FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 29 No 3 (2014), 209–232

# APPLICATION OF BLOCK CAYLEY-HAMILTON THEOREM TO GENERALIZED INVERSION

**Aleksandar S. Ranđelović, Predrag S. Stanimirović**

**Abstract.** In this paper we propose two algorithms for computation of the outer inverse with prescribed range and null space and the Drazin inverse of block matrix. The proposed algorithms are based on the extension of the Leverrier-Faddeev algorithm and the block Cayley-Hamilton theorem. These algorithms are implemented using symbolic and functional possibilities of the packages *Mathematica* and using numerical possibilities of *Matlab*.

**Keywords**: Generalized inverse; Leverrier-Faddeev algorithm; block Cayley-Hamilton theorem; Block matrix.

## 1. Introduction

Let $\mathbb{C}$ be the set of complex numbers and $\mathbb{C}^{m\times n}$ be the set of $m \times n$ complex matrices. The $m \times n$ matrices with elements in $\mathbb{C}$ are denoted by $\mathbb{C}^{m\times n}$. By $I_{r\times r}$ we denote the identity matrix of the order $r$, and by $\mathbb{O}_{m\times n}$ is denoted an $m \times n$ zero matrix. Also, $\otimes$ denotes the Kronecker product of matrices. Similarly, $\mathbb{C}(x)$ denotes rational functions with complex coefficients in the indeterminate $x$. The set of $m\times n$ matrices with elements in $\mathbb{C}(x)$ is denoted by $\mathbb{C}(x)^{m\times n}$. The trace of a given square matrix is denoted by $\mathrm{Tr}(A)$.

For any matrix $A \in \mathbb{C}^{m\times n}$ the Moore-Penrose inverse $A^\dagger$ of $A$ is the unique matrix $X$ satisfying the following Penrose equations [21]:

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

For any matrix $A \in \mathbb{C}^{n\times n}$ the Drazin inverse of $A$, denoted by $A^D$, is the unique matrix $X$ satisfying the following three equations [6]:

$$(1^k)\ \ A^kXA = A^k,\ \ \ (2)\ \ XAX = X,\ \ \ (5)\ \ AX = XA.$$

Received July 08, 2014.; Accepted September 05, 2014.
2010 *Mathematics Subject Classification.* Primary 15A09; Secondary 65F05

The generalized inverse $X := A^{(2)}_{T,S}$ of a matrix $A$ is the matrix satisfying $XAX = X$ and possesses the prescribed range $\mathcal{R}(X) = T$ and null space $\mathcal{N}(X) = S$. The $\{2\}$-inverses have applications in finding the solution of systems of nonlinear equations with singular Jacobians [2] as well as in Statistics [10]. In particular, outer inverses play an important role in stable approximations of ill-posed problems and in linear and nonlinear problems involving rank-deficient generalized inverses [32].

Generally speaking, it is well known that the Moore-Penrose inverse $A^\dagger$, the weighted Moore-Penrose inverse $A^\dagger_{M,N}$, the Drazin inverse $A^D$, the group inverse $A^\sharp$, as well as the Bott-Duffin inverse $A^{(-1)}_{(L)}$ and the generalized Bott-Duffin inverse $A^{(\dagger)}_{(L)}$ (see e.g. [3]) can be presented by a unified approach, as generalized inverses $A^{(2)}_{T,S}$ for appropriate choice of matrices $T$ and $S$. For example, the next formulas (see [2]) are valid for a rectangular matrix $A$:

$$A^\dagger = A^{(2)}_{\mathcal{R}(A^*),\mathcal{N}(A^*)}, \quad A^\dagger_{M,N} = A^{(2)}_{\mathcal{R}(A^\sharp),\mathcal{N}(A^\sharp)}, \tag{1.1}$$

where $M, N$ are Hermitian positive definite matrices of appropriate orders and $A^\sharp = N^{-1}A^*M$. The next identities (see [28]) are also satisfied for a given square matrix $A$:

$$A^D = A^{(2)}_{\mathcal{R}(A^k),\mathcal{N}(A^k)}, \quad A^\# = A^{(2)}_{\mathcal{R}(A),\mathcal{N}(A)}, \tag{1.2}$$

where $k = \operatorname{ind}(A)$ stands for the index of $A$.

