An Improved Toeplitz Algorithm for Polynomial Matrix Null-Space Computation

J. C. Zúñiga Anaya[1,*] and D. Henrion[2,†]

1. Department of Mathematics, University of Guadalajara,
Av. Revolución 1500, 44430 Guadalajara, Jalisco, Mexico.

2. LAAS-CNRS,
7 Avenue du Colonel Roche, 31077 Toulouse, France.

January 22, 2009

Abstract

In this paper we present an improved algorithm to compute the minimal null-space basis of polynomial matrices, a problem which has many applications in control and systems theory. This algorithm takes advantage of the block Toeplitz structure of the Sylvester matrix associated with the polynomial matrix. The analysis of algorithmic complexity and numerical stability shows that the algorithm is reliable and can be considered as an efficient alternative to the well-known pencil (state-space) algorithms found in the literature.

Keywords
Polynomial matrices, numerical linear algebra, computer-aided control system design.

1 Introduction

1.1 The problem and its applications

In this paper we consider the problem of finding a polynomial basis for the *null-space* of an arbitrary $m \times n$ polynomial matrix

$$A(s) = A_0 + A_1 s + A_2 s^2 + \cdots + A_d s^d. \tag{1}$$

of degree d and rank $\rho \leq \min(m, n)$.

The right null-space of $A(s)$ is the set of non-zero rational vector $z(s)$ such that

$$A(s)z(s) = 0 \tag{2}$$

for all s. The right null-space is clearly a sub space over the field of rational functions. A basis of the right null-space of $A(s)$ is then formed by any set of $n - \rho$ linearly independent vectors (in the field of rational functions) satisfying (2). Let us denote by $Z(s)$ a full-rank matrix having these vectors as columns, so that $A(s)Z(s) = 0$. Basis $Z(s)$ is in general a rational matrix (a matrix whose entries are rational functions).

*juan.zuniga@red.cucei.udg.mx
†henrion@laas.fr

Notice that any rational basis $Z(s)$ could be transformed into a strictly polynomial basis via the multiplication of $Z(s)$ by a common multiple of all its denominators. In this paper we present an algorithm to compute such a polynomial basis, moreover, our algorithm also ensures minimality of this basis. Let δ_i for $i = 1, 2, \ldots, n-\rho$ be the degree of each vector in a polynomial basis. If the sum of all the degrees δ_i is minimal over the choice of all polynomial basis, then we have a *minimal basis* in the sense of Forney [1].

Analogously, a basis of the left null-space of $A(s)$ is formed by any set of $m-\rho$ linearly independent non-zero vectors satisfying $A^T(s)z^T(s) = 0$. Because of this duality, in the remainder of the paper we only consider the right null-space, that we simply call the null-space of $A(s)$.

Computing the null-space basis of a polynomial matrix has several applications in different areas of applied mathematics, in particular control and systems theory. Here we mention only a few examples, for more details see for instance [1]. As a first example, the pole-zero structure of a linear system[1] represented by the state space matrices (A, B, C, D) can be recovered from the eigenstructure of some polynomial matrices [2]. This structural information is important in problems such as decoupling [3]. In particular, the degrees of the vectors in a minimal basis of the null-space of the pencil

$$P(s) = \left[\begin{array}{cc} sI - A & B \\ C & D \end{array}\right],$$

which are defined as the Kronecker invariant indices [4], correspond to invariant lists defined in [5]. These invariants are key information when solving problems of structural modification for linear systems [6].

Computing null-space basis is also important when solving the problem of column reduction of a polynomial matrix [7]. Column reduction is the initial step in several elaborated control algorithms. Column reducedness of polynomial matrices is a property often required in computer-aided control system design.

With the polynomial equation approach of control theory, introduced by Kučera [8], the solution of several control problems has been reformulated in terms of polynomial matrix equations or Diophantine equations [9]. Such formulations are also relevant in the behavioral approach of J. C. Willems and co-workers [10]. Consider, for instance, the typical polynomial matrix equation $A(s)X(s) = B(s)$. By analogy with the constant case, the null-space of $[A(s)\ B(s)]$ contains key information about the existence and the uniqueness of solution $X(s)$. Many other problems of the polynomial and behavioral approaches to systems control boil down to computing the null-space basis of a suitable polynomial matrix. Consider for example a linear multivariable system represented by a left coprime matrix fraction description $T(s) = D_L^{-1}(s)N_L(s)$. It can be shown [2, 11] that a right coprime factorization $T(s) = N_R(s)D_R^{-1}(s)$ can be obtained via the computation of the null-space

$$[D_L(s)\ \ -N_L(s)] \left[\begin{array}{c} N_R(s) \\ D_R(s) \end{array}\right] = 0.$$

In fault diagnostics the residual generator problem can be transformed into the problem of finding the null-space basis of a polynomial matrix [12]. The problem is formulated as follows: consider the perturbed system $y(s) = G(s)u(s) + H(s)d(s) + L(s)f(s)$ where $d(s)$ is the Laplace transform of the unknown disturbances and $f(s)$ is the Laplace transform of the monitored faults. It is shown that the residual generator is given by $Q(s) = p^{-1}(s)Q(s)$, where $p(s)$ is a polynomial of suitable degree and $Q(s)$ generates the left null-space of matrix

$$M(s) = \left[\begin{array}{cc} G(s) & H(s) \\ I & 0 \end{array}\right].$$

In [13] an algorithm for J-spectral factorization of a polynomial matrix is presented. When the analyzed polynomial matrix is singular, it is necessary to extract a minimal basis of its null-space as an intermediate step of the factorization algorithm. The J-spectral factorization appears in the solution of robust, H_2 and H_∞ optimization control problems [14].

[1]The set of all the poles and zeros with their multiplicities.

1.2 Brief review of existing algorithms

Computational algorithms for the eigenstructure of polynomial matrices appear early in the 1970s. Most of the algorithms take as a general approach the *linearization* of the analyzed polynomial matrix. In [4] it is shown that the structural indices of a pencil, i.e. the multiplicities of the finite and infinite zeros, and the degrees of the vectors in a minimal null-space basis, are contained in its Kronecker canonical form, and reliable algorithms to compute this canonical form are developed. In [15] it is proved that the structural indices of an arbitrary polynomial matrix can be recovered from the Kronecker canonical form of a related pencil, a companion matrix associated to $A(s)$. So, reliable algorithms to obtain the eigenstructure of $A(s)$ were presented. Seminal works [4, 15] are currently the basis of many algorithms for polynomial matrix analysis. However, a drawback of this approach called in the following the *pencil approach*, is that it only returns the structural indices. Moreover, in several applications the explicit computation of the associated polynomial eigenvectors is also necessary.

Pencil algorithms that also compute the vectors in a polynomial null-space basis of $A(s)$ are based on the results in [11]. First the associated pencil is transformed into the generalized Schur form

$$U(sB-C)V = \begin{bmatrix} sB_z - C_z & * \\ 0 & sB_* - C_* \end{bmatrix} \tag{3}$$

by the methods explained in [4]. Here $sB_z - C_z$ is in upper staircase form, containing only the infinite and null-space structural indices of $A(s)$. Then, to obtain also the vectors in a minimal basis, a post-processing of $sB_z - C_z$ is needed: a minimal basis $Z_z(s)$ of $sB_z - C_z$ is computed with the recursive process presented in Section 4 of [11], then, it is shown that the n bottom rows of $V[Z_z^T(s)\ 0]^T$ form the searched basis. Unfortunately the process presented in [11] is usually computationally expensive and its numerical stability is not guaranteed.

Another pencil algorithm was presented in [16]. This algorithm is based on descriptor system techniques and computes rational null-space basis. First, a minimal descriptor realization of $A(s)$, a rational matrix, in general, is obtained. Then, using the methods in [4], the corresponding pencil is reduced to a staircase form, and finally some rows are picked up to form the searched basis. Reliable algorithms to compute the descriptor realization are presented in [17]. Nevertheless, no study on the backward error in the coefficients of $A(s)$ is presented.

On the other hand, the classical numerical methods to manipulate polynomial matrices are based on the use of elementary operations over the ring of polynomials. The poor performance of these basic methods was quickly identified. The major criticism concerns numerical stability, lost because of pivoting w.r.t. power monomials of the indeterminate s [18]. However, we can also find in the literature some reliable polynomial methods which do not use the linearization of the analyzed matrix, see [19, 20] for instance. These methods process directly coefficients A_i of $A(s)$ and avoid elementary operations over polynomials.

In [21], an algorithm to obtain the structural indices of $A(s)$ at any given value $s = \alpha$ is presented. This algorithm is based on the rank computation of different Toeplitz matrices obtained from the Laurent expansion of $A(s)$ at α. Similarly, in [22] it is presented an algorithm to obtain the finite and infinite structural indices of rational matrices, also based on rank computation of successive Toeplitz matrices.

More recently, another algorithm was presented in [23]. The basic idea of this algorithm is also the computation of the rank and null-space of some associated block Toeplitz matrices. The singular value decomposition (SVD) is used as the rank revealing method. Now, unlike with the pencil algorithms, one obtains the structural indices and the vectors in the minimal basis with the same computational method. Moreover, the algorithm in [23] does not perform elementary polynomial operations, and the use of standard methods like the SVD improves numerical properties with respect to the classical polynomial methods. A similar work is presented in [24] using matrix resultants. We remark that both papers [23] and [24] appeared after submission of our conference papers [25, 26], on which the present paper is based. Moreover, as shown in the remainder of this paper, our algorithm performs faster than the algorithms in [23, 24] without jeopardizing accuracy and numerical stability.

1.3 Contribution

In this paper we contribute with an improved algorithm that takes advantage of the Toeplitz structure of the analysed matrices at each step, and uses the LQ factorization as the rank revealing method. Our algorithm allows to determine in an unequivocal way the vectors which form the minimal basis directly from each computed orthogonal matrix. This avoids the possibility of choosing two linearly dependent vectors to form the basis, but in a simpler and more direct way than the method proposed in the 3rd step of Algorithm 1 in [23].

Moreover, in this paper we present a full analysis in two aspects: numerical stability and algorithmic complexity. We show how the backward error over the coefficients of the analysed polynomial matrix is bounded, and we give some upper bounds. We also determine simple expressions of the complexity order of our algorithm. A detailed complexity and stability analysis is not carried out for the algorithms proposed in [23] and [24].

