BLOCK TOEPLITZ ALGORITHMS FOR POLYNOMIAL MATRIX NULL-SPACE COMPUTATION

JUAN CARLOS ZÚÑIGA* AND DIDIER HENRION†

Abstract. In this paper we present new algorithms to compute the minimal basis of the null-space of polynomial matrices, a problem which has many applications in control. These algorithms take advantage of the block Toeplitz structure of the Sylvester matrix associated with the polynomial matrix. The analysis done in both aspects, algorithmic complexity and numerical stability, shows that the algorithms are reliable and can be considered as an alternative to the well-known pencil (state-space) algorithms found in the current literature. Some numerical examples are also presented to illustrate our results.

Key words. Polynomial matrices, Toeplitz 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 basis of the *null-space* of an arbitrary $m \times n$ polynomial matrix

$$A(s) = A_0 + A_1 s + A_2 s^2 + \cdots + A_d s^d. \tag{1.1}$$

of degree d and rank $\rho \leq \min(m, n)$. The right null-space of $A(s)$ is the set of non-zero polynomial vectors $z(s)$ such that

$$A(s)z(s) = 0. \tag{1.2}$$

A basis of the right null-space of $A(s)$ is formed by any set of $n-\rho$ linearly independent vectors satisfying (1.2). We gather these vectors as the columns of matrix $Z(s)$ such that $A(s)Z(s) = 0$. Let δ_i for $i = 1, 2, \ldots, n-\rho$ be the degree of each vector in the basis. If the sum of all the degrees δ_i is minimal then we have a *minimal basis* in the sense of Forney [7]. 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 analyze only the right null-space of $A(s)$.

Computing the null-space of a polynomial matrix has several applications in control theory. As a first example, the structure of a linear system represented by the state space matrices (A, B, C, D) can be recovered from the eigenstructure of a polynomial matrix [17]. This structural information is important in problems such as decoupling [38]. In particular, the degrees of the vectors in a minimal basis of the null-space of the pencil

$$P(s) = \begin{bmatrix} sI - A & B \\ C & D \end{bmatrix} \tag{1.3}$$

which are defined as the Kronecker invariant indices [32], correspond to the invariant lists I_2 and I_3 defined in [25]. These invariants are key information when solving problems of structural modification for linear systems [24].

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

†LAAS-CNRS, and Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 16627 Prague, Czech Republic. Corresponding author. Tel: +33 561 33 63 08. Fax: +33 561 33 69 69. e-mail: henrion@laas.fr

Computing the null-space is also critical when solving the problem of column reduction of a polynomial matrix [26]. Column reduction is the initial step in several elaborated algorithms. Column reducedness of polynomial matrices is a property often required in computer aided control system design [17].

With the polynomial control theory approach introduced by Kučera [21], the solution of several control problems has been reformulated in terms of polynomial matrix equations or Diophantine equations [22]. Consider, for instance, the typical polynomial matrix equation $A(s)X(s) = B(s)$. Solving this kind of equations requires a direct manipulation of polynomial matrices. By analogy with the constant case, the null space of $\begin{bmatrix} A(s) & -B(s) \end{bmatrix}$ contains key information about the existence and the uniqueness of solution $X(s)$. Other operations often required in the polynomial approach are easily solved by computing the null-space of a suitable polynomial matrix. Consider for example a linear multi-variable system represented by a left coprime matrix fraction description $T(s) = D_L^{-1}(s)N_L(s)$. It can be shown [17, 4] that a right coprime factorization $T(s) = N_R(s)D_R^{-1}(s)$ can be obtained via the null-space computation:

$$\begin{bmatrix} D_L(s) & -N_L(s) \end{bmatrix} \begin{bmatrix} N_R(s) \\ D_R(s) \end{bmatrix} = 0.$$

It is also possible to obtain the transfer function of a linear system by computing the null-space of a polynomial matrix. Consider a single-input single-output system governed by the second-order differential equation

$$M\frac{\mathrm{d}^2}{\mathrm{d}t^2}x + Kx = Bu, \tag{1.4}$$

and let $y = Cx$ be the output. The Laplace transform of equation (1.4) is given by $D(s)x = Bu$, where $D(s) = Ms^2 + K$. So, the system transfer function is given by

$$G(s) = CD^{-1}(s)B = C\frac{N(s)}{d(s)} = \frac{n(s)}{d(s)},$$

from which it follows that

$$D^{-1}(s)B = \frac{N(s)}{d(s)}, \quad D(s)N(s) = Bd(s).$$

From the vector of minimal degree of the null-space of $\begin{bmatrix} D(s) & -B \end{bmatrix}$, we obtain the numerator $n(s)$ and the denominator $d(s)$ of the transfer function as follows:

$$\begin{bmatrix} D(s) & -B \end{bmatrix} \begin{bmatrix} \times & \times & \cdots & n(s) & d(s) \end{bmatrix}^T = 0. \tag{1.5}$$

Several physical systems can be modeled as above: connected mass-spring systems, earthquake damping systems in a building, or vibrating rods [31].

In fault diagnostics the residual generator problem can be transformed into the problem of finding the null-space of a polynomial matrix [8]. The problem is formulated as follows. Consider the perturbed system

$$y = G(s)u + H(s)d + L(s)f$$

where $d(t)$ are the unknown disturbances and $f(t)$ are the monitored faults. It is shown that the residual generator $Q(s)$ belongs to the following left null-space

$$Q(s)\begin{bmatrix} G(s) & H(s) \\ I & 0 \end{bmatrix} = Q(s)M(s) = 0.$$

There are other applications for the null-space of polynomial matrices. In [41] we present an algorithm for J-spectral factorization of a polynomial matrix. When the analyzed polynomial matrix is singular, we need to extract 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 problems [11].

1.2. Previous works and algorithms. Since the null-space is a part of the eigenstructure, its computation is often related with the computation of the whole eigenstructure of a polynomial matrix. Computational algorithms for the eigenstructure of polynomial matrices appear early in the 1970s. Most of the researchers take as a general approach the *linearization* of the analyzed polynomial matrix. The idea comes from control theory where the structure of a linear multi-variable system represented by a rational transfer function $C(sI-A)^{-1}B+D$ is captured by the structure of the pencil (1.3) [17]. In [32] it is showed 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 null-space basis, are contained in its Kronecker canonical form, and reliable algorithms to compute this canonical form are developed. In [34] it is proven that the structural indices of an arbitrary polynomial matrix $A(s) = A_0 + A_1 s + A_2 s^2 + \cdots + A_d s^d$ can be recovered from the structure of the pencil

$$sB_0 - A_0 = \begin{bmatrix} I & & & sA_d \\ -sI & \ddots & & \vdots \\ & \ddots & I & sA_2 \\ & & -sI & sA_1 + A_0 \end{bmatrix}. \tag{1.6}$$

So, reliable algorithms to obtain the eigenstructure of $A(s)$ were presented. Seminal works of Van Dooren [32, 34] 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 obtain not only the degrees but also the vectors of the null-space of $A(s)$ are developed in [4]. First, pencil (1.6) is transformed in the generalized Schur form

$$U(sB_0 - A_0)V = \begin{bmatrix} sB_z - A_z & * \\ 0 & sB_* - A_* \end{bmatrix} \tag{1.7}$$

by the methods explained in [32]. Here $sB_z - A_z$ is in upper staircase form containing only the infinite and null-space structural indices of $A(s)$. To obtain also the vectors of the minimal basis of $A(s)$, a post-processing of $sB_z - A_z$ is needed. First, a minimal basis $Z_z(s)$ of $sB_z - A_z$ is computed with the recursive process presented in Section 4 of [4]. Then, it is showed that the n bottom rows of $V[\ Z_z^T(s)\ \ 0\ ]^T$ are the searched basis. It is important to say that the process presented in [4] is usually computationally expensive and its numerical stability is not guaranteed. Another pencil algorithm was recently presented in [37]. This algorithm is based on descriptor system techniques. First, a minimal descriptor realization of $A(s)$ of the form

$$\begin{aligned} E\dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1.8}$$

is obtained. Then, using the methods in [32], the pencil

$$P(s) = \begin{bmatrix} A - sE & B \\ C & D \end{bmatrix}$$

is reduced to the staircase form

$$UP(s)V = \begin{bmatrix} B' & A' - sE' & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}. \tag{1.9}$$

Finally, the n bottom rows of

$$V \begin{bmatrix} I \\ (sE' - A')^{-1}B' \\ 0 \\ 0 \end{bmatrix}$$

is the searched basis. Reliable algorithms to compute the descriptor realization (1.8) are presented in [36], nevertheless, no study of the backward error in the coefficients of $A(s)$ is presented.