In addition, if $A$ is the $L$-positive semi–definite matrix and $L$ is a subspace of $\mathbb{C}^n$ which satisfies $AL \oplus L^\perp = \mathbb{C}^n$, $S = \mathcal{R}(P_L A)$, then the next identities are satisfied [28, 29]:

$$A^{(-1)}_{(L)} = A^{(2)}_{L,L^\perp}, \quad A^{(\dagger)}_{(L)} = A^{(2)}_{S,S^\perp}. \tag{1.3}$$

The Leverrier-Faddeev algorithm (also called Souriau-Frame algorithm) with its numerous applications, modifications and generalizations has been often used to calculate various classes of generalized inverses. An application of the Cayley-Hamiiton theorem to matrix polynomials in several variables was presented in [22]. Extension of the classical Cayley-Hamilton theorem to continuous-time linear systems with delays was introduced in [17] and the extension to nonlinear time-varying systems with square and rectangular system matrices was proposed in [18]. Algorithms for computing the inverse of a constant nonsingular matrix $A \in \mathbb{C}^{m\times n}$ by means of the Leverrier-Faddeev algorithm were presented in [1, 7]. A more general finite algorithm for computing the Moore-Penrose generalized inverses of a given rectangular or singular constant matrix $A \in \mathbb{C}^{m\times n}$ is originated in [5]. Finite algorithm for computing the weighted Moore-Penrose inverse was developed in [27]. Later, a finite algorithm for computing the Drazin inverse was introduced in [8] by Grevile. Hartwig in [9] continued investigation of this algorithm. An alternative finite algorithm for computation of the Drazin inverse was introduced in [11]. Finite algorithms for the nontrivial $A^{(2)}_{T,S}$ inverse were established in [4].

After the application of the Leverrier-Faddeev algorithm to complex matrices, two different streams of generalizations of finite algorithms can be observed. One stream is applicable to rational and polynomial matrices and the second stream is applicable to block matrices.

We firstly restate main extensions to rational matrices. Karampetakis in [19] introduced a representation of the Moore-Penrose inverse $A(x)^{\dagger}$ of a rational matrix.

**Lemma 1.1.** [19] **(Karampetakis 1997)** *Let $A = A(x) \in \mathbb{R}(x)^{m\times n}$ be an arbitrary rational matrix and*

$$f(x) \quad = \quad a_0\lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n,\ a_0 = 1$$

*be the characteristic polynomial of $H = AA^T$. Let $k$ be a maximal index such that $a_k \neq 0$. If $k > 0$ then the Moore-Penrose inverse of $A(x)$ is given by*

$$A^{\dagger} \quad = -a_k^{-1}A^T\left[H^{k-1} + a_1H^{k-2} + \cdots + a_{k-1}I_n\right].$$

*Otherwise, if $k = 0$, $A^{\dagger} = \mathbb{O}_{n\times m}$.*

The following Lemma 1.2 gives the representation of the Drazin inverse and it is valid for both rational and polynomial square matrices [11, 20, 23]. This representation is derived as a natural extension of the corresponding representation from [8], applicable to constant square matrices.

**Lemma 1.2.** *Consider a nonregular one-variable $n\times n$ rational matrix $A = A(x)$. Assume that*

$$f(x) = a_0\lambda^n + a_1\lambda^{n-1} + \cdots a_{n-1}\lambda + a_n,\ \ a_0 \equiv 1,\ a_i \in \mathbb{R}(x)$$

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

$$B_i \quad = a_0A^i + a_1A^{i-1} + \cdots a_{i-1}A + a_iI_n,\ \ i = 0,\ldots,n.$$

*Let*

$$\begin{aligned} a_n \qquad & \equiv 0,\ldots,a_{t+1} \equiv 0, \quad a_t \neq 0, \\ B_n \quad & \equiv B_{n-1} \equiv \cdots \equiv B_r \equiv \mathbb{O}_{n\times n},\ B_{r-1} \neq \mathbb{O}_{n\times n} \end{aligned}$$

