

# Unitary rank structured matrices

*Steven Delvaux*      *Marc Van Barel*

*Report TW 464, July 2006*



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

# Unitary rank structured matrices

*Steven Delvaux*      *Marc Van Barel*

*Report TW 464, July 2006*

Department of Computer Science, K.U.Leuven

## **Abstract**

In this paper we describe how one can represent a unitary rank structured matrix in an efficient way as a product of unitary or Givens transformations. We provide also some basic operations for manipulating the representation, such as the transition to zero-creating form, the transition to a unitary/Givens-weight representation, as well as an internal pull-through process of the two branches of the representation. Finally, we characterize how to determine the ‘shift’ correction term to the rank structure, and we provide some applications to this result.

**Keywords :** unitary matrix, rank structured matrix, Givens transformation, pull-through operation, shift correction term.

**AMS(MOS) Classification :** Primary : 65F25, Secondary : 15A03, 15A21.

# UNITARY RANK STRUCTURED MATRICES

STEVEN DELVAUX \*, MARC VAN BAREL \*

**Abstract.** In this paper we describe how one can represent a unitary rank structured matrix in an efficient way as a product of unitary or Givens transformations. We provide also some basic operations for manipulating the representation, such as the transition to zero-creating form, the transition to a unitary/Givens-weight representation, as well as an internal pull-through process of the two branches of the representation. Finally, we characterize how to determine the ‘shift’ correction term to the rank structure, and we provide some applications to this result.

**Keywords:** unitary matrix, rank structured matrix, Givens transformation, pull-through operation, shift correction term.

**AMS subject classifications:** 65F25, 15A03, 15A21.

**1. Introduction.** In the present paper, we consider the representation of a unitary rank structured matrix as a product of unitary or Givens transformations.

Let us start with some discussion of the literature. For the special case of a unitary Hessenberg matrix, William B. Gragg used the so-called Schur parametrization for solving the corresponding eigenvalue problem [19]. For related implicit QR-algorithms, we refer the reader to [21, 9, 26, 27]. This Schur parametrization can also be used to derive efficient algorithms in the context of orthogonal polynomials on the unit circle (Szegő polynomials), least squares approximation using trigonometric polynomials, the construction of Gaussian quadrature on the unit circle, frequency estimation, ... [20, 2, 3].

A generalization to unitary block-Hessenberg matrices and orthonormal polynomial vectors can be found in [22, 23, 8, 24]. The size of the blocks of the matrix determines the degree structure of the sequence of corresponding orthonormal polynomial vectors.

Instead of unitary banded matrices, one could also consider unitary rank structured matrices of a more general type. We use the following definition.

**DEFINITION 1.** (See [14]:) We define a pure rank structure  $\mathcal{R}$  on  $\mathbb{C}^{m \times n}$  as a collection of so-called pure structure blocks  $\mathcal{R} = \{\mathcal{B}_k\}_k$ . Each pure structure block  $\mathcal{B}_k$  is characterized as a 3-tuple

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

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

$$\text{Rank}A(i_k : m, 1 : j_k) \leq r_k.$$

---

\*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.

The above definition uses the word *pure* to distinguish from the more general rank structures involving a ‘shift’ correction term [15]. Since these more general rank structures make their appearance only in Section 6, in the first part of this paper we will often simplify notation by just dropping the word *pure* everywhere from the notation.

Note that by definition, all structure blocks have to start from the lower left matrix corner. In practice, it often happens that also the block *upper* triangular part is rank structured, i.e., that also the matrix  $A^T$  satisfies rank structure in the sense of Definition 1. By abuse of notation, we will indiscriminately use the term *rank structure* also in this case.

Chapter 14 of the book [16] describes a decomposition of certain unitary rank structured matrices in terms of individual Givens transformations. But the scope in the latter reference is rather limited. In contrast, in the present paper we show how to build up the Givens product representation for a general unitary rank structured matrix, including the way how to manipulate it or bring it to canonical form in a stable way.

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 structure blocks to the main diagonal, and where  $n$  denotes the matrix size.

In recent years, some interest has grown also in the manipulation of matrices which are unitary plus some low rank correction term [6, 7, 1, 5]. We refer to subsequent work for a treatment of these matters.

The remainder of this paper is organized as follows. Section 2 recalls the main ideas of the Givens-weight representation for rank structured matrices, not requiring unitarity. Section 3 considers an alternative representation for a unitary rank structured matrix in the form of a unitary or Givens product. Section 4 shows how the unitary/Givens product representation can be transformed into a unitary-weight representation. Section 5 deals with the internal pull-through process of the two branches of the unitary/Givens product representation. Section 6 treats some topics concerning the shift correction term to the rank structure and its relation to the representation.

**2. Givens-weight representation for rank structured matrices.** In this section we recall the main ideas of the Givens-weight representation. We should emphasize that this section serves only as a reminder of the main ideas of [11]; the reader not familiar with these ideas is referred to [11] for more details.

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.

We will assume in what follows that we are working with a pure rank structure  $\mathcal{R}$  for which there are no structure blocks that are ‘contained’ in each other, i.e., for which the structure blocks  $\mathcal{B}_k$  can be ordered such that both their row and column indices  $i_k$  and  $j_k$  increase in a strictly monotonic way. Moreover, we will work here with a *general* rank structured matrix  $A$ , i.e., the matrix does not have to be unitary

at this moment.

DEFINITION 2. (*Unitary-weight representation. See [11].*) Let  $A \in \mathbb{C}^{m \times n}$  be a matrix satisfying a pure rank structure  $\mathcal{R} = \{\mathcal{B}_k\}_{k=1}^K$ , where the structure blocks are ordered from top left to bottom right. A unitary-weight representation of the matrix  $A$  according to the structure  $\mathcal{R}$  consists of a pair  $(\{U_k\}_{k=1}^K, W)$ . Here the  $U_k$ ,  $k = K, \dots, 1$  form a sequence of localized unitary transformations proceeding from bottom to top of the matrix, as indicated by the fat arrows in Figure 2.1, serving to create zeros in the subsequent  $\text{Rk } r_k$  structure blocks  $\mathcal{B}_k$  except for their top  $r_k$  rows. On the other hand, the matrix  $W \in \mathbb{C}^{m \times n}$  is called the weight matrix, and it contains the blocks of elements  $W_k$  obtained at the top border of the rank structure at the moment just after applying  $U_k$ . See Figure 2.1.

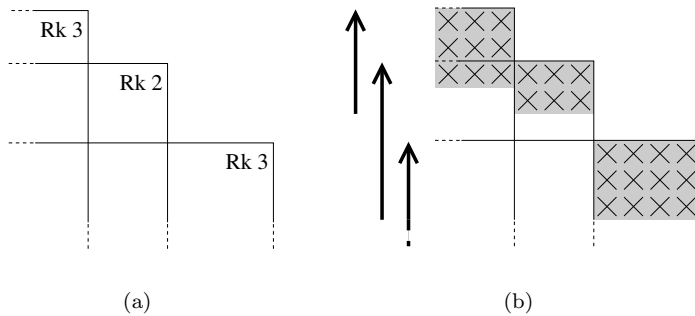


FIGURE 2.1. For the pure rank structure in the left picture, the right figure shows a schematic picture of the unitary-weight representation. The notation ‘Rk  $r$ ’ denotes that the structure block is of rank at most  $r$ .

The basic idea of this definition is to compress the given rank structured matrix by means of subsequent localized unitary transformations, hereby proceeding from bottom to top of the matrix, and storing each time the elements just before they reach the top border of the rank structure.

Note that this definition leads to an *internal* representation of the rank structure, in the sense that it involves no information about the matrix part lying outside the reach of the structure blocks. Moreover, it implies that each unitary operation  $U_k$  has a certain *action radius* in the sense that it acts only on a limited number of columns. This action radius is a monotonically decreasing function when the  $U_k$  proceed from bottom to top of the matrix. For example, the middle unitary operation in Figure 2.1 has action radius spanning over the first six columns of the depicted matrix. On the other hand, the last four columns of this matrix do not belong to this action radius anymore, since they have not been influenced by this middle unitary operation during the compression process.

If a unitary-weight representation of a matrix is given, we can restore the full matrix by *spreading out* the representation. This means that we gradually consider the subsequent weight blocks, proceeding from top to bottom of the matrix, and multiply them with the ‘decompressing’ unitary operations  $U_k^{-1}$ , each one acting of course only on the columns on the left of its action radius. While this process proceeds from top to bottom of the matrix, we will gradually retrieve the original, full matrix which we started from.

We can now specify from unitary-weight to Givens-weight representations. In

what follows, we will use the term *Givens transformation* to denote a 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.

Rather than individual Givens transformations, it will be useful to work with *Givens arrows*: these are defined as collections of subsequent Givens transformations, each of them having row index precisely one more than the previous one. This means that each Givens transformation is situated precisely one position below the previous one: see Figure 2.2.

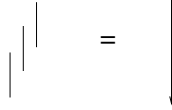


FIGURE 2.2. A Givens arrow consisting of 3 Givens transformations. Concerning this figure, we recall the reader that we consider each Givens transformation as ‘acting’ on the rows of an (invisible) matrix standing on the right of it, and hence that the Givens transformations in the figure should be evaluated from right to left, hereby explaining the downward direction of the Givens arrow.

The number of Givens transformations of which a Givens arrow consists will be called the *width* of the Givens arrow. Moreover, we define the *top* and the *tail* of the Givens arrow to be the largest and the smallest row index of the Givens transformations of which the Givens arrow consists, respectively. These notions have an obvious graphical interpretation.

DEFINITION 3. (*Givens-weight representation. See [11]:*) Let  $A \in \mathbb{C}^{m \times n}$  be a matrix satisfying a pure rank structure  $\mathcal{R} = \{\mathcal{B}_k\}$ , where the structure blocks are ordered from top left to bottom right. A Givens-weight representation of  $A$  according to the structure  $\mathcal{R}$  is a unitary-weight representation where additionally each unitary component  $U_k$  is decomposed into a product of Givens arrows, such that

- each of the Givens arrows has width at most  $r_k$ ,
- both the tops and the tails of the subsequent Givens arrows of each  $U_k$  are monotonically proceeding upwards. For the tails, we assume that this monotonicity is strict.

See Figure 2.3.

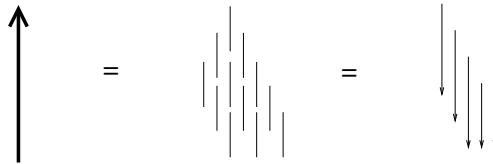


FIGURE 2.3. Suppose that the current structure block is  $\text{Rk } 3$ , and that the corresponding unitary transformation  $U_k$  spans over 6 rows. Then we assume for this unitary transformation a decomposition into a product of Givens arrows of width at most 3.

We should still explain why the assumption is made that each Givens arrow in the decomposition of  $U_k$  has width at most  $r_k$ . To this end, recall that the unitary transformation  $U_k$  serves to create zeros in a certain  $\text{Rk}(r_k)$  submatrix, except for its top  $r_k$  rows. No matter if we do this by means of a singular value decomposition or by a pivoted QR-factorization or any other unitary operation, this effect can always

be realized by a succession of Givens arrows as prescribed.

Note that by decomposing each unitary transformation  $U_k$  as specified in Definition 3, we formally obtain a decomposition into a product of *too many* Givens transformations, in the sense that the beginning and trailing Givens transformations of two subsequent unitary transformations  $U_k$  may overlap. It was shown in [11] how these superfluous parameters can be removed from the representation; see also in what follows.

For further use, we recall also the following result.

LEMMA 4. (*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 2.4.