The classical numerical methods to manipulate polynomial matrices are based on the use of elementary operations over polynomials. The poor performance of these basic methods was quickly identified [9]. The major criticism relies on the numerical stability, lost because of the pivoting on the powers of the indeterminate s. However, we can also find in the literature some reliable polynomial methods which do not use the linearization of the analyzed matrix [13, 14]. These methods process directly coefficients A_i of $A(s)$ and avoid elementary operations over polynomials. In [33], 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. More recently, Tan [30] presented an algorithm to obtain the finite and infinite structural indices also based on rank computation of successive Toeplitz matrices.

During the writing of this paper, an algorithm to compute polynomial null-spaces was presented in [3][1]. The basic idea of this algorithm is the computation of the rank and null-space of some associated block Toeplitz matrices. The singular value decomposition is used to compute these ranks and null-spaces. We used the same numerical method in [40] to compute the infinite structural indices. In [43, 44] we investigated the reliability of different numerical methods like the QR factorization, the Column Echelon form and also some displacement structure methods based on the generalized Schur algorithm [18].

In this paper we review these previous results and we develop a new blocked algorithm to compute the basis of null-spaces of polynomial matrices which is faster than the algorithm in [3]. Moreover, here we present a full analysis in two aspects, numerical stability and algorithmic complexity. In the following sections, we also show the relations between our algorithm and the existing pencil algorithms. We show how our algorithm can be extended to obtain the whole eigenstructure of $A(s)$, and we sketch some comparisons with the pencil algorithms.

[1] In this paper another application for the null-space of polynomial matrices is presented: the solution of the generalized polynomial Bézout identity.

1.3. Outline and notations.

- In section 2 we define the whole eigenstructure of a polynomial matrix and we establish some useful relations and results for the analysis in the next sections.
- In section 3 we analyze the problem of solving polynomial null-space extraction (1.2) for a vector $z(s)$ of fixed degree. We divide this problem in two parts, first the derivation of a constant block Toeplitz matrix equation equivalent to (1.2), and second the estimation of the polynomial null-space degree.
- In section 4 we describe the different numerical methods to solve the constant block Toeplitz matrix equation associated with (1.2).
- In section 5 we sketch our algorithm to obtain the minimal basis of the null-space of $A(s)$.
- In section 6 we present a full analysis of the algorithmic complexity and numerical stability of our algorithm. We also present some comparisons with the pencil algorithms and we show how our algorithm can be naturally extended to obtain the whole eigenstructure of $A(s)$.

The following notations are used in this paper. Integer $\delta_{\max}$ is the degree of $Z(s)$, namely the maximum degree of any vector in a minimal basis of the null-space of $A(s)$. Real ϵ is the machine precision in the floating point arithmetic system. $O(\epsilon)$ is a constant depending on ϵ, of the same order of magnitude.

In this paper we sketch comparisons with the pencil algorithms. Nevertheless, the analysis of these algorithms is not our purpose. For the sake of clarity we explain the basic ideas of these algorithms in section 1.2, and some other aspects are mentioned when needed. Readers are refered to [32, 4, 34, 37] for a full description of these pencil algorithms.

2. Preliminaries. In this section we present some definitions and results that we use in the remainder of the paper for the analysis of our algorithms. We start with the infinite and finite structures of $A(s)$. Together with the null-space structure, they form the *eigenstructure* of $A(s)$.

A finite zero of $A(s)$ is a complex number α such that there exists a non-zero complex vector v satisfying $A(\alpha)v = 0$. Vector v is called characteristic vector or eigenvector associated to the finite zero α.

From Theorem A.1 in [1], 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

$$\begin{bmatrix} \bar{A}_0 & & \\ \vdots & \ddots & \\ \bar{A}_{k_i-1} & \cdots & \bar{A}_0 \end{bmatrix} \begin{bmatrix} v_{i1} \\ \vdots \\ v_{ik_i} \end{bmatrix} = 0 \tag{2.1}$$

with $v_{11}, v_{21}, \ldots, v_{m_g 1}$ 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 characteristic vectors 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 [17]

$$\bar{A}(s) = A_d + A_{d-1}s + \cdots + A_0 s^d.$$

So, if $\alpha = 0$ has algebraic multiplicity $\bar{m}_a$ and geometric multiplicity $\bar{m}_g$, 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

$$\begin{bmatrix} A_d & & \\ \vdots & \ddots & \\ A_{d-k_i+1} & \cdots & A_d \end{bmatrix} \begin{bmatrix} \bar{v}_{i1} \\ \vdots \\ \bar{v}_{i\bar{k}_i} \end{bmatrix} = 0 \tag{2.2}$$

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 characteristic vectors associated to the infinity. The orders of the zeros or poles at infinity, as they appear in the Smith-McMillan form at infinity, depend on integers $\bar{k}_i$ [33]. 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.

THEOREM 2.1. *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, γ_l and γ_r the sum of the degrees of all the vectors in a minimal basis of the left and right null-spaces respectively. Then,*

$$\rho d = n_f + \bar{m}_A + \gamma_r + \gamma_l \tag{2.3}$$

Proof. The proof is derived from section 3.6 in [35] using the definition of zeros at infinity given above. □

3. Block Toeplitz Approach. Our approach to obtain $Z(s)$, a minimal basis of the null-space of $A(s)$, is divided in two parts. First, the derivation of a constant (non-polynomial) equation related to (1.2), and second, the estimation of the dimension of this equation, i.e. the estimation of the minimal degrees δ_i.

3.1. The Sylvester equation. Taking $A(s)$ in the form (1.1) and $z(s) = z_0 + z_1 s + z_2 s^2 + \cdots + z_\delta s^\delta$, we can see that polynomial equation (1.2) is equivalent to the constant linear system of equations

$$\begin{bmatrix} A_d & & \\ \vdots & \ddots & \\ A_0 & & A_d \\ & \ddots & \vdots \\ & & A_0 \end{bmatrix} \begin{bmatrix} z_\delta \\ z_{\delta-1} \\ \vdots \\ z_0 \end{bmatrix} = \mathcal{A}_{\delta+1} \mathcal{Z}_\delta = 0. \tag{3.1}$$

Matrix $\mathcal{A}_{\delta+1}$ is called the Sylvester matrix of degree $\delta+1$ associated to $A(s)$. Sylvester matrices arise in several control problems [2, 3, 17, 27]. In this case, the approach allows to obtain $Z(s)$ by computing the constant null-space $\mathcal{A}_{\delta+1}$. Matrix $\mathcal{A}_{\delta+1}$ is in block Toeplitz form, so we call our algorithms *block Toeplitz algorithms*.

To compute the null-space $\mathcal{A}_{\delta+1}$, several numerically reliable methods can be used. We briefly describe some of these methods in section 4.

3.2. Estimating the degree. The problem remaining when solving (3.1) is the estimation of the degree δ. We can process iteratively block Toeplitz matrices $\mathcal{A}_i$ of dimensions $m(d+i) \times ni$ at index $i = 1, 2, \ldots$ If $\mathcal{A}_i$ has full rank, then matrix $A(s)$ has no vectors of degree $i-1$ in the minimal basis of its null-space.

THEOREM 3.1. *Let the nullity of $\mathcal{A}_i$ be $\gamma_i = in - r_i$ with $r_i = \operatorname{rank} \mathcal{A}_i$. Then $Z(s)$ has $\gamma_i - 2\gamma_{i-1} + \gamma_{i-2}$ columns of degree $i-1$ where $\gamma_j = 0$ for $j \leq 0$. When $\gamma_f - \gamma_{f-1} = n - \operatorname{rank} A(s)$ for some sufficiently large index f, we have obtained all the columns in $Z(s)$.*

Proof. The proof is constructive. Suppose that $\operatorname{rank} \mathcal{A}_1 = r_1$ and then $\gamma_1 = n - r_1$ is the number of linearly dependent columns in $\mathcal{A}_1$. Suppose that the ith column of $\mathcal{A}_1$ is dependent, so the ith column of $A(s)$ is a combination of degree 0 of the other columns. Namely, we have a column $z_1(s)$ of degree 0 in $Z(s)$.

