M a t h e m a t i c a
B a l k a n i c a

New Series Vol. 19, 2005, Fasc. 1-2

Partitioning Method for Two-Variable Rational and Polynomial Matrices

M.D. Petković[1], *P.S. Stanimirović*[2]

We introduce an extension of the Grevile's partitioning method for computing the Moore-Penrose inverse of two-variable rational and polynomial matrices. Also we developed corresponding effective algorithm, applicable in the case when the degrees in $A(s_1, s_2)$ have sufficiently large gaps between each other. These algorithms are implemented in the symbolic computational package MATHEMATICA.

AMS Subj. Classification: 15A09, 68Q40.
Key words: Pseudoinverse, MATHEMATICA, partitioning method, rational and polynomial matrices.

1. Introduction

Let $\mathcal{C}$ be the set of complex numbers and $\mathcal{C}^{m\times n}$ be the set of $m\times n$ complex matrices. As usual, $\mathcal{C}[s_1, s_2]$ (resp. $\mathcal{C}(s_1, s_2)$) denotes the polynomials (resp. rational functions) with complex coefficients of the variables s_1, s_2. The $m \times n$ matrices with elements in $\mathcal{C}[s_1, s_2]$ (resp. $\mathcal{C}(s_1, s_2)$) are denoted by $\mathcal{C}[s_1, s_2]^{m\times n}$ (resp $\mathcal{C}(s_1, s_2)^{m\times n}$).

For any matrix $A \in \mathcal{C}^{m\times n}$ the Moore-Penrose inverse of A is the unique matrix, denoted by $A^\dagger$, satisfying the following Penrose equations in X:

$$(1)\quad AXA=A,\quad (2)\quad XAX=X,\quad (3)\quad (AX)^*=AX,\quad (4)\quad (XA)^*=XA$$

Computation of the Moore-Penrose inverse of polynomial and/or rational matrices, based on the Leverrier-Faddeev algorithm, is investigated in [3, 4, 5]. In [3] 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.

In the second section we describe an extension of Grevile's partitioning method to the set of two-variable rational matrices. In the third section we

proved main theorem describe an algorithm based on the partitioning method and adjusted for two variable polynomial matrices. In fourth section we introduced effective structures for representing polynomials and basic operations on these structures. Also an algorithm for effective computing of Moore-Penrose inverse is presented. In the last section we discussed implementation of previous algorithm in symbolic package MATHEMATICA. A few illustrative examples are also presented. There proposed results can be considered as a continuation of the papers [3, 4, 5, 2] and an extension of the paper [1] to the set of rational and two variable polynomial matrices. This paper is a first attempt to compute the Moore-Penrose inverse of a two-variable polynomial and rational matrices using the Grevile's algorithm.

2. Partitioning method for rational matrices

Consider two-variable rational matrix $A(s_1, s_2)$ with dimensions $m \times n$. Let $\widehat{A}_i(s_1, s_2)$ be submatrix consisting of first i columns of matrix $A(s_1, s_2)$ and let $A_i(s_1, s_2)$ be i-th column of A. The next algorithm computes the Moore-Penrose inverse of $\widehat{A}_i(s_1, s_2)$ as a function of $A_i(s_1, s_2)$, $\widehat{A}_{i-1}(s_1, s_2)$ and $\widehat{A}^{\dagger}_{i-1}(s_1, s_2)$. It is a generalization of well-known Grevile's partitioning method to the set of two variable rational matrices with complex coefficients.

Algorithm 2.1 *Partitioning method for two-variable rational matrices*

Step 1 *Initial values:*

$$A_1^{\dagger}(s_1, s_2) = \widehat{A}_1^{\dagger}(s_1, s_2) = \begin{cases} \frac{1}{A_1^*(s_1,s_2)A_1(s_1,s_2)} A_1^*(s_1, s_2), & A_1(s_1, s_2) \neq 0, \\ A_1^*(s_1, s_2), & A_1(s_1, s_2) = 0 \end{cases}$$

