COMPARISON OF ALGORITHMS FOR COMPUTING INFINITE STRUCTURAL INDICES OF POLYNOMIAL MATRICES

J.C. Zúñiga[1] **and D. Henrion**[1,2,3]

1. Laboratoire d'Analyse et d'Architecture des Systèmes,
Centre National de la Recherche Scientifique.
7 Avenue du Colonel Roche,
31 077 Toulouse, France.

2. Institute of Information Theory and Automation,
Academy of Sciences of the Czech Republic.
Pod vodárenskou věží 4,
182 08 Prague, Czech Republic.

3. Corresponding author. FAX: +33 5 61 33 69 69.
E-mail: henrion@laas.fr

Keywords : Polynomial matrices, structural indices, decoupling control, numerical linear algebra, computer-aided control system design.

Abstract

A new algorithm is proposed to compute the infinite structural indices of a polynomial matrix, i.e. the algebraic and geometric multiplicities of its poles and zeros at infinity. The algorithm is based on numerically stable operations only, and takes full advantage of the block Toeplitz structure of a constant matrix built directly from the polynomial matrix coefficients. Comparative numerical examples and a full computational complexity analysis indicate that the Toeplitz algorithm can be viewed as a competitive alternative to the well-known state-space pencil matrix algorithm for obtaining structural indices.

1 Introduction

Polynomials and polynomial matrices arise naturally and cannot be avoided in linear system theory. By analogy with scalar rational transfer functions, matrix transfer functions of linear systems can be written in polynomial matrix fraction form as $N(s)D^{-1}(s)$ where $D(s)$ is a non-singular polynomial matrix [8, 6, 1].

Zeros of polynomial matrices naturally represent either poles or zeros of linear multivariable systems described by polynomial matrix fractions. They are frequently encountered when analyzing and/or designing linear systems or filters. Associated with the zeros are the finite and infinite structures of a polynomial matrix $A(s)$, defined from specific canonical forms under matrix equivalence: the Smith form for the finite structure [3] and the Smith-MacMillan form at infinity for the infinite structure [15].

In this paper, we are concerned with numerical algorithms for computing the so-called infinite structural indices of a polynomial matrix, i.e. algebraic and geometric multiplicities of its zeros and poles at infinity. Dynamically, infinite zero orders of the system are related to the number of times that the outputs of the system have to be differentiated for a component of the input vector to appear explicitly. Thus, this information is crucial in the solution of important control problems. For instance, a linear multivariable system is row by row decouplable by static state feedback if and only if the sum of its infinite zero orders equals the sum of its row infinite zero orders. Other typical control problems where infinite structural indices of polynomial matrices play a fundamental role are model matching and disturbance rejection, see the introduction in [5] and the references therein.

Up to our knowledge there basically exist two algorithms to compute structural indices of polynomial matrices:

- The pencil algorithm, based on generalized state-space realizations and the correspondence between the Smith-MacMillan form of a rational matrix and the Kronecker canonical form of an associated pencil matrix [13]. The pencil algorithm allows to compute all the structural indices (finite and infinite) of a polynomial matrix [14].

- The Toeplitz algorithm, proposed in [5] as an extension of a method originally described in [4] to compute and extract the finite structure of a polynomial matrix. The algorithm is rather similar in spirit to the algorithm proposed in [12]. In this reference, the structure of a rational matrix at any point is computed via orthogonal transformations on Toeplitz ma-

trices built from the Laurent expansion.

In this paper, our objective is to revisit the Toeplitz algorithm of [5] in order to take full advantage of the block Toeplitz structure and therefore to reduce the computational burden as much as possible. The algorithm is designed while keeping with our main impetus, which is the development of reliable numerical methods for dealing with polynomial matrices and their implementation in a user-friendly commercial Matlab package called the Polynomial Toolbox, developed and produced by the company PolyX Ltd. [10]. In this regard, all the routines used in the algorithm are numerically stable in the sense that they are based on backward stable (orthogonal) transformations [2, 9].

