The Schur algorithm for matrices with Hessenberg displacement structure

G. Heinig and V. Olshevsky

Abstract. A Schur-type algorithm is presented for computing recursively the triangular factorization $R = LU$ of a strongly nonsingular $n \times n$ matrix satisfying a displacement equation $RY - VR = GH^T$ with Hessenberg matrices Y and V and $n \times \alpha$ matrices G, H. If α is small compared with n and the matrices Y and V admit fast matrix-vector multiplication, the new algorithm is fast in the sense that it will require less than $O(n^3)$ arithmetic operations.

1. Introduction

Displacement structure. Many problems in applied sciences and engineering can be reduced to linear algebra problems. Unfortunately, standard available methods may not be practical, e.g., because of the large size of the arising matrices. Fortunately, such matrices often have a structure that can be exploited in order to design more efficient *fast algorithms*, i.e., those requiring significantly less arithmetic operations. Among structured matrices for which this is possible are Toeplitz matrices $[a_{i-j}]$, Hankel matrices $[a_{i+j}]$, Toeplitz-plus-Hankel matrices, Vandermonde matrices $[a_i^{j-1}]$, and Cauchy matrices $[\frac{1}{a_i - b_j}]$.

It turned out that all classes listed above belong to the more general family of matrices with a *displacement structure*, and that fast algorithms can be designed for these more general classes as well. Perhaps the first reference explicitly discussing *fast algorithms* for matrices with displacement structure is the thesis [**36**][1]. The first journal papers on displacement structure were probably [**10**] and [**28**]. But let us also mention the papers [**42, 43**] in which the idea of displacement structure in principle appears for integral operators with a difference kernel, the paper [**26**], where this idea appeared in connection with the Chandrasekhar equations, and

1991 *Mathematics Subject Classification.* Primary: 15A23, 15A57, Secondary: 65F05.

The work of the first author was supported by research grant SM 07/00 of Kuwait University and the work of the second author was supported by NSF grants CCR 9732355 and 0098222.

[1] In Sec. 4.0.1 M.Morf writes: *"In October 1970 the crucial* shift-low-rank updating property *was recognized by the author as the proper generalization of the Toeplitz and Hankel structure of matrices... The algorithm was called* "Fast Cholesky decomposition" *(or RMH4, the forth in a series of related algorithms, see Sec 6.1). In June 1971 computer programs for both scalar and block decompositions were successfully completed by the author. It was found that the known numerical well behaviour of the Cholesky decomposition seemed to be inherited."*

also the papers [**44, 45**], where displacement equations appeared in the context of factorization of rational matrix functions. We note, however, that both in [**36**] and [**28**] a displacement structure is considered that is related to Toeplitz matrices. A general displacement approach was proposed in [**19**] and was applied in [**25**] to study matrices related not only to Toeplitz but to Vandermonde and Cauchy matrices as well. For more information and references we refer also to the monographs [**5**], [**6**], and to a recent collection of papers [**33**].

To explain the contribution of the present paper, let us recall some facts and definitions. In his seminal paper [**41**] I. Schur designed an algorithm whose matrix interpretation is computing the triangular factorization $T = LL^*$ for a positive definite (quasi) Toeplitz matrix T. We refer to [**35**] for the details. This classical Schur algorithm belongs to the family of *fast algorithms*: it needs only $O(n^2)$ operations whereas standard methods require $O(n^3)$. It turned out that Schur-type algorithms can be designed for the more general classes of matrices described next. It was noticed in [**28**] that the shift-invariant structure of a Toeplitz matrix $T = [\, a_{i-j}\,]$ yields that the (displaced) matrix $T - ZTZ^*$ has rank two. Here Z is the lower shift matrix. It was shown in [**36, 8, 34, 37**] that fast Schur-type algorithms exist for matrices R for which the integer

$$\alpha = \operatorname{rank}(R - ZRZ^*) \tag{1.1}$$

is small (though possibly bigger than two). Such matrices R were called Toeplitz-like.

In [**25**] the following general definition was given. Let $\{Y, V\}$ be two fixed $n \times n$ matrices. A $n \times n$ matrix R is said *to possess a* $\{Y, V\}$*-displacement structure* [2] if

$$\alpha = \operatorname{rank}(RY - VR) \tag{1.2}$$

is small compared with n. The integer α is called the $\{Y, V\}$*-displacement rank of* R. It was observed that Vandermonde and Cauchy matrices have displacement structure as well. For example, for Vandermonde matrices $W = [a_i^{j-1}]$, and for Cauchy matrices $C = [\, \frac{1}{a_i - b_j}\,]$ we have

$$\operatorname{rank}(D_a^{-1}W - WZ^*) = 1, \qquad \operatorname{rank}(D_aC - CD_b) = 1,$$

where $D_a = \operatorname{diag}(a_1, \dots, a_n)$, and $D_b = \operatorname{diag}(a_1, \dots, a_n)$. It was shown in [**25**] that fast $O(n^2)$ algorithms exist for Vandermonde-like and Cauchy-like matrices defined as those having small displacement ranks. Let us note that the algorithms in [**25**] are not of the Schur type but of the Levinson type. The Levinson-type algorithms produce a triangular factorization of the inverse matrix R^{-1}, whereas the Schur-type algorithms output the triangular factorization of the matrix R itself. Both algorithms can also be used for fast solving linear systems of equations $Rx = b$. Schur-type algorithms seem to attract more attention lately, mostly because in many cases they are more numerically accurate.