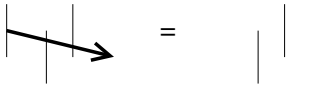


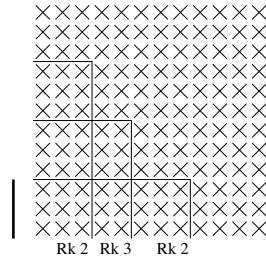
FIGURE 2.4. *Pull-through lemma applied in the downward direction. One could imagine that the leftmost Givens transformation is really ‘pulled through’ the two rightmost Givens transformations.*

**3. Representation for a unitary rank structured matrix.** Given a unitary rank structured matrix, one could use the unitary/Givens-weight representation of Section 2 to represent it. Indeed, the reason why this is possible is that for a unitary rank structured matrix, the rank structure in the lower triangular part induces rank structure in the *upper* triangular part as well (Section 6), so that one can represent both the lower and upper triangular part efficiently by means of a unitary/Givens-weight representation.

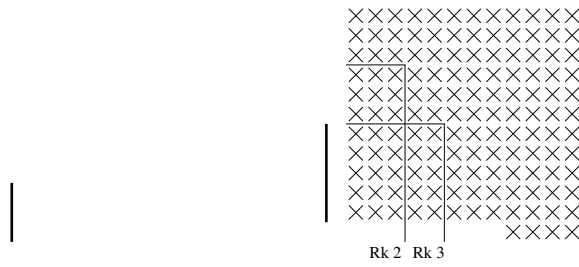
A drawback of the unitary/Givens-weight representation is that the structured lower and upper triangular parts must be represented separately. Hence, the unitarity of the matrix can not be ‘embedded’ in the representation, and hence it may be expected that this property will get weakened when one performs practical algorithms to the representation, due to round-off errors.

A solution to this drawback is to use a representation based on a product of localized unitary or Givens transformations, to be described in the present section.

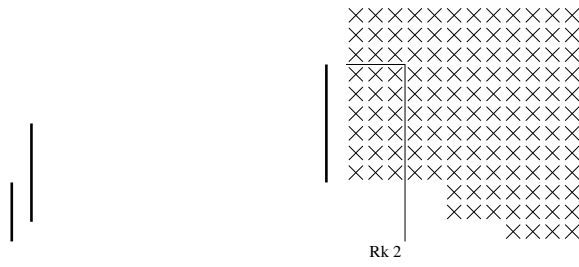
The idea of the representation is straightforward and will be described next. We start computing a QR-factorization of the given unitary rank structured matrix  $A$ . Due to the rank structure, the Q-factor of the QR-factorization can be represented as a product of a limited number of localized unitary or Givens transformations: see e.g. [14, 13]. Moreover, by suitable choice of normalization it can be assumed that the R-factor of the QR-factorization has positive diagonal elements. But then this R-factor is a unitary upper triangular matrix with positive diagonal elements, so that it must be the identity matrix. Stated in another way, the QR-equation  $A = QR$  reduces to  $A = Q$ , i.e., the given unitary rank structured matrix equals the Q-factor of its QR-factorization. In particular, it will have a decomposition as a product of a limited number of localized unitary or Givens transformations.



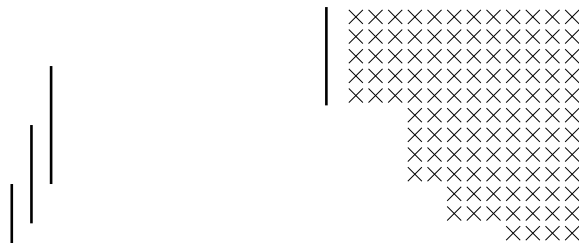
(a)



(b)

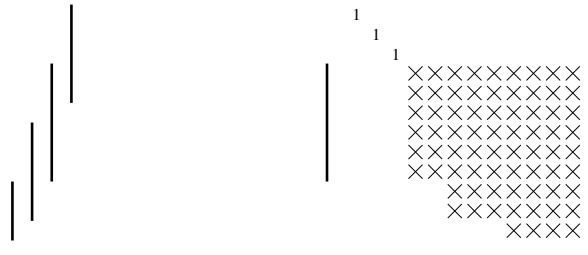


(c)

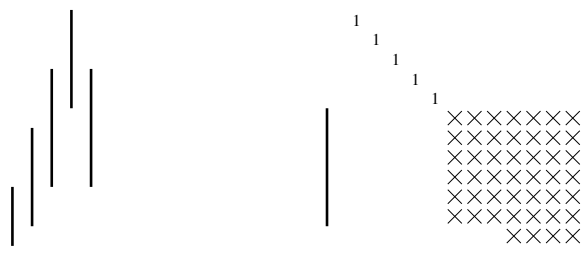


(d)

FIGURE 3.1. *building up the unitary product representation (a-d).*



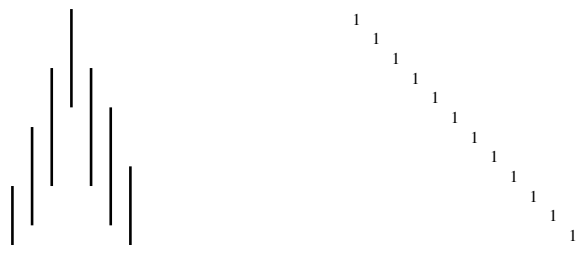
(e)



(f)



(g)



(h)

FIGURE 3.1. *building up the unitary product representation (e-h).*

These ideas are illustrated for a particular example of rank structure in Figure 3.1.

Let us comment on this figure. First, Figure 3.1(a-c) starts applying consecutive unitary operations to compress the subsequent  $\text{Rk } r$  structure blocks, except for their top  $r$  rows, as in Section 2. Note that this process moves upwards.

Second, Figure 3.1(d-h) applies a sequence of unitary operations to annihilate the remaining non-zero elements below the main diagonal. Note that this process moves downwards. By the mechanism already mentioned, the upper triangular part will vanish too under the action of these operations. By using a suitable normalization, one can then succeed in reducing the unitary matrix to the identity matrix, so that we will have computed a matrix  $Q^H$  for which  $Q^H A = I$ , the identity matrix.

We should still explain the meaning of the operations in the left part of each of the subfigures of Figure 3.1. These operations denote the representation of the matrix  $Q$  which is built up during the compression process. In particular, these operations contain the Hermitian transposes of the applied compression operations of  $Q^H$ , so that they are *not* the compression operations themselves.

The representation of the matrix  $Q$  resulting at the end of this process is summarized in Figure 3.2.

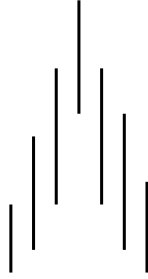


FIGURE 3.2. *Final unitary product representation.*

Concerning Figure 3.2, a first observation could be that the unitary product representation consists of two *branches* of unitary operations.

First, the left branch of the representation contains 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 (cf. Figure 3.1). In particular, the left branch contains information about the ranks of the given rank structure, in just the same way as this was the case for the unitary-weight representation in Section 2. (The difference is again that the matrix  $Q$  consists of the *decompressing* operations, i.e., the Hermitian transposes of the unitary operations used to compress the structure. This should be contrasted to the unitary-weight representation where we preferred to work with the *compressing* unitary operations, as in Figure 2.1).

Second, the right branch of the representation in Figure 3.2 contains the Hermitian transposes of the operations serving to annihilate the remaining non-zero elements below the main diagonal. This branch is a kind of unitary analogue of the *weight matrix* of the unitary-weight representation of Section 2. It reflects information not only about the ranks, but also about the size of the structure blocks of the given rank structure.

Let us formalize these ideas.

DEFINITION 5. (*Unitary product representation:*) Let  $Q \in \mathbb{C}^{n \times n}$  be a unitary matrix satisfying a certain rank structure  $\mathcal{R} = \{\mathcal{B}_k\}_{k=1}^K$ , where the structure blocks  $\mathcal{B}_k : (i_k, j_k, r_k)$  are ordered from top left to bottom right. Then a unitary product representation for this unitary matrix is a product of the form

$$Q = U_{K,\text{left}} \dots U_{1,\text{left}} U_0 U_{1,\text{right}} \dots U_{K,\text{right}}, \quad (3.1)$$

where the  $k$ th unitary components of the right and left branches act respectively on the rows

$$\begin{cases} U_{k,\text{right}} : j_k + 1, \dots, i_{k+1} + r_{k+1} - 1, \\ U_{k,\text{left}} : i_k, \dots, i_{k+1} + r_{k+1} - 1, \\ U_0 : 1, \dots, i_1 + r_1 - 1. \end{cases} \quad (3.2)$$

Here we assume a trivially satisfied structure block  $\mathcal{B}_{K+1} : (n+1, n, 0)$  to be added to the structure. See Figure 3.2.

In the sequel, the unitary product representation (3.1) will sometimes be called  $\Lambda$ -shaped. The reason for this terminology should be clear from Figure 3.2.

Keeping in mind the construction of (3.1) (cf. Figure 3.1), the correctness of the index sets (3.2) can be easily checked. For example, the index sets for the left branch can be read off immediately from Figure 2.1.

Now we will further fine-tune the unitary product representation.

DEFINITION 6. (*Givens product representation:*) Under the same conditions as in Definition 5, a unitary product representation is called a Givens product representation if each unitary component  $U_{k,\text{right}}$  of the right branch is given as a product of downward pointing Givens arrows such that

- the tails of the subsequent Givens arrows of each  $U_{k,\text{right}}$  are strictly monotonically proceeding upwards.

Moreover, it is assumed that each unitary component  $U_{k,\text{left}}$  of the left branch has a decomposition into a product of upward pointing Givens arrows, such that

- each of the Givens arrows has width at most  $r_k$ ,
- both the tops and the tails of the subsequent Givens arrows of each  $U_{k,\text{left}}$  are monotonically proceeding downwards. For the tops, we assume that this monotonicity is strict.

See Figures 3.3 and 3.4.



FIGURE 3.3. Givens factorization of a unitary transformation of the right branch of a Givens product representation. Note that each unitary matrix allows such a factorization.

Note that the above condition on the right branch is not really a restriction. Indeed, every unitary transformation  $U$  has a decomposition as in Figure 3.3, as is easily shown (see also in what follows).

On the other hand, the above condition on the left branch is an exact analogue of the definition of Givens-weight representation in Section 2. The difference is again



FIGURE 3.4. *Givens factorization of a unitary transformation of the left branch of a Givens product representation. Note that this figure is precisely transposed to Figure 2.3.*

that we work here with the decompressing, rather than the compressing unitary operations, which causes all the notions for Givens-weight representations to be Hermitian transposed. This corresponds graphically to a vertical reflection of all the figures w.r.t. those of the Givens-weight representation, cf. Figures 2.3 and 3.4.

Let us show how an arbitrary unitary product representation can be brought into Givens product form. This will be achieved by transferring the superfluous Givens transformations from the left to the right branch: see Figure 3.5.

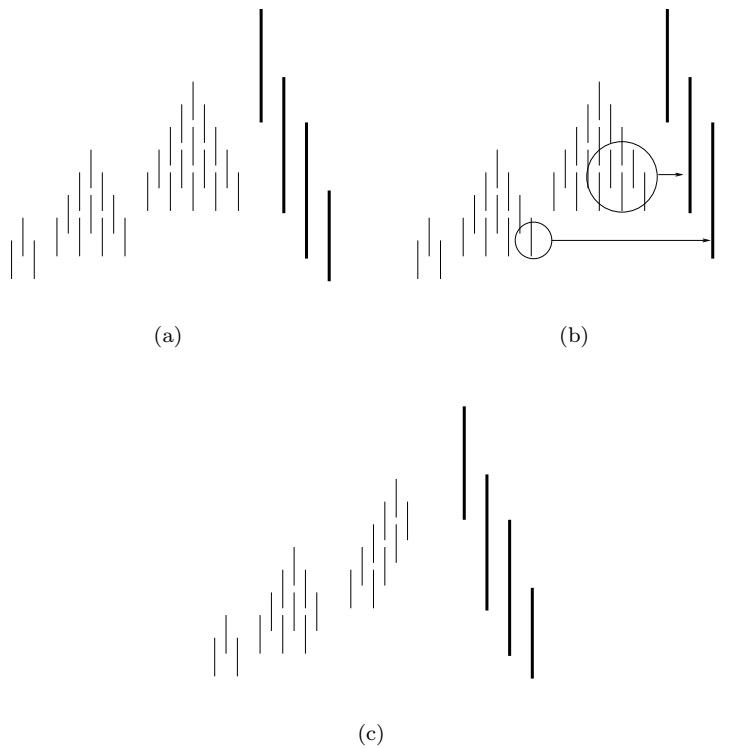


FIGURE 3.5. *Bringing the left branch of the representation in the form required by the definition of Givens product representation.*

Let us comment on this figure. Figure 3.5(a) shows the starting situation, where the operations of the left branch of the representation are shown in terms of their individual Givens transformations. We assume that each of them is given as a *full* decomposition in the style of Figure 3.3.