Step 2 *Recursive Step: For each* $i = 2, \ldots, n$ *compute*

$$\widehat{A}_i^{\dagger}(s_1, s_2) = \begin{bmatrix} \widehat{A}_{i-1}^{\dagger}(s_1, s_2) - d_i(s_1, s_2) b_i^*(s_1, s_2) \\ b_i^*(s_1, s_2) \end{bmatrix},$$

where

Step 2.1 $d_i(s_1, s_2) = \widehat{A}_{i-1}(s_1, s_2)^{\dagger} A_i(s_1, s_2)$,

Step 2.2 $c_i(s_1, s_2) = A_i(s_1, s_2) - \widehat{A}_{i-1}(s_1, s_2) d_i(s_1, s_2)$,

Step 2.3 $b_i(s_1, s_2) = \begin{cases} \frac{1}{c_i^*(s_1,s_2)c_i(s_1,s_2)} c_i(s_1, s_2), & c_i(s_1, s_2) \neq 0 \\ \frac{1}{1+d_i^*(s_1,s_2)d_i(s_1,s_2)} (\widehat{A}_{i-1}(s_1, s_2)^{\dagger})^* d_i(s_1, s_2), & c_i(s_1, s_2) = 0 \end{cases}$

Step 3 Stopping criterion is $i = n$. *Return* $A^{\dagger}(s_1, s_2) = \widehat{A}_n^{\dagger}(s_1, s_2)$.

We used MATHEMATICA function $Simplify$ to enable simplifications of rational expressions.

3. Partitioning method for polynomial matrices

For any matrix A with $\widehat{A}_i$ denote first i columns of A and with A_i denote ith column of A. Also define $s_4 = \overline{s_1}$ and $s_3 = \overline{s_2}$.

Further, all matrices we will consider in polynomial form:

$$(1)\quad M(s_1, s_2) = M(s_1, s_2, s_3, s_4) = \sum_{j_1=0}^{q_1} \sum_{j_2=0}^{q_2} \sum_{j_3=0}^{q_3} \sum_{j_4=0}^{q_4} M_{j_1,j_2,j_3,j_4} s_1^{j_1} s_2^{j_2} s_3^{j_3} s_4^{j_4}$$

Obviously, Algorithm 2.1 is applicable on polynomial matrix $A(s_1, s_2)$. For simplifying further expressions we need the next notation:

Denote $S^J = s_1^{j_1} s_2^{j_2} s_3^{j_3} s_4^{j_4}$, $Q = (q_1, q_2, q_3, q_4) = \mathrm{dg}M$ degree of matrix polynomial $M(s_1, s_2, s_3, s_4)$ and $J = (j_1, j_2, j_3, j_4)$. Then by $\sum_{J=0}^{Q} M_J S^J$ denote the sum (1). Also with $\overline{Q}$ denote $\overline{Q} = (q_4, q_3, q_2, q_1)$. It can be easily checked that holds $M^* = \sum_{J=0}^{\overline{Q}} M_J^* S^J$ for any $M \in C^{m\times n}[s_1, s_2, s_3, s_4]$.

The main result is the following algorithm:

Algorithm 3.1 *Partitioning method for polynomial matrices*

Step 1 Denote $Q_i = \mathrm{dg}X_i$, $P_i = \mathrm{dg}y_i$ *and compute initial conditions:*

$$X_{1,J} = A_{1,J}^*, \quad Q_1 = Q, \quad y_{1,J} = \sum_{K=0}^{J} A_{i,J-\overline{K}}^* A_{i,K}, \quad P_1 = Q + \overline{Q}$$

Step 2 Recursive step: For $i = 2, \ldots, n$ *perform following steps*

Step 2.1 Compute

$$D_{i,J} = \sum_{K=0}^{J} X_{i-1,J-K} A_{i,K}, \quad \mathrm{dg}D_i = Q_{i-1} + Q$$

$$C_{i,J} = \sum_{K=0}^{J} (y_{i-1,J-K} A_{i,K} - \widehat{A}_{i-1,J-K} D_{i,K}), \quad \mathrm{dg}C_i = Q_{i-1} + 2Q$$

