

# A parallel QR-factorization/solver of structured rank matrices

*Vandebril Raf*  
*Van Barel Marc*  
*Mastronardi Nicola*

*Report TW 474, October 2006*



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

# A parallel QR-factorization/solver of structured rank matrices

*Vandebril Raf*  
*Van Barel Marc*  
*Mastronardi Nicola*

*Report TW 474, October 2006*

Department of Computer Science, K.U.Leuven

## Abstract

This manuscript focuses on the development of a parallel  $QR$ -factorization of structured rank matrices, which can then be used for solving systems of equations. First we will prove the existence of two types of Givens transformations, named rank decreasing and rank expanding Givens transformations. Combining these two types of Givens transformations leads to different patterns for annihilating the lower triangular part of structured rank matrices. How to obtain different annihilation patterns, for computing the upper triangular factor  $R$ , such as the  $\vee$  and  $\wedge$  pattern will be investigated. Another pattern namely the  $X$ -pattern will be used for computing the  $QR$ -factorization in a parallel way.

As an example of such a parallel  $QR$ -factorization, we will implement it for a quasiseparable matrix. This factorization can be run on 2 processors, with one step of intermediate communication in which one row needs to be sent from one processor to the other and back. Another example, showing how to deduce a parallel  $QR$ -factorization for a more general rank structure will also be discussed.

Numerical experiments are included for demonstrating the accuracy and speed of this parallel algorithm w.r.t. the existing factorization of quasiseparable matrices. Also some numerical experiments on solving systems of equations using this approach will be given.

**Keywords:** Parallel  $QR$ -factorization, structured rank matrices, quasiseparable matrix

**Keywords :** Parallel  $QR$ -factorization, structured rank matrices, quasiseparable matrix

**AMS(MOS) Classification :** Primary : 65F05, Secondary : 15A21.

# A parallel QR-factorization/solver of structured rank matrices \*

Vandebril Raf<sup>†</sup>, Van Barel Marc<sup>‡</sup> and Mastronardi Nicola<sup>§</sup>

20th October 2006

## Abstract

This manuscript focuses on the development of a parallel  $QR$ -factorization of structured rank matrices, which can then be used for solving systems of equations. First we will prove the existence of two types of Givens transformations, named rank decreasing and rank expanding Givens transformations. Combining these two types of Givens transformations leads to different patterns for annihilating the lower triangular part of structured rank matrices. How to obtain different annihilation patterns, for computing the upper triangular factor  $R$ , such as the  $\vee$  and  $\wedge$  pattern will be investigated. Another pattern namely the  $\times$ -pattern will be used for computing the  $QR$ -factorization in a parallel way.

As an example of such a parallel  $QR$ -factorization, we will implement it for a quasiseparable matrix. This factorization can be run on 2 processors, with one step of intermediate communication in which one row needs to be sent from one processor to the other and back. Another example, showing how to deduce a parallel  $QR$ -factorization for a more general rank structure will also be discussed.

Numerical experiments are included for demonstrating the accuracy and speed of this parallel algorithm w.r.t. the existing factorization of quasiseparable matrices. Also some numerical experiments on solving systems of equations using this approach will be given.

**Keywords:** Parallel  $QR$ -factorization, structured rank matrices, quasiseparable matrix

## 1 Introduction

Due to the interest nowadays in structured rank matrices, the knowledge on this class of matrices is growing rapidly. A structured rank matrix is characterized by the fact that specific parts taken out of the matrix satisfy low rank properties, such as for example quasiseparable, semiseparable, unitary Hessenberg matrices and so forth. Various accurate and fast algorithms are already known for computing for example the  $QR$ - and  $URV$ -factorization [1, 2, 3, 4], the eigenvalue decomposition [5, 6, 7, 8], the singular value decomposition of certain types of structured rank matrices [9].

In this manuscript we will focus on the  $QR$ -factorization of structured rank matrices. Currently, all the  $QR$ -factorizations of structured rank matrices consist of two main steps. A first step consists of removing the low rank part in the lower triangular part of the matrix. This results in a generalized Hessenberg matrix, having several subdiagonals different from zero. The second part consists of removing the remaining

---

\*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), Iterative methods in numerical Linear Algebra, 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 first author has a grant of “Postdoctoraal Onderzoeker” from the Fund of Scientific Research Flanders (FWO-Vlaanderen). The research of the third author was partially supported by MIUR, grant number 2004015437, by the short term mobility program, Consiglio Nazionale delle Ricerche and by VII Programma Esecutivo di Collaborazione Scientifica Italia–Comunità Francese del Belgio, 2005–2006. The scientific responsibility rests with the authors.

<sup>†</sup>K.U.Leuven, Dept. Computerwetenschappen, raf.vandebril@cs.kuleuven.be

<sup>‡</sup>K.U.Leuven, Dept. Computerwetenschappen, marc.vanbarel@cs.kuleuven.be

<sup>§</sup>Istituto per le Applicazioni del Calcolo M. Picone, sez. Bari, n.mastronardi@ba.iac.cnr.it

subdiagonals in order to obtain an upper triangular matrix in this fashion. In the terminology of this paper this means that first a sequence of rank decreasing Givens transformations is performed, namely the low rank part is removed, and this is done by reducing consecutively the rank of this part to zero. The second part consists of a sequence of rank expanding Givens transformations. The generalized Hessenberg matrix has a zero block in the lower left corner and by performing these rank expanding Givens transformations this block of zero rank expands until it reaches the diagonal and the matrix becomes upper triangular.

In this paper we will focus on two specific issues. First we will prove the existence of rank expanding Givens transformations in a general context and secondly we will investigate the possibility of interchanging the mutual position of rank expanding and rank decreasing Givens transformations, by means of a shift through lemma.

Interchanging the position of Givens transformations will lead to different patterns, to annihilate the lower triangular structure of matrices. For example one can now first perform a sequence of rank expanding Givens transformations, followed by a sequence of rank decreasing Givens transformations. This order is different than the traditional one, but leads to a similar factorization.

In this manuscript we will first focus attention to the most simple case, namely the case of quasiseparable matrices. Further on in the text also indications and examples are given to show the applicability of these techniques to higher order structured rank matrices. For the class of quasiseparable matrices one sequence of rank decreasing Givens transformations and one sequence of rank expanding Givens transformations is needed to compute the  $QR$ -factorization. Due to our knowledge on the different patterns, we know that we can interchange the order of these sequences. Moreover, we can construct a special pattern (called an  $\times$ -pattern), such that we start on top of the matrix with a descending sequence of rank expanding Givens transformations, and on the bottom with an upgoing rank decreasing sequence Givens transformations. When these two sequences of Givens transformations meet each other in the middle of the matrix, we have to perform a specific Givens transformation, after which we have again two sequences of independent Givens transformations. One sequence goes back to the top and the other one goes back to the bottom. After these transformations, we have computed the  $QR$ -factorization.

This  $\times$ -pattern was firstly discussed in [10], by Delvaux and Van Barel. Also the graphical representation, leading to the interpretation in terms of  $\times$  and  $\vee$ -shaped patterns of annihilation can be found in their manuscript.

