

Eigenvalue computation for unitary rank structured matrices

Steven Delvaux Marc Van Barel

Report TW 465, July 2006



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Eigenvalue computation for unitary rank structured matrices

Steven Delvaux *Marc Van Barel*

Report TW465, July 2006

Department of Computer Science, K.U.Leuven

Abstract

In this paper we describe how to compute the eigenvalues of a unitary rank structured matrix in two steps. First we perform a reduction of the given matrix into Hessenberg form, next we compute the eigenvalues of this resulting Hessenberg matrix via an implicit QR-algorithm. Along the way, we explain how the knowledge of a certain 'shift' correction term to the structure can be used to speed up the QR-algorithm for unitary Hessenberg matrices, and how this observation was implicitly used in a paper due to William B. Gragg. We also treat an analogue of this observation in the Hermitian tridiagonal case.

Keywords : unitary matrix, rank structured matrix, eigenvalue computation, pull-through operation.

AMS(MOS) Classification : Primary : 65F15, Secondary : 65F25,15A03.

Eigenvalue computation for unitary rank structured matrices

Steven Delvaux *, Marc Van Barel *

27th July 2006

Abstract

In this paper we describe how to compute the eigenvalues of a unitary rank structured matrix in two steps. First we perform a reduction of the given matrix into Hessenberg form, next we compute the eigenvalues of this resulting Hessenberg matrix via an implicit QR-algorithm. Along the way, we explain how the knowledge of a certain ‘shift’ correction term to the structure can be used to speed up the QR-algorithm for unitary Hessenberg matrices, and how this observation was implicitly used in a paper due to William B. Gragg. We also treat an analogue of this observation in the Hermitian tridiagonal case.

Keywords: unitary matrix, rank structured matrix, eigenvalue computation, pull-through operation.

AMS subject classifications: 65F15,65F25,15A03.

1 Introduction

This paper concerns the eigenvalue computation for unitary rank structured matrices. To be able to devise efficient algorithms for this purpose, a first question to be addressed is the efficient representation of unitary rank structured matrices. To this end, we will use a representation based on a product of unitary or Givens transformations [15]. This unitary/Givens product representation is a

*Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven (Heverlee), Belgium. email: {Steven.Delvaux,Marc.VanBarel}@cs.kuleuven.be.

The research was partially supported by the Research Council K.U.Leuven, project OT/05/40 (Large rank structured matrix computations), Center of Excellence: Optimization in Engineering, 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 Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister’s Office for Science, Technology and Culture, project IUAP V-22 (Dynamical Systems and Control: Computation, Identification & Modelling). The scientific responsibility rests with the authors.

direct generalization of the so-called Schur parametrization to represent unitary Hessenberg matrices [18]; see also Chapter 14 of the book [16].

In contrast to the quasiseparable, uv -, or Givens-weight representations [16, 11] to represent rank structured matrices, the Givens product representation has the advantage that the unitarity of the matrix is an explicit part of the representation. Moreover, this representation leads to an asymptotically optimal number of $O((r+s)n)$ parameters, where r is a measure for the average semiseparability rank, where s is a measure for the distance of the low rank blocks to the main diagonal, and where n denotes the matrix size.

In the present paper, we consider the problem of eigenvalue computation for unitary rank structured matrices. We proceed by means of a two-step procedure.

The first step is to apply a reduction to Hessenberg form. The algorithm which we describe in this respect, can be considered as a kind of unitary equivalent of the corresponding algorithm for the Givens-weight representation described in [12]. The complexity will also be of the same order as in the Givens-weight case, i.e., $O((r+s)n)$ operations for the Hessenberg reduction of a single column of the matrix, and hence $O((r+s)n^2)$ operations for the global Hessenberg reduction process.

The second step is to perform the implicit QR-algorithm on the resulting unitary Hessenberg matrix. This algorithm translates to a certain chasing scheme for Givens transformations. It leads to a complexity of $O(n)$ operations for each implicit QR-step, and hence $O(n^2)$ for the global implicit QR-algorithm. Along the way, we investigate some underlying structures leading to an additional speed-up of the unitary Hessenberg implicit QR-algorithm, and we show how these were implicitly exploited in [18]. We also describe some analogous observations that can lead to a speed-up for the implicit QR-algorithm in the classical Hermitian tridiagonal case.

Several algorithms for solving the eigenvalue problem for unitary Hessenberg matrices have already been developed. In [23], Rutishauser designed an LR-iteration. Implicit QR-algorithms for unitary Hessenberg matrices were described and analyzed in [18, 24, 10, 29, 30]. In [19, 5, 6, 20], divide and conquer algorithms were constructed. Other approaches are an algorithm using two half-size singular value decompositions [1], a method involving matrix pencils [8], and a unitary equivalent of the Sturm sequence method [9].

Inverse eigenvalue problems involving unitary matrices are considered in several papers. In [3], an algorithm to construct a unitary Hessenberg matrix from spectral data is designed. The unitary Hessenberg matrix is represented by its Schur parametrization. The algorithm can be used to solve the discrete least squares approximation problem by trigonometric polynomials [22]. A generalization to unitary block-Hessenberg matrices and orthonormal polynomial vectors can be found in [25, 26, 7, 27]. The size of the blocks of the matrix determines the degree structure of the sequence of corresponding orthonormal polynomial vectors. Although the *inverse* eigenvalue problem statement is quite different from the *direct* eigenvalue problem, it turns out that the solution methods rely on chasing procedures in terms of Givens transformations which are mathematically equivalent to the chasing techniques for the direct eigenproblem. Other

applications of unitary Hessenberg matrices include computations involving orthogonal polynomials on the unit circle (Szegő polynomials), the construction of Gaussian quadrature on the unit circle, frequency estimation, ... [2, 4].

The remainder of this paper is organized as follows. Section 2 recalls the main ideas for building a unitary/Givens product representation from [15]. Section 3 describes the reduction of a unitary rank structured matrix to Hessenberg form. Section 4 deals with an implicit QR-algorithm for the resulting unitary Hessenberg matrix. Finally, Section 5 reports on the results of some numerical experiments.

2 Representation for unitary rank structured matrices

In this section we recall the main ideas of building a representation for unitary rank structured matrices from [15].

Let us first define the rank structures which will be of interest in this paper.

Definition 1 (See [13]:) *We define a rank structure \mathcal{R} on $\mathbb{C}^{n \times n}$ as a collection of so-called structure blocks $\mathcal{R} = \{\mathcal{B}_k\}_k$. Each structure block \mathcal{B}_k is characterized as a 4-tuple*

$$\mathcal{B}_k = (i_k, j_k, r_k, \lambda_k),$$

where i_k is the row index, j_k the column index, r_k the rank upper bound and $\lambda_k \in \mathbb{C}$ is called the shift element. We say a matrix $A \in \mathbb{C}^{n \times n}$ to satisfy the rank structure \mathcal{R} if for each k ,

$$\text{Rank}A_k(i_k : n, 1 : j_k) \leq r_k, \quad \text{where } A_k = A - \lambda_k I.$$

Thus after subtracting the shift element λ_k from the diagonal entries, we must get a low rank block.

As a special case, when a structure block \mathcal{B}_k has shift element equal to zero, or when it is situated strictly below the main diagonal, then we call it pure. We sometimes denote such a structure block by $\mathcal{B}_{\text{pure},k}$.

Figure 1 shows an example with two structure blocks.

In what follows, we will often use unitary transformations which are *localized* in the sense that they equal the identity matrix, except for a set of subsequent rows and columns i, \dots, j . Sometimes we will explicitly denote such a localized unitary transformation as $U_{i, \dots, j}$. More explicitly, such a transformation can be written in block diagonal form as $I \oplus U \oplus I$, where I denotes the identity matrix of appropriate size.

Let us now consider a unitary rank structured matrix. We will assume that the rank structure is *pure*. The process of building a suitable representation for such a matrix is based on performing a QR-factorization of this matrix, where the Q-factor can be obtained as a sparse product of localized unitary

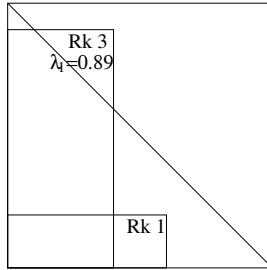


Figure 1: Example of a rank structure with two structure blocks. The left structure block \mathcal{B}_1 intersects the diagonal and has shift $\lambda_1 = 0.89$, while the second structure block \mathcal{B}_2 is ‘pure’. The notation ‘Rk r ’ denotes that the structure block is of rank at most r .

transformations, due to the presence of the rank structure, and where the R-factor can be chosen to be the identity matrix, due to the unitarity of the matrix. This construction process of the QR-factorization is illustrated in Figure 2. Here the fat vertical line segments denote the localized unitary operations applied during the compression process. More details can be found in [15].