*and let $k = r - t$. Then the Drazin inverse of $A(x)$ is given by*

$$A^D = (-1)^{k+1}a_t^{-k-1}A^kB_{t-1}^{k+1}. \tag{1.4}$$

**Lemma 1.3.** [31] **(Yu, Wang 2009)** *Let $\mathcal{R}$ be an integral domain, $T$ right $\mathcal{R}$-submodule of $\mathcal{R}^n$ and $S$ right $\mathcal{R}$-submodule of $\mathcal{R}^m$. Let $A \in \mathcal{R}^{m\times n}$, $G \in \mathcal{R}^{n\times m}$ with $\mathcal{R}(G) = T$, $\mathcal{N}(G) = S$, and*

$$f(x) = a_0\lambda^m + a_1\lambda^{m-1} + \cdots + a_{m-1}\lambda + a_m,\ a_0 = 1$$

*be the characteristic polynomial of $AG$. Suppose $\rho(G) = s$ and $k$ is the largest integer such that $a_k \neq 0$. Then the generalized inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ of $A$ exists if and only if $k = s$ and $a_k$ is a unit in $\mathcal{R}$.*

*In the case $k > 0$,*

$$A^{(2)}_{T,S} = -a_k^{-1} G\left[(AG)^{k-1} + a_1(AG)^{k-2} + \cdots + a_{k-1}\right].$$

*If $k = 0$ is the largest integer such that $a_k \neq 0$, then $A^{(2)}_{T,S} = \mathbb{O}_{n\times m}$.*

Let us mention that the representation of the Moore-Penrose inverse presented in Lemma 1.1 could be derived from the particular choice $G = A^T$ in Lemma 1.3.

On the other hand, the Cayley-Hamilton theorem is extended to rectangular matrices [13, 14]. An extension of Cayley-Hamilton theorem to square block matrices is defined in [25]. Also, Kaczorek in [12, 15, 16] defined extensions of the Cayley-Hamilton theorem to block matrices.

Vitoria in [25] used the block Cayley-Hamilton theorem to compute the usual inverse of a block matrix. Wang in [26] also introduced representations of various generalized inverses (the Drazin and group inverse, the weighted Moore-Penrose inverse and the usual inverse), using the block Cayley-Hamilton theorem.

Our idea in the present paper is to unify both approaches. In the present paper we give the block Cayley-Hamilton theorem to compute the outer inverse $A^{(2)}_{T,S}$ with prescribed range and null space of a given block matrix $A = A(x)$ whose elements are from $\mathbb{C}(x)^{n\times n}$). Also, conditions for the application of the block Cayley-Hamilton theorem are not considered in [26]. These conditions are defined in this paper.

Our main result in this paper is application of the block Cayley–Hamilton theorem in finding the outer inverse with prescribed range and null space as well as in finding the Drazin inverse of block matrices. Conditions required for the application of the block Cayley-Hamilton theorem in the generalized matrix inversion are investigated. In cases when may be applied, our algorithms run much faster than the usual Leverrier-Faddeev algorithms for computing the Moore–Penrose and the Drazin inverse.

The paper is organized as follows. In the second section we investigate necessary conditions for the application of the Cayley-Hamilton theorem for block matrices in computation of various outer inverses. A necessary and sufficient condition for the existence of the generalized inverse $A(x)^{(2)}_{T,S}$ and a finite algorithm for its computation are presented in the third section. An extension of the Grevile's modification of the Leverrier-Faddeev algorithm, introduced in [8], to the set of block matrices is presented in the fourth section. In the fifth section we compare computational complexities of introduced and standard algorithms. Finally, in the sixth section we compare times of evaluation of these algorithms on some test matrices from [33]. A comparison on some numerical examples is also presented.

In the seventh section we describe implementation details of the introduced algorithms in the symbolic package *Mathematica* and in the programming language *Matlab*.

