

On the computation of the nullspace of Toeplitz–like matrices

Nicola Mastronardi Marc Van Barel
Raf Vandebril

Report TW 516, January 2008



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

On the computation of the nullspace of Toeplitz–like matrices

Nicola Mastronardi Marc Van Barel
Raf Vandebril

Report TW 516, January 2008

Department of Computer Science, K.U.Leuven

Abstract

The knowledge of the nullspace and its size of some structured matrices, like Hankel and Toeplitz matrices, is important in many applications arising in system theory.

In this paper we describe an algorithm to compute the kernel of Toeplitz and Hankel matrices based on the generalized Schur algorithm.

Keywords : nullspace, Toeplitz matrix, Hankel matrix, generalized Schur algorithm.

MSC : Primary : 65F30; Secondary : 15A15, 15A09, 15A23.

ON THE COMPUTATION OF THE NULLSPACE OF TOEPLITZ-LIKE MATRICES*

NICOLA MASTRONARDI[†], MARC VAN BAREL , AND RAF VANDEBRIL[‡]

Abstract. The knowledge of the nullspace and its size of some structured matrices, like Hankel and Toeplitz matrices, is important in many applications arising in system theory.

In this paper we describe an algorithm to compute the kernel of Toeplitz and Hankel matrices based on the generalized Schur algorithm.

Key words. nullspace, Toeplitz matrix, Hankel matrix, generalized Schur algorithm

AMS subject classifications. 15A15, 15A09, 15A23

1. Introduction. The knowledge of the nullspace and its size of some structured matrices, like Hankel and Toeplitz matrices, is important in many applications arising in system theory [1, 6, 8].

The properties of the kernel of Hankel matrices have been analyzed in [5].

The generalized Schur algorithm (GSA) [7] allows to compute many classical factorizations of a matrix, such as the QR -factorization, the LDL^T -factorization of a symmetric matrix, the “Cholesky” factorization of a symmetric positive definite matrix. If the involved matrix is Toeplitz-like structured, the computation of such factorizations can be done in a fast way via GSA.

In this paper we describe an algorithm to compute the kernel of Toeplitz and Hankel matrices based on GSA.

The paper is organized as follows. In section 2 the properties of the nullspace of Hankel matrices are shortly recalled. The generalized Schur algorithm and its extension to compute the nullspace of Hankel and Toeplitz matrices is described in section 3. Some numerical examples are reported in section 4 followed by the conclusions in section 5.

2. The kernel of Toeplitz-like matrices. The properties of the kernel of Hankel matrices have been analyzed in [5]. In the sequel, we shortly recall the properties of the nullspace of Hankel matrices described in [5], useful to design the algorithm for computing it.

The properties of the kernel of Toeplitz matrices can be easily derived from those of Hankel matrices, since Toeplitz matrices can be obtained from Hankel ones, just exchanging the order of the columns (or, alternatively, of the rows).

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

[†]Istituto per le Applicazioni del Calcolo “M. Picone”, sede di Bari, Consiglio Nazionale delle Ricerche, Via G. Amendola, 122/D, I-70126 Bari, Italy. (n.mastronardi@ba.iac.cnr.it).

[‡]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven (Heverlee), Belgium. ({Raf.Vandebril, Marc.VanBarel, Nicola.Mastronardi}@cs.kuleuven.be).

DEFINITION 2.1. Let $b = [b_1, b_2, \dots, b_{n-1}, b_n] \in \mathbb{R}^n$. The following Hankel matrices $H^{[i,n]}(b)$, $i = 1, \dots, n$, can be generated by b .

$$H^{[1,n]}(b) = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}, \quad H^{[2,n]}(b) = \begin{bmatrix} b_1 & b_2 \\ b_2 & \ddots \\ \ddots & \ddots \\ \ddots & b_{n-1} \\ b_{n-1} & b_n \end{bmatrix}, \quad H^{[3,n]}(b) = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_2 & b_3 & \ddots \\ b_3 & \ddots & b_{n-2} \\ \ddots & b_{n-2} & b_{n-1} \\ b_{n-2} & b_{n-1} & b_n \end{bmatrix},$$

$$H^{[n-1,n]}(b) = \begin{bmatrix} b_1 & b_2 & \ddots & \ddots & b_{n-1} \\ b_2 & \ddots & \ddots & b_{n-1} & b_n \end{bmatrix}, \quad H^{[n,n]}(b) = [b_1 \ b_2 \ \cdots \ b_{n-1} \ b_n]$$