2 Preliminaries

The following notations are used in this paper. Real ε is the machine precision in the floating point arithmetic system. $O(\varepsilon)$ is a constant depending on ε, of the same order of magnitude. $\|A\|_2$ denotes the 2-norm or spectral norm of matrix A, it is given by $\|A\|_2 = \sigma_{\max}(A)$, the greatest singular value of A [27]. A^T is the transpose of A, and I_n denotes the identity matrix of dimension n. We also use some Matlab-like notations, i.e. $A(p:q,:)$ extracts rows p to q from matrix A. For example, $A(1:3,1:4)$ represents the sub-matrix containing the first 3 rows and the first 4 columns of A, and $A(:,[1\ 3\ 6\ 9\ ])$ is a matrix containing the first, 3rd, 6th and 9th columns of A.

In this section we present some definitions and results that we use in the remainder of the paper. We start with the infinite and finite zero structures of $A(s)$. Together with the null-space structure, they form the *eigenstructure* of $A(s)$. Finite and infinite structures are usually defined via the Smith and Smith MacMillan at infinity canonical forms, see for instance [2]. However, we give here an equivalent and more algebraic definition which will be useful later. A finite zero of a polynomial matrix $A(s)$ with rank ρ is a complex number α such that rank $A(\alpha) < \rho$. So, there exists a non-zero complex vector v, apart from those forming the basis of the null-space of $A(s)$, satisfying $A(\alpha)v = 0$. Vector v is called characteristic vector or eigenvector associated to the finite zero α.

From Theorem A.1 in [28], if α is a finite zero of $A(s)$ with algebraic multiplicity m_a and geometric multiplicity m_g, then there exists a series of integers $k_i > 0$ for $i = 1,2,\ldots,m_g$ such that $m_a = k_1 + k_2 + \cdots + k_{m_g}$, and a series of characteristic vectors $v_{i1}, v_{i2}, \ldots, v_{ik_i}$ associated to α such that

$$\begin{bmatrix} \bar{A}_0 & & \\ \vdots & \ddots & \\ \bar{A}_{k_i-1} & \cdots & \bar{A}_0 \end{bmatrix} \begin{bmatrix} v_{i1} \\ \vdots \\ v_{ik_i} \end{bmatrix} = 0 \tag{4}$$

with $v_{11}, v_{21}, \ldots, v_{m_g1}$ linearly independent and where

$$\bar{A}_j = \frac{1}{j!}\left[\frac{\mathrm{d}^j A(s)}{\mathrm{d}s^j}\right]_{s=\alpha}.$$

Integer k_i is the length of the ith chain of eigenvectors associated to α.

The infinite structure or structure at infinity of $A(s)$ is equivalent to the finite structure at $\alpha = 0$ of the dual matrix $\bar{A}(s) = A_d + A_{d-1}s + \cdots + A_0 s^d$ [2]. So, if $\alpha = 0$ has algebraic multiplicity $\bar{m}_a$ and geometric multiplicity $\bar{m}_g$ in the dual matrix $\bar{A}(s)$, then there exists a series of integers $\bar{k}_i > 0$ for $i = 1,2,\ldots,\bar{m}_g$ such that $\bar{m}_a = \bar{k}_1 + \bar{k}_2 + \cdots + \bar{k}_{\bar{m}_g}$, and a series of eigenvectors at infinity $\bar{v}_{i1}, \bar{v}_{i2}, \ldots, \bar{v}_{i\bar{k}_i}$ such that

$$\begin{bmatrix} A_d & & \\ \vdots & \ddots & \\ A_{d-k_i+1} & \cdots & A_d \end{bmatrix} \begin{bmatrix} \bar{v}_{i1} \\ \vdots \\ \bar{v}_{i\bar{k}_i} \end{bmatrix} = \bar{T}_{k_i} V = 0 \tag{5}$$

with $\bar{v}_{11}, \bar{v}_{21}, \ldots, \bar{v}_{\bar{m}_g 1}$ linearly independent. Integer $\bar{k}_i$ is the length of the ith chain of eigenvectors associated to the infinity. The orders of the zeros or poles at infinity, as they appear in the Smith-McMillan form at infinity of $A(s)$, depend on integers $\bar{k}_i$, see [21]. For our purposes in this paper we simply consider that $A(s)$ has $\bar{m}_A$ zeros at infinity. The following result is instrumental for the analysis of our algorithms.

Lemma 1 *Let n_f be the number of finite zeros (including multiplicities) of a polynomial matrix $A(s)$ of degree d and rank ρ, $\bar{m}_A$ the number of zeros at infinity, d_l and d_r the sum of the degrees of all the vectors in a minimal basis of the left and right null-spaces respectively. Then,*

$$\rho d = n_f + \bar{m}_A + d_r + d_l \tag{6}$$

Proof: The proof is easily derived from section 3.6 in [29] using the definition of zeros at infinity given above. See also [30]. ■

Notice that from (4) and (5), the problem of finding the zero structure of $A(s)$ is reduced to the problem of finding the null-space of some constant block Toeplitz matrices. We follow the same idea to compute the null-space basis of $A(s)$. Taking $A(s)$ in the form (1) and $z(s) = z_0 + z_1 s + z_2 s^2 + \cdots + z_\delta s^\delta$, it follows that, by identifying like powers of s, the polynomial equation (2) is equivalent to the constant linear system of equations

$$\begin{bmatrix} A_d & & \\ \vdots & \ddots & \\ A_0 & & A_d \\ & \ddots & \vdots \\ & & A_0 \end{bmatrix} \begin{bmatrix} z_\delta \\ z_{\delta-1} \\ \vdots \\ z_0 \end{bmatrix} = T_{\delta+1} \mathcal{Z}_\delta = 0. \tag{7}$$

Matrix $T_{\delta+1}$ is called the Sylvester matrix of degree $\delta+1$ associated to $A(s)$. Sylvester matrices arise in several areas [2, 23, 31, 32], and the idea is certainly not new. In this case, the approach allows to obtain $z(s)$ by computing the constant null-space of $T_{\delta+1}$. Matrix $T_{\delta+1}$ is in block Toeplitz form, so we call our algorithms *block Toeplitz algorithms*.

To compute the null-space of $T_{\delta+1}$, several numerically reliable methods can be used. Here, based on the results in [26, 33] where we investigated the reliability and efficiency of different numerical methods like the QR factorization, the Column Echelon form and also some displacement structure methods based on the generalized Schur algorithm [34], we propose the LQ factorization as the rank revealing method. The LQ factorization is reviewed in the next subsection. The problem remaining when solving (7) is the estimation of the degree δ. To solve this problem we process iteratively block Toeplitz matrices T_i of increasing dimensions as we explain in Section 3.

2.1 LQ factorization

The LQ factorization of a $m \times n$ matrix is given by $M = LQ^T$, where matrix L is in lower quasi-triangular form and matrix Q is orthogonal, i.e. $QQ^T = Q^T Q = I_n$. For $m \geq n$, the number of floating point elementary operations (flops) required by the LQ factorization is $O(2mn^2 - \frac{2}{3}n^3)$ when matrix Q is not updated. If we require Q, we have to perform $O(4m^2 n - 4mn^2 + \frac{4}{3}n^3)$ more operations [35].

For the LQ factorization to reveal the rank ρ of M, a row pivoting is necessary. In practice, we apply at step j a Householder transformation to zero the $n-j$ rightmost entries of a previously chosen row of $L_{j-1} = MQ_1 Q_2 \ldots Q_{j-1}$. To choose the row at step j, several strategies can be followed. For instance, we can compute $\|L_{j-1}(i, j+1:n)\|_2$ for all rows i not yet analyzed, and choose the row with the largest norm. Notice that row pivoting yields the position of the linearly independent rows of M. So, a row permutation matrix P can be applied to put L in the form

$$PL = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \tag{8}$$

where L_{11} is a triangular $\rho \times \rho$ matrix with non-zero elements along the diagonal.

The LQ factorization is backward stable. It can be shown [27, 36] that the computed matrices $\hat{L}$ and $\hat{Q}$ satisfy $(M+\Delta)\hat{Q} = \hat{L}$ with $\|\Delta\|_2 \leq O(\varepsilon)\|M\|_2$. Moreover, if we partition L as in (8), then (to first order),

$$\frac{\|\hat{L}_{22} - L_{22}\|_2}{\|M\|_2} \leq \frac{\|\Delta\|_2}{\|M\|_2}(1 + \|L_{11}^{-1}L_{21}^T\|_2) \tag{9}$$

where $\|L_{11}^{-1}L_{21}^T\|_2$, the conditioning of the LQ factorization for rank-deficient matrices, is usually small [37].

So, from equation (9) we can expect to recover the rank ρ of M by applying the LQ factorization and counting the number of columns of $\hat{L}_{22}$ such that $\|\hat{L}_{22}\|_2 \leq \mu\|M\|_2$ where μ is a tolerance depending on ε. If $\operatorname{rank} M = \rho$, then $\|M\hat{Q}(:, \rho+1:n)\|_2$ is suitably small. So, matrix $\hat{Q}(:, \rho+1:n)$ is a computed basis of the null-space of M. The problem with the LQ factorization as a rank revealing method is that $\|\hat{L}_{22}\|_2$ is not always sufficiently small. Fortunately this problem occurs only with some very carefully chosen matrices, see Example 5.5.1 in [35].

Row permutation P in (8) is instrumental to proving stability and accuracy of the LQ factorization as a rank revealing method. It is also important when solving linear systems of equations since it puts the system in a triangular (or echelon) form allowing for forward substitution. However, this permutation has no practical effect in revealing the rank. If one finds ρ linearly independent rows in M, then matrix $Q(:, \rho+1:n)$ is a basis of its null-space regardless on the order of its rows. Moreover, row permutation would destroy the Toeplitz structure when used to solve (7). So, when we use the LQ factorization in this paper, we apply the pivoting strategy only to choose the linearly independent rows but *we do not perform row permutations*. The next example clarifies this idea.