## 2. Conditions for application of the block Cayley–Hamilton theorem

As it is well known, an application of the block Cayley-Hamilton theorem to a block matrix $A$ requires that the blocks contained in $A$ are pairwisely commuting. In this section we investigate necessary conditions for application of the block Cayley-Hamilton theorem on some matrix products required in computation of the Moore–Penrose inverse, the Drazin inverse and outer inverses. Denote by $M_{m,n}(\mathbb{C}(x)^{u\times u})$ the set containing block $m\times n$ matrices with $u\times u$ blocks. In the case $m=n$ we simply denote $M_{m,n}(\mathbb{C}(x)^{u\times u})$ by $M_n(\mathbb{C}(x)^{u\times u})$. Any matrix $A\in M_{m,n}(\mathbb{C}(x)^{u\times u})$ possesses the block-partitioned form

$$A=[A_{ij}]=\begin{bmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \dots & A_{mn}\end{bmatrix}\in\mathbb{C}(x)^{m\,u\times n\,u}, \tag{2.1}$$

where each block $A_{ij}=A_{ij}(x)\in\mathbb{C}(x)^{u\times u}$ is a rational matrix of the order $u\times u$.

From the finite algorithm for computing $A^{(2)}_{T,S}$ inverses presented in [31], we conclude that a generalization of these results to the block Cayley-Hamilton theorem requires usage of blocks included in the matrix product $AG$ or $GA$. Therefore, blocks of these matrices must be pairwisely commuting.

**Lemma 2.1.** *Let $A=[A_{ij}]\in M_{m,n}(\mathbb{C}(x)^{u\times u})$ and $G=[G_{ij}]\in M_{n,m}(\mathbb{C}(x)^{u\times u})$ be two block matrices, both with pairwisely commuting blocks. Assume that the blocks $A_{ij}, G_{ij}\in\mathbb{C}(x)^{u\times u}$ are pairwisely commuting between themselves. Then both $AG$ and $GA$ is the block matrix with pairwisely commuting blocks.*

*Proof.* According to assumptions, the following equalities are satisfied between the blocks of $A$ and $G$:

$$\begin{aligned} A_{ij}A_{kl} &= A_{kl}A_{ij}, \quad i,l=1,\dots,m;\ j,k=1,\dots,n \\ G_{ij}G_{kl} &= G_{kl}G_{ij}, \quad i,k=1,\dots,n;\ j,l=1,\dots,m \\ A_{ij}G_{kl} &= G_{kl}A_{ij}, \quad i,l=1,\dots,m;\ j,k=1,\dots,n. \end{aligned} \tag{2.2}$$

Since arbitrary block $\Gamma_{ij}$ of the matrix $AG$ is defined by $\Gamma_{ij}=\sum_{p=1}^{n}A_{ip}G_{pj}$, it is clear that

$$\Gamma_{ij}\Gamma_{kl} = \sum_{p=1}^{n}\sum_{q=1}^{n}A_{ip}G_{pj}A_{kq}G_{ql}=\sum_{p=1}^{n}\sum_{q=1}^{n}A_{kq}G_{ql}A_{ip}G_{pj}=\Gamma_{kl}\Gamma_{ij}.$$

It means that the matrix $AG$ contains pairwisely commuting blocks. The dual statement, related with the block matrix $GA$ can be verified in a similar way. □

Decell in [5] applied the Cayley-Hamilton theorem on the matrix $AA^*$ and derive corresponding representation of the Moore-Penrose inverse of arbitrary complex matrix $A$,. In order to apply the block Cayley–Hamilton theorem to compute the Moore-Penrose inverse of a block matrix, the blocks in $AA^*$ must be pairwisely commutative. First we give a supporting statement which ensures this condition.

**Corollary 2.1.** *Let $A = [A_{ij}] \in M_{m,n}(\mathbb{C}(x)^{u\times u})$ be given block matrix with pairwisely commuting blocks. Assume that the blocks $A_{ij} \in \mathbb{C}(x)^{u\times u}$, $i = 1,\dots,m, j = 1,\dots,n$ and $A^*_{kl} \in \mathbb{C}(x)^{u\times u}$, $k = 1,\dots,n, j = 1,\dots,m$ are pairwisely commuting between themselves. Then both $AA^*$ and $A^*A$ is the block matrix with pairwisely commuting blocks.*

*Proof.* According to assumptions, the following equalities are satisfied between the blocks of $A$ and $A^*$:

$$\begin{aligned} A_{ij}A_{kl} &= A_{kl}A_{ij}, \quad i,k = 1,\dots,m;\ j,l = 1,\dots,n \\ A_{ij}A^*_{kl} &= A^*_{kl}A_{ij}, \quad i,l = 1,\dots,m;\ j,k = 1,\dots,n. \end{aligned} \tag{2.3}$$