By a thorough computational complexity analysis, we show that the Toeplitz algorithm may be considered as a competitive "polynomial approach" alternative to the "state-space approach" algorithms proposed in [13] and recently implemented in the Fortran library Slicot [11].

2 Toeplitz Approach

Consider a non-singular $n \times n$ polynomial matrix

$$A(s) = A_0 + A_1 s + \cdots + A_{d-1} s^{d-1} + A_d s^d \qquad (1)$$

of degree d and let $T_k = A_{d+1-k}$.

The algorithm to obtain the structure at infinity of $A(s)$ processes the block Toeplitz matrix

$$T = \begin{bmatrix} T_1 & & & \\ T_2 & T_1 & & \\ \vdots & \vdots & \ddots & \\ T_k & T_{k-1} & \cdots & T_1 \end{bmatrix}$$

and is described as follows:

- Step 1. Obtain the singular value decomposition (SVD) $S = P^T T_1 Q_1$ and the rank r_1 of T_1.

  If $v_1 = n - r_1$ is not zero, make the column compression $T_1 Q_1 = [C_1\ 0]$ where C_1 has r_1 columns and update the next row block of T, $\widehat{T}_2 = [T_2 \bar{Q}_1\ T_1]$ where $\bar{Q}_1$ contains the v_1 rightmost columns of Q_1.

  If v_1 is zero then go to End.

- Step k. Obtain the SVD $S = P^T \widehat{T}_k \widehat{Q}_k$ and the rank r_k of $\widehat{T}_k$.

  If $v_k = n - r_k$ is not zero, make the column compression $\widehat{T}_k \widehat{Q}_k = [C_k\ 0]$ where C_k has r_k columns, let

  $$Q_k = \begin{bmatrix} Q_{k-1} & 0 \\ 0 & I_n \end{bmatrix} \begin{bmatrix} I_{r_1+r_2+\cdots+r_{k-1}} & 0 \\ 0 & \widehat{Q}_k \end{bmatrix}$$

  and update the next row block of T, $\widehat{T}_{k+1} = [[T_{k+1} \cdots\ T_3\ T_2]\bar{Q}_k\ T_1]$ where $\bar{Q}_k$ contains the $w_k = v_1 + v_2 + \cdots + v_k$ rightmost columns of Q_k.

  If v_k is zero then go to End.

- End. For $i = 1, 2, \ldots, k-1$, if $i \geq d$ then $A(s)$ has $v_i - v_{i+1}$ zeros at infinity of degree $i - d$. If $i \leq d$ then $A(s)$ has $v_i - v_{i+1}$ poles at infinity of degree $d - i$. $A(s)$ has $n - v_1$ poles at infinity of degree d.

In order to count the number of operations that the previous algorithm requires, we present its basic code in a Matlab-like syntax.

Function `sinftoep`
Returns vector $v = [v_1\ v_2\ \cdots\ ]$ of nullities characterizing the structure at infinity of a given non-singular polynomial matrix $A(s) \in \mathbb{R}^{n \times n}[s]$ of degree d.

- Step 0. $\widehat{T}_1 = T_1$
  $w_0 = 0$
  $Q_0 = I_n$
  $k = 1$

- Step k. $[P, D, \widehat{Q}_k] = \texttt{svd}(\widehat{T}_k)$
  $r_k = \text{rank}(\widehat{T}_k)$
  $v_k = n - r_k$
  $w_k = w_{k-1} + v_k$
  `If` $v_k = 0$, `goto` End
  $Q_{k_1} = Q_{k-1} * \widehat{Q}_k(1 : w_{k-1}, (r_k + 1) : (n + w_{k-1}))$
  $Q_{k_2} = \widehat{Q}_k((w_{k-1} + 1) : (n + w_{k-1}), (r_k + 1) : (n + w_{k-1}))$
  $Q_k = [Q_{k_1}; Q_{k_2}]$
  $\widehat{T}_{k+1} = [[T_{k+1} \cdots\ T_3\ T_2] * Q_k\ T_1]$
  $k = k + 1$

- End. `Return` v

We consider that an SVD for a matrix $A \in \mathbb{R}^{m \times n}$ requires $4m^2n + 8mn^2 + 9n^3$ operations, and that a multiplication $C = AB$ with $A \in \mathbb{R}^{m_A \times n}$ and $B \in \mathbb{R}^{n \times n_B}$ requires $(2n-1)m_A n_B$ operations [2].