Now we transfer from  $U_{k,\text{left}}$  to  $U_{k,\text{right}}$  as many downward pointing Givens arrows as possible, i.e., as long as no Givens arrow is encountered which has a row in common with the unitary operation  $U_{k-1,\text{left}}$ , and this for each  $k$ : see Figures 3.5(b) and 3.5(c).

It can be checked that from the decomposition of  $U_{k,\text{left}}$  as a product of downward pointing Givens arrows as in Figure 3.3, only those Givens arrows with tails  $i_k, \dots, i_k + r_k - 1$  will remain after the transfer process. Indeed, this could be verified by using the index sets in (3.2). It follows that, if we refactorize  $U_{k,\text{left}}$  as a product of *upward* pointing Givens arrows, then each of these Givens arrows has width at most  $r_k$ , which is indeed the correct value required in Figure 3.4.

This ends the description of the transition algorithm from a unitary into a Givens product representation.

We point out that the above reduction algorithm allows an analogue in case of the unitary/Givens-weight representation, by replacing the role of the right branch of Givens transformations by the *weight matrix* of the representation. The superfluous Givens transformations can then just be removed, without any practical computation, by the fact that they do not act on any weight of the internal Givens-weight representation anymore.

Suppose now that we have available a Givens product representation as in Definition 6. It should be clear that this representation may still suffer from an overload of Givens transformations. Removing the superfluous Givens transformations will lead to a very special kind of Givens product representation, to be described next.

DEFINITION 7. (*Zero-creating:*) *Under the same conditions as in Definition 6, a Givens product representation is called zero-creating if*

- *the tails of the subsequent Givens arrows of the right branch altogether are strictly monotonically proceeding upwards;*
- *the tops of the subsequent Givens arrows of the left branch altogether are strictly monotonically proceeding downwards.*

See Figure 3.8.

It is assumed in Definition 7 that the Givens factorization of the topmost unitary operation  $U_0$  is ‘consistent’ with both monotonicity constraints in the definition, in the sense of Figure 3.8.

The process of bringing a Givens product representation into zero-creating form is illustrated in Figure 3.6.

Let us comment on this figure. The main flow of the algorithm is determined by chasing the superfluous upward pointing Givens arrows of the left branch downwards by means of the pull-through lemma. At the end of this process, we will have brought the left branch of the representation completely in zero-creating form.

For more details about this reduction method, we refer to [12], where a very similar reduction to zero-creating form has been described in terms of the Givens-weight representation. The current reduction is in fact a simplification of the latter algorithm since we have here no weight matrix to keep track of.

For the final step in the reduction process, we have to bring also the right branch of the representation in zero-creating form. This process is very similar to the corresponding process for the left branch, and it is demonstrated in Figure 3.7.

The final zero-creating Givens product representation is shown in Figure 3.8.

To motivate the terminology of zero-creating, let us have a second look at Figure 3.3. The downward pointing Givens arrows in this figure can be identified as (the Hermitian transposes of) the Givens arrows creating zeros in the subsequent columns of the unitary matrix, as can be easily seen.

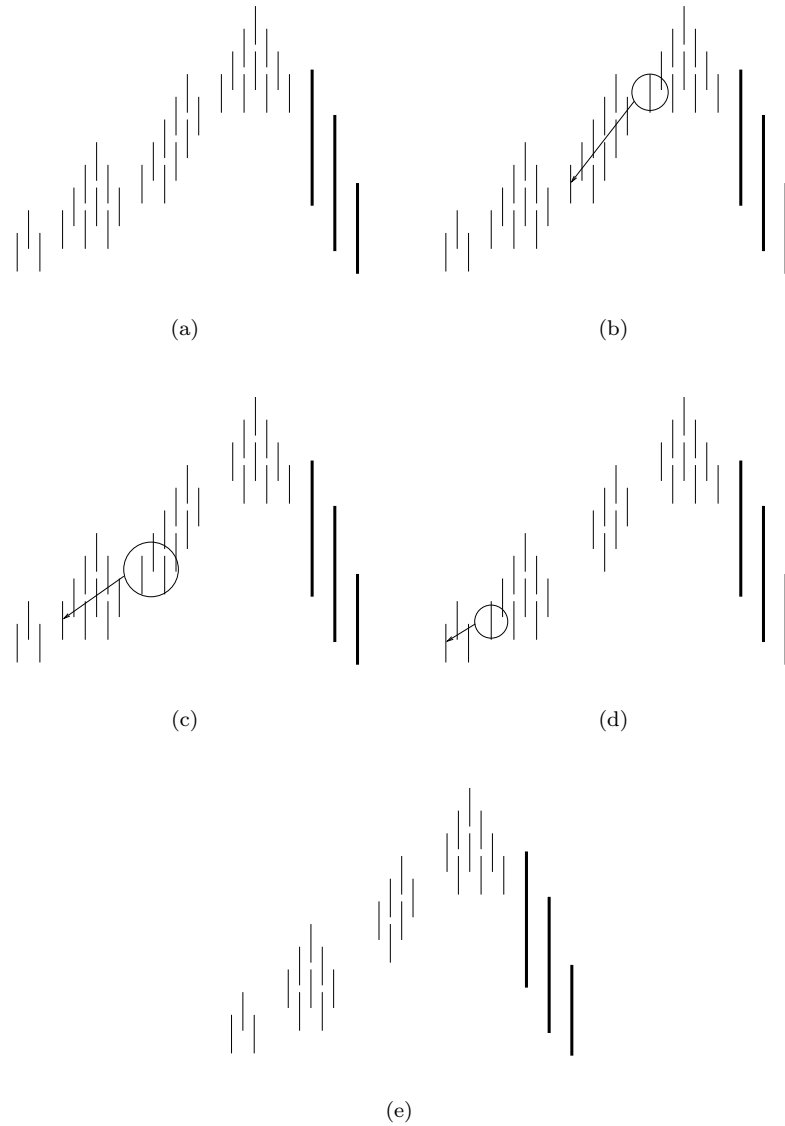


FIGURE 3.6. *Bringing the left branch of the representation in zero-creating form.*

What we have achieved above was now to show that *every* unitary rank structured matrix can be brought in the form of Figure 3.3, but now subject to an additional sparsity pattern. This sparsity can be understood by the fact that, while we start creating zeros in the first columns of the matrix, automatically zeros will be induced in some of the further columns as well, due to presence of the rank structure; see also [14]. This leads to the particular  $\Lambda$ -shape of Figure 3.8, as compared to the full  $\Delta$ -shape of Figure 3.3.

Let us point out that the right branch of Figure 3.8 is obviously thicker than the left one. This reflects the fact that the underlying rank structure of Figure 3.1(a) is

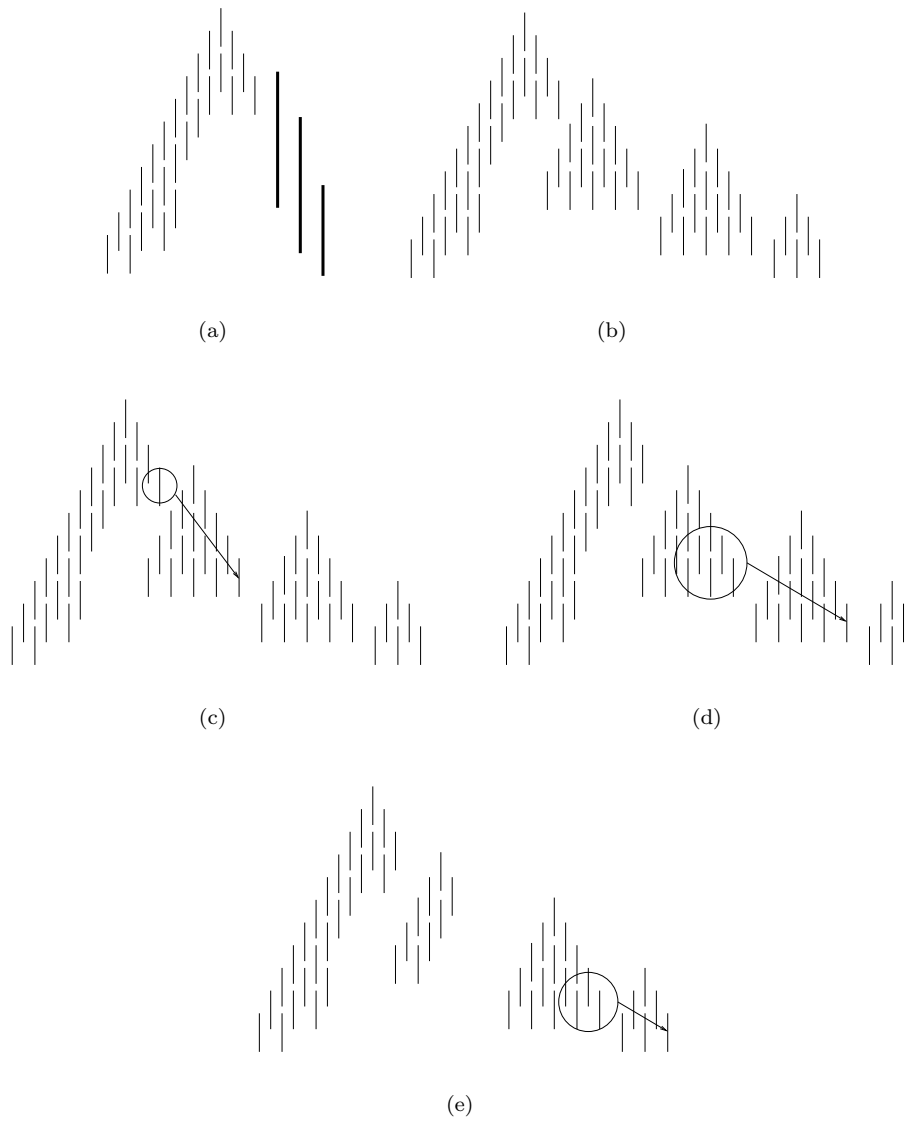


FIGURE 3.7. *Bringing the right branch of the representation in zero-creating form.*

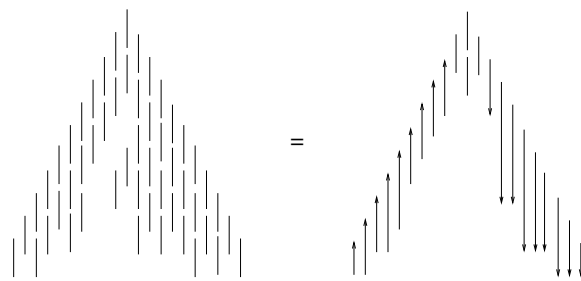


FIGURE 3.8. *Final zero-creating Givens product representation.*

such that (i) the structure lies strictly below the main diagonal, and (ii) there are huge gaps between the structure blocks. Indeed, these two facts imply that ‘many’ Givens transformations are needed for annihilating the remaining non-zero elements below the main diagonal during the QR-factorization, explaining the thickness of the right branch of the representation.

To stress this point, we show also an example where the underlying rank structure is (i) situated just below the main diagonal, and (ii) dense, in the sense that the structure blocks are following immediately one after the other. The left and right branch of the representation will then have exactly the same thickness: see Figure 3.9.

