Far East J. Math. Sci. (FJMS) **26**(1) (2007), 1-24
This paper is available online at http://www.pphmj.com

ON THE COMPUTATION OF THE DRAZIN INVERSE OF A POLYNOMIAL MATRIX

N. P. KARAMPETAKIS, PREDRAG S. STANIMIROVIĆ and MILAN B. TASIĆ

Department of Mathematics
Aristotle University of Thessaloniki
Thessaloniki 54006, Greece
e-mail: karampet@ccf.auth.gr

Department of Mathematics, Faculty of Science
University of Niš
Višegradska 33, 18000 Niš
Serbia and Montenegro
e-mail: pecko@pmf.pmf.ni.ac.yu

Faculty of Technology, University of Niš
Bulevar Oslobodjenja 124, 16000 Leskovac
Serbia and Montenegro
e-mail: milan12t@ptt.yu

Abstract

There are proposed two algorithms for symbolic computation of the Drazin inverse of a given polynomial square matrix, based on the extension of the Leverrier-Faddeev algorithm. The implementation of these algorithms is developed in the symbolic package MATHEMATICA. Effectiveness of these algorithms is investigated and illustrated by appropriate examples.

2000 Mathematics Subject Classification: 68Q40, 15A09.

Keywords and phrases: polynomial matrix, Leverrier-Faddeev algorithm, generalized inverses, symbolic computation.

Received February 28, 2005

1. Introduction

Let $\mathbb{C}$ be the set of complex numbers, $\mathbb{C}^{m\times n}$ be the set of $m \times n$ complex matrices, and $\mathbb{C}_r^{m\times n} = \{X \in \mathbb{C}^{m\times n} : \text{rank}(X) = r\}$. As usual, $\mathbb{C}[s]$ denotes the polynomials with complex coefficients in the indeterminate s. The $m \times n$ matrices with elements in $\mathbb{C}[s]$ are denoted by $\mathbb{C}[s]^{m\times n}$. By I_r we denote the identity matrix of the order r, and by O is denoted an appropriate null matrix. Also, by $\text{Tr}(A)$ is denoted the trace of A.

Computation of Moore-Penrose inverse of a constant complex matrix $A(s) \equiv A_0 \in \mathbb{C}^{m\times n}$ by means of the Leverrier-Faddeev algorithm (also called Souriau-Frame algorithm) is introduced in [2]. A modification of the Leverrier-Faddeev algorithm for computation of the Drazin inverse was given in [3]. Hartwig in [4] continues investigation of this algorithm. In [5] a new proof for Grevile's algorithm is introduced.

A new extension of Leverrier's algorithm for computing the inverse of a matrix polynomial of arbitrary degree is introduced in [12]. Also, computation of the Moore-Penrose inverse of polynomial matrices, based on the Leverrier-Faddeev algorithm, has been investigated in [6], [7, 8, 9]. Moreover, in the literature it is known a large number of applications of generalized inverses of polynomial matrices [6], [7, 8, 9]. In [7] it is utilized a representation and two algorithms for computation of the Moore-Penrose inverse of a nonregular polynomial matrix of an arbitrary degree. In [6] it is described an implementation of the algorithm for computing the Moore-Penrose inverse of a singular rational matrix in the symbolic computational language MAPLE.

It is known that the Leverrier-Faddeev algorithm is ill-conditioned. This fact is a motivation for the use of symbolic programming language MATHEMATICA [13, 14] in our approach here. Main additional advantages of the symbolic implementation in MATHEMATICA are as follows. Variables can be stored without loss of accuracy during calculations. Symbolic computation is free from the truncation error. Moreover, algorithms presented for polynomial or rational matrices are applicable to significantly wider class of matrices and to a wider set of

problems, with respect to algorithms intended for constant matrices. Also algorithms applicable to polynomial or rational matrices can be used in the construction of test matrices and in the verification of some hypothesis. Finally, these algorithms can be verified directly on test matrices. But, symbolic implementation possesses some disadvantages. It is known that the formula manipulation by a computer requires much more time and memory space with respect to the numerical implementation. Also, symbolic packages are in general "slow".

In the second section we restate well-known modification of the Leverrier-Faddeev algorithm for computing the Drazin inverse, introduced in [3] and alternatively verified in [5]. In the third section it is presented an extension of the Grevile's modification of the Leverrier-Faddeev algorithm, to the set of one-variable nonregular polynomial matrices. We also propose two algorithms for computing the Drazin inverse of one-variable polynomial matrix. In the fourth section we introduce a quick and effective in memory requirements algorithm for the matrices which have greater gaps between the powers of unknown variables. In Section 5 we describe implementation details of introduced algorithms. In Section 6 we present illustrative examples and compare introduced algorithms.

2. Preliminaries

In this section we consider polynomial matrices $A(s) \in \mathbb{C}[s]^{n\times n}$. In [10] the following statement is proved.

Lemma 2.1 [10]**.** *Consider a nonregular one-variable polynomial matrix* $A(s)$*. Assume that*

$$a(z, s) = \det[zI_n - A(s)]$$

$$= a_0(s)z^n + a_1(s)z^{n-1} + \cdots + a_{n-1}(s)z + a_n(s),\ \ a_0(s) \equiv 1,\ \ z \in \mathbb{C}$$

is the characteristic polynomial of $A(s)$*. Also, consider the following sequence of* $n \times n$ *polynomial matrices*

$$B_j(s) = a_0(s)A(s)^j + a_1(s)A(s)^{j-1} + \cdots + a_{j-1}(s)A(s) + a_j(s)I_n,$$

$$a_0(s) = 1, \quad j = 0, \ldots, n.$$

Let

$$a_n(s) \equiv 0, \ldots, a_{t+1}(s) \equiv 0, \quad a_t(s) \neq 0.$$

Define the following set:

$$\Lambda = \{s_i \in \mathbb{C} : a_t(s_i) = 0\}.$$

Also, assume

$$B_n(s) \equiv O, \ldots, B_r(s) \equiv O, \quad B_{r-1}(s) \neq O$$

and $k = r - t$.