The variant of displacement structure in (1.2) is sometimes called the Hankel or Sylvester or continuous-time. Another variant of displacement structure (1.1) was extended in [**7**] to capture the matrices R for which the rank of $R - VRY$ is small. This is called the Toeplitz or Stein or discrete-time displacement structure. For the more general displacement structures see [**32**]. As far as the design of fast algorithms is concerned, a particular choice of displacement structure (i.e., discrete-time or continuous-time) seems to be relevant in the Hermitian (or symmetric) case

[2] Actually, in [**25**] it was called to *posses a* (Y, V)*-reduction*

but irrelevant for the discussion in this paper, in which we consider non-Hermitian matrices.

Examples of matrices with Hessenberg displacement structure. In order to derive fast recursive algorithms some conditions have to be imposed on the matrices Y and V. In [**25**] is was shown that the assumption of Y and V to be Hessenberg matrices is sufficient to obtain fast Levinson-type algorithms. In the papers of T. Kailath and his coauthors (see [**32**], [**33**] and the references therein) it was shown that the (stronger) assumption for V and Y to be triangular is sufficient to design fast Schur-type algorithms. The case of Hessenberg matrices Y and V is not treated there.

However, there are many important cases in which Y and V cannot be chosen as triangular but only as Hessenberg. We next describe two such families. In the first one the matrix V is diagonal and Y is a tridiagonal matrix. In the second case both V and Y are tridiagonal. The first family includes matrices of the form $[\,Q_{j-1}(x_i)\,]$, where $\{Q_k(x)\}$ is a system of orthogonal polynomials on the real line. These matrices appear in interpolation and they are, in a sense, close to Vandermonde. In this case V is diagonal and Y is tridiagonal Jacobi matrix capturing the recurrence relations (see e.g. [**18**], [**23**], [**40**], [**29**], [**31**]). (An analogous case is when $\{Q_k(x)\}$ are orthogonal on the unit circle, in this case Y should be chosen as unitary Hessenberg [**39**].) For these structures the Schur-type algorithms are missing.

Among the most important examples of matrices from the second family (for which both V and Y are tridiagonal) seems to be Toeplitz-plus-Hankel matrices. In this case one has to choose $Y = V = Z + Z^T$ in order to get a displacement rank of 4. A Schur-type algorithm for these matrices was designed in [**47**], based on the Levinson-type algorithm from [**24**][3]. A Schur-type algorithm for Toeplitz-plus-Hankel matrices was also proposed in the Ph.D. Thesis of P. Jankowski, TU Chemnitz 1990. A Schur-type algorithm for quasi-Toeplitz-plus-quasi-Hankel matrices were presented in [**46**].

There are several more classes of matrices with a displacement structure defined via non-triangular Y and V. One such structure is exhibited by the generalized Bezoutians of two polynomials $F(x)$ and $G(x)$ of the form $B = [b_{ij}]$ where the entries are obtained from

$$\frac{F(\lambda)G(\mu) - F(\mu)(\lambda)}{\lambda - \mu} = \sum_{i,j=0}^{n-1} b_{ij}\, P_i(\lambda)Q_j(\mu),$$

where $\{Q_k(x)\}$ and $\{P_k(x)\}$ are two families of orthogonal polynomials (see [**15**]).

Another important representative of this family is the class of modified moment matrices occurring, e.g., in least squares problems for orthogonal polynomials expansions [**23**], and in preconditioning of ill-conditioned Hankel systems [**9**]. These matrices have displacement rank two for some tridiagonal Y and V. In [**23**] and [**11**] Levinson-type algorithms were designed for this kind of matrices, and using an approach different from the one of this paper, Schur-type and superfast algorithms are being presented in [**21**] for the case when the displacement rank is two.

[3]Let us note that it is possible to design Schur-type algorithms for Toeplitz-plus-Hankel and Toeplitz-plus-Hankel-like matrices transforming them first into Cauchy-like matrices with the help of the discrete Fourier or real trigonometric transformations, see, e.g., [**20**], [**12**], [**22**].

Main results. In [**30**] a fairly general class of *polynomial Hankel-like matrices* was considered. We next recall (a non-symmetric version of) this definition; the class includes, as special cases, all types of structured matrices presented above, as well as their generalizations to higher displacement ranks. It is based on the well known fact that that a general Hessenberg matrices (called a *confederate* matrix in [**38**]) can be associated with a system of polynomials. Let $R = L_1 H L_2^T$ with a Hankel matrix H, and with L_1 and L_2 being lower triangular matrix capturing the coefficients of certain polynomial systems $\{Q(x)\}$ and $\{P(x)\}$, resp. It follows that R has $\{Y, V\}$ -displacement rank two, where Y and V are the confederate matrices of $\{Q(x)\}$ and $\{P(x)\}$, resp. If the corresponding displacement rank is small, then such R is referred to as a polynomial Hankel-like matrix. It turns out that all structured matrices in the above examples are polynomial Hankel-like, corresponding to the special choices of polynomial systems.

The main goal of the present paper is to show that Schur-type algorithms can be constructed for recursive triangular factorization of general polynomial Hankel-like matrices, i.e., those with $\{Y, V\}$-displacement structure defined via Hessenberg matrices Y and V. The new algorithm is fast in the sense that it requires less than $O(n^3)$ arithmetic operations if the matrix-vector multiplication by Y and V can be done with less than $O(n^2)$ computational complexity.