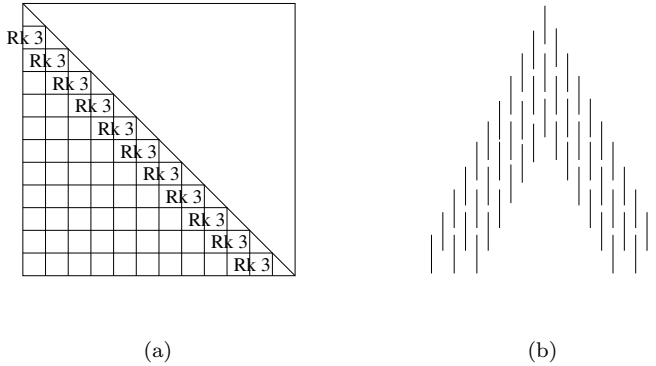


FIGURE 3.9. The figure shows (a) the induced pure rank structure for a unitary lower semiseparable plus diagonal matrix, and (b) its zero-creating Givens product representation. Note the symmetric shape of the representation.

Summarized, in this section we have described how to build the unitary and Givens product representations, and how to reduce the representation into zero-creating form. More transformation algorithms for the Givens product representation will be provided in the next sections.

**4. Transition to Givens-weight representation.** This section considers the transformation of the unitary/Givens product representation into unitary/Givens-weight form.

First, we recall a fact mentioned before, namely, that a unitary rank structured matrix could also be represented using the unitary/Givens-weight representation of Section 2. This follows by the fact that the ranks in both the lower and the upper triangular part of such a matrix are bounded (Section 6).

We will describe now a constructive procedure for obtaining a unitary-weight representation, starting from a unitary product representation (3.1). The algorithm for doing this will be of a ‘central’ type: starting from the topmost unitary component  $U_0$ , it gradually incorporates the subsequent *pairs* of transformations  $U_{k,\text{left}}$ ,  $U_{k,\text{right}}$  of the unitary product representation,  $k = 1, \dots, K$ , going from inside to outside of the unitary product factorization (3.1): see Figure 4.1.

Let us comment on this figure. Starting from the topmost unitary component  $U_0$ , the  $k$ th step of the transformation process is to multiply the already expanded unitary-weight representation on the left with  $U_{k,\text{left}}$ , and on the right with  $U_{k,\text{right}}$ .

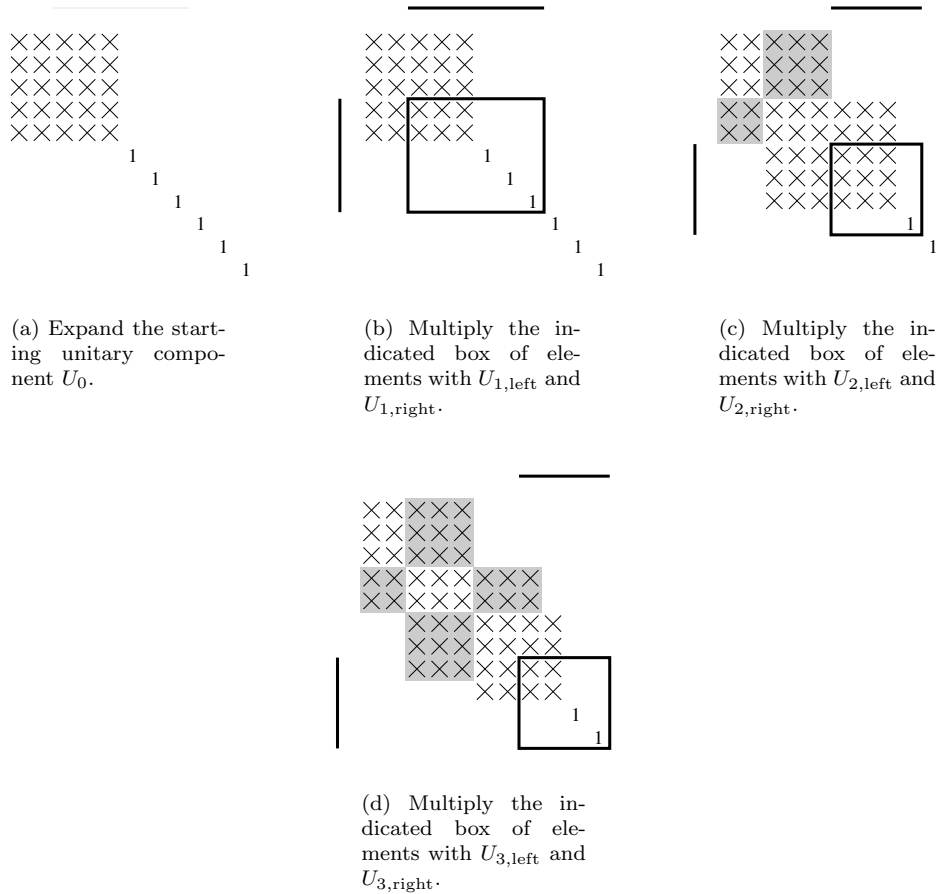


FIGURE 4.1. Transforming the unitary product representation into a unitary-weight representation.

But we do not do this on the entire matrix. Instead, we only update the box of elements spanning over the *intersection* of the rows influenced by  $U_{k,\text{left}}$ , and the columns influenced by  $U_{k,\text{right}}$ : see Figures 4.1(b-d).

It is clear that the above expansion procedure is ‘incorrect’, in the sense that to obtain the full matrix, we should still apply the unitary operations to the rows or columns on which they have not been applied yet, with each unitary component acting inside its own ‘action radius’. But this is precisely the concept of the unitary-weight representation. Hence, we see that during the above algorithm, we are gradually building up a unitary-weight representation for the given unitary rank structured matrix. This is indicated by the gradual appearance of new weight blocks in Figures 4.1(b-d), standing on a grey background.

The final unitary-weight representation is shown in Figure 4.2.

Incidentally, note that this transformation process clearly reveals that the ranks of the complementary submatrices of a unitary rank structured matrix are coupled. Indeed, it follows from this algorithm that the  $k$ th structure block in the lower triangular part has rank bounded by the number of indices shared by  $I_{k-1,\text{left}}$  and  $I_{k,\text{left}}$ , while the  $k$ th structure block in the upper triangular part has rank bounded by the

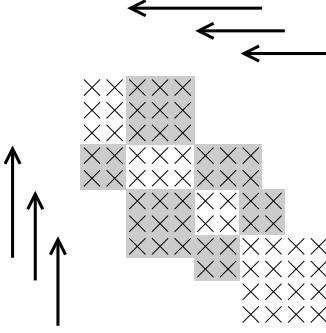


FIGURE 4.2. Final unitary-weight representation for the example in Figure 4.1.

number of indices shared by  $I_{k-1,\text{right}}$  and  $I_{k,\text{right}}$ . By means of the index sets in (3.2), this yields us the value  $r_k = \#\{i_k, \dots, i_k + r_k - 1\}$  for the rank of the structure block  $\mathcal{B}_k$  in the lower triangular part, and  $r_k + i_k - j_k - 1 = \#\{j_k + 1, \dots, i_k + r_k - 1\}$  for the corresponding structure block in the upper triangular part.

Finally, it is clear that the above reduction process can be performed also for a *Givens* product, rather than a unitary product representation. This leads then to a *Givens*-weight instead of a unitary-weight representation. The description of the algorithm remains exactly the same in this case.

**5. Internal pull-through.** In this section we describe an internal pull-through process of the two branches of the unitary/*Givens* product representation.

The following lemma will be pivotal in this respect.

LEMMA 8. (*Block pull-through lemma:*) Given a unitary  $k$  by  $k$  matrix  $Q$  which is factorized as

$$Q = A_{a,\dots,k} B_{1,\dots,b} C_{c,\dots,k}, \quad (5.1)$$

where each factor denotes a localized unitary operation, with ranges determined by the subscript indices. Then there exists a refactorization

$$Q = \tilde{A}_{1,\dots,a-1} \tilde{B}_{a-b+c-1,\dots,k} \tilde{C}_{1,\dots,c-1}, \quad (5.2)$$

provided that the range index for  $\tilde{B}$  satisfies  $a - b + c - 1 > 1$ : see Figure 5.1. In case where the latter inequality is not satisfied, the lemma does not provide any useful information.

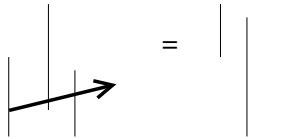


FIGURE 5.1. Example of the block pull-through lemma applied in the upward direction. One could imagine that the leftmost unitary transformation is really ‘pulled through’ the two rightmost unitary transformations, or vice versa.

PROOF. We proceed here by a constructive proof. Let us consider the matrix  $ABC$  in its full form as a  $k$  by  $k$  matrix. It follows from (5.1) that this matrix must

satisfy a certain structure block in its block lower triangular part, having coordinates  $\mathcal{B} : (i, j, r) = (a, c - 1, b - a + 1)$ . (See Figure 5.2). The nullity, i.e., the dimension of the right null space of this structure block equals

$$\text{null}_{\mathcal{B}} := c - 1 - (b - a + 1) = a - b + c - 2.$$

One can now apply a unitary transformation  $\tilde{C}_{1, \dots, c-1}^H$  to the columns, in order to compress the structure block, hereby transforming the first  $\text{null}_{\mathcal{B}}$  columns of  $\mathcal{B}$  into zeros. Next, one can apply a unitary transformation  $\tilde{A}_{1, \dots, a-1}^H$  to the rows, to extend the created block of zeros completely to the top of the matrix. The resulting matrix  $\tilde{B}$  will then be the identity in its first  $\text{null}_{\mathcal{B}}$  rows and columns; it is clear that we have obtained now the desired refactorization (5.2).  $\square$

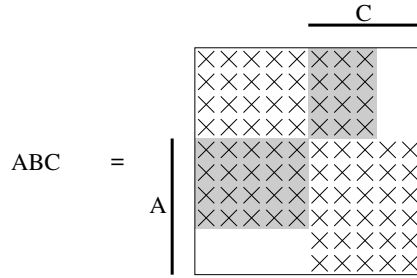


FIGURE 5.2. Via the algorithm of Section 4, one can obtain the indicated unitary-weight representation for the matrix  $ABC$  in (5.1). This representation reveals then immediately the position of the induced structure block  $\mathcal{B}$ .

Note that Lemma 8 was formulated entirely in terms of index sets. But there exists also a more geometrical formulation of this lemma: note that the outermost unitary transformations before and after the pull-through act precisely on *complementary sets*, e.g., compare the index sets of  $A$  and  $\tilde{A}$ . Also for the middle transformation  $\tilde{B}$ , one can obtain such a geometrical formulation, by remarking that it acts precisely on the indices of the original rightmost unitary transformation, to which are added  $b - a + 1$  extra indices, corresponding to the number of indices in the intersection of the two original leftmost unitary transformations: see (5.1), (5.2).

We recall that in case the new index  $a - b + c - 1$  in the statement of Lemma 8 is less than or equal to 1, the lemma does not provide any useful information. Indeed, it tells then just that the unitary matrix  $ABC$  can be refactorized as a full unitary matrix  $\tilde{B}$ , which is obvious of course; the other factors  $\tilde{A}$  and  $\tilde{C}$  could then be absorbed in this matrix as well.

Let us provide a more ‘intrinsic’ characterization of the critical equation  $a - b + c - 1 > 1$  in Lemma 8.

REMARK 9. Denoting in Lemma 8 with  $\#A := k - a + 1$ ,  $\#B := b$  and  $\#C := k - c + 1$  the number of indices on which each of the initial three unitary transformations acts, one can rewrite the critical equation  $a - b + c - 1 > 1$  as

$$\#A + \#B + \#C < 2k. \quad (5.3)$$

It is curious to note that the usual, scalar pull-through lemma is in the situation where equality in the critical bound (5.3) is achieved. Thus the block pull-through lemma is *no* generalization in the strict sense of the scalar pull-through lemma.

A generalization that is more in spirit with the scalar pull-through lemma, is the following.