DEFINITION 2.2. Let us denote by I_n the identity matrix of order n . Let $s \leq n$. A sequence of vectors $x_1, x_2, \dots, x_r \in \mathbb{C}^n$, is called a U -chain of length r if there is a vector $p \in \mathbb{C}^s$ ($p \neq 0, s + r = n + 1$) such that

$$x_k = I_n(:, k : s + k - 1)p, \quad (k = 1, 2, \dots, r).$$

The vector p is said a generating vector of the U -chain. The generating vector is essentially unique, i.e., is unique up to multiplication by a constant different from zero. Hence,

$$[x_1, \ x_2, \ \cdots \ x_r] = \begin{bmatrix} p_1 & 0 & \ddots & 0 \\ p_2 & p_1 & \ddots & \ddots \\ \ddots & p_2 & \ddots & 0 \\ p_s & \ddots & \ddots & p_1 \\ 0 & p_s & \ddots & p_2 \\ \ddots & \ddots & \ddots & \ddots \\ 0 & \ddots & 0 & p_s \end{bmatrix}_{n \times r}.$$

The following result holds.

COROLLARY 2.3.

$$\text{rank}(H^{[k,n]}(b)) = \min\{k, n - k + 1, \text{rank}(H^{[l,n]}(b))\}, \quad k = 1, \dots, n,$$

with

$$l = \begin{cases} n/2 & \text{if } l \text{ is even,} \\ (n+1)/2 & \text{if } l \text{ is odd.} \end{cases}$$

The rank of the Hankel matrices $H^{[k,n]}(b)$, $k = 1, \dots, n$, has the behavior depicted in Figure 2.1.

The following result characterizes the kernel of the Hankel matrices $H^{[k,n]}(b)$, $k = 1, \dots, n$.

THEOREM 2.4. Let $H \equiv H^{[k,n]}(b)$, for some $k \in \{1, 2, \dots, n\}$, be an Hankel matrix with non-trivial kernel. Then the following assertions hold:

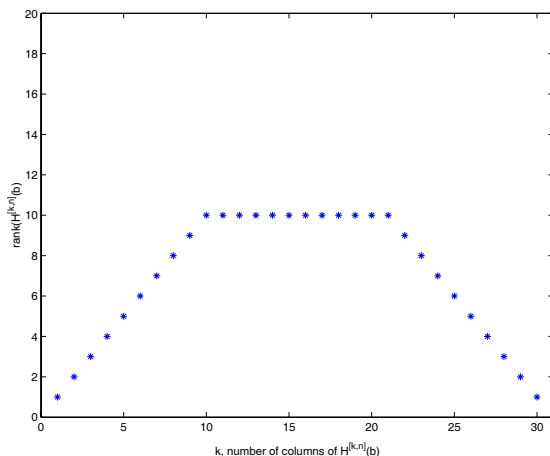


FIG. 2.1. Example of the behaviour of the rank of the Hankel matrices $H^{[k,n]}(b)$, $k = 1, \dots, n$, with $n = 30$. The number of columns of the matrices $H^{[k,n]}(b)$ is reported on the x -axis. The rank of the matrices $H^{[k,n]}(b)$ is reported on the y -axis.

1. If $H^{[k,n]}$ has not maximal rank, then the kernel of H and its transposed $H^{[k,n]T}$ is the linear hull of one U -chain generated by the same vector.
2. In general, the kernel of $H^{[k,n]}$ can be represented as the linear hull of one or two U -chains.

3. Generalized Schur Algorithm. The generalized Schur algorithm allows to compute many classical factorizations of a matrix, such as the QR -factorization, the LDL^T -factorization of a symmetric matrix, the ‘‘Cholesky’’ factorization of a symmetric positive definite matrix and the inverse of the Cholesky factor. If the involved matrix is Toeplitz-like structured, the computation of such factorizations can be done in a fast way via GSA. A comprehensive treatment of the topic can be found in [7].

The proposed algorithm for computing the left nullspace of a Toeplitz-like structured matrix is based on GSA for computing the Cholesky factor and its inverse. For the sake of simplicity, in this section we will consider a simple Toeplitz matrix instead of a general Toeplitz-like one. We first describe how GSA can compute the R factor of the QR factorization of a full rank Toeplitz matrix T , i.e., the Cholesky factor of $T^T T$, and the inverse of the R factor.

Let

$$T = \begin{bmatrix} t_n & t_{n-1} & \ddots & t_1 \\ t_{n+1} & t_n & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & t_{m-1} \\ t_{m+n-1} & \ddots & t_{m+1} & t_m \end{bmatrix}, \quad (3.1)$$

with $m \geq n$. Let us first consider the case $\text{rank}(T) = \rho = n$.