Let us finally note that it was widely believed for some time that such algorithms cannot be constructed. For instance, this claim was made by T. Kailath at his talk at a conference in Santa Barbara in 1996. We wrote down the main algorithm presented here at the end of that conference. Recently this claim appeared again in [**27**], where it is written: *"The restriction that $\{\Omega, \Delta, F, A\}$ are lower triangular is the most general one that allows recursive triangular factorization (cf. a result in* [**35**]*)."*

A missing connection to rational matrix functions. Let us conclude this introduction with an open problem. As is well-known, there is a close relation between displacement structure and rational interpolation and rational matrix function theory [**4**]. Specifically, in the case of triangular Y and V Schur-type algorithms correspond to the factorization of a certain associated rational matrix function. Moreover, the above discussed *triangularity condition* is a well-known necessary condition for the existence of factorization, see, e.g., [**45**], [**2**]; A particular variant of the factorization theorem that is the closest to our discussion here can be found in theorem 2.3 of [**13**]. Since we design here a Schur-type algorithm for non-triangular $\{Y, V\}$ hence a rational matrix interpretation of our method is not a factorization of rational matrix functions. It would be interesting to find an interpretation for the case of Hessenberg Y and V.

The structure of the paper. The structure of the paper is as follows. In the next section we present formulas without imposing any assumptions on $\{Y, V\}$. In Section 3 we specialize these formulas for the case when Y and V are Hessenberg and discuss some problems related to the design of the algorithm. Then in Section 4 we use these results to present a Schur-type algorithm for recursive factorization of a matrix R with a Hessenberg displacment structure. In Section 5 we present several important special Hessenberg matrices for which the algorithm is fast and has computational complexity $O(n^2)$.

2. Displacement structure of Schur complements

We start with recalling that the integer

$$\alpha = \operatorname{rank}(RY - VR) \tag{2.1}$$

is called the $\{Y, V\}$- displacement rank or R. Clearly, (2.1) implies that there exist two (non-unique) $n \times \alpha$ matrices $\{H, G\}$ such that

$$RY - VR = GH^T. \tag{2.2}$$

The pair of matrices $\{H, G\}$ in such a representation is called a *generator* of R [**28**] for the following reason. In the case when the spectra of the matrices Y and V are disjoint the equation (2.2) has a unique solution R. That means, assuming that Y and V are given and fixed, a generator contains the full information about R. In many cases the information on R can efficiently retrieved from the generator. This observation is the basis for the now standard technique to construct efficient algorithms, which can briefly be described as follows.

If two matrices $R^{(1)}$ and $R^{(2)}$ have the same displacement ranks, then the computation $R^{(1)} \longrightarrow R^{(2)}$ can be substituted by updating $\{H^{(1)}, G^{(1)}\} \longrightarrow \{H^{(2)}, G^{(2)}\}$. Since the n^2 entries of a matrix are compressed here into only $2\alpha n$ entries of its generator, this method typically leads to computational savings. Perhaps the first explicit formulation of this principle appeared in [**37**], where it was applied to speeding up the computation of the Cholesky factorization $R = LL^*$ for a positive definite Toeplitz-like matrix R. The essence of the Cholesky method is in recursive computing the successive Schur complements, and the following fact was established in [**37**]. The displacement structure of a matrix $R^{(1)}$ is inherited by its Schur complement $R^{(2)}$. Thus, the principle of generator updating can be applied. We next present a generalization of this observation.

LEMMA 2.1. *Let $R^{(1)}$ satisfy*

$$R^{(1)}Y^{(1)} - V^{(1)}R^{(1)} = G^{(1)}(H^{(1)})^T. \tag{2.3}$$

and let the latter matrices be partitioned as

$$R^{(1)} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, Y^{(1)} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}, V^{(1)} = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix},$$

$$G^{(1)} = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}, H^{(1)} = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}.$$

If R_{11} is nonsingular then the Schur complement $R^{(2)} = R_{22} - R_{21}R_{11}^{-1}R_{12}$ satisfies

$$R^{(2)}Y^{(2)} - V^{(2)}R^{(2)} = G^{(2)}(H^{(2)})^T,$$

with

$$\begin{array}{ll} G^{(2)} = G_2 - QG_1\ , & (H^{(2)})^T = H_2^T - H_1^T P \\ Y^{(2)} = Y_{22} - Y_{21}P, & V^{(2)} \quad\ = V_{22} - QV_{12}, \end{array} \tag{2.4}$$

where we denote

$$P := R_{11}^{-1}R_{12}, \qquad Q := R_{21}R_{11}^{-1}.$$

PROOF. From the standard Schur complement formula and (2.3) we obtain

$$R = \begin{bmatrix} I & 0 \\ Q & I \end{bmatrix} \begin{bmatrix} R_{11} & 0 \\ 0 & \tilde{R} \end{bmatrix} \begin{bmatrix} I & P \\ 0 & I \end{bmatrix}$$

we obtain

$$\begin{bmatrix} R_{11} & 0 \\ 0 & \tilde{R} \end{bmatrix} Y^{\#} - V^{\#} \begin{bmatrix} R_{11} & 0 \\ 0 & \tilde{R} \end{bmatrix} = \begin{bmatrix} I & 0 \\ -Q & I \end{bmatrix} GH^T \begin{bmatrix} I & -P \\ 0 & I \end{bmatrix}, \tag{2.5}$$

where

$$Y^{\#} = \begin{bmatrix} I & P \\ 0 & I \end{bmatrix} \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} I & -P \\ 0 & I \end{bmatrix} = \begin{bmatrix} * & * \\ * & Y_{22} - Y_{21}P \end{bmatrix},$$

$$V^{\#} = \begin{bmatrix} I & 0 \\ -Q & I \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ Q & I \end{bmatrix} = \begin{bmatrix} * & * \\ * & V_{22} - QV_{12} \end{bmatrix}.$$