LEMMA 10. (*Block pull-through lemma, second variant:*) *Given an arbitrary unitary  $k$  by  $k$  matrix  $Q$ , this matrix can be refactorized as in (5.1), where  $a$ ,  $b$  and  $c$  can be chosen to be any indices leading to equality in the critical bound (5.3).*

The proof is very similar to that of Lemma 8. Indeed, we note that the structure block needed to obtain the required factorization will be ‘trivially satisfied’ in this case, so that indeed no condition on the unitary matrix  $Q$  was needed in the statement of Lemma 10.

Note that Lemma 10 determines the block pull-through operation in a highly non-canonically determined way, in contrast to the canonical statement of Lemma 8. Therefore, for reasons of simplicity the latter will be our choice of preference in what follows.

Now we apply the block pull-through lemma (Lemma 8) to perform the internal pull-through of the two branches of the representation. See Figure 5.3.

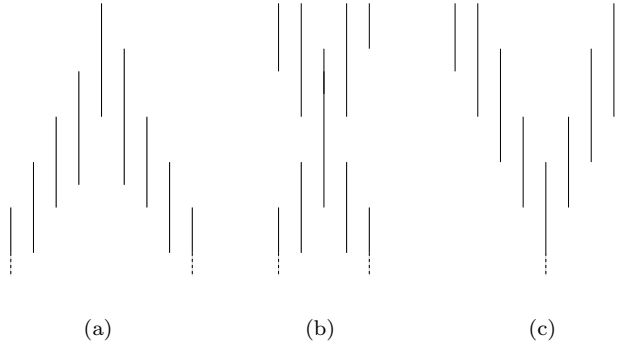


FIGURE 5.3. *Internal pull-through of the two branches of a unitary product representation.*

Concerning this figure, note that the original  $\Lambda$ -shaped factorization as in (3.1), transforms into a sequence of  $X$ -shaped factorizations under the action of the subsequent applications of the block pull-through lemma, while the central ‘bulge’ of the factorization gradually moves downwards. Some intermediate steps in this process are shown in Figure 5.3(a-c).

Note that the final unitary product representation in Figure 5.3(c) has a  $V$ -shaped appearance, as defined next.

DEFINITION 11. (*V-shaped unitary product representation:*) *Under the same conditions as in Definition 5, a V-shaped unitary product representation for the unitary matrix  $Q$  is a product of the form*

$$Q = V_{1,\text{left}} \cdots V_{K,\text{left}} V_{K+1} V_{K,\text{right}} \cdots V_{1,\text{right}}, \quad (5.4)$$

where the  $k$ th unitary component of the left and right branch act respectively on the rows

$$\begin{cases} V_{k,\text{left}} : j_{k-1} - r_{k-1} + 1, \dots, i_k - 1, \\ V_{k,\text{right}} : j_{k-1} - r_{k-1} + 1, \dots, j_k, \\ V_{K+1} : j_K - r_K + 1, \dots, n. \end{cases} \quad (5.5)$$

We assume here a trivially satisfied structure block  $\mathcal{B}_0 : (1, 0, 0)$  to be added to the structure. See Figure 5.3(c).

One can show the correctness of the index sets (5.5) by an induction argument. Indeed, suppose that we are at the point of applying the block pull-through lemma with  $A := U_{k,\text{left}}$ , with  $B$  equal to the current ‘bulge’ of the  $X$ -shaped representation, and with  $C := U_{k,\text{right}}$ . Suppose by induction that  $B$  acts on the indices

$$B : j_{k-1} - r_{k-1} + 1, \dots, i_k + r_k - 1. \quad (5.6)$$

Then since the block pull-through equation (5.2) causes  $\tilde{A}$  and  $\tilde{C}$  to act precisely on the indices complementary to those of  $A$  and  $C$ , one easily obtains the bounds (5.5) from those of (3.2), (5.6). Finally, the induction hypothesis (5.6) could be verified in the same way, by using the characterization of the new index set of  $\tilde{B}$ .

A straightforward interpretation of the  $V$ -shaped representation is by thinking in terms of *column* operations acting on the unitary rank structured matrix  $A$ , and transforming it to triangular form by means of the equation  $AQ^H = I$ . The verification that this process leads indeed to the factorization in Definition 11, is nothing but a straightforward translation of the usual description of Definition 5 from the row to the column case.

Thus we see that the  $V$ -shaped representation has a natural interpretation in terms of *column* operations. But suppose now that we still consider them as row operations acting on the matrix  $A$ , and transforming it to triangular form by means of the equation  $Q^H A = I$ . How can we then interpret the action of each of the components of Definition 11? It turns out that we can not call them ‘zero-creating’ anymore; instead they will have a different interpretation which we call ‘structure-enlarging’. An application of this idea to devise an implicit QR-algorithm will be the subject of our future work.

We will now show how the internal pull-through process can be further fine-tuned in terms of individual Givens transformations. Instead of using the block pull-through lemma, the reduction will then be expressed in terms of the scalar pull-through lemma (Lemma 4).

Let us now describe this process. The first step consists in transforming the left and right branch of the representation to *rank-decreasing* form, i.e., such that

- the tops of the subsequent Givens arrows of the right branch altogether are strictly monotonically proceeding upwards;
- the tails of the subsequent Givens arrows of the left branch altogether are strictly monotonically proceeding downwards.

See Figure 5.4. (The term rank-decreasing was taken from [12].)

The process of bringing the left and right branch to rank-decreasing form, is very similar to the one for bringing the Givens transformations of the left and right branch of the representation as much as possible to the *bottom*, described in the reduction to zero-creating form, with the exception that the Givens transformations are now chased upwards instead of downwards. The details of this process will not be shown anymore.

We are now ready to perform the actual internal pull-through of both branches of the representation. The algorithm is again of a ‘central’ type: starting from the topmost Givens transformations, it gradually incorporates each time a new *pair* of opposite Givens arrows in the left and right branch of the representation. A specific step of this algorithm is illustrated in Figure 5.5.

Let us comment on this figure. First we draw the box of Givens transformations with number of rows and columns equal to the width of the new Givens arrows to

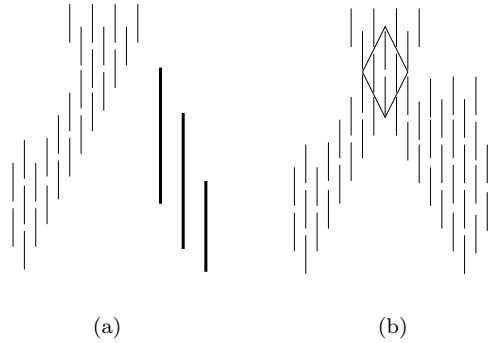


FIGURE 5.4. *Situation after bringing (a) the left branch and (b) both the left and right branch of the Givens product representation in rank-decreasing form.*

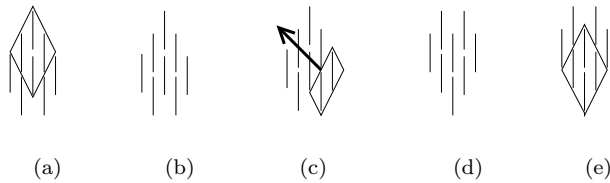


FIGURE 5.5. *Internal pull-through process for a Givens product representation: one step of the transformation process.*

be pulled through on the bottom left and bottom right, respectively. The algorithm consists then in the pull-through of each of the Givens transformations on the bottom right: see Figure 5.5(c-d).

We can now apply these techniques iteratively to pull through the subsequent Givens arrows: see Figure 5.6.

Concerning this figure, note that the central ‘bulge’ of the representation is chased towards the bottom during the internal pull-through, until it reaches the bottom at the very end of the algorithm. This leads then to the final  $V$ -shaped representation.

We recall that the central boxes drawn in Figure 5.6 are an auxiliary device to localize the central bulge of the representation, with dimensions determined by the Givens arrows immediately on the bottom left and right of it. Moreover, the pull-through process is not always unique, since e.g. in the transition from Figure 5.6(g) to Figure 5.6(h), one has several choices to distribute the Givens transformations of the central bulge over the two branches of the representation.

A hypothetical application of the internal pull-through process is that, during the  $k$ th step of this process, one has a relatively easy access to the elements around the  $(k, k)$  position of the given unitary matrix. This feature could (hypothetically) be useful for certain algorithms on unitary or related rank structured matrices, provided that one lets the internal pull-through process follow the flow of the algorithm.

We provide also an example for a very regular rank structure: see Figure 5.7. Note that the starting representation is here the one in Figure 3.9.

Summarized, we have now completely described the internal pull-through process of both branches of the unitary/Givens product representation.

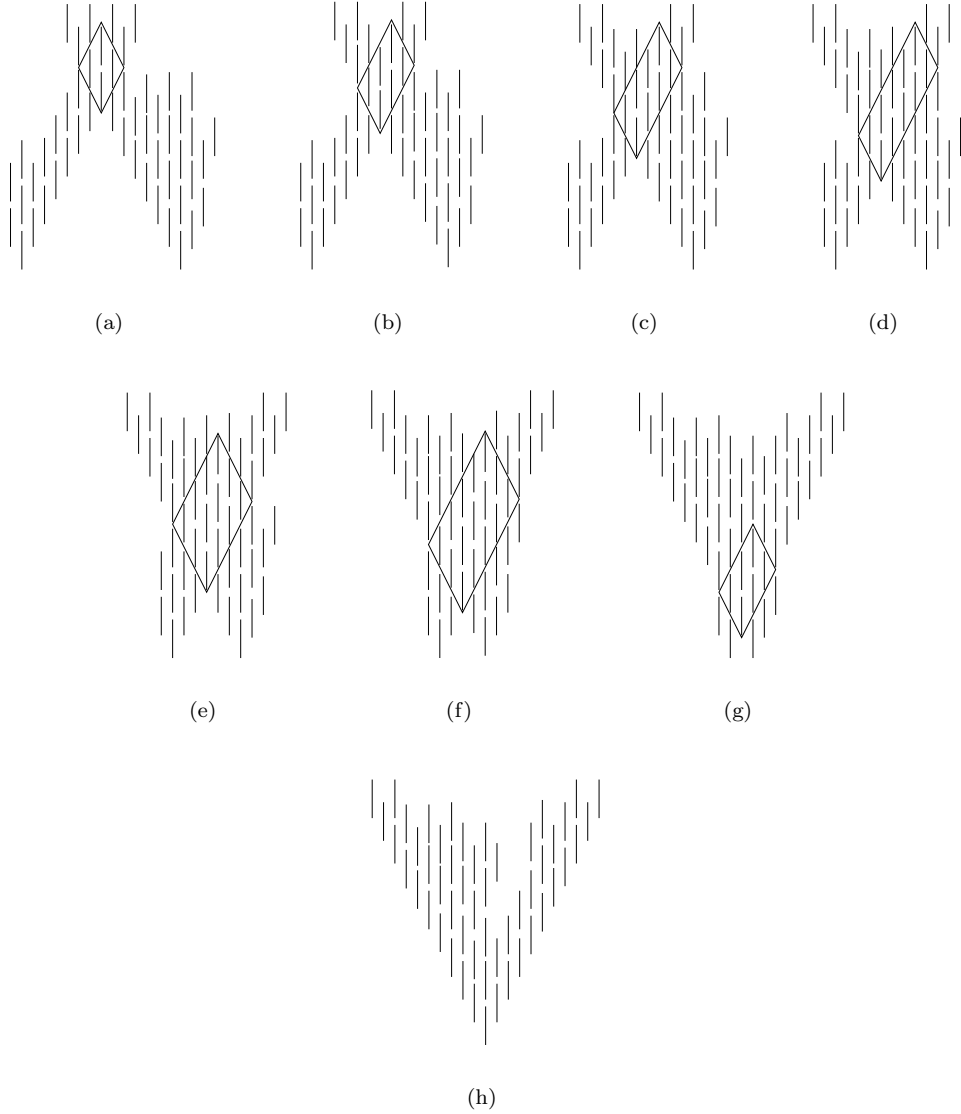


FIGURE 5.6. *Internal pull-through process for a Givens product representation: irregular case.*

**6. Shift correction term.** In this section we investigate how the ‘shift’ correction term to a rank structured matrix can be read off in terms of its unitary/Givens-weight or unitary/Givens product representation, and we provide several consequences to this observation.