Step 2.2 *If there exists* $C_{i,J} \neq 0$ *compute*

$$W_{i,J} = \sum_{K=0}^{J} C_{i,J-K} y^*_{i-1,\overline{K}}, \quad v_{i,J} = \sum_{K=0}^{J} C^*_{i,J-K} C_{i-1,\overline{K}}$$

$$\mathrm{dg}W_i = Q_{i-1} + \overline{Q_{i-1}} + 2Q + \overline{Q}, \quad \mathrm{dg}v_i = \mathrm{dg}W_i + \overline{Q}$$

otherwise compute

$$W_{i,J} = \sum_{K=0}^{\overline{J}} X^*_{i-1,J-K} D_{i,\overline{K}}, \quad v_{i,J} = \sum_{K=0}^{\overline{J}} (y^*_{i-1,J-K} y_{i,\overline{K}} + D^*_{i,J-K} D_{i,\overline{K}})$$

$$\mathrm{dg}W_i = \mathrm{dg}v_i = Q_{i-1} + \overline{Q_{i-1}} + Q + \overline{Q}$$

Step 2.3 *Compute*

$$\begin{aligned} \Theta_{i,J} &= \sum_{K=0}^{J} v^*_{i,K} X_{i-1,J-\overline{K}} - D_{i,J-\overline{K}} W^*_{i,K}, \quad \mathrm{dg}\Theta_i = \overline{dgW_i} + Q + Q_{i-1} \\ \phi_{i,J} &= \sum_{K=0}^{J} v^*_{i,K} y_{i-1,J-\overline{K}}, \quad \mathrm{dg}\phi_i = \overline{dgv_i} + Q + Q_{i-1} \end{aligned}$$

Step 2.4 *Compute*

$$X_{i,J} = \begin{bmatrix} \sum\limits_{K=0}^{J} v^*_{i,K} \Theta_{i,J-\overline{K}} \\ \sum\limits_{K=0}^{J} \phi_{i,J-\overline{K}} W^*_{i,K} \end{bmatrix}, \quad Q_i = Q_{i-1} + Q + \overline{dgW_i} + \overline{dgv_i}$$

$$y_{i,J} = \sum_{K=0}^{J} v^*_{i,K} \phi_{i,J-\overline{K}}, \quad P_i = Q_i + \overline{Q}$$

Step 3 *Stopping criterion* $i = n$. *Return*

$$A(s_1, s_2)^\dagger = A_n(s_1, s_2)^\dagger = \frac{\sum\limits_{J=0}^{Q_n} X_{n,J} S^J}{\sum\limits_{J=0}^{P_n} y_{n,J} S^J}$$

Theorem 3.1 *Let $A(s_1, s_2) \in \mathcal{C}^{m\times n}$. The Moore-Penrose inverse $\widehat{A}_i^{\dagger} \in \mathcal{C}^{m\times n}[s_1, s_2, s_3, s_4]$ is given with*

$$\widehat{A}_i^{\dagger}(s_1, s_2, s_3, s_4) = \frac{X_i(s_1, s_2, s_3, s_4)}{y_i(s_1, s_2, s_3, s_4)}$$

where $X_i \in C^{m\times n}[s_1, s_2, s_3, s_4]$, $y_i \in C[s_1, s_2, s_3, s_4]$ are corresponding polynomials computed using recurrence relations in Algorithm 3.1.

Proof.

From Step 1 we can calculate:

$$\widehat{A}_1^{\dagger} = \frac{A_1(S)^*}{A_1(S)^* A_1(S)} = \frac{\sum_{K=0}^{\overline{Q}} A_{1,J}^* S^J}{\sum_{K=0}^{\overline{Q}+Q} \sum_{K=0}^{J} A_{i,J-\overline{K}}^* A_{i,K} S^J} = \frac{\sum_{K=0}^{Q_1} X_{1,J} S^J}{\sum_{K=0}^{Q_1} y_{1,J} S^J} \tag{2}$$

Suppose that theorem holds for $i-1$ and let we prove it for i. From Step 2.1 using direct calculation we have:

$$d_i(S) = \frac{X_{i-1}(S)}{y_{i-1}(S)} A_i(S) = \frac{\sum_{J=0}^{Q_{i-1}+Q} \sum_{K=0}^{J} X_{i-1,J-K} A_{i,K} S^J}{\sum_{J=0}^{P_{i-1}} y_{i-1,J} S^J} = \frac{\sum_{J=0}^{Q_{i-1}+Q} D_{i,J} S^J}{\sum_{J=0}^{P_{i-1}} y_{i-1,J} S^J} \tag{3}$$