This  $\times$ -pattern for quasiseparable matrices is suitable for implementation on a parallel computer. Divide the matrix into two parts. The first  $n_1$  rows are sent to a first processor and the last  $n_2 = n - n_1$  rows are sent to another processor. Both processors perform their type of Givens transformation, either a descending or an upgoing sequence of Givens transformations. Then one step of communication is necessary and both processors can finalize the process. Finally the first processor has the top  $n_1$  rows of the factor  $R$  and the second processor has the last  $n_2$  rows of the factor  $R$  of the  $QR$ -factorization.

The manuscript is organized as follows. In the second section we will briefly recapitulate some results on structured rank matrices and on the computation of the  $QR$ -factorization for quasiseparable matrices. In Section 3 we introduce the two types of Givens transformations we will be working with. Namely the rank expanding Givens and the rank decreasing Givens transformations. These two types of transformations form the basis for the development of the parallel algorithm. Section 4 discusses some lemmas which give us some flexibility for working with Givens transformations. Based on these possibilities we will be able to change the order of consecutive Givens transformations leading to different patterns for annihilating when computing the  $QR$ -factorization. In Section 5 we will discuss the possibilities for parallelizing the previously discussed schemes. In Section 6 different possibilities for developing parallel algorithms for higher order structured rank matrices will be presented. The final section of this manuscript contains numerical results related to the  $QR$ -factorization and also to solving systems of equations involving a parallel  $QR$ -algorithm. Timings as well as results on the accuracy will be presented.

## 2 Definitions and preliminary results

The main focus of this paper is the development of a parallel  $QR$ -factorization for quasiseparable matrices. Let us briefly introduce what is meant with a quasiseparable matrix, and how we can compute the  $QR$ -factorization of this quasiseparable matrix. A first definition of quasiseparable matrices, as well as an

inversion method for them, can be found in [11], see also [12].

**Definition 1.** A matrix  $A \in \mathbb{R}^{n \times n}$  is named a (lower) quasiseparable matrix (of quasiseparability rank 1) if any submatrix taken out of the strictly lower triangular part has rank at most 1. More precisely this means that for every  $i = 2, \dots, n$ <sup>1</sup>:

$$\text{rank}A(i : n, 1 : i - 1) \leq 1.$$

The matrices considered in this manuscript only have structural constraints posed on the lower triangular part of the matrix. Quite often these matrices are also referred to as lower quasiseparable matrices.

A structured rank matrix in general is a matrix for which certain blocks in the matrix satisfy specific rank constraints. Examples of structured rank matrices are semiseparable matrices, band matrices, Hessenberg matrices, unitary Hessenberg matrices, semiseparable plus band matrices, etc. In this manuscript we will mainly focus on the development of a parallel  $QR$ -algorithm for quasiseparable matrices of quasiseparability rank one. In the section before the numerical experiments we will briefly indicate how the presented results are also applicable onto higher order structured rank matrices.

Let us briefly repeat the traditional  $QR$ -factorization of a quasiseparable matrix. Let us depict our quasiseparable matrix as follows:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \boxtimes & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times \end{bmatrix}.$$

The arbitrary elements in the matrix are denoted by  $\times$ . The elements satisfying a specific structure are denoted by  $\boxtimes$ . Performing now on this matrix a first sequence of Givens transformations from bottom to top, one can annihilate the complete part of quasiseparability rank 1, denoted by the elements  $\boxtimes$ . Combining all these Givens transformations into one orthogonal matrix  $Q_1^T$  this gives us the following result:

$$Q_1^T A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

Hence we obtain a Hessenberg matrix, which can be transformed into an upper triangular matrix, by performing a sequence of descending Givens transformations, removing thereby the subdiagonal. Combining these Givens transformations into the orthogonal matrix  $Q_2^T$  gives us:

$$Q_2^T Q_1^T A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

This leads in a simple manner to the  $QR$ -decomposition of the matrix  $Q$  in which we first perform an upgoing sequence of Givens transformations (removing the low rank part), followed by a descending sequence of Givens transformations (expanding the part of rank zero). All the Givens transformations used in this factorization are zero creating Givens transformations. There exist however also other types of Givens transformations, which we will need for the parallel  $QR$ -factorization.

### 3 Types of Givens transformations

Givens transformations are common tools for creating zeros in matrices [13, 14]. But Givens transformations can also be used for creating rank 1 blocks in matrices. In this section we will prove the existence of a rank expanding Givens transformation, creating rank 1 blocks in matrices.

<sup>1</sup>We use MATLAB-style notation.

### 3.1 The Givens transformation

In this subsection, we will propose an analytic way of computing a Givens transformation for expanding the rank structure. We will prove the existence of a Givens transformation, which will be used afterwards in the next subsection for developing a sequence of descending rank expanding Givens transformations. In the example following the theorem, we will use the reduction of a Hessenberg matrix to upper triangular form as an example of a descending rank expanding sequence of Givens transformations.

**Theorem 1 (Descending rank expanding Givens transformation).** *Suppose the following  $2 \times 2$  matrix is given*

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \quad (1)$$

*Then there exists a Givens transformation such that the second row of the matrix  $G^T A$  and the row  $[e, f]$  are linearly dependent. The value  $t$  in the Givens transformation  $G$  as in (2), is defined as*

$$t = \frac{af - be}{cf - de},$$

*under the assumption that  $cf - de \neq 0$ , otherwise one can simply take  $G = I_2$ .*

*Proof.* Suppose we have the matrix  $A$  and the Givens transformation  $G$  as follows:

$$vA = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ and } G = \frac{1}{\sqrt{1+t^2}} \begin{bmatrix} t & -1 \\ 1 & t \end{bmatrix}. \quad (2)$$

Assume  $[c, d]$  and  $[e, f]$  to be linearly independent, otherwise we could have taken the Givens transformation equal to the identity matrix.

Let us compute the product  $G^T A$ :

$$\frac{1}{\sqrt{1+t^2}} \begin{bmatrix} t & 1 \\ -1 & t \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{1}{\sqrt{1+t^2}} \begin{bmatrix} at + c & bt + d \\ -a + ct & -b + dt \end{bmatrix}.$$

The second row being dependent of  $[e, f]$  leads to the following relation:

$$f(-a + ct) - e(-b + dt) = 0.$$

Rewriting this equation towards  $t$  gives us the following well-defined equation:

$$t = \frac{af - be}{cf - de}.$$

This equation is well defined, as we assumed  $[c, d]$  to be independent of  $[e, f]$ . □

This type of Givens transformation was already used before in [15, 16]. Let us show that the rank expanding Givens transformations as we computed them here are a generalization of the transformations used for bringing an upper Hessenberg matrix back to upper triangular form.