**6.1. Some generalities.** We start with some generalities concerning the definition and properties of the shift correction term. We recall the following result.

**THEOREM 12.** (*Nullity theorem; see [15]:*) *Let  $A \in \mathbb{C}^{n \times n}$  and define an index set  $N = \{1, \dots, n\}$ . Suppose that we have partitions*

$$N = R \cup S \cup T = \tilde{R} \cup \tilde{S} \cup \tilde{T}$$

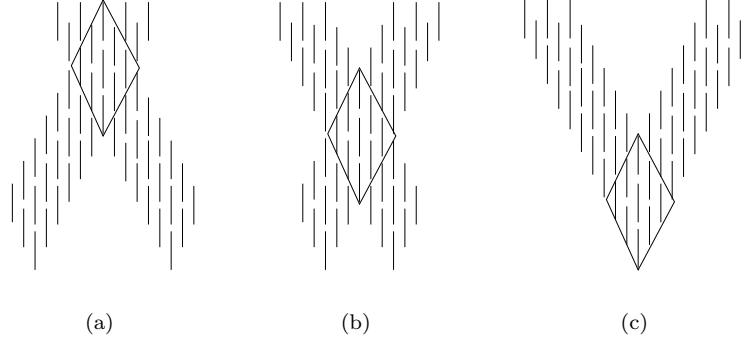


FIGURE 5.7. Internal pull-through process for a Givens product representation: regular case.

with  $S$  and  $\tilde{S}$  having the same size. Then

$$\text{Rank}(A_{\Lambda^{-1}}^{-1}(\tilde{S} \cup \tilde{T}, R \cup S)) = \text{Rank}(A_{\Lambda}(S \cup T, \tilde{R} \cup \tilde{S})) + |R| - |\tilde{R}|, \quad (6.1)$$

where  $A_{\Lambda^{-1}}^{-1}$  is defined from  $A^{-1}$  by putting  $A_{\Lambda^{-1}}^{-1}(\tilde{S}, S) = A^{-1}(\tilde{S}, S) - \Lambda^{-1}$ , and similarly  $A_{\Lambda}$  is defined from  $A$  by putting  $A_{\Lambda}(S, \tilde{S}) = A(S, \tilde{S}) - \Lambda$ .

An illustration of this property is given in Figure 6.1, where the property is illustrated for the distribution of index sets which is of main interest.

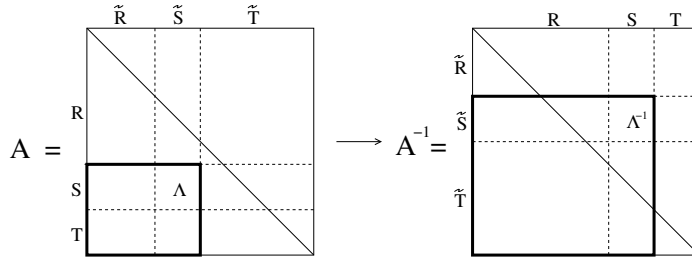


FIGURE 6.1. Inheritance of structure by the inverse matrix

REMARK 13.

1. It is possible to reformulate (6.1) as

$$\text{Null}(A_{\Lambda^{-1}}^{-1}(\tilde{S} \cup \tilde{T}, R \cup S)) = \text{Null}(A_{\Lambda}(S \cup T, \tilde{R} \cup \tilde{S})).$$

This explains the word ‘nullity theorem’.

2. We will often refer to the matrix  $\Lambda$  in Figure 6.1 under the name shift correction term, or briefly shift matrix. This terminology expresses that the underlying matrix  $A$  must satisfy a low rank block, at least when the shift matrix  $\Lambda$  is first subtracted from the appropriate entries. Hence, the shift matrix can be considered as an explicit part of the rank structure. In analogy with Definition 1, we will denote the corresponding structure block as  $\mathcal{B} = (i, j, r, \Lambda)$ . We will use this terminology also in case where  $\Lambda$  is rectangular.
3. One may ask what happens with the above theorem when  $\Lambda$  is singular. It can be shown that the matrix  $\Lambda^{-1}$  will have then a certain component ‘equal to

$\infty'$ , in a sense made exact in [15]. But we will not be concerned about this here.

The following, trivial corollary of Theorem 12 was also indicated in [15], although it was not stated there in explicit form.

COROLLARY 14. For  $A \in \mathbb{C}^{n \times n}$  a unitary matrix, and under the same conditions as in Theorem 12, we have

$$\text{Rank}(A_{\Lambda^{-H}}(R \cup S, \tilde{S} \cup \tilde{T})) = \text{Rank}(A_{\Lambda}(S \cup T, \tilde{R} \cup \tilde{S})) + |R| - |\tilde{R}|, \quad (6.2)$$

where we used the same notations as in Theorem 12: see Figure 6.2.

PROOF. Trivial from Theorem 12 by using that for a unitary matrix  $A$ , we have that  $A^{-1} = A^H$ .  $\square$

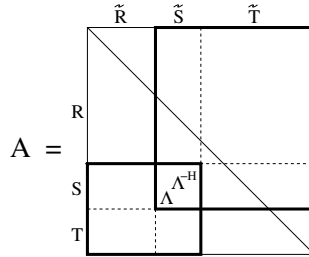


FIGURE 6.2. Specification of Figure 6.1 in case of a unitary matrix  $A$ . It follows that for such a matrix, the structure blocks always come in pairs, with shift correction terms related as  $\Lambda$  and  $\Lambda^{-H}$ .

It is worth pointing out the extension to unitary plus low rank matrices. Indeed: when  $A$  is unitary plus a correction term of rank at most  $r$ , then it is straightforward to check that  $A^{-1} = A^H + \text{Rk } 2r$ , from which it follows that Corollary 14 holds true, provided that one adds the value  $2r$  to the right hand side of (6.2). Note that we assume here  $A^{-1}$  to exist. The practical exploitation of the rank structure for unitary plus low rank matrices is a topic of ongoing research, originating from [6].

Corollary 14 can be interpreted by saying that the structure blocks of a unitary matrix always come in *pairs*. Given a structure block with shift correction term  $\Lambda$  in the block lower triangular part of a unitary matrix, there is also a structure block with shift matrix  $\Lambda^{-H}$  in the block upper triangular part. These structure blocks act precisely on complementary subsets, except that the rows and columns on which the shift matrices act are common to both matrices. Note here that the shift  $\Lambda$  is assumed to be square.

Note that Corollary 14 could be formulated for *pure* structure blocks as well, i.e., when  $S$  and  $\tilde{S}$  are empty sets. It reveals then that for each *pure* structure block of a unitary matrix, the complementary submatrix must have low rank as well. We have already encountered this result earlier in this paper, in the algorithm for transforming the unitary product representation into a unitary-weight representation (Section 4).

While the information about pure structure blocks is directly embedded in the sparsity of the unitary/Givens-weight or unitary/Givens product representation, this is no longer true when a shift correction term  $\Lambda$  is involved. So we pose ourselves the problem: given two subsequent *pure* structure blocks  $\mathcal{B}_k = (i_k, j_k, r)$  and  $\mathcal{B}_{k+1} = (i_{k+1}, j_{k+1}, r)$ , having the *same* rank index  $r$ , how can one compute the shift correction term which arises by ‘gluing’ these structure blocks into a huge  $\text{Rk } r$  structure block  $\mathcal{B} = (i_k, j_{k+1}, r, \Lambda)$ ? This problem will be investigated in the next two subsections.

**6.2. Shift matrix for the unitary/Givens-weight representation.** In this subsection we consider the problem of determining the shift correction term for the case of a unitary-weight representation: see Figure 6.3.

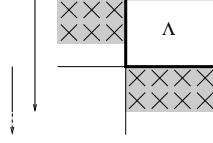


FIGURE 6.3. *Determining the shift correction term for a unitary-weight representation: problem statement.*

Concerning this figure, we note that the arrows on the left of the figure show the decompressing, rather than the compressing unitary operations of the unitary-weight representation. We search then for the correction matrix  $\Lambda$  of size  $i_{k+1} - i_k$  by  $j_{k+1} - j_k$  which has to be subtracted from the entries surrounded by the fat box in Figure 6.3, in order to extend these two Rk  $r$  structure blocks to a huge structure block, if such a  $\Lambda$  exists.

Let  $U := U_k^{-1}$  denote the decompressing transformation of the unitary-weight transformation, used to spread out the structure block  $\mathcal{B}_k$ . It will be useful to partition

$$U = \begin{bmatrix} U_{1,1} & U_{1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix}, \quad (6.3)$$

where  $U_{2,1}$  is square of size  $r$  by  $r$ . Hence, the complementary submatrix  $U_{1,2}$  contains precisely those rows which are not influenced by the next unitary operations  $U_{k+1}, \dots$  anymore, as well as those columns which are not influenced by the previous unitary operations  $U_{k-1}, \dots$  anymore: see Figure 6.4.

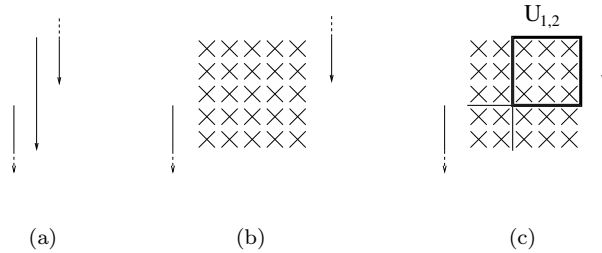


FIGURE 6.4. *Submatrix  $U_{1,2}$  relevant for the shift correction term.*

Assume now that the action radius of  $U$  has been enlarged so that it acts on columns  $j_k + 1, \dots, j_{k+1}$  as well: see Figure 6.5(a). Let  $W$  denote the huge weight block on which  $U$  acts, after this enlargement of action radius, and partition

$$W = \begin{bmatrix} W_{1,1} & W_{1,2} \\ 0 & W_{2,2} \end{bmatrix}, \quad (6.4)$$

where  $W_{1,1}$  has size  $r$  by  $j_k - j_{k-1}$ : see Figure 6.5.

We have then the following result.

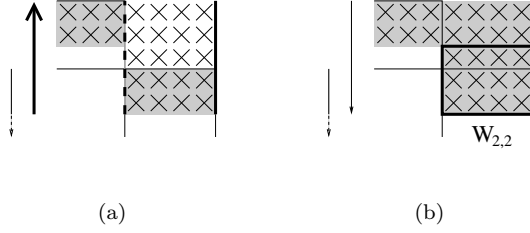


FIGURE 6.5. After enlarging the action radius of the unitary operation  $U := U_k^{-1}$ , we can read off the weight block  $W_{2,2}$  relevant for the shift correction term.

**THEOREM 15.** (*Shift correction term:*) Let there be given a unitary-weight representation for two subsequent  $\text{Rk } r$  structure blocks  $\mathcal{B}_k, \mathcal{B}_{k+1}$ . Then the shift correction term  $\Lambda$  can be expressed as

$$\Lambda = S_{U,1,2}W_{2,2}, \quad (6.5)$$

where we defined the Schur complement

$$S_{U,1,2} := U_{1,2} - U_{1,1}U_{2,1}^{-1}U_{2,2},$$

assuming that  $U_{2,1}^{-1}$  exists, and where we used the notation in the paragraphs preceding this theorem; see in particular Figures 6.4 and 6.5.

**PROOF.** Let us assume that the  $r$  by  $r$  submatrix  $U_{2,1}$  of (6.3) is nonsingular. We can then use  $U_{2,1}$  as a pivot block to eliminate the block element  $U_{1,1}$ , by means of a Gaussian row operation on the matrix  $U$ . Clearly, this operation will cause  $U$  to be transformed into

$$\begin{bmatrix} 0 & S_{U,1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix}, \quad (6.6)$$

where  $S_{U,1,2}$  is the Schur complement in the statement of this theorem.