By equating the (2,2) blocks on the both sides of (2.5) we obtain (2.4). □

If $Y^{(1)}$ is block upper triangular ($Y_{21} = 0$) and $V^{(1)}$ is block lower triangular ($V_{12} = 0$), then the matrices $Y^{(2)}$ and $V^{(2)}$ are obtained by just peeling off the first columns and the first rows of the matrices $Y^{(1)}$ and $V^{(1)}$, respectively. Therefore, in the triangular case only the first two formulas in (2.4) are actually used to update the generator $\{H^{(1)}, G^{(1)}\} \longrightarrow \{H^{(2)}, G^{(2)}\}$. Formulas (2.4) for this case can be found, e.g., in [**14, 12**]. However, in the general non-triangular case Lemma 2.1 says that now all four matrices have to be updated:

$$\{H^{(1)}, Y^{(1)}, V^{(1)}, G^{(1)}\} \longrightarrow \{H^{(2)}, Y^{(2)}, V^{(2)}, G^{(2)}\}. \tag{2.6}$$

This indicates that for general matrices $Y^{(1)}$ and $V^{(1)}$ there would be no computational or storage savings in the process.

In the next section we show that, nevertheless, in the case when the matrices $Y^{(1)}$ and $V^{(1)}$ are Hessenberg the formulas (2.4) can efficiently be applied if the generator is properly extended.

Let us discuss the relation of Lemma 2.1 with Gaussian elimination. Let $R^{(1)}$ be strongly nonsingular, i.e., it admits an LDU-factorization

$$R^{(1)} = LDU, \tag{2.7}$$

where L is unit lower triangular, U is unit upper triangular, and D is diagonal. The Gaussian elimination procedure computes (2.7) as follows:

1. One uses the entries of the first column and row of $R^{(1)} = \begin{bmatrix} r_{11}^{(1)} & r_{1,\cdot}^{(1)} \\ r_{\cdot,1}^{(1)} & R_{22}^{(1)} \end{bmatrix}$ to write down the first column l_1 to L, the first diagonal entry d_{11} to D, and the first row u_1 to U, as in

$$l_1 = \begin{bmatrix} 1 \\ \frac{1}{r_{11}^{(1)}} r_{\cdot,1}^{(1)} \end{bmatrix}, \qquad d_{11} = r_{11}^{(1)}, \qquad u_1 = \begin{bmatrix} 1 & \frac{1}{r_{11}^{(1)}} r_{1,\cdot}^{(1)} \end{bmatrix}. \tag{2.8}$$

2. One updates $R^{(1)} \longrightarrow R^{(2)} := R_{22}^{(1)} - r_{\cdot,1}^{(1)} \frac{1}{r_{11}^{(1)}} r_{1,\cdot}^{(1)}$.
3. Then one proceeds with the Schur complement $R^{(2)}$ recursively, i.e., writes down the second column to L, the second diagonal entry to D, and the second row to U, computes $R^{(3)}$, and so on.

We next discuss several issues occurring in applying formulas (2.4) to speeding up the GE procedure when $R^{(1)}$ satisfies (2.3) with Hessenberg $Y^{(1)}$ and $V^{(1)}$.

3. The case of Hessenberg matrices $Y^{(1)}$ and $V^{(1)}$

Hessenberg generator. From now on, we assume that $Y^{(1)}$ is an upper and $V^{(1)}$ a lower Hessenberg matrix. Our basic observation is that the matrices $\{Y^{(k)}, V^{(k)}\}$ remain in the same class during the recursion and, moreover, only a few their entries have to be updated. Indeed, in the Hessenberg case only the first entry of the column Y_{21} is nonzero, so that the computation of $Y^{(2)}$ by (2.4) reduces to the scheme

(3.1)

$$Y^{(2)} = Y_{22} - Y_{21}P = \begin{bmatrix} \times & \times & \cdots & \cdots & \times \\ \times & \times & \cdots & \cdots & \times \\ 0 & \times & \times & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \times \\ 0 & \cdots & 0 & \times & \times \end{bmatrix} - \begin{bmatrix} \times \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times & \times \end{bmatrix}.$$

This means that one has to update only the first row $y^{(k)}_{1,\cdot}$ of the *upper* Hessenberg $Y^{(k)}$ and the first column $v^{(k)}_{\cdot,1}$ of the *lower* Hessenberg $V^{(k)}$.

Another issue to address is how to obtain the entries of the matrices P and Q needed to run the recursion (2.4). These entries are essentially the elements of the first row and column of $R^{(1)}$, and it is typically not difficult to reconstruct them from $\{H^{(1)}, G^{(1)}\}$ in the cases of triangular $\{Y^{(1)}, V^{(1)}\}$, cf., e.g., with [**14**]. One way to handle them in the Hessenberg case is to just to store and to update them. Alltogether we have to update

$$\{H^{(k)}, v^{(k)}_{\cdot,1}, y^{(k)}_{1,\cdot}, G^{(k)}, r^{(k)}_{\cdot,1}, r^{(k)}_{1,\cdot}\}, \tag{3.2}$$

which contains about $2(\alpha+2)(n-k+1)$ parameters. We call (3.2) a *Hessenberg generator* of $R^{(k)}$.

We next present the details of these updates.

Reconstruction of the second row and column. In order to compute the first row and column of $R^{(2)}$ we actually need to know the entries of the first and the second rows and columns of $R^{(1)}$. This can be seen from the Schur complement formula $R^{(2)} = R_{22} - R_{21}R_{11}^{-1}R_{12}$, which implies

$$r^{(2)}_{\cdot,1} = r^{(1)}_{\cdot,2} - r^{(1)}_{\cdot,1}\frac{1}{r^{(1)}_{11}} r_{1,\cdot}\ .$$

Thus, we have to show how to reconstruct the second row and second column of $R^{(1)}$ from the Hessenberg generator (3.2). This will be described next.

