was discussed and developed. In general a real Hessenberg-like matrix does not only have real eigenvalues, and hence one needs to switch to complex arithmetic for computing the eigenvalues of Hessenberg-like matrices via this approach. The multishift QR -step for Hessenberg-like matrices we are about to propose will solve this problem. The setting discussed in this manuscript is very general and covers k shifts, also applicable for semiseparable matrices.

Due to the preservation of the Hessenberg-like structure under a QR -step, we know that the structure is also preserved under a multishift QR -step (see e.g. [19]). To design an implicit multishift QR -algorithm for Hessenberg-like matrices, we use the idea of introducing the distortion by reducing the first column \mathbf{v} (as in Equation (3), but for the Hessenberg-like case) to \mathbf{e}_1 . In the Hessenberg case we will see that the vector \mathbf{v} has only $k + 1$, in case of k shifts, elements different from zero, located all in the top positions of the vector. To annihilate these elements, and transforming the vector \mathbf{v} to $\beta\mathbf{e}_1$, one can easily perform a single Householder transformation or k Givens transformations. Unfortunately, in case of a Hessenberg-like matrix Z we will see that the vector \mathbf{v} is generally full. How to efficiently choose transformations for reducing \mathbf{v} to $\beta\mathbf{e}_1$, such that the corresponding similarity transformations can be efficiently performed onto the Hessenberg-like matrix Z , is the subject of the next section.

4 An efficient transformation from \mathbf{v} to \mathbf{e}_1

The reduction of an arbitrary vector \mathbf{v} to $\beta\mathbf{e}_1$, a multiple of the first basis vector can easily be accomplished by performing a Householder transformation or $n - 1$ Givens transformations. Unfortunately applying the Householder or the Givens transformations, onto the Hessenberg-like matrix creates an overhead on distortion in the matrix Z , which cannot be removed efficiently.

It seems that a more appropriate technique is needed to reduce the complexity of restoring the structure of the disturbed matrix Z .

Reconsidering the single shift case, we know that the Givens transformations needed for bringing the vector \mathbf{v} back to $\beta\mathbf{e}_1$ are closely related to the Givens transformations used in the Givens-vector representation (see Section 3). Hence, we will search how to link the transformation of \mathbf{v} to $\beta\mathbf{e}_1$ with the Givens-vector representation of the matrix Z .

Assume we have the following matrix \tilde{Z} , whose first column we would like to reduce efficiently to $\beta\mathbf{e}_1$. Denote, $Z^{(1)} = Z$ and the σ_i are suitably chosen shifts:

$$\tilde{Z} = \left(Z^{(1)} - \sigma_1 I\right) \left(Z^{(1)} - \sigma_2 I\right) \dots \left(Z^{(1)} - \sigma_k I\right).$$

The matrix $Z^{(1)}$ is represented with the Givens-vector representation, and hence we have $Z^{(1)} = Q^{(1)}R^{(1)}$, in which the matrix $Q^{(1)}$ contains the Givens transformations, from the representation, and the matrix $R^{(1)}$ is an upper triangular matrix. This leads to:

$$\tilde{Z} = Q^{(1)} \left(R^{(1)} - H_1^{(1)}\right) Q^{(1)} \left(R^{(1)} - H_2^{(1)}\right) \dots Q^{(1)} \left(R^{(1)} - H_k^{(1)}\right),$$

in which $H_i^{(1)} = Q^{(1)H} \sigma_i I$, for $i = 1, \dots, k$ and all matrices $H_i^{(1)}$ are Hessenberg matrices.

We will now transform the first column of the matrix \tilde{Z} to $\beta\mathbf{e}_1$, but in such a way that we can perform these transformations efficiently, as similarity transformations onto the matrix $Z^{(1)}$.

Perform the transformation $Q^{(1)H}$ onto the matrix \tilde{Z}

$$\begin{aligned} Q^{(1)H} \tilde{Z} &= \left(R^{(1)} - H_1^{(1)}\right) Q^{(1)} \left(R^{(1)} - H_2^{(1)}\right) \dots Q^{(1)} \left(R^{(1)} - H_k^{(1)}\right) \\ &= \left(R^{(1)} Q^{(1)} - \sigma_1 I\right) \left(R^{(1)} Q^{(1)} - \sigma_2 I\right) \dots \left(R^{(1)} Q^{(1)} - \sigma_{k-1} I\right) \left(R^{(1)} - H_k^{(1)}\right). \end{aligned}$$

The matrix product $R^{(1)} Q^{(1)}$ produces a new Hessenberg-like matrix $Z^{(2)}$. (Check the correspondence with the right to left representation of Hessenberg-like matrices). The RQ -factorization of the matrix considered can easily be transformed to the QR -factorization. Due to the connection with the Givens-vector representation this can be done efficiently, moreover we will see further on in the text that we do need this transition anyway. Denote this as $Z^{(2)} = Q^{(2)}R^{(2)}$. Important to remark is the fact that the matrix $Z^{(2)}$ is in fact the result of performing a step of the QR -algorithm without shift onto the matrix $Z^{(1)}$, this is equivalent as in the single shift approach discussed in [28]. We have the equality $Z^{(2)} = Q^{(1)H} Z^{(1)} Q^{(1)}$.

We obtain the following relations (with $H_i^{(2)} = Q^{(2)H} \sigma_i I$, for $i = 1, \dots, k-1$):

$$\begin{aligned} Q^{(1)H} \tilde{Z} &= Q^{(2)} \left(R^{(2)} - H_1^{(2)}\right) Q^{(2)} \left(R^{(2)} - H_2^{(2)}\right) \dots Q^{(2)} \left(R^{(2)} - H_{k-1}^{(2)}\right) \left(R^{(1)} - H_k^{(1)}\right) \\ Q^{(2)H} Q^{(1)H} \tilde{Z} &= \left(R^{(2)} Q^{(2)} - \sigma_1 I\right) \left(R^{(2)} Q^{(2)} - \sigma_2 I\right) \dots \left(R^{(2)} Q^{(2)} - \sigma_{k-2} I\right) \left(R^{(2)} - H_{k-1}^{(2)}\right) \left(R^{(1)} - H_k^{(1)}\right). \end{aligned}$$

It is clear that the procedure can easily be continued by defining

$$Z^{(3)} = Q^{(2)H} Z^{(2)} Q^{(2)} = R^{(2)} Q^{(2)},$$

which is the result of applying another step of QR -without shift onto the matrix $Z^{(2)}$.

As a result we obtain that, if for every $i = 1, \dots, k$

$$\begin{aligned} Z^{(i)} &= Q^{(i)} R^{(i)}, \\ Z^{(i+1)} &= Q^{(i)H} Z^{(i)} Q^{(i)} = R^{(i)} Q^{(i)}, \\ H_{k-i+1}^{(i)} &= Q^{(i)H} \sigma_{k-i+1} I \end{aligned}$$

the following relation holds

$$Q^{(k)H} \dots Q^{(1)H} \tilde{Z} = \left(R^{(k)} - H_1^{(k)}\right) \left(R^{(k-1)} - H_2^{(k-1)}\right) \dots \left(R^{(3)} - H_{k-2}^{(3)}\right) \left(R^{(2)} - H_{k-1}^{(2)}\right) \left(R^{(1)} - H_k^{(1)}\right) \quad (4)$$

The first column of $Q^{(k)H} \dots Q^{(1)H} \tilde{Z}$ has only the first $k + 1$ elements different from zero as it is the product of k Hessenberg matrices. These elements can be computed easily as the matrices $H_{k-i+1}^{(i)} = Q^{(i)H} \sigma_{k-i+1} I$, in which $Q^{(k)}$ is a sequence of Givens transformations from bottom to top. For computing the first column of the matrix product $Q^{(k)H} \dots Q^{(1)H} \tilde{Z}$, we efficiently compute the multiplication of the right-hand side of Equation 4 with \mathbf{e}_1 .

