axioms

MDPI

Article

The Generalized Schur Algorithm and Some Applications

Teresa Laudadio [1,*], Nicola Mastronardi [1] and Paul Van Dooren [2]

[1] Istituto per le Applicazioni del Calcolo "M. Picone", CNR, Sede di Bari, via G. Amendola 122/D, 70126 Bari, Italy; n.mastronardi@ba.iac.cnr.it

[2] Catholic University of Louvain, Department of Mathematical Engineering, Avenue Georges Lemaitre 4, B-1348 Louvain-la-Neuve, Belgium; paul.vandooren@uclouvain.be

* Correspondence: t.laudadio@ba.iac.cnr.it; Tel.: +39-080-5929752

Received: 2 October 2018; Accepted: 7 November 2018; Published: 9 November 2018

Abstract: The generalized Schur algorithm is a powerful tool allowing to compute classical decompositions of matrices, such as the QR and LU factorizations. When applied to matrices with particular structures, the generalized Schur algorithm computes these factorizations with a complexity of one order of magnitude less than that of classical algorithms based on Householder or elementary transformations. In this manuscript, we describe the main features of the generalized Schur algorithm. We show that it helps to prove some theoretical properties of the R factor of the QR factorization of some structured matrices, such as symmetric positive definite Toeplitz and Sylvester matrices, that can hardly be proven using classical linear algebra tools. Moreover, we propose a fast implementation of the generalized Schur algorithm for computing the rank of Sylvester matrices, arising in a number of applications. Finally, we propose a generalized Schur based algorithm for computing the null-space of polynomial matrices.

Keywords: generalized Schur algorithm; null-space; displacement rank; structured matrices

1. Introduction

The generalized Schur algorithm (GSA) allows computing well-known matrix decompositions, such as QR and LU factorizations [1]. In particular, if the involved matrix is structured, i.e., Toeplitz, block-Toeplitz or Sylvester, the GSA computes the R factor of the QR factorization with complexity of one order of magnitude less than that of the classical QR algorithm [2], since it relies only on the knowledge of the so-called *generators* [2] associated to the given matrix, rather than on the knowledge of the matrix itself. The stability properties of the GSA are described in [3–5], where it is proven that the algorithm is weakly stable provided the involved hyperbolic rotations are performed in a stable way.

In this manuscript, we first show that, besides the efficiency properties, the GSA provides new theoretical insights on the bounds of the entries of the R factor of the QR factorization of some structured matrices. In particular, if the involved matrix is a symmetric positive definite (SPD) Toeplitz or a Sylvester matrix, we prove that all or some of the diagonal entries of R monotonically decrease in absolute value.

We then propose a faster implementation of the algorithm described in [6] for computing the rank of a Sylvester matrix $S \in \mathbb{R}^{(m+n)\times(m+n)}$, whose entries are the coefficients of two polynomials of degree m and n, respectively. This new algorithm is based on the GSA for computing the R factor of the QR factorization of S. The proposed modification of the GSA-based method has a computational cost of $O(rl)$ floating point operations, where $l = \min\{n, m\}$ and r is the computed numerical rank.

It is well known that the upper triangular factor R factor of the QR factorization of a matrix $A \in \mathbb{R}^{n\times n}$ is equal to the upper triangular Cholesky factor $R_c \in \mathbb{R}^{n\times n}$ of A^TA, up to a diagonal sign

matrix D, i.e., $R = DR_c$, $D = \text{diag}(\pm 1, \cdots, \pm 1) \in \mathbb{R}^{n \times n}$. In this manuscript, we assume, without loss of generality, that the diagonal entries of R and R_c are positive and since the matrices are then equal, we denote both matrices by R.

Finally, we propose a GSA-based approach for computing a null-space basis of a polynomial matrix, which is an important problem in several systems and control applications [7,8]. For instance, the computation of the null-space of a polynomial matrix arises when solving the column reduction problem of a polynomial matrix [9,10].

The manuscript is structured as follows. The main features of the GSA are provided in Section 2. In Section 3, a GSA implementation for computing the Cholesky factor R of a SPD Toeplitz matrix is described, which allows proving that the diagonal entries of R monotonically decrease. In Section 4, a GSA-based algorithm for computing the rank of a Sylvester matrix S is introduced, based on the computation of the Cholesky factor R of $S^T S$. In addition, in this case, it is proven that the first diagonal entries of R monotonically decrease. The GSA-based method to compute the null-space of polynomial matrices is proposed in Section 5. The numerical examples are reported in Section 6 followed by the conclusions in Section 7.

2. The Generalized Schur Algorithm

Many of the classical factorizations of a symmetric matrix, e.g., QR and LDL^T, can be obtained by the GSA. If the matrix is Toeplitz-like, the GSA computes these factorizations in a fast way. For the sake of completeness, the basic concepts of the GSA for computing the R factor of the QR factorization of structured matrices, such as Toeplitz and block-Toeplitz matrices, are introduced in this Section. A comprehensive treatment of the topic can be found in [1,2].

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite (SPD) matrix. The semidefinite case is considered in Sections 4 and 5. The *displacement* of A with respect to a matrix Z of order n, is defined as

$$\nabla_Z A = A - ZAZ^T, \tag{1}$$

while the *displacement rank* k of A with respect to Z is defined as the rank of $\nabla_Z A$. If $\text{rank}(\nabla_Z A) = k$, Equation (1) can be written as the sum of k rank-one matrices,

$$\nabla_Z A = \sum_{i=1}^{k_1} \mathbf{g}_i^{(p)} \mathbf{g}_i^{(p)^T} - \sum_{i=1}^{k_2} \mathbf{g}_i^{(n)} \mathbf{g}_i^{(n)^T},$$

where $(k_1, n - k_1 - k_2, k_2)$ is the inertia of $\nabla_Z A$, $k = k_1 + k_2$, and the vectors $\mathbf{g}_i^{(p)} \in \mathbb{R}^n$, $i = 1, \ldots, k_1$, $\mathbf{g}_i^{(n)} \in \mathbb{R}^n$, $i = 1, \ldots, k_2$, are called the *positive* and the *negative generators* of A with respect to Z, respectively, conversely, if there is no ambiguity, simply the positive and negative generators of A. The matrix $G \equiv [\mathbf{g}_1^{(p)}, \mathbf{g}_2^{(p)}, \ldots, \mathbf{g}_{k_1}^{(p)}, \mathbf{g}_1^{(n)}, \mathbf{g}_2^{(n)}, \ldots, \mathbf{g}_{k_2}^{(n)}]^T$ is called the *generator* matrix.

The matrix Z is a nilpotent matrix. In particular, for Toeplitz and block-Toeplitz matrices, the matrix Z can be chosen as the shift and the block shift matrix

$$Z_1 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}, \quad Z_2 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ Z_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & Z_1 & 0 \end{bmatrix},$$

respectively.

The implementation of the GSA relies only on the knowledge of the generators of A rather than on the knowledge of the matrix itself [1].

Let

$$J = \text{diag}(\underbrace{1, 1, \ldots, 1}_{k_1}, \underbrace{-1, -1, \ldots, -1}_{k_2}).$$

Since

$$\begin{array}{lcl} A - ZAZ^T & = & G^T J G, \\ ZAZ^T - Z^2 AZ^{2T} & = & ZG^T J G Z^T, \\ \vdots & & \vdots \\ Z^{n-2} AZ^{n-2T} - Z^{n-1} AZ^{n-1T} & = & Z^{n-2} G^T J G Z^{n-2T}, \\ Z^{n-1} AZ^{n-1T} & = & Z^{n-1} G^T J G Z^{n-1T}, \end{array} \tag{2}$$

then, adding all members of the left and right-hand sides of Equation (2) yields

$$A = \sum_{j=0}^{n-1} Z^j G^T J G Z^{jT}, \tag{3}$$

which expresses the matrix A in terms of its generators.

Exploiting Equation (2), we show how the GSA computes R by describing its first iteration. Observe that the matrix products involved in the right-hand side of Equation (2) have their first row equal to zero, with the exception of the first product, $G^T J G$.

A key role in GSA is played by J-orthogonal matrices [11,12], i.e., matrices Φ satisfying $\Phi^T J \Phi = J$.

Any such matrix Φ can be constructed in different ways [11–14]. For instance, it can be considered as the product of Givens and hyperbolic rotations. In particular, a Givens rotation acting on rows i and j of the generator matrix is chosen if $J(i,i)J(j,j) > 0$, $i, j \in \{1, \ldots, n\}$, $i \neq j$. Otherwise, a hyperbolic rotation is considered. Indeed, suitable choices of Φ allow efficient implementations of GSA, as shown in Section 4.