In the case $s \in \mathbb{C}\backslash\Lambda$ *and* $k > 0$, *the Drazin inverse of* $A(s)$ *is given by*

$$A(s)^D = (-1)^{k+1} a_t(s)^{-k-1} A(s)^k B_{t-1}(s)^{k+1}.$$

In the case $s \in \mathbb{C}\backslash\Lambda$ *and* $k = 0$, *we get* $A(s)^D = A(s)^{-1}$.

For $s_i \in \Lambda$ *denote by* t_i *the largest integer satisfying* $a_{t_i}(s_i) \neq 0$, *and by* r_i *the smallest integer satisfying* $B_{r_i}(s_i) \equiv O$. *Then the Drazin inverse of* $A(s_i)$ *is equal to*

$$A(s_i)^D = (-1)^{k_i+1} a_{t_i}(s_i)^{-k_i-1} A(s_i)^{k_i} B_{t_i-1}(s_i)^{k_i+1},$$

where $k_i = r_i - t_i$.

In view of the results of Lemma 2.1 we present the following algorithm for computation of the Drazin inverse [11].

Algorithm 2.1. It is assumed that $A(s) \in \mathbb{C}[s]^{n\times n}$ is polynomial matrix.

Step 1. Construct the sequence of polynomials $\{a_0(s), a_1(s), \ldots, a_n(s)\}$ and the sequence of polynomial matrices $\{B_0(s), B_1(s), \ldots, B_n(s)\}$ as in the following:

$$\begin{array}{lll} A_0(s) = O, & a_0(s) = 1, & B_0(s) = I_n \\ A_1(s) = A(s)B_0(s), & a_1(s) = -\dfrac{\mathrm{Tr}(A_1(s))}{1}, & B_1(s) = A_1(s) + a_1(s)I_n \\ \cdots & \cdots & \cdots \\ A_n(s) = A(s)B_{n-1}(s), & a_n(s) = -\dfrac{\mathrm{Tr}(A_n(s))}{n}, & B_n(s) = A_n(s) + a_n(s)I_n. \end{array} \tag{2.1}$$

Step 2. Let

$$t = \max\{l : a_l(s) \neq 0\}, \quad r = \min\{l : B_l(s) = O\}, \quad k = r - t.$$

For $s \in \mathbb{C}\backslash\Lambda$ the Drazin inverse $A(s)^D$ is given by

$$A(s)^D = (-1)^{k+1} a_t(s)^{-k-1} A(s)^k B_{t-1}(s)^{k+1}. \tag{2.2}$$

Output: For those $s_i \in \Lambda$, denote by t_i the largest integer satisfying $a_{t_i}(s_i) \neq 0$, and by r_i the smallest integer satisfying $B_{r_i}(s_i) \equiv O$. For the integer $k_i = r_i - t_i$, the Drazin inverse of $A(s_i)$ is equal to

$$A(s_i)^D = (-1)^{k_i+1} a_{t_i}(s_i)^{-k_i-1} A(s_i)^{k_i} B_{t_i-1}(s_i)^{k_i+1}.$$

3. Drazin Inverse of a Polynomial Matrix

Now, we assume that $A(s) \in \mathbb{C}[s]^{n\times n}$ is a polynomial matrix of the form

$$A(s) = A_q s^q + A_{q-1} s^{q-1} + \cdots + A_1 s + A_0 \in \mathbb{C}[s]^{n\times n}, \tag{3.1}$$

where $A_i \in \mathbb{C}^{n\times n}$, $i = 0, ..., q$ are constant complex matrices and s is an unknown variable. The coefficients $a_i(s)$ can be considered as polynomials of s. Also, matrix coefficients $B_i(s)$ can be considered as polynomial matrices with respect to powers of the variable s. We give an analogous representation of the Drazin inverse and an analogous algorithm for computing the Drazin inverse.

Theorem 3.1. *Let the polynomial matrix* $A(s)$ *be of the form* (3.1). *Consider the polynomials* $\{a_0(s), a_1(s), ..., a_n(s)\}$ *and the sequence of polynomial matrices* $\{B_0(s), B_1(s), ..., B_n(s)\}$, *defined as in Algorithm* 2.1. *Rewrite* $a_i(s)$ *and* $B_i(s)$ *as*

$$a_i(s) = \sum_{j=0}^{iq} a_{i,j} s^j, \quad i = 1, ..., n, \tag{3.2}$$

$$B_i(s) = \sum_{j=0}^{iq} B_{i,j} s^j, \quad i = 1, ..., n, \tag{3.3}$$

where $a_{i,j}$, $i = 1, ..., n$, $j = 0, ..., iq$ *are scalars and* $B_{i,j}$, $i = 1, ..., n$, $j = 0, ..., iq$ *are constant coefficient matrices corresponding to powers* s^j. *Then we have*

$$a_{i+1,j} = -\frac{1}{i+1}\operatorname{Tr}\left(\sum_{l=0}^{j} A_{j-l}B_{i,l}\right),$$
$$i = 0, 1, ..., n-1, \quad j = 0, 1, ..., (i+1)q, \tag{3.4}$$

$$B_{i+1,j} = \sum_{l=0}^{j} A_{j-l}B_{i,l} + a_{i+1,j}I_n,$$
$$i = 0, 1, ..., n-1, \quad j = 0, 1, ..., (i+1)q, \tag{3.5}$$

where in (3.4) *and* (3.5) *it is assumed* $A_k = O$, $k \geq q+1$, $B_{ik} = O$, $k \geq iq+1$.