Moreover combining all these transformations, and perform them as a similarity transformation onto the matrix $Z^{(1)}$ gives us

$$\begin{aligned} Q^{(k)H} \dots Q^{(2)H} Q^{(1)H} Z^{(1)} Q^{(1)} Q^{(2)} \dots Q^{(k)} \\ &= Q^{(k)H} \dots Q^{(2)H} Z^{(2)} Q^{(2)} \dots Q^{(k)} \\ &= Q^{(k)H} \dots Q^{(3)H} Z^{(3)} Q^{(3)} \dots Q^{(k)} \\ &= Z^{(k)}. \end{aligned}$$

This corresponds to performing k steps of the QR -method without shift onto the matrix $Z^{(1)}$, which can be performed efficiently in $O(n^2)$. (Especially in the semiseparable case one can perform these steps in linear time $O(n)$.)

To complete the reduction of \mathbf{v} to $\beta \mathbf{e}_1$, we have to annihilate k of the $k + 1$ nonzero elements of the vector $Q^{(k)H} \dots Q^{(1)H} \tilde{Z} \mathbf{e}_1$. This can be done by performing k Givens transformations $\hat{G}_k \dots \hat{G}_1$ annihilating first the nonzero element in position $k + 1$, followed by annihilating the element in position k and so forth. This results in the matrix

$$\hat{G}_k^H \dots \hat{G}_1^H Z^{(k)} \hat{G}_1 \dots \hat{G}_k,$$

which we have to bring back via similarity transformations to Hessenberg-like form.

Let us summarize this step. In order to start the implicit procedure, in case of k shifts, we have to perform k steps of QR -without shift onto the matrix $Z^{(1)}$. Moreover we have to use these unitary transformations to compute the first column $Q^{(k)H} \dots Q^{(1)H} \tilde{Z} \mathbf{e}_1$ in an efficient way. The result of these k QR -steps without shift onto $Z^{(1)}$ gives us again a Hessenberg-like matrix $Z^{(k)}$. The vector $Q^{(k)H} \dots Q^{(1)H} \tilde{Z} \mathbf{e}_1$, is still not a multiple of \mathbf{e}_1 and hence an extra k Givens transformations are needed to transform this vector to $\beta \mathbf{e}_1$.⁵ These final k Givens transformations disturb significantly the Hessenberg-like structure of the matrix $Z^{(k)}$. As a result of a step of QR , we know that the matrix will again be of Hessenberg-like form. Hence we will develop in the next section a chasing technique for restoring the structure.

5 The chasing method

The chasing method consists of two parts. First we need to perform some transformations to prepare the matrix for the chasing procedure. Let us name this first part the initialization procedure.⁶ The second part is the effective chasing part. For simplicity we will demonstrate the techniques on a 7×7 matrix for $k = 3$ multishifts. The technique can be extended easily to larger matrices and more shifts.

5.1 Initialization

The matrix we are working with, $Z^{(3)}$, is of Hessenberg-like form. Denote the initial disturbing Givens transformations, three in this case, with \hat{G}_1, \hat{G}_2 and \hat{G}_3 .

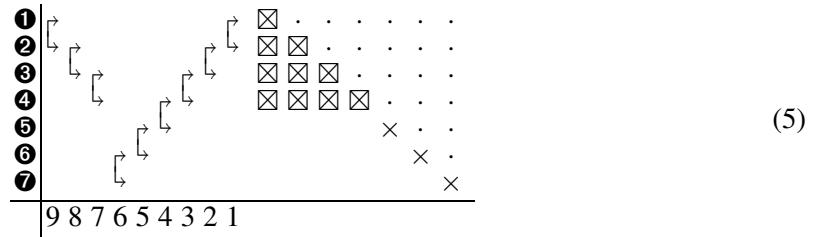
The chasing procedure needs to be performed onto the following matrix

$$\hat{G}_3^H \hat{G}_2^H \hat{G}_1^H Z^{(3)} \hat{G}_1 \hat{G}_2 \hat{G}_3.$$

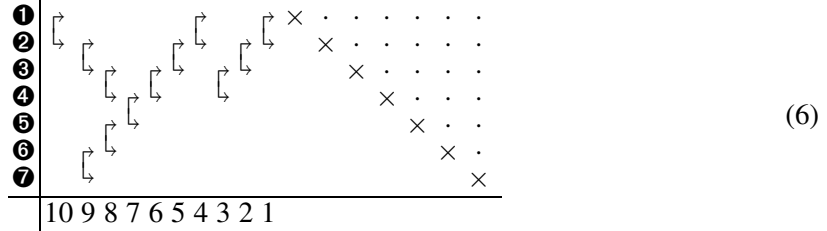
⁵ The performance of k steps of the QR -algorithm without shift do not dramatically increase the complexity of the multishift w.r.t. single shift strategy, also in the single shift strategy a QR -step without shift is needed.

⁶ In some sense the initialization is not needed, see a remark later on.

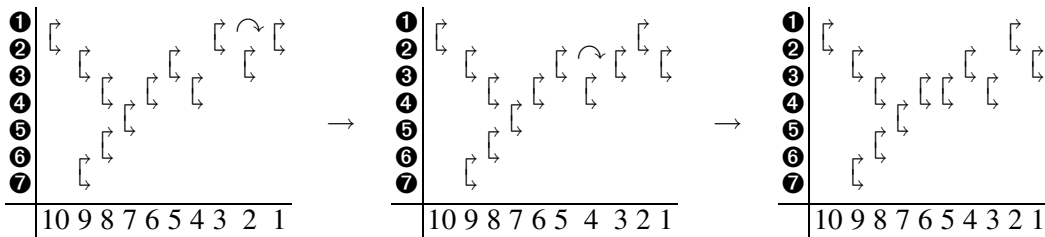
Recall that the transformation \hat{G}_1^H acts on rows 3 and 4, \hat{G}_2^H on rows 2 and 3 and \hat{G}_3^H on rows 1 and 2. We will switch now to the graphical representation of the matrix $Z^{(3)}$. For theoretical considerations we consider the graphical schemes as QR -factorizations of the matrix $Z^{(3)} = Q^{(3)}R^{(3)}$. In the following figure, we have to apply the transformations on the left to the matrix $Q^{(3)}$ and the transformations on the right are applied onto the upper triangular matrix $R^{(3)}$. We obtain the following figure, in which the upper structured rank part, denoted by the elements \boxtimes is created by performing the three Givens transformations on the right. More precisely in the figure we see on the left $\hat{G}_3^H \hat{G}_2^H \hat{G}_1^H Q^{(3)}$, where the three disturbing Givens transformations are found in position 7, 8 and 9 and the matrix $Q^{(3)}$ consisting of 6 Givens transformations is found in the positions 1 to 6. The matrix on the right of the figure is the matrix $R^{(3)} \hat{G}_1 \hat{G}_2 \hat{G}_3$, the result of applying the three Givens transformations onto the upper triangular matrix. This new upper left part, denoted by \boxtimes is in fact a small Hessenberg-like matrix.



