ON THE COMPUTATION OF THE DRAZIN INVERSE OF A POLYNOMIAL MATRIX

N. P. Karampetakis * **Predrag S. Stanimirovic** **

* *Department of Mathematics, Aristotle University of Thessaloniki, Thessaloniki 54006 Greece, karampet@ccf.auth.gr*
** *Department of Mathematics, Faculty of Philosophy, University of Nis, Cirila i Metodija 2, 18000 Nis, Yugoslavia, ecko@filfak.filfak.ni.ac.yu*

Abstract: In this paper we propose two new Leverrier-Faddeev algorithms for the computation of the Drazin inverse of a polynomial matrix. These algorithms represent extensions of the ones presented in (Grevile, 1973; Ji, 1994) and a continuation of the papers (Jones et. al., 1998) and (Karampetakis et.al., 2001).

Keywords: drazin inverse, polynomial matrix, Leverrier-Faddeev algorithm.

1. INTRODUCTION

For any matrix A of the order $n \times n$ consider the following equations in X :

$$(a) \quad AX = XA \quad (b) \quad XAX = X$$
$$(c) \quad A^{k+1}X = A^k$$

If $X = A^D$ satisfies (a), (b) and (c) (for some positive integer k) then is said to be the *Drazin inverse* of A. If A is nonsingular then $A^{-1} = A^D$.

Computation of the Drazin inverse for constant matrices i.e. $A(s) \equiv A \in C^{m\times m}$ by means of the Leverrier-Faddeev algorithm (also called the Souriau-Frame algorithm) has been investigated in (Grevile, 1973). (Hartwig, 1976) continues the investigation of this algorithm. In (Ji, 1994) is presented a new proof for Grevile's finite algorithm. A symbolic implementation for the computation of the Drazin inverse of rational matrices (and therefore polynomial matrices) has recently been presented in (Stanimirovic and Karampetakis, 2000). In this paper we use the results presented in (Stanimirovic and Karampetakis, 2000), in order to present a Leverrier-Faddeev algorithm for the computation of the Drazin inverse of a polynomial matrix i.e. $A(s) = s^q A_q + \cdots + sA_1 + A_0 \in C[s]^{m\times m}$, in terms of its coefficient matrices. Thus the proposed algorithm can be easily implemented both in symbolic programming languages and high level programming languages. The paper is organized as follows. In the second section we restate well-known modification of the Leverrier-Faddeev algorithm for computation of the Drazin inverse, introduced in (Stanimirovic and Karampetakis, 2000). In the third section we apply the above Leverrier-Faddeev algorithm to the special case of polynomial matrices. As a result we investigate two new numerical algorithms which calculate the Drazin inverse of $A(s)$ in terms of its coefficient matrices A_i. An appropriate example is given in order to implement the best of these algorithms.

2. PRELIMINARIES

In this section we consider polynomial matrices i.e. $A(s) \in C[s]^{n\times n}$. In (Karampetakis et.al., 2001) is introduced the following representation of the Drazin inverse.

Theorem 1. (Stanimirovic and Karampetakis, 2000) Consider a nonregular one-variable polynomial matrix $A(s)$. Assume that

$$a(z,s) = \det[zI_n - A(s)] =$$
$$= a_0(s)z^n + a_1(s)z^{n-1} + \cdots a_{n-1}(s)z + a_n(s),$$
$$a_0(s) \equiv 1, \quad z \in \mathbb{C}$$

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

$$B_j(s) = a_0(s)A(s)^j + a_1(s)A(s)^{j-1} + \cdots + a_j(s)I_n,$$
$$a_0(s) = 1, \quad j = 0, \ldots, n$$

Let

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

Define the following set:

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

Also, assume

$$B_n(s) \equiv \mathbb{O}, \ldots, B_r(s) = 0, B_{r-1}(s) \neq 0$$

and $k = r - t$. In the case $s \in C\backslash\Lambda$ and $k > 0$, the Drazin inverse of $A(s)$ is given by

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

In the case $s \in C\backslash\Lambda$ and $k = 0$, we get $A(s)^D = O$.