**Example 1.** *Suppose we have a Hessenberg matrix  $H$  and we want to reduce it to upper triangular form. Instead of using the standard Givens transformations, eliminating the subdiagonal elements, we will use here the Givens transformations from Theorem 1 to expand the zero rank below the subdiagonal. This is done by a sequence of Givens transformations going from top to bottom.*

*Suppose we have for example the following Hessenberg matrix:*

$$H = \begin{bmatrix} 1 & \frac{-1}{\sqrt{6}} & \frac{3}{\sqrt{3}} \\ 1 & \frac{3}{\sqrt{6}} & \frac{-1}{\sqrt{3}} \\ 0 & \frac{2\sqrt{2}}{\sqrt{3}} & \frac{5}{\sqrt{3}} \end{bmatrix}.$$

Computing, the first Givens transformation applied on row 1 and 2 in order to make part of the transformed second row dependent of

$$[e, f] = \left[ 0, \frac{2\sqrt{2}}{\sqrt{3}} \right],$$

gives us the following transformation (use the same notation as in Corollary 1):

$$\begin{aligned} t &= \frac{af - be}{cf - de} \\ &= \frac{a}{c} \\ &= 1. \end{aligned}$$

Hence our Givens transformation, will be of the following form:

$$\check{G}_1^T = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}.$$

Applying the transformation  $G_1^T$  (the  $2 \times 2$  Givens transformation  $\check{G}_1^T$  is embedded into a  $3 \times 3$  Givens transformation  $G_1^T$ ) onto the matrix  $H$  annihilates the first subdiagonal element, thereby expanding the zero rank structure below the subdiagonal. One can easily continue this procedure and conclude that the rank expanding Givens transformations lift up the zero structure and hence create an upper triangular matrix. In this example, we can clearly see that a zero creating Givens transformation, can also be at the same time a rank expanding Givens transformation.

For the implementation of this specific Givens transformation, we adapted the standard implementation of a zero creating Givens transformation. We obtained the following code in MATLAB style notation by changing the one from [14]. The matrix  $A$  corresponds to the two by two matrix the Givens transformation is acting on and the vector  $V$  contains the elements  $[e, f]$ . The output consists of the cosine  $c$  and the sine  $s$  of the transformation, as well as the transformed matrix  $A$ .

```
function [c,s,A] = Givensexp(A,V);

y=V(1)*A(2,2)-A(2,1)*V(2);
x=-(A(1,1)*V(2)-A(1,2)*V(1));

if (x == 0)
    % In case this is zero, we obtain immediately G=I
    c = 1; s = 0;
else
    if (abs(x) >= abs(y))
        t = y/x; r = sqrt(1 + t*t);
        c = 1/r; s = t*c; r = x*r;
    else
        t = x/y; r = sqrt(1 + t*t);
        s = 1/r; c = t*s;
    end
    A(1:2,:)=[c,s;-conj(s),c]*A(1:2,:);
end
```

We remark that in the presented code the criterion  $x == 0$ , can be made relatively depending on the machine precision.

### 3.2 A sequence of these transformations

In the previous subsection already an example of a sequence of descending rank expanding transformations was presented.

In general, when having a rank 1 part in a matrix one is always able to lift up this part, such that it includes at most the main diagonal. For example start from the following matrix. The elements  $\boxtimes$  denote the elements belonging to the rank one part. After performing a sequence of descending rank expanding Givens transformations, one obtains the matrix on the right.

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \boxtimes & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times \end{bmatrix} \text{ resulting in } \begin{bmatrix} \boxtimes & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix}$$

**Note 1.** The expansion of a rank 1 structure never includes any of the superdiagonals, unless the matrix is singular. This remark can be verified easily as otherwise the global matrix rank otherwise changes. We will come back to this remark later on in the section on more general structures.

For the development of the parallel  $QR$ -algorithm for quasiseparable matrices, which is the main focus of this manuscript, the expansion of the rank 1 part as shown in the figure above is sufficient. For the development of a parallel  $QR$ -algorithm for higher order structured rank matrices one also needs to be able to lift up for example parts of matrices of rank 2. This will be discussed briefly in a forthcoming section.

### 3.3 Rank decreasing sequence of transformations

A sequence of Givens transformations, removing a rank 1 structure in a matrix is called a sequence of rank decreasing Givens transformations, simply because it reduces the rank from 1 to 0. For example the following matrix, discussed before (left), results after such a sequence of rank decreasing Givens transformations from bottom to top in the following matrix (right).

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \boxtimes & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times \end{bmatrix} \text{ resulting in } \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

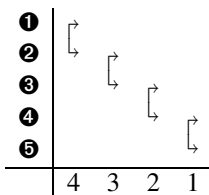
Similarly as in the previous case we remark that the existence of such a sequence as discussed here is sufficient for the development of the parallel  $QR$ -factorization for quasiseparable matrices. Further on in the text we will briefly reconsider other cases.

Let us now first discuss the traditional  $QR$ -factorization of a quasiseparable matrix, and then we will discuss how we can change the considered annihilation pattern to obtain a different order in the Givens transformations.

## 4 Different annihilation patterns

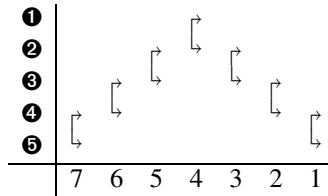
To be able to design different patterns of annihilation, and to characterize them, we introduce a new kind of notation. For example, to bring a semiseparable matrix to upper triangular form, we use one sequence of Givens transformations from bottom to top. This means that for a  $5 \times 5$  matrix the first applied Givens transformation works on the last two rows, followed by a Givens transformation working on row 3 and 4 and so on.

To depict graphically these Givens transformations, w.r.t. their order and the rows they are acting on, we use the following figure.



The numbered circles on the vertical axis depict the rows of the matrix, to indicate on which rows the Givens transformations will act. The bottom numbers represent in some sense a time line to indicate in which order the Givens transformations are performed. The brackets in the table represent graphically a Givens transformation acting on the rows in which the arrows of the brackets are lying. Let us explain more in detail this scheme. First a Givens transformation is performed, acting on row 5 and row 4. Secondly a Givens transformation is performed acting on row 3 and row 4 and this process continues. So the scheme given above just represents in a graphical way the orthogonal factor  $Q^T$  and a factorization of this matrix in terms of Givens transformations.

Let us illustrate this graphical representation with a second example. Suppose we have a quasiseparable matrix. To make this matrix upper triangular, we first perform a sequence of Givens transformations from bottom to top to remove the low rank part of the quasiseparable matrix, secondly we perform a sequence of Givens transformations from top to bottom, to remove the subdiagonal elements of the remaining Hessenberg matrix. This process was already discussed before in the introduction. Graphically this is depicted as follows (involving 7 Givens transformations acting on a  $5 \times 5$  quasiseparable matrix):



The first four transformations clearly go from bottom to top, whereas the last four transformations go from top to bottom.

Using this notation, we will construct some types of different annihilation patterns. Based on the sequences of Givens transformations as initially designed for bringing the matrix to upper triangular form, it is interesting to remark that we can derive other patterns of Givens transformations leading to the same  $QR$ -factorization. For some of the newly designed patterns we will illustrate the effect of these new annihilation sequences on the matrix they are acting on.

#### 4.1 Theorems connected to Givens transformations

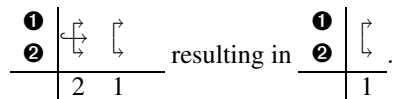
In the next subsections, we need to have more flexibility for working with Givens transformations. In order to do so, we need two lemmas. The first lemma shows us that we can concatenate two Givens transformations acting on the same rows. The second lemma shows us, that under some mild conditions, we can rearrange the order of some Givens transformations.