The initial step consists of removing this newly created small structured rank part denoted by \boxtimes . One can easily remove this part by performing three Givens transformations, just like when reducing a Hessenberg-like matrix to upper triangular form. In fact this is a swapping procedure in which the representation from right to left acting on the columns is transformed into a representation from top to bottom acting on the rows (see e.g. [26]). This leads to the following figure in which the first three Givens transformations (in positions 1, 2 and 3) are chosen to annihilate the newly created part denoted by \boxtimes . At the same time we compressed the notation in the left part.

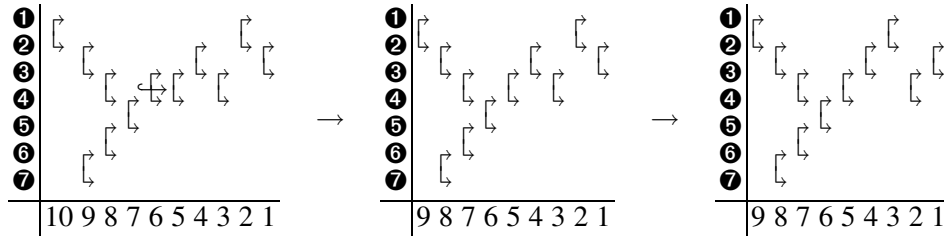


For the moment we do not need the upper triangular part on the right and we will therefor not depict it in the following figures. We will now try to rearrange the Givens transformations shown in Figure (6), in order to remove the Givens transformation in position 1. This is done by applying few times the shift through lemma. Interchanging the Givens transformations in position 3 and 4 from Figure (6), leads to the left figure below, in which we already depicted the first application of the shift through lemma. The second figure shows the result of applying the shift through lemma and indicates the next application. The right figure is the result of applying the shift through lemma as indicated in the middle figure.



We have now finished applying the shift through lemma, generally we need to apply it $k - 1$ times. To complete the procedure, a fusion of the Givens transformations in position 5 and 6 needs

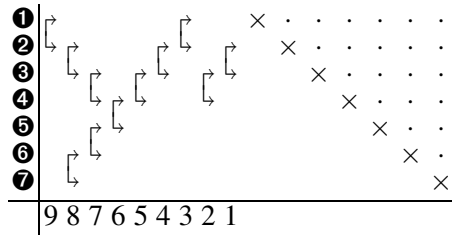
to be performed. The left figure you see on which Givens operations the fusion will act, the middle figure shows the result, and in the right figure we have rewritten the scheme by interchanging the order of transformations 3 and 2.



We can clearly see that this form is similar to the one we started from in Figure (6), except for the fact that the Givens transformation in the first position of Figure (6) is now removed. This is the structure on which we will perform the chasing method. This initialization procedure only has to be performed once, before starting the chasing.

5.2 Restoring the structure

The figure at the end of the previous section can be decomposed into four main parts. The complete figure, including the matrix on which the transformations act is of the following form.



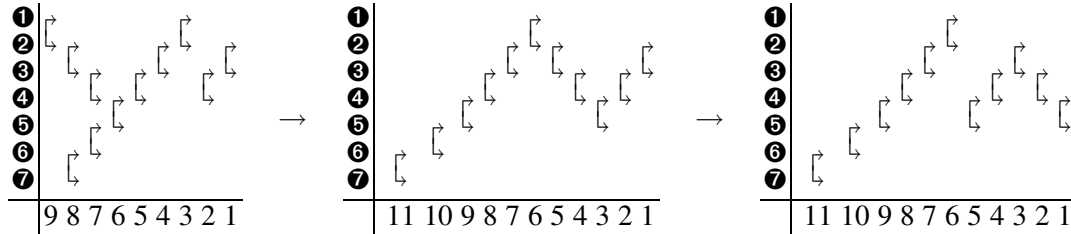
Let us discuss these 4 different parts and indicate the restoring procedure. A first part is the matrix on the right on which the Givens transformations proposed on the left act. The remaining three parts denote specific parts in the Givens transformations on the left. A first part contains the Givens transformations in position 1 and 2, generally this will contain in case of k shifts $k - 1$ Givens transformations. Secondly we have the Givens transformations from the representation, this is the sequence from top to bottom, ranging from position 3 until 8. Finally we have three more Givens transformations in position 7, 8 and 9 shown on the top left part of the figure, generally there are k Givens transformations in this part.

The aim of the procedure is to remove the Givens transformations on the left upper part and the ones in positions 1 and 2, to obtain only a sequence from top to bottom which will denote the new representation. The right-hand side should remain an upper triangular matrix. This should be accomplished by performing unitary similarity transformations onto the matrix, remark that for all transformations performed on the right-hand side, we cannot change the first column: $\hat{Q}e_1$ needs to be equal to e_1 .

The idea is to reshuffle the Givens transformations based on the shift-through operation, then apply them onto the upper triangular matrix, followed by a structure restoring similarity transformation. This structure restoring similarity transformation will result in a scheme similar to the one above, only the undesired Givens transformations in the positions 1, 2, 7, 8 and 9 will have shifted down one position. This procedure can hence be continued until these Givens transformations are slid off the figure.

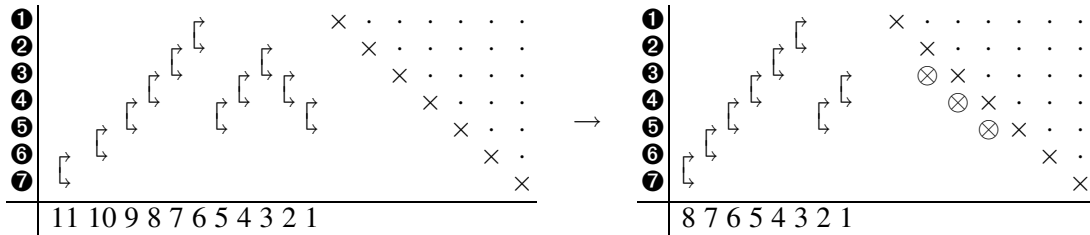
We start by rearranging the Givens transformations. First we change the \vee -pattern on top ranging from position 3 to 9 (see the left figure below) to a \wedge -pattern as discussed in Section 2.3. This transition is only possible if we shift the Givens transformations on the bottom, in position 7 and 8 backwards. The new \wedge -patterns is of course still located between position 3 and 9 as shown

in the middle figure. Now, see the middle figure, we have again a ∇ -pattern ranging from position 1 to 5. We transform it to a \wedge -pattern shown on the right.

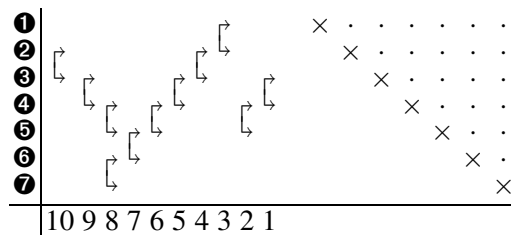


In fact we dragged the top three transformations in the positions 7,8 and 9 from the left figure completely to the right located now in positions 1,2 and 3. Remark that instead of acting on the first four rows, they act now on the rows 2, 3, 4 and 5. Moreover the Givens transformations located in the left figure in the positions 1 and 2 can be found now in the positions 4 and 5, also shifted down one row.