Let $G_0 \equiv G$ and Φ_1 be a J-orthogonal matrix such that

$$\tilde{G}_1 = \Phi_1 G_0, \quad \tilde{G}_1 \mathbf{e}_1 = [\alpha_1, 0, \ldots, 0]^T, \quad \text{with} \quad \alpha_1 > 0, \tag{4}$$

and $\mathbf{e}_i$, $i = 1, \ldots, n$, be the ith column of the identity matrix. Furthermore, let $\tilde{\mathbf{g}}_1^T$ and $\tilde{\Gamma}_1$ be the first and last $k-1$ rows of $\tilde{G}_1$, respectively, i.e., $\tilde{G}_1 = \begin{bmatrix} \tilde{\mathbf{g}}_1^T \\ \tilde{\Gamma}_1 \end{bmatrix}$.

From Equation (4), it turns out that the first column of $\tilde{\Gamma}_1$ is zero. Let $\tilde{J}$ be the matrix obtained by deleting the first row and column from J. Then, Equation (2) can be written as follows,

$$\begin{array}{rcl} A & = & \sum_{j=0}^{n-1} Z^j G_0^T J G_0 Z^{jT} \\ & = & \sum_{j=0}^{n-1} Z^j G_0^T \Phi_1^T J \Phi_1 G_0 Z^{jT} \\ & = & \sum_{j=0}^{n-1} Z^j \begin{bmatrix} \tilde{\mathbf{g}}_1^T \\ \tilde{\Gamma}_1 \end{bmatrix}^T J \begin{bmatrix} \tilde{\mathbf{g}}_1^T \\ \tilde{\Gamma}_1 \end{bmatrix} Z^{jT} \\ & = & \tilde{g}_1 \tilde{g}_1^T + \sum_{j=1}^{n-1} Z^j \tilde{g}_1 \tilde{g}_1^T Z^{jT} + \sum_{j=0}^{n-2} Z^j \tilde{\Gamma}_1^T \tilde{J} \tilde{\Gamma}_1 Z^{jT} + \underbrace{Z^{n-1} \tilde{\Gamma}_1^T \tilde{J} \tilde{\Gamma}_1 Z^{n-1T}}_{=0} \\ & = & \tilde{\mathbf{g}}_1 \tilde{\mathbf{g}}_1^T + \sum_{j=0}^{n-2} Z^j \begin{bmatrix} \tilde{\mathbf{g}}_1^T Z^T \\ \tilde{\Gamma}_1 \end{bmatrix}^T J \begin{bmatrix} \tilde{\mathbf{g}}_1^T Z^T \\ \tilde{\Gamma}_1 \end{bmatrix} Z^{jT} \\ & = & \tilde{\mathbf{g}}_1 \tilde{\mathbf{g}}_1^T + \sum_{j=0}^{n-2} Z^j G_1^T J G_1 Z^{jT}, \\ & = & \tilde{\mathbf{g}}_1 \tilde{\mathbf{g}}_1^T + A_1, \end{array}$$

where $G_1 \equiv [Z\tilde{\mathbf{g}}_1, \tilde{\Gamma}_1^T]^T$, that is, G_1 is obtained from $\tilde{G}_1$ by multiplying $\tilde{g}_1$ with Z, and $A_1 \equiv \sum_{j=0}^{n-2} Z^j G_1^T J G_1 Z^{j^T}$. If A is a Toeplitz matrix, this multiplication with Z corresponds to displacing the entries of $\tilde{g}_1$ one position downward, while it corresponds to a block-displacement downward in the first generator if A is a block-Toeplitz matrix.

Thus, the first column of G_1 is zero and, hence, $\tilde{\mathbf{g}}_1^T$ is the first row of the R factor of the QR factorization of A. The above procedure is recursively applied to A_1 to compute the other rows of R.

The jth iteration of GSA, $j = 1, \ldots, n$, involves the products $\Phi_j G_{j-1}$ and $Z\tilde{\mathbf{g}}_1$. The former multiplication can be computed in $O(k(n-j))$ operations [11,12], and the latter is done for free if Z is either a shift or a block–shift matrix. Therefore, if the displacement rank k of A is small compared to n, the GSA computes the R factor in $O(kn^2)$ rather than in $O(n^3)$ operations, as required by standard algorithms [15].

For the sake of completeness, the described GSA implementation is reported in the following `matlab` style function. (The function `givens` is the `matlab` function having as input two scalars, x_1 and x_2, and as output an orthogonal 2×2 matrix Θ such that $\Theta \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{bmatrix}$. The function `Hrotate` computes the coefficients of the 2×2 hyperbolic rotation Φ such that, given two scalars x_1 and x_2, $|x_1| > |x_2|$, $\Phi \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_1^2 - x_2^2} \\ 0 \end{bmatrix}$. The function `Happly` applies Φ to two rows of the generator matrix. Both functions are defined in [12]).

```
function[R] =GSA(G, n);
for  i = 1 : n,
     for  j = 2 : k_1,
          Θ =givens(G(1, i), G(j, i));
          G([1, j], i : n) = Θ ∗ G([1, j], i : n);
     end % for
     for  j = k_1 + 2 : k_1 + k_2,
          Θ =  givens(G(k_1 + 1, i), G(j, i));
          G([k_1 + 1, j], i : n) = Θ ∗ G[k_1 + 1, j], i : n);
     end % for
     [c_1, s_1] =  Hrotate(G(1, i), G(k_1 + 1, i));
     G([1, k_1 + 1], i : n) = Happly(c_1, s_1, G([1, k_1 + 1], i : n), n − i + 1);
     R(i, i : n) = G(1, i : n);
     G(1, i + 1 : n) = G(1, i : n − 1);  G(1, i) = 0;
end % for
```

The GSA has been proven to be weakly stable [3,4], provided the hyperbolic transformations involved in the construction of the matrices Φ_j are performed in a stable way [3,11,12].

3. GSA for SPD Toeplitz Matrices

In this section, we describe the GSA for computing the R factor of the Cholesky factorization of a SPD Toeplitz matrix A, with R upper triangular, i.e., $A = R^T R$. Moreover, we show that the diagonal entries of R decrease monotonically.

Let $A \in \mathbb{R}^{n \times n}$ and $Z \in \mathbb{R}^{n \times n}$ be a SPD Toeplitz matrix and a shift matrix, respectively, i.e.,

$$A = \begin{bmatrix} t_1 & t_2 & \ddots & t_n \\ t_2 & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & t_2 \\ t_n & \ddots & t_2 & t_1 \end{bmatrix}, \quad Z_n = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix},$$

and let $\mathbf{t} = A(:,1)$. Then,

$$\nabla_Z A = \left[\begin{array}{c|ccc} t_1 & t_2 & \cdots & t_n \\ \hline t_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ t_n & 0 & \cdots & 0 \end{array}\right],$$

i.e., $\nabla_Z A$ is a symmetric rank-2 matrix. Moreover, the generator matrix G is given by

$$G = \left[\begin{array}{c} \mathbf{g}_1^T \\ \mathbf{g}_2^T \end{array}\right], \quad \text{with } \mathbf{g}_1 = \frac{\mathbf{t}}{\sqrt{t_1}}, \ \ \mathbf{g}_2 = [0, \mathbf{g}_1(2:n)^T]^T.$$

In this case, the GSA can be implemented in `matlab`-like style as follows.

```
function[R] = GSA_chol(G_0)
for  i = 1 : n,
     [c_1, s_1] =Hrotate(G_{i-1}(1,i), G^{(i)}(2,i));  G_{i-1}(:, i:n) = Happly(c_1, s_1, G_{i-1}(:, i:n), n-i+1);
     R(i, i:n) = G_{i-1}(1, i:n);
     G_i(1, i+1:n) = G_{i-1}(1, i:n-1);  G_i(2, i+1:n) = G_{i-1}(2, i+1:n-1);
end % for
```

The following lemma holds.

Lemma 1. *Let A be a SPD Toeplitz matrix and let R be its Cholesky factor, with R upper triangular. Then,*

$$R(i-1, i-1) \geq R(i,i), \quad i = 2, \ldots, n.$$

Proof. At each step i of `GSA_chol`, $i = 1, \ldots, n$, first a hyperbolic rotation is applied to G_{i-1} in order to annihilate the element $G_i(2,i)$. Hence, the first row of G_{i-1} becomes the row i of R. Finally, $G_i(1,:)$ is obtained displacing the entries of the first row of G_{i-1} one position right, while $G_i(2,:)$ is equal to $G_{i-1}(2,:)$. Taking into account that $G_{i-1}(2,1) = 0$, the diagonal entries of R are

$$\begin{aligned}
R(1,1) &= G_0(1,1) \\
R(2,2) &= \sqrt{G_1^2(1,2) - G_1^2(2,2)} = \sqrt{R^2(1,1) - G_1^2(2,2)} \leq R(1,1); \\
&\vdots \\
R(i,i) &= \sqrt{G_{i-1}^2(1,i) - G_{i-1}^2(2,i)} = \sqrt{R^2(i-1,i-1) - G_{i-1}^2(2,i)} \leq R(i-1,i-1); \\
&\vdots \\
R(n,n) &= \sqrt{G_{n-1}^2(1,n) - G_{n-1}^2(2,n)} = \sqrt{R^2(n-1,n-1) - G_{n-1}^2(2,n)} \leq R(n-1,n-1).
\end{aligned}$$

□

4. Computing the Rank of Sylvester Matrices

In this section, we focus on the computation of the rank of Sylvester matrices. The numerical rank of a Sylvester matrix is a useful information for determining the degree of the greatest common divisor of the involved polynomials [6,16,17].