**Lemma 1.** *Suppose two Givens transformations  $G_1$  and  $G_2$  are given:*

$$G_1 = \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix} \text{ and } G_2 = \begin{bmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{bmatrix}.$$

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

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



The next lemma is slightly more complicated and changes the order of three Givens transformations.

**Lemma 2 (Shift through lemma).** *Suppose three  $3 \times 3$  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 (when applied on the left to a matrix).*

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.

*Proof.* The proof is straightforward, based on the factorization of a  $3 \times 3$  orthogonal matrix. Suppose we have an orthogonal matrix  $U$ . We will now depict a factorization of this matrix  $U$  into two sequences of Givens transformations like described in the lemma.

The first factorization of this orthogonal matrix, makes the matrix upper triangular in the traditional way. The first Givens transformation  $\hat{G}_1^T$  acts on row 2 and 3 of the matrix  $U$ , creating thereby a zero in the lower-left position:

$$\hat{G}_1^T U = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix}.$$

The second Givens transformation acts on the first and second row, to create a zero in the second position of the first column:

$$\hat{G}_2^T \hat{G}_1^T U = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}.$$

Finally the last transformation  $\hat{G}_3^T$  creates the last zero to make the matrix of upper triangular form.

$$\hat{G}_3^T \hat{G}_2^T \hat{G}_1^T U = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix}.$$

Suppose we have chosen all Givens transformations in such a manner that the upper triangular matrix has positive diagonal elements. Due to the fact that the resulting upper triangular matrix is orthogonal it has to be the identity matrix, hence we have the following factorization of the orthogonal matrix  $U$ :

$$U = \hat{G}_1 \hat{G}_2 \hat{G}_3. \quad (3)$$

Let us consider now a different factorization of the orthogonal matrix  $U$ . Perform a first Givens transformation, to annihilate the upper-right element of the matrix  $U$ , where the Givens transformation, acts on the first and second row:

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

Similarly as above, one can continue, to reduce the orthogonal matrix to lower triangular form, with positive diagonal elements. Hence one obtains a factorization of the following form:

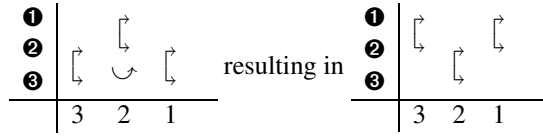
$$U = G_1 G_2 G_3. \quad (4)$$

Combining Equations 3 and 4, leads to the desired result.  $\square$

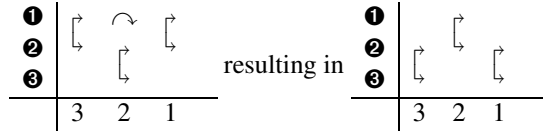
**Note 2.** *Two remarks have to be made.*

- *We remark that in fact there is more to the proof than we mention here. In fact the transformations acting on the orthogonal matrix, reducing it to lower triangular form have also a specific effect on the lower triangular part. Looking in more detail at the lower triangular part, one can see that the first Givens transformation, creates a  $2 \times 2$  low rank block in the lower left corner of the orthogonal matrix.*
- *In some sense one can consider the fusion of two Givens transformations as a special case of the shift through lemma. Instead of directly applying the fusion, the reader can put the identity Givens transformation in between these two transformations. Then he can apply the shift through lemma. The final outcome will be identical to applying directly the fusion of these two Givens transformations.*

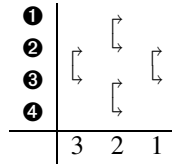
Graphically we will depict the shift through lemma as follows:



and in the other direction this becomes:



Remark, that, if we cannot place the  $\curvearrowright$  or  $\curvearrowleft$  arrow at that specific position, that we cannot apply the shift through lemma. The reader can verify, that for example in the following graphical scheme, we cannot use the lemma.

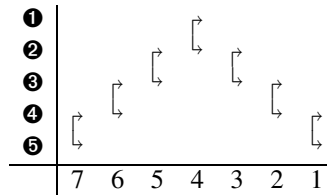


To apply the shift through lemma, in some sense, we need to have some extra place to perform the action. Based on these operations we can interchange the order of the upgoing and descending sequences of Givens transformations. Let us mention some of the different patterns.

## 4.2 The $\wedge$ -pattern

The  $\wedge$ -pattern for computing the  $QR$ -factorization of a structured rank matrix is in fact the standard pattern as described in the introduction and used throughout most of the papers (see e.g. [1]). First, we remove the rank structure by performing sequences of Givens transformations from bottom to top. This gives us in fact the following sequences of Givens transformations (e.g. two in this case)  $\searrow$ . Depending on the number of subdiagonals in the resulting matrix, we need to perform some rank expanding sequences of Givens transformations, from top to bottom  $\swarrow$  (two in this case). Combining these Givens transformations from both sequences gives us the following pattern  $\swarrow\searrow$ , which we briefly call the  $\wedge$ -pattern.

Suppose, e.g., that we have a quasiseparable matrix of rank 1. Performing the Givens transformations as described before, we get the following graphical representation of the reduction:



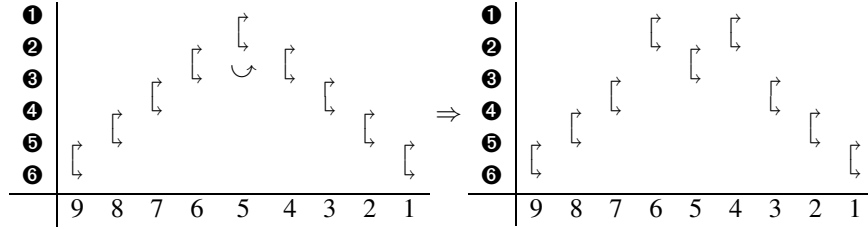
This is called a  $\wedge$ -pattern.

The reader can observe that the top three Givens transformations admit the shift through Lemma. In this way we can drag the Givens transformation in position 5 through the Givens transformations in position 4 and 3. Let us observe what kind of patterns we get in this case.

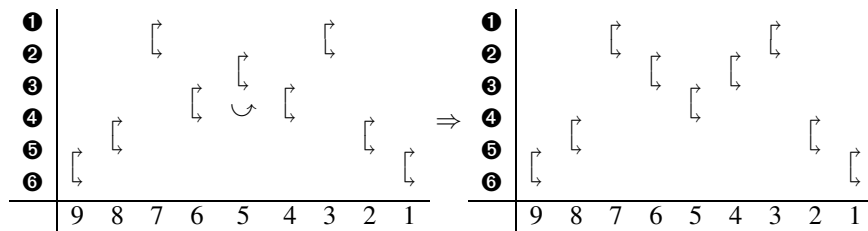
## 4.3 The $\times$ -pattern

We will graphically illustrate what happens if we apply the shift through lemma as indicated in the previous section. Suppose we have the following graphical reduction scheme for reducing our matrix to upper

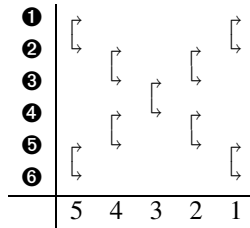
triangular form. For esthetical reasons in the figures, we assume here, our matrix to be of size  $6 \times 6$ . First we apply the shift through lemma at positions 6,5 and 4.