Proof. It is not difficult to verify from (2.1) that the greatest powers of $A_i(s)$ (and thus of $B_i(s)$) are equal to iq, $i = 0, ..., n-1$. Also, the degree of the polynomial quantities $a_i(s)$, $i = 1, ..., n$ is at most equal to iq. Hence $a_i(s)$ and $B_i(s)$ can be written in the form (3.2) and (3.3), respectively. Also, from (3.1), (2.1) and (3.3), we get the following:

$$A_{i+1}(s) = A(s)B_i(s) = \left(\sum_{j=0}^{q} A_j s^j\right)\left(\sum_{l=0}^{iq} B_{i,l}s^l\right)$$
$$= \sum_{j=0}^{(i+1)q}\left(\sum_{l=0}^{j} A_{j-l}B_{i,l}\right)s^j. \tag{3.6}$$

An application of (2.1) and (3.6) leads to

$$a_{i+1}(s) = -\frac{1}{i+1}\operatorname{Tr}(A_{i+1}(s))$$
$$= \sum_{j=0}^{(i+1)q}\left[-\frac{1}{i+1}\operatorname{Tr}\left(\sum_{l=0}^{j} A_{j-l}B_{i,l}\right)\right]s^j. \tag{3.7}$$

The identity (3.4) follows from (3.2) and (3.7).

On the other hand, using (2.1), (3.7) and (3.2), we get

$$\begin{aligned} B_{i+1}(s) &= A_{i+1}(s) + a_{i+1}(s) I_n \\ &= \sum_{j=0}^{(i+1)q} \left(\sum_{l=0}^{j} A_{j-l} B_{i,l} \right) s^j + \sum_{j=0}^{(i+1)q} a_{i+1,j} I_n s^j \\ &= \sum_{j=0}^{(i+1)q} \left(\sum_{l=0}^{j} A_{j-l} B_{i,l} + a_{i+1,j} I_n \right) s^j. \end{aligned} \tag{3.8}$$

The identity (3.5) follows from (3.3) and (3.8).

Now we are in a position to state the following algorithm for computation of the Drazin inverse $A(s)^D$ for the polynomial matrix $A(s)$ of the form (3.1).

Algorithm 3.1.

Initial conditions:

$$B_{0,0} = I_n, \quad A_k = 0, \quad k = q+1, \ldots, nq.$$

Boundary conditions:

$$B_{0,j} = O \quad \forall j \in \mathbb{N},$$

$$B_{i,j} = O, \quad i = 0, \ldots, n-1, \; j = iq+1, \ldots, (n-1)q.$$

Main algorithm:

$i = 0$

DO WHILE at least one $B_{i,j} <> 0$, $j = 0, 1, \ldots, (i+1)q$:

Initial conditions for $B_{i,j}$:

$$B_{i,j} = O, \quad j = iq+1, \ldots, (n-1)q.$$

Recursive relations for $a_i(s)$:

$$a_{i+1,j} = -\frac{1}{i+1} \operatorname{Tr}\left(\sum_{l=0}^{j} A_{j-l} B_{i,l} \right), \; i = 0, 1, \ldots, n-1, \; j = 0, 1, \ldots, (i+1)q.$$

Recursive relations for $B_i(s)$:

$$B_{i+1,\,j} = \sum_{l=0}^{j} A_{j-l}B_{i,\,l} + a_{i+1,\,j}I_n,\ i = 0,\ 1,\ \ldots,\ n-1,\ j = 0,\ 1,\ \ldots,\ (i+1)q.$$

$i = i + 1.$

END DO.

Termination criteria: Compute first t satisfying

$$a_{t+1,\,j} = a_{t+2,\,j} = \cdots = a_{n,\,j} = 0 \quad \forall j \in \mathbb{N}.$$

and first r satisfying

$$B_{r,\,j} = O, \quad j = 0,\ 1,\ \ldots,\ rq.$$

Output: Compute $k = r - t$ and return the rational matrix

$$A(s)^D = (-1)^{k+1}\left(\sum_{j=0}^{tq} a_{r-1,\,j}s^j\right)^{-k-1}\left(\sum_{i=0}^{q} A_i s^i\right)^k\left(\sum_{l=0}^{(t-1)q} B_{t-1,\,l}s^l\right)^{k+1}. \quad (3.9)$$

Example 3.1. Consider the polynomial matrix

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

Applying Algorithm 3.1 we have that

$$A_0(s) = 0_{3,\,3};\quad a_0(s) = 1;\quad B_0(s) = I_3;$$

$$A_1(s) = A(s)B_0(s) = A(s);\quad a_1(s) = -\frac{\mathrm{Tr}(A_1(s))}{1!} = -(1+3s);$$

$$B_1(s) = A_1(s) + a_1(s)I_3 = \begin{bmatrix} -2s & s & 1+s \\ s & -2-2s & s \\ 1+s & s & -2s \end{bmatrix};$$

$$A_2(s) = A(s)B_1(s) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix};\quad a_2(s) = -\frac{\mathrm{Tr}(A_2(s))}{2!} = -\frac{4}{2} = -2;$$

$$B_2(s) = A_2(s) + a_2(s)I_3 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix};$$

$$A_3(s) = A(s)B_2(s) = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} = 0_{3,3};\ a_3(s) = -\frac{\mathrm{Tr}(A_3(s))}{3!} = 0.$$

Thus

$$t = \max\{l : a_l(s) \neq 0\} = 2, \quad r = \min\{l : B_l(s) = O\} = 3, \quad k = r - t = 1.$$

Therefore

$$A(s)^D = a_2(s)^{-2} A(s)^1 B_1(s)^2$$

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

$$= \begin{bmatrix} \frac{1}{4} - \frac{1}{4}s & \frac{1}{2}s & \frac{1}{4} - \frac{1}{4}s \\ \frac{1}{2}s & -1-s & \frac{1}{2}s \\ \frac{1}{4} - \frac{1}{4}s & \frac{1}{2}s & \frac{1}{4} - \frac{1}{4}s \end{bmatrix}.$$

4. Effective Computation of the Drazin Inverse

It is easily checked that the number of coefficients a_{ij} and the matrices $B_{i,j}$ embedded in the evaluation of the Drazin inverse $A(s)^D$ of $A(s) \in A[s]^{n \times n}$ is equal to

$$\sum_{i=1}^{n} (iq + 1) = \frac{n(n+1)}{2} q + n.$$

Thus, even in case where there is only one big power of s in $A(s)$, for example s^{80}, then the number of matrices used for the evaluation of the

Drazin inverse of $A(s)$ is $\frac{n(n+1)}{2}80+n$, regardless to the absence of the powers s^{79}, ..., s. In that case where big gaps are existing between the powers of s in $A(s)$, we propose in the sequel an improved algorithm.