Example 1 *Let*

$$M = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 0 & 2 & 2 \\ 2 & -2 & 0 & -4 & -4 \\ 0 & 1 & 1 & 2 & 3 \end{bmatrix}.$$

The norm of row 3 is the largest, so we apply a first Householder transformation to zero all its elements but the first one[2]

$$L_1 = MQ_1 = \begin{bmatrix} -0.32 & -0.61 & 1.00 & -1.22 & -0.22 \\ -3.16 & 0.00 & 0.00 & 0.00 & 0.00 \\ 6.32 & -0.00 & 0.00 & -0.00 & -0.00 \\ -3.48 & -0.61 & 1.00 & -1.22 & -0.22 \end{bmatrix}.$$

Row 3 of M is then our first linearly independent row. We observe that row 2 of M is a linear combination of row 3.

Norms of $L_1(1, 2:5)$ and $L_1(4, 2:5)$ are equal so we can apply the next Householder transformation on row 1 for instance

$$L = MQ_1Q_2 = \begin{bmatrix} -0.32 & 1.70 & 0.00 & -0.00 & 0.00 \\ -3.16 & -0.00 & 0.00 & 0.00 & 0.00 \\ 6.32 & 0.00 & -0.00 & 0.00 & -0.00 \\ -3.48 & 1.70 & 0.00 & -0.00 & 0.00 \end{bmatrix} = MQ.$$

Row 1 of M is then our second linearly independent row.

[2]To save space we display only 2 digits behind the decimal point.

This means that row 4 of M is a linear combination of rows 1 and 3. So, matrix M has rank 2 and a basis of its null-space is given by $Q(:,3:5)$. No row permutations were carried out!

The LQ factorization of M could be performed using the standard numerical linear algebra software libraries as the dual of the QR factorization with column pivoting, namely, working on the transpose of M, i.e. $M^T P = QR$ where P is a column permutation matrix (for more details see [35]). Clearly $P^T M = \tilde{L}Q^T$ and, if no row permutation is required, $M = P\tilde{L}Q^T = LQ^T$.

3 Main results

Knowing the degrees δ_i, solving block Toeplitz equation (7) yields the expected null-space basis vectors. Now we analyze how to obtain these degrees. Let

$$T_i = \begin{bmatrix} A_d & & \\ \vdots & \ddots & \\ A_0 & & A_d \\ & \ddots & \vdots \\ & & A_0 \end{bmatrix}$$

be a block Toeplitz matrix with i block columns associated to the polynomial matrix $A(s)$.

Proposition 1 *The following structural information of $A(s)$ is obtained by analyzing iteratively block Toeplitz matrices T_i of increasing dimensions. Let $r_0 = n$, $r_1 = \operatorname{rank} T_1$ and $r_i = \operatorname{rank} T_i - \operatorname{rank} T_{i-1}$ for $i > 1$. Then*

(a) Indices r_i satisfy $r_i \geq r_{i+1}$.

(b) $x_i = r_i - r_{i+1}$ is the number of vectors of degree i in a minimal polynomial basis.

(c) When $r_i = \rho$ for some $i = w$, we have determined all the degrees of the vectors in a minimal polynomial basis.

(d) If $r_w = \rho$ then $r_i = \rho$ for $i > w$.

(e) Index w is finite.

Proof: First notice that matrix T_2 has the form

$$T_2 = \begin{bmatrix} \boxed{T_1} & 0 \\ 0 & \boxed{T_1} \end{bmatrix} = \begin{bmatrix} \boxed{T_1} & \boxed{T_2^*} \\ 0 & \end{bmatrix}$$

and let T_2^* denotes the rightmost block column of T_2. Suppose that $\operatorname{rank} T_1 = r_1$ and that the column k of T_1 is linearly dependent, so columns k and $k+n$ in T_2 are also linearly dependent. On the other hand, because of the Toeplitz structure, if column j in T_1 is linearly independent, it may happen that column $j+n$ in T_2, namely column j in T_2^*, is linearly dependent. In other words $\operatorname{rank} T_2 \leq \operatorname{rank} T_1 + r_1$, i.e. $r_2 \leq r_1$.

Similarly,

$$T_3 = \begin{bmatrix} \boxed{T_2} & 0 \\ 0 & \boxed{T_2^*} \end{bmatrix} = \begin{bmatrix} \boxed{T_2} & \boxed{T_3^*} \\ 0 & \end{bmatrix},$$

where T_3^* denotes the rightmost block column of T_3. So, if column j in T_2^* is linearly independent, it may happen that column j in T_3^* is linearly dependent, namely, $\operatorname{rank} T_3 \leq \operatorname{rank} T_2 + r_2$, i.e. $r_3 \leq r_2$.

In general, matrix T_i can be written as

$$T_i = \left[\begin{array}{cc} \boxed{T_{i-1}} & 0 \\ 0 & \boxed{T_{i-1}^*} \end{array} \right] = \left[\begin{array}{cc} \boxed{T_{i-1}} & \boxed{T_i^*} \\ 0 & \end{array} \right],$$

so, using the same reasoning as above, it follows that $r_i \leq r_{i-1}$ which proves (a).

From (7) it follows that, if T_1 has full rank, then $A(s)$ has no vectors of degree 0 in the minimal basis of its null-space. On the other hand, if column k in T_1 is linearly dependent, then column k in $A(s)$ is a combination of degree 0 of the other columns. Namely, we have a column $z_1(s) = z_{10}s^0$ in $Z(s)$. As we explain above, column $k+n$ in T_2, i.e. column k in T_2^*, is also linearly dependent on the other columns in T_2^*. In that way, a linear combination for column $k+n$ in T_2 could be $[0; z_{10}]$ which corresponds to a vector $z_2(s) = z_1(s)$ which cannot be in the basis of the null-space. However note that if column j, linearly independent in T_1, is dependent in T_2^*, then a linear combination of column $j+n$ in T_2 has the form $[z_{21}; z_{20}]$ which corresponds to a valid vector $z_2(s) = z_{20} + z_{21}s$ for the minimal basis $Z(s)$. In conclusion, the number x_1 of vectors of degree 1 in a minimal basis is equal to the number of linearly dependent columns added by T_2^*, i.e. $x_1 = n - r_2 - (n - r_1) = r_1 - r_2$.

Similarly, column $j + 2n$ in T_3 is a linear combination of the form $[0; z_{21}; z_{20}]$, namely, it corresponds to an invalid vector for the minimal basis $Z(s)$. On the contrary, if column q is linearly independent in T_2^* but dependent in T_3^*, then we can find a new vector in the form $z_3(s) = z_{30} + z_{31}s + z_{32}s^2$ for the minimal basis. So, the number x_2 of vectors of degree 2 in a minimal basis is equal to $x_2 = n - r_3 - (n - r_2) = r_2 - r_3$. In general, the number x_i of vectors of degree i in a minimal basis is equal to the number of linearly dependent columns added by T_{i+1}^*, i.e. $x_i = r_i - r_{i+1}$ which proves (b).

Now, suppose the last vectors in the minimal basis $Z(s)$ have degree $w-1$, so the sum $x_0 + x_1 + \cdots + x_{w-1}$ is equal to $n - \rho$, the nullity of $A(s)$. Now using (b) to substitute x_i in the last equation, it follows that $r_w = \rho$ which proves (c). Point (d) is easily derived from (b) and (c): $x_i = 0$ for $i \geq w$, so $\rho - r_{i+1} = 0$.

Finally, from (6) we know that $d_r = \delta_1 + \cdots + \delta_{n-r}$ is finite, so no vector in the minimal basis $Z(s)$ can have infinite degree. In other words, $r_w = \rho$ for a finite index w, which proves (e). Notice that $w \leq \rho d + 1$. A tight upper bound for w is given by $w \leq \sum_{i=1}^{n} \deg A_i(s) - \min_i \deg A_i(s) + 1$, with $\deg A_i(s)$ denoting the degree of the ith column of $A(s)$, see [19]. ■

An algorithm to obtain a full-rank minimal degree polynomial matrix $Z(s)$ such that $A(s)Z(s) = 0$ was sketched in the above Proposition and its proof, see also Theorem 1 and its proof in [31]. Some remarks on this algorithm are as follows:

Remark 1. In order to know when to stop the algorithm, it is necessary to determine the value of ρ, the rank of the analyzed matrix. This could be done iteratively by computing also the infinite structure of $A(s)$. From (5), the infinite structure of $A(s)$ is also contained in the null-space of some Toeplitz matrices. Let

$$\bar{T}_i = \begin{bmatrix} A_d & & \\ \vdots & \ddots & \\ A_{d-k_i+1} & \cdots & A_d \end{bmatrix}$$

be a block Toeplitz matrix with i block columns associated to the polynomial matrix $A(s)$.

Proposition 2 *The following structural information of $\bar{A}(s)$ is obtained by analyzing iteratively block Toeplitz matrices $\bar{T}_i$ of increasing dimensions. Let $\bar{r}_0 = 0$, $\bar{r}_1 = \operatorname{rank} \bar{T}_1 = \operatorname{rank} A_d$ and $\bar{r}_i = \operatorname{rank} \bar{T}_i - \operatorname{rank} \bar{T}_{i-1}$ for $i > 1$. Then*

(a) Indices $\bar{r}_i$ satisfy $\bar{r}_i \leq \bar{r}_{i+1}$.

(b) *Integer $x_i = \overline{r_{i+1}} - \overline{r_i}$ is the number of chains containing i eigenvectors at infinity.*

(c) *When $\overline{r_i} = \rho$ for some $i = w$, then we have determined the lengths of all the chains of eigenvectors at infinity.*

(d) *If $\overline{r_w} = \rho$ then $\overline{r_i} = \rho$ for $i > w$.*

(e) *Index w is finite.*

Proof: Similar to the one of Proposition 3.1, for more details see [38]. ■

So, in the case we do not know the rank of $A(s)$, we can simply start with $\rho = \min(m,n)$ and update this value when finding vectors in $Z(s)$. At the same time integers $\overline{r_i}$ must be computed and, since $\overline{r_i}$ is increasing and r_i is decreasing, the algorithm should stop when $\overline{r_w} = r_w$.