Now, because of the block Toeplitz structure, at least the ith and the $(i+n)$th columns of $\mathcal{A}_2$ are dependent, so $\gamma_2 = 2n - r_2 = 2\gamma_1 + x_2$ where x_2 is the number of new linearly dependent columns. This result implies that the ith column of $A(s)$ is also a combination of degree 1 of the other columns. Indeed, if $z_1(s)$ belongs to the null-space of $A(s)$, so does $sz_1(s)$. Nevertheless, the minimal basis only contains $z_1(s)$. On the other hand, the x_2 new dependent columns found in $\mathcal{A}_2$ imply that the corresponding columns in $A(s)$ –distinct from the ith column– are a combination of degree 1 from the other columns. So, the number of columns of degree 1 in $Z(s)$ is $x_2 = \gamma_2 - 2\gamma_1$. Notice that $\gamma_2 - \gamma_1 = x_2 + \gamma_1$ is the number of columns in $Z(s)$ obtained so far.

Analyzing $\mathcal{A}_3$ we see that $\gamma_3 = 3n - r_3 = 3\gamma_1 + 2x_2 + x_3$ where x_3 is the number of columns of $A(s)$ which are combinations of degree 2 of the others. So, we have $x_3 = \gamma_3 - 2\gamma_2 + \gamma_1$ columns of degree 2 in $Z(s)$. Now, $\gamma_3 - \gamma_2 = x_3 + x_2 + \gamma_1$ is the number of columns in $Z(s)$ obtained so far.

Finally, we derive that $x_i = \gamma_i - 2\gamma_{i-1} + \gamma_{i-2}$, which is the required result. Moreover, we can see that $\gamma_i - \gamma_{i-1} = \gamma_1 + x_2 + \cdots + x_i$ is the number of columns of $Z(s)$ obtained so far. For some sufficiently large index f it holds $\gamma_f - \gamma_{f-1} = n - \operatorname{rank} A(s)$. ∎

Index f in Theorem 3.1 is equal to $\delta_{\max} + 1$ and depends only on the structure of $A(s)$ and not on the algorithm. From Theorem 2.1 we can easily find an upper bound for $\delta_{\max}$. From (2.3), we can see that γ_r (or γ_l) can, at most, be equal to ρd, then $\delta_{\max} \leq \rho d$. Nevertheless, notice that if rank $A_d < \rho$ there are at least $\rho - \operatorname{rank} A_d$ zeros at infinity, then $\delta_{\max} \leq \rho(d-1) + \operatorname{rank} A_d$. From Lemma 4.1 in [13] a tight upper bound for $\delta_{\max}$ can be found independently of ρ. It can be proven that

$$\delta_{\max} \leq \sum_{i=1}^{n} \deg(\operatorname{col}_i A(s)) - \min_i \deg(\operatorname{col}_i A(s)) = \bar{\delta}, \tag{3.2}$$

with $\deg(\operatorname{col}_i A(s))$ denoting the degree of the ith column of $A(s)$.

There are at least three different strategies to analyze block Toeplitz matrices $\mathcal{A}_i$:

1. If we analyze matrix $\mathcal{A}_i$ increasing i by one at each step, we can find $Z(s)$ in at most $\bar{\delta} + 1$ steps. This implies analyzing a block Toeplitz matrix of dimension $m(d+1) \times n$ at step 1, of dimension $m(d+2) \times 2n$ at step 2, and so on. In the worst case, a Toeplitz matrix of dimension $m(d + \bar{\delta} + 1) \times (\bar{\delta} + 1)n$ could be analyzed at the last step.
2. We can obtain $Z(s)$ in only one step by analyzing $\mathcal{A}_{\bar{\delta}+1}$, which can be computationaly expensive if $\bar{\delta}$ is large.
3. We can increase i by a fixed ratio of $\bar{\delta}$ at each step.

It is our experience that there is no policy outperforming the others systematically, see [43, 44] for more details. In any case, it is important to notice that the algorithm stops, in the worst case, in $\bar{\delta}+1$ steps, and that the dimension of the analyzed Toeplitz matrices depends only on this limit.

4. Numerical Methods. In the previous section we explained that we can obtain $Z(s)$, a minimal basis of the null-space of a polynomial matrix $A(s)$, by obtaining the rank and the constant null-spaces of some block Toeplitz matrices. To compute these constant null-spaces several numerically reliable methods can be used: the singular value decomposition (SVD), Gaussian elimination, the QR factorization with column pivoting or other *rank revealing methods* (RRM). All these numerical methods are well studied in the literature [10] and their favorable numerical properties are proven [16, 29]. Fore the sake of clarity, in this section we briefly review some facts on these methods.

4.1. Singular Value Decomposition (SVD). The SVD $\Sigma = P^T M Q$ of a constant $m \times n$ matrix M defined in [10] can be implemented as a backward stable algorithm[2]. If $\hat{P}$, $\hat{Q}$ and $\hat{\Sigma}$ are the computed matrices, it can be shown [29] that $\hat{\Sigma} = \hat{P}^T(M + \Delta)\hat{Q}$ where $\|\Delta\|_2 \leq O(\epsilon)\|M\|_2$. Moreover, if $\hat{\sigma}_i$ are the computed singular values of M, then $|\sigma_i - \hat{\sigma}_i| \leq O(\epsilon)\|M\|_2$.

So, we can expect to recover the rank ρ of M by applying the SVD and counting the number of computed singular values less than or equal to $\mu\|M\|_2$, where μ is a tolerance depending on ϵ. If rank $M = \rho$, then $\|M\hat{Q}(:, \rho + 1 : n)\|_2$ is suitably small. So, matrix $\hat{Q}(:, \rho + 1 : n)$ is a basis of the null-space of M.

The SVD is an useful method to obtain the numerical rank of a constant matrix, but it has two drawbacks. First, it is relatively computationally expensive: it requires approximately $4m^2n + 8mn^2 + 9n^3$ elementary operations (flops) [10]. Second, it does not accept updating, namely, if a row or column is added to M, matrices P and A can not be updated easily.

4.2. LQ factorization. Another RRM that is easier to compute than the SVD is the LQ factorization. The LQ factorization $L = MQ$ of matrix M is the dual of its QR factorization defined in [10]. In this case, matrix L is in lower triangular form. The number of flops required by the LQ factorization is approximately $4(m^2n - mn^2 + \frac{1}{3}n^3)$ [10].

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 M. To choose the row at step j, several strategies can be followed. For instance, we can compute $\|M(i, j + 1 : n)\|$ for all rows i not yet analyzed, and choose the row with the largest norm. In a simpler way, we can choose the first row with non-zero entry j. Notice that row pivoting allows to know the position of the linearly independent rows of M. So, a row permutation matrix P can be applied to put L in the form

$$PL = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \tag{4.1}$$

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

The LQ factorization is backward stable. It can be shown [16, 29] that the computed matrices $\hat{L}$ and $\hat{Q}$ satisfy $(M+\Delta)\hat{Q} = \hat{L}$ with $\|\Delta\|_2 \leq O(\epsilon)\|M\|_2$. Moreover, if we partition L as in (4.1), then (to first order),

$$\frac{\|\hat{L}_{22} - L_{22}\|_2}{\|M\|_2} \leq \frac{\|\Delta\|_2}{\|M\|_2}(1 + \|L_{11}^{-1}L_{21}^T\|_2) \tag{4.2}$$

[2]The implementation of the SVD in a reliable computational algorithm is not easy. There are several ways to do it, but this analysis is out of the scope of this paper. For more details see [10].

where $\|L_{11}^{-1}L_{21}^T\|_2$, the conditioning of the LQ factorization for rank-deficient matrices, is usually small [15].

So, from equation (4.2) 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 $\mathrm{rank}\, M = \rho$, then $\|M\hat{Q}(:, \rho+1 : n)\|_2$ is suitably small. So, matrix $\hat{Q}(:, \rho+1 : n)$ is a basis of the null-space of M. The problem with the LQ factorization as a RRM 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 [10].