To remove now the Givens transformations in the first three positions, we need to apply a similarity transformation. To achieve this goal, we need the upper triangular matrix $R^{(3)}$. Applying the first three Givens transformations onto the matrix $R^{(3)}$ results in the following figure. The elements \otimes are filled in by applying these Givens transformations onto the upper triangular matrix.



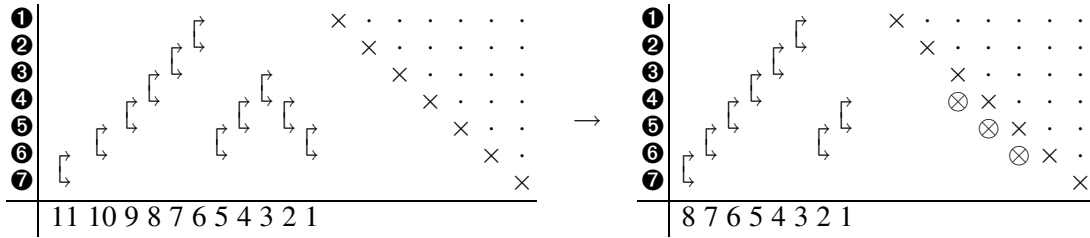
One can consider the right figure above as $\hat{Q}^{(3)}\hat{R}^{(3)}$, in which $\hat{Q}^{(3)}$ denotes the combination of all Givens transformations and the matrix $\hat{R}^{(3)}$ denotes the upper triangular matrix disturbed in few subdiagonal elements. To remove the bulges, denoted by \otimes in the matrix $\hat{R}^{(3)}$ we apply three Givens transformations acting on its columns 2,3,4 and 5. As only similarity transformations are allowed for restoring the structure also three Givens transformations have to be applied onto the matrix $\hat{Q}^{(3)}$. This completes one step of the chasing procedure and results in the figure below. The similarity transformation, removed the bulges, but introduced three new Givens transformations shown in positions 8,9 and 10.



As a result we obtain that the unwanted Givens transformations have shifted down one row, and this completes the first similarity transformation in the chasing procedure.

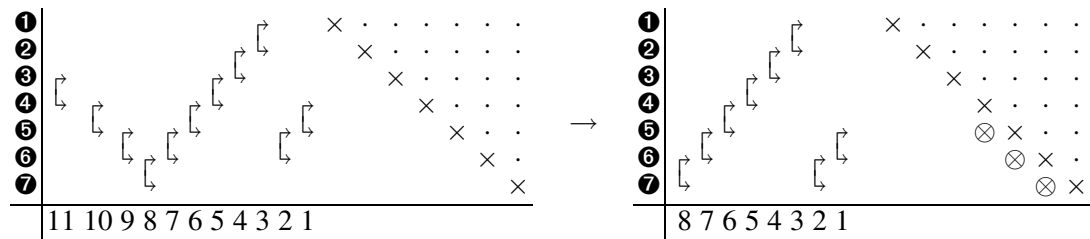
Let us continue now, in an identical way to determine the second similarity transformation. Rearranging the Givens transformation, by twice applying the transformation from a ∇ to a \wedge -pattern gives the left figure and then applying the right Givens transformations to the matrix gives

us the right figure.



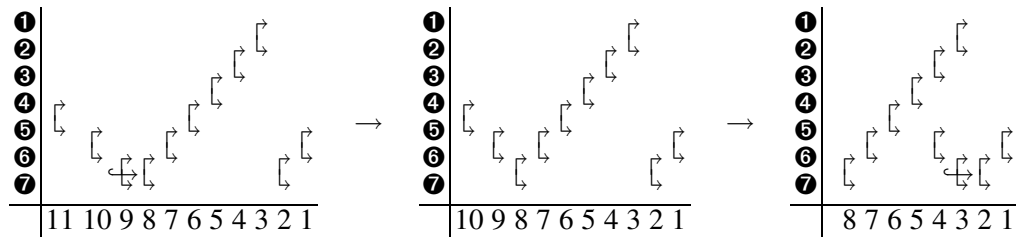
Again we have to apply a similarity transformation acting on the columns of the right disturbed upper triangular matrix to remove the bulges, denoted by \otimes .

This results in the matrix in the bottom left figure. Rewriting all the Givens transformations and applying them onto the upper triangular matrix reveals another similarity transformation, that needs to be performed.

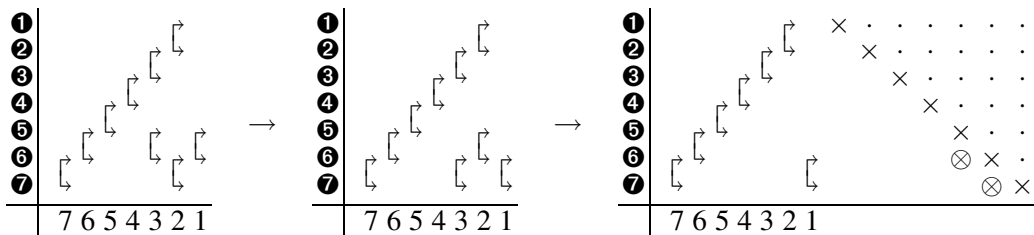


This chasing procedure needs to be performed as many times, until the unwanted Givens transformations reach the bottom rows. We see that the Givens transformations in the positions 1 and 2 have now arrived at the bottom row. From now on the chasing procedure is finishing, and gradually all the undesired Givens transformations will vanish.

The left figure below denotes the result of applying the similarity transformation proposed by the figure above to remove the bulges. We see in the left figure that the removing of the Givens transformations has started. The fusion of the Givens transformations in positions 9 and 8 is depicted. We do not show the upper triangular matrix as it does not change in the following operations. After the fusion we perform the change from the ∇ to the \wedge -pattern (from the middle to the right figure), again another fusion is indicated.

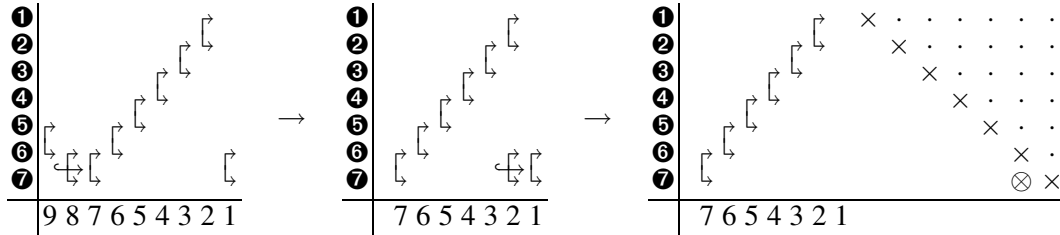


The fusion of the Givens transformations in positions 2 and 3 creates again the possibility of changing the ∇ to the \wedge -pattern and applying the first 2 disturbing Givens transformations. This is shown in the following figures.