At the end of this process, one obtains the ‘ Λ -shaped’ unitary product representation for the Q-factor of the QR-factorization, and hence for the given unitary rank structured matrix itself. This is illustrated in Figure 3.

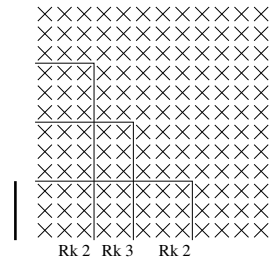
Note that the unitary product representation consists of two *branches*. The left branch contains the Hermitian transposes of the localized unitary operations serving to transform the structure blocks of the rank structure into blocks of zeros, except for their top r_k rows. On the other hand, the right branch contains the Hermitian transposes of the operations serving to annihilate the remaining subdiagonal elements (cf. the origin of these two branches in Figure 2).

In what follows, we will use the term *Givens transformation* to denote a localized unitary operation which differs from the identity matrix only in two subsequent rows i and $i + 1$. This transformation will sometimes be denoted as $G_{i,i+1}$, and the index i will be called the *row index* of the Givens transformation.

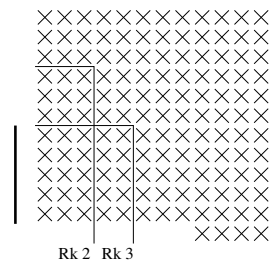
One can now specify the unitary product representation of Figure 3 to individual Givens transformations, by factoring each localized unitary component as a product of Givens transformations: see Figure 4.

Concerning this figure, we point out that each little line segment in this figure represents a Givens transformation $G_{i,i+1}$. In fact we consider each Givens transformation as ‘acting’ on the rows of an (invisible) matrix standing on the right of it, and so the Givens transformations in Figure 4 should be evaluated from right to left. The row index i of the Givens transformation $G_{i,i+1}$ can be derived from the height at which the Givens transformation is situated in the figure.

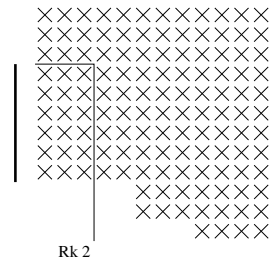
Note that factoring each unitary component of the unitary product representation as in Figure 4, may lead to a superfluous amount of Givens trans-



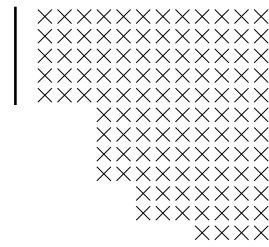
(a)



(b)

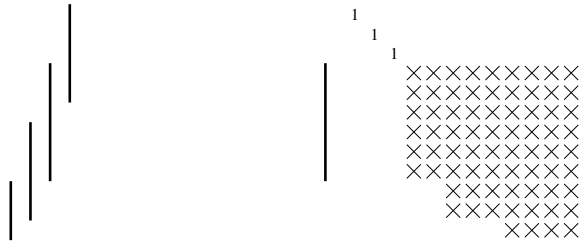


(c)

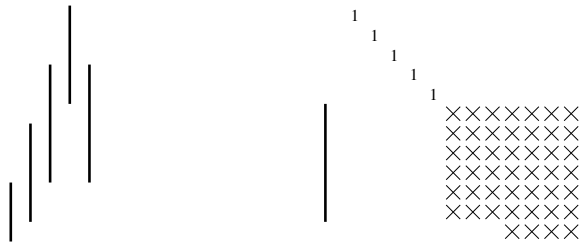


(d)

Figure 2: Building the unitary product representation (a-d).



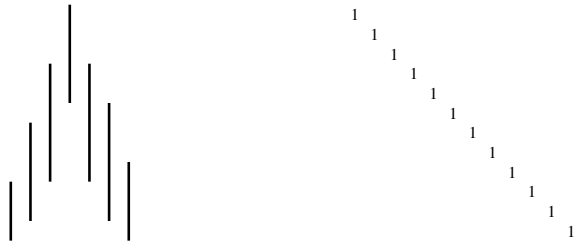
(e)



(f)



(g)



(h)

Figure 2: Building the unitary product representation (e-h).

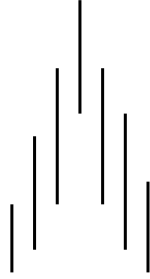


Figure 3: Final unitary product representation.

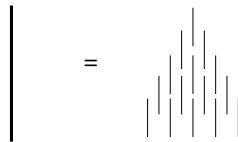


Figure 4: Givens factorization of a localized unitary component of the unitary product representation. Note that each localized unitary matrix allows such a factorization.

formations, in the sense that the beginning and ending Givens transformations of two subsequent unitary factors may overlap. To remove these superfluous Givens transformations, one can use the following lemma.

Lemma 2 (*Pull-through lemma:*) *Given a unitary 3 by 3 matrix Q which is factorized as*

$$Q = G'_{1,2}G_{2,3}G_{1,2},$$

then there exists a refactorization

$$Q = \tilde{G}'_{2,3}\tilde{G}_{1,2}\tilde{G}_{2,3}.$$

See Figure 5.

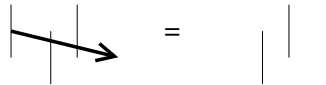


Figure 5: Pull-through lemma applied in the downward direction. One could imagine that the leftmost Givens transformation is really ‘pulled through’ the two rightmost Givens transformations.

Lemma 2 is part of mathematical folklore. It appears e.g. in implicit form in [18] and in explicit form in [26]; the graphical formulation given here is based on [11]. We will return to the pull-through lemma and some details of its implementation in Section 4.

Applying the pull-through lemma to remove the superfluous Givens transformations in the Givens product representation, leads to what we call a *zero-creating* Givens product representation for the given unitary rank structured matrix, as demonstrated in Figure 6.

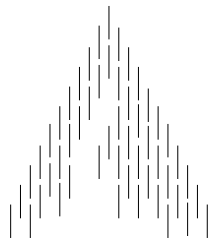


Figure 6: Representation in terms of individual Givens transformations; zero-creating version. Compare with Figure 3.

Note that the Givens transformations of the Givens product representation in Figure 6 can be ordered in a natural way in a two-dimensional array, with

the topmost Givens transformations forming a very fine-grained Λ -shape. We say that the representation is in *zero-creating* form. See for more details in [15].

Note that the width of the left branch of the Givens product representation in Figure 6 reflects information about the *ranks* of the rank structure, by the fact that these are the Hermitian transposes of the operations serving to transform the structure blocks of the rank structure into blocks of zeros, except for their top r_k rows. On the other hand, the width of the right branch reflects information on both the ranks and the distance of the structure blocks to the main diagonal.

For the aim of this paper, we will suffice with a weaker notion.

Definition 3 (*Weakly zero-creating:*) *A Givens product representation for a unitary rank structured matrix $Q \in \mathbb{C}^{n \times n}$ is called weakly zero-creating if it is given in the form*

$$Q = G_{n-1,n} \dots G_{1,2} \tilde{Q},$$

where \tilde{Q} contains a product of Givens transformations $\tilde{G}_{i,i+1}$ with $i \neq 1$.

In other words, a Givens product representation is weakly zero-creating if the left outermost series of Givens transformations in the left branch of the representation is monotonically increasing, and if all the other Givens transformations act on rows 2, 3, \dots . It is trivial to see that this implies the left outermost sequence of Givens transformations to contain precisely the Hermitian transposes of the Givens transformations bringing the first column of the unitary rank structured matrix in upper triangular form. See also [12] for a similar definition in case of the Givens-weight representation.

In what follows, we will now start to use the Givens product representation as described above for a 2-step procedure for eigenvalue computation. These matters are taken up in the following two sections.

3 Hessenberg reduction algorithm

In this section we describe an algorithm to bring the Givens product representation in Hessenberg form by means of a unitary similarity operation. The algorithm will be basically the unitary equivalent of the Givens-weight algorithm in [12], which will drastically simplify here since we have no ‘weight matrix’ to keep track of. We describe also a reduction to diagonal plus semiseparable form.

3.1 Hessenberg reduction

We will first focus on the Hessenberg reduction. We assume in what follows that the Hessenberg reduction process is based on Givens transformations, i.e., that in the k th step of this process, $k = 1, \dots, n - 1$, the k th column of the given unitary matrix $Q \in \mathbb{C}^{n \times n}$ is brought in Hessenberg form by an upward pointing sequence of Givens transformations $G_{n-1,n}, \dots, G_{k+1,k+2}$. In order to preserve the eigenvalue spectrum, we multiply then with the Hermitian transposes of these Givens transformations $G_{n-1,n}^H, \dots, G_{k+1,k+2}^H$ to the columns, hereby not

destroying the created zeros anymore. This process is illustrated for a 4 by 4 example in Figure 7.