Since the next two statements involve one matrix only, we omit the superscript for $R^{(k)}$ and for its entries and its generator as well.

LEMMA 3.1. *Let R satisfy (2.2), where $Y = [\ y_{ij}\ ]$ is upper and $V = [\ v_{ij}\ ]$ lower Hessenberg. Then the entries of the matrix R can be reconstructed recursively from $\{H, G\}$ and the first row and column of R with the help of the formulas*

$$r_{\cdot,k+1} = \frac{1}{y_{k+1,k}}\,(Vr_{\cdot,k} + Gh_k - \sum_{j=1}^{k} y_{jk} r_{\cdot,j}) \qquad (k = 1, \ldots, n-1),$$

where h_k denotes the k-th column of H^T, or

$$r_{k+1,\cdot} = \frac{1}{v_{k,k+1}} (r_{k,\cdot} Y - g_k^T H^T - \sum_{j=1}^{k} v_{kj} r_{j,\cdot}) \qquad (k = 1, \ldots, n-1),$$

where g_k denotes the k-th column of G.

PROOF. Let e_k $(k = 1, \ldots, n)$ denote the vectors of the standard basis in $\mathbb{C}^n$. We have

$$RYe_k = \sum_{j=1}^{k+1} u_{jk} r_{\cdot,j}$$

and

$$RYe_k = VRe_k + GH^T e_k = Gr_{\cdot,k} + Gh_k.$$

Comparing these relations we obtain the assertion. □

As a special case we obtain the following.

COROLLARY 3.2. *The second row and second column of R are given by*

$$r_{2,\cdot} = \frac{1}{v_{12}} (r_{1,\cdot}(Y - v_{11} I) + g_1^T H^T), \quad r_{\cdot,2} = \frac{1}{y_{21}} ((V - y_{11} I_n) r_{\cdot,1} + Gh_1), \tag{3.3}$$

where g_1 denotes the first column of G, and h_1 denotes the first column of of H^T.

4. The Schur-Hessenberg algorithm

We are now ready to present an algorithm that outputs the factorization

$$R = LDU \tag{4.4}$$

for the case when R satisfies (2.2). It recursively updates the Hessenberg generators (3.2) of the matrices $R^{(k)} = \left[r_{ij}^{(k)} \right]_{i,j=1}^{n+1-k}$ $(k = 1, \ldots, n)$, where $R^{(1)} = R$ and $R^{(k+1)}$ is the Schur complement of the $(1,1)$-entry $r_{11}^{(k)}$ of $R^{(k)}$. Since (3.2) includes the first row and the first column of $R^{(k)}$, the factors in (4.4) can be easily computed as follows. The k-th column l_k of L and the k-th row u_k of U are given by

$$l_k = \frac{1}{r_{11}^{(k)}} \begin{bmatrix} 0 \\ r_{\cdot,1}^{(k)} \end{bmatrix}, \quad u_k = \frac{1}{r_{11}^{(k)}} \begin{bmatrix} r_{1,\cdot}^{(k)} & 0 \end{bmatrix}.$$

Here "0" stand for a zero vector of appropriate length. The diagonal factor is given by $D = \operatorname{diag} \left[r_{11}^{(k)} \right]_{k=1}^{n}$.

THE SCHUR-HESSENBERG ALGORITHM

Initialization:

$$r_{\cdot,1}^{(1)} = r_{\cdot,1}, \quad r_{1,\cdot}^{(k)} = r_{1,\cdot},$$

$$G^{(1)} = G, \quad H^{(1)} = H, \quad Y^{(1)} = Y, \quad V^{(1)} = V.$$

Recursion: For $k = 1, \ldots, n-1$, compute

1. The k th entry of D, the k th column of L and k th row of U by

$$d^{(k)} = r_{11}^{(k)}, \quad l^{(k)} = \frac{1}{d^{(k)}} r_{\cdot,1}^{(k)}, \quad u^{(k)} = \frac{1}{d^{(k)}} r_{1,\cdot}^{(k)}.$$

2. The second column and the second row of $R^{(k)}$ by

$$r^{(k)}_{2,\cdot} = \frac{1}{y^{(k)}_{21}}((V^{(k)} - y^{(k)}_{11}I)r^{(k)}_{1,\cdot} + G^{(k)}h^{(k)}_1),$$

$$r^{(k)}_{\cdot,2} = \frac{1}{v^{(k)}_{12}}((Y^{(k)} - v^{(k)}_{11}I)r^{(k)}_{\cdot,2} + g^{(k)T}_1 H^{(k)T}).$$

where $g^{(k)}_1$ denotes the top row of $G^{(k)}$, and $h^{(k)}_1$ denotes the top row of $H^{(k)}$.

3. The first column and the first row of $R^{(k+1)}$ by

$$r^{(k+1)}_{\cdot,1} = r^{(k)'}_{\cdot,2} - r^{(k)}_{12}l^{(k)'}, \quad .r^{(k+1)}_{1,\cdot} = r^{(k)'}_{2,\cdot} - r^{(k)}_{21}u^{(k)'}$$

Here the prime means that the first component is peeled off.

4. The remaining parts of the new Hessenberg generator by

$$G^{(k+1)} = G^{(k)'} - l^{(k)}(g^{(k)}_1)^T, \quad H^{(k+1)} = H^{(k)'} - h^{(k)}_1(u^{(k)})^T.$$

$$Y^{(k+1)} = Y^{(k)''} - u^{(k)}_{21}e_1(u^{(k)'})^T, \quad V^{(k+1)} = V^{(k)''} - v^{(k)}_{12}l^{(k)'}e^T_1.$$

Here double prime means peeling off the top row and the first column of a matrix.