From Step 2.2 we calculate coefficients of polynomial $c_i[S]$

$$\begin{aligned} c_i(S) &= \frac{y_{i-1}(S) A_i(S) - \widehat{A}_{i-1}(S) D_i(S)}{y_{i-1}(S)} \\ &= \frac{\sum_{J=0}^{Q_{i-1}+2Q} (\sum_{K=0}^{J} y_{i-1,J-K} A_{i,K} - \widehat{A}_{i-1,J-K} D_{i,K}) S^J}{\sum_{J=0}^{P_{i-1}} y_{i-1,J} S^J} = \frac{\sum_{J=0}^{Q_{i-1}+2Q} C_{i,J} S^J}{\sum_{J=0}^{P_{i-1}} y_{i-1,J} S^J} \end{aligned}$$

If $c_i(S) \neq 0$ then according to Step 2.3 we have

$$b_i(S) = \frac{(\sum_{J=0}^{Q_{i-1}+2Q} C_{i,J} S^J)(\sum_{J=0}^{P_{i-1}} y_{i-1,J} S^J)^*}{(\sum_{J=0}^{Q_{i-1}+2Q} C_{i,J} S^J)^* (\sum_{J=0}^{Q_{i-1}+2Q} C_{i,J} S^J)}$$

$$= \frac{\sum_{J=0}^{Q_{i-1}+\overline{Q_{i-1}}+2Q+\overline{Q}} (\sum_{K=0}^{J} C_{i,J-K} y^*_{i-1,\overline{K}}) S^J}{\sum_{J=0}^{Q_{i-1}+\overline{Q_{i-1}}+2Q+2\overline{Q}} (\sum_{K=0}^{J} C^*_{i,J-K} C_{i-1,\overline{K}}) S^J} = \frac{\sum_{J=0}^{dgW_i} W_{i,J} S^J}{\sum_{J=0}^{dgv_i} v_{i-1,J} S^J}$$

also in case $c_i(S) = 0$ we have

$$b_i(S) = \frac{\frac{X^*_{i-1}(S) D_i(S)}{y^*_{i-1}(S) y_{i-1}(S)}}{1 + \frac{D^*_i(S) D_i(S)}{y^*_{i-1}(S) y_{i-1}(S)}} = \frac{\sum_{J=0}^{Q_{i-1}+\overline{Q_{i-1}}+Q+\overline{Q}} (\sum_{K=0}^{\overline{J}} X^*_{i-1,J-K} D_{i,\overline{K}}) S^J}{\sum_{J=0}^{Q_{i-1}+\overline{Q_{i-1}}+Q+\overline{Q}} (\sum_{K=0}^{\overline{J}} (y^*_{i-1,J-K} y_{i,\overline{K}} + D^*_{i,J-K} D_{i,\overline{K}})) S^J} = \frac{\sum_{J=0}^{dgW_i} W_{i,J} S^J}{\sum_{J=0}^{dgv_i} v_{i-1,J} S^J}$$

In both cases we used temporary polynomials $W[S] \in \mathcal{C}[S]^{m \times i}$ and $v[S] \in \mathcal{C}[S]$ with degrees satisfying $dgW_i = dgv_i + \overline{Q}$. From Step 2, applying previous result, we calculate $\widehat{A}_i(S)^\dagger$:

$$\widehat{A}_i(S)^\dagger = \begin{bmatrix} \widehat{A}_{i-1}(S)^\dagger - d_i(S) b_i(S)^* \\ b_i(S)^* \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sum_{J=0}^{Q_{i-1}} X_{i-1,J} S^J}{\sum_{J=0}^{Q_{i-1}+Q} y_{i-1,J} S^J} - \frac{\sum_{J=0}^{\overline{dgW_i}+Q+Q_{i-1}} (\sum_{K=0}^{J} D_{i,J-\overline{K}} W^*_{i-1,J-K} S^J)}{\sum_{J=0}^{\overline{dgv_i}+Q+Q_{i-1}} (\sum_{K=0}^{J} v^*_{i,K} y_{i-1,J-\overline{K}} S^J)} \\ \frac{\sum_{J=0}^{\overline{dgW_i}} W^*_{i,J} S^J}{\sum_{J=0}^{\overline{dgv_i}} v^*_{i,J} S^J} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\sum_{J=0}^{dg\Theta_i} (\sum_{K=0}^{J} (v^*_{i,K} X_{i-1,J-\overline{K}} - D_{i,J-\overline{K}} W^*_{i,K}))}{\sum_{J=0}^{dg\phi_i} (\sum_{K=0}^{J} v^*_{i,K} y_{i-1,J-\overline{K}}) S^J} \\ \frac{\sum_{J=0}^{\overline{dgW_i}} W^*_{i,J} S^J}{\sum_{J=0}^{\overline{dgv_i}} v^*_{i,J} S^J} \end{bmatrix} = \begin{bmatrix} \frac{\sum_{J=0}^{dg\Theta_i} \Theta_{i,J} S^J}{\sum_{J=0}^{dg\phi_i} \phi_{i,J} S^J} \\ \frac{\sum_{J=0}^{\overline{dgW_i}} W^*_{i,J} S^J}{\sum_{J=0}^{\overline{dgv_i}} v^*_{i,J} S^J} \end{bmatrix}$$

Here we also used temporary polynomials $\Theta(S) \in \mathcal{C}(S)^{m \times i}$ and $\phi(S) \in \mathcal{C}(S)$

with coefficients defined in Step 2.3 of Algorithm 3.1. Finally,

$$\widehat{A}_i(S)^\dagger = \begin{bmatrix} \frac{\sum_{J=0}^{dg\Theta_i+\overline{dgv_i}} (\sum_{K=0}^{J} v^*_{i,K}\Theta_{i,J-\overline{K}})S^J}{\sum_{J=0}^{dg\phi_i+\overline{dgv_i}} (\sum_{K=0}^{J} v^*_{i,K}\phi_{i,J-\overline{K}})S^J} \\ \frac{\sum_{J=0}^{\overline{dgW_i}+dg\phi_i} (\sum_{K=0}^{J} \phi_{i,J-\overline{K}}W^*_{i,K})S^J}{\sum_{J=0}^{\overline{dgv_i}+dg\phi_i} (\sum_{K=0}^{J} v^*_{i,K}\phi_{i,J-\overline{K}})S^J} \end{bmatrix} = \frac{\sum_{J=0}^{Q_i} X_{i,J}S^J}{\sum_{J=0}^{Q_i+Q} y_{i,J}S^J}$$

where holds:

$$(4)\quad X_{i,J} = \begin{bmatrix} \sum_{K=0}^{J} v^*_{i,K}\Theta_{i,J-\overline{K}} \\ \sum_{K=0}^{J} \phi_{i,J-\overline{K}}W^*_{i,K} \end{bmatrix},\ dgX_i = Q_i = Q_{i-1} + Q + \overline{dgW_i} + \overline{dgv_i}$$

$$(5)\qquad y_{i,J} = \sum_{K=0}^{J} v^*_{i,K}\phi_{i,J-\overline{K}},\quad dgy_i = P_i = \overline{dgv_i} + dg\phi_i = Q_i + Q$$

$$(6)\qquad dgX_i = Q_i = Q_{i-1} + Q + \overline{dgW_i} + \overline{dgv_i},$$

which completes the proof. ∎

4. Effective computation of Moore-Penrose Inverse

In practice we work with polynomial matrices $A(s_1, s_2)$ with a small number of nonzero coefficients. In that case, previous algorithm is not effective because there are a lot of unnecessary operations. Therefore, we constructed sparse structure and effective algorithm for computing $A^\dagger(s_1, s_2)$.

Definition 4.1 *Define as Ψ_A set of degrees with non-zero coefficients in matrix polynomial A:*

$$\Psi_A = \{J \mid A_J \neq 0\}$$

Let $n_A = |\Psi_A|$. Define effective structure of A as:

$$E_A = \{(J, A_J) \mid J \in \Psi_A\}$$

Define operations $\oplus$, $\ominus$ and $\odot$ on sparse structures as: $E_A \oplus E_B = E_{A+B}$, $E_A \ominus E_B = E_{A-B}$, $E_A \odot E_B = E_{A\dot{B}}$. Also define $\overline{E_A} = E_{A^*}$

Obviously we have

$$A \cdot B = \sum_{J_A \in \Psi_A,\ J_B \in \Psi_B} A_{J_A} B_{J_B} S^{J_A + J_B}$$

and holds $\Psi_{AB} \subseteq \Psi_A + \Psi_B$. If $C = AB$ then coefficients

$$C_J = \sum_{K \in \Psi_A,\ J-K \in \Psi_B} A_K B_{J-K}$$

for $J \in \Psi_C$ (also $E_C = E_A \odot E_B$) can be computed in time $O(n_A \cdot n_B)$.