4.3. *L*-method. Now suppose we want to reduce even more the algorithmic complexity. One possibility is to compute the LQ factorization without storing matrix Q, which contains the successive Householder transformations H_i applied to matrix M. Computation of matrix L only requires $2(mn^2 - \frac{1}{3}n^3)$ flops [10]. In this case we cannot recover an orthogonal basis of the right null-space of M. However, from L we can compute a natural basis of its left null-space by solving suitable triangular systems of equations. For example suppose that for a given matrix M we obtain

$$L = MH_1H_2H_3 = \begin{bmatrix} \times_{11} & 0 & 0 & 0 \\ \times_{21} & \times_{22} & 0 & 0 \\ \times_{31} & 0 & 0 & 0 \\ \times_{41} & \times_{42} & \times_{43} & 0 \\ \times_{51} & \times_{52} & \times_{53} & 0 \end{bmatrix}.$$

This means that the third and the fifth rows of M are linearly dependent. From L we can see that the coefficients of linear combinations for the fifth row of M can be recovered by solving, via a back-substitution process, the linear triangular system

$$\begin{bmatrix} k_1 & k_2 & k_4 \end{bmatrix} \begin{bmatrix} \times_{11} & 0 & 0 \\ \times_{21} & \times_{22} & 0 \\ \times_{41} & \times_{42} & \times_{43} \end{bmatrix} = -\begin{bmatrix} \times_{51} & \times_{52} & \times_{53} \end{bmatrix}, \tag{4.3}$$

and then

$$\begin{bmatrix} k_1 & k_2 & 0 & k_4 & 1 \end{bmatrix} M = 0.$$

Back-substitution is a backward stable method [16], so the quality of the solution of (4.3) is as good as the quality of the computed factor $\hat{L}$.

Notice that with a suitable row permutation matrix P, factor L can be put in a Column Echelon Form (CEF). For this reason in previous works [43, 44] we refer to the L-factor as the CEF of M. The LQ factorization of M^T is dually the QR factorization of M. Then, in a dual form we can define the R-method to obtain a natural basis of the right null-space of M.

4.4. Displacement structure methods. The Toeplitz structure of matrices $\mathcal{A}_i$ in Theorem 3.1 allows the application of *fast* versions of the LQ factorization or the L-method described above. Fast methods are based on the displacement structure theory [18]. The relevance of fast methods is related with a gain of one order of magnitude in the algorithmic complexity when factorizing structured matrices. This feature of the fast methods can be very convincing. Nevertheless, other aspects must be considered. First, backward stability of the fast methods is not guaranteed. Second, significant differences between execution times of the fast and classical methods only arise when the analyzed matrix is large, and *large* depends on the problem context.

Here we want to factorize an $mk \times nl$ block Toeplitz matrix

$$\left[\begin{array}{ccc|ccc} T_0 & \cdots & T_{r-1} & T_r & \cdots & T_{l-1} \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ T_{-r+1} & \cdots & T_0 & T_1 & \cdots & T_{l-r} \\ \hline T_{-r} & \cdots & T_{-1} & T_0 & \cdots & T_{l-r-1} \\ \vdots & & \vdots & \vdots & & \vdots \\ T_{-k+1} & \cdots & T_{-k+r} & T_{-k+r+1} & \cdots & T_{-k+l} \end{array}\right]$$

with blocks T_i of dimension $m \times n$ and where $r = \min(k, l)$. Then, a block formulation of the displacement structure theory is required [20, 19]. The problem with this block formulation is that it is not always possible to determine how large the analyzed matrix must be so that the fast methods perform faster than the classical methods. This problem is illustrated in [20] where fast methods are used to obtain the QR factorization of a block Toeplitz matrix M for different values of n, m, k and l. In our previous work [43, 44] we apply fast factorization methods to matrices $\mathcal{A}_i$ in order to obtain a basis of the null-space of $A(s)$. We show that the fast methods are more efficient than the classical methods when nl is very large. Nevertheless, the performance is not the same when n is larger than l or vice versa. We also notice that for a given matrix $A(s)$, nl is not necessarily large enough since l is bounded by $\delta_{\max}$, the degree of a minimal basis of the null-space of $A(s)$.

5. Algorithms. An algorithm to obtain a minimal degree polynomial matrix $Z(s)$ such that $A(s)Z(s) = 0$ was sketched in the proof of Theorem 3.1. There can be several implementations of this algorithm. This depends on the degree search strategy (see Section 3), and the RRM chosen to compute the null-spaces of the related block Toeplitz matrices $\mathcal{A}_i$ (see Section 4).

A first option is to find $Z(s)$ by obtaining only the null-space of $\mathcal{A}_{\bar{\delta}}$, where $\bar{\delta}$, given by (3.2), is the maximum expected degree of $Z(s)$. We can do that with the R-method described in the previous section. Column pivoting yields the position of the linearly independent columns in $\mathcal{A}_{\bar{\delta}}$, so we can know the rank of matrices $\mathcal{A}_i$ for $i = 1, 2, \ldots, \bar{\delta}$ which are the block columns of $\mathcal{A}_{\bar{\delta}}$ and then, we can apply Theorem 3.1. Finally, the vectors in a minimal basis can be computed by finding the linear combination of the dependent columns of $\mathcal{A}_{\bar{\delta}}$ as in (4.3). This is the simplest algorithm to find $Z(s)$ but if $\delta_{\max}$, the actual degree of $Z(s)$, is significantly smaller than $\bar{\delta}$, then the number of operations required in the computation of the R-factor of $\mathcal{A}_{\bar{\delta}}$ could be larger than necessary.

Another strategy is to analyze matrices $\mathcal{A}_i$ of increasing dimensions at each step. As explained in Section 3.2, we can increase index i by a fixed ratio c of $\bar{\delta}$. The idea is to obtain the R-factor of $\mathcal{A}_{i-1+c\bar{\delta}}$ at step i until we find all the degrees of the vectors in the minimal basis of the null-space of $A(s)$. Finally, $Z(s)$ can be obtained from the last computed R-factor. This algorithm is more efficient than the previous one when $\delta_{\max}$ is significantly smaller than $\bar{\delta}$. Nevertheless, we have no method to determine the optimal ratio c. As we showed in [43, 44] this turns out to be a critical point in many cases.

In this paper we analyze in detail an algorithm for which we increase i by one at each step. In this case it is not necessary to know the position of the linearly independent columns of $\mathcal{A}_i$. In fact it is more efficient to obtain the rank of $\mathcal{A}_i$ and a basis of its null-space with the same RRM, so we use the LQ factorization or the SVD. The generic algorithm can be described as follows:

- *Step i*:
  With an RRM obtain the rank and the nullity γ_i of $\mathcal{A}_i$. If $k = \gamma_i - 2\gamma_{i-1} + \gamma_{i-2} > 0$, then, from the γ_i rightmost columns of the orthogonal matrix Q, insert the k vectors of degree $i-1$ as the rightmost columns in $Z(s)$.
- *Stopping rule*:
  If $\gamma_i - \gamma_{i-1} = n - \text{rank } A(s)$, then stop. We have in $Z(s)$ a minimal basis of the null-space of $A(s)$.

The above algorithm is the one proposed in [3]. As an RRM we can use the SVD but the computational effort to obtain orthogonal matrix P is wasted because P is never used. On the other hand, we can use the LQ factorization without sacrificing significantly the accuracy [10, 16, 29] but performing less operations. Now we show how the block structure of matrices $\mathcal{A}_i$ can be used to improve this algorithm by reducing even more the number of operations. We show that the rank and the nullity of $\mathcal{A}_{i-1}$ can be used to process $\mathcal{A}_i$. Consider for example

$$\mathcal{A}_1 = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}$$

corresponding to a 2×3 polynomial matrix $A(s)$ of degree 1, and suppose that with an LQ factorization we obtain an orthogonal matrix Q_1 such that

$$\mathcal{A}_1 Q_1 = \begin{bmatrix} \# & 0 & 0 \\ \# & 0 & 0 \\ \# & \# & 0 \\ \# & \# & 0 \end{bmatrix}.$$