Now since the Gaussian operation described above was chosen to add a linear combination of the *bottom*  $r$  rows of  $U$  to the rows above, it is trivial to see that applying this operation also to the rank structured matrix of Figure 6.3, does not affect the value of the shift correction term  $\Lambda$ . Hence, the shift remains invariant under the applied Gaussian operation.

Let us show now why it was useful to apply this Gaussian operation. Spreading out the unitary-weight representation leads to

$$\begin{bmatrix} 0 & S_{U,1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix} \begin{bmatrix} W_{1,1} & W_{1,2} \\ 0 & W_{2,2} \end{bmatrix} = \begin{bmatrix} 0 & S_{U,1,2}W_{1,2} \\ X & X \end{bmatrix}, \quad (6.7)$$

where we used (6.6) and (6.4) (in fact we should extend here  $W_{1,1}$  to contain also the weight blocks obtained by spreading out each of the structure blocks  $\mathcal{B}_1, \dots, \mathcal{B}_{k-1}$  on the left of  $\mathcal{B}_k$ ), and where we denoted the irrelevant block entries by  $X$ . Now it is clear by inspection that the shift correction term which must be subtracted from the (1,2) entry of (6.7) to make this matrix of rank at most  $r$ , will be precisely equal to  $S_{U,1,2}W_{1,2}$ , which proves (6.5). (Incidentally, note that the uniqueness of this shift

correction term is only guaranteed when the (2,1) block of (6.7) attains its maximum possible rank of  $r$ .)  $\square$

Note that in the above proof, it was nowhere used that we had really given a *unitary-weight* representation, and hence that the matrix  $U := U_k^{-1}$  is unitary. When we make use of this additional data, we can drive Theorem 15 one step further. We need the following, trivial observation.

LEMMA 16. *If  $U$  is a unitary matrix partitioned as in (6.3), then the Schur complement  $S_{U,1,2} := U_{1,2} - U_{1,1}U_{2,1}^{-1}U_{2,2}$  equals  $U_{1,2}^{-H}$ .*

PROOF. Consider the matrix equation  $UU^H = I$ . One can apply to both sides of this equation a Gaussian row operation that uses  $U_{2,1}$  as a pivot block to eliminate  $U_{1,1}$ , hereby obtaining the new equation

$$\begin{bmatrix} 0 & S_{U,1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix} \begin{bmatrix} U_{1,1}^H & U_{2,1}^H \\ U_{1,2}^H & U_{2,2}^H \end{bmatrix} = \begin{bmatrix} I & X \\ 0 & I \end{bmatrix},$$

where the irrelevant block entry is denoted as  $X$ . The proof can now be finished by evaluating the (1,1) block element of the above equation.  $\square$

COROLLARY 17. *For a unitary-weight representation, and under the same conditions as in Theorem 15, the shift correction term is given by*

$$\Lambda = U_{1,2}^{-H}W_{2,2}. \quad (6.8)$$

Note that Corollary 17 shows that there can always be found a suitable shift correction matrix  $\Lambda$  to the structure, provided the matrix  $U_{1,2}$  (or equivalently,  $U_{2,1}$ ) is nonsingular. In the case where  $U_{1,2}$  is singular, one could say that the shift matrix  $\Lambda = U_{1,2}^{-H}W_{2,2}$  has a certain component equal to ‘ $\infty$ ’, again in the sense of [15].

We provide an illustration of the above results. We recall the following definition.

DEFINITION 18. (*uv-representability, see e.g. [11]:*) *Let  $\mathcal{R}$  be a rank structure with a global rank upper bound  $r_k =: r$  for each  $k$ . We say  $A \in \mathbb{C}^{m \times n}$  to be *uv-representable w.r.t.  $\mathcal{R}$*  if there exists a factorization*

$$A = uv + A_{\text{completion}}, \quad (6.9)$$

where  $u \in \mathbb{C}^{m \times r}$ ,  $v \in \mathbb{C}^{r \times n}$  and where  $A_{\text{completion}}$  is a ‘completion’ matrix having the property that its restriction to each of the structure blocks of  $\mathcal{R}$  is zero. The factorization (6.9) is called a *uv-representation* of  $A$ .

The above definition states that for a matrix to be *uv-representable*, the low rank generators of the different structure blocks  $\mathcal{B}_k$  of  $\mathcal{R}$  should be ‘compatible’ in the sense that they can be completed to a global matrix  $uv$  of rank at most  $r$ .

Let us investigate the connection with shift matrices. First, it is trivial to see that the shift matrices  $\Lambda_k$  are nothing but the block diagonal elements of the completion matrix of the *uv-representation*, if both exist. Indeed, the shift matrices were defined precisely as the ‘correction terms’ which have to be subtracted from the block diagonal entries in order to extend the  $\text{Rk } r$  structure, which is the same idea as (6.9).

This connection can be made tighter.

LEMMA 19. *Given a matrix  $A$  satisfying a rank structure  $\mathcal{R}$  whose structure blocks have the same rank upper bound  $r$ . A sufficient condition for the *uv-representability* of  $A$  w.r.t.  $\mathcal{R}$  is that for each unitary component  $U_k$  of a unitary-weight representation*

of  $A$  according to the structure  $\mathcal{R}$ , partitioning  $U := U_k$  as in (6.3), then the  $r$  by  $r$  bottom left submatrix  $U_{2,1}$  is invertible.

PROOF. Assume that the conditions in the lemma are satisfied. We know from our earlier results that the invertibility of the  $r$  by  $r$  bottom right submatrix of each unitary component  $U_k$  of the unitary-weight representation guarantees the existence of each of the shift matrices  $\Lambda_k$ . Now we will show how this result can be extended by ‘gluing’ the unitary components of the unitary-weight representation, so that also the other elements of the completion matrix in (6.9) can be obtained.

To this end, let us glue the structure blocks  $\mathcal{B}_k, \mathcal{B}_{k+1}$  into a single huge structure block, as in Figure 6.5. Correspondingly, we glue the unitary components as

$$U_{k+1}U_k = \begin{bmatrix} X & U_{k;1,2} & 0 \\ X & X & U_{k+1;1,2} \\ X & X & X \end{bmatrix}.$$

Partitioning this matrix as in (6.3), we have that the top right submatrix is block lower triangular with block diagonal entries  $U_{k;1,2}$  and  $U_{k+1;1,2}$ . The assumed invertibility of these submatrices implies then the invertibility of this entire block lower triangular matrix, and hence we can determine the (extended) shift by means of the same formula (6.8) as usual. Repeating this gluing argument, we can finally determine the entire completion matrix in (6.9).  $\square$

The following generalizes a result of [25].

**THEOREM 20.** *Given a rank structure  $\mathcal{R}$  whose structure blocks have the same rank upper bound  $r$ . The class of  $uv$ -representable matrices w.r.t.  $\mathcal{R}$  is dense in the class of all matrices satisfying  $\mathcal{R}$ .*

PROOF. The fact that the class of  $uv$ -representable matrices, and even its closure must satisfy  $\mathcal{R}$  is trivial.

To prove the other direction, let there be given a matrix  $A \in \mathbb{C}^{m \times n}$  satisfying  $\mathcal{R}$ . It will be sufficient to construct a sequence of  $uv$ -representable matrices  $A_\epsilon, \epsilon \in \mathbb{C} \setminus \{0\}$  such that  $\lim_{\epsilon \rightarrow 0} A_\epsilon = A$ .

To this end, consider a unitary-weight representation for the matrix  $A$  according to the structure  $\mathcal{R}$ . By means of the previous lemma, it will be sufficient to construct the family  $A_\epsilon$  such that each unitary component  $U_{k,\epsilon}, \epsilon \in \mathbb{C} \setminus \{0\}$ , has a nonsingular  $r$  by  $r$  bottom left submatrix. But this can be easily reached: assuming for example a Givens-weight representation, then it suffices to replace each Givens transformation  $G$  which equals a diagonal matrix, by the new value

$$G_\epsilon := \frac{1}{\sqrt{1+|\epsilon|^2}} \begin{bmatrix} 1 & \epsilon \\ -\bar{\epsilon} & 1 \end{bmatrix} G.$$

It is then easy to show that, by the fact that each Givens transformation of the Givens-weight decomposition in Figure 3.4 is now different from a diagonal matrix, it follows that the  $r$  by  $r$  bottom left submatrix of  $U_{k,\epsilon}$  is nonsingular. (This can be seen e.g. by the fact that the complementary, top right submatrix must be lower triangular with non-zero diagonal elements.)  $\square$

We refer to [28, 18, 17] for some further references on  $uv$ -representation problems.

**6.3. Shift matrix for the unitary/Givens product representation.** Now we consider the case of a *unitary* matrix  $A$ , and we consider the analogous question

of how the shift can be expressed in terms of the parameters of the unitary/Givens product representation.

Let us work with the  $\Lambda$ -shaped representation of Definition 5, i.e.,

$$Q = U_{K,\text{left}} \dots U_{1,\text{left}} U_0 U_{1,\text{right}} \dots U_{K,\text{right}}. \quad (6.10)$$

It turns out that the shift will now be determined by the pairs of *opposite* unitary operations of the left and the right branch:  $\{U_{k,\text{left}}, U_{k,\text{right}}\}$ . For convenience, let us fix  $k$  and set

$$U := U_{k,\text{left}}, \quad V := U_{k,\text{right}}. \quad (6.11)$$

Partition the *left* unitary operation as in (6.3), i.e.,

$$U = \begin{bmatrix} U_{1,1} & U_{1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix}, \quad (6.12)$$

where  $U_{2,1}$  is square of size  $r$  by  $r$ .

We do now the same in the *right* branch of the representation: we partition

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} \\ V_{2,1} & V_{2,2} \end{bmatrix} \quad (6.13)$$

where again  $V_{2,1}$  contains precisely those rows and columns of  $V$  which are not influenced by the previous and next unitary operations  $U_{k-1,\text{right}}$  and  $U_{k+1,\text{right}}$  anymore, respectively: see Figure 6.6.

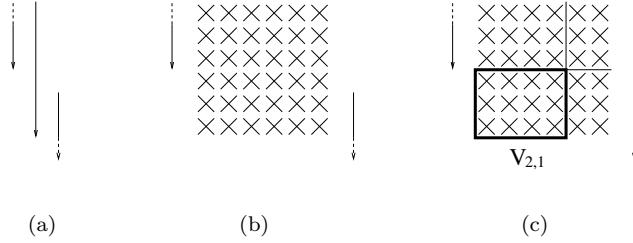


FIGURE 6.6. Submatrix  $V_{2,1}$  relevant for the shift correction term in the unitary case.

We have the following result.

**THEOREM 21.** (*Shift correction term, unitary case:*) *Let there be given a unitary product representation for two subsequent  $\text{Rk } r$  structure blocks  $\mathcal{B}_k, \mathcal{B}_{k+1}$ . Then the shift correction term  $\Lambda$  can be expressed as*

$$\Lambda = S_{U,2,1} V_{2,1} = U_{1,2}^{-H} V_{2,1}, \quad (6.14)$$

where we used the notations of the paragraph preceding this proof.

**PROOF.** We prove the theorem by transforming the unitary product representation into a unitary-weight representation: see Figure 6.7.

Let us comment on this figure. Starting from the identity matrix, we assume that we are applying the process of building up the unitary-weight representation, as in Figure 4.1 of Section 4.

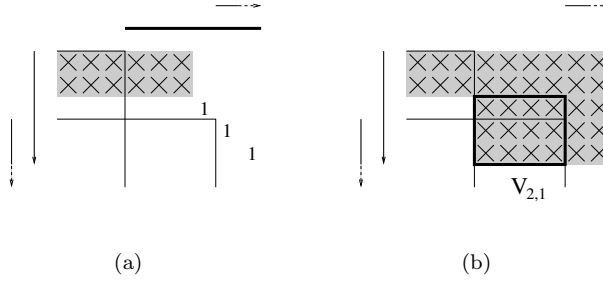


FIGURE 6.7. Transforming the unitary product representation into a unitary-weight representation causes the submatrix  $V_{2,1}$  to appear.