Define as:

$\Phi_A = \{(\mu_i)$: the set of degrees of nonzero coefficient matrices of $A(s)\}$,

$\Phi_A(i) =$ the ith element of Φ_A (let (μ_i)),

$n_A = q =$ the total number of elements in Φ_A.

Then the polynomial matrix $A(s) \in \mathbb{C}[s]^{n\times n}$ can be represented in the form

$$A(s) = \sum_{i=1}^{q} A_{\Phi_A(i)} s^{\Phi_A(i)} = \sum_{i=1}^{q} A_{\mu_i} s^{\mu_i}, \tag{4.1}$$

where $A_{\Phi_A(i)} = A_{\mu_i} \in \mathbb{C}^{n\times n}$, $i = 0, ..., q$ are constant nonzero complex matrices in (3.1). We also rewrite $a_i(s)$ and $B_i(s)$ as in the following:

$$a_i(s) = \sum_{j=1}^{n_i} a_{i,\Phi_i(j)} s^{\Phi_i(j)} \tag{4.2}$$

$$B_i(s) = \sum_{j=1}^{n_i} B_{i,\Phi_i(j)} s^{\Phi_i(j)}. \tag{4.3}$$

Then we have that

$$A_{i+1}(s) = A(s)B_i(s) = \left(\sum_{j=1}^{n_A} A_{\Phi_A(j)} s^{\Phi_A(j)}\right)\left(\sum_{k=1}^{n_i} B_{i,\Phi_i(k)} s^{\Phi_i(k)}\right)$$

$$= \sum_{j=1}^{n_A}\sum_{k=1}^{n_i} A_{\Phi_A(j)} B_{i,\Phi_i(k)} s^{\Phi_A(j)+\Phi_i(k)}$$

$$= \sum_{j=1}^{n_A}\sum_{k=1}^{n_i} Q_{\Phi_A(j)+\Phi_i(k)} s^{\Phi_A(j)+\Phi_i(k)}.$$

Hence

$$\Phi_{i+1} = \Phi_{i+1} \cup \{\Phi_A(j) + \Phi_i(k)\},\ i = 1, ..., n-1,\ j = 1, ..., n_A,\ k = 1, ..., n_i.$$

We subtract from the set Φ_{i+1}, $i = 1, ..., n-1$ those degrees $\Phi_{i+1}(j)$ which correspond to zero products $A_{\Phi_A(j)}B_{i,\Phi_i(k)}$ and form the new set Φ_{i+1} with total number of elements n_{i+1} instead of $\tilde{n}_{i+1}$ which the previous Φ_{i+1} had.

Substituting (4.1), (4.2) and (4.3) in the recursive relations (2.1) we obtain recursive algorithm that determines $a_{i+1,\Phi_{i+1}(j)}$ and $B_{i+1,\Phi_{i+1}(j)}$ for $j = 1, ..., n_i$.

Algorithm 4.1. (Computation of the Drazin inverse of $A(s)$).

Initialization:

$$B_{0,0} = I_n.$$

Boundary conditions:

$\Phi_i = \{0\}$, $n_i = 1$, $i = 0, 1, ..., n$,

n_E = maximal exponent for s in $A(s)$,

$Q_i = O$, $i = 0, 1, ..., 2n_E + 1$,

$\Phi_A = \{\Phi_A(1), ..., \Phi_A(n_A)\}$ = the set of nonzero coefficient matrices of $A(s)$,

$n_A = q$ = the total number of elements in Φ_A.

Main program:

$i = 0$

DO WHILE $\Phi_i \neq \varnothing \wedge i < n$.

Step 1. Compute:

(a) the coefficient matrix which correspond to the $\Phi_A(j) + \Phi_i(k)$-degree of s in $A(s)B_i(s)$, and

(b) the set Φ_{i+1} in terms of Φ_A and Φ_i

$$Q_{\Phi_A(j)+\Phi_i(k)} = Q_{\Phi_A(j)+\Phi_i(k)} + A_{\Phi_A(j)}B_{i,\Phi_i(k)},$$

$$\Phi_{i+1} = \Phi_{i+1} \cup \{\Phi_A(j) + \Phi_i(k)\},\ j = 1, ..., n_A,\ k = 1, ..., n_i.$$

Step 2. Compute the total number of elements in Φ_{i+1}

$$\tilde{n}_{i+1} = \text{ total number of elements in } \Phi_{i+1}$$

Step 3. Computation of $a_{i+1,\Phi_{i+1}(j)}$ and $B_{i+1,\Phi_{i+1}(j)}$:

Set $s = 0$

For $j = 1, ..., \tilde{n}_{i+1}$ perform the following:

If $Q_{\Phi_{i+1}(j)} = O$, then execute the following:

$$\Phi_{i+1} = \Phi_{i+1} - \{\Phi_{i+1}(j)\},\quad s = s + 1,$$

else compute

$$a_{i+1,\Phi_{i+1}(j)} = -\frac{1}{i+1}\operatorname{Tr}(Q_{\Phi_{i+1}(j)}).$$

end if

If $i < n - 1$, then

$$B_{i+1,\Phi_{i+1}(j)} = Q_{\Phi_{i+1}(j)} + a_{i+1,\Phi_{i+1}(j)}I_r$$

end if

Set $Q_{\Phi_{i+1}(j)} = O$

Next j

$n_{i+1} = \tilde{n}_{i+1} - s$

$i = i + 1$

END DO

Terminate:

Find $t = \max\{l : a_{l,\Phi_l(j)} \neq 0\}$ and $r = \min\{l : B_l(s) = O\}$, i.e., find

$$t : a_{t+1,\Phi_{t+1}(j)} = a_{t+2,\Phi_{t+2}(j)} = \cdots = a_{n_i,\Phi_{n_i}(j)} = 0,$$

$$r : B_{r,\Phi_r(i)} = 0,\ i = 1, ..., n_i.$$

For $k = r - t$ place

$$B_{\Phi_{t-1}(j)} = B_{t-1,\Phi_{t-1}(j)},\quad j = 1, ..., n_{t-1},$$

$$a_{\Phi_t(j)} = a_{t,\Phi_t(j)},\quad j = 1, ..., n_t.$$

Output:

$$A(s)^D = (-1)^{k+1}\left(\sum_{j=1}^{n_t} a_{\Phi_t(j)} s^{\Phi_t(j)}\right)^{-k-1}\left(\sum_{i=1}^{q} A_{\Phi_A(i)} s^{\Phi_A(i)}\right)^{k}$$

$$\times\left(\sum_{l=1}^{n_{t-1}} B_{\Phi_{t-1}(l)} s^{\Phi_{t-1}(l)}\right)^{k+1}.$$

Remark 4.1. The number of needed computations of coefficients $a_{i,\Phi_i(j)}$ and coefficient matrices $B_{i,\Phi_i(j)}$ is equal to $\sum_{j=1}^{q} n_i$, regardless of values n and q.

Example 4.1. Consider the polynomial matrix $A(s)$ as in Example 3.1:

$$A(s) = \begin{bmatrix} 1+s & s & 1+s \\ s & -1+s & s \\ 1+s & s & 1+s \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} s + \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = A_1 s + A_0.$$

Applying Algorithm 4.1 we get:

Initialization:

$$B_{0,0} = I_n = I_3.$$

Boundary conditions:

$$\Phi_i = \{0\},\quad n_i = 1, \text{ for } i = 0, 1, 2, 3,$$

$$\Phi_A = \{0, 1\},\quad \Phi_A(1) = 0,\quad \Phi_A(2) = 1,$$

$$n_A = 2 = \text{the total number of elements in } \Phi_A,$$

$$Q_i = O_{3,3}, \quad i = 0, 1.$$

Step 1. Apply for $i = 0, 1, 2$ the following steps.

$i = 0$

Step 1.1.

$$Q_0 = Q_0 + A_0 B_{0,0} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \ (j = 1,\ k = 1);$$

$$Q_1 = Q_1 + A_1 B_{0,0} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \ (j = 2,\ k = 1);$$

$$\Phi_1 = \Phi_1 \cup \{0, 1\} = \{0, 1\}.$$

Step 1.2. Computation of the total number of elements in Φ_1

$$\tilde{n}_1 = 2 = \text{total number of elements in } \Phi_1.$$

Step 1.3. Computation of $a_{1,\Phi_1(j)}$ and $B_{1,\Phi_1(j)}$:

Set $s = 0$

$$Q_0 \neq O_{3,3} \Rightarrow a_{1,0} = -\frac{1}{1}\mathrm{Tr}(Q_0) = -1 \ \ (j = 1);$$

$$B_{1,0} = Q_0 + a_{1,0} I_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{bmatrix};$$

$$Q_1 \neq O_{3,3} \Rightarrow a_{1,1} = -\frac{1}{1}\mathrm{Tr}(Q_1) = -3 \ \ (j = 1);$$

$$B_{1,1} = Q_1 + a_{1,1} I_3 = \begin{bmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix};$$

$$n_1 = \tilde{n}_1 - s = 2 - 0 = 2.$$

We vanish the entries of Q_i:

$$Q_0 = Q_1 = O_{3,3}$$

$i = 1$.

Step 1.1.

$$Q_0 = Q_0 + A_0 B_{1,0} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (j = 1,\ k = 1);$$

$$Q_1 = Q_1 + A_0 B_{1,0} = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad (j = 1,\ k = 2);$$

$$Q_1 = Q_1 + A_1 B_{1,0} = O_{3,3} \quad (j = 2,\ k = 1);$$

$$Q_2 = Q_2 + A_1 B_{1,1} = O_{3,3} \quad (j = 2,\ k = 2);$$

$$\Phi_2 = \Phi_2 \cup \{0, 1, 2\} = \{0, 1, 2\}.$$

Step 1.2. Computation of the total number of elements in Φ_2.

$$\tilde{n}_2 = 3 = \text{total number of elements in } \Phi_2$$

Step 1.3. Computation of $a_{2,\Phi_2(j)}$ and $B_{2,\Phi_2(j)}$:

Set $s = 0$

$$Q_0 \neq O_{3,3} \Rightarrow a_{2,0} = -\frac{1}{2}\mathrm{Tr}(Q_0) = -2 \quad (j = 1);$$

$$B_{2,0} = Q_0 + a_{2,0} I_3 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix};$$

$$Q_1 = O_{3,3} \Rightarrow \Phi_2 = \Phi_2 - \{1\} = \{0, 2\},\ s = s + 1 = 1 \quad (j = 2);$$

$$Q_2 = O_{3,3} \Rightarrow \Phi_2 = \Phi_2 - \{2\} = \{0\},\ s = s + 1 = 2 \quad (j = 3);$$

$$n_2 = \tilde{n}_2 - s = 3 - 2 = 1.$$

Vanish the entries of Q_i:

$$Q_0 = O_{3,3}.$$

$i = 2$.