Example 2 *Let*

$$A(s) = \begin{bmatrix} 1 & s^3 & 0 & 0 \\ 0 & 1 & s & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Supposing that we do not know the rank of $A(s)$, we can start with the guess $\rho = \min(m,n) = 3$. First we obtain $\overline{r_1} = 1$ and $r_1 = 3$ which means that there is one vector of degree 0 in $Z(s)$. We have $r_1 = \rho$ but $\overline{r_1} \neq \rho$ which means that there are zeros at infinity and therefore that the rank of $A(s)$ cannot be 3. After 3 more steps we obtain values $\overline{r_2} = 1, \overline{r_3} = \overline{r_4} = 2$ and $r_2 = r_3 = r_4 = 3$, which confirms the existence of zeros at infinity in $A(s)$. Finally, we obtain $\overline{r_5} = r_5 = 2$ and the algorithm stops. In conclusion, matrix $A(s)$ has rank $\rho = r_5 = 2$, it has one vector of degree 0 and one vector of degree 4 in a minimal basis of its null-space, and also $\overline{r_3} - \overline{r_2} = 1$ chain of eigenvectors at infinity of length 2, i.e. 2 zeros at infinity. Notice that equation (6) *is satisfied.*

Remark 2. Now note that Toeplitz matrix $\overline{T_i}$ is located in the uppermost *im* rows of T_i. Therefore, indices $\overline{r_i}$ can also be obtained when analysing matrices T_i by searching first for pivots in its uppermost *im* rows. This strategy could also help to improve the accuracy of the computed results as explained in the subsection 4.2.

Remark 3. The LQ factorization can be used as the rank revealing method without sacrificing significantly the accuracy [27, 35, 36], but performing less operations than the SVD. Next we also show how the block structure of matrices T_i can be used to improve the algorithm by reducing even more the number of operations. We point out that the rank and the nullity of T_{i-1} can be used to process T_i. Moreover, we show how we can determine the vectors in the minimal basis directly from each computed orthogonal matrix Q.

Let us consider a 2×4 polynomial matrix $A(s)$ with degree $d = 1$. Suppose that after applying an LQ factorization to T_1, we obtain[3]

$$T_1 Q_1 = \begin{bmatrix} \times & \times & 0 & 0 \\ \times & 0 & 0 & 0 \\ \times & \times & \times & 0 \\ \times & \times & \times & 0 \end{bmatrix} = L_1$$

and indices $\overline{r_1} = 2$, $r_1 = 3$. So there is a vector $z_1(s) = Q_1(:,4)$ of degree 0 in the minimal basis. Now notice that

$$T_2 \begin{bmatrix} Q_1 & 0 \\ 0 & Q_1 \end{bmatrix} = \begin{bmatrix} L_1 & 0 \\ 0 & L_1 \end{bmatrix} = \left[\begin{array}{cc|cccccc} \times & \times & 0 & 0 & & & & \\ \times & 0 & 0 & 0 & & & & \\ \hline \times & \times & \times & 0 & \times & \times & 0 & 0 \\ \times & \times & \times & 0 & \times & 0 & 0 & 0 \\ & & & & \times & \times & \times & 0 \\ & & & & \times & \times & \times & 0 \end{array}\right] = T_2'. \qquad (10)$$

[3]Notice that no row permutation is performed. We follow a pivoting strategy to chose the linear independent rows and we first search for pivots in the uppermost 2 rows of each analyzed matrix. For more details about pivoting and numerical stability, see sections 4.1 and 4.2.

So, it follows that at step 2 we can apply the LQ factorization only to the sub-matrix

$$T_2'' = T_2'(3:6,[3\ 5\ 6\ 7]). \tag{11}$$

Doing that, suppose we obtain

$$T_2''Q_2'' = \begin{bmatrix} + & + & 0 & 0 \\ + & 0 & 0 & 0 \\ + & + & + & 0 \\ + & 0 & 0 & 0 \end{bmatrix} = L_2.$$

Then, if $Q_2'' = [q_{ij}]$, we can write

$$Q_2' = \left[\begin{array}{cc|cccccc} 1 & 0 & & & & & & \\ 0 & 1 & & & & & & \\ \hline & & q_{11} & 0 & q_{12} & q_{13} & q_{14} & 0 \\ & & 0 & 1 & 0 & 0 & 0 & 0 \\ & & q_{21} & 0 & q_{22} & q_{23} & q_{24} & 0 \\ & & q_{31} & 0 & q_{32} & q_{33} & q_{34} & 0 \\ & & q_{41} & 0 & q_{42} & q_{43} & q_{44} & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right], \tag{12}$$

so that

$$T_2\begin{bmatrix} Q_1 & 0 \\ 0 & Q_1 \end{bmatrix}Q_2' = T_2Q_2 = \left[\begin{array}{cc|cccccc} \times & \times & & & & & & \\ \times & 0 & & & & & & \\ \hline \times & \times & + & 0 & + & 0 & 0 & 0 \\ \times & \times & + & 0 & 0 & 0 & 0 & 0 \\ & & + & 0 & + & + & 0 & 0 \\ & & + & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

We obtain indices $\overline{r}_2 = 2$, $r_2 = 2$ and then there is also a vector of degree 1 in the minimal basis. Moreover $\overline{r}_2 = r_2 = 2$, so the rank of matrix $A(s)$ is 2 and we stop. The columns of orthogonal matrix Q_2 are given by

$$Q_2 = \begin{bmatrix} Q_1(:,1) & Q_1(:,2) & \# & Q_1(:,4) & \# & \# & \# & 0 \\ 0 & 0 & \# & 0 & \# & \# & \# & Q_1(:,4) \end{bmatrix}.$$

It is easy to see that $Z(s) = [z_1(s)z_2(s)]$, where $z_2 = Q_2(1:4,7)s + Q_2(5:8,7)$. Finally notice that $Q_2(:,[4\,8])$ corresponds to the polynomial vectors $sz_1(s)$ and $z_1(s)$ which cannot be in the basis as we pointed out in the proof of Proposition 3.1.

The *blocked* formulation explained above allows to reduce the number of operations required at each step. In fact, the number of rows of the analyzed matrices is always equal to $m(d+1)$, and the number of columns is smaller than *in* at each step. More about algorithmic complexity is presented in section 4.3. Next we describe the algorithm formally using a Matlab like pseudo-code.

Algorithm 1 *(Blocked LQ algorithm)*
Input: An $m \times n$ polynomial matrix $A(s)$ of degree d.
Output: The rank ρ of $A(s)$, and a matrix $Z(s) = [z_1(s)\ z_2(s)\ \cdots\ z_{n-\rho}(s)]$ where the $z_i(s)$ are vectors in a minimal basis of the null-space of $A(s)$, namely $A(s)Z(s) = 0$. As a sub-product, the algorithm returns the structure at infinity of $A(s)$.

1. *Build matrix T_1 and compute its LQ factorization $T_1Q_1 = L_1$ searching first for pivots in the upper-most row block of T_1. Determine r_1 and $\overline{r}_1$. If $r_1 < n$ then let $Z = Q_1(:,r_1+1:n)$. If $r_1 = \overline{r}_1$ then let $k = 0$ and go to step f.*

i. *Construct matrix T_i'' from L_{i-1} and L_1 as exemplified in (10) and (11). Carry out the LQ factorization $T_i''Q_i'' = L_i$ searching first for pivots in the uppermost row block of T_i''. Construct matrix Q_i' as exemplified in (12), update matrix*

$$Q_i = \begin{bmatrix} Q_{i-1} & 0 \\ 0 & Q_1 \end{bmatrix} Q_i'$$

and determine r_i and $\overline{r_i}$. If $x_{i-1} = r_{i-1} - r_i > 0$ then update matrix

$$Z = \left[\begin{array}{c|c} 0 & Q_i(:,(i-1)n + r_i + 1 : (i-1)n + r_{i-1}) \\ Z & \end{array} \right].$$

If $r_i = \overline{r_i}$ then let $k = i-1$ and go to step f.

f. *The rank of $A(s)$ is $\rho = r_{k+1}$, and*

$$Z(s) = Z(1:n,:)s^k + Z(n+1:2n,:)s^{k-1} + \cdots + Z((k-1)n+1:kn,:)s^1 + Z(kn+1:(k+1)n,:).$$

As a sub-product notice that $A(s)$ has $x_i = \overline{r_{i+1}} - \overline{r_i}$ chains containing i eigenvectors at infinity, for $i = 1,2,\ldots,q-1$ where q is such that $\overline{r_q} = \rho$. Moreover, these eigenvectors can also be obtained in a similar way from the orthogonal matrices computed at each step.

Example 3 *Let*

$$A(s) = \begin{bmatrix} 1 & -2 & s & -2s^2 & -2+s^3 \\ 2 & s & 2s & s^3 & s+s^3 \end{bmatrix}.$$

At step 1 we compute the factorization $T_1Q_1 = L_1$ where[4]

$$L_1 = \begin{bmatrix} 0.71 & 0.71 & 0.00 & 0.00 & 0.00 \\ 1.41 & 0.00 & 0.00 & 0.00 & 0.00 \\ -1.41 & 1.41 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.89 & 0.39 & -0.22 \\ 0.71 & 0.71 & 2.24 & 0.00 & -0.00 \\ -1.41 & -1.41 & -0.89 & 2.05 & -0.00 \\ -0.00 & 0.00 & 0.00 & 0.98 & 1.75 \end{bmatrix}$$

and

$$Q_1 = \begin{bmatrix} -0.00 & 0.00 & 0.00 & 0.49 & 0.87 \\ 0.00 & -0.00 & 0.45 & -0.78 & 0.44 \\ 0.00 & 0.00 & 0.89 & 0.39 & -0.22 \\ 0.71 & -0.71 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.71 & 0.00 & 0.00 & 0.00 \end{bmatrix}.$$

So $r_1 = 5$ and $\overline{r_1} = 2$. At step 2 we construct the matrix

$$\begin{aligned} T_2'' &= \left[\begin{array}{c|c} L_1(3:8,3:5) & L_1 \\ 0 & \end{array} \right] \\ &= \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.71 & 0.71 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.41 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.89 & 0.39 & -0.22 & -1.41 & 1.41 & 0.00 & 0.00 & 0.00 \\ 2.24 & 0.00 & -0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.89 & 2.05 & -0.00 & 0.00 & 0.00 & 0.89 & 0.39 & -0.22 \\ 0.00 & 0.98 & 1.75 & 0.71 & 0.71 & 2.24 & 0.00 & -0.00 \\ 0.00 & 0.00 & 0.00 & -1.41 & -1.41 & -0.89 & 2.05 & -0.00 \\ 0.00 & 0.00 & 0.00 & -0.00 & 0.00 & 0.00 & 0.98 & 1.75 \end{bmatrix}. \end{aligned}$$

[4]Notice that no row permutations are performed and that pivots are searched first in the uppermost two rows of the analysed Toeplitz matrices.