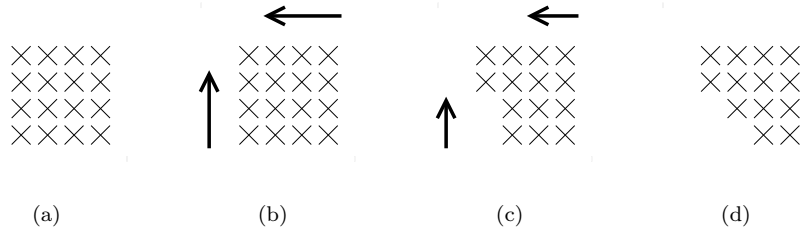


Figure 7: Hessenberg reduction of a 4 by 4 matrix.

We consider the structure propagation at the end of the Hessenberg reduction of the first column.

Theorem 4 (*Structure propagation; see [12]:*) *Let $A \in \mathbb{C}^{n \times n}$ be a matrix satisfying a structure block $\mathcal{B} = (i, j, r, \lambda)$ with $i \geq 2$, such that the first column of $A|_{\mathcal{B}}$ is not equal to the zero vector. (Here $A|_{\mathcal{B}}$ denotes the submatrix of A which is ‘cut out’ by \mathcal{B}). Then by reducing the first column of A into Hessenberg form by means of Givens transformations, the structure block \mathcal{B} moves one position to the bottom right position, i.e., it transforms into a new structure block $\tilde{\mathcal{B}} = (i + 1, j + 1, r, \lambda)$: see Figure 8.*

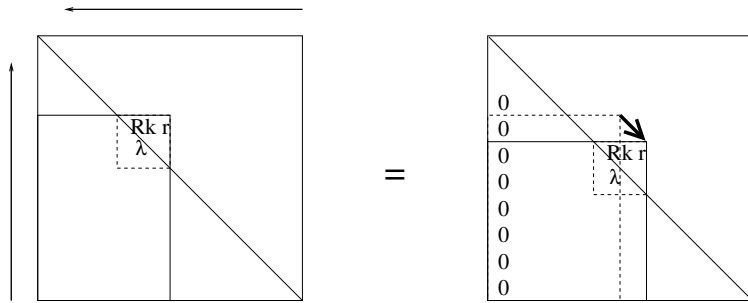


Figure 8: Structure propagation after the Hessenberg reduction of the first column.

Theorem 4 opens the door for an efficient Hessenberg reduction process, in which the given rank structure is exploited throughout the algorithm.

First, we should find out how to determine the Givens transformations used to create zeros in the first column of the given unitary rank structured matrix Q . This is where our assumption on the (weakly) zero-creating form of the Givens

product representation (Definition 3) comes to help: it was already mentioned before that the Givens transformations used to create zeros in the first column, can just be read off from such a representation. Indeed, these transformations are the Hermitian transposes of the left outermost Givens transformations of the weakly zero-creating Givens product representation: the latter are highlighted in Figure 9(b).

We have now all ingredients needed for bringing the first column of the unitary rank structured matrix Q into Hessenberg form. The algorithm is illustrated in Figure 9.

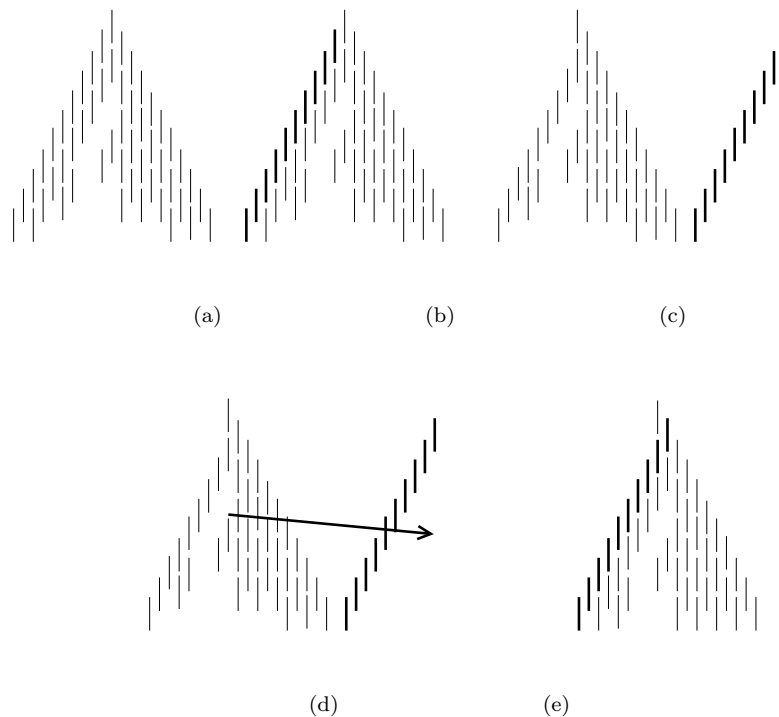


Figure 9: Hessenberg reduction of the first column.

Let us comment on this figure. The algorithm starts by applying the Givens transformations creating zeros in the first column of the matrix Q . By the remarks above, these are precisely the Hermitian transposes of the outermost sequence of Givens transformations $G_{n-1,n}^H, \dots, G_{2,3}^H$, and hence they have already been precomputed in the representation: see Figure 9(b).

By multiplying with the Givens transformations $G_{n-1,n}^H, \dots, G_{2,3}^H$ to the rows, these Givens transformations will be *peeled off*, i.e., they will disappear from the representation.

To complete the similarity, we should then multiply with the Hermitian transposes of the row operations $G_{n-1,n}, \dots, G_{2,3}$ to the columns. This will cause the peeled off Givens transformations to reappear on the outer right part of the representation: see Figure 9(c).

It suffices then to bring the Givens product representation back to its (weakly) zero-creating form. This can be achieved by applying the pull-through lemma a maximal number of times in the downward direction. For practical reasons, one can best do this by considering the sequence of peeled off Givens transformations to form a chain of *fixed shape*: these are the highlighted Givens transformations in Figure 9(c). One can then pull what is left of the original Givens product representation, rightwards through this chain of Givens transformations, one Givens transformation after the other: see Figure 9(d).

Note that the pull-through process causes almost each Givens transformation of the unitary product representation to arrive one place *lower* than in its original position: see Figure 9(e). Moreover, the Givens transformations acting on the bottommost rows $n-1, n$ do not ‘survive’ the pull-through process, in the sense that, each time that such a Givens transformation must be pulled through the highlighted chain of Givens transformations, it can be just absorbed in the bottommost Givens transformation $G_{n-1,n}$ of this chain. In fact, the above observations reflect the fact that each of the structure blocks of the rank structure moves towards the bottom right corner during the Hessenberg reduction of the first column, as detailed in Figure 8.

There is one Givens transformation which is left unaltered by the above pull-through process: this is the top rightmost Givens transformation $G_{1,2}$ of the representation, since the latter is positioned too ‘high’ for being able to pull it through the highlighted chain of Givens transformations. This is the reason why this Givens transformation is still standing isolated on the left of the highlighted chain of Givens transformations in Figure 9(e).

We have now completed the Hessenberg reduction of the first column. It is clear that this process has complexity $O((r+s)n)$, since each of the $O((r+s)n)$ Givens transformations of the Givens product representation has undergone exactly one pull-through operation.

The technique as just described can now be iterated to bring also the next columns in Hessenberg form. Some steps in this process are shown in Figure 10.

It can be clearly observed from Figure 10 that the Givens transformations, and hence the underlying structure blocks, move indeed towards the bottom right corner of the matrix. Moreover, note that the total algorithm complexity of the Hessenberg reduction process is $O((r+s)n^2)$ operations.

The unitary Hessenberg matrix obtained at the end of the entire reduction process is shown in Figure 11. It remains then to compute the eigenvalues of this resulting unitary Hessenberg matrix, which will be the topic of Section 4.

3.2 Reduction to diagonal plus semiseparable form

This subsection describes an analogue of the unitary Hessenberg reduction, namely, a reduction into *lower semiseparable plus diagonal* form.

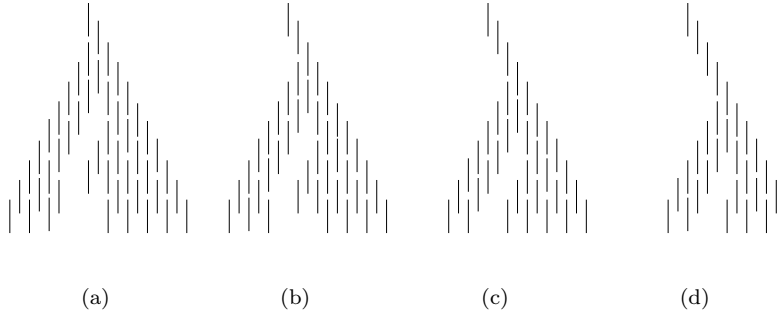


Figure 10: Hessenberg reduction of the next columns.