Similary holds $\Psi_{A+B} \subseteq \Psi_A \cup \Psi_B$ and sum $E_A \oplus E_B = \{(J, A_J + B_J) | J \in \Psi_{A+B}\}$ can be computed in $O(n_A + n_B)$. Also using

$$E_{A^*} = \{(\overline{J}, A_J^*) | J \in \Psi_A\}$$

$\overline{E_A}$ can be computed in $O(n_A)$ and holds $\Psi_{A^*} = \overline{\Psi_A}$.

The next algorithm is effective partitioning method for computing Moore-Penrose inverse for 2-variable polynomial matrices.

Algorithm 4.1 *Effective Partitioning Method for polynomial matrices*

Step 1 *Compute effective structures of initial values:* $E_{X_1} = \overline{E_{A_1}}$ *and* $E_{Y_1} = \overline{E_{A_1}} \odot E_{A_1}$.

Step 2 *Recursive step: For* $i = 2, \ldots, n$ *perform following steps*

Step 2.1 *Compute* $E_{D_i} = E_{X_{i-1}} \odot E_{A_i}$

Step 2.2 *Compute* $E_{C_i} = E_{y_{i-1}} \odot E_{A_i} \ominus E_{\widehat{A}_{i-1}} \odot E_{D_i}$.

Step 2.2 *If* $\Psi_{C_i} \neq \emptyset$ *compute*

$$E_{W_i} = \overline{E_{y_{i-1}}} \odot E_{C_i}, \quad E_{v_i} = \overline{E_{C_{i-1}}} \odot E_{C_i}$$

otherwise compute

$$E_{W_i} = \overline{E_{X_{i-1}}} \odot E_{D_i}, \quad E_{v_i} = \overline{E_{y_{i-1}}} \odot E_{y_{i-1}} \oplus \overline{E_{D_{i-1}}} \odot E_{D_{i-1}}.$$

Step 2.3 *Compute*

$$E_{\Theta_i} = \overline{E_{v_i}} \odot E_{X_{i-1}} \ominus E_{D_i} \odot \overline{E_{W_i}}, \quad E_{\phi_i} = \overline{E_{v_i}} \odot E_{y_{i-1}}$$

Step 2.4 Compute

$$E'_{y_i} = \overline{E'_{v_i}} \odot E'_{\phi_i}, \quad E'_{R_i} = \overline{E'_{v_i}} \odot E'_{\theta_i}, \quad E'_{S_i} = \overline{E'_{\phi_i}} \odot E'_{W_i}$$

For all $J \in \Psi_{R_i} \cup \Psi_{S_i}$ *form* $X_{i,J} = \begin{bmatrix} R_{i,J} \\ S_{i,J} \end{bmatrix}$ *where* $R_{i,J}$ *and* $S_{i,J}$ *are elements corresponding to* J *in* E_{R_i} *and* E_{S_i} *or zero if such doesn't exists.*

Step 3 Stopping criterion is $i = n$. *Return*

$$A^{\dagger}(s_1, s_2) = \frac{\sum\limits_{J \in \Psi_{X_n}} X_{n,J} S^J}{\sum\limits_{J \in \Psi_{y_n}} y_{n,J} S^J}$$

5. Examples

We implemented Algorithm 4.1 in symbolic package MATHEMATICA. Implementation is tested on several test matrices from [6] (S_n, F_n, V_n, H_n etc). Here we presented two short examples.