Column $Q_1(:,3)$ is a basis of the null-space of $\mathcal{A}_1$, and the columns of $Q_1(:,2:3)$ generate a basis of the null-space of $\mathcal{A}_1(1:2,:)$. Notice also that

$$\mathcal{A}_2 \begin{bmatrix} Q_1 & 0 \\ 0 & I_n \end{bmatrix} = \begin{bmatrix} \# & 0 & 0 & 0 & 0 & 0 \\ \# & 0 & 0 & 0 & 0 & 0 \\ \# & \# & 0 & \times & \times & \times \\ \# & \# & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} = \bar{\mathcal{A}}_2,$$

so we can apply the RRM only to the sub-matrix $\bar{\mathcal{A}}_2(3:6,2:6)$. Doing this, suppose that we obtain

$$\bar{\mathcal{A}}_2(3:6,2:6)\bar{Q}_2 = \begin{bmatrix} + & 0 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 0 \\ + & + & 0 & 0 & 0 \\ + & + & 0 & 0 & 0 \end{bmatrix}$$

then it follows that

$$\mathcal{A}_2 \begin{bmatrix} Q_1 & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \bar{Q}_2 \end{bmatrix} = \mathcal{A}_2 Q_2 = \left[\begin{array}{c|ccccc} \# & 0 & 0 & 0 & 0 & 0 \\ \# & 0 & 0 & 0 & 0 & 0 \\ \hline \# & + & 0 & 0 & 0 & 0 \\ \# & + & 0 & 0 & 0 & 0 \\ 0 & + & + & 0 & 0 & 0 \\ 0 & + & + & 0 & 0 & 0 \end{array}\right].$$

The columns of $Q_2(:, 3:6)$ then generate a basis of the null-space of $\mathcal{A}_2$.

This *blocked* formulation 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 can be smaller than in at each step. Next we describe the algorithm formally.

ALGORITHM 1. *(Blocked LQ algorithm)*
Input: Polynomial matrix $A(s)$ and its rank ρ.
Output: Polynomial matrix $Z(s) = \left[\begin{array}{ccccc} z_1(s) & z_2(s) & \cdots & z_{n-\rho}(s) \end{array}\right]$, where $z_i(s)$ are the vectors in a minimal basis of the null-space of $A(s)$, namely $A(s)Z(s) = 0$.

- Step 1*:*
  Compute the LQ factorization $\mathcal{A}_1 Q_1 = L_1$. Obtain $\bar{r}_1 = \operatorname{rank} \mathcal{A}_1(1:m,:)$ and $r_1 = \operatorname{rank} \mathcal{A}_1$.
  Let $\gamma_1 = n - r_1$. If $\gamma_1 \neq 0$ then the columns of $Q_1(:, r_1+1:n)$ are the vectors of degree zero in $Z(s)$. Let $Z(s) = Q_1(:, r_1+1:n)$.
- Step *i:*
  Compute the LQ factorization

$$\left[\begin{array}{cc} L_{i-1}(m(i-1)+1:(d+i)m, \bar{r}_{i-1}+1:(i-1)n) & \mathcal{A}_1 \end{array}\right] \bar{Q}_i = \bar{L}_i.$$

  The LQ factorization of $\mathcal{A}_i$ is given by $\mathcal{A}_i Q_i = L_i$ where

$$Q_i = \left[\begin{array}{cc} Q_{i-1} & 0 \\ 0 & I_n \end{array}\right] \left[\begin{array}{cc} I_{\bar{r}_{i-1}} & 0 \\ 0 & \bar{Q}_i \end{array}\right].$$

  Obtain $\bar{r}_i = \operatorname{rank} \mathcal{A}_i(1:im,:)$ and $r_i = \operatorname{rank} \mathcal{A}_i$.
  Let $\gamma_i = in - r_i$. If $\mu = \gamma_i - 2\gamma_{i-1} + \gamma_{i-2} \neq 0$ then construct matrix X with μ columns of $Q_i(:, r_i+1:in)$ and update $Z(s)$ as follows:

$$\begin{array}{rcl} X(s) & = & X(1:n,:)s^{i-1} + \ldots + X(n(i-1)+1:in,:) \\ Z(s) & = & \left[\begin{array}{cc} Z(s) & X(s) \end{array}\right]. \end{array}$$

- *Stopping rule:*
  If $\gamma_i - \gamma_{i-1} = n - \rho$, then stop. We have in $Z(s)$ a minimal basis of the null-space of $A(s)$.

We can also consider the use of displacement structure methods [18, 20]. A fast version of the LQ factorization can be formulated easily. A feature of such a method is that the generators associated to $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ are similar. For instance, consider the LQ factorization

$$A_d = \left[\begin{array}{cc} L_d & 0 \end{array}\right] \left[\begin{array}{c} Q_r^T \\ Q_{n-r}^T \end{array}\right].$$

So, a generator of $\mathcal{A}_1$ is given by $G_1 = [X_1^T \quad Y_1^T]^T$ where

$$X_1 = \left[\begin{array}{|ccc|c|} \hline L_d & 0 & 0 & 0 \\ A_{d-1}Q_r & A_{d-1} & A_{d-1}Q_r & A_d \\ A_{d-2}Q_r & A_{d-2} & A_{d-2}Q_r & A_{d-1} \\ \vdots & \vdots & \vdots & \vdots \\ A_0 Q_r & A_0 & A_0 Q_r & A_0 \\ \hline \end{array}\right], \qquad Y_1 = \left[\begin{array}{cccc} Q_r & I_n & Q_r & 0 \end{array}\right].$$

Then we can prove that a generator for $\mathcal{A}_2$ is given by $G_2 = [X_2^T \quad Y_2^T]^T$ where

$$X_2 = \begin{bmatrix} \boxed{\begin{matrix} L_d & 0 & 0 \\ A_{d-1}Q_r & A_{d-1} & A_{d-1}Q_r \\ A_{d-2}Q_r & A_{d-2} & A_{d-2}Q_r \\ \vdots & \vdots & \vdots \\ A_0Q_r & A_0 & A_0Q_r \end{matrix}} & \begin{matrix} 0 \\ \boxed{\begin{matrix} 0 \\ A_d \\ A_{d-1} \\ \vdots \\ A_0 \end{matrix}} \end{matrix} \\ \begin{matrix} & 0 & \end{matrix} & \end{bmatrix}, Y_2 = \begin{bmatrix} Q_r & I_n & Q_r & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

So a blocked formulation can be kept when using a fast version of the LQ factorization in Algorithm 1. The number of columns of the generator does not grow at each step, so when matrix $\mathcal{A}_i$ has a number of columns sufficiently larger than the number of columns of the generator, the advantage of using the generator is evident. The problem is, as we showed in [43], that this condition is not always verified. When this is the case, the fast LQ factorization could be inefficient with respect to the classical LQ factorization. Moreover, it is difficult, if not impossible, to determine based only on the degree and the dimension of a polynomial matrix if the fast algorithm will out-perform the classical algorithm. For this reason and the absence of a proof for backward stability, we recommend to use a fast version of Algorithm 1 very carefully.

6. Algorithmic analysis. Here we analyze the features of the blocked LQ algorithm described in the previous section. The analysis is divided in three parts. First we analyze the numerical stability of Algorithm 1, then we focus on the algorithmic complexity, and finally we review some extensions and perspectives of the algorithm and the Toeplitz approach. We also sketch comparisons with the pencil algorithms and we present numerical examples of some control problems solved by computing a polynomial null-space.

6.1. Numerical stability. We have seen in section 4.2 that the LQ factorization of $\mathcal{A}_i$ is backward stable. Showing the backward stability of the blocked LQ algorithm is not difficult. In fact we only have to consider the blocked process described in Section 5 as a sequence of some particular Householder transformations and then apply the proof of Theorem 18.3 in [16]. For simplicity we illustrate this on a 2×3 polynomial matrix $A(s)$ of degree 1. The extension to an arbitrary matrix is direct.

Consider the computed LQ factorization $\hat{L}_1 = (\mathcal{A}_1 + \Phi_1)\hat{H}_1\hat{H}_2 = (\mathcal{A}_1 + \Phi_1)\hat{Q}_1$ with

$$\hat{L}_1 = \begin{bmatrix} \# & 0 & 0 \\ \# & 0 & 0 \\ \# & \# & 0 \\ \# & \# & 0 \end{bmatrix}, \quad \|\Phi_1\|_2 \leq \phi_1 \|\mathcal{A}_1\|_2.$$

Here $\phi_1 = O(\epsilon)$ is a small constant depending on the dimensions of $\mathcal{A}_1$, the machine precision ϵ, and the number of applied Householder reflections. Now suppose that we want to obtain the LQ factorization of $\mathcal{A}_2$ using the blocked process of Section 5. First we have:

$$(\mathcal{A}_2 + \Phi_2) \begin{bmatrix} \hat{Q}_1 & 0 \\ 0 & I_n \end{bmatrix} = \hat{\mathcal{A}}_2.$$

Here the product of the second block column of $\mathcal{A}_2$ by the identity does not have to be computed and so, the second block column of Φ_2 can be simply set to zero.