In step k, `sinftoep` executes one SVD of dimension $n \times (n + w_{k-1})$, one multiplication of dimensions $(k-1)n \times w_{k-1}$ by $w_{k-1} \times w_k$, and one multiplication of dimensions $n \times kn$ by $kn \times w_k$, namely

$$\begin{aligned} \text{flops}_{T_k} = {} & 9w_{k-1}^3 + n(33 + 2k)w_{k-1}^2 + \\ & n(47n + 2kn + 2v_k k - 2v_k - k)w_{k-1} + \\ & kn(2n-1)v_k + 21n^3 \end{aligned} \qquad (2)$$

operations.

Supposing that q steps are needed to recover all the structure at infinity of matrix $A(s)$, that is to say $v_{q+1} = 0$,

then the total number of operations that `sinftoep` performs is

$$\text{flops}_T = \sum_{k=1}^{q+1} \text{flops}_{T_k} \tag{3}$$

3 Pencil Approach

The algorithm to obtain the structure at infinity of the matrix $A(s)$ given in (1) processes the pencil

$$P(s) = sE - L = s \begin{bmatrix} & -I_n & & \\ & & \ddots & \\ & & & -I_n \\ A_0 & A_1 & \cdots & A_{d-1} \end{bmatrix} - \begin{bmatrix} -I_n & & & \\ & \ddots & & \\ & & -I_n & \\ & & & -A_d \end{bmatrix}$$

and is described as follows. For more details see [13].

Let us take $L_1 = L$, $E_1 = E$, $w_0 = 0$, and $r_0 = nd$.

- Step k. Obtain the SVD $S = P^T L_k Q$ and the rank r_k of L_k.

  If $v_k = r_{k-1} - r_k$ is not zero, make the column compressions $L_k Q = [R_{L_k}\ 0]$, and $E_k Q = [R_{E_k}\ N_k]$, where R_{E_k} and R_{L_k} have r_k columns. Obtain the SVD $S = P^T N_k Q$, and make the permuted row compressions

  $$I_p P^T R_k = \begin{bmatrix} L_{k+1} \\ R_{L_{k,2}} \end{bmatrix}, I_p P^T R b_k = \begin{bmatrix} E_{k+1} \\ R_{E_{k,2}} \end{bmatrix}, I_p P^T N_k = \begin{bmatrix} 0 \\ N_{k,2} \end{bmatrix}$$

  where L_{k+1} and E_{k+1} have r_k rows, and

  $$I_p = \begin{bmatrix} 0 & I_{r_k} \\ I_{v_k} & 0 \end{bmatrix}.$$

  If v_k is zero then go to End.

- End. For $i = 1, 2, \ldots, k-1$, if $i \geq d$ then $A(s)$ has $v_i - v_{i+1}$ zeros at infinity of degree $i - d$. If $i \leq d$ then $A(s)$ has $v_i - v_{i+1}$ poles at infinity of degree $d - i$. $A(s)$ has $n - v_1$ poles at infinity of degree d.

In order to count the number of operations that the previous algorithm requires, we also present its basic code in a Matlab-like syntax.

Function `sinfpenc`
Returns vector $v = [v_1\ v_2\ \cdots\ ]$ of nullities characterizing the structure at infinity of a given non-singular polynomial matrix $A(s) \in \mathbb{R}^{n \times n}[s]$ of degree d.