Define

$$M = \left[\begin{array}{c|c} T^T T & I_n \\ \hline I_n & 0_n \end{array} \right], \quad (3.2)$$

with I_n and 0_n the identity matrix and the null matrix of order n , respectively. The R factor of the QR -factorization of T and its inverse R^{-1} can be retrieved from the LDL^T factorization of M , with L lower triangular and D diagonal matrices, respectively. In fact, it can be easily shown that

$$M = LDL^T = \left[\begin{array}{c|c} R^T & \\ \hline R^{-1} & R^{-1} \end{array} \right] \left[\begin{array}{c|c} I_n & \\ \hline & -I_n \end{array} \right] \left[\begin{array}{c|c} R & R^{-T} \\ \hline & R^{-T} \end{array} \right]. \quad (3.3)$$

Therefore, to compute R and R^{-1} it is sufficient to compute the first n columns of L . This can be accomplished with $O(n^2)$ floating point operations by means of GSA.

We first describe how GSA can be used to compute R and its inverse R^{-1} .

Let $Z \in \mathbb{R}^{n \times n}$ be the shift matrix

$$Z = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 1 & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}_{n \times n} \quad (3.4)$$

and

$$\Phi = Z \oplus Z. \quad (3.5)$$

It turns out that

$$M - \Phi M \Phi^T = G \hat{D} G^T,$$

with $\hat{D} = \text{diag}(1, 1, -1, -1)$ and $G \in \mathbb{R}^{2n \times 4}$, called a *generator* matrix. Hence the *displacement rank* of M w.r.t. Φ , defined as the rank of $M - \Phi M \Phi^T$, is 4. The matrix Φ is called *displacement* matrix.

Denote $v = T^T(T(:, 1))$. The columns of G can be chosen as

$$\begin{aligned} G(:, 1) &= \frac{1}{\sqrt{v(1)}} \left[v^T \mid e_1^{(n)T} \right]^T, \\ G(:, 2) &= \left[0 \mid t_{n-1} \ t_{n-2} \ \cdots \ t_1 \mid 0 \ \cdots \ 0 \right]^T, \\ G(:, 3) &= \left[0 \mid G^T(2 : 2n - 1, 1) \right]^T, \\ G(:, 4) &= \left[0 \mid t_{m+n-1} \ \cdots \ t_{m+2} \ t_{m+1} \mid 0 \ \cdots \ 0 \right]^T. \end{aligned} \quad (3.6)$$

Since the number of columns of the generator matrix G is $4 \ll n$, the GSA for computing R and R^{-1} has $O(n^2)$ computational complexity. It relies only on the knowledge of the matrix G and not on the knowledge of the matrix T itself.

Each iteration of GSA relies on the following steps.

- Reduction of the generator matrix in *proper form*, i.e., at the i -th iteration all the entries of the i -th row of the generator matrix are annihilated but one by means of a sequence of Givens and hyperbolic rotations. The column of the generator matrix corresponding to the latter entry is called *pivot* column.

- Multiplication of the pivot column by the displacement matrix. In this way, all the entries of the i -th row of the generator matrix are now zero.

GSA for computing R , the Cholesky factor and R^{-1} , can be summarized in the following algorithm, written in a `matlab`-like¹ style.

```
% ALGORITHM 1. Generalized Schur algorithm.
% INPUT:  G, the generator matrix of the Toeplitz matrix T.
% OUTPUT: R and R^{-1}, with R the R factor of a QR-factorization of T.
function [R, R^{-1}] =schur(G);

1) for k = 1 : n,
2)   [c_G, s_G] = giv(G(k, 1), G(k, 2));
3)   G(k : n + k, 1 : 2) = G(k : n + k, 1 : 2) * [ c_G  s_G ; -s_G  c_G ];
4)   [c_G, s_G] = giv(G(k, 3), G(k, 4));
5)   G(k : n + k - 1, 3 : 4) = G(k : n + k - 1, 3 : 4) * [ c_G  s_G ; -s_G  c_G ];
6)   [c_H, s_H] = hyp(G(k, 1), G(k, 3));
7)   G(k : n + k, [1, 3]) = G(k : n + k, [1, 3]) * [ c_H  -s_H ; -s_H  c_H ];
8)   R(k, k : n) = G(k : n, 1)^T;
9)   R^{-1}(1 : k, k) = G(n + 1 : n + k, 1);
10)  G(:, 1) = Phi(G(:, 1));
11) end
```

We have denoted by `giv` the function that computes the parameters $[c_G, s_G]$ of the Givens rotation G :

$$[c_G, s_G] = \text{giv}(w_1, w_2) \text{ such that } \begin{bmatrix} c_G & s_G \\ -s_G & c_G \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \sqrt{w_1^2 + w_2^2} \\ 0 \end{bmatrix},$$

with $w \in \mathbb{R}^2$. Moreover, suppose $w_1 > w_2$. We have denoted by `hyp` the function that computes the parameters $[c_H, s_H]$ of the hyperbolic rotation² H ,

$$[c_H, s_H] = \text{hyp}(w_1, w_2) \text{ such that } \begin{bmatrix} c_H & -s_H \\ -s_H & c_H \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \sqrt{w_1^2 - w_2^2} \\ 0 \end{bmatrix}.$$

The function `function[G] = gener(T)` computes the generator matrix G , having as input the Toeplitz matrix T .