Nevertheless, the dimension has changed, so we write $\|\Phi_2\|_2 \leq \bar{\phi}_1\|\mathcal{A}_2\|_2$ where $\bar{\phi}_1$ depends on the dimension of $\mathcal{A}_2$. This second bound is more pessimistic than ϕ_1. Then we apply a third and a fourth Householder reflections as follows

$$\hat{L}_2 = (\mathcal{A}_2 + \Phi_2)\begin{bmatrix} \hat{Q}_1 & 0 \\ 0 & I_n \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \hat{H}_3 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \hat{H}_4 \end{bmatrix} = (\mathcal{A}_2 + \Phi_2)\hat{Q}_2,$$

and so we write $\|\Phi_2\|_2 \leq \phi_2\|\mathcal{A}_2\|_2$. This bound for Φ_2 is even more pessimistic. This can be expected because we have performed two more Householder reflections. In general we can prove the following lemma.

LEMMA 6.1. *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* (3.1) *is backward stable, namely*

$$(\mathcal{A}_{\delta+1} + \Phi)\hat{\mathcal{Z}}_\delta = 0, \qquad \|\Phi\|_2 \leq O(\epsilon)\|\mathcal{A}_{\delta+1}\|_2.$$

This result shows that $\hat{\mathcal{Z}}_\delta$ is the exact null-space of the slightly perturbed matrix $\mathcal{A}_{\delta+1} + \Phi$. Nevertheless, what we want is 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 $\Delta(s) = \Delta_0 + \Delta_1 + \cdots + \Delta_d s^d$ is small for some polynomial matrix norm. Ideally we want

$$\|\Delta_i\|_2 \leq O(\epsilon)\|A_i\|_2, \qquad i = 0, 1, \ldots, d. \tag{6.1}$$

Equivalently, we want that $(\mathcal{A}_{\delta+1} + \bar{\Delta})\hat{\mathcal{Z}}_\delta = 0$ where

$$\bar{\Delta} = \begin{bmatrix} \Delta_d & & \\ \vdots & \ddots & \\ \Delta_0 & & \Delta_d \\ & \ddots & \vdots \\ & & \Delta_0 \end{bmatrix}, \qquad \|\Delta_i\| \leq O(\epsilon)\|A_i\|_2 \quad i = 0, 1, \ldots, d$$

and where $\hat{\mathcal{Z}}_\delta$ is the computed null-space of $\mathcal{A}_{\delta+1}$. The following result gives a bound for the error $\Delta(s)$ when using the LQ factorization to solve (3.1).

THEOREM 6.2. *Let $A(s)$ be a polynomial matrix and suppose that it has a vector $z(s)$ of degree δ in a basis of its null-space. The computed vector $\hat{z}(s)$, obtained from* (3.1) *via the LQ factorization, is the exact null-space vector of the slightly perturbed matrix $A(s) + \Delta(s)$ with*

$$\|\Delta_i\|_2 \leq O(\epsilon)\|\mathcal{A}_{\delta+1}\|_2, \qquad i = 0, 1, \ldots, d. \tag{6.2}$$

Proof. From Lemma 6.1 we know that $(\mathcal{A}_{\delta+1}+\Phi)\hat{\mathcal{Z}}_\delta = 0$ with $\|\Phi\|_2 \leq O(\epsilon)\|\mathcal{A}_{\delta+1}\|_2$. Error matrix Φ has the following general form:

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

where $\times$ represents possibly non zero blocks. Nevertheless, we can always apply row elementary operations gathered in a left multiplier $P = I + E$ with $\|E\|_2 \leq O(\epsilon)\|\mathcal{A}_{\delta+1}\|_2$ such that

$$P(\mathcal{A}_{\delta+1} + \Phi)\hat{\mathcal{Z}}_\delta = (\mathcal{A}_{\delta+1} + \bar{\Delta})\hat{\mathcal{Z}}_\delta = 0.$$

Transformation matrix P and thus $\bar{\Delta}$ are not unique but we can check that $\|\bar{\Delta}\|_2 \leq O(\epsilon)\|\mathcal{A}_{\delta+1}\|_2$ or equivalently $\|\Delta_i\|_2 \leq O(\epsilon)\|\mathcal{A}_{\delta+1}\|_2$ which is the required result. □

Theorem 6.2 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. Rank determination is not only a problem in matrix computations, but an exercise of interpretation of the numerical results. It is well-known that the problem of detecting whether rank $M = r$ is ill-posed[3] 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. So, the problem of finding the eigenstructure of a polynomial matrix is also ill-posed and the forward error is not bounded. First, 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 $\mathcal{A}_{\delta+1}$ has not full row-rank.

A similar analysis can be done for the pencil algorithm developed in [32] to obtain the staircase forms (1.7) and (1.9). Nevertheless, the stability of the process explained in [4] to compute the searched null-space basis from (1.7) is not guaranteed. On the other hand, the null-space basis is obtained directly from (1.9) but a complementary algorithm is necessary to obtain the descriptor realization (1.8) of the analyzed polynomial matrix. Reliable algorithms to compute descriptor realizations are found in [36]. However, no expressions for the backward error bound $\|\Delta(s)\|$ are presented there.

6.2. Algorithmic complexity. Several parameters enter the final expression of the number of performed floating point operations or flops: the dimensions m and n of $A(s)$, its degree d, the maximum expected degree $\bar{\delta}$ of the minimal basis of its null-space, and the actual null-space degree $\delta_{\max}$. Moreover, the final number of performed flops f_f depends on the RRM and the degree search strategy chosen. In fact, f_f is the sum of the flops performed to obtain the rank and a basis of the null-space of $\mathcal{A}_i$ at each step, and the number of steps depends on the degree search strategy. Considering, for simplicity, a square $n \times n$ polynomial matrix $A(s)$ of degree d, in [43, 44] we present some plots of the evolution of the execution time for different versions of the algorithm. We also analyze the fast versions using the block displacement structure theory to compute the LQ factorization and the L-factor of $\mathcal{A}_i$. In Table 1 in [43] we present the execution time required for the different algorithms when solving equation (1.5) for a mass-spring system. As a conclusion of this previous works we consider that the algorithmic complexity of Algorithm 1 is $O(dn^3)$. At each step a sequence of Householder reflections is applied to the columns of $\mathcal{A}_i$ and this number of columns depends on n. The degree d only takes part in the number of rows of $\mathcal{A}_i$ and, with the blocked structure of the algorithm, we always factorize a matrix with $(d+1)n$ rows.

[3] in the classical sense of Hadamard which means that the solution of the problem does not depends continuously on the data or the parameters [12]

The pencil algorithms, on the other hand, are divided in two parts. For the descriptor algorithm [37], a first part consists in computing the descriptor realization (1.8) of the analyzed polynomial matrix and a second part consists in computing the staircase form (1.9). For the algorithm in [4], the first part consists in computing the staircase form (1.7) and the second part consists in computing a basis of the null-space of $sB_z - A_z$. These features of the pencil algorithms represent, in some cases, a disadvantage with respect to the Toeplitz algorithm, since the latter obtains the searched null-space with only one process. In the remainder of this section we explain these ideas. We only compare our Toeplitz algorithm with the pencil algorithm in [4] because the analysis of methods to compute descriptor realizations needed by the algorithm in [37] are very remote from our objective in this paper.

The method presented in [32] to obtain the staircase form (1.7) has an algorithmic complexity in $O(d^3n^3)$. The algorithm always starts by obtaining the SVD of B_0 in pencil (1.6) which has nd columns. It can be shown that $sB_z - A_z$ in (1.7) has the form

$$sB_z - A_z = \begin{bmatrix} -A_{11} & sB_{12} - A_{12} & \cdots & sB_{1k} - A_{1k} \\ & -A_{22} & \cdots & sB_{2k} - A_{2k} \\ & & \ddots & \vdots \\ & & & -A_{kk} \end{bmatrix}$$

where matrix A_{ii} has dimensions $m_i \times n_i$ for $i = 1, 2, \ldots, k$. Moreover, matrix $A(s)$ has $n_i - m_i$ vectors of degree $i - 1$ in the basis of its null-space, and $m_i - n_{i+1}$ infinite elementary divisors of order i. Obtaining $Z(s)$ from a basis of the null-space of $sB_z - A_z$ is done with the process described in Section 4 of [4]. The amount of flops required by this process is generally large and it has to be added to the amount of flops performed by the first part of the algorithm. When n and d are large, the dominant term will be d^3n^3, so, we say that the algorithmic complexity of the whole pencil algorithm is $O(d^3n^3)$.