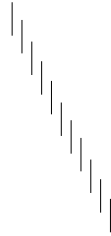


Figure 11: Final unitary Hessenberg matrix resulting at the end of the Hessenberg reduction process.

We define the class of lower semiseparable plus diagonal matrices by means of the structure blocks

$$\mathcal{B}_k : (i_k, j_k, r_k, \lambda_k) = (k, k, 1, \lambda_k),$$

where $k = 2, 3, \dots, n - 1$. Note that the shift elements λ_k are part of the structure. Formally, the class of Hessenberg matrices can also be described in this format, by choosing each of the λ_k equal to ∞ , in the sense of [14].

The reduction of a tridiagonal matrix to diagonal plus semiseparable form is described in [28]. Since we assume here the given matrix to satisfy an arbitrary rank structure, we will follow the description of [12].

The idea of the reduction process is the same as for the Hessenberg reduction: we proceed upwards by creating zeros in the first column of the matrix. During this process, each of the present structure blocks will be propagated towards the bottom right corner of the matrix, in much the same way as in Theorem 4.

The difference with the Hessenberg reduction is that we proceed in each step *completely* to the top of the matrix. Moreover, at the end of each step, the determination of the top Givens transformation $G_{1,2}$ to be applied depends on

the new shift element λ_{n-k} of the diagonal plus semiseparable structure that one would like to install. See Figure 12.

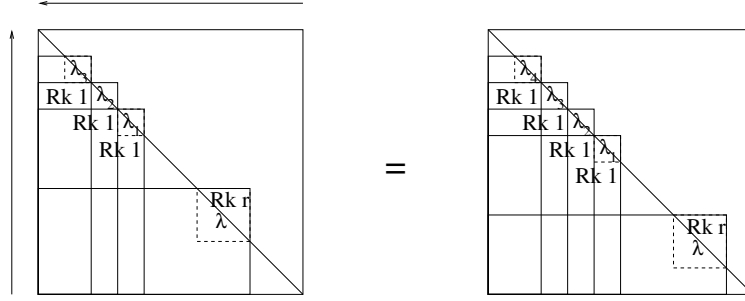


Figure 12: Structure propagation in one step of the diagonal plus semiseparable reduction. Compare with Figure 8.

For ease of reference, let us repeat here the procedure for computing the topmost Givens transformation $G_{1,2}$. Thus suppose that we have already applied the Givens transformations $G_{n-1,n}, \dots, G_{2,3}$ to the rows, and their Hermitian transposes to the columns, resulting in the first column being in Hessenberg form:

$$\begin{bmatrix} a & \times & \dots & \times \\ b & \times & \dots & \times \\ 0 & \times & \dots & \times \\ \vdots & & & \vdots \\ 0 & \times & \dots & \times \end{bmatrix},$$

where the irrelevant matrix entries are denoted as \times . Let us now rewrite this matrix as

$$\begin{bmatrix} a - \lambda & \times & \dots & \times \\ b & \times - \lambda & \dots & \times \\ 0 & \times & \dots & \times \\ \vdots & & & \vdots \\ 0 & \times & \dots & \times \end{bmatrix} + \begin{bmatrix} \lambda & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}, \quad (1)$$

where λ denotes the new shift element that we would like to install. Now it is clear that the second term of (1) will be preserved under the similarity transformation with the topmost Givens transformation $G_{1,2}$. Therefore, if we choose $G_{1,2}$ such that it creates a zero in the (2, 1) position of the first term of (1), i.e.,

$$G_{1,2} \begin{bmatrix} a - \lambda \\ b \end{bmatrix} = \begin{bmatrix} \times \\ 0 \end{bmatrix}, \quad (2)$$

we will obtain the desired diagonal plus semiseparable structure block $\mathcal{B} : (i, j, r, \lambda) = (2, 2, 1, \lambda)$. See also [28, 12].

Let us now exploit the above ideas and perform the diagonal plus semiseparable reduction of a unitary rank structured matrix in a practical way. Due to the unitary product representation, we are forced to describe the algorithm in a *pure* form, i.e., we represent the diagonal plus semiseparable matrix by means of its induced pure rank structure, situated just below the main diagonal:

$$\mathcal{B}_{k,\text{pure}} : (i_k, j_k, r_k) = (k, k - 1, 1),$$

where $k = 3, 4, \dots, n - 1$.

The installation of the first diagonal plus semiseparable structure block is illustrated in Figure 13.

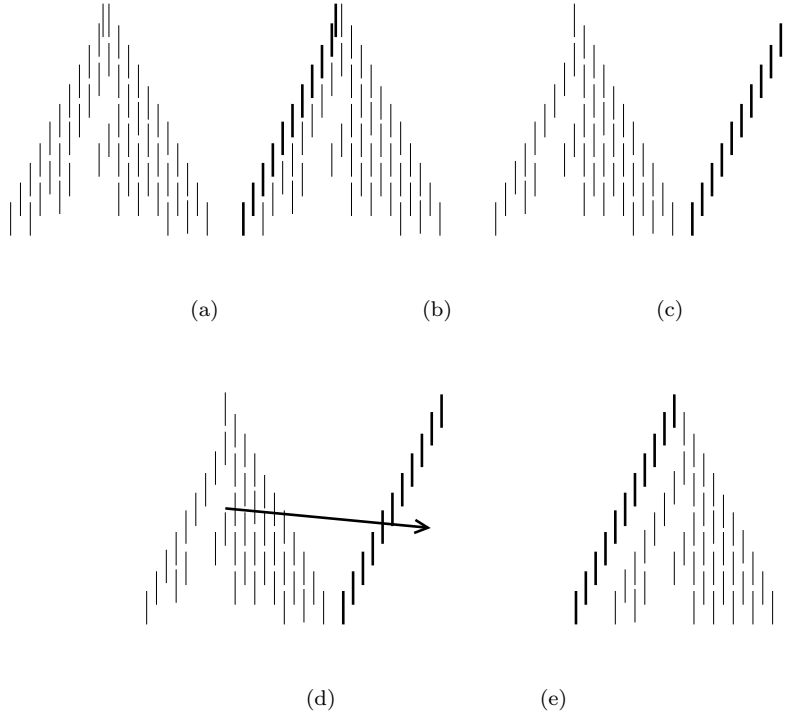


Figure 13: Diagonal plus semiseparable reduction of the first column.

Let us comment on this figure. The algorithm is very similar to the reduction to unitary Hessenberg form. The difference is that each Givens sequence to be peeled off goes from $G_{n-1,n}$ to $G_{1,2}$, hence completely up to the top of the matrix at each step of the algorithm. Only the topmost Givens transformation $G_{1,2}$ can be chosen freely, according to the new shift element of the diagonal plus semiseparable structure that one would like to install: this is indicated by the duplication of the top left lines in Figures 13(a) and 13(b).

The applied Givens transformations are then again peeled off and transported rightwards in Figure 13(c). Having done this, the way how to restore the (weakly) zero-creating form of the unitary product representation is similar to the Hessenberg case: see Figures 13(d) and 13(e). This ends the installation of the first diagonal plus semiseparable structure block.

We can then again iterate this procedure. Some steps in this process are shown in Figure 14. The diagonal plus semiseparable matrix resulting at the very end of this process is shown in Figure 15. Note that the reduction algorithm has the same asymptotic complexity of $O((r+s)n^2)$ operations as in case of the Hessenberg reduction.

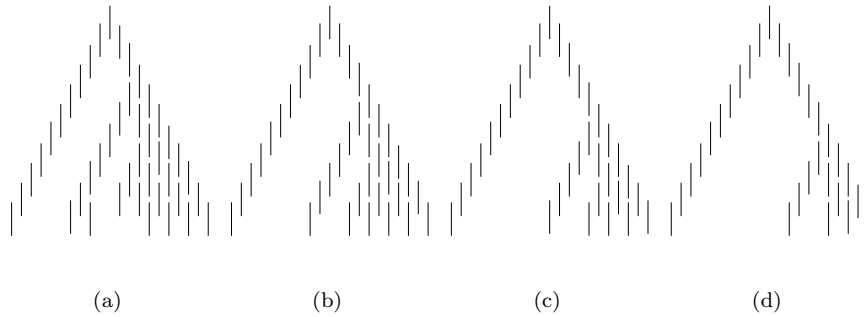


Figure 14: Diagonal plus semiseparable reduction of the next columns.

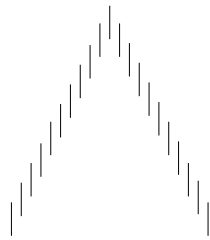


Figure 15: Final unitary diagonal plus semiseparable matrix resulting at the end of the reduction process.

We recall that the Hessenberg reduction can formally be seen as a special case of the diagonal plus semiseparable reduction when the shift elements tend to ∞ .