REMARK 4.1. *There are classes of matrices with a Hessenberg displacement structure in which Y is not upper but lower and V is not lower but upper Hessenberg. The following observation allows to use the Schur-Hessenberg algorithm in computations with these matrices. Let J denote the $n \times n$ counter-identity*

$$J = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}. \tag{4.5}$$

If R has the $\{Y, V\}$-displacement structure with lower *Hessenberg Y and* upper *Hessenberg V, then the matrix $\widehat{R} = JRJ$ has $(\widehat{U}, \widehat{V})$-displacement structure with $\widehat{Y} = JYJ$, which is upper, and $\widehat{V} = JVJ$, which is lower Hessenberg.*

In order to consider in the next subsection several special Hessenberg displacement structures for which our algorithm is fast, we need to recall several definitions for the general Hessenberg case. The upper Hessenberg matrix

$$H_Q = \begin{bmatrix} a_{n-1,n} & \cdots & a_{2,n} & a_{1,n} & a_{0,n} \\ a_{n-1,n-1} & \ddots & & \vdots & \vdots \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & a_{22} & a_{12} & a_{02} \\ 0 & \cdots & 0 & a_{11} & a_{01} \end{bmatrix} \qquad (a_{kk} \neq 0) \tag{4.6}$$

was associated in [**38**] with a system of polynomials $Q = \{Q_0(x), Q_1(x), ..., Q_n(x)\}$ defined by recurrence relations

$$x \cdot Q_{k-1}(x) = a_{k,k} \cdot Q_k(x) + a_{k-1,k} \cdot Q_{k-1}(x) + ... + a_{0,k} \cdot Q_0(x). \tag{4.7}$$

The matrix (4.6) was called a *confederate* matrix of $\{Q_n(x)\}$ [**38**]. Let $R = LHL^T$ with Hankel H, and L being lower triangular matrix capturing the coefficients

of $\{Q_k(x)\}$. It can be shown that R has the $\{H_Q, H_Q\}$-displacement rank two. Therefore matrices R having a low (Hessenberg) displacement rank

$$\alpha_Q(R) = \operatorname{rank}(H_Q^T \cdot R - R \cdot H_Q) \tag{4.8}$$

were called *polynomial Hankel-like* matrices in [**30**]. In fact, we have presented here a Schur algorithm for (non-symmetric) polynomial Hankel-like matrices.

REMARK 4.2. *We observe that the top row of the matrix* (4.6) *captures the coefficients of only one polynomial* $Q_n(x)$ *in* (4.7). *Hence the recursion* (2.4) *in view of the pattern in* (3.1) *can be seen as a polynomial recursion (see* [**21**] *for a special case). Some more details will be given in the next section.*

5. The fast Schur-Hessenberg algorithm

Here we discuss the complexity of the Schur-Hessenberg algorithm.

PROPOSITION 5.1. *Let* $M(n)$ *denote an upper bound on the costs of multiplying matrices* $Y^{(k)}$ *and* $V^{(k)}$ *by vectors* ($k = 1, 2, \ldots, n$). *Assuming that the displacement rank* α *is small compared to the size of* R, *the overall cost of the algorithm Schur-Hessenberg is*

$$C(n) = O(M(n)n + n^2). \tag{5.9}$$

Since for dense, unstructured Hessenberg matrices Y and V we have $M(n) = O(n^2)$, the Schur-Hessenberg algorithm will have complexity $O(n^3)$ in general, i.e. it is not fast. The algorithm becomes fast in case that Y and V are sparse or structured. If Y and V are, for example, banded, then we have $M(n) = O(n)$. This cost does not change if Y and V have a few more additional nonzero rows or columns. If Y and V are Toeplitz or Toeplitz-like (besides beeing Hessenberg), then $M(n) = O(n \log n)$. Also in this case a few unstructured rows or columns will not change the cost of the algorithm.

Fortunately, in many applications we have Y and V with this property. This concerns, for example, matrices that are related to orthogonal polynomials. As it was mentioned in the Introduction such matrices occur in solving interpolation and least squares problems as well as in the root localization problems for polynomials given in orthogonal polynomial expansions.

For example, if polynomials $Q_k(x)$ ($k = 1, \ldots, n-1$) are orthogonal with respect to some (definite or indefinite) weighted inner product on the real line, then they satisfy three-term recurrence relations of the form

$$Q_k(x) = (\alpha_k x - \beta_k)Q_{k-1}(x) - \gamma_k Q_{k-2}(x). \qquad (k = 1, \ldots, n-1)$$

Let $Q_n(x)$ be given by

$$Q_n(x) = q_n x Q_{n-1}(x) + q_{n-1}(x)Q_{n-1}(x) + \ldots + q_0 Q_0(x). \tag{5.10}$$

The related Hessenberg matrix is

$$Y = \begin{bmatrix} -\frac{q_{n-1}}{q_n} & -\frac{q_{n-2}}{q_n} & \cdots & & -\frac{q_1}{q_n} & -\frac{q_0}{q_n} \\ \frac{1}{\alpha_{n-1}} & \frac{\beta_{n-1}}{\alpha_{n-1}} & \frac{\gamma_{n-1}}{\alpha_{n-1}} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \frac{\beta_3}{\alpha_3} & \frac{\gamma_3}{\alpha_3} & 0 \\ \vdots & \vdots & \ddots & \frac{1}{\alpha_2} & \frac{\beta_2}{\alpha_2} & \frac{\gamma_2}{\alpha_2} \\ 0 & 0 & \cdots & 0 & \frac{1}{\alpha_1} & \frac{\beta_1}{\alpha_1} \end{bmatrix},$$

which differs from a tridiagonal matrix only by its top row. Thus $M(n) = O(n)$. Note that the latter matrix was introduced in [**1**] and called *comrade matrix*.