From this analysis we conclude that an advantage of Algorithm 1 with respect to the pencil algorithm is the weak dependency on the degree d. Obviously, expressions $O(dn^3)$ and $O(d^3n^3)$ are only approximations of the exact number of performed flops. Exact expressions are intricate, see for example [40] where we derive exact expression of the number of flops performed by a Toeplitz algorithm and a pencil algorithm to obtain the infinite structural indices of a polynomial matrix. Here we avoid this kind of analysis. Instead we present the execution times of the algorithms when solving some relevant numerical examples.

We consider the execution time of the algorithms as an indicator of the number of performed flops. So, if we want to compare them we have to ensure that both algorithms perform the same type of operations. For this reason, we have programmed the SVD[4] and the LQ factorization from scratch as a sequence of Householder reflections. Then we have implemented Algorithm 1 and the pencil algorithm using these handmade functions. In order to distinguish each programmed algorithm, we use the acronym `toepLQ` to refer to Algorithm 1, `toepSVD` to refer to Algorithm 1 using the SVD as the RRM, and `penSVD` to refer to the pencil algorithm. The pencil algorithm can also be programmed using the LQ factorization, and we use `penLQ` to refer to this version. The process presented in [4] to obtain a basis of the null-space of $sB_z - A_z$

[4]Our handmade function SVD computes the singular value decomposition of M by obtaining the Schur form of M^TM. In practice we can consider it as a sequence of two LQ factorizations.

implies products of polynomial matrices which requires times of execution generally larger than the time required to obtain the generalized form (1.7). In order to underline this fact, in Tables 6.1 and 6.2 we split the execution time of algorithms `penSVD` and `penLQ`: the first value represents the time required to obtain the generalized Schur form (1.7), and the second value represents the time required to obtain the null-space of $sB_z - A_z$. The following examples were performed in a Sun Ultra 5 work station with Matlab 6.5. All the executions times are given in seconds.

EXAMPLE 1. *First we consider the problem of computing a coprime factorization of a transfer function of a linear multivariable system. Consider the transfer function given in* [4]

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

with

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

where we added an integer parameter a. If $a = 1$ we have the same system as in Example 5.1 *in* [4]. *We can obtain a left coprime factorization $T(s) = D_l^{-1}(s)N_l(s)$ from $Z(s)$ which is a solution of*

$$A(s)Z(s) = \begin{bmatrix} N_r^T(s) & -D_r^T(s) \end{bmatrix} \begin{bmatrix} D_l^T(s) \\ N_l^T(s) \end{bmatrix} = 0. \tag{6.3}$$

Matrix $Z(s)$ has the following general form

$$Z(s) = \begin{bmatrix} 0 & 0 & 0 & 0 & c_1(1+s)^a \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_2(1+s)^2 & 0 \\ 0 & 0 & c_3(1+s) & 0 & 0 \\ 0 & 0 & 0 & 0 & c_1 s^2 \\ 0 & 0 & 0 & c_2 s^2 & 0 \\ 0 & 0 & 0 & c_2(s+s^2) & 0 \\ 0 & 0 & -c_3 s & 0 & 0 \end{bmatrix}$$

with some suitable constants c_i, see [4]. *Notice that if we vary a, then the degree of $P(s)$ and the degree of the minimal basis $Z(s)$ also vary. In* Table 6.1 *we present the execution times for the algorithms when solving equation* (6.3) *for different values of a.*

Notice the fast growth of the execution time of the pencil algorithms with respect to the Toeplitz algorithms. When the degree of $Z(s)$ grows, the number of steps also grows and therefore the number of columns of the last analyzed Toeplitz matrices can

TABLE 6.1
Execution times in seconds when applying the different algorithms to solve equation (6.3) *for different values of* a.

a	`penSVD`	`penLQ`	`toepSVD`	`toepLQ`
3	0.07 + 0.69	0.05 + 0.7	0.14	0.11
5	0.14 + 1.12	0.09 + 1.12	0.21	0.15
10	0.52 + 2.76	0.29 + 2.75	0.64	0.37
15	1.4 + 4.52	0.78 + 4.51	1.29	0.72
20	4.13 + 8.23	2.14 + 8.25	2.69	1.46

be large. Nevertheless, because of the blocked formulation of Algorithm 1, the actual number of columns analyzed at step i is smaller than in, which has a positive impact on the execution time.

EXAMPLE 2. *Now, we consider an interconnected mass-spring system with* $p = 2, 3, \ldots, 20$ *masses. We apply the algorithms to find the transfer function matrix from a force acting on the first mass to the position of the last mass. We have to solve equation* (1.5) *where*

$$D(s) = \begin{bmatrix} 1+s^2 & -1 & & & \\ -1 & 2+s^2 & -1 & & \\ & -1 & 2+s^2 & & \\ & & & \ddots & -1 \\ & & & -1 & 2+s^2 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Matrix $D(s)$ *has dimension* $p \times p$. *The polynomial matrix* $A(s) = [\ D(s) \quad -B\ ]$ *has full row rank* p, *no finite zeros and since* $\operatorname{rank} A_2 = p$ *it has no infinite zeros either. So, from* Theorem 2.1, *the degree of the unique vector in the minimal basis of the null-space of* $A(s)$ *is always* $\delta_{\max} = 2p$. *We present the execution times of each algorithm in* Table 6.2.

Now notice that the execution time of the Toeplitz algorithms grows significantly. In fact, when p is large, matrix $A(s)$ represents the worst case for Algorithm 1. The reason is that it is necessary to perform the maximum number of steps and the analyzed Toeplitz matrices are large. On the other hand the degree of $A(s)$ is always 2 and all the RRM performed to obtain the generalized form (1.7) have dimensions smaller than $2p \times (2p+1)$. This has a positive impact on the execution time of the first part of the pencil algorithms. The problem is that at this step we have not yet the vector which forms the minimal basis of the null-space of $A(s)$ but only its degree. If we want to obtain the vector, then a large execution time is required by the procedure outlined in [4]. With this example we can see the advantage of the Toeplitz algorithms: we obtain, with the same computational effort, the degrees and the vectors of the minimal null-space basis.

6.3. Extension of the Toeplitz algorithm. When using the pencil algorithm we can extract from $sB_z - A_z$ in (1.7) the structural indices of the null-space of $A(s)$ and also its infinite structural indices. Moreover, we do not require the rank of $A(s)$ since we can also deduce it from $sB_z - A_z$. Now we show how the Toeplitz algorithm allows to obtain not only $Z(s)$ but also the infinite structural indices, the associated infinite vectors and the rank of $A(s)$ with the same computational effort.

First suppose that $\rho = \operatorname{rank} A(s)$ is given. Then, from (2.2), the infinite structure of $A(s)$ is also contained in the null-space of some Toeplitz matrices. Moreover notice

TABLE 6.2
Execution times in seconds when applying the different algorithms to solve equation (1.5) *of increasing dimensions.*

p	`penSVD`	`penLQ`	`toepSVD`	`toepLQ`
3	0.11 + 1.64	0.06 + 1.65	0.16	0.11
5	0.18 + 3.06	0.11 + 3.05	0.32	0.20
10	0.42 + 7.14	0.24 + 7.14	0.95	0.56
15	0.95 + 14.57	0.51 + 14.58	2.14	1.27
20	1.80 + 24.79	0.94 + 24.78	5.49	3.84

that Toeplitz matrix in (2.2) is equal to the uppermost $\bar{k}_i m$ rows of $\mathcal{A}_{\bar{k}_i}$, so, indices $\bar{r}_i$ obtained by Algorithm 1 contain the infinite structure of $A(s)$. In fact, we can show that if $\beta_i = i\rho - \bar{r}_i - \sum_{k=1}^{i-1} \beta_k$, then matrix $A(s)$ has $\beta_j - \beta_{j+1}$ infinite elementary divisors of order $j-d$ for $j = 1, 2, \ldots, l$ where l is such that $\beta_l = 0$. In simpler words, following our notations, we say that $A(s)$ has $\sum_{i=1}^{l} \beta_i$ zeros at infinity. From (2.3) it holds that $l \leq \min(m,n)d$.