For those $s_i \in \Lambda$, we follow the same procedure. □

An alternative algorithm for the computation of the Drazin inverse of $A(s)$, given in (Stanimirovic and Karampetakis, 2000) is the following:

Algorithm 1. (Stanimirovic and Karampetakis, 2000) It is assumed that $A(s) \in C(s)^{n\times n}$ is a given rational matrix.

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

$$\begin{gathered} A_0(s) = 0 \qquad a_0(s) = 1 \\ B_0(s) = I_n \\ A_1(s) = A(s)B_0(s) \qquad a_1(s) = -\frac{Tr(A_1(s))}{1} \\ B_1(s) = A_1(s) + a_1(s)I_n \\ \cdots\cdots\cdots\cdots \\ A_n(s) = A(s)B_{n-1}(s) \qquad a_n(s) = -\frac{tr(A_n(s))}{n} \\ B_n(s) = A_n(s) + a_n(s)I_n \end{gathered} \tag{1}$$

Step 2. Let

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

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

$$A(s)^D = a_t(s)^{-k-1}A(s)^k B_{t-1}(s)^{k+1} \tag{2}$$

For those $s_i \in \Lambda$, we follow the same procedure. □

3. DRAZIN INVERSE OF POLYNOMIAL MATRICES

For simplicity we assume that $A(s) \in R[s]^{n\times n}$ is a polynomial matrix of the form

$$A(s) = A_q s^q + A_{q-1}s^{q-1} + \cdots A_1 s + A_0 \in \mathbb{R}[s]^{n\times n} \tag{3}$$

where $A_i \in R^{n\times n}$, $i = 0, \ldots, q$ are constant matrices and s is an unknown variable. The coefficients $a_i(s)$, defined in (1), can be considered as polynomials of s. Also, matrix coefficients $B_i(s)$, which are defined in (1), can be considered as polynomial matrices with respect to powers of the variable s. This approach leads to an adequate representation of the Drazin inverse and an adequate algorithm for computation of the Drazin inverse.

Algorithm 2. Let the polynomial matrix $A(s)$ is of the form (3). Consider the sequence of polynomials $\{a_0(s), \ldots, a_n(s)\}$ and polynomial matrices $\{B_0(s), \ldots, B_n(s)\}$, defined as in Algorithm 1. Rewrite $a_i(s)$ and $B_i(s)$ as

$$a_i(s) = \sum_{j=0}^{iq} \hat{a}_{i,j}s^j, \quad i = 0, \ldots, n, \tag{4}$$

$$B_i(s) = \sum_{j=0}^{iq} B_{i,j}s^j, \quad i = 0, \ldots, n \tag{5}$$

where $\hat{a}_{i,j}$, $i = 0, \ldots, n$, $j = 0, \ldots, iq$ are real scalars, and $B_{i,j}$, $i = 0, \ldots, n$, $j = 0, \ldots, iq$ are constant coefficient matrices of the powers s^j. Then $\hat{a}_{i,j}$ and $B_{i,j}$, $i = 0, \ldots, n$, $j = 0, \ldots, iq$ can be generated as follows:

$$\hat{a}_{i+1,j} = -\frac{1}{i+1}\mathrm{Tr}\left(\sum_{l=0}^{j} A_{j-l}B_{i,l}\right)$$
$$i = 0, \ldots, n-1, \quad , j = 0, \ldots, (i+1)q \tag{6}$$

$$B_{i+1,j} = \sum_{l=0}^{j} A_{j-l}B_{i,l} + \hat{a}_{i+1,j}I_n \tag{7}$$
$$i = 0, \ldots, n-1, \quad , j = 0, \ldots, (i+1)q$$

where $A_k = O$, $k \geq q+1$, $B_{ik} = O$, $k \geq iq+1$.