- Step 0. Obtain pencil $P(s) = sE - L$
  $L_1 = L$
  $E_1 = E$
  $w_0 = 0$
  $r_0 = n * d$
  $k = 1$

- Step k. $[P, D, Q] = \text{svd}(L_k)$
  $r_k = \text{rank}(L_k)$
  $v_k = r_{k-1} - r_k$
  $w_k = w_{k-1} + v_k$
  `If` $v_k = 0$, `goto` End
  $L_{k+1} = L_k * Q(:, 1 : r_k)$
  $E_{k+1} = E_k * Q$
  $[P, D, Q] = \text{svd}(E_{k+1}(:, (r_k + 1) : (nd - w_{k-1}))$
  $\bar{P} = I_p * P^T$
  $L_{k+1} = \bar{P} * L_{k+1}$
  $L_{k+1} = L_{k+1}(1 : r_k, :)$
  $E_{k+1} = \bar{P} * E_{k+1}(:, 1 : r_k)$
  $E_{k+1} = E_{k+1}(1 : r_k, :)$
  $k = k + 1$

- End. `Return` v

In step k, `sinfpenc` executes one SVD of dimension $(nd - w_{k-1}) \times (nd - w_{k-1})$, another one of dimension $(nd - w_{k-1}) \times v_k$, two multiplications of dimensions $(nd - w_{k-1}) \times (nd - w_{k-1})$ by $(nd - w_{k-1}) \times (nd - w_{k-1})$, and three multiplications of dimensions $(nd - w_{k-1}) \times (nd - w_{k-1})$ by $(nd - w_{k-1}) \times (nd - w_k)$, namely

$$\begin{aligned} \text{flops}_{P_k} = {} & -31w_{k-1}^3 + (93nd - 2v_k - 5)w_{k-1}^2 + \\ & (10nd + 4ndv_k - 93n^2d^2 - 3v_k - 8v_k^2)w_{k-1} + \\ & 9v_k^3 + 8ndv_k^2 + nd(3 - 2nd)v_k + n^2d^2(31nd - 5) \end{aligned} \tag{4}$$

operations.

Now, supposing that q steps are needed to recover all the structure at infinity of matrix $A(s)$, the total number of operations that `sinftoep` performs is equal to

$$\text{flops}_P = \sum_{k=1}^{q+1} \text{flops}_{P_k} \tag{5}$$

4 Comparison

As we can see from equations (2) and (4), when obtaining the structure at infinity of a polynomial matrix $A(s)$ several quantities are involved, not only the dimension n of the matrix, but also its degree d and in general its structure, namely nullities v_k and number of steps q. In this section we present an analysis of the dependence of the functions `sinfpenc` and `sinftoep` on these dimensions.

4.1 Dependence on the degree d and the dimension n for given infinite structural indices

If we consider the nullities fixed, we can see from equation (4) that the algorithmic complexity of the pencil approach is $O(n^3d^3)$. On the other hand, with the Toeplitz approach we can eliminate the dependence on degree d. We can see from equation (2) that the algorithmic complexity is $O(n^3)$. The reason is that the pencil algorithm always executes an SVD of dimension $nd \times nd$.

Test 1 *Consider the polynomial matrix of degree d*

$$A(s) = \begin{bmatrix} s^d & p_1(s) & p_2(s) \\ 0 & s^{d-a} & p_3(s) \\ 0 & 0 & s^{d-a-b} \end{bmatrix},$$

where a, b are given integers and $p_1(s)$, $p_2(s)$, $p_3(s)$ are given polynomials.

If the degree of $p_3(s)$ is less than $d - a$, then we can see that the vector of the nullities of $A(s)$ is

$$v = [\underbrace{2, 2, \ldots, 2}_{a}, \underbrace{1, 1, \ldots, 1}_{b}, 0].$$

Now fix the structure of $A(s)$ with $a = 5$ and $b = 2$ and vary the degree d in order to show how it affects the number of operations in functions `sinftoep` *and* `sinfpenc`*. Some representative results are: with $d = 20$, $\text{flops}_T = 136683$ and $\text{flops}_P = 37181275$ and with $d = 60$, $\text{flops}_T = 136683$ and $\text{flops}_P = 1246161955$.*

We can see that the algorithmic complexity of function `sinftoep` does not depend directly on the degree of the matrix. On the other hand, we can see the rapid growth in the number of operations executed by function `sinfpenc` when degree d increases.