Since the blocks of $A$ are pairwisely commuting, it is not difficult to verify that the block of the matrix $A^*$ are also pairwisely commuting. Then the proof follows from Lemma 2.1. □

In Lemma 2.3 we find a class of matrices which satisfy conditions required in Corrolary 2.1. More precisely, we show that the normal and pairwisely commuting blocks in $A$ enable an application of the block Cayley–Hamilton theorem on the matrix $AA^*$. For this purpose we exploit the results of auxiliary Lemma 2.2.

**Lemma 2.2.** *Let $\mathbb{W}$ be the set of normal and pairwisely commuting matrices. Then for $A, B \in \mathbb{W}$ we have that $AB^* = B^*A$.*

*Proof.* We know that a matrix is normal if and only if it is unitarily similar to a diagonal matrix, i.e. if it is diagonalisable by a unitary matrix. In other words, there exists a unitary matrix $U$ such that $U^*AU$ is diagonal for every $A$ from the set $\mathbb{W}$. Let $A$ and $B$ are two arbitrary matrices from the set $\mathbb{W}$. Then we have that $A = UD_1U^*$ and $B^* = UD_2^*U^*$, where $D_1$ and $D_2$ are diagonal matrices. Now, using

$$AB^* = UD_1U^*UD_2^*U^* = UD_1D_2^*U^* = UD_2^*D_1U^* = UD_2^*U^*UD_1U^* = B^*A$$

we complete the proof. □

**Lemma 2.3.** *Let $A$ has the form (2.1), where the blocks $A_{ij} \in \mathbb{C}(x)^{u\times u}$ are normal and pairwisely commuting. Then $AA^*$ is the block matrix with pairwisely commuting blocks.*

*Proof.* Let $B = AA^*$. We have that arbitrary blocks $B_{ij}$ and $B_{kl}$ from $B$ are equal to

$$B_{ij} = \sum_{p=1}^{n} A_{ip}A^*_{pj}, \qquad B_{kl} = \sum_{q=1}^{n} A_{kq}A^*_{ql}.$$

According to Lemma 2.2 we have

$$B_{ij}B_{kl} \quad = \sum_{p=1}^{n}\sum_{q=1}^{n} A_{ip}A_{pj}^{*}A_{kq}A_{ql}^{*} = \sum_{p=1}^{n}\sum_{q=1}^{n} A_{kq}A_{ql}^{*}A_{ip}A_{pj}^{*} = B_{kl}B_{ij},$$

so the matrix $B = AA^{*}$ possesses commutative blocks. $\square$

**Remark 2.1.** In Corollary 2.1 we suppose the set of conditions (2.3) which is much stronger condition compared to the conditions arising from the assumptions of Lemma 2.3.

## 3. Outer inverses of block matrix

In this section, we present a finite algorithm for computing the generalized inverse $A_{T,S}^{(2)}$ of a block matrix $A$. Also, we deduce a necessary and sufficient condition for its existence arising from the finite algorithm.

**Definition 3.1.** Let $A$ has the form (2.1). The matrix $A_{i*} = \begin{bmatrix} A_{i1} & \cdots & A_{in} \end{bmatrix} \in M_{1,n}(\mathbb{C}(x)^{u\times u})$, where $A_{i1},\dots,A_{in}$ are blocks of $A$, is called the $i$th block row of the block matrix $A$. Similarly the matrix $A_{*j} = \begin{bmatrix} A_{1j} \\ \vdots \\ A_{mj} \end{bmatrix} \in M_{m,1}(\mathbb{C}(x)^{u\times u})$, where $A_{1j},\dots,A_{mj}$ are blocks of $A$, is called the $j$th block column of the block matrix $A$.

**Definition 3.2.** Block rank of the block matrix $A$ in the form (2.1), denoted by $\mathrm{Brank}(A)$, is a maximal number of linearly independent block rows (block columns) in $A$, where all rows (columns) in linearly independent block rows (block columns) must be linearly independent in $A$.