Finally, let us consider the shift elements. It was already indicated in Figure 12 that these shift elements move downwards along with the corresponding structure blocks. We pose ourselves now the question how this behavior can be

read off in terms of the Givens product representation.

To this end, we recall from [15] that the k th shift element can be expressed as the quotient of the sines of the corresponding, opposite pair of Givens transformations with row index k , provided that the sines are defined in an appropriate way (See also Section 4). This property is illustrated in the left part of Figure 16.

Now after peeling off this leftmost Givens sequence and transporting it to the right, the usual ‘ Λ -shaped’ form of the representation will transform into a V-shaped representation. In the latter case, exactly the same property about the quotient of sines holds true, except that the position of the corresponding shift element is now *one position lower*, in the direction of the central point of the V-shape [15]: see Figure 16. Hence, we see indeed that the transmission of the shift elements is reflected in the representation.

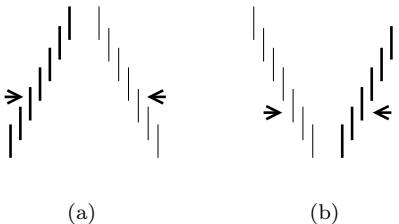


Figure 16: Transmission of the shift elements during the diagonal plus semiseparable reduction.

Concerning Figure 16, note that initially, the fifth shift element λ_5 can be expressed as the quotient of the sines of the two indicated Givens transformations in the left figure. After transporting the complete left branch of Givens transformations to the right of the representation, as in the right part of Figure 16, this same quotient of sines describes now the *sixth* shift element λ_6 of the structure, corresponding to the fact that the shift elements are all transported to the bottom right.

Summarized, we have now described the reduction of an arbitrary unitary rank structured matrix to Hessenberg or diagonal plus semiseparable form. The computation of the eigenvalues in the resulting Hessenberg case will be the topic of the next section.

4 Implicit QR-algorithm

In this section we describe an implicit QR-algorithm for unitary Hessenberg matrices. Some classical references where such an algorithm is described and analyzed are [18, 24], but we will proceed here in a more conceptual way w.r.t. these papers. We will consider also some analogous observations in the classical

Hermitian tridiagonal case.

The mechanism of the bulge chasing is resumed in Figure 17.

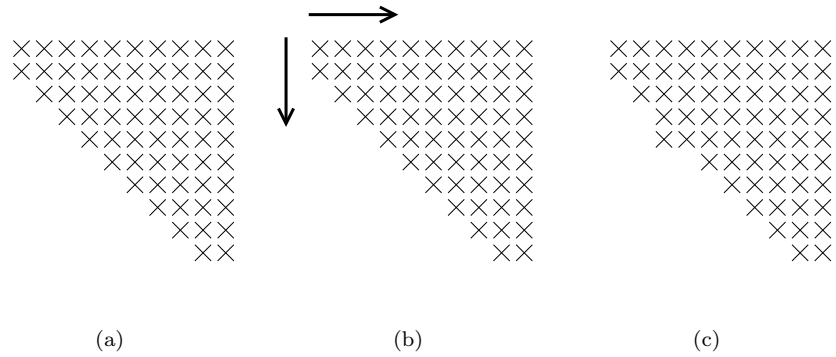


Figure 17: Bulge chasing during the implicit QR-algorithm for Hessenberg matrices.

4.1 Implicit QR-algorithm for unitary Hessenberg matrices

In this subsection we consider the implicit QR-algorithm in the unitary Hessenberg case. The algorithm is expressed in terms of the Givens product representation, and it is illustrated in Figure 18.

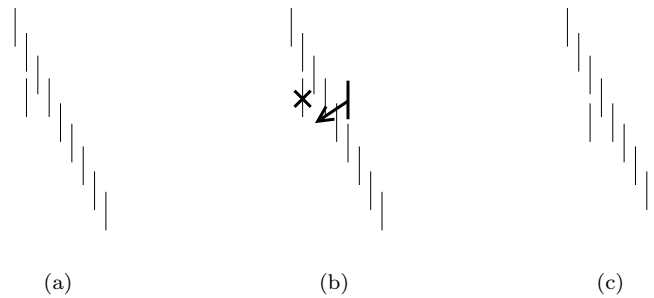


Figure 18: One step of the implicit QR-algorithm for unitary Hessenberg matrices.

Let us comment on this figure. The algorithm starts with applying a similarity operation based on a Givens transformation $G_{1,2}$, determined by the shift

of the shifted QR-algorithm. This similarity operation will create a bulge at the $(3, 1)$ position of the matrix.

Having done this initial perturbation, we should now restore the Hessenberg shape of the matrix by means of unitary similarity transformations. The tool for doing this will be nothing but the general Hessenberg reduction algorithm of the previous section, which will specialize here to a very specific chasing procedure due to the regularity of the structure.

Let us now describe this chasing process in detail. Assume that we are at the k th step of the chasing process. The unitary Hessenberg matrix will then have obtained a bulge at its $(k+1, k-1)$ position, translating in an extra Givens transformation situated at the very left of the unitary Hessenberg Givens chain: see Figure 18(a).

The next operation of the implicit QR-algorithm consists then in chasing this bulge one position towards the bottom right corner of the matrix. In terms of the Givens product representation, this means that we have to peel off the bulge Givens transformation G from the left of the representation, by multiplying with G^H on the rows, and complete the similarity operation by multiplying with G on the columns. Similarly to the previous section, one could say that the bulge Givens transformation G has been ‘transported’ from the left to the right: see Figure 18(b).

To bring the extra Givens transformation back from the right to the left, we can now apply the pull-through lemma (Lemma 2). The bulge Givens transformation appears then one position lower on the left, corresponding to the fact that the bulge has been chased one position to the bottom right: see Figure 18(c).

Continuing this process will result in the bulge being chased away towards the bottom right corner of the matrix, where it can ultimately be absorbed in the Givens transformation $G_{n-1,n}$ at the end of the last chasing step. The matrix has then reobtained its unitary Hessenberg form, and hence it will be precisely the new QR-iterate of the shifted QR-algorithm.

We believe that the above described algorithm is more or less well-known to the unitary Hessenberg matrix community, see e.g. [18, 26, 10]. Nevertheless, we are not aware of any graphical formulation in the literature as in Figure 18.

Let us now focus on some more practical aspects of the above algorithm. More precisely, we will investigate how to speed up the application of the pull-through lemma during this algorithm, to make it as efficient as possible.

One could start by considering the following, straightforward implementation of the pull-through lemma. Using the notations of Lemma 2, we start by explicitly computing the 3 by 3 matrix $Q := G'_{1,2}G_{2,3}G_{1,2}$ in its full form. Subsequently we compute Givens transformations such that

$$(\tilde{G}_{2,3})^H (\tilde{G}_{1,2})^H (\tilde{G}'_{2,3})^H G'_{1,2} G_{2,3} G_{1,2} \quad (3)$$

is upper triangular, with positive diagonal elements. Then since (3) is both upper triangular and unitary, with positive diagonal elements, it must be the identity matrix, and hence we have obtained the required refactorization of the matrix Q as a product of 3 Givens transformations.

We will now speed up the implementation of the pull-through lemma in two phases. First, one can work with Givens transformations having determinant one, i.e., Givens transformations having the form

$$\begin{bmatrix} c & s \\ -\bar{s} & \bar{c} \end{bmatrix}.$$

where c and s are complex numbers such that $|c|^2 + |s|^2 = 1$; in fact, it can even be imposed that s must be real. The key point is here that the determinant one property allows the Givens transformation to be known based on its first column (or first row) only. Moreover, since this property is maintained under multiplication, and provided that the first two pulled-through Givens transformations are also chosen of this form, the determination of the last Givens transformation, which was the most expensive operation, can now be performed based only on its first column. It turns out that this almost halves the number of operations.

Now we will explain how one can use information about the shift element to obtain a further speed-up. The first step will be the following result, essentially due to Watkins.

Theorem 5 (cf. [31]:) *Given a Hessenberg matrix, then in the k th step of the implicit QR-algorithm, the shift induces a non-pure Rk 1-structure block $\mathcal{B} : (i, j, r, \lambda) = (k, k, 1, \lambda)$, which propagates towards the bottom right corner of the matrix during the implicit QR-algorithm: see Figure 19.*

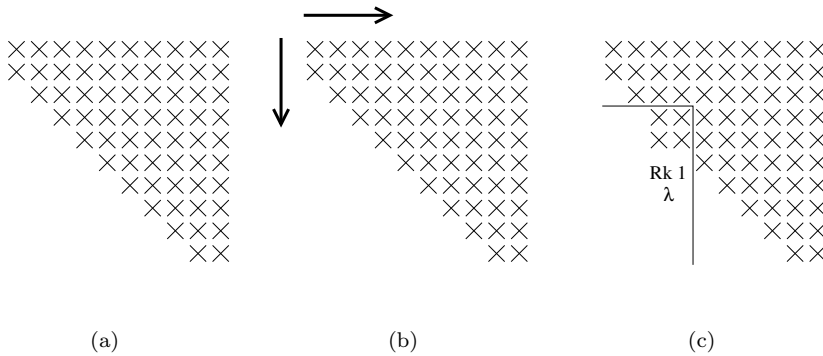


Figure 19: Transmission of the shift during the implicit QR-algorithm for Hessenberg matrices.