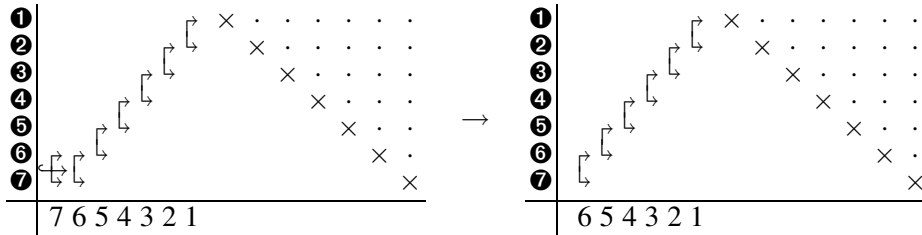


It is clear from the right figure that the next similarity transformation, to remove the bulges, involves only 2 Givens transformations instead of 3 as in all previous chasings. Performing the

similarity transformation and continuing this procedure will remove all unwanted Givens transformations. Let us depict the main steps in figures. Applying the similarity transformation creates the figure on the left. The fusion is depicted. Applying the fusion and rewriting the \vee -pattern leads to the middle figure, in which again a fusion is depicted. Applying the fusion and performing this transformation onto the upper triangular matrix results in one bulge in the bottom row.



A final similarity transformation consisting of one Givens transformation, needs to be performed. The similarity transformation, acting on the last two rows and the last two columns leads to the following figures, in which one final fusion needs to be performed, in order to obtain the new Givens-vector representation of the Hessenberg-like matrix shown in the right figure.



Remark 1 There are other variations in the chasing procedure. Instead of applying the initialization procedure, thereby performing the initial transformations on the columns, and transforming them to Givens transformations acting on the rows, one can just leave them on the right, acting on the columns.

The changed chasing procedure needs one swapping of the \vee -pattern to the \wedge -pattern and then immediately applying these transformations onto the rows. The created bulges then need to be removed by Givens transformations acting on the columns. As there are still existing Givens transformations acting on the columns we have to perform another pattern change (but then on the right-side of the matrix, instead of the left side), before one knows which similarity transformations need to be performed. Applying the similarity transformation introduces again some Givens transformations on the left. The reader can try to develop this figures himself. It gets more complicated to depict the figures as Givens transformations acting on the columns as well as on the rows are present now.

The chasing procedure discussed above, only involved Givens transformations performed on the columns not changing the first column. This means that combining all similarity transformations into the matrix \hat{Q} satisfies $\hat{Q}e_1$. Based on the implicit Q -theorem for Hessenberg-like matrices [27], we know that this procedure corresponds with a step of the multishift QR -algorithm on the matrix $Z^{(1)}$.

6 Implementation and numerical tests

In this section we will first briefly discuss some issues related to the numerical implementation and secondly we will show some numerical experiments.

6.1 The implementation

The algorithm is implemented in MATLAB⁷ and can be downloaded from the author's site. In the following, we discuss several issues related to the implementation.

- **The shifts in the implementation.** The shifts considered are chosen to be the eigenvalues of the lower right $k \times k$ block, these are the generalized Rayleigh shifts.
- **The shift through lemma.** The interchanging of the order of the Givens transformations can be computed by a straightforward QR -factorization, based on 3 Givens transformations, of the involved unitary 3×3 matrix. Due to the fact that the matrix is unitary, there is flexibility in computing this factorization into 3 Givens transformations. Let us clarify what is meant with this flexibility.

Suppose a first Givens transformation is performed on the 3×3 unitary matrix U , in order, to annihilate the lower-right element of the matrix U . The Givens transformation G_1^T acts on the two last rows:

$$G_1^T U = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}.$$

As the final outcome of performing the three Givens transformations needs to be the identity matrix, this Givens transformation also has to make the upper-right 2×2 block of rank 1. This is necessary because the following Givens transformation acting on row 1 and 2 needs to create zeros in positions (2, 1), (1, 2) and (1, 3) at once. Hence performing this first Givens transformation gives us in fact the following matrix:

$$G_1^T U = \begin{bmatrix} \times & \boxtimes & \boxtimes \\ \times & \boxtimes & \boxtimes \\ 0 & \times & \times \end{bmatrix}.$$

It is clear that the first Givens transformation could be chosen in two ways: to annihilate the lower-right element or to create a rank 1 block in the upper-right position. Similar remarks hold for the remaining Givens transformations, e.g. the Givens transformation G_2^T can be chosen to annihilate one of the following three elements marked with \otimes , thereby acting on the first and second row:

$$\begin{bmatrix} \times & \boxtimes & \boxtimes \\ \otimes & \boxtimes & \boxtimes \\ 0 & \times & \times \end{bmatrix} \quad \begin{bmatrix} \times & \otimes & \boxtimes \\ \times & \boxtimes & \boxtimes \\ 0 & \times & \times \end{bmatrix} \quad \begin{bmatrix} \times & \boxtimes & \otimes \\ \times & \boxtimes & \boxtimes \\ 0 & \times & \times \end{bmatrix}.$$

The outcome of either one of the Givens transformations will be theoretically identical, hence one can choose the most numerically reliable operation.

The flexibility in computing these Givens transformations is exploited in order to make the routine as robust as possible. Details can be found in the implementation.

- **The deflation criterion.** In [26] a deflation criterion based on the norm of the off-diagonal blocks is described. This is a robust cutting criterion. In the implementation we used a faster computable deflation criterion. The only assumption for this criterion to be robust is nonsingularity of the matrix. Deflation is admitted when the following criterion is satisfied

$$\frac{|s_i|}{|c_i|} \frac{|r_{i,i}|}{|r_{i,i} + r_{i+1,i+1}|} \leq \varepsilon, \quad (7)$$

in which ε is a suitably chosen threshold. For example in the symmetric case, one can apply aggressive deflation, this means $\varepsilon = \sqrt{\varepsilon_M}$, ε_M denotes the machine precision.

- The QR -step with shifts can easily be divided into two parts. A first part in which some steps of the QR -algorithm without shift are performed, followed by a chasing technique, which assures that we performed a multishift QR -algorithm. Both steps have their specific convergence behavior. The first part, the QR -without shift, creates a convergence behavior towards the largest

⁷ MATLABR2006a, is a registered trademark of the Mathworks inc.

eigenvalues, found in the top-left position of the matrix. The second part causes a convergence towards the eigenvalues, closest to the shifts. This convergence occurs in the lower-right part of the matrix. Typically we obtain a convergence behavior as seen in the following figure (Figure 1). The figure shows a logarithmic plot of the size of the elements on the left of Equation (7). One can clearly see that there is convergence towards the top-left and convergence towards the bottom-right.

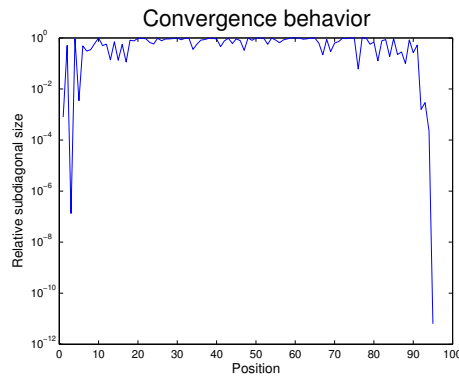


Figure 1. Typical convergence behavior.