E x a m p l e 5.1 *Moore-Penrose inverse for two-parameter matrix* B *of the order* 4×3 *from* [6] *is equal to:*

$$B{=}2\begin{bmatrix} x+2t & -2x-t & -2x+2t \\ -x-2t & 2x+t & 2x-2t \\ -x+2t & 2x-t & 2x+2t \\ -x+2t & 2x-t & 2x+2t \end{bmatrix},\ B^{\dagger}{=}\begin{bmatrix} \frac{t+2x}{72tx} & -\frac{t+2x}{72tx} & \frac{1}{72}\left(\frac{2}{t}-\frac{1}{x}\right) & \frac{1}{72}\left(\frac{2}{t}-\frac{1}{x}\right) \\ -\frac{2t+x}{72tx} & \frac{2t+x}{72tx} & \frac{1}{72}\left(-\frac{1}{t}+\frac{2}{x}\right) & \frac{1}{72}\left(-\frac{1}{t}+\frac{2}{x}\right) \\ \frac{-t+x}{36tx} & \frac{t-x}{36tx} & \frac{t+x}{36tx} & \frac{t+x}{36tx} \end{bmatrix}$$

E x a m p l e 5.2 *Finally, we form a two-parameter complex matrix* C *of the order* 4×3 *from, similar to test matrix* B*:*

$$C = 2\begin{bmatrix} x+2It & -2x-t & -2Ix+2t \\ -x-2t & 2x+t & 2x-2t \\ -x+2It & 2x-t & 2Ix+2t \\ -x+2t & 2x-t & 2x+2t \end{bmatrix}$$

Moore-Penrose inverse is equal to

$$C^{\dagger} = \begin{bmatrix} \frac{(14+14I)t^2+(9-3I)tx+(2+8I)x^2}{-224t^3-104tx^2} & \frac{(14+14I)t^2+(7+I)tx+6x^2}{-224t^3-104tx^2} \\ \frac{(8-4I)t^3+(10+16I)t^2x+(11+5I)tx^2+(1+4I)x^3}{-224t^3x-104tx^3} & \frac{12t^3+(6-8I)t^2x+(3-7I)tx^2-3x^3}{224t^3x+104tx^3} \\ \frac{(-8+I)t^2+(18+12I)tx+(13+13I)x^2}{448t^2x+208x^3} & \frac{(12t^2-(22-20I)tx+(13+13I)x^2}{448t^2x+208x^3} \\ \frac{(14+14I)t^2-(9-3I)tx+(2+8I)x^2}{-224t^3-104tx^2} & \frac{(14+14I)t^2-(7+I)tx+6x^2}{224t^3+104tx^2} \\ \frac{(8-4I)t^3-(10+16I)t^2x+(11+5I)tx^2-(1+4I)x^3}{224t^3x+104tx^3} & \frac{12t^3-(6-8I)t^2x+(3-7I)tx^2+3x^3}{224t^3x+104tx^3} \\ \frac{(8-4I)t^2+(18+12I)tx-(13+13I)x^2}{448t^2x+208x^3} & \frac{12t^2+(22-20I)tx+(13+13I)x^2}{448t^2x+208x^3} \end{bmatrix}$$

6. Conclusion

We investigate the computation of the Moore-Penrose inverse of two-variable rectangular, rational or polynomial matrices, using the Grevile's recursive algorithm. We state an algorithm for computing the Moore-Penrose inverse of rational matrix. Then we prove main theorem which provides algorithm for computation of the Moore-Penrose inverse of polynomial matrices. Also an effective variant of previous algorithm and symbolic implementation in package MATHEMATICA are also investigated.

References

[1] T.N.E. Grevile. Some applications of the pseudo-inverse of matrix. *SIAM Rev.*, **3**, 1960, 15–22.

[2] F. Bu , Y. Wei. The algorithm for computing the Drazin inverse of two-variable polynomial matrices. *IEEE Trans. Auto. Appl. Math. Comput* (To appear).

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

[4] N.P. Karampetakis. Computation of the generalized inverse of a polynomial matrix and applications. *Linear Algebra Appl.*, **252**, 1997, 35–60.

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

[6] G. Zielke. Report on test matrices for generalized inverses. *Computing*, **36**, 1986, 105–162.

University of Niš,
Department of Mathematics,
Faculty of Science
Višegradska 33, 18000
Niš, Yugoslavia.
E-mail: [1] *dexter_of_nis@bankerinter.net,* [2] *pecko@pmf.pmf.ni.ac.yu*

Received 30.09.2003