Proof. It is not difficult to verify from (1) that the greatest powers of $A_i(s)$ (and thus of $B_i(s)$) are equal to iq, $i = 0, \ldots, n-1$. Also, the degrees of the polynomial quantities $a_i(s)$, $i = 1, \ldots, n$ are at most equal to iq. Hence $a_i(s)$ and $B_i(s)$ can be written in the form (4) and (5), respectively. Also, from (1), (3) and (5), we get the following

$$A_{i+1}(s) = A(s)B_i(s) = \sum_{j=0}^{(i+1)q}\left(\sum_{l=0}^{j} A_{j-l}B_{i,l}\right)s^j \tag{8}$$

Applying (8) to (1) we get :

$$a_{i+1}(s) = -\frac{1}{i+1}\mathrm{Tr}(A_{i+1}(s)) = \quad (9)$$
$$= \sum_{j=0}^{(i+1)q}\left[-\frac{1}{i+1}\mathrm{Tr}\left(\sum_{l=0}^{j}A_{j-l}B_{i,l}\right)\right]s^j$$

The identity (6) follows from (4) and (9). On the other hand, using (8) and (4), we get

$$B_{i+1}(s) = A_{i+1}(s) + a_{i+1}(s)I_n = \quad (10)$$
$$= \sum_{j=0}^{(i+1)q}\left(\sum_{l=0}^{j}A_{j-l}B_{i,l} + \hat{a}_{i+1,j}I_n\right)s^j$$

The identity (7) follows from (5) and (10). □

Now we are in a position to state the following algorithm for computation of the Drazin inverse $A(s)^D$, where $A(s)$ is a given polynomial matrix of the form (3).

Algorithm 3. **Initial condition:**

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

Boundary conditions

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

$i = 0$

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

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

Recursive relations for $a_i(s)$**:**

$$\hat{a}_{i+1,j} = -\frac{1}{i+1}\mathrm{Tr}\left(\sum_{l=0}^{j}A_{j-l}B_{i,l}\right)$$
$$j = 0,1,\ldots,(i+1)q$$

Recursive relations for $B_i(s)$**:**

$$B_{i+1,j} = \sum_{l=0}^{j}A_{j-l}B_{i,l} + \hat{a}_{i+1,j}I_n$$
$$j = 0,1,\ldots,(i+1)q$$

$i = i+1$

END DO

Termination criteria: Compute first t satisfying

$$\hat{a}_{t+1,j} = \hat{a}_{t+2,j} = \cdots = \hat{a}_{i,j} = 0 \quad \forall j \in \mathbb{N}.$$
while $\exists j : \hat{a}_{t,j} \neq 0$

and $r = i$ satisfying

$$r = \min\{l : B_l(s) = \mathbb{O}\}$$

Let $k = r - t$ and

$$B_j = B_{t-1,j}, \quad j = 0,1,\ldots,(k-1)q$$
$$\hat{a}_j = \hat{a}_{t,j}, \quad j = 0,\ldots,kq$$

Output: Define the following set:

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

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

$$A(s)^D = \left(\sum_{j=0}^{tq}\hat{a}_j s^j\right)^{-k-1} \times$$
$$\times \left(\sum_{i=0}^{q}A_i s^i\right)^k \left(\sum_{l=0}^{(t-1)q}B_l s^l\right)^{k+1}$$

For those $s_i \in \Lambda$, we follow the same procedure. □

It is easily checked that the number of the matrices embedded in the evaluation of the Drazin inverse is analogous to the square of the degree of $A(s)$ (see also (Karampetakis et.al., 2001)). Thus in case where for example there is only one big power of s in $A(s)$ i.e. s^{80}, then the number of matrices used for the evaluation of the Drazin inverse of $A(s)$ is analogous to $80^2 = 6400$, although the matrix may be of dimension 2×2. 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 :

$$\Phi_A = \left\{\begin{array}{c}(\mu_i) : \text{the set of degrees of} \\ \text{nonzero coefficient matrices of } A(s)\end{array}\right\}$$

$$\Phi_A(i) = \text{the } i\text{th element of } \Phi_A \;\; (\text{let } (\mu_i))$$