This result can be easily established here. We note that this result was originally stated for the multi-shift case.

Now we apply this result to speed up the implicit QR-algorithm. Recall that the global algorithm flow is shown in Figure 18. To exploit the result of

Watkins, we will need the following characterization of where the shift is in terms of the Givens product representation.

Theorem 6 (See [15]:) *The shift element equals the quotient of sines: $\lambda = \frac{s_1}{s_3}$, where*

$$s_3 \left| \begin{array}{c} | \\ | \\ | \end{array} \right| s_1$$

Here the ‘sines’ must be appropriately defined: for the rightmost Givens transformation it is defined as the $(2, 1)$ element of the matrix, while for the leftmost Givens transformation it is defined as the complex conjugate of the $(1, 2)$ element of the matrix.

Note that Theorem 6 can be used as a possible way for deriving Theorem 5 in terms of the Givens product representation, in much the same way as it was done earlier in Figure 16. (The difference compared to the latter figure is now that the left branch of the representation consists only of a single Givens transformation anymore.)

Let us now apply these results to speed up the implementation of the pull-through operations in the implicit QR-algorithm for unitary Hessenberg matrices. The implementation of a single pull-through step is recapitulated in Figure 20. We see from this figure that the determination of the last pulled through Givens transformation $(\tilde{G}_{2,3})^H$, is by far the most expensive, since it requires an accumulated keeping track of the second and/or third column of the unitary 3 by 3 matrix.

But by what we know from Theorem 6, we can relate this Givens transformation to the Givens transformation $(\tilde{G}'_{2,3})^H$, at least concerning its sine. The cosine should then still be determined directly.

Let us describe then the optimized implementation of the pull-through lemma in detail:

- Form the first column of the 3 by 3 unitary matrix Q : starting with $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, multiply rows 1, 2 with $G_{1,2}$ (this involves no floating point operations), rows 2, 3 with $G_{2,3}$ and subsequently rows 1, 2 with $G'_{1,2}$.
- Form also the third column of the matrix Q : starting with $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, multiply rows 2, 3 with $G_{2,3}$ (this involves no floating point operations), and rows 1, 2 with $G'_{1,2}$ (since only the second element will be required, the computation of the top element is in fact not needed).
- Compute the first pulled through Givens transformation $(\tilde{G}'_{2,3})^H$ such that it creates a zero in the bottom element of the first column of Q .

$$\begin{array}{|c|} \hline \times \times \times \\ \times \times \times \\ \times \times \times \\ \hline \end{array} = \begin{array}{|c|} \hline \times \times \times \\ \times \times \times \\ \times \times \times \\ \hline \end{array}$$

(a)

$$\begin{array}{|c|} \hline \times \times \times \\ \times \times \times \\ \times \times \times \\ \hline \end{array} = \begin{array}{|c|} \hline \times \times \times \\ \times \times \times \\ \times \times \times \\ \hline \end{array}$$

(b)

$$\begin{array}{|c|} \hline \times \times \times \\ \times \times \times \\ \times \times \times \\ \hline \end{array} = \begin{array}{|c|} \hline \times \times \times \\ \times \times \times \\ \times \times \times \\ \hline \end{array}$$

(c)

$$\begin{array}{|c|} \hline 1 \\ \times \times \times \\ \times \times \times \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \times \times \times \\ \times \times \times \\ \hline \end{array}$$

(d)

$$\begin{array}{|c|} \hline 1 \\ 1 \\ 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ 1 \\ 1 \\ \hline \end{array}$$

(e)

Figure 20: ‘Brute-force’ implementation of the pull-through operations in the implicit QR-algorithm for unitary Hessenberg matrices.

- Determine the second pulled through Givens transformation $(\tilde{G}_{1,2})^H$ from the information in the two top elements of the first column of Q , which should coincide with the first column of $\tilde{G}_{1,2}$.
- Compute the sine of the final pulled through Givens transformation $(\tilde{G}_{2,3})^H$ by means of Theorem 6.
- Finally, to compute the cosine of $(\tilde{G}_{2,3})^H$, update the third column of Q by applying the first pulled through operation: $(\tilde{G}'_{2,3})^H$, to rows 2,3 of this column (since only the third element will be required, the computation of the second element is in fact not needed). The bottom element of the third column of Q should then coincide with the bottom right element of $\tilde{G}_{2,3}$.

We note that the above observations for speeding up the pull-through lemma have also been used in implicit form in the implementational details of the paper by William B. Gragg [18].

One point should be addressed here. The above implementation does not guarantee that the last two pulled through Givens transformations $(\tilde{G}_{1,2})^H$, $(\tilde{G}_{2,3})^H$ are exactly unitary, i.e., the sum of squares of sine and cosine might slightly deviate from one due to roundoff errors. To remedy this, one could

combine the above implementation with an explicit *renormalization procedure*. Since this renormalization procedure increases the computational complexity of the pull-through, we advise to use it only a limited number of times during the algorithm, depending on the flow direction of the algorithm.

It should also be taken into account that the above renormalization procedure slightly changes the value of the shift element λ , since it changes the quotient of the sines during the algorithm. We could interpret this by saying that the *quality of the shift element* deteriorates during the algorithm: see also [31].

In case where the shift element λ is unimodular, the renormalization procedure may cause λ to *drift away* from unimodularity. In this case, it may be numerically advisable to implement the renormalization procedure such that this feature is explicitly preserved. For some details in this direction we refer to Stewart [24].

In concluding this subsection, let us provide here an illustration of the implicit QR-algorithm for unitary Hessenberg matrices in the case of *double shift*: see Figure 21. Although Watkins' result allows an analogue for the multiple shift case as well [31], it seems that, unfortunately, the information about the shift elements (plural, since we are applying a double shift step), can not be easily exploited in this case, so that the above speed-up of the pull-through lemma has no analogue to the double shift case.

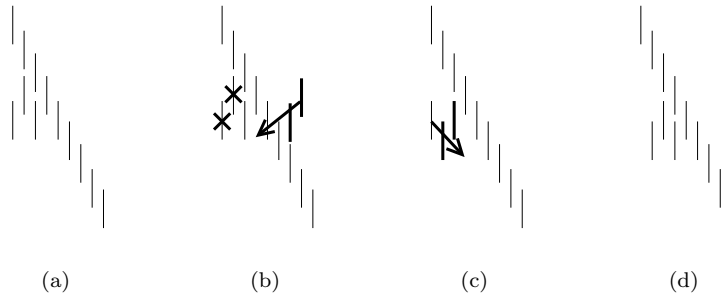


Figure 21: One step of the implicit QR-algorithm for unitary Hessenberg matrices: the case of double shift.

Finally, inspired by the analogy of the previous section, we point out that an implicit QR-algorithm could be devised also for unitary diagonal plus semiseparable matrices, rather than unitary Hessenberg matrices. This topic requires some additional concepts and chasing tools and therefore will be deferred to our future work.

4.2 Implicit QR-algorithm for Hermitian tridiagonal matrices

In this subsection we consider some analogues of the above observations for unitary Hessenberg matrices to the Hermitian tridiagonal case. More precisely, we will show how Watkins' result can be used for a speed-up of the number of operations. We refer to the books [21, 17] for some classical references on the implicit QR-algorithm in the Hermitian tridiagonal case.

First of all, since a tridiagonal matrix can be represented simply by means of its individual entries, one can freely add or subtract the shift element of the shifted QR-algorithm $\lambda \in \mathbb{C}$ from the main diagonal. Doing this in the very beginning and at the very end of the algorithm, one can immediately reduce the problem to implementing the QR-algorithm *without shift*.

We recall then that in the k th step of the chasing procedure, there is a bulge at the $(k + 1, k - 1)$ position of the tridiagonal matrix. Moreover, at the bulge there is situated a rank-one structure block: recall Figure 19 above.