Rearranging slightly the Givens transformations from positions, we can again re-apply the shift through lemma. We can change the order of some of the Givens transformations, in the scheme above 7 and 6 (and 4 and 3), as they act on different rows and hence do not interfere with each other.



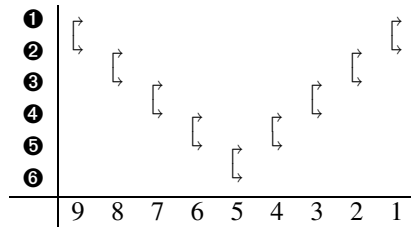
Let us compress the above representation.



This shows us another pattern of performing the Givens transformations, namely the X-pattern. Continuing to apply the shift through lemma gives us another pattern.

#### 4.4 The V-pattern

Continuing now this procedure, by applying two more times the shift through lemma, gives us the following graphical representation of a possible reduction of the matrix to upper triangular form.



This presents us clearly the V-pattern for computing the QR-factorization. In case there are more upgoing and descending sequences of Givens transformations, one can also shift all of the descending sequences through. In fact this creates an incredible amount of possibilities, as shown in the next example.

**Example 2.** Suppose we have a matrix brought to upper triangular form by performing two upgoing sequences of Givens transformations and two descending sequences of transformations (e.g. a quasiseparable matrix of quasiseparability rank 2). The following, not complete list, shows some possibilities of combinations of these sequences for making the matrix upper triangular. We start with the  $\wedge$ -pattern, and change continuously the order of the involved transformations, to arrive at the V-pattern.

- The standard  $\wedge$ -pattern giving us briefly the following sequences:  $\//\backslash$ .
- In the middle we can create one  $\times$ -pattern:  $\//$ .
- In the middle we can have one  $\vee$ -pattern:  $\wedge\wedge$ .
- Combinations with  $\times$ -patterns:  $\times\wedge$  or  $\wedge\times$  or  $\times\times$ .
- Combinations following from the previous patterns:  $\//\wedge$  and  $\//\vee$ .
- In the middle one can have one  $\wedge$ -pattern:  $\vee\vee$ .
- In the middle we can create another  $\times$ -pattern:  $\//$ .
- The  $\vee$ -pattern:  $\//\vee$ .

Clearly there are already numerous possibilities for 2 upgoing and 2 descending sequences.

In the following section we will take a look at the effect of the first sequence of Givens transformations on the matrix in case we apply a  $\vee$ -pattern for computing the  $QR$ -factorization of a structured rank matrix.

#### 4.5 More on the Givens transformations in the $\vee$ -pattern

We investigate this  $\vee$ -pattern via reverse engineering. Suppose we have a  $\vee$ -pattern for making a  $5 \times 5$  lower quasiseparable matrix upper triangular, assuming the matrix to be of quasiseparability rank 1. We will investigate now, what the effect of the first sequence of descending Givens transformations on this matrix  $A$  needs to be. We have the following equation:

$$\hat{G}_1^T \hat{G}_2^T \hat{G}_3^T \hat{G}_4^T G_3^T G_2^T G_1^T A = R, \quad (5)$$

where  $R$  is a  $5 \times 5$  upper triangular matrix. Moreover the first applied sequence of Givens transformations  $G_3^T G_2^T G_1^T$ , works on the matrix  $A$  from top to bottom. More precisely  $G_1^T$  acts on row 1 and row 2,  $G_2^T$  acts on row 2 and 3 and so on. The sequence of transformations  $\hat{G}_1^T \hat{G}_2^T \hat{G}_3^T \hat{G}_4^T$  works from bottom to top, where  $\hat{G}_4^T$  acts on row 4 and 5,  $\hat{G}_3^T$  acts on row 3 and 4, and so on. Rewriting Equation 5, by bringing the upgoing sequence of transformations to the right gives us

$$\begin{aligned} G_3^T G_2^T G_1^T A &= \hat{G}_4 \hat{G}_2 \hat{G}_2 \hat{G}_1 R \\ &= S. \end{aligned}$$

Because the sequence of transformations applied on the matrix  $R$  goes from top to bottom, we know that these transformations transform the matrix  $R$  into a matrix having a lower triangular part of semiseparable form. Hence we have that the transformations from top to bottom, namely  $G_3^T G_2^T G_1^T$  lift up in some sense the strictly lower triangular semiseparable structure to a lower triangular semiseparable structure. The following figures denote more precisely what is happening. We start on the left with the matrix  $A$ , and we depict what the impact of the transformations  $G_3^T G_2^T G_1^T$  needs to be on this matrix, to satisfy the equation above. Assume  $A_0 = A$ . To see more clearly what happens, we include already the upper left and lower right element in the strictly lower triangular semiseparable structure:

$$\begin{array}{ccc} \left[ \begin{array}{ccccc} \boxtimes & & & & \\ \boxtimes & \times & & & \\ \boxtimes & \boxtimes & \times & & \\ \boxtimes & \boxtimes & \boxtimes & \times & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{array} \right] & \xrightarrow{G_1^T A_0} & \left[ \begin{array}{ccccc} \boxtimes & & & & \\ \boxtimes & \boxtimes & & & \\ \boxtimes & \boxtimes & \times & & \\ \boxtimes & \boxtimes & \boxtimes & \times & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{array} \right] \\ & & \updownarrow \\ & & A_0 \xrightarrow{G_1^T A_0} A_1. \end{array}$$

As the complete result needs to be of lower triangular semiseparable form, the transformation  $G_1^T$  needs to add one more element into the semiseparable structure. This results in an inclusion of diagonal element 2 in the lower triangular rank structure. Givens transformation  $G_2^T$  causes the expansion of the low rank structure towards diagonal element 3.

$$\begin{array}{ccc}
 \begin{bmatrix} \boxtimes & & & & \\ \boxtimes & \boxtimes & & & \\ \boxtimes & \boxtimes & \times & & \\ \boxtimes & \boxtimes & \boxtimes & \times & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix} & \xrightarrow{G_2^T A_1} & \begin{bmatrix} \boxtimes & & & & \\ \boxtimes & \boxtimes & & & \\ \boxtimes & \boxtimes & \boxtimes & & \\ \boxtimes & \boxtimes & \boxtimes & \times & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix} \\
 & & \updownarrow \\
 A_1 & \xrightarrow{G_2^T A_1} & A_2.
 \end{array}$$

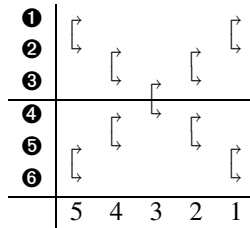
Finally the last Givens transformation  $G_3^T$  creates the following structure.

$$\begin{array}{ccc}
 \begin{bmatrix} \boxtimes & & & & \\ \boxtimes & \boxtimes & & & \\ \boxtimes & \boxtimes & \boxtimes & & \\ \boxtimes & \boxtimes & \boxtimes & \times & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix} & \xrightarrow{G_3^T A_2} & \begin{bmatrix} \boxtimes & & & & \\ \boxtimes & \boxtimes & & & \\ \boxtimes & \boxtimes & \boxtimes & & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix} \\
 & & \updownarrow \\
 A_2 & \xrightarrow{G_3^T A_2} & A_3.
 \end{array}$$

Hence the result of applying this sequence of Givens transformations from top to bottom is a matrix which has the lower triangular structure shifted upwards one position. In fact we have performed a rank expanding sequence of Givens transformations.