Unfortunately this double convergence behavior, when not monitored, can considerably influence the convergence speed. It can occur that the top-left convergence has evolved so far (assume close to the machine precision), that the chasing procedure cannot get through this gap anymore. This slows down the convergence from quadratic to linear, due to the fact that the chasing and hence the QR -step with shift is not really performed, only the QR -step without shift remains.

Due to the fact that the top-left convergence proceeds linear and the bottom-right quadratically, problems can only occur when initially, before starting a multishift QR -step, the convergence in the upper-left part is much better w.r.t. the one in the lower-right part. The problem can be solved easily, when swapping the matrix if this is the case. More precisely, if the convergence on the top-left is better than the convergence in the bottom-right, the matrix is swapped. This results in a better convergence already in the bottom-right part. Due to the quadratic convergence in the bottom-right part (the shifts are chosen in this part), problems will not occur anymore. Moreover, in practice this creates an advantageous convergence behavior, as there is already convergence.

- **The number of shifts used.** Even though we can choose a fixed number of shifts in each iteration, this does not guarantee the fastest convergence in practice. A fixed upper bound on the number of shifts is initially chosen. Two criteria are used. Firstly we assume the shift to be less than $n/2$ in which n is the problem size. Secondly, based on the previous item, we know in which positions there are already small subdiagonal elements. We choose the number of shifts such that the convergence is forced to a $k \times k$ block in the bottom-right, exploiting the fact that this $k \times k$ block is already an approximation, based on the smallest found subdiagonal element.
- **The implementation is based on indexing.** This means that when deflation has occurred, indices are kept subdividing the original problem into subproblems. These subproblems are then reconsidered, until there is again deflation or their size is at most 2. The eigenvalues of problems of size at most 2 are directly computed.

6.2 Numerical experiments

In the remainder of the manuscript we considered three types of matrices. Real symmetric matrices, real matrices and complex matrices.

6.2.1 Real symmetric matrices The problems are generated by taking the lower triangular part as being equal to the lower triangular part of a rank 1 matrix $\mathbf{u}\mathbf{v}^T$. The elements of these vectors \mathbf{u} and \mathbf{v} are randomly chosen via the `randn()` function in MATLAB. Based on these vectors the Givens-vector representation of the lower triangular part is constructed.

To compare the accuracy of the proposed method with MATLAB's EIG method we computed for each eigenvalue the smallest singular value of $A - \lambda_i I$. This smallest singular value is of course a measure of the singularity and hence of the accuracy of λ_i . These values are computed for all eigenvalues computed by MATLAB and for all eigenvalues computed by the new routine (see Figure 2). The plotted results are depicted for a single shift strategy.

The figure shows that both approaches are equally accurate. Moreover the proposed method is more accurate than the results presented in [28]. Deflation was allowed, when the relative sub-diagonal elements were smaller than 10^{-10} .

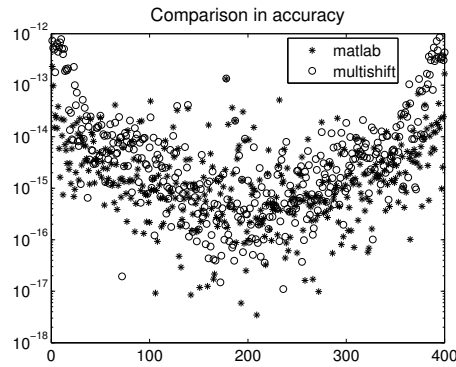


Figure 2. Accuracy of computed eigenvalues.

In the following three figures we compare the results of the multishift QR -algorithm for the single shift and the double shift strategy. The sizes range from 100 to 1000 via steps of size 100. Every experiment was repeated 5 times. The relative error, between MATLAB and the proposed method is plotted. Denote with $\tilde{\lambda}$ the eigenvalues computed via MATLAB and with $\hat{\lambda}$ the eigenvalues computed via the new technique, in the following figures the value $\|\tilde{\lambda} - \hat{\lambda}\|/\|\tilde{\lambda}\|$ is depicted. The average for the five experiments is connected via a line. The y-axis depicts the relative accuracy, whereas the x-axis depicts the size of the experiment.

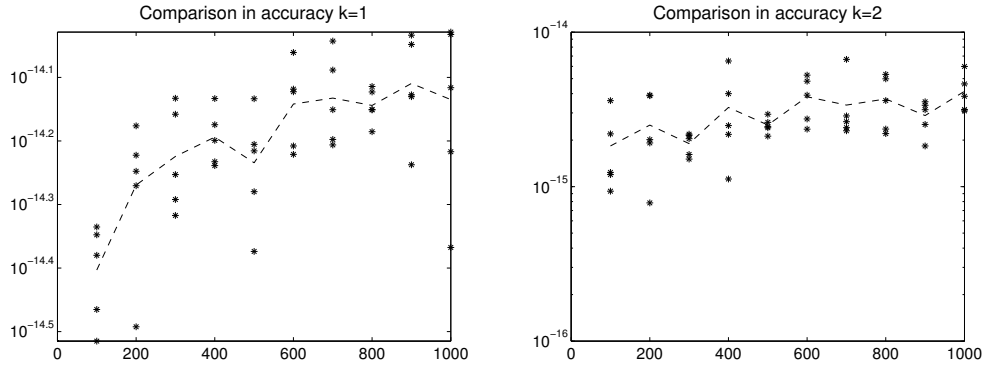


Figure 3. Comparison of different shifts.

The figures below denote for the same set of experiments the average number of QR -steps before deflation occurred. The average is computed by dividing the total number of performed QR -steps by the matrix size.

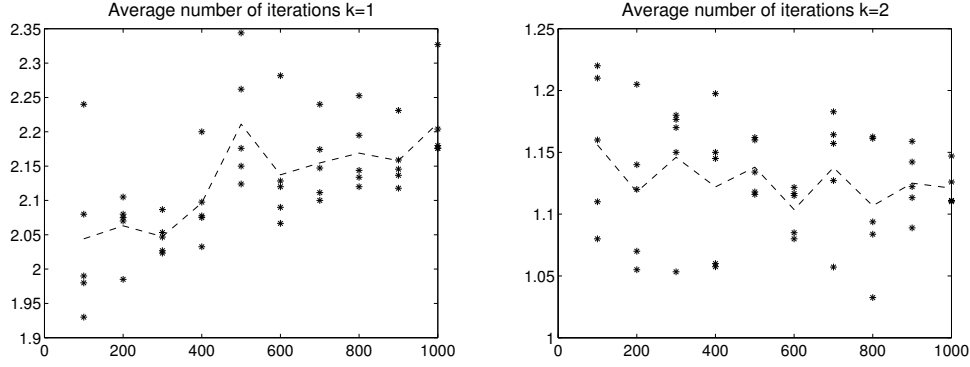


Figure 4. Number of QR -steps.

6.2.2 Real matrices The lower triangular part of these matrices is constructed similarly as in the previous examples. For the upper triangular part one has to be careful. It is known (see [22]), that a random upper triangular matrix is extremely poor conditioned. In order to obtain relatively well conditioned matrices we constructed the upper triangular part by taking the upper triangular part of the matrix R , in which $QR = A$, where A is a random constructed matrix. Experiments illustrate that the condition numbers of the corresponding eigenvalues are much better than in the previous case. Moreover in most testcases condition numbers of the eigenvalues are bound by 1000.