Test 2 *Now consider the polynomial matrix*

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

where b is a given integer.

For this matrix, we can increase b in order to increase the dimension n of $A(s)$, nevertheless the degree $d = 3$ and the vector of nullities v do not change.

We are going to vary b in order to show how it affects the number of operations in functions `sinftoep` *and* `sinfpenc`*. Some representative results are: with $n = 20$, $\text{flops}_T = 2518093$ and $\text{flops}_P = 39830120$ and with $n = 60$, $\text{flops}_T = 45335133$ and $\text{flops}_P = 1275721160$.*

Now we can see that algorithmic complexities of both functions depend on the dimension n.

From these results we can conclude that one apparent advantage of Toeplitz approach is the independence on the matrix degree in some specific cases. Nevertheless, generally a change in the degree or the dimension of a polynomial matrix causes a change in its structure at infinity, namely a change in the values of nullities v_k and in the number of steps q. Then, in order to determine which approach is more efficient, we are going to analyze equations (3) and (5) in a more general way.

4.2 General case, dependence on infinite structural indices

First, we have to bound numbers q, v_k and w_k for $k = 1, 2, \ldots, q$. Let us consider a non-singular polynomial matrix $A(s) \in \mathbb{R}^{n\times n}[s]$ of degree d. From the Toeplitz approach it is easy to see that

$$n - 1 \geq v_1 \geq v_2 \geq v_3 \geq \cdots \geq v_q \geq 1,$$
$$k \leq w_k \leq k(n-1).$$

However, from the pencil approach we know that w_k cannot be greater than nd. Said in another way, when $w_k = nd$ we have all the structure at infinity of $A(s)$, then in general $k \leq w_k \leq nd$.

From this results we conclude that in the worst case, when $v_1 = v_2 = \cdots = v_q = 1$, at most $q = nd$ steps are required to obtain the structure at infinity of $A(s)$, namely

$$1 \leq q \leq nd.$$

As to the number of operations, it can be seen that the worst case for the pencil approach is when v_k is small. v_k small implies not only more steps, but also operations on larger matrices because the deflation at each step will be smaller. On the other hand, when v_k is large, the number of steps and the dimension of the operations are reduced. It is more difficult to find the worst or the best case for the Toeplitz approach. If v_k is small we have less steps, however we also have a smaller deflation, namely operations on larger matrices. On the other hand, if v_k is large the number of steps increases but the deflation increases too, namely, smaller matrices are processed. To continue with this analysis we consider the extreme cases.

Extreme case 1: $v_k = n - 1$

First let us consider that $v_k = n - 1$ for $k = 1, 2, \ldots, q$. We know that at the last step $w_{q-1} = (q-1)(n-1) \leq nd$, therefore the maximum number of steps is $q_{max} = \text{floor}(\frac{nd}{n-1} + 1)$, where $\text{floor}(x)$ is the largest integer less than or equal to x. However, it is important to notice that in a non-singular polynomial matrix the number of poles (we only have poles at infinity) must be equal to the number of zeros (finite or infinite) including multiplicities [15]. This condition limits the maximum number of steps q_{max}. For example consider a polynomial matrix $A(s)$ of degree 3 and dimension 7, if $v_k = n - 1 = 6$, $A(s)$ has one pole at infinity of degree 3 and we can not have

$q = \text{floor}(\frac{nd}{n-1} + 1) = 4$ because this would imply that $A(s)$ has 6 zeros at infinity of degree 1, hence 3 finite poles which is impossible. Then, in general we can verify that $q_{max} = d$.

Test 3 *Consider an $n \times n$ polynomial matrix of degree d, with nullities $v_1 = v_2 = \cdots = v_q = n - 1$.*

For various values of n, d and q, we obtain the number of operations performed by the Toeplitz and the pencil algorithm. Some conclusions are as follows.