A GSA-based algorithm for computing the rank of S has been recently proposed in [6]. It is based on the computation of the Cholesky factor R of $S^T S$, with R upper triangular, i.e., $R^T R = S^T S$.

Here, we propose a more efficient variant of this algorithm that allows proving that the first entries of R monotonically decrease.

Let $w_i \in \mathbb{R},\ i = 0, 1, \ldots, n$, and let $y_i \in \mathbb{R},\ i = 0, 1, \ldots, m$. Denote by $w(x)$ and $y(x)$ two univariate polynomials,

$$\begin{aligned} w(x) &= w_n x^n + w_{n-1} x^{n-1} + \cdots + w_1 x + w_0, \quad w_n \neq 0, \\ y(x) &= y_m x^m + y_{m-1} x^{m-1} + \cdots + y_1 x + y_0, \quad y_m \neq 0. \end{aligned} \tag{5}$$

Let $S \in \mathbb{R}^{(m+n)\times(m+n)}$ be the Sylvester matrix defined as follows,

$$S = \left[\begin{array}{c|c} W & Y \end{array}\right], \quad W = \begin{bmatrix} w_n & & & \\ w_{n-1} & w_n & & \\ \vdots & w_{n-1} & \ddots & \\ w_1 & \vdots & \ddots & w_n \\ w_0 & w_1 & \ddots & w_{n-1} \\ & w_0 & \ddots & \vdots \\ & & \ddots & w_1 \\ & & & w_0 \end{bmatrix}, \ Y = \begin{bmatrix} y_m & & & \\ y_{m-1} & y_m & & \\ \vdots & y_{m-1} & \ddots & \\ y_1 & \vdots & \ddots & y_m \\ y_0 & y_1 & \ddots & y_{m-1} \\ & y_0 & \ddots & \vdots \\ & & \ddots & y_1 \\ & & & y_0 \end{bmatrix}, \tag{6}$$

with $W \in \mathbb{R}^{(m+n)\times m}$ and $Y \in \mathbb{R}^{(m+n)\times n}$ band Toeplitz matrices.

We now describe how the GSA-based algorithm proposed in [6] for computing the rank of S can be implemented in a faster way. This variant is based on the computation of the Cholesky factor $R \in \mathbb{R}^{(m+n)\times(m+n)}$ of $S^T S$, with R upper triangular, i.e., $R^T R = S^T S$.

Defining

$$Z = \left[\begin{array}{c|c} Z_m & \\ \hline & Z_n \end{array}\right], \ \text{with} \ Z_k = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}_{k\times k}, \ k \in \mathbb{N}, \tag{7}$$

the generator matrix G of $S^T S$ with respect to Z is then given by [6]

$$G = \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \mathbf{g}_3 & \mathbf{g}_4 \end{bmatrix}^T$$

where

$$\begin{aligned} &\mathbf{g}_1 = \mathbf{x}_1 / \|S(:,1)\|_2, \\ &\mathbf{g}_2([2 : n+m]) = \mathbf{x}_2([2 : n+m]) / \|S(:, m+1)\|_2,\ \mathbf{g}_2(1) = 0, \\ &\mathbf{g}_3(2 : n+m) = \mathbf{g}_1(2 : n+m),\ \mathbf{g}_3(1) = 0, \\ &\mathbf{g}_4([1 : m, m+2 : n+m]) = \mathbf{g}_2([1 : m, m+2 : n+m]),\ \mathbf{g}_4(m+1) = 0, \end{aligned} \tag{8}$$

with $\mathbf{x}_1 = S^T S \mathbf{e}_1$, $\mathbf{x}_2 = S^T S \mathbf{e}_{m+1}$, $\mathbf{e}_j$ the jth vector of the canonical basis of $\mathbb{R}^{m+n}$, and $J = \mathrm{diag}(1, 1, -1, -1)$.

The algorithm proposed in [6] is based on the following GSA implementation for computing the R factor of the QR factorization of S.

```
function[R] = GSA_chol2(G)
for  i = 1 : n,
     Θ1 =givens(G(1,i), G(2,i));          Θ2 =givens(G(3,i), G(4,i));
     G(1 : 2, i : n) = Θ1 G(1 : 2, i : n);     G(3 : 4, i : n) = Θ2 G(3 : 4, i : n);
     [c1, s1] =Hrotate(G(1,i), G(3,i));
     G([1,3], i : n) = Happly(c1, s1, G([1,3], i : n), n − i + 1);
     R(i, i : n) = Gi(1, i : n);
     G(1, i + 1 : n) = G(1, i : n − 1)Z^T;     G(2, i + 1 : n) = G(2 : 4, i + 1 : n − 1);
end % for
```

At the ith iteration of the algorithm, $i = 1, \ldots, n$, the Givens rotations Θ_1 and Θ_2 are computed and applied, respectively, to the first and second generators, and to the third and fourth generators, to annihilate $G(2,i)$ and $G(4,i)$. Hence, the hyperbolic rotation $\begin{bmatrix} c_1 & -s_1 \\ -s_1 & c_1 \end{bmatrix}$ is applied to the first and the third row of G to annihilate $G(3,i)$. Finally, the first row of G becomes the ith row of R and the first row of G is multiplied by Z^T.

Summarizing, at the first step of the ith iteration of GSA, all entries of the ith column but the first one of G, are annihilated. If the number of rows of G is greater than 2, this can be accomplished in different ways (see [5,14]) .

Analyzing the pattern of the generators in Equation (8), we are able to derive a different implementation of GSA that costs $O(rl)$, with $l = \min\{n, m\}$. Moreover, this implementation allows proving that the first l diagonal entries of R are monotonically decreasing.

We observe that the matrix W^TW in Equation (6) is the SPD Toeplitz matrix

$$W^TW = \begin{bmatrix} t_1 & t_2 & \cdots & t_n & t_{n+1} & & \\ t_2 & t_1 & t_2 & \ddots & t_n & \ddots & \\ \vdots & t_2 & \ddots & \ddots & \ddots & \ddots & t_{n+1} \\ t_n & \vdots & \ddots & \ddots & \ddots & \vdots & t_n \\ t_{n+1} & t_n & \ddots & \ddots & \ddots & t_2 & \vdots \\ & \ddots & \ddots & \vdots & t_2 & t_1 & t_2 \\ & & t_{n+1} & t_n & \cdots & t_2 & t_1 \end{bmatrix}_{m \times m}, \tag{9}$$

with

$$t_i = \sum_{j=i}^{n+1} w_{j-1} w_{j-i}, \quad i = 1, 2, \ldots, n+1.$$

Since

$$S^TS = \left[\begin{array}{c|c} W^TW & W^TY \\ \hline Y^TW & Y^TY \end{array}\right],$$

if $n \ll m$, from Equation (9), it turns out that $G([1,3], n+2:m) = 0$. Moreover, the rows $G(2,:)$ and $G(4,:)$ have their first entry equal to zero and differ only in their entry in column $m+1$. This particular pattern of G is close to the ones described in [13,14,18], allowing to design an alternative GSA implementation with respect to that considered in [6], and thereby reducing the complexity from $O(r(n+m))$ to $O(rl)$, where r is the computed rank of S and $l = \min\{n, m\}$.

Since the description of the above GSA implementation is quite cumbersome and similar to the algorithms reported in [13,14,18], we omit it here. The corresponding `matlab` pseudo–code can be obtained from the authors upon request.

If the matrix S has rank $r < (n+m)$, at the $k = (n+m-r+1)$st iteration, it turns out that $G^2(1,k) - G^2(3,k) = 0$ in exact arithmetic [6]. Therefore, at each iteration of the algorithm we check whether

$$G^2(1,k) - G^2(3,k) > tol, \tag{10}$$

where tol is a fixed tolerance. If Equation (10) is not satisfied, we stop the computation considering k as the computed numerical rank of S.

The R factor of the QR factorization of S is unique if the diagonal entries of R are positive. The considered GSA implementation, yielding the rank of S and based on computing the R factor of the QR factorization of S, allows us to prove that the first l entries of the diagonal of R are ordered in a decreasing order, with $l = \min\{m, n\}$. In fact, the following theorem holds.