Step 1.1.

$$Q_0 = Q_0 + A_0B_{2,0} = O_{3,3} \quad (j = 1,\ k = 1);$$

$$Q_1 = Q_1 + A_1B_{1,0} = O_{3,3} \quad (j = 2,\ k = 1);$$

$$\Phi_3 = \Phi_3 \bigcup \{0, 1\} = \{0, 1\}.$$

Step 1.2. Computation of the total number of elements in Φ_3

$$\widetilde{n}_3 = 2 = \text{total number of elements in } \Phi_3$$

Step 1.3. Computation of $a_{3,\Phi_3(j)}$ and $B_{3,\Phi_3(j)}$:

Set $s = 0$

$$Q_0 = O_{3,3} \Rightarrow \Phi_3 = \Phi_3 - \{0\} = \{1\},\ s = s + 1 = 1\ (j = 1);$$

$$Q_1 = O_{3,3} \Rightarrow \Phi_3 = \Phi_3 - \{1\} = \varnothing,\ s = s + 1 = 2\ (j = 2);$$

$$n_3 = \widetilde{n}_3 - s = 2 - 2 = 0.$$

$i = 3$.

$$\Phi_3 = \varnothing \text{ therefore } \textit{END DO}.$$

Terminate:

$$t = 2 = \max\{l,\ a_{l,\Phi_l(j)} \neq 0\},\ r = i = 3 = \min\{l : B_l(s) = O\}.$$

Place

$$B_0 = B_{1,0},\ \ B_1 = B_{1,1}\ \ a_0 = a_{2,0}.$$

OUTPUT: For $k = r - t = 3 - 2 = 1$ the Drazin inverse of $A(s)$ is

$$A(s)^D = (-1)^2 \frac{A(s)^1 B_{2-1}(s)^{1+1}}{a_2(s)} = \frac{(A_0 + A_1s)^1 (B_0 + B_1s)^2}{a_2(s)}$$

$$= \begin{bmatrix} -\frac{1}{4}s + \frac{1}{4} & \frac{1}{2}s & -\frac{1}{4}s + \frac{1}{4} \\ \frac{1}{2}s & -1 - s & \frac{1}{2}s \\ -\frac{1}{4}s + \frac{1}{4} & \frac{1}{2}s & -\frac{1}{4}s + \frac{1}{4} \end{bmatrix}.$$

5. Implementation

In this section we briefly describe the implementation of Algorithm 3.1 and Algorithm 4.1 in the package MATHEMATICA.

The polynomial matrix of the form (3.1) is represented by the three-dimensional list $\{l_1, ..., l_{q+1}\}$, where the element l_i denotes the matrix A_{i-1}, $i = 1, ..., q+1$. Elements of the global array A are equal to

$$A[[i]] = l_{i-1}, \quad i = 0, ..., q, \qquad A[[i]] = O, \quad i = q+1, ..., nq.$$

Recursive relations (3.4) and (3.5) for computing the real numbers $a_{i,j}$ and the matrices $B_{i,j}$, respectively, are implemented in the following two functions.

```
Fa[i1_,j1_] := -1/i1*Tr[Sum[A[[j1-l+1]].FB[i1-1,l],l,0,j1]]

FB[u_,v_] := Block[{w={ }},
  nul=Table[0, {n}, {n}];
  If[u==0 && v==0, w=IdentityMatrix[n],
  If[u==0, w=nul,
    If[(v>=u q+l) && (v<=(n-1) q), w=nul,
      w=Sum[A[[v-k+1]].FB[u-1,k], {k,0,v}]+Fa[u,v]*IdentityMatrix[n]]
    ];
  ];
  Return[w]
];
```

By means of the auxiliary function $FrmA[a\_List, var\_]$ it is possible to perform a transformation of the polynomial matrix $A[var]$ into the internal representation of the form $\{l_1, ..., l_{q+1}\}$, where the list l_i

represents the matrix A_{i-1}, $i = 1, ..., q+1$.

```
FrmA[a_List,var_] :=
  Block[{L={ },L1,nul,n,n1,m1,i},
    {n1,m1}=Dimensions[a]; nul=Table[0,{i,n1},{j,m1}]; L1=nul;
    n=Max[Exponent[a,var,List]];
    For[i=1,i<=n,i++,
      koef=Coefficient[a,var^i];
      L=Append[L,koef]; L1=L1+koef*var^i];
    If[L==={ }, Return[{a-L1}] ];
    L1={a-L1};
    For[i=1,i<=Length[L],i++, AppendTo[L1,L[[i]]
    ]];
    Return[L1]
  ]
```

The method for computing the Drazin inverse which is described in Algorithm 3.1, is implemented in the following function *DrzPoly*[*mat_List*], where it is assumed that the formal parameter *mat_* denotes the three-dimensional list of the form $\{A_0, ..., A_q\}$.

```
DrzPoly[mat_List] :=
  Block[{A,i,j,nul,k,q,r=t=0,log1=log2=True,var=Variables[mat]},
    If[var=={ }, A={mat}, A=FrmA[mat,First[var]]];
    q=Length[A]-1; {n,n}=Dimensions[A[[1]]]; nul=Table[0,{n},{n}];
    For[i=q+1, i<=n q, i++, A=Append[A, nul]];
    For[i=0, i<=n-1, i++,
      For[j=0, j<=(i+1) q, j++,
          If[Fa[i+1,j]==0, log1=log1 && True, log1=log1 && False];
        If[FB[i+1,j]===nul, log2=log2 && True, log2=log2 && False]];
      If[!log1, t=i+1; log1=True]; If[!log2, r=i+1]
    ];
    k=r-t;
```

```
  rez1=(-1)^(k+1)*Sum[Fa[t, j]*First[var]^j, {j,0,t q}]^(-k-1);
  rez2=MatrixPower[Sum[A[[i+1]] First[var]^i, {i,0,q}],k];
  rez3=MatrixPower[Sum[FB[t-1,l] First[var]^l, {l,0,(t-1)q}], k+1];
  rez = rez1*rez2.rez3;
  Return[Simplify[rez]]
]
```