## 5 A parallel QR-factorization for quasiseparable matrices

In the previous subsection a specific  $\times$  shaped pattern was shown. This pattern can perform 2 Givens transformations simultaneously in the first step, see the graph below.



The extra horizontal line shows the action radius of the two processors. The first processor can only work on the top three rows and the second processor on the bottom three rows. The algorithm starts by performing a rank expanding Givens transformation on the top two rows and a rank decreasing Givens transformation on the bottom two rows. Then one can again continue by performing simultaneously Givens transformations on the top part and on the bottom part, until one reaches the shared Givens transformation in the middle, which is intersected by the horizontal line (this is the transformation at position 3). This indicates that information has to be exchanged from one processor to the other. After having performed this intermediate Givens transformation, one can again perform several Givens transformations simultaneously on both processors. For higher order structured rank matrices, we will present another scheme in a forthcoming section.

Let us present some information on possible tunings of this algorithm.

## 5.1 Some parameters of the algorithm

When designing a parallel algorithm there are several important concepts which have to be taken into consideration. First of all the simultaneous work has to be balanced. One wants to load both of the active processors with the same amount work, such that both processors do not have to wait as little as possible for the communication. Secondly we want to submit as little information as possible. In this section we provide some information on how to tune the load balance of both processors.

The X-pattern we showed divides the matrix  $A$  into two equally distributed parts, each containing the same ( $\pm 1$ ) amount of rows. Due to the quasiseparable structure however, the bottom  $n_2$  rows are much more structured than the upper  $n_1$  rows. The top matrix contains  $n_1 n - n_1^2/2 + 5n_1/2 - 1$  elements to be stored<sup>2</sup>, whereas the bottom matrix contains only  $n_2^2/2 + 5n_2/2$  elements. Considering now the performance of the Givens transformations on both matrices we see that applying the 2 sequences of Givens transformations on the top costs approximately  $12n_1 n + 6n_1^2$  operations, whereas this costs only  $6n_2^2$  operations for the bottom part. This means that when both processors obtain the same number of rows  $n_1 \approx n_2$ , processor one has to do much more work than processor two.

Looking back, however, at the intermediate steps to reduce the  $\wedge$ -pattern into the X-pattern we see that it is possible to obtain vertically nonsymmetric X-patterns (see for example some of the patterns in Section 4.3). This means that we can choose any matrix division, as long as  $n_1 + n_2 = n$ . This leads to a flexible way for dividing the matrix, such that processing the top matrix part takes as long as processing the bottom part. A natural choice of division might be such that  $12n_1 n + 6n_1^2 \approx 6n_2^2$ . This is a good choice in case both processors are of the same type. If both processors do not have the same architecture, this division does not necessarily lead to an equally distributed time for processing the matrices and hence needs to be taken case dependent.

As the amount of data submitted through the network is only dependent on the position of  $n_1$  and  $n_2$ , we cannot change this. The amount of data submitted is of the order  $2n_2$ . In the next subsection we will present a high level algorithm for computing QR-factorization in parallel.

## 5.2 The implementation

Let us briefly describe a high-level implementation of a parallel QR-factorization/solver of a quasiseparable matrix. The actual algorithm was implemented in MATLAB, using thereby the MatlabMPI package, for running the parallel algorithm on different machines.

We assume that we divided initially the work load of both machines, and moreover we assume that the local processor contains the top  $n_1$  rows and the remote processor contains the bottom  $n_2$  rows. The items in italics only need to be performed in case one wants to solve a system of equations by the implemented QR-factorization. In case one wants to solve a system of equations, also the right-hand side needs to be divided into two parts, the top part for the local and the bottom part for the remote processor. The main algorithm consists of the following steps.

- Perform in parallel:
  - Perform the rank expanding, descending Givens transformations on the local processor.  
*Perform the Givens transformations simultaneously on the right-hand side.*
  - Perform the rank annihilating, upgoing Givens transformations on the remote processor.  
*Perform the Givens transformations simultaneously on the right-hand side.*
- Send the top row of the matrix from the remote to the local processor.  
*Send the top element from the right-hand side from the remote to the local processor.*
- Perform the intersecting Givens transformation.  
*Perform this Givens transformations also on the right-hand side.*

---

<sup>2</sup>The number of elements stored, depends also on the representation of the quasiseparable part of the matrix. We silently assumed our quasiseparable matrix to be generator representable. This means that its strictly lower triangular part can be written as the strictly lower triangular part of a rank 1 matrix.

- Send the row back from the local to the remote machine.  
*Send the bottom element from the right-hand side back.*
- Perform in parallel:
  - Perform the rank annihilating upgoing Givens transformations on the local processor.  
*Perform this Givens transformations simultaneously on the right-hand side.*
  - Perform the rank expanding descending Givens transformations on the remote processor.  
*Perform this Givens transformations simultaneously on the right-hand side.*
- One has now computed the top part of the  $R$  factor on the local and the bottom part of the  $R$  factor on the remote processor.
- *Solve the bottom part of the system of equations on the remote processor*
- *Transmit this solution to the local machine.*
- *Solve the top part of the remaining upper triangular system on the local machine.*
- *The solution is now available at the local machine.*

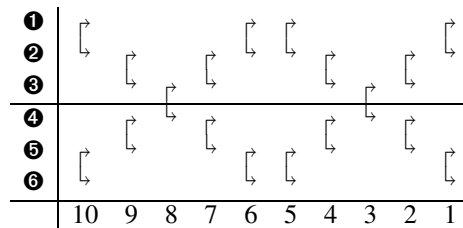
So one can clearly see that only the computation of the  $QR$ -factorization can be done in parallel. Solving the system of equations via backward substitution needs to be done first at the remote processor, and then this result needs to be sent to the local processor for computing the complete solution.

The backward substitution step is of the order  $O(n^2)$ , just like computing the  $QR$ -factorization itself. Even though it is possible to parallelize the backward substitution, we do not do so, because it becomes too fine grained. To parallelize it, one has to send immediately every computed value  $x_i$  from the remote processor to the local processor, who can then already use this to compute the solution of the upper part, by making the appropriate subtraction. Being able to halve the complexity of computing the  $QR$ -factorization has an important impact on the complexity of the global solver, as the computation of the  $QR$ -factorization is the most time consuming operation, as we will see in the numerical experiments.