We observe that when $d = 1$ the pencil algorithm is always more efficient. The reason is that in step $q + 1$, while the pencil algorithm processes a matrix of dimension 1×1, the Toeplitz algorithm processes a matrix of dimension $n \times (2n + 1)$.

Nevertheles, when degree d increases q must be equal to d for the pencil algorithm to be more efficient. If d keeps increasing, not only q must be equal to d but also n must be very large. For example, when $d = 3$, the pencil algorithm is more efficient only when $n > 36$.

Extreme case 2: $v_k = 1$

Now let us consider that $v_k = 1$ for $k = 1, 2, \ldots, q$. In this case we can easily check that $q_{max} = nd$. For example consider a polynomial matrix $A(s)$ of degree 3 and dimension 7. If $v_k = 1$ with $q = 1, 2, \ldots, nd$, $A(s)$ has 6 poles at infinity of degree 3 and one zero at infinity of degree 18.

Test 4 *Consider an $n \times n$ polynomial matrix of degree d with nullities $v_1 = v_2 = \cdots = v_q = 1$.*

For various values of n, d and q, we obtain the number of operations performed by the Toeplitz and the pencil algorithm. Some conclusions are as follows.

Here too we observe that when $d = 1$ the pencil algorithm is always more efficient.

When d increases we observe that the pencil algorithm is more efficient only when q approaches $q_{max} = nd$. If d keeps increasing, not only q must be close to nd but also n must be large. For example, when $d = 6$ the pencil algorithm is more efficient only when $n > 5$.

General conclusion

From the results of Tests 3 and 4, where we considered the extreme cases, we can say as a general conclusion that the type of matrices for which the pencil algorithm is more efficient than the Toeplitz algorithm are matrices with a small degree d but with a structure that implies a number of steps q close to q_{max}, namely matrices with zeros at infinity of high degree. Also notice that in general n must be large. On the other hand, the type of favorable matrices for the Toeplitz algorithm are matrices with d large and q small, namely matrices with poles at infinity of high degree.

We have seen that the algorithmic complexity of the pencil algorithm depends on dimension n, degree d and expectedly on the number of steps q. Then although the matrices with n large, q large and d small are more favorable for the pencil algorithm than for the Toeplitz one, the number of operations performed by both algorithms will be comparatively large. On the other hand, we can easily find a polynomial matrix very favorable for the Toeplitz algorithm and simultaneously very unfavorable for the pencil algorithm, as we can see in the following example.

Example 1 *First consider the 40×40 polynomial matrix*

$$A(s) = \begin{bmatrix} 1 & s^2 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & s^2 \\ 0 & & & 1 \end{bmatrix}$$

which has 39 poles at infinity of degree 2 and 1 zero at infinity of degree 78. The vector of nullities is

$$v = [\underbrace{1, 1, \ldots, 1}_{80}, 0].$$

The pencil algorithm is more efficient: the number of operations are $\text{flops}_P = 324244800$ and $\text{flops}_T = 2052129600$. We can notice that although the pencil algorithm is more efficient, the number of operations that both algorithms perform is very large.

Now consider the matrix

$$A(s) = \begin{bmatrix} s^{40} & & \\ & s^{39} & \\ & & s^{39} \end{bmatrix}$$

which has 1 pole at infinity of degree 40 and 2 poles at infinity of degree 39. The vector of nullities is

$$v = [2, 0].$$

The Toeplitz algorithm is more efficient: the number of operations are $\text{flops}_P = 87946704$ and $\text{flops}_T = 2502$. We can see the great difference between the number of operations performed by the pencil algorithm and the Toeplitz algorithm.

5 Conclusions

In this paper we have improved the Toeplitz algorithm originally proposed in [5] to compute infinite structural indices of polynomial matrices. By taking advantage of the special block Toeplitz structure of the problem we have reduced the number of operations involved in the

computation. A bunch of numerical experiments indicate that the Toeplitz algorithm can be considered as an alternative, if not a competitor, to the classical pencil matrix algorithm originally proposed in [13].