Each iteration of GSA (algorithm 1) involves two products of Givens rotations by an $n \times 2$ matrix, each of those can be accomplished with $6n$ floating point operations, followed by the product of an hyperbolic rotation by an $n \times 2$ matrix, accomplished with $6n$ floating point operations. Therefore the computational complexity of GSA is $18n^2$ floating point operations. We remark that GSA exhibits a lot of parallelism that can be exploited to reduce the computational complexity. For instance, the products involving the Givens rotations and the hyperbolic rotations can be done in parallel.

If only the R factor is needed, the computation can be done via GSA, considering only the first n rows and columns of M , i.e., $T^T T$. Therefore, a generator matrix \hat{G} for

¹`Matlab` is a registered trademark of The MathWorks, Inc.

²Hyperbolic rotations can be computed in different ways. For “stable” implementations see [2, 3].

T w.r.t. the displacement matrix Z in (3.4) can be straightforwardly obtained from the one of M , considering only the first n rows of G in (3.6). In fact, the following one is a generator matrix of $T^T T$ w.r.t. Z .

$$\begin{aligned}\hat{G}(:, 1) &= \frac{1}{\sqrt{v(1)}}v, \\ \hat{G}(:, 2) &= [0 \quad t_{n-1} \quad t_{n-2} \quad \cdots \quad t_1]^T, \\ \hat{G}(:, 3) &= [0 \quad G^T(2:n-1, 1)]^T, \\ \hat{G}(:, 4) &= [0 \quad t_{m+n-1} \quad \cdots \quad t_{m+2} \quad t_{m+1}]^T.\end{aligned}\tag{3.7}$$

The algorithm for computing the R factor varies from ALGORITHM 1 only for the length of the involved matrices. In fact, the sizes of the generator matrix, and therefore of the matrix R , are $n \times 4$ and $n \times n$, respectively.

```
% ALGORITHM 2. Generalized Schur algorithm.
% INPUT:  $\hat{G}$ , the generator matrix of the Toeplitz matrix  $T$ .
% OUTPUT:  $R$ , the  $R$  factor of a  $QR$ -factorization of  $T$ .
function [R] =schur( $\hat{G}$ );
```

```
1) for  $k = 1 : n$ ,
2)   [ $c_G, s_G$ ] = giv( $\hat{G}(k, 1), \hat{G}(k, 2)$ );
3)    $\hat{G}(k:n, 1:2) = \hat{G}(k:n, 1:2) \begin{bmatrix} c_G & s_G \\ -s_G & c_G \end{bmatrix}$ ;
4)   [ $c_G, s_G$ ] = giv( $\hat{G}(k, 3), \hat{G}(k, 4)$ );
5)    $\hat{G}(k:n, 3:4) = \hat{G}(k:n, 3:4) \begin{bmatrix} c_G & s_G \\ -s_G & c_G \end{bmatrix}$ ;
6)   [ $c_H, s_H$ ] = hyp( $\hat{G}(k, 1), \hat{G}(k, 3)$ );
7)    $\hat{G}(k:n, [1, 3]) = \hat{G}(k:n, [1, 3]) \begin{bmatrix} c_H & -s_H \\ -s_H & c_H \end{bmatrix}$ ;
8)    $R(k, k:n) = \hat{G}(k:n, 1)^T$ ;
9)    $\hat{G}(:, 1) = Z\hat{G}(:, 1)$ ;
10) end
```

The structure and the computation of the R factor of a singular Toeplitz matrix of rank $\rho < n \leq m$, has been considered in [4]. Let $R_a^T R_a$ be the ‘‘Cholesky’’ decomposition of the positive semidefinite matrix $T^T T$. Then, the *rank profile* of R_a can be either of *trapezoidal* type (see figure 3.1), or of *double trapezoidal* type (see

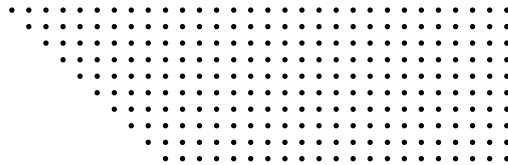


FIG. 3.1. Example of the rank profile of the R factor of the QR -factorization of a singular Toeplitz matrices (trapezoidal case).

figure 3.2). The trapezoidal case happens when the nullspace of the Toeplitz matrix is generated by one U -chain sequence, while the double trapezoidal case happens when the nullspace of the Toeplitz matrix is generated by two U -chain sequences [5].