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

Now $A(s)$ can be rewriten as follows :

$$A(s) = \sum_{i=1}^{q}A_{\Phi_A(i)}s^{\Phi_A(i)} \in C[s]^{n\times n} \quad (11)$$

where $A_{\Phi_A(i)} \neq 0_{n,n}\ \forall i \in q$. Let also

$$B_i(s) = \sum_{j=1}^{n_i}B_{i,\Phi_i(j)}s^{\Phi_i(j)} \in C[s]^{n\times n} \quad (12)$$

where

Φ_i = the set of degrees of nonzero matrices of $B_i(s)$

$$\Phi_i(j) = \text{the } j\text{th element of } \Phi_i$$

$$n_i = \text{the total number of elements in } \Phi_i$$

Then we have that

$$A(s)B_i(s) = \sum_{j=1}^{n_A}\sum_{k=1}^{n_i} A_{\Phi_A(j)}B_{i,\Phi_i(k)}s^{\Phi_A(j)+\Phi_i(k)} \tag{13}$$

and thus

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

for $i = 1,2,...,n-1$, $j = 1,2,...,n_A$ and $k = 1,2,...,n_i$.We subtract from the set Φ_{i+1} $i = 1,2,...,n-1$ those degrees $\Phi_{i+1}(j)$ which corresponds to zero products $C_{\Phi_A(j)}B_{\Phi_i(k)}$ and we 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 (11), (12) and (13) in the recursive relations (1) we obtain the following recursive algorithm that determines $p_{i+1,\Phi_{i+1}(j)}$ and $B_{i+1,\Phi_{i+1}(j)}$ for $j = 1,2,...,n_i$.

Algorithm 4. (Computation of the Drazin inverse of $A(s)$)

Initialize :

$$B_{0,0} = I_m$$

Boundary conditions :

$$\Phi_i = \{0\},\ n_i = 1 \text{ for } i = 0,1,...,n$$
$$Q_i = 0, i = 0,1,..,n_A$$

$$\Phi_A = \{\mu_1,\mu_2,...,\mu_q\} =$$
$$= \text{the set of degrees of nonzero}$$
$$\text{coefficient matrices of } A(s)$$

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

Main Program

Step 1. $i = 0$

DO WHILE $\Phi_i \neq \emptyset$

Step 1.1. Computation of a) the coefficient matrix which correspond to the $\Phi_A(j)+\Phi_i(k)-$degree of s in $A(s)R_i(s)$ and b) the set Φ_{i+1} in terms of Φ_A and Φ_i

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

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

$$\text{for } j = 1,2,...,n_A \text{ and } k = 1,2,...,n_i$$

Step 1.2. Computation of the total number of elements in Φ_{i+1}

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

Step 1.3. Computation of $p_{i+1,\Phi_{i+1}(j)}$ and $B_{i+1,\Phi_{i+1}(j)}$

Set $s = 0$

For $j = 1,2,..,\tilde{n}_{i+1}$

If $Q_{\Phi_{i+1}(j)} = 0$ then

$$\Phi_{i+1} = \Phi_{i+1} - \{\Phi_{i+1}(j)\} \text{ and } s = s+1$$

else

$$p_{i+1,\Phi_{i+1}(j)} = -\frac{1}{i+1}tr\left[Q_{\Phi_{i+1}(j)}\right]$$

If $i < n-1$ then

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

$$\text{Set } Q_{\Phi_{i+1}(j)} = 0$$

endif

Next j

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

$i = i+1$

END DO

Terminate :

$$t = \max\{l : p_l(s) \neq 0\}$$
$$(\ t : p_{t+1,\Phi_{t+1}(j)} = p_{t+2,\Phi_{t+2}(j)} = \cdots = p_{n,\Phi_n(j)} = 0)$$

$$\text{WHILE } \exists j : p_{t,\Phi_t(j)} \neq 0$$

$$r = i \quad (\ r : \min\{l : B_l(s) = \mathbb{O}\})$$

Place

$$B_{\Phi_{t-1}(i)} = B_{t-1,\Phi_{t-1}(i)} \text{ for } i = 1,...,n_{t-1}$$

$$p_{\Phi_t(i)} = p_{t,\Phi_t(i)} \text{ for } i = 1,...,n_t$$

OUTPUT : Let $k = r-t$. The Drazin inverse of $A(s)$ will be

$$A^D(s) = -\frac{A(s)^k B_{t-1}(s)^{k+1}}{p_t(s)} = \qquad (14)$$

$$= -\frac{\left(\sum_{j=1}^{n_A} A_{\Phi_A(j)} s^{\Phi_A(j)}\right)^k \left(\sum_{j=1}^{n_{t-1}} B_{\Phi_{t-1}(j)} s^{\Phi_{t-1}(j)}\right)^{k+1}}{\left(\sum_{j=1}^{n_t} p_{\Phi_t(j)} s^{\Phi_t(j)}\right)^{k+1}}$$

for those $s \in L\,(\neq \emptyset)$: $p_k(s) \neq 0$. For those $s \in C - L$ we use the Grevile algorithm (Grevile, 1973). ■