Theorem 1. *Let $R^TR = S^TS$ be the Cholesky factorization of S^TS with S the Sylvester matrix defined in Equation (6) with rank $r \geq l = \min\{m,n\}$. Then,*

$$R(i-1,i-1) \geq R(i,i) \geq 0, \quad i = 2,\ldots,l. \tag{11}$$

Proof. Each entry i of the diagonal of R is determined by the ith entry of the first row of G at the end of iteration i, for $i = 1,\ldots,m+n$. Let us define $\hat{G} \equiv G(:,1:l)$ and consider the following alternative implementation of the GSA for computing the first l columns of the Cholesky factor of S^TS.

```
for  i = 1 : l,
     Θ =givens(Ĝ(1,i), Ĝ(2,i));
     Ĝ(1:2, i:l) = Θ * Ĝ(1:2, i:l);
     [c1, s1] =Hrotate(Ĝ(1,i), Ĝ(4,i));  Ĝ([1,4],:) = Happly(c1, s1, Ĝ([1,4],:), l);
     [c2, s2] =Hrotate(Ĝ(1,i), Ĝ(3,i));  Ĝ([1,3],:) = Happly(c2, s2, Ĝ([1,3],:), l);
     R(i, i:l) = Ĝ(1, i:l);
     Ĝ(1, i+1:l) = Ĝ(1, i:l-1);
     Ĝ(1,i) = 0;
end % for
```

We observe that, for $i = 1$, $\hat{G}(1,1)$ is the only entry in the first column of $\hat{G}$ different from 0. Hence, $R(1,i) = \hat{G}(1,1:l)$ and the first iteration amounts only to shifting $\hat{G}(1,1:l)$ one position rightward, i.e., $\hat{G}(1,2:l) = \hat{G}(1,1:l-1)$, $\hat{G}(1,1) = 0$.

At the beginning of iteration $i = 2$, the second and the fourth row of $\hat{G}$ are equal Equation (8). Hence, when applying a Givens rotation to the first and the second row in order to annihilate the entry $\hat{G}(2,i)$ and when subsequently applying a hyperbolic rotation to the first and fourth row of $\hat{G}$ in order to annihilate $\hat{G}(4,i)$, it turns out that $\hat{G}(2,i:l)$ and $\hat{G}(4,i:l)$ are then modified but still equal to each other, while $\hat{G}(1,i:l)$ remains unchanged. The equality between $\hat{G}(2,:)$ and $\hat{G}(4,:)$ is maintained throughout the iterations $1,2,\ldots,l$.

Therefore, the second and the fourth row of $\hat{G}$ do not play any role in computing $R(1:l,1:l)$ and can be neglected. Hence, the GSA for computing $R(1:l,1:l)$ reduces only to applying a hyperbolic rotation to the first and the third generators, as described in the following algorithm.

```
for  i = 1 : l,
     [c2, s2] =Hrotate(Ĝ(1,i), Ĝ(3,i));  Ĝ([1,3],:) = Happly(c2, s2, Ĝ([1,3],:), l);
     R(i, i:l) = Ĝ(1, i:l);
     Ĝ(1, i+1:l) = Ĝ(1, i:l-1);
     Ĝ(1,i) = 0;
end % for
```

Since at the beginning of iteration i, $i = 2,\ldots,i$, $\hat{G}(1,i:l) = R(i-1,i-1:l-1)$, then the involved hyperbolic rotation $\Phi = \begin{bmatrix} c_2 & -s_2 \\ -s_2 & c_2 \end{bmatrix}$ is such that

$$\Phi \begin{bmatrix} \hat{G}(1,i) \\ \hat{G}(3,i) \end{bmatrix} = \Phi \begin{bmatrix} R(i-1,i-1) \\ \hat{G}(3,i) \end{bmatrix} = \begin{bmatrix} \hat{G}(1,i) \\ 0 \end{bmatrix} = \begin{bmatrix} R(i,i) \\ 0 \end{bmatrix},$$

where the updated $\hat{G}(1,i)$ is equal to $\sqrt{\hat{G}(1,i)^2 - \hat{G}(3,i)^2} \geq 0$. Therefore, $R(i,i) = \sqrt{R(i-1,i-1)^2 - \hat{G}(3,i)^2} \geq 0$, and thus $R(i,i) \leq R(i-1,i-1)$. □

Remark 1. *The above GSA implementation allows to prove the inequality Equation (11). This property is difficult to obtain if the QR factorization is performed via Householder transformations or if the classical Cholesky factorization of $S^T S$ is used.*

5. GSA for Computing the Null-Space of Polynomial Matrices

In this section, we consider the problem of computing a polynomial basis $X(s) \in \mathbb{R}^{n\times(n-\rho)}$ of the null-space of an $m \times n$ polynomial matrix of degree δ and rank $\rho \leq \min(m,n)$,

$$M(s) = \sum_{i=0}^{\delta} M_i s^i, \qquad M_i \in \mathbb{R}^{m\times n},\ i = 0,\ldots,\delta. \tag{12}$$

As described in [8,19,20], the above problem is equivalent to that of computing the null-space of a related block-Toeplitz matrix. Algorithms to solve this problem are proposed in [8,19] but they do not explicitly exploit the structure of the involved matrix. Algorithms to solve related problems have also been described in the literature, e.g., in [8,19,21,22].

In this paper, we propose an algorithm for computing the null-space of polynomial matrices based on a variant of the GSA for computing the null-space of a related band block-Toeplitz matrix [8].

5.1. Null-Space of Polynomial Matrices

A polynomial vector $\mathbf{v}(s) = \sum_{i=0}^{\gamma} \mathbf{v}_i s^i,\ \mathbf{v}_i \in \mathbb{R}^n,\ i = 0,\ldots,\gamma,\ \gamma \in \mathbb{N}$, is said to belong to the null-space of (12) if

$$M(s)\mathbf{v}(s) = 0 \Leftrightarrow \sum_{j=0}^{\delta} M_j s^j \sum_{i=0}^{\gamma} \mathbf{v}_i s^i = 0.$$

The polynomial vector $\mathbf{v}(s)$ belongs to the null-space of $M(s)$ iff $\mathbf{v} = [\mathbf{v}_0^T, \mathbf{v}_1^T, \ldots, \mathbf{v}_\gamma^T]^T$, $\mathbf{v}_i \in \mathbb{R}^n,\ i = 0,\ldots,\gamma$, is a vector belonging to the null-space of the band block-Toeplitz matrix

$$T = \begin{bmatrix} M_0 & & & & \\ M_1 & M_0 & & & \\ \vdots & M_1 & \ddots & & \\ M_\delta & \vdots & \ddots & M_0 & \\ & M_\delta & \ddots & M_1 & M_0 \\ & & \ddots & \vdots & M_1 \\ & & & M_\delta & \vdots \\ & & & & M_\delta \end{bmatrix}_{\hat{m}\times\hat{n}}, \tag{13}$$

where $\hat{m} = m(\delta + n_b),\ \hat{n} = nn_b$, with $n_b = \gamma + 1$ the number of block columns of T, that can be determined, e.g., by the algorithm described in [8]. Hence, the problem of computing the null-space of the polynomial matrix in Equation (12) is equivalent to the problem of computing the null-space of the matrix in Equation (13). To normalize the entries in this matrix, it is appropriate to first perform a QR factorization of each block column of T :

$$\begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_\delta \end{bmatrix} = \begin{bmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_\delta \end{bmatrix} U, \quad \text{where} \quad \sum_i Q_i^T Q_i = I_n,$$

and to absorb the upper triangular factor U in the vector $\mathbf{u}(s) := U\mathbf{v}(s)$. The convolution equation $M(s)\mathbf{v}(s) = 0$ then becomes an equation of the type $Q(s)\mathbf{u}(s) = 0$, but where the coefficient matrices Q_i of $Q(s)$ form together an orthonormalized matrix.

Remark 2. *Above, we have assumed that there are no constant vectors* $\mathbf{v}$ *in the kernel of* $M(s)$*. If there are, then, the block column of* M_i *matrices has rank less than n and the above factorization will discover it in the sense that the matrix U is nonsquare and the matrices* Q_i *have less columns than* M_i*,* $i = 0, 1, \ldots, \delta$*. This trivial null-space can be eliminated and we therefore assume that the rank was full. For simplicity, from now on, we also assume that the coefficient matrices of the polynomial matrix* $M(s)$ *were already normalized in this way and the norm of the block columns of T are thus orthonormalized. This normalization proves to be very useful in the sequel.*

Denote by

$$Z_{n_b} = \begin{bmatrix} 0_n & & & \\ I_n & 0_n & & \\ & \ddots & \ddots & \\ & & I_n & 0_n \end{bmatrix}_{\hat{n}\times\hat{n}} \quad \text{and} \quad Z = \begin{bmatrix} Z_{n_b} & \\ & Z_{n_b} \end{bmatrix}_{2\hat{n}\times 2\hat{n}},$$

where 0_n is the null–matrix of order $n \in \mathbb{N}$.

If $\mathbf{v}_i \neq 0$, $0 < i < \gamma$, and $\mathbf{v}_j = 0$, $j = i+1, \ldots, \gamma$, i.e.,

$$\mathbf{v} = [\mathbf{v}_0^T, \mathbf{v}_1^T, \ldots, \mathbf{v}_i^T, \underbrace{0, \ldots, 0}_{\gamma - i}]^T,$$

and $\mathbf{v} \in \ker(T)$, then also $Z_{n_b}^k \mathbf{v} \in \ker(T)$, $k = 0, 1, \ldots, \gamma - i$. In this case, the vector $\mathbf{v}$ is said to be a *generator vector* of a chain of length $\gamma - i + 1$ of the null-space of T.