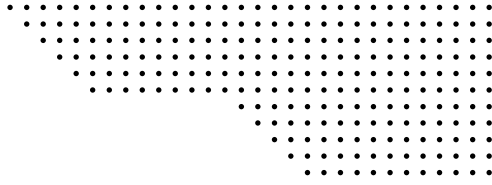


FIG. 3.2. Example of the rank profile of the R factor of the QR-factorization of a singular Toeplitz matrices (double trapezoidal case).

We will show how the nullspace of singular Toeplitz matrices can be computed by a modification of GSA.

We shortly describe how GSA is modified in [4] to compute R_a . Since the interest is in computing R_a , also in this case GSA is not applied to the extended matrix M , but directly to the matrix $T^T T$ w.r.t. the displacement matrix Z in (3.4). Again, a generator matrix \tilde{G} is given by (3.7).

If $\text{rank}(T) = \rho < n$, then, at the k -th iteration, $k \leq \rho$, the hyperbolic rotation at step 6) of ALGORITHM 1 can not be applied, because $\tilde{G}(k, 1) = \tilde{G}(k, 3)$, for some $k \in \{1, \dots, n\}$. Moreover, it turns out that the vectors $\tilde{G}(k : n, 1)$ and $\tilde{G}(k : n, 3)$ are equal. Hence, the first upper trapezoidal part of the matrix R_a is already computed.

The computation continues dropping the first and the third columns of \tilde{G} . Since only two columns of \tilde{G} are now involved in the computation, the second and the fourth one, the steps 2), 3), 4) and 5) in ALGORITHM 1 are skipped. The computation continues until $\tilde{G}(j, 2) = \tilde{G}(j, 4)$, for some $j \in \{k + 1, \dots, n\}$. Hence, the second trapezoidal part of R_a is also computed. The gap between the two trapezoidal forms can be bigger than 1. This happens when the entries $k + 1, k + 2, \dots, k + l \leq n$, of $\tilde{G}(:, 2)$ and $\tilde{G}(:, 4)$, at the end of the k -th iteration, are zero. In the latter case we have a gap of length $l + 1$.

In order to show how to compute the nullspace of a Toeplitz matrix, let us consider the following modified augmented matrix

$$M_\varepsilon = \left[\begin{array}{c|c} T^T T + \varepsilon^2 I_n & I_n \\ \hline I_n & 0_n \end{array} \right] = M + \varepsilon^2 \left[\begin{array}{c|c} I_n & 0_n \\ \hline 0_n & 0_n \end{array} \right], \quad (3.8)$$

with $\varepsilon > 0$. The displacement rank of M_ε with respect to Φ is still 4, and a generator matrix G_ε of M_ε w.r.t. Φ can be constructed as follows. Denote $\tilde{v} = (T^T T + \varepsilon^2 I_n)e_1$. The columns of G_ε are

$$\begin{aligned} G_\varepsilon(:, 1) &= \frac{1}{\sqrt{\tilde{v}(1)}} \left[\tilde{v}^T \mid e_1^{(n)T} \right]^T, \\ G_\varepsilon(:, 2) &= \left[0 \quad t_{n-1} \quad t_{n-2} \quad \cdots \quad t_1 \mid 0 \quad \cdots \quad 0 \right]^T, \\ G_\varepsilon(:, 3) &= \left[0 \quad G_\varepsilon^T(2 : 2n - 1, 1) \right]^T, \\ G_\varepsilon(:, 4) &= \left[0 \quad t_{m+n-1} \quad \cdots \quad t_{m+2} \quad t_{m+1} \mid 0 \quad \cdots \quad 0 \right]^T. \end{aligned} \quad (3.9)$$

We examine the behaviour of GSA applied to the latter problem while $\varepsilon \rightarrow 0^+$. For simplicity, we assume that the first ρ columns of T are linearly independent. At the $(\rho + 1)$ -th iteration of GSA, after steps 2), 3), 4) and 5) of ALGORITHM 1, it turns

out [4] that

$$G_\varepsilon(\rho, 1) - G_\varepsilon(\rho, 3) = \delta_\varepsilon$$

and

$$|G_\varepsilon(j, 1) - G_\varepsilon(j, 3)| = c_{j,\varepsilon} \leq \gamma_\varepsilon, \quad j = \rho + 1, \rho + 2, \dots, n,$$

with $\delta_\varepsilon \in \mathbb{R}_+^*$ depending on ε and such that $\delta_\varepsilon, \gamma_\varepsilon \rightarrow 0^+$ as $\varepsilon \rightarrow 0^+$.