Then we compute $T_2''Q'' = L_2$ and we obtain[5] *$r_2 = 4$ and $\bar{r}_2 = 2$. Finally we update*

$$Q_2 = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_1 \end{bmatrix} Q_2', \quad \text{where} \quad Q_2' = \left[\begin{array}{c|c} I_2 & 0 \\ \hline 0 & Q_2'' \end{array}\right]$$

and $Z = Q_2(:,10)$.

At step 3 we construct

$$T_3'' = \left[\begin{array}{c|c} L_2(3:8,3:7) & L_1 \\ 0 & \end{array}\right].$$

Then we compute $T_3''Q_3'' = L_3$ and we obtain $\bar{r}_3 = 2$ and $r_3 = 3$. We update matrix

$$Q_3 = \begin{bmatrix} Q_2 & 0 \\ 0 & Q_1 \end{bmatrix} Q_3'$$

where

$$Q_3' = \left[\begin{array}{c|ccc} I_4 & 0 & 0 & 0 \\ \hline 0 & Q_3''(1:5,1:5) & 0 & Q_3''(1:5,6:10) \\ 0 & 0 & 1 & 0 \\ 0 & Q_3''(6:10,1:5) & 0 & Q_3''(6:10,6:10) \end{array}\right]$$

and we update

$$Z = \begin{bmatrix} 0 & Q_3(:,14) \\ Z & \end{bmatrix}.$$

At step 4 we construct

$$T_4'' = \left[\begin{array}{c|c} L_3(3:8,3:8) & L_1 \\ 0 & \end{array}\right]$$

and we compute $T_4''Q_4'' = L_4$. We obtain $\bar{r}_4 = 2$, and $r_4 = 2$, then we update

$$Q_4 = \begin{bmatrix} Q_3 & 0 \\ 0 & Q_1 \end{bmatrix} Q_4'$$

where

$$Q_4' = \left[\begin{array}{c|ccccc} I_6 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & Q_4''(1:3,1:3) & 0 & Q_4''(1:3,4:6) & 0 & Q_4''(1:3,7:11) \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & Q_4''(4:6,1:3) & 0 & Q_4''(4:6,4:6) & 0 & Q_4''(4:6,7:11) \\ 0 & 0 & 0 & 0 & I_2 & 0 \\ 0 & Q_4''(7:11,1:3) & 0 & Q_4''(7:11,4:6) & 0 & Q_4''(7:11,7:11) \end{array}\right],$$

and we update

$$Z = \begin{bmatrix} 0 & Q_4(:,18) \\ Z & \end{bmatrix}.$$

Finally, we stop since $r_4 = \bar{r}_4 = 2$. The rank of $A(s)$ is $\rho = 2$ and a minimal basis is given by

$$\begin{aligned} Z(s) &= Z(1:5,:)s^3 + Z(6:10,:)s^2 + Z(11:15,:)s + Z(16:20,:) \\ &= \begin{bmatrix} -0.71s & 0.70s & -0.13s + 0.08s^3 \\ 0.00 & -0.07s^2 & 0.32 + 0.08s - 0.60s^2 - 0.04s^3 \\ 0.71 & -0.70 & 0.13 + 0.08s^2 + 0.08s^3 \\ 0.00 & 0.07 & 0.60 - 0.04s \\ 0.00 & 0.00 & -0.32 - 0.08s \end{bmatrix}. \end{aligned}$$

Since $\bar{r}_1 = \bar{r}_2 = \bar{r}_3 = \bar{r}_4 = 2 = \rho$, it follows that $A(s)$ has no zeros at infinity. Moreover, since the sum of the degrees of the vectors in the minimal basis $Z(s)$ is $6 = \rho d$, we conclude that $A(s)$ has no finite zeros either, cf. equation (6).

[5] To save space we do not show computed matrices L_i and Q_i''. Row pivoting is performed as required by Algorithm 1 and no row permutation is applied.

4 Algorithmic analysis

Now we analyze the features of the blocked LQ algorithm described in the previous section. The analysis is divided in three parts. First we prove the numerical stability, then we argue about the accuracy of the computed results and the pivoting strategy, finally we focus on the algorithmic complexity.

4.1 Numerical stability

We have seen in section 2.1 that the LQ factorization of T_i is backward stable. Showing the backward stability of our blocked LQ algorithm is not difficult. One only has to consider the blocked process described in the Remark 3 as a sequence of some particular Householder transformations and then apply the proof of Theorem 18.3 in [27], as illustrated next.

Consider the computed LQ factorization

$$\hat{L}_1 = (T_1 + \Phi_1)\hat{H}_1\hat{H}_2\cdots\hat{H}_p = (T_1 + \Phi_1)\hat{Q}_1$$

where $\|\Phi_1\|_2 \leq \phi_1\|T_1\|_2$. Here $\phi_1 = O(\varepsilon)$ is a small constant depending on the dimensions of T_1, the machine precision ε, and the number of applied Householder reflections (§18.3 of [27]). Now, when computing the LQ factorization of T_2 using the blocked process described above, we have

$$\hat{L}_2 = (T_2 + \Phi_2)\begin{bmatrix} \hat{Q}_1 & 0 \\ 0 & \hat{Q}_1 \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & \hat{H}_{p+1} \end{bmatrix}\cdots\begin{bmatrix} I & 0 \\ 0 & \hat{H}_{p+q} \end{bmatrix} = (T_2 + \Phi_2)\hat{Q}_2,$$

where $\|\Phi_2\|_2 \leq \phi_2\|T_2\|_2$. Similarly, an upper bound for $\phi_2 = O(\varepsilon)$ can be computed as in §18.3 of [27]. Clearly, this bound for the backward error will be more pessimistic, which can be expected since we have performed q more Householder reflections and the dimension of the matrices has increased. We summarize these results in the following lemma.

Lemma 2 *Consider that polynomial matrix $A(s)$ has a vector of degree δ in a basis of its null-space. The application of the blocked LQ factorization to solve system* (7) *is backward stable, namely $(T_{\delta+1} + \Delta)\hat{\mathcal{Z}}_\delta = 0, \quad \|\Delta\|_2 \leq \phi\|T_{\delta+1}\|_2$, where $\hat{\mathcal{Z}}_\delta$ contains the computed coefficients of the vector in the null-space basis, and $\phi = O(\varepsilon)$ is a small constant depending on the dimensions of $T_{\delta+1}$, the machine precision ε, and the number of applied Householder reflections.*

This result shows that $\hat{\mathcal{Z}}_\delta$ is the exact null-space of the slightly perturbed matrix $T_{\delta+1} + \Delta$. Nevertheless, we require that vector $\hat{z}(s)$, constructed from $\hat{\mathcal{Z}}_\delta$, is the exact polynomial vector in the null-space of the slightly perturbed matrix $A(s) + \Delta(s)$, where $\Delta(s) = \Delta_0 + \Delta_1 s + \cdots + \Delta_d s^d$, and $\|\Delta(s)\|^* \leq O(\varepsilon)\|A(s)\|^*$ for some polynomial matrix norm $\|\cdot\|^*$. Ideally we want

$$\|\Delta_i\|_2 \leq O(\varepsilon)\|A_i\|_2, \quad i = 0, 1, \ldots, d. \tag{13}$$

The following result gives an upper bound for the error $\Delta(s)$ when using the blocked LQ factorization to solve (7), see also [39].

Theorem 1 *Let $A(s)$ be a polynomial matrix and suppose that it has a vector $z(s)$ of degree δ in a minimal basis of its null-space. The computed vector $\hat{z}(s)$, obtained from* (7) *via the blocked LQ algorithm, is the exact null-space vector of the slightly perturbed matrix $A(s) + \Delta(s)$ where*

$$\|\Delta(s)\|^* = \|\bar{\Delta}\|_2 \leq \gamma\|T_{\delta+1}\|_2 = \gamma\|A(s)\|^*, \tag{14}$$

$$\bar{\Delta} = \begin{bmatrix} \Delta_d & & \\ \vdots & \ddots & \\ \Delta_0 & & \Delta_d \\ & \ddots & \vdots \\ & & \Delta_0 \end{bmatrix},$$

and $\gamma = O(\varepsilon)$ is a small constant depending on ε.