Suppose that in this process, we are at the point of multiplying with the next unitary operations  $U := U_{k,\text{left}}$  and  $V := U_{k,\text{right}}$  to the rows and columns, respectively. Let us first apply the operation  $V$  to the columns. It is clear that this multiplication will cause the submatrix  $V_{2,1}$  to appear on the positions indicated in Figure 6.7(b).

Clearly, the submatrix  $V_{2,1}$  will now play the role of the weight block  $W_{2,2}$  in the proof of Theorem 15. The proof can be finished by invoking the latter theorem.  $\square$

REMARK 22. (The  $V$ -shaped case.) In case of the  $V$ -shaped representation (5.4), the analogue of (6.14) is that

$$\Lambda = V_{2,1} S_{U,2,1} = V_{2,1} U_{1,2}^{-H},$$

where  $V := V_{k,\text{left}}$  and  $U := V_{k,\text{right}}$  denote the  $k$ th unitary components of the left and right branch of the representation, and where the corresponding partitions are defined in exactly the same way as before. Hence, the only difference with (6.14) is that the order of the two factors must be reversed. We do not go further into this.

For the next corollary, we will assume a Givens product representation where the Givens transformations of the right branch take the form

$$\begin{bmatrix} c & \times \\ s & \times \end{bmatrix}, \quad (6.15)$$

where  $c$  and  $s$  are complex numbers such that  $|c|^2 + |s|^2 = 1$  (the cosine and sine), and where the Givens transformations of the left branch take the Hermitian transposed form of (6.15).

COROLLARY 23. (Quotient of sines.) Assume that  $A$  is a lower semiseparable plus diagonal unitary matrix of semiseparability rank  $r$ , as in Figure 3.9. Then invoking the conditions in the paragraph before this corollary, the  $k$ th shift element  $\lambda_k \in \mathbb{C}$  to the structure equals the product of the sines of the Givens transformations belonging to the Givens arrow with tail index  $k$  in the right branch, divided by the corresponding product of sines in the left branch: see Figure 6.8.

PROOF. It follows from Theorem 21 that the shift element  $\lambda_k \in \mathbb{C}$  will equal the quotient of the bottom left element of  $V$ , divided by the complex conjugate of the top right element of  $U$ , where  $U$  and  $V$  denote the corresponding unitary components of the unitary product representation. It suffices then to expand these elements; for the unitary component in the right branch, and assuming that we have rank index 3, this

yields

$$V = G_{3,4}G_{2,3}G_{1,2} = \left[ \begin{array}{c|ccc} \times & \times & 0 & 0 \\ \times & \times & \times & 0 \\ \times & \times & \times & \times \\ \hline s_1 s_2 s_3 & \times & \times & \times \end{array} \right],$$

whose bottom left element equals indeed the product of the sines  $s_k$  as in (6.15). The result is now easy to check.  $\square$

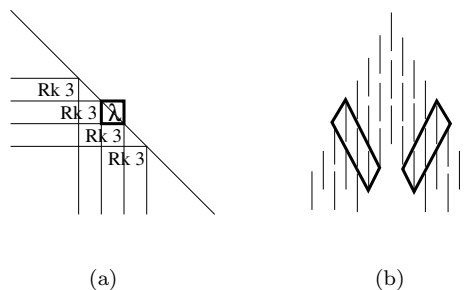


FIGURE 6.8. The indicated shift element  $\lambda \in \mathbb{C}$  of the lower semiseparable plus diagonal structure equals the product of all the boxed sines in the right branch, divided by the product of all the boxed sines in the left branch.

We provide an illustration of the above results. In what follows, a unitary matrix  $Q \in \mathbb{C}^{n \times n}$  will be called *Householder-like* if it equals a rank-one modification of the identity, i.e., if

$$Q = I + \text{Rk } 1. \quad (6.16)$$

**THEOREM 24.** (*Householder-like matrices:*) Let  $Q \in \mathbb{C}^{n \times n}$  be a unitary matrix. The following are equivalent:

1.  $Q$  is a Householder-like matrix with determinant  $d$  (cf. (6.16));
2.  $Q$  can be written as

$$Q = I + (d - 1)uu^H, \quad (6.17)$$

for a certain normalized column vector  $u \in \mathbb{C}^n$ ;

3.  $Q$  can be written as an upward, followed by a downward Givens sequence, where the opposite Givens transformations of both branches are each others Hermitian transposes, and with topmost Givens transformation  $G_{1,2}$  being Householder-like of size 2 by 2 with determinant  $d$ .

See the right part of Figure 6.9.

**PROOF.** Consider the implication from the first to the second statement. From the unitarity of  $Q$  we know that there exists a spectral decomposition

$$Q = UDU^H, \quad (6.18)$$

for a certain unitary matrix  $U$  and unitary diagonal matrix  $D$ . Inserting this spectral decomposition in (6.16) yields  $\text{Rk } 1 = U(D - I)U^H$ . It follows that the unitary

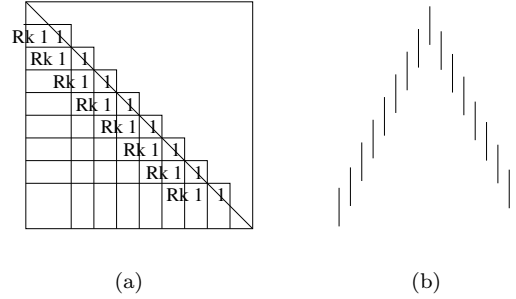


FIGURE 6.9. Two related representations of a Householder-like matrix. The left picture shows the induced lower semiseparable plus diagonal structure blocks, each of them having rank index  $r = 1$  and shift element  $\lambda = 1$ . The right picture shows the corresponding decomposition as a product of Givens transformations. Here the opposite Givens transformations of the left and right branch have to be each others Hermitian transposes, and the topmost Givens transformation must have one of its eigenvalues equal to one.

diagonal matrix  $D$  can have at most one diagonal entry  $d$  for which  $d \neq 1$ , leading to

$$\text{Rk } 1 = (d - 1)uu^H,$$

for some normalized column vector  $u \in \mathbb{C}^n$ , which leads to the desired equation (6.17). Incidentally, note that  $d$  can be identified as the determinant of  $Q$  due to (6.18).

Next, we show the implication from the second to the third statement. We do this by applying a similarity operation with Givens transformations  $G_{n-1,n}^H, \dots, G_{2,3}^H$  to both sides of (6.17), chosen to compress the vector  $u$  except for its top 2 by 2 block of elements. What is left after this compression process will then constitute the topmost Givens transformation  $G_{1,2}$ , and hence it follows that we can decompose  $Q$  as

$$Q = UG_{1,2}U^H, \tag{6.19}$$

where  $U := G_{n-1,n} \dots G_{2,3}$ .

Finally, let us establish the implication from the third to the first statement. Thus assume that  $Q$  has the form of a Givens product as in (6.19). From the assumption that  $G_{1,2}$  is Householder-like of size 2 by 2, it follows that

$$G_{1,2} = I + \text{Rk } 1, \tag{6.20}$$

where  $\text{Rk } 1$  is a matrix of rank at most one, having only its top left 2 by 2 block of entries non-zero. It follows that also the matrix  $Q = UG_{1,2}U^H$  must obviously have a factorization of the same form as in (6.20).  $\square$

REMARK 25.

1. The implication from the first to the second statement in Theorem 24 can be found also in a more general block form in [4].
2. The implication from the third to the first statement in Theorem 24 can be considered as a non-standard way for constructing Householder-like operations. Special care may be spent here to the case  $d = -1$ , which corresponds

to the classical Householder reflection. Under the assumption of real arithmetic, the topmost Givens transformation  $G_{1,2}$  in the third statement must then assume the form

$$G_{1,2} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix},$$

*i.e.*,  $G_{1,2}$  must be an elementary orthogonal reflection matrix.

We still have to explain the connection of Theorem 24 with the above results about shift matrices. To this end, note first that a Householder-like matrix must be obviously *lower semiseparable plus diagonal* of semiseparability rank one, with all shift elements equal to one, since it is defined as a rank-one modification of the identity matrix. This is illustrated in the left part of Figure 6.9.

The existence of the Givens product representation in the third statement of Theorem 24, follows then as a direct consequence of this lower semiseparable plus diagonal structure. Furthermore, the fact that the opposite pairs of Givens transformations of the two branches of the representation can be chosen as each others Hermitian transposes, is an illustration of the fact that the shift elements of the lower semiseparable plus diagonal structure, and hence the *quotient of the sines*, must be all equal to one, by means of Corollary 23.

Summarized, we described now how to characterize the shift correction term in terms of the unitary/Givens-weight or unitary/Givens product representation. Some practical applications of these results will be provided in our future work, starting with [10].

**7. Conclusion.** In this paper we described how to build up and manipulate unitary and Givens product representations. We showed how to determine the shift matrices to the rank structure. Our future work includes (i) eigenvalue computation of unitary rank structured matrices (see [10]), and (ii) some unitary plus low rank variants.

#### REFERENCES

- [1] G. S. Ammar, W. B. Gragg, and C. He. An efficient QR algorithm for a Hessenberg submatrix of a unitary matrix. In W. Dayawansa, A. Lindquist, and Y. Zhou, editors, *New Directions and Applications in Control Theory*, volume 321 of *Lecture Notes in Control and Information Sciences*, pages 1–14. Springer-Verlag, 2005.
- [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. DOWDATING OF SZEGŐ POLYNOMIALS AND DATA-FITTING APPLICATIONS. *Linear Algebra and its Applications*, 172:315–336, 1992.
- [4] 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.
- [5] D. Bindel, S. Chandrasekaran, J. W. Demmel, D. Garmire, and M. Gu. A fast and stable nonsymmetric eigensolver for certain structured matrices, May 2005.
- [6] D. A. Bini, F. Daddi, and L. Gemignani. On the shifted QR-iteration applied to Frobenius matrices. *Electronic Transactions on Numerical Analysis*, 18:137–152, October 2004.
- [7] D. A. Bini, Y. Eidelman, L. Gemignani, and I. C. Gohberg. Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices.
- [8] 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.

- [9] 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.
- [10] S. Delvaux and M. Van Barel. Eigenvalue computation for unitary rank structured matrices. Technical Report TW465, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, July 2006.
- [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. 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.
- [14] 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.
- [15] S. Delvaux and M. Van Barel. Structures preserved by matrix inversion. *SIAM Journal on Matrix Analysis and its Applications*, 28(1):213–228, 2006.
- [16] P. Dewilde and A.-J. van der Veen. *Time-varying systems and computations*. Kluwer Academic Publishers, Boston, June 1998.
- [17] H. Dym and I. C. Gohberg. Extensions of band matrices with band inverses. *Linear Algebra and its Applications*, 36:1–24, 1981.
- [18] I. C. Gohberg, M. A. Kaashoek, and H. J. Woerdeman. A note on extensions of band matrices with invertible maximal and submaximal blocks. *Linear Algebra and its Applications*, 150:157–166, 1991.
- [19] W. B. Gragg. The QR algorithm for unitary Hessenberg matrices. *Journal of Computational and Applied Mathematics*, 16:1–8, 1986.
- [20] L. Reichel, G. S. Ammar, and W. B. Gragg. Discrete least squares approximation by trigonometric polynomials. *Math. Comp.*, 57:273–289, 1991.
- [21] M. Stewart. An error analysis of a unitary Hessenberg QR algorithm. *SIAM Journal on Matrix Analysis and its Applications*, 28(1):40–67, 2006.
- [22] M. Van Barel and A. Bultheel. A parallel algorithm for discrete least squares rational approximation. *Numerische Mathematik*, 63:99–121, 1992.
- [23] 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.
- [24] 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.
- [25] R. Vandebril, M. Van Barel, and N. Mastronardi. A note on the representation and definition of semiseparable matrices. *Numerical Linear Algebra with Applications*, 12(8):839–858, October 2005.
- [26] 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.
- [27] 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.
- [28] H. J. Woerdeman. *Matrix and operator extensions*. CWI Tract 68, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, 1989.