The proposed algorithm for the computation of the null-space of polynomial matrices is based on the GSA for computing the R factor of the QR-factorization of the matrix T in Equation (13) and, if R is full column rank, its inverse R^{-1}.

Let us first assume that the matrix T is full rank, i.e., $\text{rank}(T) = \rho = \min\{\hat{m}, \hat{n}\}$. Without loss of generality, we suppose $\hat{m} \geq \hat{n}$. If $\hat{m} < \hat{n}$, the algorithm still computes the R factor in trapezoidal form [23]. Moreover, in this case, we compute the first $\hat{m}$ rows of the inverse of the matrix obtained appending the last $\hat{n} - \hat{m}$ rows of the identity matrix of order $\hat{n}$ to R.

Let us consider the SPD block-Toeplitz matrix

$$\hat{T} = T^T T = \begin{bmatrix} \hat{T}_0 & \hat{T}_1 & \cdots & \hat{T}_\delta & & \\ \hat{T}_1 & \hat{T}_0 & \hat{T}_1 & \ddots & \ddots & \\ \vdots & \hat{T}_1 & \ddots & \ddots & \ddots & \hat{T}_\delta \\ \hat{T}_\delta & \ddots & \ddots & \ddots & \ddots & \vdots \\ & \ddots & \ddots & \ddots & \hat{T}_0 & \hat{T}_1 \\ & & \hat{T}_\delta & \cdots & \hat{T}_1 & \hat{T}_0 \end{bmatrix} \in \mathbb{R}^{\hat{n}\times\hat{n}}, \tag{14}$$

whose blocks are

$$\hat{T}_{i-j} = \begin{cases} \sum_{k=0}^{\delta-|i-j|} M_{k+|i-j|}^T M_k, & \text{if } i \leq j \\ \sum_{k=0}^{\delta-|i-j|} M_k^T M_{k+|i-j|}, & \text{if } i > j. \end{cases} \tag{15}$$

Notice that, because of the normalization introduced before, we have that $\hat{T}_0 = I_n$ and $\|\hat{T}_i\|_2 \leq 1$. This is used below. The matrix

$$W = \left[\begin{array}{c|c} \hat{T} & I_{\hat{n}} \\ \hline I_{\hat{n}} & 0_{\hat{n}} \end{array}\right] \tag{16}$$

can be factorized in the following way,

$$W = \hat{R}^T \hat{J} \hat{R} \equiv \begin{bmatrix} R^T & \\ R^{-1} & R^{-1} \end{bmatrix} \begin{bmatrix} I_{\hat{n}} & \\ & -I_{\hat{n}} \end{bmatrix} \begin{bmatrix} R & R^{-T} \\ & R^{-T} \end{bmatrix}, \tag{17}$$

where $R \in \mathbb{R}^{\hat{n}\times\hat{n}}$ is the factor R of the QR-factorization of T, i.e., the Cholesky factor of $\hat{T}$. Hence, R and its inverse R^{-1} can be retrieved from the first $\hat{n}$ columns of the matrix $\hat{R}^T$.

The displacement matrix and the displacement rank of W with respect to Z, are given by

$$\nabla_Z(W) = W - ZWZ^T = \left[\begin{array}{cccccccc|cccc} I_n & \hat{T}_1 & \cdots & \hat{T}_\delta & 0_n & \cdots & 0_n & I_n & 0_n & \cdots & 0_n \\ \hat{T}_1 & & & & & & & & & & \\ \vdots & & & & & & & & & & \\ \hat{T}_\delta & & & & & & & & & & \\ 0_n & & & & & & & & & & \\ \vdots & & & & & & & & & & \\ 0_n & & & & & & & & & & \\ \hline I_n & & & & & & & & & & \\ 0_n & & & & & & & & & & \\ \vdots & & & & & & & & & & \\ 0_n & & & & & & & & & & \end{array}\right] \tag{18}$$

and $\rho(W, Z) = \operatorname{rank}(\nabla_Z(W))$, respectively, with $\nabla_Z(W) \in \mathbb{R}^{2\hat{n}\times 2\hat{n}}$.

Then, taking the order n of the matrices $\hat{T}_i, i = 0, 1, \ldots, \delta$, into account, it turns out that $\rho(W, Z) \leq 2n$.

Hence, Equation (18) can be written as the difference of two matrices of rank at most n, i.e.,

$$\nabla_Z(W) = {G^{(+)}}^T G^{(+)} - {G^{(-)}}^T G^{(-)} = G^T J G, \quad \text{where } G := \begin{bmatrix} G^{(+)} \\ G^{(-)} \end{bmatrix} \text{ and } J = \operatorname{diag}(I_n, -I_n).$$

Since $\hat{T}_0 = I_n$, the construction of G does not require any computation: it is easy to check that G is given by

$$G := \begin{bmatrix} G^{(+)} \\ G^{(-)} \end{bmatrix} = \left[\begin{array}{cccccccc|cccc} I_n & \hat{T}_1 & \cdots & \hat{T}_\delta & 0_n & \cdots & 0_n & I_n & 0_n & \cdots & 0_n \\ \hline 0_n & \hat{T}_1 & \cdots & \hat{T}_\delta & 0_n & \cdots & 0_n & 0_n & 0_n & \cdots & 0_n \end{array}\right]. \tag{19}$$

Remark 3. *Observe that increasing n_b, with $n_b \geq \delta + 1$, the structures of W and $\nabla_Z(W)$ do not change due to the block band structure of the matrix W. Consequently, the length of the corresponding generators changes but their structure remains the same since only $T_0, T_1, \ldots, T_\delta$ and I_n are different from zero in the first block row.*

The computation by the GSA of the R factor of T and of its inverse R^{-1} is made by only using the matrix G rather than the matrix T. Its implementation is a straightforward block matrix extension of the GSA described in Section 2.

Remark 4. *By construction, the initial generator matrix G_0 has the first $\delta+1$ block rows and the block row n_b+1 different from zero. Therefore, the multiplication of G_0 by the J-orthogonal matrix H_1 does not modify the structure of the generator matrix.*

Let $G_0 = G$. At each iteration i (for $i = 1, \ldots, n_b$,), we start from the generator matrix G_{i-1} having the blocks (of length n) $i, i+1, \ldots, i+\delta$ and $n_b+1, \ldots, n_b+i$ different form zero. We then look for a J-orthogonal matrix H_i such that the product $H_i G_{i-1}$ has in position $(1:n, (i-1)n+1:in)$ and $(n+1:2n, (i-1)n+1:in)$ a nonsingular upper triangular and zero matrix, respectively.

Then, G_i is obtained from $\begin{bmatrix} \tilde{G}_i^{(+)} \\ \tilde{G}_i^{(-)} \end{bmatrix} \equiv H_i G_{i-1}$ by multiplying the first n columns with Z, i.e.,

$$G_i = \begin{bmatrix} \tilde{G}_i^{(+)} Z^T \\ \tilde{G}_i^{(-)} \end{bmatrix}.$$

The computation of the J-orthogonal matrix H_i at the ith iteration of the GSA can be constructed as a product of n Householder matrices $\hat{H}_{i,j}$ and n hyperbolic rotations $\hat{Y}_{i,j}$, $j = 1, \ldots, n$.

The multiplication by the Householder matrices $\hat{H}_{i,j}$ modifies the last n columns of the generator matrix, annihilating the last n entries but the $(n+1)$st in the row $(i-1)n+j$, $j = 1, \ldots, n$, while the multiplication by the hyperbolic rotations $\hat{Y}_{i,j}$ acts on the columns i and $n+1$, annihilating the entry in position $((i-1)n+j, n+1)$.

Given $v_1, v_2 \in \mathbb{R}$, $|v_1| > |v_2|$, a hyperbolic matrix $Y \in \mathbb{R}^{2\times 2}$ can be computed

$$Y = \begin{bmatrix} c & -s \\ -s & c \end{bmatrix}, \quad \text{with } c = \frac{v_1}{\sqrt{v_1^2 - v_2^2}}, \ s = \frac{v_2}{\sqrt{v_1^2 - v_2^2}},$$

such that $[v_1, v_2]Y = [\sqrt{v_1^2 - v_2^2}, 0]$.

The modification of the sparsity pattern of the generator matrix after the first and ith iteration of the GSA are displayed in Figures 1 and 2, respectively.

The reliability of the GSA strongly depends on the way the hyperbolic rotation is computed. In [4,5,24], it is proven that the GSA is weakly stable if the hyperbolic rotations are implemented in an appropriate manner [3,11,12,24].