Another important example is when $Q_k(x)$ $(k = 1, \ldots, n-1)$ are Szegö polynomials, i.e. orthogonal on the unit circle with respect to some weighted inner product, and $Q_n(x)$ is given by (5.10). In this case (see [**17**]) there exist recurrence relations for $Q_k(x)$ that are completely described by the so-called *reflection coefficients* $\{\rho_k\}_{k=1}^n$. The corresponding Hessenberg matrix (see, e.g., [**39**] and references therein) is given by

$$Y = \begin{bmatrix} -\frac{q_{n-1}}{q_n} & -\frac{q_{n-2}}{q_n} & \cdots & & -\frac{q_1}{q_n} & -\frac{q_0}{q_n} \\ \mu_{n-1} & -\rho_{n-1}\rho_{n-2}^* & \cdots & \cdots & -\rho_{n-1}\mu_{n-2}\cdots\mu_2\rho_1^* & -\rho_{n-1}\mu_{n-2}\cdots\mu_2\rho_0^* \\ 0 & \mu_{n-2} & \ddots & & \vdots & \vdots \\ \vdots & \ddots & \ddots & -\rho_3\rho_2^* & -\rho_3\mu_2\rho_1^* & -\rho_3\mu_2\mu_1\rho_0^* \\ \vdots & & \ddots & \mu_2 & -\rho_2\rho_1^* & -\rho_2\mu_1\rho_0^* \\ 0 & \cdots & \cdots & 0 & \mu_1 & -\rho_1\rho_0^* \end{bmatrix},$$

where $\mu_k = \sqrt{1 - |\rho_k|^2}$ ($\mu_k := 1$ if $\rho_k = 1$). This matrix differs from unitary only by its first row. It is well-known (see, e.g., [**39**] and the references therein) that in this case $M(n) = O(n)$ so that the cost of the Schur-Hessenberg algorithm reduces again to $O(n^2)$ operations.

Returning to the remark 4.2 we mention that in the two above mentioned cases it is transparent that the recursion (2.4) in view of (3.1) can be seen as a polynomial recursion (see [**21**] for a special case).

References

[1] S. Barnett, *A companion matrix analogue for orthogonal polynomials*, Linear Algebra Appl., 12 (1975), 197 – 208.

[2] H. Bart, I. Gohberg and M.A. Kaashoek, *Minimal factorization of matrix and operator functions*, OT1, Birkhäuser Verlag, Basel, 1979.

[3] H. Bart, I. Gohberg, M.A. Kaashoek, P. Van Dooren, *Factorizations of transfer functions*, SIAM J. Control and Optim., 18 (1980), 675–696.

[4] J. Ball, I.Gohberg and L.Rodman, *Interpolation of rational matrix functions*, OT45, Birkhäuser Verlag, Basel, 1990.

[5] D. Bini, V. Pan, *Polynomial and matrix computations*, vol.1, Birkhäuser, Boston, 1994.

[6] A. Bultheel, M. Van Barel, *Linear algebra, rational approximation, and orthogonal polynomials*, Elsevier, Amsterdam, 1997.

[7] J. Chun, T. Kailath and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorizations*, J. Sci. Stat. Comput., 8 (1987), 899–913.

[8] J.M.Delosme, *Fast algorithms for finite shift-rank processes*, Ph.D.Thesis,(Director: Martin Morf), Stanford University, Stanford, CA, 1982.

[9] D. Fasino, *Preconditioning finite moment problems*, J. Comp. Appl. Math., 65 (1995), 145–155.

[10] B. Friedlander, M. Morf, T. Kailath, L. Ljung, *New inversion formulas for matrices classified in terms of their distance from Toeplitz matrices*, Linear Algebra and Appl., 27 (1979) 31–60.

[11] L. Gemignani, *A fast algorithm for generalized Hankel matrices arising in finite-moment problems*, Linear Algebra Appl., 267 (1997), 41–52.

[12] I. Gohberg, T. Kailath, V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. of Comp., 64 (1995), 1557–1576.

[13] I.Gohberg, V.Olshevsky, *Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems*, Integral Equations and Operator Theory, 20, No. 1 (1994), 44 – 83.

[14] I. Gohberg, V. Olshevsky, *Complexity of multiplication with vectors for structured matrices*, Linear Algebra Appl., 202 (1994), 163 – 192.

[15] I. Gohberg, V. Olshevsky, *Fast inversion of Chebyshev-Vandermonde matrices*, *Numerische Mathematik*, 67, 1, (1994), 71–92.

[16] S.A. Gustafson, *On computational applications of the theory of moment problems*, Rocky Mountain J.Math., 2 (1974), 227–240.

[17] U. Grenader and G. Szegő, *Toeplitz forms and Applications*, University of California Press, 1958.

[18] Higham N, *Fast solution of Vandermonde–like systems involving orthogonal polynomials*, IMA J. Numerical Analysis, 8, 473 – 486.

[19] G. Heinig *Ein Prinzip zur Invertierung einiger Klassen linearer Operatoren*, Proceedings of the 7th Conference on Methods in Mathematical Physics (7th TMP), TH Karl-Marx-Stadt 1979, vol.2, pp.45–50.

[20] G. Heinig, *Inversion of Toeplitz-like matrices via generalized Cauchy matrices and rational interpolation*, In: U. Helmke, R.Mennicken, J. Sauer (Eds.), Systems and Networks: Mathematical Theory and Applications, vol.2, pp. 707–712.