Note that due to our reduction to the QR-algorithm without shift, the rank-one structure block spanning over the bulge will be a *pure* structure block. Such a pure structure block can then be represented by means of a *Givens-weight representation*: in the present context it simply amounts to applying an auxiliary Givens transformation to 'compress' this rank one block, i.e., to eliminate its bottom row. The representation for the rank-one block consists then of the Givens transformation used in this compression process, which is indicated by the little line segment in the right part of Figure 22, combined with the elements resulting on the top of the structure block at the end of the compression process. Since the latter contain a kind of 'decoded' information about the matrix, they are distinguished from the other entries by placing them on a grey background: see Figure 22.

$$\begin{array}{c} \times \times \\ \times \times \end{array} = \left| \begin{array}{c} \times \times \\ \times \times \end{array} \right.$$

Rk 1

Figure 22: Compressed representation of the rank-one structure block. The representation consists of the Givens transformation used to compress the bottom row of the rank-one block, indicated by the vertical line segment, combined with the elements obtained on the top row after applying this compression process, the latter indicated on a grey background.

The implicit QR-algorithm for a Hermitian tridiagonal matrix is then shown in Figure 23.

Let us comment on this figure. First, note that the figure shows only the lower triangular part of the tridiagonal matrix, since the upper triangular part is then known by symmetry. Since we are at the point now of applying a Givens

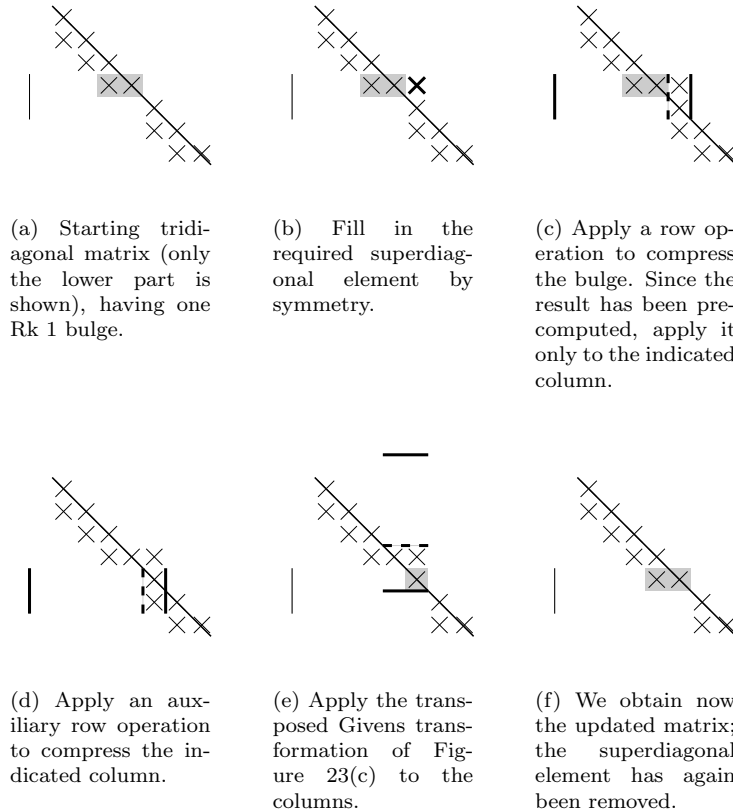


Figure 23: Implicit QR-algorithm for a Hermitian tridiagonal matrix.

transformation to rows and columns $k, k + 1$, it is now the right moment to fill in the $(k, k + 1)$ superdiagonal element by symmetry: see Figure 23(b).

We want now to apply a Givens transformation G used to compress the current rank-one ‘bulge’ structure block. Since the required Givens transformation G , as well as the action of this Givens transformation on the rank-one block have both been precomputed in the representation, we have to apply G only to the column on the right of the rank-one block: this means that it has to be applied only between the two fat vertical line segments in Figure 23(c).

Note that the Givens-weight representation, and the corresponding grey background have both disappeared after applying this Givens transformation. This is valid since at this moment, the representation does not contain any decoded information anymore, but instead just contains the real-size elements of the matrix, positioned there at this point of the algorithm.

We apply now an auxiliary row operation to compress the $(k + 2, k + 1)$ element of the matrix. Note that doing this implies building up a new Givens-weight representation: see Figure 23(d).

Having done this, we complete the similarity operation by multiplying G^H to the columns: see Figure 23(e).

The final situation is shown in Figure 23(f). Since the $(k, k+1)$ superdiagonal element is not needed anymore, it can be removed from the representation. The next operations are not shown anymore.

It can be checked that the cost of the above algorithm is *less* than if one does not exploit the presence of the rank-one structure block predicted by Watkins' result. Hence, we obtain here a speed-up compared to the straightforward implementation of the Hermitian tridiagonal implicit QR-algorithm.

Remark 7 (*Accuracy:*) *We have to warn that the above variant could possibly be less accurate than the standard bulge chasing scheme for Hermitian tridiagonal matrices, since it requires the shift of the shifted QR-algorithm to be explicitly added and subtracted from the diagonal entries. This might possibly give rise to numerical problems in case of a large shift. Note that this problem does not arise in the unitary case, since we can assume there that the shifts are chosen unimodular all the time, i.e., that $|\lambda| = 1$ during each QR-step on a unitary matrix, and moreover, in the unitary case, the shift is not explicitly added or subtracted from the representation.*

5 Numerical experiments

We report now on the results of some numerical experiments. The algorithms were implemented in Matlab*. The experiments were executed on an Intel PC running Matlab Version 7.0.1.24704 (R14) under Linux having 1GByte of memory and an Intel Pentium 4 processor running at 3.2 GHz. The software of these experiments can be requested from the authors.

We constructed unitary rank structured matrices of sizes $n = 2^k$ for $k = 4, 5, \dots, 12$. The exact values of the eigenvalues $\lambda_{\text{exact},i}$ were generated uniformly random on the unit circle. We considered then the unitary diagonal matrix D containing these eigenvalues on the main diagonal. Subsequently, we applied to the matrix D a similarity operation with 3 subsequent series of Givens transformations, each of them going completely from the top to the bottom of the matrix. This resulted in a 'Λ-shaped' representation for a unitary diagonal plus semiseparable matrix of semiseparability rank 3, with structure blocks situated just below the main diagonal, following immediately one after the other.

Having constructed these test matrices, we invoked our method for the eigenvalue computation of unitary rank structured matrices to compute the eigenvalues in a numerical way, $\lambda_{\text{computed},i}$. The used method is the two-step procedure consisting of unitary Hessenberg reduction, followed by the implicit

*Matlab is a registered trademark of The MathWorks, Inc.

QR-algorithm for the resulting unitary Hessenberg matrix. We made use of the straightforward implementation of the pull-through lemma, with a renormalization procedure applied in each step. As a measure for the error on the obtained eigenvalues, we took

$$\max\{\max_i \text{dist}(\lambda_{\text{computed},i}, \Lambda_{\text{exact}}), \max_j \text{dist}(\lambda_{\text{exact},j}, \Lambda_{\text{computed}})\}$$

with

$$\text{dist}(\lambda_{\text{computed},i}, \Lambda_{\text{exact}}) = \min_j |\lambda_{\text{computed},i} - \lambda_{\text{exact},j}|.$$

For each size, 3 samples were taken.

The results averaged over these 3 samples are shown in Figures 24, 25. Note that the latter figure confirms that both algorithms are quadratic in the matrix size n .

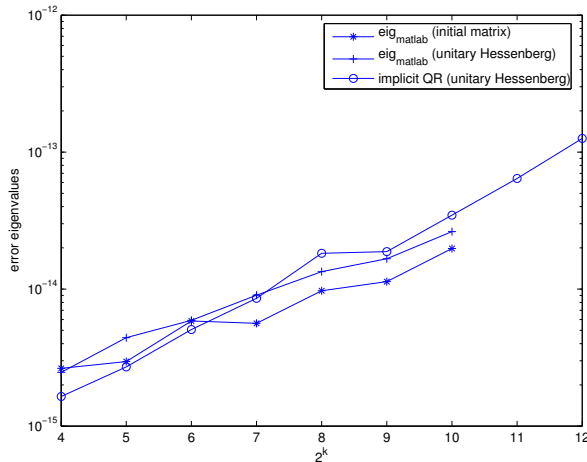


Figure 24: The figure shows the error for the eigenvalues as computed by Matlab before and after the Hessenberg reduction process of Section 3, and for the resulting matrix obtained at the end of the implicit QR-algorithm. The matrix size was taken as $n = 2^k$ for $k = 4, 5, \dots, 12$ and the rank index $r = 3$. These results were averaged over 3 random samples.

For the second experiment, we took a fixed size $n = 2^9 = 512$, and we let the rank index r increase from 1 up to 10. The test matrices were generated in exactly the same way as in the previous experiment. There was taken one sample for each rank index.

The results are shown in Figures 26, 27. Note that the latter figure confirms that the Hessenberg reduction algorithm is only linear in the rank index r , corresponding to the complexity $O((r + s)n^2)$, with distance of the structure blocks to the main diagonal s equal to zero in the present case.