Proof: From Lemma 2 we know that $(T_{\delta+1}+\Delta)\hat{\mathcal{Z}}_\delta = 0$ with $\|\Delta\|_2 \leq \phi\|T_{\delta+1}\|_2$. Error matrix Δ has the following (non block Toeplitz) general form:

$$\Delta = \begin{bmatrix} \Delta_d^0 & & \\ \vdots & \ddots & \\ \Delta_0^0 & & \Delta_d^\delta \\ & \ddots & \vdots \\ \times & & \Delta_0^\delta \end{bmatrix}$$

where $\times$ represents possibly non zero blocks. However, note that we can apply row elementary operations gathered in a left multiplier P such that $P(T_{\delta+1}+\Delta)\hat{\mathcal{Z}}_\delta = (T_{\delta+1}+\bar{\Delta})\hat{\mathcal{Z}}_\delta = 0$. Existence of P is equivalent to having a matrix E with $\|E\|_2 \leq \phi\|T_{\delta+1}\|_2$, such that $P = I+E$ and then $P(T_{\delta+1}+\Delta) = T_{\delta+1}+\Delta+E(T_{\delta+1}+\Delta) = T_{\delta+1}+\bar{\Delta}$, namely

$$\Delta + E(T_{\delta+1}+\Delta) = \bar{\Delta}. \tag{15}$$

For the error analysis to make sense, we require that Δ and $\bar{\Delta}$ do not modify the rank of $T_{\delta+1}$ (since otherwise the structure of the null-space would change and the problem would be ill posed), so matrix E satisfying (15) has infinite solutions. In consequence error matrix $\bar{\Delta}$ is not unique, but we can always ensure that, to first order, $\|\bar{\Delta}\|_2 \leq \gamma\|T_{\delta+1}\|_2$ which is the expected result. ■

Theorem 1 and its proof seem to be incomplete whereas the value of γ is unknown. Obtaining exact expressions for the backward error is out of the scope of this paper, here we simply propose an expression for γ using the standard definition of relative backward error as in [27], namely, we look for the smallest value satisfying (14) in Theorem 1, i.e.

$$\begin{aligned} \gamma \;&=\; \min\Big\{\varepsilon \;\Big|\; (A(s)+\Delta(s))\hat{z}(s) = 0, \\ &\qquad \|\Delta(s)\|^* = \|\bar{\Delta}\|_2 \leq \varepsilon\|T_{\delta+1}\|_2 = \varepsilon\|A(s)\|^*\Big\}. \end{aligned} \tag{16}$$

Taking the norm of the residual $r(s) = r_0 + r_1 s + \cdots + r_k s^k$ given by $r(s) = A(s)\hat{z}(s) = -\Delta(s)\hat{z}(s)$, we obtain a lower bound for ε, i.e. $\|r(s)\|^* \leq |\varepsilon|\|T_{\delta+1}\|_2\|\hat{z}(s)\|^*$, so, the solution of (16) is

$$\gamma = \frac{\|r(s)\|^*}{\|T_{\delta+1}\|_2\|\hat{z}(s)\|^*} \tag{17}$$

where $\|r(s)\|^*$ and $\|\hat{z}(s)\|^*$ are, similarly, equal to the Euclidean norms of the vector of coefficients in the monomial basis, i.e. $\|\hat{z}(s)\|^* = \|\hat{\mathcal{Z}}_\delta\|_2$, $\|r(s)\|^* = \|\mathcal{R}\|_2$ where $\mathcal{R} = \begin{bmatrix} r_k \\ \vdots \\ r_0 \end{bmatrix}$.

Example 4 *Let*

$$A(s) = \begin{bmatrix} s+1.0034 & 2.075 & 1.0034 \\ s^2 & 1 & s^2 - 0.4819277s \end{bmatrix}.$$

Its null-space, computed with Algorithm 1, is given by

$$\hat{z}(s) = \begin{bmatrix} -0.6693113 + 3.563\times10^{-35}s \\ 0.3225597s \\ 0.6693113 - 3.563\times10^{-35}s \end{bmatrix}.$$

The residual is

$$r(s) = A(s)\hat{z}(s) = \begin{bmatrix} 1.998\times10^{-15}s + 3.563\times10^{-35}s^2 \\ 9.437\times10^{-16}s + 1.069\times10^{-50}s^3 \end{bmatrix}$$

so the relative backward error is equal to

$$\gamma = \frac{\|\mathcal{R}\|_2}{\|T_2\|_2\|\hat{\mathcal{Z}}_1\|_2} = 7.298\times10^{-16}.$$

Sharper bounds than (14) and (17), such as the one given by (13), can be expected from a componentwise error analysis [40, 41], see also the second edition of [27]. An analysis with geometrical arguments such as in [42, 43] can also be considered as a future line of research.

A similar analysis on stability can be done for the pencil algorithm to obtain the staircase form (3), see [4] for some ideas. Nevertheless, the stability of the process explained in [11] to compute the searched null-space basis from (3) is not guaranteed. On the other hand, reliable algorithms to compute descriptor realizations are found in [17], however, no expressions for an upper bound of the backward error $\|\Delta(s)\|$ are presented there.

4.2 Accuracy and row pivoting

Theorem 1 shows that our Toeplitz algorithm has a bounded backward error. Nevertheless, the accuracy of the obtained results depends not only on the backward stability of the algorithm but also on the conditioning of the problem. It is well-known that the problem of detecting whether $\operatorname{rank} M = r$ is ill-posed[6] or ∞-conditioned when r is less than the row or column dimension of M. As an important implication, the problem of finding the right null-space of a constant matrix M can be well-conditioned only if M has full row-rank. In that way, the problem of finding the null-space of a polynomial matrix is also ill-posed and the forward error is not bounded. First, notice that the number and the degree of the computed vectors could be different from those corresponding to the exact null-spaces. Second, for a given degree δ, the coefficients of the computed vector could be also different from the exact ones because, in general, matrix $T_{\delta+1}$ has not full row-rank.

Some strategies to render the problem well-posed can be formulated along the lines in [43], see also [44]. Given a first approximation of the structure at infinity or the null-space of $A(s)$, we can think in some kind of iterative refinement over some manifold where this structure is invariant, and hence over which the problem is well-posed. Another possible way to improve the conditioning (posedness) of the matrix can be a scaling along the lines in [45, 46]. The extension of all these results to the polynomial matrix eigenstructure problem is however out of the scope of this paper.

Since these results are still missing, at least we must handle correctly the information obtained at each step by the algorithms. With infinite precision, the rank of T_i computed by only one SVD or LQ factorization is equal to the rank computed using the blocked process of Algorithm 1. Nevertheless, with finite precision this may not be true. In this case, it is important to remember that what we want to obtain ultimately is the infinite and null-space structures of a polynomial matrix, and not a series of ranks of some Toeplitz matrices (rank determination is, after all, not only a problem in matrix computation, but an exercise of interpretation of the numerical results). In the same order of ideas, with the following academic example we show how the blocked process of Algorithm 1 with its particular row pivoting strategy could help to obtain results which correspond to a valid eigenstructure of a given badly scaled polynomial matrix.

Example 5 *Consider the full rank matrix*

$$A(s) = \begin{bmatrix} 10^{-8}s & 10^{-8}s^2 & 1 \\ 20 & 10s & 0 \\ 0 & 1+20s & 10^8 \end{bmatrix}.$$

To compute the infinite structure of $A(s)$, first we use Proposition 2 and a simple LQ factorization (without the blocked process) as rank revealing method. Matrix $\bar{T}_1 = A_2$ has exact rank 1. The rank computed with the LQ factorization is also 1 (a pivot is chosen in the first row), so $\hat{\bar{r}}_1 = 1$. Now, looking at matrix $\bar{T}_2$, we can see that the exact rank is 2. The rank obtained by only one LQ factorization is also 2, so $\hat{\bar{r}}_2 = 1$. Notice, that the LQ factorization in this case picks pivots in rows 6 and 4. Row 1 is now dependent because the pivot was chosen in row 6 which has a larger norm. At step 3, while the exact rank of $\bar{T}_3$ is 5 (i.e. $\bar{r}_3 = \operatorname{rank} A(s) = 3$ which means that $A(s)$ has 2 chains of 2 eigenvectors at infinity), the rank computed by

[6]In the classical sense of Hadamard which means that the solution of the problem does not depend continuously on the data or the parameters.

only one LQ factorization is only 4, so $\hat{\bar{r}}_3 = 2$. Now the pivot in row 4 is also lost, but there is no other row to replace it. Finally, when computing the rank of $\bar{T}_4$ we recover $\hat{\bar{r}}_4 = 3$. This means that $A(s)$ has one computed chain of 2 eigenvectors and another one with 3 eigenvectors, namely 5 computed zeros at infinity, which is far from the correct result.

A similar phenomena occur when obtaining also the null-space. Matrix T_1 has exact rank 3. The rank computed by an LQ factorization is also 3 but no pivot is chosen in the first 3 rows of T_1, so $\hat{r}_1 = 3$ and $\hat{\bar{r}}_1 = 0$. Matrix T_2 has exact rank 6, but the rank computed by only one LQ factorization is 5. In this case, a pivot is chosen in row six, so $\hat{r}_2 = 2$ and $\hat{\bar{r}}_2 = 1$. At step 3, while the exact rank of T_3 is 9, the computed rank is only 7. Aditional pivots are chosen in rows 8 and 9, so $\hat{r}_3 = 2$ and $\hat{\bar{r}}_3 = 2$. In summary, matrix $A(s)$ has a computed rank $\hat{\rho} = 2$, one computed chain of 1 eigenvector at infinity, another one with 2 eigenvectors at infinity (namely $\hat{\bar{m}}_A = 3$), and 1 vector of degree 1 in a minimal basis of its null-space (namely $\hat{d}_r = 1$).

The computed null-space vector is

$$\hat{z}(s) = \begin{bmatrix} 3.236 \times 10^{-18} + 0.4472136s \\ -0.8944272 - 3.388 \times 10^{-25}s \\ 8.944 \times 10^{-10} + 1.789 \times 10^{-8}s \end{bmatrix}$$

with residual

$$r(s) = \begin{bmatrix} 8.944 \times 10^{-10} + 1.789 \times 10^{-8}s - 4.472 \times 10^{-8}s^2 - 3.388 \times 10^{-32}s^3 \\ 6.471 \times 10^{-17} - 3.388 \times 10^{-24}s^2 \\ -3.331 \times 10^{-16} + 0.0000002s - 6.777 \times 10^{-24}s^2 \end{bmatrix}.$$

From (17), the relative backward error is 1.75×10^{-16}, which is in the order of magnitude of the machine precision, nevertheless this result does not correspond to a valid eigenstructure of $A(s)$. First, $\hat{\bar{r}}_1$ cannot be 0 because this means that $A_2 = 0$, namely that the degree of $A(s)$ is only 1. Moreover, the determinant of $A(s)$ is exactly computed as $\det A(s) = 20 + 400s - 1000s^2$ which means that $A(s)$ has two finite zeros (namely $\hat{n}_f = 2$), so the obtained structure is inconsistent since equation (6) cannot be satisfied $\hat{\rho} d = 4 \neq \hat{n}_f + \hat{\bar{m}}_A + \hat{d}_r + \hat{d}_l$ for any value of $\hat{d}_l \geq 0$.

On the other hand, the lector could easily verify that the row pivoting strategy proposed with the blocked Algorithm 1 allows to recover the correct structure: $\hat{\bar{r}}_1 = \hat{\bar{r}}_2 = 1$, $\hat{\bar{r}}_3 = 3$, and $\hat{r}_1 = \hat{r}_2 = \hat{r}_3 = 3$. This is mainly because pivots are chosen first in the uppermost row block of the analysed matrices, and then they are respected in the subsequent steps. Notice that (6) is now satisfied, the rank of $A(s)$ is 3 and, in addition to the 2 finite zeros, we have only 4 zeros at infinity.