[21] G. Heinig, *Fast and superfast algorithms for Hankel-like matrices related to orthogonal polynomials*, In: *Numerical Analysis and Its Applications*, Lectures Notes in Computer Science, vol. 1988, Springer-Verlag, Berlin, Heidelberg 2001, pp. 385–392.

[22] G. Heinig, A. Bojanczyk, *Transformation techniques for Toeplitz and Toeplitz-plus-Hankel matrices*, Linear Algebra Appl. 254 (1997), 193–226, and 27 (1998), 11–36.

[23] G. Heinig, W. Hoppe, K. Rost, *Structured matrices in interpolation and approximation problems*, Wiss. Zeitschr. d. TU Karl-Marx-Stadt, 31, 2 (1989), 196–202.

[24] G. Heinig, P. Jankowski, K. Rost, *Fast algorithms for Toeplitz-plus-Hankel matrices*, Numerische Mathematik, 52 (1988), 628–665 .

[25] G. Heinig, K. Rost, *Algebraic Methods for Toeplitz-like matrices and operators*, Akademie-Verlag Berlin and Birkhäuser Basel, Boston, Stuttgart, 1984.

[26] T. Kailath, *Some new algorithms for recursive estimation in constant linear systems*, IEEE transactions on Information Theory, IT-19, Nov. 1973, 750–760.

[27] T. Kailath, *Displacement structure and array algorithms*, In: *Fast reliable algorithms for matrices with structure*, (T. Kailath, A.H. Sayed, Eds.),SIAM Publications, Philadelphia 1999.

[28] T. Kailath, S.Y. Kung, M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68, 2 (1979), 395–407.

[29] T. Kailath, V. Olshevsky, Displacement structure approach to Chebyshev-Vandermonde matrices, Integral Equations and Operator Theory, 22 (1995), 65–92.

[30] T. Kailath, V. Olshevsky *Displacement structure approach to discrete transform based preconditioners of G.Strang type and of T.Chan type,* Calcolo (Italian journal), 33 (1996), 191–208.
The full version can be downloaded from www.cs.gsu.edu/~matvro/papers/prec.ps

[31] T. Kailath, V. Olshevsky, *Displacement Structure Approach to Polynomial Vandermonde and Related Matrices*, Linear Algebra Appl., 261 (1997), 49–90.

[32] T. Kailath, A.H. Sayed, *Displacement structure: Theory and applications*, SIAM Review, 37, 3 (1995), 297–386.

[33] T. Kailath, *Fast reliable algorithms for matrices with structure*, In: (T. Kailath, A.H. Sayed, Eds.), SIAM Publications, Philadelphia 1999.

[34] H. Lev-Ari, *Nonstationary lattice-filter modeling*, Ph.D. Thesis, (Director: Thomas Kailath), Stanford University, December 1983.

[35] H. Lev-Ari, T. Kailath, *Triangular factorization of structured Hermitian matrices*, Operator Theory: Advances and Appl., 18 (1986), 301–324.

[36] M. Morf, *Fast algorithms for mulivariable systems*, Ph.D. Thesis, (Director: Thomas Kailath), Stanford University, Dept. of Electrical Engineering, August 1974.

[37] M. Morf, *Doubling algorithms for Toeplitz and related equations*, Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing, Denver, 1980, 954–959.

[38] J. Maroulas, S. Barnett, *Polynomials with respect to a general basis. I. Theory*, J. of Math. Analysis and Appl., **72** : 177 -194 (1979).

[39] V. Olshevsky, *Eigenvector computation for almost unitary Hessenberg matrices and inversion of Szegö- Vandermonde matrices via discrete transmission lines,* Linear Algebra Appl., 285 (1998), 37–67.

[40] L. Reichel, G. Opfer, *Chebyshev–Vandermonde systems,* Math. of Computation, 57, 196 (1991), 703 – 721.

[41] I. Schur, *Über Potenzreihen die im Inneren des Einheitskreises beschränkt sind*, Journal für die Reine und Angewandte Mathematik, 147 (1917), 205–232. English translation in *Operator Theory : Advances and Applications* (I.Gohberg. ed.), vol.18 (1986), 31–88, Birkhäuser, Boston, 1986.

[42] L.A. Sakhnovich, *On similarity of operators* (in Russian), Sibirskij Mat.J. 8, 4 (1972), 8686–883.

[43] L.A. Sakhnovich, *On the integral equation with kernel depending on the difference of the arguments* (in Russian), Matem. Issledovanya, Kishinev, 8, 2 (1973), 138–146.

[44] L.A. Sakhnovich, *The factorization of the operator-valued transfer functions,* Soviet Math. Dokl., 17 (1976), 203–207.

[45] L.Sakhnovich, *Factorization problems and operator identities*, Russian Mathematical Surveys, 41, 1 (1986), 1 – 64.

[46] A.H. Sayed, H. Lev-Ari, T. Kailath, *Fast triangular factorization of the sum of quasi-Toeplitz and quasi-Hankel matrices*, Linear Algebra Appl., 191 (1993), 77–106.

[47] C.J. Zarowski, *A Schur algorithm and linearly connected processor array for Toeplitz-plus-Hankel matrices*, IEEE Transact. on Inf. Theory, 40, 8 (1992), 2065–2078.

Kuwait University, Dept. of Mathematics&Computer Science, P.O.Box 5969, Safat 13060, Kuwait

E-mail address: georg@mcs.sci.kuniv.edu.kw

Department of Mathematics and Statistics, Georgia State University, Atlanta, GA 30303

E-mail address: volshevsky@gsu.edu