Let

$$H_{i,j} = \begin{bmatrix} I_n & \\ & \hat{H}_{i,j} \end{bmatrix} \qquad Y_{i,j} = \begin{bmatrix} I_{j-1} & & & & \\ & c_j & & -s_j & \\ & & I_{n-j} & & \\ & -s_j & & c_j & \\ & & & & I_{n-1} \end{bmatrix}.$$

Then,

$$H_i = H_{i,1} Y_{i,1} \cdots H_{i,n-1} Y_{i,n-1} H_{i,n} Y_{i,n}.$$

As previously mentioned, GSA relies only on the knowledge of the generators of W rather than on the matrix $\hat{T}$ itself. Its computation involves the product $\hat{T}^T \hat{T}$, which can be accomplished with $\delta^2 n^3$ flops. The ith iteration of the GSA involves the multiplication of n Householder matrices of size n times a matrix of size $((i+\delta+1)n \times n)$. Therefore, since the cost of the multiplication by the hyperbolic rotation is negligible with respect to that of the multiplication by the Householder matrices, the computational cost at iteration i is $4n^3(\delta+i)$. Hence, the computational cost of GSA is $2n^3 n_b(2\delta + n_b^2/2)$.

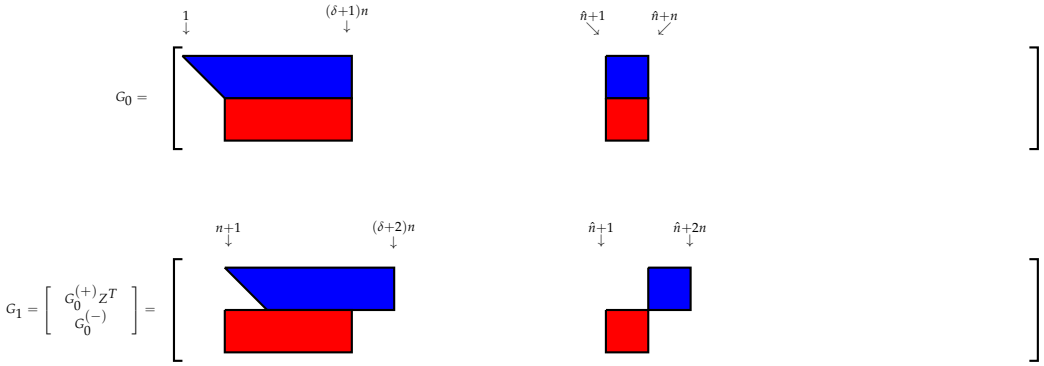

Figure 1. Modification of the sparsity pattern of the generator matrix G after the first iteration.

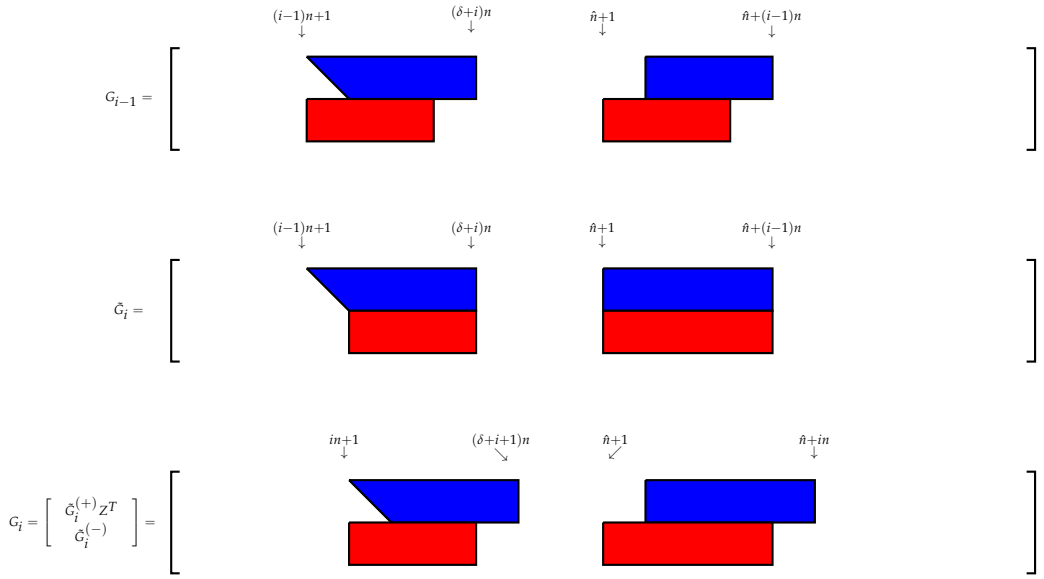

Figure 2. Modification of the sparsity pattern of the generator matrix G after the ith iteration.

5.2. GSA for Computing the Right Null-Space of Semidefinite Block Toeplitz Matrices

As already mentioned in Section 5.1, the number of desired blocks n_b of the matrix T in Equation (13) can be computed as described in [8]. For the sake of simplicity, in the considered examples, we choose n_b large enough to compute the null-space of T.

The structure and the computation via the GSA of the R factor of the QR factorization of the singular block Toeplitz matrix T with rank $\rho < n \leq m$, is considered in [23].

A modification of the GSA for computing the null-space of Toeplitz matrices is described in [25]. In this paper, we extend the latter results to compute the null-space of T by modifying GSA.

Without loss of generality, let us assume that the first $\hat{n}-1$ columns of T are linear independent and suppose that the $\hat{n}$th column linearly depends on the previous ones. Therefore, the first $\hat{n}-1$ principal minors of $\hat{T}$ are positive while the $\hat{n}$th one is zero. Let $\hat{T} = Q\Lambda Q^T$ be the spectral decomposition of $\hat{T}$, with $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_{\hat{n}}]$ orthogonal and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{\hat{n}-1}, \lambda_{\hat{n}})$, with

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{\hat{n}-1} > \lambda_{\hat{n}} = 0,$$

and let $\hat{T}_\varepsilon = Q\Lambda_\varepsilon Q^T$, with $\Lambda_\varepsilon = \mathrm{diag}(\lambda_1, \ldots, \lambda_{\hat{n}-1}, \varepsilon^2)$, with $\varepsilon \in \mathbb{R}_+^*$. Hence,

$$\hat{T}_\varepsilon^{-1} = \frac{1}{\varepsilon^2}\left(\sum_{i=1}^{\hat{n}-1} \frac{\varepsilon^2}{\lambda_i}\mathbf{q}_i\mathbf{q}_i^T + \mathbf{q}_{\hat{n}}\mathbf{q}_{\hat{n}}^T\right).$$

Let R_ε be the Cholesky factor of $\hat{T}_\varepsilon$, with R_ε upper triangular, i.e., $\hat{T}_\varepsilon = R_\varepsilon^T R_\varepsilon$. Then,

$$\varepsilon^2 \hat{T}_\varepsilon^{-1} e_{\hat{n}}^{(2\hat{n})} = \left(\sum_{i=1}^{\hat{n}-1} \frac{\varepsilon^2 \mathbf{q}_i^T e_{\hat{n}}^{(2\hat{n})}}{\lambda_i} \mathbf{q}_i + (\mathbf{q}_{\hat{n}}^T e_{\hat{n}}^{(2\hat{n})}) \mathbf{q}_{\hat{n}} \right).$$

On the other hand,

$$\varepsilon^2 \hat{T}_\varepsilon^{-1} e_{\hat{n}}^{(2\hat{n})} = \varepsilon^2 R_\varepsilon^{-1} R_\varepsilon^{-T} e_{\hat{n}}^{(2\hat{n})} = \varepsilon^2 r_{\hat{n},\hat{n}}^{-1} R_\varepsilon^{-1} e_{\hat{n}}^{(2\hat{n})},$$

where $r_{\hat{n},\hat{n}} = {e_{\hat{n}}^{(2\hat{n})}}^T R_\varepsilon e_{\hat{n}}^{(2\hat{n})}$. Hence, as $\varepsilon \to 0^+$, the last column of R_ε^{-1} becomes closer and closer to a multiple of $\mathbf{q}_{\hat{n}}$, the eigenvector corresponding to the 0 eigenvalue of $\hat{T}$.

Therefore, given

$$W_\varepsilon = \left[\begin{array}{c|c} \hat{T}_\varepsilon & I_{\hat{n}} \\ \hline I_{\hat{n}} & 0_{\hat{n}} \end{array} \right],$$

we have that

$$\nabla_Z(W_\varepsilon) = {G^{(+)}}^T G^{(+)} - {G^{(-)}}^T G^{(-)} + \varepsilon^2 e_{\hat{n}}^{(2\hat{n})} {e_{\hat{n}}^{(2\hat{n})}}^T.$$

Let

$$G_{0,\varepsilon} = \left[\begin{array}{c} G^{(+)} \\ G^{(-)} \\ \varepsilon {e_{\hat{n}}^{(2\hat{n})}}^T \end{array} \right].$$

Define

$$J_\varepsilon = \mathrm{diag}(\underbrace{1, 1, \ldots, 1}_{\hat{n}}, \underbrace{-1, -1, \ldots, -1}_{\hat{n}}, 1).$$

Hence,

$$\nabla_Z(W_\varepsilon) = G_{0,\varepsilon}^T J_\varepsilon G_{0,\varepsilon}.$$