The following theorem holds.

THEOREM 3.1. *Let $T \in \mathbb{R}^{m \times n}$, $m \geq n$, $\text{rank}(T) = \rho < n$. Suppose the first ρ columns of T are linearly independent. Let R be the R -factor of the QR factorization of T , with $R \in \mathbb{R}^{\rho \times n}$ upper trapezoidal. Let \mathcal{S}_1 be the subspace of \mathbb{R}^n generated by R^T . and let \mathcal{S}_2 be its orthogonal complement. Let $M = T^T T =$ and $M_\varepsilon = T^T T + \varepsilon^2 I_n$, with $\varepsilon \in \mathbb{R}_+^*$. Let R_ε be the Cholesky factor of M_ε and $\mathcal{S}_3 = \text{span}(R_\varepsilon^{-1}(:, \rho + 1 : n))$. Then*

$$\mathcal{S}_3 \rightarrow \mathcal{S}_2, \quad \text{as } \varepsilon \rightarrow 0^+. \quad (3.10)$$

Proof. Let $M = U\Lambda U^T$ be the spectral decomposition of M , with $U \in \mathbb{R}^{n \times \rho}$ orthogonal and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_\rho)$, with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_\rho > 0$. Of course, $\text{span}(U) = \mathcal{S}_1$. Let

$$M_\varepsilon = U_\varepsilon \Lambda_\varepsilon U_\varepsilon^T$$

be the spectral decomposition of M_ε , with $U_\varepsilon = [u_1^{(\varepsilon)}, \dots, u_n^{(\varepsilon)}] \in \mathbb{R}^{n \times n}$ orthogonal and $\Lambda_\varepsilon = \text{diag}(\lambda_1 + \varepsilon^2, \dots, \lambda_\rho + \varepsilon^2, \varepsilon^2, \dots, \varepsilon^2)$. Let e_l , $l \in \{\rho + 1, \rho + 2, \dots, n\}$ be the l -th vector of the canonical basis of \mathbb{R}^n . Then

$$\begin{aligned} M_\varepsilon^{-1} e_l &= U_\varepsilon \Lambda_\varepsilon^{-1} U_\varepsilon^T e_l \\ &= \sum_{i=1}^{\rho} \frac{u_i^{(\varepsilon)T} e_l}{\lambda_i + \varepsilon^2} u_i^{(\varepsilon)} + \sum_{i=\rho+1}^n \frac{u_i^{(\varepsilon)T} e_l}{\varepsilon^2} u_i^{(\varepsilon)}. \end{aligned}$$

Therefore, as $\varepsilon \rightarrow 0^+$,

$$M_\varepsilon^{-1} e_l \rightarrow \frac{1}{\varepsilon^2} \sum_{i=\rho+1}^n u_i^{(\varepsilon)T} e_l u_i^{(\varepsilon)}, \quad (3.11)$$

that is, $M_\varepsilon^{-1} e_l$ becomes closer and closer to a vector belonging to \mathcal{S}_2 . On the other hand,

$$\begin{aligned} M_\varepsilon^{-1} e_l &= (R_\varepsilon^T R_\varepsilon)^{-1} e_l \\ &= R_\varepsilon^{-1} R_\varepsilon^{-T} e_l \end{aligned} \quad (3.12)$$

Hence, (3.10) follows from (3.11) and (3.12). \square

At step 6) of the $(\rho + 1)$ -th iteration of ALGORITHM 1, the corresponding hyperbolic rotation to apply to $G_\varepsilon(\rho + 1 : 2n, 1)$ and $G_\varepsilon(\rho + 1 : 2n, 3)$ is

$$H_n = \frac{G_\varepsilon(\rho, 3)}{\sqrt{\delta_\varepsilon^2 + 2G_\varepsilon(n, 3)\delta_\varepsilon}} \begin{bmatrix} 1 + \frac{\delta_\varepsilon}{G_\varepsilon(\rho+1, 3)} & -1 \\ -1 & 1 + \frac{\delta_\varepsilon}{G_\varepsilon(\rho+1, 3)} \end{bmatrix}.$$

As $\varepsilon \rightarrow 0^+$, δ_ε becomes closer and closer to 0, making the $(\rho + 1)$ -th column of R_ε^{-1} , i.e., the vector made up by the entries of $G_\varepsilon(:, 1)$ from $n + 1$ up to $2n$ after the multiplication of the matrix having $G_\varepsilon(n + 1 : 2n, 1)$ and $G_\varepsilon(n + 1 : 2n, 3)$ as columns, by H , closer and closer to a vector belonging to the subspace generated by \tilde{Q} .