Now, consider that we do not know the rank of $A(s)$. In this case we can simply start with $\rho = \min(m,n)$ and update this value when finding vectors in $Z(s)$ as explained with the following simple example.

EXAMPLE 3. *We want to obtain the infinite and the null-space structure of matrix*

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

Supposing that we do not know its rank, starting with the guess $\rho = \min(m,n) = 3$.

After the first step of Algorithm 1 *we have* $\bar{r}_1 = 1$ *and* $r_1 = 3$. *So, we have* $\beta_1 = \rho - 1 = 2$ *and* $\gamma_1 = n - 3 = 1$. *This shows that* $A(s)$ *has a vector of degree 0 in a basis of its null-space.*

After the following three steps we have:

$$\begin{aligned} \beta_2 &= 2, \quad \gamma_2 = 2, \\ \beta_3 &= 1, \quad \gamma_3 = 3, \\ \beta_4 &= 1, \quad \gamma_4 = 4. \end{aligned}$$

Finally, at the fourth step we obtain

$$\beta_5 = 1, \qquad \gamma_5 = 6.$$

At this moment we know that ρ *cannot be 3 but at most 2 because we have* $\gamma_5 - 2\gamma_4 + \gamma_3 = 1$ *which means that there is a vector of degree 4 in a basis of the null-space of* $A(s)$. *So we update the values*

$$\rho = 2, \quad \beta_1 = 1, \quad \beta_2 = 1, \quad \beta_3 = 0.$$

Then, since $\beta_3 = 0$, *we know that the rank of* $A(s)$ *is actually* $\rho = 2$. *Moreover, as* $\gamma_5 - \gamma_4 = n - \rho = 2$, *we do not go further. We conclude that matrix* $A(s)$ *has one vector of degree 0 and one vector of degree 4 in a minimal basis of its null-space. Matrix* $A(s)$ *has also 2 zeros at infinity. Notice that equation* (2.3) *is satisfied.*

With this example we show that the Toeplitz algorithm is as general as the pencil algorithm. Moreover, from equation (2.1) we can see that the finite structure of $A(s)$ can also be recovered from the null-spaces of some Toeplitz matrices as soon as the finite zeros are known. In [45] we showed that for this case the backward error $\Delta(s)$ is also bounded.

7. 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. This algorithm can naturally be extended to obtain the whole eigenstructure of the polynomial matrix [42], and thus it can be seen as a reliable alternative to the pencil methods presented in [34]. Our algorithm process in a numerical way some related block Toeplitz matrices, 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 [32, 4].

The algorithm can be implemented in several ways depending on the numerical method used to factorize the related Toeplitz matrices, and on the strategy to vary the dimension of the Toeplitz matrix analyzed at each step. All these options are also analyzed in detail in our previous works [43, 44]. The possibility of different implementations allows the algorithm to perform in a very efficient way in some particular cases. For instance, we can obtain the structural indices of a polynomial matrix (i.e. the multiplicities of the zeros at infinity and the degree of the vectors in a minimal null-space basis) in only one step by factorizing a large Toeplitz matrix. We can also consider using some block displacement structure methods [18, 20] to factorize this large Toeplitz matrix. Despite this versatility, in this paper we developed only one version: the most efficient in most of the cases. This version is based on the LQ factorization and has a blocked formulation. We showed that it is more efficient than a similar algorithm presented recently in [3]. Moreover, in this paper we presented a full analysis of the numerical stability and complexity of our algorithm and we compared it with the pencil algorithms.

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 (flops) is its blocked formulation. Because of this blocked formulation, we have seen that even in the cases when the number of steps performed by our algorithm is maximal, the number of performed flops is comparable with the number of flops performed by the pencil algorithm [32]. The difference is that with our algorithm we obtain, with the same computational effort, the structural indices and the associated vectors, whereas with the pencil algorithm we only obtain the structural indices. If we want also the associated vectors, then we have to apply the process described in [4] which implies many more flops. A similar conclusion can be done with respect to the algorithm presented in [37]. In this case algorithm [32] applied over a descriptor realization of the polynomial matrix gives the searched null-space basis. Nevertheless, the computational effort required to obtain such a descriptor realization has also to be considered.

Concerning numerical stability, we have proven the backward stability of our algorithm. A bound for the backward error produced in the coefficients of the analyzed polynomial matrix is determined. Similar bounds are determined for the algorithm in [32]. Nevertheless, the stability of the process presented in [4] is not guaranteed. On the other hand, reliable algorithms to obtain descriptor realizations are described in [36] but bounds for the backward error produced in the coefficients of the polynomial matrix are not presented. The bound for our backward error is given by equation (6.2). This bound is more pessimistic than the ideal one given by equation (6.1). Sharper

bounds can be expected from a componentwise error analysis [16]. An analysis with geometrical arguments as in [5] can be considered as a future line of research. The objective of such an analysis should be the derivation of exact first order expressions of the errors in the coefficients of the polynomial matrix due to the perturbations in the analyzed Toeplitz matrices. In [6, 23], 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 [28], a lot of meaningful ill-posed problems can be solved numerically very satisfactorily in practice. In [39] 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 in control.

Acknowledgments. Juan Carlos Zúñiga acknowledges support from the National Council of Science and Technology of Mexico (CONACYT) as well as from the Secretariat of Public Education of Mexico (SEP). Didier Henrion acknowledges support from project No. 102/02/0709 of the Grant Agency of the Czech Republic as well as from project No. ME 698/2003 of the Ministry of Education of the Czech Republic.

REFERENCES

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

[2] S. Barnett, *Polynomial and linear control synthesis*, Marcel Dekker, London, 1983.

[3] J. C. Basilio and M. V. Moreira, *A robust solution of the generalized polynomial Bezout identity*, Linear Algebra Appl., 385, pp. 287–303, 2004.

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

[5] A. Edelman and H. Murakami, *Polynomial roots from companion matrix eigenvalues*, Math. Comput., 64, pp. 763–776, 1995.

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

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

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

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

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

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

[12] J. Hadamard, *Sur les problèmes aux dérivées partielles et leur signification physique*, Princeton University Bulletin, pp. 49–52, 1902.

[13] 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.

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

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

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

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

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

[19] D. Kressner and P. M. Van Dooren, *Factorizations and linear system solvers for matrices with Toeplitz structure*, SLICOT Working Notes, 2000.

[20] D. Kressner, *Numerical Methods for Structured Matrix Factorizations*, Ph.D. thesis, Faculty of Mathematics, Technical University of Chemnitz, Germany, 2001.

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

[22] V. Kučera, *Diophantine equations in control–a survey*, Automatica, 29, pp. 1361–1375, 1993.

[23] D. Lemonnier and P. Van Dooren, *Optimal scaling of block companion pencils*, International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium, 2004.

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

[25] A. S. Morse, *Structural invariants of linear multivariable systems*, SIAM J. Control Optim., 11, pp. 446–465, 1973.

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

[27] P. 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.

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

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

[30] 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, pp. 129–151, 2001.

[31] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43, pp. 235–286, 2001.

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

[33] 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, pp. 180–189, 1979.

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

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

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

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

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

[39] Z. Zeng, *Computing multiple roots of inexact polynomials.* To appear in Math. Comput., 2004.

[40] J. C. Zúñiga and D. Henrion, *Comparison of algorithms for computing infinite structural indices of polynomial matrices*, European Control Conference, Cambridge, UK, 2003.

[41] J. C. Zúñiga and D. Henrion, *A Toeplitz algorithm for the polynomial J-spectral factorization*, IFAC Symposium on System, Structure and Control, Oaxaca, Mexico, 2004.

[42] 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.

[43] 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.

[44] 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.

[45] J. C. Zúñiga and D. Henrion, *Numerical stability of block Toeplitz algorithms in polynomial matrix computations.* To be registered as a LAAS-CNRS Research Report. Submitted to the IFAC World Congress on Automatic Control, Prague, Czech Republic, 2005.