## 6 General structured rank matrices

We will not go into the details on possible computations of  $QR$ -factorizations for higher order structured rank matrices. Results on how to compute these factorizations as well as on the existence of different patterns can be found in [4, 17, 18]. This is based on the fact that a rank  $r$  matrix can be written as the sum of  $r$  rank 1 matrices and similarly a matrix of semiseparability rank  $r$  can be written as the sum of  $r$  matrices of semiseparability rank 1. More information on this subject can be found e.g. in [19]. Because of limit of space, we will not go into these details.

In this section we will briefly illustrate what happens with a quasiseparable matrix of rank 2 if we want to implement the  $QR$ -factorization in a parallel way. The scheme we will follow, to bring our quasiseparable matrix to upper triangular form is the following.



The matrix we will consider is of the following form:

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \square & \times & \times & \times & \times & \times \\ \square & \square & \times & \times & \times & \times \\ \hline \square & \square & \square & \times & \times & \times \\ \square & \square & \square & \square & \times & \times \\ \square & \square & \square & \square & \square & \times \end{bmatrix}$$

In the subsequent schemes we denote the parts of the matrix satisfying the rank 2 condition by  $\square$ , the parts of rank 1 by  $\boxtimes$  and the other unstructured elements by  $\times$ .

Eventhough we show the action of both processors at the matrix in one figure, there is a line shown in the matrix to distinguish between the two processors.

We start working at the bottom and perform an upgoing rank decreasing sequence of Givens operations and on the top a descending rank expanding sequence of Givens operations. This leads to the following result.

$$\begin{bmatrix} \square & \times & \times & \times & \times & \times \\ \square & \square & \times & \times & \times & \times \\ \square & \square & \square & \times & \times & \times \\ \hline \square & \square & \square & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times & \times \end{bmatrix}$$

After these transformations communication between both processors is necessary and a Givens transformation intersecting the middle line is performed. This leads to the following situation:

$$\begin{bmatrix} \square & \times & \times & \times & \times & \times \\ \square & \square & \times & \times & \times & \times \\ \square & \square & \square & \times & \times & \times \\ \hline \boxtimes & \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times & \times \end{bmatrix}$$

After the communication step we perform an upgoing rank decreasing sequence of Givens transformations on the top three rows, and a rank expanding sequence of Givens transformations on the bottom three rows. This gives us the following matrix:

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \boxtimes & \times & \times & \times & \times & \times \\ \boxtimes & \boxtimes & \times & \times & \times & \times \\ \hline \boxtimes & \boxtimes & \boxtimes & \times & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times & \times \\ \boxtimes & \boxtimes & \boxtimes & \boxtimes & \boxtimes & \times \end{bmatrix}$$

It is clear that the resulting matrix is a quasiseparable matrix of quasiseparability rank 1. For this matrix we can easily compute the  $QR$ -factorization in a parallel way. In fact we did not even prove the existence of a parallel  $QR$ -factorization, but we also showed a constructive way for transforming the lower part from rank 2 quasiseparable form to rank 1 quasiseparable form.

## 7 Numerical examples

In the next subsections some results are presented concerning the accuracy and speed of the  $QR$ -factorization and the solver based on this factorization.

## 7.1 Accuracy of the $QR$ -factorization

Before showing the parallel timings, we will first present some results concerning the accuracy of this new  $\chi$ -pattern for computing the resulting  $QR$ -factorization. We ran examples on arbitrary random quasiseparable matrices for which the sizes range from 1000 until 9000. The case of  $n = 9000$  reached the memory limit of our machine, taking into consideration that also the original matrix had to be stored to compare the backward error. We will see in upcoming numerical examples, that we can go beyond  $n = 9000$  when computing in parallel. For every problem dimension 5 examples were considered. The backward relative error measure considered was the following one:

$$\|A - QR\|_1 / \|A\|_1,$$

in which the  $QR$  factorization was computed based on the  $\chi$ -pattern. In figure 1 (left), the line represents the average error, whereas the separate stars represent the independent errors of each experiment separately.

The plotted figure clearly illustrates the numerical backward stability of computing the  $QR$ -factorization.

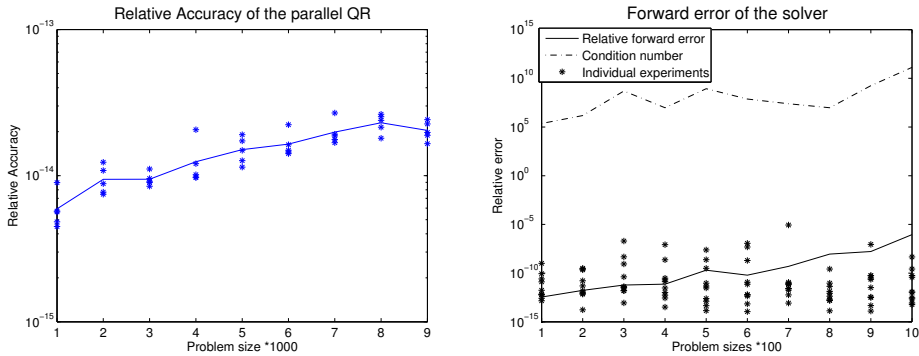


Figure 1: Backward error of the  $QR$ -factorization and forward error of the solver.

## 7.2 Accuracy of the solver

Upper triangular random matrices are known to be extremely ill conditioned [20, 21], as the upper triangular part of the quasiseparable matrix is random in our examples, this also has a large influence on the conditioning of the quasiseparable matrix.

To reduce the ill-conditioning of these matrices we included a kind of smoothing factor, such that the elements further away from the diagonal gradually become smaller. This procedure bounds the condition number between  $10^5$  and  $10^8$ . In the upcoming experiments we needed to take the condition number into consideration and we computed the following forward error:

$$\frac{\|x - \tilde{x}\|}{\|x\|},$$

which should be bounded by the machine precision multiplied with the condition number.

Due to the computational overhead, caused by generating the test matrices and computing the condition number, the problem sizes of the problems involved are limited. Figure 1 (right) shows the average condition number of the examples ran, the individual forward error of each experiment and an average forward error.

## 7.3 Timings

In this section, results concerning the speed, memory division between the processors and the total number of data to be transmitted are shown.

We know that the overall complexity of the solver is of the order  $O(n^2)$ . In the following figure we compare the cost of computing the  $QR$ -factorization w.r.t. the time needed to solve the system via backward substitution. The figure clearly shows that the time needed for computing the  $QR$ -factorization dominates

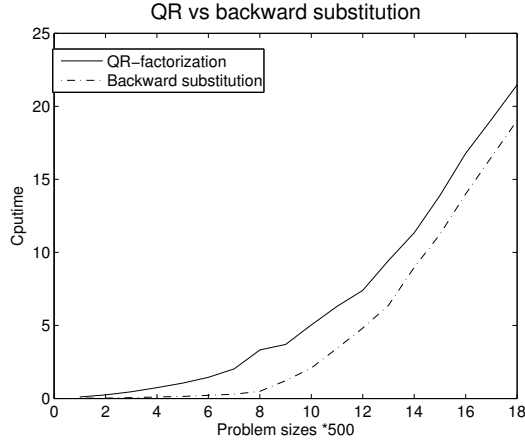


Figure 2: Speed comparisons between the  $QR$ -factorization and backward substitution.

the overall computational cost. Being able to halve the time needed for computing the  $QR$ -factorization, will reduce the overall computing time with more than 25%.