Especially important for real Hessenberg-like matrices is the fact that they only have real eigenvalues and complex conjugate eigenvalues. Based on a double shift strategy, we can hence restrict the computations of these eigenvalues to real operations (see [19]).

Assume we are working with the double shift strategy and the eigenvalues returned from the lower right 2×2 block Σ are complex conjugate eigenvalues σ_1 and σ_2 . Considering Equation 4, we obtain the following relation:

$$Q^{(2)H} Q^{(1)H} \tilde{Z} = \left(R^{(2)} - H_{k-1}^{(2)} \right) \left(R^{(1)} - H_k^{(1)} \right), \quad (8)$$

in which $H_k^{(i)} = Q^{(i)} \sigma_k I$, for $i = 1, 2$, $Z^{(1)} = Q^{(1)} R^{(1)}$, and $Z^{(2)} = R^{(1)} Q^{(1)} = Q^{(2)} R^{(2)}$. Rounding errors will however prevent the first column of the above equation to be real. The complex part of the first column, will however be small, w.r.t. the machine precision. Rearranging the terms in the above formula gives us the following:

$$\begin{aligned} Q^{(2)H} Q^{(1)H} \tilde{Z} &= \left(R^{(2)} - Q^{(2)H} \sigma_2 \right) \left(R^{(1)} - Q^{(1)H} \sigma_1 \right), \\ &= R^{(2)} R^{(1)} + Q^{(2)H} Q^{(1)H} \sigma_1 \sigma_2 - Q^{(2)H} R^{(1)} \sigma_1 - R^{(2)} Q^{(1)H} \sigma_2, \\ &= R^{(2)} R^{(1)} + Q^{(2)H} Q^{(1)H} \det(\Sigma) - Q^{(2)H} R^{(1)} (\sigma_1 + \sigma_2), \\ &= R^{(2)} R^{(1)} + Q^{(2)H} Q^{(1)H} \det(\Sigma) - Q^{(2)H} R^{(1)} \text{trace}(\Sigma). \end{aligned}$$

Using the first column of the equation above ensures that all computations remain in the real field.

An initial comparison of the proposed method with MATLAB's EIG is made, thereby plotting the smallest singular value of $A - \lambda_i I$ (see Figure 8). This is done similarly as in the real symmetric case. Due to computational limits (we need to compute 800 times the singular value decomposition), we only show results for a 400×400 matrix.

The figure clearly shows that both methods are equally accurate.

In the following figure a comparison in accuracy between MATLAB and the proposed method is made. Test matrices are generated, sizes ranging from 100 to 900, with stepsize 100. For each size 5 examples were run. The error plotted takes into consideration the condition number of the eigenvalues. Denote with κ_i the condition number corresponding to the eigenvalue λ_i . Let us consider the vector x and y for which

$$x_i = \left(\tilde{\lambda}_i - \hat{\lambda}_i \right) / \kappa_i$$

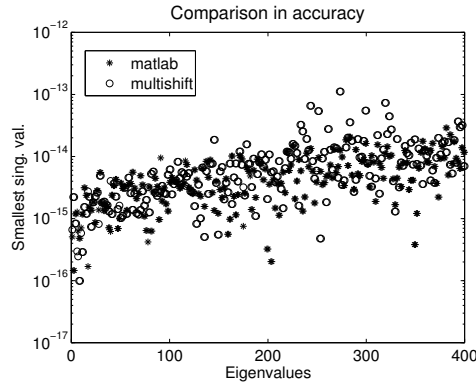


Figure 5. Accuracy of computed eigenvalues.

and

$$y_i = \left(\tilde{\lambda}_i \right) / \kappa_i.$$

The error measure considered is the following:

$$\|x\| / \|y\|.$$

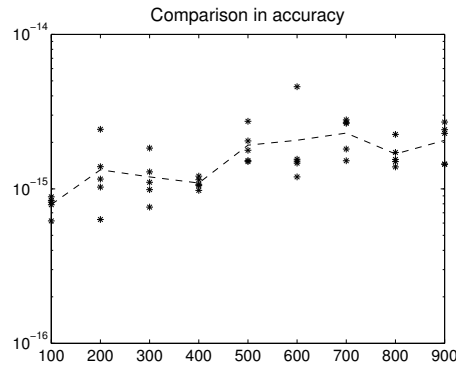


Figure 6. Accuracy of computed eigenvalues.

The error is plotted for each testcase. The average results of the five testcases are connected via a line. In the next figure (Figure 7) we show the average number of iterations for computing an eigenvalue. This is the total number of QR -steps performed divided by the number of eigenvalues. Each marker represents the results of an individual experiment, whereas the line depicts the average among all 5 experiments of the same size.

The average number of iterations is rather small, due to the fact that the QR -step also consists of performing some steps of the QR -algorithm without shift. This creates an extra convergence behavior, which is efficiently exploited in the routine.

6.2.3 Complex matrices In this subsection the considered matrices are complex Hessenberg-like matrices. The matrices are generated as in the previous example, except the matrix elements are now complex instead of real.

Similarly as in the previous examples we generated a 300×300 matrix and plotted the singular values of $A - \lambda_i$ for each eigenvalue computed by MATLAB and each eigenvalue computed by the new method. The x-axis denotes the problem sizes whereas the y-axis denotes the relative accuracy measure based on the smallest singular value, as described in the first experiment (see Figure 8).

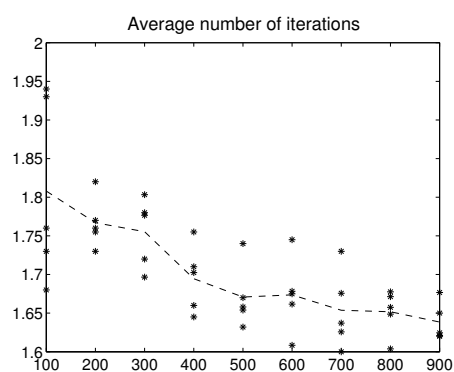


Figure 7. Average number of iterations.

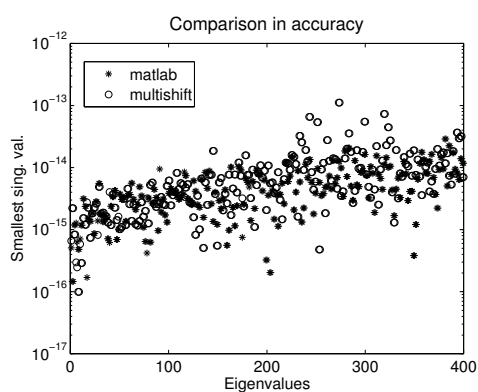


Figure 8. Accuracy of computed eigenvalues.

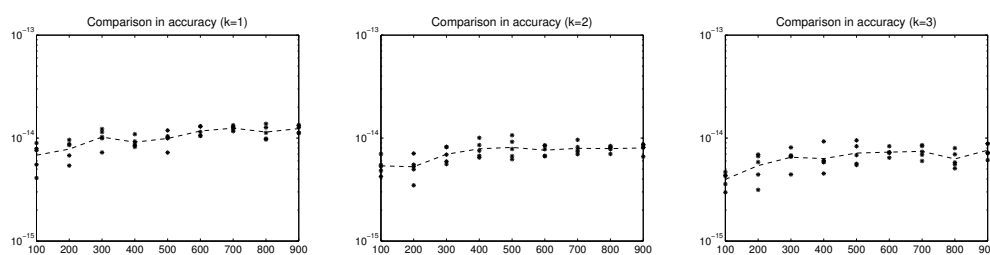


Figure 9. Accuracy of computed eigenvalues.