The method for computing the Drazin inverse which is described in Algorithm 4.1, is implemented in the following function $DrzEf[mat\_List]$, where it is assumed that the formal parameter $mat\_$ represents a three-dimensional list $A1 = FrmA[mat, First[Variables[mat]]]$ which contains coefficient matrices A_0, ..., A_q from (3.1).

```
DrzEf[mat_List] :=
  Module[{A1,i,j,ne,na,s,r=t=0,log1=log2=True,var=Variables[mat],
  FA,FK,m,n},
    If[var=={ }, A1=mat, A1=FrmA[mat, First[var]]];
    {n,m}=Dimensions[mat];
    nul=Table[0, {i,n}, {j,n}]; FA={ };
    ne=Max[Exponent[mat, First[var]]];
    For[i=0, i<Length[A1], i++,
      If[A1[[i+1]]=!=nul, AppendTo[FA, i]; A[i]=A1[[i+1]]]];
    na=Length[FA];
    For[i=0, i<=2*ne+1, i++, Q[i]=nul];
    For[i=0, i<=n, i++, F[i]={0}; ni[i] = 1];
    i=0; B[0,0]=IdentityMatrix[n];
    While[(F[i]=!={ }) && (i<n),
      For[j=1, j<=Length[FA], j++,
        For[k = 1, k <= ni[i], k++,
          Q[FA[[j]]+F[i][[k]]]=Q[FA[[j]]+F[i][[k]]]+A[FA[[j]]].B[i,
          F[i][[k]]];
          F[i+1]=Union[F[i+1], {FA[[j]]+F[i][[k]]}]]];
      nk[i+1]=Length[F[i+1]];
```

```
  s=0; FK={ };
  For[j=1, j<=nk[i+1], j++,
    If[Q[F[i+1][[j]]]===nul,
      FK=Append[FK,F[i+1][[j]]];
      s=s+1,
      a[i+1, F[i+1][[j]]]=-1/(i+1)*Tr[Q[F[i+1][[j]]]];
      If[a[i+1, F[i+1][[j]]]===0, log1=log1 && True, log1=log1 &&
      False];
      If[i<(n-1),
        B[i+1,F[i+1][[j]]]=Q[F[i+1][[j]]]+a[i+1,F[i+1][[j]]]*
        IdentityMatrix[n];
        If[B[i+1,F[i+1][[j]]]===nul, log2=log2 && True, log2=log2
        && False];
        ];
      Q[F[i+1][[j]]]=nul]];
    F[i+1]=Complement[F[i+1],FK];
    If[log1, t=i, log1=True];
    If[log2, r=i+1, log2=True];
    ni[i+1]=nk[i+1]-s; i=i+1];
  k=r-t;
  If[t==0, Return[Simplify[Inverse[mat]]]];
  rez1=(-1)^(k+1)*Sum[a[t,F[t][[j]]]*First[var]^F[t][[j]],
  {j,1,ni[t]}]^(-k-1);
  rez2=MatrixPower[Sum[A[FA[[i]]]*First[var]^FA[[i]],{i,1,na}],
  k];
  rez3=MatrixPower[Sum[B[t-1,F[t-1][[l]]]*First[var]^F[t-1][[l]],
  {l,1,ni[t-1]}], k+1];
  Clear[Q]; Clear[B]; Clear[ni];
  Clear[nk]; Clear[F]; Clear[a];
  Return[MatrixForm[Simplify[rez1*rez2.rez3]]];
]
```

6. Experimental Results

Example 6.1. The polynomial matrix

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

$$= \begin{bmatrix} 1+s & s & 1+s \\ s^2 & -1+s & s \\ 1+s & s & 1+s \end{bmatrix}$$

is represented by the following three-dimensional list, generated by the expression $mat = FrmA[a, s]$:

{{{1, 0, 1}, {0, –1, 0}, {1, 0, 1}}, {{1, 1, 1}, {0, 1, 1}, {1, 1, 1}},

{{0, 0, 0}, {1, 0, 0}, {0, 0, 0}}}.

Applying the program $DrzPoly[mat]$, we get $t = 2$, $r = 3$ and the following Drazin inverse of $A(s)$:

$$A(s)^D = \begin{bmatrix} \dfrac{1-s+2s^3-2s^4}{(2-s^2+s^3)^2} & \dfrac{s}{2-s^2+s^3} & \dfrac{1-s-s^2+s^4}{(2-s^2+s^3)^2} \\ \dfrac{s(-1+s^2+s^3)}{(1+s)(2-2s+s^2)^2} & \dfrac{2}{2-2s+s^2} & \dfrac{3s-2s^3}{(1+s)(2-2s+s^2)^2} \\ \dfrac{1-s+2s^3-2s^4}{(2-s^2+s^3)^2} & \dfrac{s}{2-s^2+s^3} & \dfrac{1-s-s^2+s^4}{(2-s^2+s^3)^2} \end{bmatrix}.$$

Note that we obtain the following values for the coefficients $a_{i,j}$:

$a[1, 0] = -1$; $a[1, 1] = -3$; $a[1, 2] = 0$; $a[2, 0] = -2$; $a[2, 1] = 0$;

$a[2, 2] = 1$; $a[2, 3] = -1$; $a[2, 4] = 0$; $a[3, 0] = a[3, 1] = \cdots = a[3, 6] = 0$.

and the following values for the matrices $B_{i,j}$:

$B[1, 0] = \{\{0, 0, 1\}, \{0, -2, 0\}, \{1, 0, 0\}\}$; $B[1, 1] = \{\{-2, 1, 1\}, \{0, -2, 1\}, \{1, 1, -2\}\}$;

$B[1, 2] = \{\{0, 0, 0\}, \{1, 0, 0\}, \{0, 0, 0\}\}$; $B[2, 0] = \{\{-1, 0, 1\}, \{0, 0, 0\}, \{1, 0, -1\}\}$;

$B[2,1] = \{\{0, 0, 0\}, \{1, 0, -1\}, \{0, 0, 0\}\}$; $B[2, 2] = \{\{0, 0, 0\}, \{0, 0, 0\}, \{-1, 0, 1\}\}$;

$B[2, 3] = \{\{0, 0, 0\}, \{-1, 0, 1\}, \{1, 0, -1\}\}$; $B[2, 4] = \{\{0, 0, 0\}, \{-1, 0, 1\}, \{1, 0, -1\}\}$;

$B[3, 0] = B[3, 1] = \cdots = B[3, 6] = \{\{0, 0, 0\}, \{0, 0, 0\}, \{0, 0, 0\}\}$.