In the following figure we present timings, related to the cputime needed by one processor, in case of the standard  $QR$ -factorization and by two processors in case the algorithm is run in parallel. The presented global timings do not include timings related to the message passing. Because MATLAB MPI uses file I/O for communication. This creates false timings for message passing depending on the mounting type of the file system, the current read/write speed, and so on.

The timings presented here present the actual cputime needed by each of the processors for computing their part of the  $QR$ -factorization. In the following figure three lines are shown, representing the cputime needed by 1 processor in case the  $QR$ -factorization is computed in the traditional way on one processor. The two other lines indicates the cputime of the processors in case the method is run in parallel. One can clearly see that the non-parallel version needs much more computing time. Also important to remark is that we can solve much larger systems of equations, when considering a work load division over two processors. In this left figure (Figure 3) we chose a fixed work load division namely  $n_1 = n/2.9$  rounded to the closest, larger integer and  $n_2 = n - n_1$ , we can clearly see in the following graph that the workload is not equally distributed. In the right we chose  $n_1 = n/4.5$ , and see a more equally distributed workload.

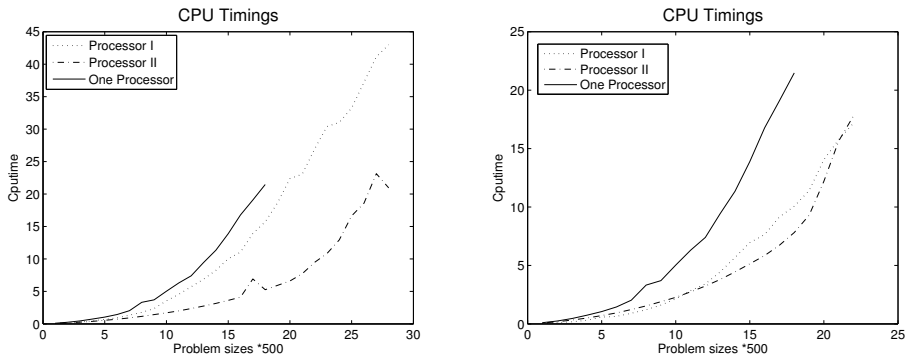


Figure 3: Cputime comparisons  $n_1 = n/2.9$  and  $n_1 = n/4.5$ .

The following table will present some results concerning global problem size ( $n$ ), memory division ( $Mem\ PI$  and  $Mem\ PII$ ), size of the problems ( $n_1$  and  $n_2$ ), and the number of double precision numbers to be transmitted ( $transfer$ ) over the network for computing the  $QR$ -factorization. These numbers are related to the timings with the left figure of Figure 3.

$n$	$n_1$	$n_2$	Mem PI	Mem PII	Transfer
500	173	327	71967	54282	656
1000	345	655	286349	216150	1312
1500	518	982	644132	484617	1966
2000	690	1310	1143674	861325	2622
2500	863	1637	1787272	1343977	3276
3000	1035	1965	2571974	1935525	3932
3500	1207	2293	3499092	2634657	4588
4000	1380	2620	4571249	3438750	5242
4500	1552	2948	5783527	4352722	5898
5000	1725	3275	7141499	5371000	6552
5500	1897	3603	8638937	6499812	7208
6000	2069	3931	10278791	7736208	7864
6500	2242	4258	12065322	9075927	8518
7000	2414	4586	13990336	10527163	9174
7500	2587	4913	16062682	12081067	9828
8000	2759	5241	18272856	13747143	10484
8500	2932	5568	20631017	15515232	11138
9000	3104	5896	23126351	17396148	11794
9500	3276	6224	25764101	19384648	12450
10000	3449	6551	28550821	21474178	13104
10500	3621	6879	31473731	23677518	13760
11000	3794	7206	34546266	25981233	14414
11500	3966	7534	37754336	28399413	15070
12000	4138	7862	41104822	30925177	15726
12500	4311	8189	44605916	33550333	16380
13000	4483	8517	48241562	36290937	17036
13500	4656	8844	52028471	39130278	17690
14000	4828	9172	55949277	42085722	18346

The table clearly shows that the total amount of data to be transmitted is small with respect to the total amount of data stored in the memory of both systems.

## 8 Concluding remarks

In this manuscript we showed the existence of another type of Givens transformation, which creates rank 1 blocks instead of zeros. Based on the shift-through lemma we showed that it is possible to change the order of Givens transformations. This resulted in the change of Givens transformations from zero creating to rank expanding. Based on several elimination patterns, involving both zero creating and rank expanding transformations, we were able to develop a parallel  $QR$ -factorization for quasiseparable matrices. Also some indications were given on how to use these results for higher order structured rank matrices. Numerical experiments were presented showing the speed and accuracy of the presented method.

## References

- [1] 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.

- [2] S. Chandrasekaran and M. Gu. Fast and stable algorithms for banded plus semiseparable systems of linear equations. *SIAM Journal on Matrix Analysis and its Applications*, 25(2):373–384, 2003.
- [3] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen. Fast stable solver for sequentially semi-separable linear systems of equations. *Lecture Notes in Computer Science*, 2552:545–554, 2002.
- [4] S. Delvaux and M. Van Barel. A QR-based solver for rank structured matrices. Technical Report TW454, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, March 2006.
- [5] R. Vandebril, M. Van Barel, and N. Mastronardi. An implicit QR-algorithm for semiseparable matrices to compute the eigendecomposition of symmetric matrices. Technical Report TW367, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, August 2003.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] R. Vandebril, M. Van Barel, and N. Mastronardi. A QR-method for computing the singular values via semiseparable matrices. *Numerische Mathematik*, 99:163–195, October 2003.
- [10] 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.
- [11] Y. Eidelman and I. C. Gohberg. On a new class of structured matrices. *Integral Equations and Operator Theory*, 34:293–324, 1999.
- [12] P. Dewilde and A.-J. van der Veen. Inner-outer factorization and the inversion of locally finite systems of equations. *Linear Algebra and its Applications*, 313:53–100, February 2000.
- [13] D. Bindel, C. J. Demure, W. Kahan, and O. Marques. On computing Givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software*, 28(2):206–238, June 2002.
- [14] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
- [15] 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.
- [16] M. Van Barel, D. Fasino, L. Gemignani, and N. Mastronardi. Orthogonal rational functions and structured matrices. *SIAM Journal on Matrix Analysis and its Applications*, 26(3):810–829, 2005.
- [17] Y. Eidelman and I. C. Gohberg. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra and its Applications*, 343-344:419–450, April 2002.
- [18] P. Dewilde and A.-J. van der Veen. *Time-varying systems and computations*. Kluwer Academic Publishers, Boston, June 1998.
- [19] H. J. Woerdeman. The lower order of lower triangular operators and minimal rank extensions. *Integral Equations and Operator Theory*, 10:859–879, 1987.

- [20] L. N. Trefethen and M. Embree. *Spectra and pseudospectra: The behaviour of nonnormal matrices and operators*. Princeton University Press, 2005.
- [21] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.