Since a vector of the nullspace of T is defined up to multiplication by a constant, we replace the hyperbolic matrix H of the step $\rho + 1$, by the matrix

$$\tilde{H} = \begin{bmatrix} 1 + \frac{\delta_\varepsilon}{G_\varepsilon(n,3)} & -1 \\ -1 & 1 + \frac{\delta_\varepsilon}{G_\varepsilon(n,3)} \end{bmatrix},$$

and eventually, for $\varepsilon \rightarrow 0^+$, by

$$\tilde{H} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (3.13)$$

i.e., a very simple singular matrix. Thus, a multiple of a vector belonging to the nullspace of T and generating a U -chain is retrieved replacing the elements of the hyperbolic rotation in step 6) of Algorithm 1 (GSA) by the elements of the matrix in (3.13).

Let t be the generating vector of the U -chain. If the columns $\rho + 1, \rho + 2, \rho + j, \rho + j \leq n$, of T are also linearly dependent, it turns out that the entries of the second and the fourth columns of the generator matrix after the iteration ρ are zero [4]. Hence the size of the U -chain is $j + 1$ and can be computed in this way.

$$U(:, i) = Z^{i-1}t, \quad i = 1, \dots, j + 1.$$

We describe the corresponding modified GSA.

% ALGORITHM 2. Modified Generalized Schur algorithm.

```
% INPUT:      G, the generator matrix of the Toeplitz matrix T.
%             tol1, a fixed tolerance to check the singularity
%             tol2, a fixed tolerance to check the length of the U-chain
% OUTPUT:     t1, the generating vector of the first possible U-chain
%             dimK1, length of the first U-chain
%             t2, the generating vector of the second possible U-chain
%             dimK2, length of the second U-chain
```

```
function [R, t1, t2, dimK1, dimK2] = schur(G, tol1);
```

```
sing = 0; k = 1; ip(1) = 1; ip(2) = 3; dimK1 = 0; dimK2 = 0;
```

```
1) while k < n & sing < 2,
```

```
2)   if sing == 0,
```

```
3)     [cG, sG] = giv(G(k, 1), G(k, 2));
```

```
4)     G(k : n + k, 1 : 2) = G(k : n + k, 1 : 2) * [ cG  sG
                                                    -sG  cG ];
```

```
5)     [cG, sG] = giv(G(k, 3), G(k, 4));
```

```
6)     G(k : n + k - 1, 3 : 4) = G(k : n + k - 1, 3 : 4) * [ cG  sG
                                                                -sG  cG ];
```

```
7)   end % if
```

```
8)   if |(G(k, ip(1)))^2 - G(k, ip(2))^2| < tol1,
```

```
9)     [cH, sH] = hyp(G(k, ip(1)), G(k, ip(3)));
```

```

10)       $G(k : n + k, [ip(1), ip(2)]) = G(k : n + k, [ip(1), ip(2)]) \begin{bmatrix} c_H & -s_H \\ -s_H & c_H \end{bmatrix};$ 
11)      elseif  $sing == 0,$ 
12)           $t_1 = G(n + 1 : n + k, ind(1)) - G(n + 1 : n + k, ind(1));$ 
13)           $sing = sing + 1;$ 
14)           $ip(1) = ip(1) + 1; ip(2) = ip(2) + 1;$ 
15)          while  $|(G(k, ip(1))^2 - G(k, ip(2))^2| < tol_2, \& k < n,$ 
16)               $dimK_1 = dimK_1 + 1;$ 
17)               $k = k + 1;$ 
18)          end % while
19)      else
20)           $t_2 = G(n + 1 : n + k, ind(1)) - G(n + 1 : n + k, ind(1));$ 
21)           $dimK_2 = n - k;$ 
22)      end % if
23)       $G(:, 1) = \Phi G(:, 1);$ 
24)       $k = k + 1;$ 
25) end % while

```

Form a theoretical point of view

$$(G(k, ip(1))^2 - G(k, ip(2))^2) \geq 0, \quad (3.14)$$

for any $k \in \{1, \dots, n\}$, because of the positive semidefiniteness of the matrix $T^T T$. However, from a computational point of view, (3.14) could assume negative values, because of the roundoff errors in floating-point arithmetic. Therefore, we consider the absolute value of (3.14) in steps 5) and 8) of the algorithm.