We observe that column $\hat{n}+1$ of the generator matrix is not involved in the GSA until the very last iteration, since only its $\hat{n}$th entry is different from 0. At the very last iteration, the hyperbolic rotation

$$Y = \left[\begin{array}{cc} c & -s \\ -s & c \end{array} \right],$$

with

$$c = \frac{\sqrt{{G^{(+)}}^2(n,\hat{n}) + \varepsilon^2}}{\sqrt{{G^{(+)}}^2(n,\hat{n}) + \varepsilon^2 - {G^{(-)}}^2(1,\hat{n})}}, \quad s = \frac{G^{(-)}(1,\hat{n})}{\sqrt{{G^{(+)}}^2(n,\hat{n}) + \varepsilon^2 - {G^{(-)}}^2(1,\hat{n})}}$$

is applied to the $\hat{n}$th and $(\hat{n}+1)$st rows of G, i.e., to the nth row of $G^{(+)}$ and the first one of $G^{(-)}$. Since $\hat{T}$ is singular, it turns out that $|G^{(+)}(n,\hat{n})| = |G^{(-)}(1,\hat{n})|$ (see [23,25]). Thus,

$$\begin{aligned} Y &= \frac{1}{\sqrt{{G^{(+)}}^2(n,\hat{n}) + \varepsilon^2 - {G^{(-)}}^2(1,\hat{n})}} \left[\begin{array}{cc} \sqrt{{G^{(+)}}^2(n,\hat{n}) + \varepsilon^2} & -G^{(-)}(1,\hat{n}) \\ -G^{(-)}(1,\hat{n}) & \sqrt{{G^{(+)}}^2(n,\hat{n}) + \varepsilon^2} \end{array} \right] \\ &= \frac{|G^{(+)}(n,\hat{n})|}{\varepsilon} \left[\begin{array}{cc} \sqrt{1 + \left(\frac{\varepsilon}{G^{(+)}(n,\hat{n})}\right)^2} & -\theta \\ -\theta & \sqrt{1 + \left(\frac{\varepsilon}{G^{(+)}(n,\hat{n})}\right)^2} \end{array} \right], \end{aligned}$$

where

$$\theta = \frac{G^{(-)}(1,\hat{n})}{|G^{(+)}(n,\hat{n})|} = \text{sign}(G^{(-)}(1,\hat{n})).$$

We observe that, as $\varepsilon \to 0^+$,

$$\frac{|G^{(+)}(n,\hat{n})|}{\varepsilon} \to \infty, \quad \sqrt{1+\left(\frac{\varepsilon}{G^{(+)}(n,\hat{n})}\right)^2} \to 1.$$

Since a vector of the right null-space of T is determined except for the multiplication by a constant, neglecting the term $|G^{(+)}(n,\hat{n})|/\varepsilon$, such a vector can be computed at the last iteration as the first column of the product

$$\begin{bmatrix} 1 & -\theta \\ -\theta & 1 \end{bmatrix} \begin{bmatrix} G^{(+)}(n,\hat{n}+1:2\hat{n}) \\ G^{(-)}(1,\hat{n}+1:2\hat{n}) \end{bmatrix}.$$

When detecting a vector of the null-space as a linear combination of row n of $G^{(+)}$ and row one of $G^{(-)}$, the new generator matrix G for the GSA is obtained removing the latter columns from G [23,25].

The implementation of the modified GSA for computing the null-space of band block-Toeplitz matrices in Equation (13) is rather technical and can be obtained from the authors upon request.

The stability properties of the GSA have been studied in [4,5,24]. The proposed algorithm inherits the stability properties of the GSA, which means that it is weakly stable.

6. Numerical Examples

All the numerical experiments were carried out in `matlab` with machine precision $\varepsilon \approx 2.22 \times 10^{-16}$. Example 1 concerns the computation of the rank of a Sylvester matrix, while Examples 2 and 3 concern the computation of the null-space of polynomial matrices.

Example 1. *Let $x_i, i = 1,\ldots,12$, $y_i,\ i = 1,\ldots,15$, and $z_i,\ i = 1,\ldots,3$, be random numbers generated by the* `matlab` *function* `randn`*. Let $w(x)$and $y(x)$ be the two polynomials of degree* 15 *and* 18*, constructed by the matlab function* `poly`*, whose roots are, respectively, x_i and $z_j,\ i = 1,\ldots,12,\ j = 1,\ldots,3$, and y_i and $z_j,\ i = 1,\ldots,15,\ j = 1,\ldots,3$.*

The greatest common divisor of w and y has degree 3 *and, therefore, the Sylvester matrix $S \in \mathbb{R}^{33\times 33}$ constructed from $w(x)$ anf $y(x)$ has rank* 30*. The diagonal entries of the R factor computed by the GSA implementation described in Section 4 are displayed in Figure 3. Observe that the rank of the matrix can be retrieved by the number of entries of R above a certain tolerance. Moreover, it can be noticed that the first $m = 15$ diagonal entries monotonically decrease.*

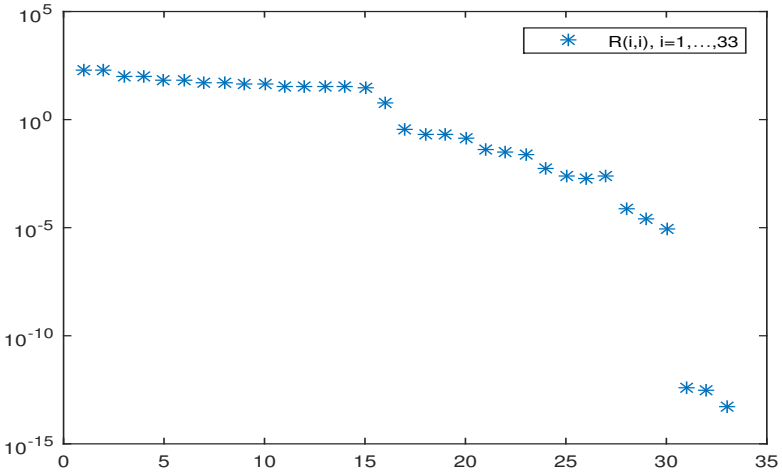

Figure 3. Diagonal entries of R.

Example 2. *As second example, we consider the computation of the coprime factorization of a transfer function matrix, described in [9,26]. The results obtained by the proposed GSA-based algorithm were compared with those obtained computing the null-space of the considered matrix by the function* `svd` *of* `matlab`.

Let $H(s) = N_r(s)D_r^{-1}(s)$ *be the transfer function with*

$$D_r(s) = \begin{bmatrix} 1-s & 0 & 0 & 0 \\ 0 & 1-s & 0 & 0 \\ 0 & -s & 1-s & 0 \\ 0 & 0 & 0 & 1-s \end{bmatrix}, \quad N_r(s) = \begin{bmatrix} s^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & s \end{bmatrix}.$$

Let

$$M(s) = \begin{bmatrix} N_r^T(s) & -D_r^T(s) \end{bmatrix} = \begin{bmatrix} s^2 & 0 & 0 & 0 & 0 & s-1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s-1 & s & 0 \\ 0 & 0 & 0 & s & 0 & 0 & 0 & s-1 & 0 \\ 0 & 0 & 0 & 0 & s & 0 & 0 & 0 & s-1 \end{bmatrix}.$$

As reported in [9,26], a minimal polynomial basis for the right null-space of $M(s)$ *is*

$$N(s) = \begin{bmatrix} 1-s & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & (1-s)^2 & 0 \\ 0 & 0 & 1-s \\ s^2 & 0 & 0 \\ 0 & s^2 & 0 \\ 0 & s-s^2 & 0 \\ 0 & 0 & s \end{bmatrix}.$$

Let us consider $n_b = 3$. *Then,* $T \in \mathbb{R}^{20\times 21}$ *is the block-Toeplitz matrix constructed from* $M(s)$ *as described in Section 5.1. Let* $\text{rank}(M) = 17$ *and* $U\Sigma V^T$ *be the rank and the singular value decomposition of* T *computed by* `matlab`, *respectively, and let us define* $V_1 = V(:, 1:17)$ *and* $V_2 = V(:, 18:21)$ *the matrices of the right singular vectors corresponding to the nonzero and zero singular values of* T, *respectively. The modified GSA applied to* T *yields four vectors* $\mathbf{v}_1, Z_{n_b}\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^{21}$ *belonging to the right null-space of* $M(s)$, *with* $Z_{n_b} = \text{diag}(\text{ones}(14,1), -7)$. *Let* $X = [\mathbf{v}_1\ Z_{n_b}\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3]$. *In Table 1, the relative norm of* TV_2, *the relative norm of* TX, *the norm of* $V_1^TV_2$ *and the norm of* V_1^TX, *are reported in Columns 1–4, respectively.*

Table 1. Relative norm of TV_2, relative norm of TX, norm of $V_1^TV_2$ and norm of V_1^TX, for Example 2.