Example 6.2. In the following example we show effectiveness of Algorithm 4.1 with respect to Algorithm 3.1. Consider the matrix

$$B(s) = \begin{bmatrix} 1+s & s & 1+s \\ s^q & -1+s & s \\ 1+s & s & 1+s \end{bmatrix}$$

for various values of the parameter q. We show the effectiveness with respect to CPU time and the elimination of redundant computations. The number of needful values $a_{i,j}$ and matrices $B_{i,j}$, required by the functions $DrzPoly[B(s)]$ and $DrzEf[B(s)]$, is arranged in columns 2 and 3 in Table 1. Also, the CPU time is represented in columns 4 and 5, respectively. These results are obtained using version 4.1 for MATHEMATICA and Pentium III 1.2 GHz.

Note that the streaks in the table mean a very long timing. Let us observe that in the case $q = 1$ the number of needful values $a_{i,j}$ and matrices $B_{i,j}$, required by the functions $DrzPoly[B(s)]$ and $DrzEf[B(s)]$ is 9 and 3, respectively. Thus we conclude that Algorithm 4.1 gives quicker and less expensive in memory use results, even there is no gap between the degrees of the polynomial matrices. However, it uses more controls than Algorithm 3.1.

Table 1

q	$DrzPoly[B(s)]$	$DrzEf[B(s)]$	*Timing* $[DrzPoly[B(s)]]$	*Timing* $[DrzEf[B(s)]]$
1	9	3	0.06 Second	0.01 Second
2	15	7	0.27 Second	0.01 Second
3	21	8	0.751 Second	0.01 Second
4	27	8	1.072 Second	0.01 Second

5	33	8	3.284 Second	0.01 Second
6	39	8	5.809 Second	0.01 Second
7	45	8	9.524 Second	0.01 Second
8	51	8	14.851 Second	0.01 Second
9	57	8	22.052 Second	0.01 Second
10	63	8	31.475 Second	0.01 Second
11	69	8	43.732 Second	0.01 Second
12	75	8	59.286 Second	0.01 Second
13	81	8	78.563 Second	0.01 Second
14	87	8	102.397 Second	0.01 Second
15	93	8	131.209 Second	0.01 Second
16	99	8	165.457 Second	0.01 Second
17	105	8	206.117 Second	0.01 Second
18	111	8	253.504 Second	0.01 Second
19	117	8	310.156 Second	0.01 Second
20	123	8	374.188 Second	0.01 Second
25	151	8	857.342 Second	0.01 Second
30	183	8	1695.46 Second	0.01 Second
35	213	8	3044.13 Second	0.01 Second
40	283	8	-	0.02 Second
80	483	8	-	0.03 Second

7. Conclusion

We present two algorithms for symbolic computation of the Drazin inverse of a given square one-variable polynomial matrix. While the first one gives good results in the case where there exists no gap between the degrees of the polynomial matrix, the second ones is quicker and less expensive in memory in all other cases. These algorithms are a continuation of the papers [6], [7, 8, 9] and extension of the papers [3] and [5]. The implementation of introduced algorithms in the package

MATHEMATICA is described. Effectiveness of the second algorithm with respect to CPU time and the elimination of redundant computations is proved and demonstrated in examples.

References

[1] S. Barnett, Leverrier's algorithm: a new proof and extensions, SIAM J. Matrix Anal. Appl. 10 (1989), 551-556.

[2] H. P. Decell, An application of the Cayley-Hamilton theorem to generalized matrix inversion, SIAM Review 7(4) (1965), 526-528.

[3] T. N. E. Grevile, The Souriau-Frame algorithm and the Drazin pseudoinverse, Linear Algebra Appl. 6 (1973), 205-208.

[4] R. E. Hartwig, More on the Souriau-Frame algorithm and the Drazin inverse, SIAM J. Appl. Math. 31(1) (1976), 42-46.

[5] J. Ji, An alternative limit expression of Drazin inverse and its applications, Appl. Math. Comput. 61 (1994), 151-156.

[6] J. Jones, N. P. Karampetakis and A. C. Pugh, The computation and application of the generalized inverse via Maple, J. Symbolic Computation 25 (1998), 99-124.

[7] N. P. Karampetakis, Computation of the generalized inverse of a polynomial matrix and applications, Linear Algebra Appl. 252 (1997), 35-60.

[8] N. P. Karampetakis, Generalized inverses of two-variable polynomial matrices and applications, Circuits Systems Signal Processing 16 (1997), 439-453.

[9] N. P. Karampetakis and P. Tzekis, On the computation of the generalized inverse of a polynomial matrix, Ima Jour. of Mathematical Control and Information 18 (2001), 83-97.

[10] P. S. Stanimirović and N. P. Karampetakis, Symbolic implementation of Leverrier-Faddeev algorithm and applications, 8th IEEE Medit. Conference on Control and Automation, Patra, Greece, 2000.

[11] P. S. Stanimirović and M. B. Tasić, Drazin inverse of one-variable polynomial matrices, Filomat, Niš 15 (2001), 71-78.

[12] G. Wang and Y. Lin, A new extension of the Leverrier's algorithm, Linear Algebra Appl. 180 (1993), 227-238.

[13] S. Wolfram, Mathematica: A System for Doing Mathematics by Computer, Addison-Wesley Publishing Co., Redwood City, California, 1991.

[14] S. Wolfram, Mathematica Book, Version 9.0, Addison-Wesley Publishing Co., Redwood City, California, 1996.