The following three figures (Figures 9) show, as in the other sections, the relative difference between the eigenvalues computed by the proposed method and MATLAB's eigenvalue routine.

These figures show respectively the results of the single, double and triple shift strategy. These results (Figures 10) are followed by the average number of steps of the *QR*-method before deflation of an eigenvalue can be applied in case of the single(left) double(middle) and triple(right) shift strategy.

Finally (Figures 11) we plotted the total cputime (in seconds), divided by n^3 . As the concerned method is cubic in n , this value should converge to a constant.

These last figures suggest that the multishift with $k = 2$ yields the best result. The accuracy is comparable, but the average number of iterations and the cputime show that the multishift $k = 2$ converges the fastest.

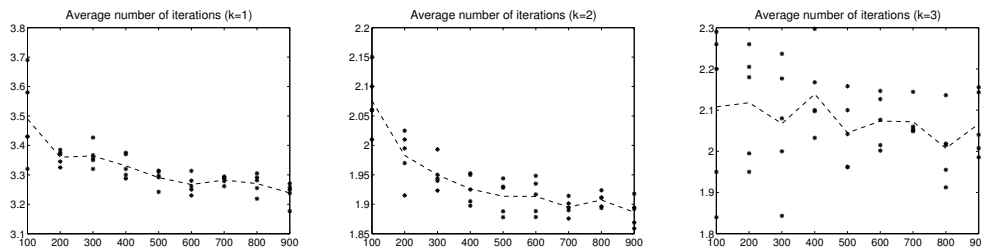


Figure 10. Average number of iterations.

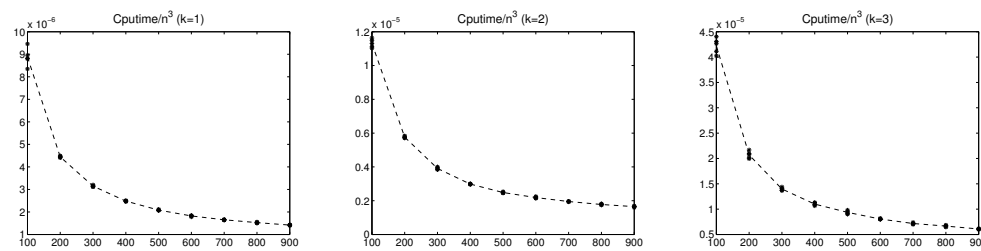


Figure 11. Timings of the method.

7 Conclusions

In this manuscript a multishift implementation of the QR -algorithm for Hessenberg-like matrices was developed. The algorithm can be subdivided into two parts: a first part performing QR -steps without shift and a second part consisting of the chasing. These two parts create two types of convergence behavior which need to be monitored. The final numerical experiments show that the developed technique is numerically reliable.

References

1. S. O. Asplund. Finite boundary value problems solved by Green's matrix. *Mathematica Scandinavica*, 7:49–56, 1959.
2. 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.
3. 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.
4. D. A. Bini, L. Gemignani, and V. Y. Pan. QR -like algorithms for generalized semiseparable matrices. Technical Report 1470, Department of Mathematics, University of Pisa, Largo Bruno Pontecorvo 5, 56127 Pisa, Italy, 2004.
5. 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.
6. S. Chandrasekaran and M. Gu. A divide and conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semi-separable matrices. *Numerische Mathematik*, 96(4):723–731, February 2004.
7. S. Delvaux and M. Van Barel. Structures preserved by the QR-algorithm. *Journal of Computational and Applied Mathematics*, 187(1):29–40, 2005.
8. S. Delvaux and M. Van Barel. Eigenvalue computation for unitary rank structured matrices. Technical Report TW465, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, July 2006.
9. 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.
10. S. Delvaux and M. Van Barel. A Givens-weight representation for rank structured matrices. Technical Report TW453, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, March 2006.
11. S. Delvaux and M. Van Barel. A Hessenberg reduction algorithm for rank structured matrices. Technical Report TW460, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, May 2006.

12. S. Delvaux and M. Van Barel. Rank structures preserved by the QR-algorithm: the singular case. *Journal of Computational and Applied Mathematics*, 189:157–178, 2006.
13. S. Delvaux and M. Van Barel. Unitary rank structured matrices. Technical Report TW464, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, July 2006.
14. P. Dewilde and A.-J. van der Veen. *Time-varying systems and computations*. Kluwer Academic Publishers, Boston, June 1998.
15. Y. Eidelman, L. Gemignani, and I. C. Gohberg. On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms. *Linear Algebra and its Applications*, 2006. (To appear).
16. Y. Eidelman and I. C. Gohberg. On a new class of structured matrices. *Integral Equations and Operator Theory*, 34:293–324, 1999.
17. Y. Eidelman, I. C. Gohberg, and V. Olshevsky. Eigenstructure of order-one-quasiseparable matrices. three-term and two-term recurrence relations. *Linear Algebra and its Applications*, 405:1–40, 2005.
18. 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.
19. G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
20. N. Mastronardi, E. Van Camp, and M. Van Barel. Divide and conquer type algorithms for computing the eigendecomposition of diagonal plus semiseparable matrices. *Numerical Algorithms*, 39(4):379–398, 2005.
21. B. N. Parlett. *The Symmetric Eigenvalue Problem*, volume 20 of *Classics in Applied Mathematics*. SIAM, Philadelphia, 1998.
22. L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.
23. M. Van Barel, R. Vandebril, and N. Mastronardi. An orthogonal similarity reduction of a matrix into semiseparable form. *SIAM Journal on Matrix Analysis and its Applications*, 27(1):176–197, 2005.
24. E. Van Camp, N. Mastronardi, and M. Van Barel. Two fast algorithms for solving diagonal-plus-semiseparable linear systems. *Journal of Computational and Applied Mathematics*, 164-165:731–747, 2004.
25. Y. Vanberghen, R. Vandebril, and M. Van Barel. A *QZ*-algorithm for semiseparable matrices. Technical Report TW471, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven (Heverlee), Belgium, October 2006.
26. R. Vandebril. *Semiseparable matrices and the symmetric eigenvalue problem*. PhD thesis, Dept. of Computer Science, K.U.Leuven, Celestijnenlaan 200A, 3000 Leuven, May 2004.
27. R. Vandebril, M. Van Barel, and N. Mastronardi. An implicit Q theorem for Hessenberg-like matrices. *Mediterranean Journal of Mathematics*, 2:59–275, 2005.
28. 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.
29. 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, October 2005.
30. R. Vandebril, M. Van Barel, and N. Mastronardi. A parallel QR-factorization/solver of structured rank matrices. Technical Report TW474, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, October 2006.
31. D. S. Watkins and L. Elsner. Chasing algorithms for the eigenvalue problem. *SIAM Journal on Matrix Analysis and its Applications*, 12(2):374–384, April 1991.