4.3 Algorithmic complexity

The blocked LQ algorithm consists on the application of an LQ factorization to some Toeplitz matrices at each step. The total amount of elementary floating point operations (flops) performed by the algorithm is mainly the sum of the flops performed at each step. It depends on several factors such as the dimension and degree of the analysed matrix, but also the structure of the null-space. Since the structure of null-space is an outcome of the algorithm, it is impossible to obtain an exact expression for the number of operations. In this section we derive an upper bound.

For notational ease, we restrict our analysis to a square $n \times n$ polynomial matrix $A(s)$ of degree $d \geq 1$. We always consider the worst case when combining the different factors that have an impact on the total amount of operations. The objective is to obtain an upper bound expression of the type $O(n^{c_1} d^{c_2} f^{c_3})$ where the c_i are integers to be determined and f is the maximum degree of a vector in the computed null-space basis. So we assume that f is a problem data, of the same nature as n and d.

Theorem 2 *Let $A(s)$ be a square polynomial matrix of degree d and dimension n, with a null-space basis of degree at most f. Then the complexity of the* blocked LQ algorithm *computing a minimal basis of its null-space is in $O(n^3 d f^3)$.*

Proof: See the appendix. ■

A similar analysis on complexity can be done for the pencil algorithm to obtain the staircase form (3). From [4, 47] the complexity of the pencil algorithm is in $O(n^3d^3)$ when considering an average value for the null-space degree f. Our analysis is finer in the sense that we incorporate a term depending on the null-space degree. If f is known and fixed, our complexity estimate $O(n^3d)$ compares favorably with $O(n^3d^3)$, and the weak dependency of our algorithm on the degree d could be seen as an advantage.

Another advantage of our algorithm arises when the null-space vectors are required. In this case the amount of operations performed by the other part of the pencil algorithms, namely the process explained in [11] to compute the searched null-space basis from (3), or the method to compute descriptor realizations, should be taken into account. In conclusion, the blocked LQ algorithm is faster, see [26, 33] for some practical and meaningful examples.

The extreme case when the analyzed matrix has only one vector of degree $(n-1)d$ in its null-space, i.e. it has no infinite or finite zeros, implies the largest upper bound for the number of steps performed by our algorithm. If $f = (n-1)d$, then our algorithmic complexity estimate becomes $O(n^6d^4)$. Fortunately, as pointed out in the literature, this kind of matrices appears rarely in practice. In control theory, for instance, the number of steps is generally small since the analyzed polynomial matrices are usually not column-reduced and have elementary divisors accounting for finite or infinite poles and zeros of the modeled linear systems.

5 Concluding Remarks

We have presented a reliable algorithm to obtain a minimal basis of the null-space of an arbitrary polynomial matrix, a problem with relevant applications in control theory. This algorithm can naturally be extended to obtain the whole eigenstructure of a polynomial matrix [25], and thus it can be seen as a reliable alternative to the pencil methods presented in [15]. Our algorithm processes numerically block Toeplitz matrices readily constructed from polynomial matrix coefficients, and no elementary operations over polynomials are needed. The dimension of the analyzed Toeplitz matrices is bounded and depends on the dimension and the degree of the polynomial matrix. The number of steps performed by the algorithm is also bounded, it is at most equal to the number of steps performed by the classical pencil algorithms presented in [4]. Our algorithm is based on the LQ factorization and has a blocked formulation. We showed that it is more efficient than the similar algorithms presented recently in [23, 24]. Moreover, in this paper we presented a full analysis of numerical stability and complexity.

Concerning algorithmic complexity, we can conclude that the weak dependency of our algorithm on the degree of the analyzed polynomial matrix is an advantage with respect to the pencil algorithm. Another feature of our algorithm that has a positive impact on the final amount of performed floating point operations is its blocked formulation. A major advantage of our algorithm is that we obtain, with the same computational method, the structural indices and the associated eigenvectors.

Concerning numerical stability, we have proved the backward stability of our algorithm. We derived a bound for the backward error produced in the coefficients of the analyzed polynomial matrix. Similar bounds are obtained for the algorithm in [4], nevertheless, the stability of the process presented in [11] is not guaranteed. On the other hand, reliable algorithms to obtain descriptor realizations are described in [17] but no bounds for the backward error produced in the coefficients of the polynomial matrix are presented. The bound for our backward error is given by equations (14) and (17). This bound is more pessimistic than the ideal one given by equation (13). Sharper bounds can be expected from a component-wise error analysis [27, 40]. An analysis with geometrical arguments such as in [42, 43] can also be considered as a future line of research. The objective of such an analysis should be the derivation of exact first-order expressions for the errors in the coefficients of the polynomial matrix due to the perturbations in the analyzed Toeplitz matrices. In [45, 46], pre-conditioning techniques to reduce the backward error for the polynomial eigenvalue problem are presented. Similar techniques for the whole polynomial eigenstructure problem are welcome.

Ill-posedness of the rank revealing problem renders the problem of obtaining the null-space, and in general the eigenstructure of a polynomial matrix, also ill-posed or ∞-conditioned. Ill-posed problems arise in several fields of scientific computing and thus, represent a challenge in numerical computations. As explained in [44], a lot of meaningful ill-posed problems can be solved numerically very satisfactorily in practice. In [43] some geometrical strategies allow to reformulate some ill-posed problems restoring the well-posedness and even making them well-conditioned. So, another line of future research could be the extension of these strategies to the polynomial eigenstructure problem. Before this is achieved, we expect that our algorithm with its particular row pivoting strategy can give reliable results in most of the cases.

References

[1] G. D. Forney, Minimal bases of rational vector spaces with applications to multivariable linear systems. *SIAM J. Control Optim.* 13:493–520 (1975).

[2] T. Kailath, *Linear Systems*. Prentice Hall, Englewood Cliffs, 1980.

[3] W. M. Wonham and A. S. Morse, Decoupling and pole assignment in linear multivariable systems: A geometric approach. *SIAM J. Control Optim.* 8:1–18 (1970).

[4] P. M. Van Dooren, The computation of Kronecker's canonical form of a singular pencil. *Linear Algebra Appl.* 27:103–140 (1979).

[5] A. S. Morse, Structural invariants of linear multivariable systems. *SIAM J. Control Optim.* 11:446–465 (1973).

[6] J. J. Loiseau, Sur la modification de la structure à l'infini par un retour d'état statique. *SIAM J. Control Optim.* 26:251–273 (1988).

[7] W. H. L. Neven and C. Praagman, Column reduction of polynomial matrices. *Linear Algebra Appl.* 188:569–589 (1993).

[8] V. Kučera, *Discrete Linear Control: The Polynomial Equation Approach.* John Wiley and Sons, Chichester, 1979.

[9] V. Kučera, Diophantine equations in control–a survey. *Automatica* 29:1361–1375 (1993).

[10] J. W. Polderman and J. C. Willems, *Introduction to Mathematical Systems Theory: a Behavioral Approach*. Springer-Verlag, 1998.

[11] Th. G. J. Beelen and G. W. Veltkamp, Numerical computation of a coprime factorization of a transfer function matrix. *Systems Control Lett.* 9:281–288 (1987).

[12] E. Frisk, Residual Generation for Fault Diagnosis. Ph.D. Thesis manuscript 716, Linköping University, Sweden, 2001.

[13] J. C. Zúñiga and D. Henrion, A Toeplitz algorithm for the polynomial J-spectral factorization. *Automatica* 42(7):1085–1093 (2006).

[14] M. Grimble and V. Kučera (Eds.), *A polynomial approach to H_2 and H_∞ robust control design*. Springer-Verlag, London, 1996.

[15] P. M. Van Dooren and P. Dewilde, The Eigenstructure of an Arbitrary Polynomial Matrix. Computational Aspects. *Linear Algebra Appl.* 50:545–580 (1983).

[16] A. Varga, Computation of least order solutions of linear rational equations. International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium, 2004.

[17] A. Varga, A Descriptor System Toolbox for Maltab. IEEE International Symposium on Computer Aided Control System Design, Anchorage, Alaska, 2000.

[18] F. R. Gantmacher, *Theory of Matrices I & II*. Chelsea, New York, 1959.

[19] D. Henrion, Reliable Algorithms for Polynomial Matrices. Ph.D. Thesis, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Prague, 1998.

[20] D. Henrion and M. Šebek, Reliable numerical methods for polynomial matrix triangularization. *IEEE Trans. Automat. Control* 44:497–508 (1999).

[21] P. M. Van Dooren, P. Dewilde and J. Vandewalle, On the determination of the Smith-Macmillan form of a rational matrix from its Laurent expansion. *IEEE Trans. Circuit Systems* 26:180–189 (1979).

[22] L. Tan and A. C. Pugh, A novel method to determine the finite and infinite frequency structure of a rational matrix. *IMA J. Math. Control Inform.* 18:129–151 (2001).

[23] J. C. Basilio and M. V. Moreira, A robust solution of the generalized polynomial Bézout identity. *Linear Algebra Appl.* 385:287–303 (2004).

[24] E. N. Antoniou, A. I. G. Vardulakis and S. Vologiannidis, Numerical Computation of Minimal Polynomial Bases: A generalized Resultant Approach. *Linear Algebra Appl.* 405:264–278 (2005).

[25] J. C. Zúñiga and D. Henrion, Block Toeplitz Methods in Polynomial Matrix Computations. International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium, 2004.

[26] J. C. Zúñiga and D. Henrion, On the application of displacement structure methods to obtain null-spaces of polynomial matrices. IEEE Conference on Decision and Control, Paradise Island, Bahamas, 2004.

[27] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[28] P. J. Antsaklis and Z. Gao, Polynomial and rational matrix interpolation: theory and control applications. *Internat. J. Control* 58:349–404 (1993).

[29] A. I. G. Vardulakis, *Linear Multivariable Control. Algebraic Analysis and Synthesis Methods*. Wiley, Chichester, 1991.