It is easily seen that the above algorithm use in computations only the nonzero coefficient matrices of $A(s)$. This is a big advantage in cases where the degrees involved on $A(s)$ are big enough or they have big enough gaps between each other. Actually the above algorithm uses n_A matrices of the form $A_{\Phi_A(j)}$, $\max\left\{\tilde{n}_i, i = 1,2,..,r\right\}$ matrices of the form $Q_{\Phi_i(j)}$ and $2\max\left\{\tilde{n}_i, i = 1,2,..,r\right\}$ matrices of the form $B_{i,\Phi_i(j)}$

$$\text{i.e. } n_A + 2\max\left\{\tilde{n}_i, i = 1,2,..,r\right\} \text{ matrices}$$

The above algorithms needs to be changed in case where the polynomial matrix $A(s) \in C\,[s]^{r\times m}$ e.g. then the set Φ_A has to be defined as :

$$\Phi_A = \left\{ \begin{array}{c} (\mu_i, \nu_j) : \text{the set of degrees} \\ \text{of nonzero coefficient matrices of } A(s,\bar{s}) \end{array} \right\}$$

and we may use the same procedures in order to obtain the new algorithms.

Example 2. Consider the polynomial matrix :

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

Then applying algorithm 4 we get :

Initialize :

$$B_{0,0} = I_m$$

Boundary conditions :

$$\Phi_i = \{0\}\,,\ n_i = 1 \text{ for } i = 0,1,2,3$$
$$\Phi_A = \{0,1\}$$
$$\Phi_A\,(1) = 0, \Phi_A\,(2) = 1$$
$$n_A = 2 = \text{the total number of elements in } \Phi_A$$
$$Q_i = 0_{3,3},\ \ i = 0,1$$

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

Step 1.1.

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

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

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

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 $p_{1,\Phi_1(j)}$ and $B_{1,\Phi_1(j)}$: Set $s = 0$

$$Q_0 \neq 0_{3,3} \Longrightarrow p_{1,0} = -\frac{1}{1} tr\,(Q_0) = -1 \quad (j = 1)$$

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

$$Q_1 \neq 0_{3,3} \Longrightarrow p_{1,1} = -\frac{1}{1} tr\,(Q_1) = -3 \quad (j = 2)$$

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

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

We vanish the entries of Q_i:

$$Q_0 = Q_1 = 0_{3,3}$$

$i = 1$

Step 1.1.

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

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

$$Q_1 = Q_1 + A_1 B_{1,0} = 0_{3,3} \quad (j = 2, k = 1)$$
$$Q_2 = Q_2 + A_1 B_{1,1} = 0_{3,3} \quad (j = 2, k = 2)$$
$$\Phi_2 = \Phi_2 \cup \{0,1,2\} = \{0,1,2\}$$

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 $p_{2,\Phi_2(j)}$ and $B_{2,\Phi_2(j)}$: Set $s = 0$

$$Q_0 \neq 0_{3,3} \Longrightarrow p_{2,0} = -\frac{1}{2} tr\left(Q_0\right) - 2 \quad (j=1)$$

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

$$Q_1 = 0_{3,3} \Longrightarrow \Phi_2 = \Phi_2 - \{1\} = \{0,2\},$$
$$s = s + 1 = 1 \quad (j=2)$$
$$Q_2 = 0_{3,3} \Longrightarrow \Phi_2 = \Phi_2 - \{2\} = \{0\},$$
$$s = s + 1 = 2 \quad (j=3)$$
$$n_2 = \tilde{n}_2 - s = 3 - 2 = 1$$