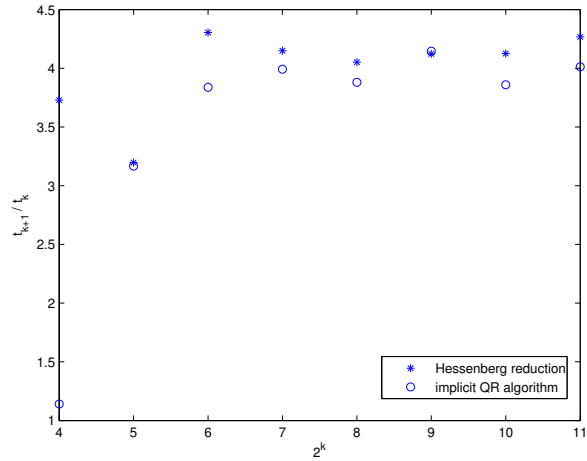


Figure 25: The figure shows the fractions of the subsequent timings for the Hessenberg reduction process, and for the implicit QR-algorithm, averaged over 3 random samples. The matrix size was taken as $n = 2^k$ for $k = 4, 5, \dots, 12$ and the rank index $r = 3$.

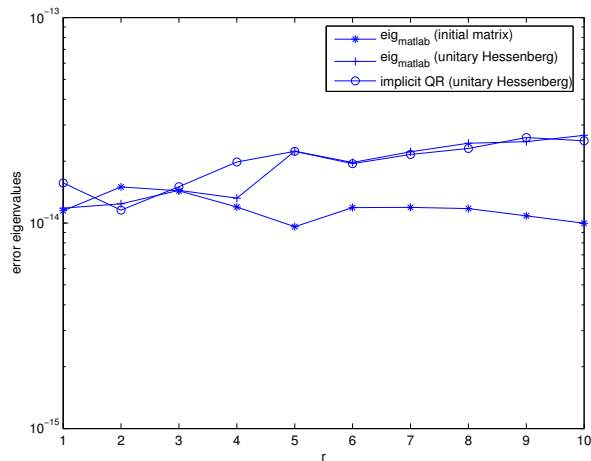


Figure 26: The figure shows the error for the eigenvalues as computed by Matlab before and after the Hessenberg reduction process of Section 3, and for the resulting matrix obtained at the end of the implicit QR-algorithm, for size $n = 2^9 = 512$ and rank index $r = 1, \dots, 10$.

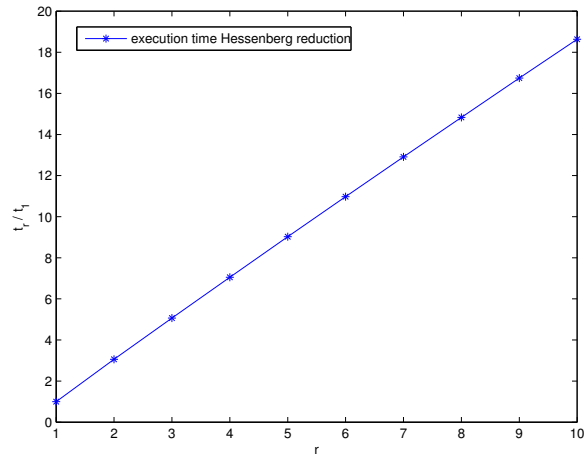


Figure 27: The figure shows the relative timings of the Hessenberg reduction process for size $n = 2^9 = 512$ and for rank index $r = 1, \dots, 10$.

References

- [1] G. S. Ammar, W. B. Gragg, and L. Reichel. On the eigenproblem for orthogonal matrices. In *25th IEEE Conference on Decision and Control, Athens, Greece*, pages 1963–1966, 1986.
- [2] G. S. Ammar, W. B. Gragg, and L. Reichel. Determination of Pisarenko frequency estimates as eigenvalues of an orthogonal matrix. In F. T. Luk, editor, *Proceedings of SPIE - The International Society for Optical Engineers, Advanced Algorithms and Architectures for Signal Processing II, California*, volume 826, pages 143–145, Bellingham, Washington, U.S.A., 1987. SPIE, SPIE. Proceedings of SPIE - The International Society for Optical Engineering.
- [3] G. S. Ammar, W. B. Gragg, and L. Reichel. Constructing a unitary Hessenberg matrix from spectral data. In G. H. Golub and P. Van Dooren, editors, *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, volume 70 of *Computer and Systems Sciences*, pages 385–395. Springer-Verlag, 1991.
- [4] G. S. Ammar, W. B. Gragg, and L. Reichel. DOWDATING OF SZEGŐ POLYNOMIALS AND DATA-FITTING APPLICATIONS. *Linear Algebra and its Applications*, 172:315–336, 1992.

- [5] G. S. Ammar, L. Reichel, and D. C. Sorensen. An implementation of a divide and conquer algorithm for the unitary eigenproblem. *ACM Transactions on Mathematical Software*, 18(3):292–307, September 1992.
- [6] P. Arbenz and G. H. Golub. On the spectral decomposition of hermitian matrices modified by low rank perturbations with applications. *SIAM Journal on Matrix Analysis and its Applications*, 9(1):40–58, January 1988.
- [7] A. Bultheel and M. Van Barel. Vector orthogonal polynomials and least squares approximation. *SIAM Journal on Matrix Analysis and its Applications*, 16(3):863–885, 1995.
- [8] A. Bunse-Gerstner and L. Elsner. Schur parameter pencils for the solution of the unitary eigenproblem. *Linear Algebra and its Applications*, 154–156:741–778, 1991.
- [9] A. Bunse-Gerstner and C. He. On a Sturm sequence of polynomials for unitary Hessenberg matrices. *SIAM Journal on Matrix Analysis and its Applications*, 16(4):1043–1055, 1995.
- [10] R. J. A. David and D. S. Watkins. Efficient implementation of the multishift QR algorithm for the unitary eigenvalue problem. *SIAM Journal on Matrix Analysis and its Applications*, 2006. To appear.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] S. Delvaux and M. Van Barel. Structures preserved by matrix inversion. *SIAM Journal on Matrix Analysis and its Applications*, 28(1):213–228, 2006.
- [15] 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.
- [16] P. Dewilde and A.-J. van der Veen. *Time-varying systems and computations*. Kluwer Academic Publishers, Boston, June 1998.

- [17] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
- [18] W. B. Gragg. The QR algorithm for unitary Hessenberg matrices. *Journal of Computational and Applied Mathematics*, 16:1–8, 1986.
- [19] W. B. Gragg and L. Reichel. A divide and conquer method for unitary and orthogonal eigenproblems. *Numerische Mathematik*, 57:695–718, 1990.
- [20] M. Gu, R. Guzzo, X.-B. Chi, and X.-Q. Cao. A stable divide and conquer algorithm for the unitary eigenproblem. *SIAM Journal on Matrix Analysis and its Applications*, 25:385–404, 2003.
- [21] B. N. Parlett. *The Symmetric Eigenvalue Problem*, volume 20 of *Classics in Applied Mathematics*. SIAM, Philadelphia, 1998.
- [22] L. Reichel, G. S. Ammar, and W. B. Gragg. Discrete least squares approximation by trigonometric polynomials. *Math. Comp.*, 57:273–289, 1991.
- [23] H. Rutishauser. Bestimmung der eigenwerte orthogonaler matrizen. *Numerische Mathematik*, 9:104–108, 1966.
- [24] M. Stewart. An error analysis of a unitary Hessenberg QR algorithm. *SIAM Journal on Matrix Analysis and its Applications*, 28(1):40–67, 2006.
- [25] M. Van Barel and A. Bultheel. A parallel algorithm for discrete least squares rational approximation. *Numerische Mathematik*, 63:99–121, 1992.
- [26] M. Van Barel and A. Bultheel. Discrete linearized least squares approximation on the unit circle. *Journal of Computational and Applied Mathematics*, 50:545–563, 1994.
- [27] M. Van Barel and A. Bultheel. Orthonormal polynomial vectors and least squares approximation for a discrete inner product. *Electronic Transactions on Numerical Analysis*, 3:1–23, March 1995.
- [28] R. Vandebril, E. Van Camp, M. Van Barel, and N. Mastronardi. Orthogonal similarity transformation of a symmetric matrix into a diagonal-plus-semiseparable one with free choice of the diagonal. *Numerische Mathematik*, 102:709–726, 2006.
- [29] T. L. Wang and W. B. Gragg. Convergence of the shifted QR algorithm, for unitary Hessenberg matrices. *Mathematics of Computation*, 71(240):1473–1496, 2002.
- [30] T. L. Wang and W. B. Gragg. Convergence of the unitary QR algorithm with unitary Wilkinson shift. *Mathematics of Computation*, 72(241):375–385, 2003.
- [31] D. S. Watkins. The transmission of shifts and shift blurring in the QR algorithm. *Linear Algebra and its Applications*, 241-243:877–896, 1996.