The constants tol_1 and tol_2 are chosen equal to $\sqrt{n}\sqrt{\epsilon}$, with ϵ the machine precision.

The stability properties of GSA have been studied in [9, 10]. The proposed algorithm inherits the stability properties of GSA, turning out to be weakly stable.

4. Numerical Examples. In this section we apply the algorithm developed in the previous section to some examples.

EXAMPLE 4.1. *In this example, the entries of the vector b are the elements of the Fibonacci sequence*

$$\begin{aligned} b_1 &= 1, \\ b_2 &= 2, \\ b_i &= b_{i-1} + b_{i-2}, \quad i = 3, 4, \dots \end{aligned}$$

The corresponding Toeplitz matrix is $T \equiv \text{toeplitz}(b(n : -1 : 1), b(n : n + m - 1))$. Only the first two columns of T are linearly independent. The generating vector (see §2) $p \in \mathbb{R}^3$ is

$$p = [1, \quad -1, \quad 1]^T,$$

that are the coefficients of the polynomial

$$p(x) = x^2 - x - 1$$

whose roots are

$$\frac{1 + \sqrt{5}}{2}, \quad \frac{1 - \sqrt{5}}{2}.$$

Let us consider $m = 12, n = 9$. Let us denote by \tilde{p} the computed generating vector. Then

$$\max_i |p_i - \tilde{p}_i| = 2.104698637594993 \times 10^{-10}.$$

Moreover, let us denote by \tilde{Z} the computed nullspace of T generated by \tilde{p} . Then

$$\|T\tilde{Z}\|_2 = 8.039173492294422 \times 10^{-11}.$$

EXAMPLE 4.2. This example can be found in [4]. The Toeplitz matrix T is

$$T = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 & 2 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 1 & 2 & 2 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 2 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 \\ 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 \\ 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 \\ 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 \\ 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 \end{bmatrix}.$$

Columns 3, 4, and 5 are linearly dependent from the first two columns, while columns 6, 7, and 8 are again linearly independent. The generator vector p of the U -chain of the nullspace of T is

$$p = [1, \quad -2 \quad 1]^T.$$

Then

$$\max_i |p_i - \tilde{p}_i| = 8.304468224196171 \times 10^{-14}.$$

and

$$\|T\tilde{Z}\|_2 = 8.336584777351642 \times 10^{-14}.$$

5. Conclusions. A modification of the generalized Schur algorithm is considered to compute the structured nullspace of Hankel and Toeplitz matrices in a fast way. The algorithm is weakly stable, inheriting the stability properties of the generalized Schur algorithm.

The extension of the described algorithm will be studied to compute the nullspace of block-Hankel and block-Toeplitz Matrices.

REFERENCES

- [1] A. C. ANTOULAS, *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, USA, 2005.
- [2] A.W. BOJAŃCZYK, R.P. BRENT, P. VAN DOOREN, F.R. DE HOOG, *A note on downdating the Cholesky factorization*, SIAM J. Sci. Statist. Comput. 8 (1987), no. 3., 210–221.

- [3] S. CHANDRASEKARAN, A. SAYED, *Stabilizing the Generalized Schur Algorithm*, SIAM J. Matrix Anal. Appl. 17 (1996), no. 4, 950–983.
- [4] K. GALLIVAN, S. THIRUMALAI, P. VAN DOOREN AND V. VERMAUT, High performance algorithms for Toeplitz and block Toeplitz matrices, *Linear Algebra and its Applications* 241–243 (1996), pp. 343–388.
- [5] G. HEINIG AND K. ROST, *Algebraic Methods for Toeplitz-like Matrices and Operators*, Birkhäuser Verlag, Basel, Boston, Stuttgart, 1984.
- [6] T. KAILATH, *Linear Systems*, Prentice–Hall, Englewood Cliffs, N.J., 1980.
- [7] T. KAILATH AND A. H. SAYED, EDS., *Fast reliable algorithms for matrices with structure*, SIAM, Philadelphia, USA, 1999.
- [8] I. MARKOVSKY, J.C. WILLEMS, B. DE MOOR AND S. VAN HUFFEL, *Exact and Approximate Modeling of Linear Systems: A Behavioral Approach*, SIAM, Philadelphia, USA, 2006.
- [9] N. MASTRONARDI, P. VAN DOOREN, AND S. VAN HUFFEL, *On the stability of the generalized Schur algorithm*, Lecture notes in Computer Science, 1988, pp. 560–567, 2001.
- [10] M. STEWART AND P. VAN DOOREN, *Stability issues in the factorization of structured matrices*, SIAM J. Matrix Anal. Appl., 18, pp. 104–118, 1997.