$\frac{\|TV_2\|_2}{\|T\|_2}$	$\frac{\|TX\|_2}{\|T\|_2}$	$\|V_1^TV_2\|_2$	$\|V_1^TX\|_2$
4.23×10^{-16}	2.89×10^{-16}	9.66×10^{-16}	2.59×10^{-15}

Such values show that the results provided by `svd` *of* `matlab` *and by the algorithm based on a modification of GSA are comparable in terms of accuracy.*

Example 3. *This example can be found in [9,26]. Let* $H(s) = D_l^{-1}(s)N_r(s)$ *be the transfer function with*

$$D_l(s) = (s+2)^2(s+3)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad N_l(s) = \begin{bmatrix} 3s+8 & 2s^2+6s+2 \\ s^2+6s+2 & 3s^2+7s+8 \end{bmatrix}.$$

Let $M(s) = [D_l(s), -N_L(s)]$. *A right coprime pair for* $M(s)$ *is given by*

$$N_r = \begin{bmatrix} 3 & 2 \\ s+2 & 3 \end{bmatrix}, \quad D_r = \begin{bmatrix} s^2+3s+4 & 2 \\ 2 & s+4 \end{bmatrix},$$

Let us choose $n_b = 4$. *Then,* $T \in \mathbb{R}^{14\times 16}$ *is the block-Toeplitz matrix constructed from* $M(s)$ *as described in Section 5.1. Let* $\text{rank}(M) = 11$ *and* $U\Sigma V^T$ *be the rank and the singular value decomposition of* T *computed by* `matlab`, *respectively, and let define* $V_1 = V(:, 1:11)$ *and* $V_2 = V(:, 12:16)$ *the matrices of the right singular vectors corresponding to the nonzero and zero singular values of* T, *respectively. The modified GSA applied to* T *yields the vectors* $\mathbf{v}_1, Z_{n_b}\mathbf{v}_1, Z_{n_b}^2\mathbf{v}_1, \mathbf{v}_2, Z_{n_b}\mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^{16}$ *of the right null-space, with* $Z_{n_b} = \text{diag}(\text{ones}(12,1), -4)$. *Let* $X = [\mathbf{v}_1,\ Z_{n_b}\mathbf{v}_1,\ Z_{n_b}^2\mathbf{v}_1,\ \mathbf{v}_2,\ Z_{n_b}\mathbf{v}_2,\ \mathbf{v}_3]$. *In Table 2 , the relative norm of* TV_2, *the relative norm of* TX, *the norm of* $V_1^TV_2$ *and the norm of* V_1^TX, *are reported in Columns 1–4, respectively.*

Table 2. Relative norm of TV_2, relative norm of TX, norm of $V_1^TV_2$ and norm of V_1^TX, for Example 3.

$\frac{\|TV_2\|_2}{\|T\|_2}$	$\frac{\|TX\|_2}{\|T\|_2}$	$\|V_1^TV_2\|_2$	$\|V_1^TX\|_2$
1.33×10^{-16}	2.04×10^{-16}	5.56×10^{-16}	4.91×10^{-15}

As in Example 2, the results yielded by the considered algorithms are comparable in accuracy.

7. Conclusions

The Generalized Schur Algorithm is a powerful tool allowing to compute classical decompositions of matrices, such as the QR and LU factorizations. If the involved matrices have a particular structure, such as Toeplitz or Sylvester, the GSA computes the latter factorizations with a complexity of one order of magnitude less than that of classical algorithms based on Householder or elementary transformations.

After having emphasized the main features of the GSA, we have shown in this manuscript that the GSA helps to prove some theoretical properties of the R factor of the QR factorization of some structured matrices. Moreover, a fast implementation of the GSA for computing the rank of Sylvester matrices and the null-space of polynomial matrices is proposed, which relies on a modification of the GSA for computing the R factor and its inverse of the QR factorization of band block-Toeplitz matrices with full column rank. The numerical examples show that the proposed approach yields reliable results comparable to those ones provided by the function `svd` of `matlab`.

Author Contributions: All authors contributed equally to this work.

Funding: This research was partly funded by INdAM-GNCS and by CNR under the Short Term Mobility Program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kailath, T.; Sayed, A.H. *Fast Reliable Algorithms for Matrices with Structure*; SIAM: Philadelphia, PA, USA, 1999.
2. Kailath, T.; Sayed, A. Displacement Structure: Theory and Applications. *SIAM Rev.* **1995**, *32*, 297–386. [CrossRef]
3. Chandrasekaran, S.; Sayed, A. Stabilizing the Generalized Schur Algorithm. *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 950–983. [CrossRef]
4. Stewart, M.; Van Dooren, P. Stability issues in the factorization of structured matrices. *SIAM J. Matrix Anal. Appl.* **1997**, *18*, 104–118. [CrossRef]
5. Mastronardi, N.; Van Dooren, P.; Van Huffel, S. On the stability of the generalized Schur algorithm. *Lect. Notes Comput. Sci.* **2001**, *1988*, 560–567.

6. Li, B.; Liu, Z.; Zhi, L. A structured rank-revealing method for Sylvester matrix. *J. Comput. Appl. Math.* **2008**, *213*, 212–223. [CrossRef]
7. Forney, G. Minimal bases of rational vector spaces with applications to multivariable linear systems. *SIAM J. Control Optim.* **1975**, *13*, 493–520. [CrossRef]
8. Zúñiga Anaya, J.; Henrion, D. An improved Toeplitz algorithm for polynomial matrix null-space computation. *Appl. Math. Comput.* **2009**, *207*, 256–272. [CrossRef]
9. Beelen, T.; van der Hurk, G.; Praagman, C. A new method for computing a column reduced polynomial matrix. *Syst. Control Lett.* **1988**, *10*, 217–224. [CrossRef]
10. Neven, W.; Praagman, C. Column reduction of polynomial matrices. *Linear Algebra Its Appl.* **1993**, *188*, 569–589. [CrossRef]
11. Bojańczyk, A.; Brent, R.; Van Dooren, P.; de Hoog, F. A note on downdating the Cholesky factorization. *SIAM J. Sci. Stat. Comput.* **1987**, *8*, 210–221. [CrossRef]
12. Higham, N.J. J-orthogonal matrices: properties and generation. *SIAM Rev.* **2003**, *45*, 504–519. [CrossRef]
13. Lemmerling, P.; Mastronardi, N.; Van Huffel, S. Fast algorithm for solving the Hankel/Toeplitz Structured Total Least Squares Problem. *Numer. Algorithm.* **2000**, *23*, 371–392. [CrossRef]
14. Mastronardi, N.; Lemmerling, P.; Van Huffel, S. Fast structured total least squares algorithm for solving the basic deconvolution problem. *SIAM J. Matrix Anal. Appl.* **2000**, *22*, 533–553.[CrossRef]
15. Golub, G.H.; Van Loan, C.F. *Matrix Computations*, 4th ed.; Johns Hopkins University Press: Baltimore, MD, USA, 2013.
16. Boito, P. *Structured Matrix Based Methods for Approximate Polynomial GCD*; Edizioni Della Normale: Pisa, Italy, 2011.
17. Boito, P.; Bini, D.A. A Fast Algorithm for Approximate Polynomial GCD Based on Structured Matrix Computations. In *Numerical Methods for Structured Matrices and Applications*; Bini, D., Mehrmann, V., Olshevsky, V., Tyrtyshnikov, E., Van Barel, M., Eds.; Birkhauser: Basel, Switzerland, 2010; Volume 199, pp. 155–173.
18. Mastronardi, N.; Lernmerling, P.; Kalsi, A.; O'Leary, D.; Van Huffel, S. Implementation of the regularized structured total least squares algorithms for blind image deblurring. *Linear Algebra Its Appl.* **2004**, *391*, 203–221. [CrossRef]
19. Basilio, J.; Moreira, M. A robust solution of the generalized polynomial Bezout identity. *Linear Algebra Its Appl.* **2004**, *385*, 287–303. [CrossRef]
20. Kailath, T. *Linear Systems*; Prentice Hall: Englewood Cliffs, NJ, USA, 1980.
21. Bueno, M.; De Terán, F.; Dopico, F. Recovery of Eigenvectors and Minimal Bases of Matrix Polynomials from Generalized Fiedler Linearizations. *SIAM J. Matrix Anal. Appl.* **2011**, *32*, 463–483. [CrossRef]
22. De Terán, F.; Dopico, F.; Mackey, D. Fiedler companion linearizations and the recovery of minimal indices. *SIAM J. Matrix Anal. Appl.* **2010**, *31*, 2181–2204. [CrossRef]
23. Gallivan, K.; Thirumalai, S.; Van Dooren, P.; Vermaut, V. High performance algorithms for Toeplitz and block Toeplitz matrices. *Linear Algebra Its Appl.* **1996**, *241–243*, 343–388. [CrossRef]
24. Stewart, M. Cholesky Factorization of Semi-definite Toeplitz Matrices. *Linear Algebra Its Appl.* **1997**, *254*, 497–526. [CrossRef]
25. Mastronardi, N.; Van Barel, M.; Vandebril, R. On the computation of the null space of Toeplitz-like matrices. *Electron. Trans. Numer. Anal.* **2009**, *33*, 151–162.
26. Antoniou, E.; Vardulakis, A.; Vologiannidis, S. Numerical computation of minimal polynomial bases: A generalized resultant approach. *Linear Algebra Its Appl.* **2005**, *405*, 264–278. [CrossRef]