We vanish the entries of $Q_i : Q_0 = 0_{3,3}$ $i = 2$

Step 1.1.

$$Q_0 = Q_0 + A_0 B_{2,0} = 0_{3,3} \quad (j=1, k=1)$$
$$Q_1 = Q_1 + A_1 B_{2,0} = 0_{3,3} \quad (j=2, k=1)$$
$$\Phi_3 = \Phi_3 \cup \{0,1\} = \{0,1\}$$

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

$\tilde{n}_3 = 2 =$ total number of elements in Φ_3

Step 1.3. Computation of $p_{3,\Phi_3(j)}$ and $B_{3,\Phi_3(j)}$: Set $s = 0$

$$Q_0 = 0_{3,3} \Longrightarrow \Phi_3 = \Phi_3 - \{0\} = \{1\},$$
$$s = s + 1 = 1 \quad (j=1)$$
$$Q_1 = 0_{3,3} \Longrightarrow \Phi_3 = \Phi_3 - \{1\} = \emptyset,$$
$$s = s + 1 = 2 \quad (j=2)$$
$$n_3 = \tilde{n}_3 - s = 2 - 2 = 0$$

$i = 3$

$\Phi_3 = \emptyset$ therefore END DO

Terminate :

$$t = 2 = \max\{l : p_l(s) \neq 0\}$$
$$(t : p_{t+1,\Phi_{t+1}(j)} = p_{t+2,\Phi_{t+2}(j)} = \cdots = p_{n,\Phi_n(j)} = 0)$$
$$\text{WHILE } \exists j : p_{t,\Phi_t(j)} \neq 0$$

$$r = i = 3 \quad (\ r : \min\{l : B_l(s) = \mathbb{O}\})$$

Place

$$B_0 = B_{1,0} \quad B_1 = B_{1,1}$$
$$p_0 = p_{2,0}$$

OUTPUT : Let $k = r - t = 3 - 2 = 1$. The Drazin inverse of $A(s)$ will be

$$A^D(s) = -\frac{A(s)^1 B_{2-1}(s)^{1+1}}{p_2(s)} =$$
$$= -\frac{(A_0 + A_1 s)^2 (B_0 + B_1 s)^2}{(p_0)^2} =$$
$$= \begin{bmatrix} -\frac{1}{4}s + \frac{1}{4} & \frac{1}{2}s & -\frac{1}{4}s + \frac{1}{4} \\ \frac{1}{2}s & -1 - s & \frac{1}{2}s \\ -\frac{1}{4}s + \frac{1}{4} & \frac{1}{2}s & -\frac{1}{4}s + \frac{1}{4} \end{bmatrix}$$

□

The number of additions and multiplications between matrices is 11 and 8 respectively while in case we use algorithm 3 we shall have 18 and 27 respectively. Thus we conclude that the last algorithm even there is no gap between the degrees of the polynomial matrices gives quicker and less expensive in memory results. However it uses more controls than the first one.

4. CONCLUSION

Two numerical Leverrier-Faddeev algorithms for the computation of the Drazin inverse of a polynomial matrix are investigated. While the first one gives good results in 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. The above algorithms have implications to the solution of polynomial matrix equations (Stanimirovic and Karampetakis, 2000) while can be extended to cover the class of nth-variable polynomial matrices as well as the complex polynomial matrices by using the same techniques.

5. REFERENCES

[1]

Grevile T.N.E. , The Souriau-Frame algorithm and the Drazin pseudoinverse, Linear Algebra Appl., 6, 1973, 205–208.

Hartwig R.E., More on the Souriau-Frame algorithm and the Drazin inverse, SIAM J. Appl. Math., 31, No 1, 1976, 42–46.

Ji J., An alternative limit expression of Drazin inverse and its applications, Appl. Math. Comput., 61, 1994, 151–156.

Jones J., Karampetakis N.P. and Pugh A.C., The computation and application of the generalized inverse via Maple, J. Symbolic Computation, 25, 1998, 99–124 .

Karampetakis N.P. and Tzekis P., 2001, On the computation of the generalized inverse of a polynomial matrix., *IMA Journal of Mathematical Control and Information*, 18, 83–97.

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