[30] D. Henrion and J. C. Zúñiga, Detecting infinite zeros in polynomial matrices. *IEEE Trans. Circuits Systems II*, 52(12):744–745 (2005).

[31] R. R. Bitmead, S. Y. Kung and B. Anderson, Greatest common divisors via generalized Sylvester and Bezout matrices. *IEEE Trans. Automat. Control* 23(6):1043–1047 (1978).

[32] Stefanidis, A. P. Papliński and M. J. Gibbard, Numerical operations with polynomial matrices: Application to multi-variable dynamic compensator design. Lecture Notes in Control and Inform. Sci., 171, Springer Verlag, New York, 1992.

[33] J. C. Zúñiga and D. Henrion, On the application of different numerical methods to obtain null-spaces of polynomial matrices. Part 1 and 2. LAAS-CNRS Research Reports no. 04124 and 04125, Toulouse, France, February 2004.

[34] T. Kailath and A. H. Sayed, *Fast Reliable Algorithms for Matrices with Structure.* SIAM, Philadelphia, 1999.

[35] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, New York, 1996.

[36] G. W. Stewart, *Matrix Algorithms*. SIAM, Philadelphia, 1998.

[37] N. J. Higham, Analysis of the Cholesky decomposition of a semi-definite matrix. *Reliable Numerical Computations, Oxford University Press* 161–185 (1990).

[38] J. C. Zúñiga, Numerical Algorithms for Polynomial Matrices with Applications in Control. Ph.D. Thesis, LAAS-CNRS Toulouse, 2005.

[39] J. C. Zúñiga and D. Henrion, Numerical stability of block Toeplitz algorithms in polynomial matrix computations. IFAC World Congress on Automatic Control, Prague, Czech Republic, 2005.

[40] H. Zha, A component-wise perturbation analysis of the QR decomposition. *SIAM J. Matrix Anal. Appl.* 14(4):1124–1131 (1993).

[41] D. J. Higham and N. J. Higham, Structured backward error and condition of generalized eigenvalue problems. *SIAM Matrix Anal. Appl.* 20(2):493–512 (1998).

[42] A. Edelman and H. Murakami, Polynomial roots from companion matrix eigenvalues. *Math. Comput.* 64:763–776 (1995).

[43] Z. Zeng, Computing multiple roots of inexact polynomials. *Math. Comput.* 64:869–903 (2005).

[44] H. J. Stetter, *Numerical Polynomial Algebra*. SIAM, Philadelphia, 2004.

[45] H. Y. Fan, W. W. Lin and P. Van Dooren, A note on optimal scaling of second order polynomial matrices. *SIAM J. Matrix Anal. Appl.* 26:252–256 (2004).

[46] H. Zhang, Numerical condition of polynomials in different forms. *Electron. Trans. Numer. Anal.* 12:66–87 (2001).

[47] T. Beelen and P. M. Van Dooren, An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Linear Algebra Appl.* 105:9–65 (1988).

Appendix: Proof of Theorem 2

Dimensions of the analysed constant block Toeplitz matrices at each step are as follows:

- At the first step of the algorithm, we apply an LQ factorization over the matrix T_1 of dimension $\bar{m} \times \bar{n}$ where $\bar{m} = n(d+1)$ and $\bar{n} = n$. It is clear that $\bar{m} \geq \bar{n}$.

- At the second step, the analysed matrix T_2'' has dimensions $\bar{m} = n(d+1)$ and $\bar{n} = r_1 + r_1 - \bar{r}_1$. Now, if $\bar{m} < \bar{n}$, then $n(d+1) < 2r_1 - \bar{r}_1$. This implies that $\bar{r}_1 < 0$ for $d \geq 1$ which is impossible, so, $\bar{m} \geq \bar{n}$.

- At step i, for $i > 2$, the analysed matrix T_i'' has dimensions $\bar{m} = n(d+1)$ and $\bar{n} = r_1 + R_{i-1} - \bar{R}_{i-1}$, where $R_i = r_1 + \cdots + r_i$ and $\bar{R}_i = \bar{r}_1 + \cdots + \bar{r}_i$. If $\bar{m} < \bar{n}$, then

$$n(d+1) < r_1 + R_{i-1} - \bar{R}_{i-1} \tag{18}$$

  which is possible in some cases. In the remainder of this appendix we consider that the number f of steps performed by the algorithm is decomposed as $f = p + q$ where p is such that $\bar{m} < \bar{n}$ for $i > p$, and $\bar{m} \geq \bar{n}$ for $i \leq p$

The LQ factorization of a $\bar{m} \times \bar{n}$ matrix with $\bar{m} \geq \bar{n}$ involves $2\bar{m}\bar{n}^2 - \frac{2}{3}\bar{n}^3$ flops when orthogonal matrix Q is not updated. We start with this analysis supposing that we only require the structural indices and not also the corresponding vectors. For simplicity we reduce last expression as $2\bar{m}\bar{n}^2$, we consider the worst case, i.e. $\bar{n} = n + (i-1)n - (i-1) = in - (i-1)$ and we also approximate $\bar{m} = n(d+1) \approx nd$. These simplifications do not affect our analysis since we search for an upper bound on k, the total number of flops.

At step 1, the algorithm performs about $k_1 = 2nd(n^2)$ operations. At step 2, the algorithm performs about $k_2 = 2nd(2n-1)^2$ operations. Similarly, at step i for $i \leq p$ the algorithm performs about $k_i = 2nd[in - (i-1)]^2$ operations. At step $p+1$ dimensions of the analysed Toeplitz matrix: $\bar{m} = n(d+1)$ and $\bar{n} = (p+1)n - p$, are such that $\bar{m} < \bar{n}$. So, we consider that the algorithm performs about $k_p = 2n^2d^2[(p+1)n - p]$ operations. In general, at step i for $i > p$, the algorithm performs about $k_i = 2n^2d^2[in - (i-1)]$ operations.

In conclusion, the total amount of operations performed by the algorithm is about $\overline{k} = nd(g_1n^2 - g_2n + g_3) + n^2d^2(g_4n - g_5)$ where

$$\begin{array}{lcl} g_1 & = & 2\sum_{i=1}^{p} i^2 = \frac{1}{3}p(p+1)(2p+1) \\ g_2 & = & 4\sum_{i=1}^{p} i^2 - 4\sum_{i=1}^{p} i = \frac{2}{3}p(p+1)(2p+1) - 2p(p+1) \\ g_3 & = & 2\sum_{i=1}^{p} i^2 - 4\sum_{i=1}^{p} i + 2\sum_{i=1}^{p} 1 = \frac{1}{3}p(p+1)(2p+1) - 2p(p+1) + 2p \\ g_4 & = & 2\sum_{i=p+1}^{p+q} i = q(q+2p+1) \\ g_5 & = & 2\sum_{i=p+1}^{p+q} i - \sum_{i=p+1}^{p+q} 1 = q(q+2p-1) \end{array}$$

It is clear that $g_1n^2 \geq g_1n^2 - g_2n + g_3$ and $g_4n \geq g_4n - g_5$, so we can write $\overline{k} = g_1n^3d + g_4n^3d^2$. To obtain the expected result we simply consider that $\overline{m} \geq \overline{n}$ for all the steps, so that the algorithm performs about $\hat{k} = (g_1 + g_6)n^3d$ operations, where

$$g_6 = 2\sum_{i=p+1}^{p+q} i^2 = \frac{1}{3}q[6p(p+q+1) + q(2q+3) + 1].$$

Now, for $\hat{k}$ to be an upper bound of the total number of operations, we require that $\overline{k} \leq \hat{k}$, i.e. $g_6 \geq g_4d$. This implies the following relation

$$\frac{1}{3}q[6p(p+q+1) + q(2q+3) + 1] \geq dq(2p+q+1). \tag{19}$$

In the worst case ($r_i = n$, $\overline{r_i} = 1$) equation (18) yields $n(d+1) < (p+1)n - p$. So we can conclude that p is always greater than d, and so that $pq(2p+2q+2) > dq(2p+q+1)$. Therefore, inequality (19) is always true and $k \leq \hat{k} = Cn^3d$ where $C = g_1 + g_6 = \frac{1}{3}f(f+1)(2f+1)$.

Now we consider the case when the computed null-space vectors are also required. The LQ factorization of a $\overline{m} \times \overline{n}$ matrix with $\overline{m} \geq \overline{n}$ performs now $4\overline{m}^2\overline{n} - 2\overline{m}\overline{n}^2 + \frac{2}{3}\overline{n}^3$ operations. It is clear that $4\overline{m}^2\overline{n} \geq 4\overline{m}^2\overline{n} - 2\overline{m}\overline{n}^2 + \frac{2}{3}\overline{n}^3$, so for simplicity we reduce the complexity expression as $4\overline{m}^2\overline{n}$, we consider the worst case, i.e. $\overline{n} = n + (i-1)n - (i-1) = in - (i-1)$ and we also approximate $\overline{m} = n(d+1) \approx nd$. These simplifications do not affect our analysis since we search for an upper bound on the total number of flops.

Following the same reasoning that above, we obtain that the total amount of operations performed by the algorithm is about $\overline{k} = g_7n^3d^2 + 2g_6n^3d$ where

$$g_7 = 4\sum_{i=1}^{p} i = 2p(p+1).$$

Now, since (19) is true, notice that $\overline{k} > (g_7 + 2g_4)n^3d^2$, and $\overline{m}$ cannot be considered greater than $\overline{n}$ for all steps. So, we explore the other possibility, namely, when $\overline{m} < \overline{n}$ for all the steps, and we take $\hat{k} = 2(g_1 + g_6)n^3d$. In this case we require that $g_7d \leq 2g_1$ for $\hat{k}$ to be an upper bound, this implies that $2dp(p+1) \leq \frac{2}{3}p(p+1)(2p+1)$. This last inequality is not always valid, in particular it is not verified when p is near to $d > 2$. So, to ensure we have an upper bound, we simply take $\hat{k} = 3(g_1 + g_6)n^3d$ from which it follows that $g_7d \leq 3g_1$. Then, $k \leq \hat{k} = 3Cn^3d$ where $C = g_1 + g_6 = \frac{1}{3}f(f+1)(2f+1)$.

So, the proof is complet, the complexity of our algorithm is in $O(f^3n^3d)$.