Our work can be extended in various directions:

- It would be insightful to carry out a backward error anaysis (see e.g. [9] for a nice introduction) of the Toeplitz algorithm and to compare the obtained error bounds with those of the pencil algorithm.

- With the Toeplitz algorithm the whole infinite eigenstructure (structural indices + eigenvectors) of the polynomial matrix can be obtained with almost no extra computational effort. We are not aware of any straightforward modification of the pencil algorithm to obtain the eigenstructure of a polynomial matrix. It is not obvious to us how the infinite eigenstructure of a polynomial matrix relates with that of its associated matrix pencil. Besides structural indices, eigenvectors are also important when solving several problems, see [5] and references therein.

- The Toeplitz algorithm as presented in this paper can also pave the way for the development of more general structured algorithms dealing with medium-size or large polynomial matrices. We are currently studying the application of block Toeplitz algorithms to extract minimal bases for polynomial null-spaces of rank deficient polynomial matrices. It is likely that similar algorithms can be used to solve large polynomial matrix Diophantine equations arising in pole placement problems. Relationships with the block Toeplitz algorithms recently described in [7] must also be clarified.

- Finally, we plan to apply the Toeplitz algorithm to perform multivariable decoupling and solve related control problems, along the direction reported in our companion paper [16].

Acknowledgments

Juan-Carlos Zúñiga acknowledges support from the National Council of Science and Technology of Mexico (CONACYT) and from the Secretariat of Public Education of Mexico (SEP). Didier Henrion acknowledges support from the Grant Agency of the Czech Republic under Project No. 102/02/0709.

References

[1] Callier F.M. and Desoer C.A., Multivariable Feedback Systems. *Springer Verlag*, Berlin, 1982.

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

[3] Gantmacher F.R., Matrix Theory. *Chelsea Publishing*, New York, 1959.

[4] Henrion D. and Šebek M., An Algorithm for Polynomial Matrix Factor Extraction. *International Journal of Control*, Vol. 73, No. 8, pp. 686-695, 2000.

[5] Henrion D., Ruiz-León J.J. and Šebek M., Extraction of Infinite Zeros of Polynomial Matrices. *Proceedings of the IEEE Conference on Decision and Control*, pp. 4221-4226, Sydney, Australia, 2000.

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

[7] Kressner D. and Van Dooren P., Factorizations and linear system solvers for matrices with Toeplitz structure. *Slicot Working Note*, No. 2000-2, Katholieke Universiteit Leuven, Belgium, 2001.

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

[9] Petkov P.Hr., Christov N.D. and Konstantinov M.M.,. Computational Methods for Linear Control Systems. *Prentice Hall*, New York, 1991.

[10] Polyx, Ltd. The Polynomial Toolbox for Matlab. Version 2.5 released in 2000. See `www.polyx.cz`

[11] Slicot: Control and Systems Library, Working Group on Software, Katholieke Universiteit Leuven, Belgium, 2001. See `www.win.tue.nl/niconet`

[12] Van Dooren P., Dewilde P. and Vandewalle J., On the Determination of the Smith-MacMillan Form of a Rational Matrix from its Laurent Expansion. *IEEE Transactions on Circuits and Systems*, Vol. 26, No. 3, pp. 180–189, 1979.

[13] Van Dooren P., The Computation of Kronecker's Canonical Form of a Singular Pencil. *Linear Algebra and Its Applications*, Vol. 27, pp. 103–141, 1979.

[14] Van Dooren P. and Dewilde P., The Eigenstructure of an Arbitrary Polynomial Matrix. Computational Aspects. *Linear Algebra and Its Applications*, Vol. 50, pp. 545–580, 1983.

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

[16] Zúñiga J.C., Ruiz-León J.J. and Henrion D., Algorithm for decoupling and pole assignment of linear multivariable systems. *Proc. of the European Control Conference 2003*, Cambridge, U.K. 2003.