We said that the block matrix $A$ satisfies the block rank condition if independent block rows (block columns) consist of linearly independent rows (columns) of $A$. If $A$ satisfies the block rank condition we may note that $\rho(A) = u\cdot \mathrm{Brank}(A)$ and so $\rho(A) \in \{u\cdot i \mid i = 0,\dots,\min\{m,n\}\}$.

**Theorem 3.1.** *Let $A = [A_{ij}] \in M_{m,n}(\mathbb{C}(x)^{u\times u})$ and $G = [G_{ij}] \in M_{n,m}(\mathbb{C}(x)^{u\times u})$ be block matrices satisfying (2.2) and the block rank condition. Further, let*

$$f(S) = \det[I_{m\times m}\otimes S - AG] = S^{m} + Q_1S^{m-1} + \cdots + Q_{m-1}S + Q_m$$

*be the characteristic polynomial of $AG$, where $S \in \mathbb{C}(x)^{u\times u}$ is the matrix (block) eigenvalue of $AG$ and $Q_i \in \mathbb{C}^{u\times u}$, $i = 1,\dots,m$. Suppose $\rho(G) = s$ and $K$ is the largest integer such that $Q_K \neq \mathbb{O}_{u\times u}$, $1 \le K \le \min\{m,n\}$. Then the generalized inverse $A_{\mathcal{R}(G),\mathcal{N}(G)}^{(2)}$ of $A$ exists if and only if $u\cdot K = u\cdot \mathrm{Brank}(G) = s$.*

*In the case $K > 0$, corresponding outer inverse of $A$ is defined by*

$$\begin{aligned} A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} &= -G\left[(AG)^{K-1} + (I_{m\times m}\otimes Q_1)(AG)^{K-2} + \cdots + (I_{m\times m}\otimes Q_{K-1})\right] \\ &\qquad \cdot (I_{m\times m}\otimes Q_K)^{-1} \\ &= -\left[(GA)^{K-1} + (I_{n\times n}\otimes Q_1)(GA)^{K-2} + \cdots + (I_{n\times n}\otimes Q_{K-1})\right] \\ &\qquad \cdot (I_{n\times n}\otimes Q_K)^{-1}G. \end{aligned}$$

*If $K = 0$ is the largest integer such that $Q_K \neq \mathbb{O}_{u\times u}$, then $A^{(2)}_{T,S} = \mathbb{O}_{n\,u\times m\,u}$.*

*Proof.* Since $\rho(AG) \le \rho(G) = u\cdot\text{Brank}(G) = s$, it follows that $Q_j = \mathbb{O}_{u\times u}$ for $u\cdot j > s$ by the argument above, and then $u\cdot K \le s$. Following known result from [30, 31], we conclude that $A^{(2)}_{T,S}$ exists if and only if $u\cdot K = u\cdot\text{Brank}(G) = s$. Since $Q_K$ is invertible, immediately follows that $I_{m\times m}\otimes Q_K$ is invertible $m\,u \times m\,u$ block diagonal matrix, whose diagonal elements are matrices $Q_K$.

According to the block Cayley-Hamilton theorem (see [12, 15]) $AG$ must satisfy the condition assumed in Lemma 2.1. In this case, by applying the block Cayley-Hamilton theorem, we get

$$(AG)^u + (I_{m\times m}\otimes Q_1)(AG)^{m-1} + \cdots + (I_{m\times m}\otimes Q_{m-1})(AG) + (I_{m\times m}\otimes Q_m) = \mathbb{O}_{m\,u\times m\,u}.$$

When $K \neq 0$ we may write

$$\begin{gathered} (AG)^{m-K}\cdot \\ \left[(AG)^K + (I_{m\times m}\otimes Q_1)(AG)^{K-1} + \cdots + (I_{m\times m}\otimes Q_{K-1})(AG) + (I_{m\times m}\otimes Q_K)\right] \\ = \mathbb{O}_{m\,u\times m\,u}. \end{gathered} \tag{3.1}$$

By [30], $(AG)_g$ exists and $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = G(AG)_g$. Pre-multiplying (3.1) by $(AG)_g^{m-K+1}$ yields

$$\begin{gathered} (AG)_g\cdot \\ \left[(AG)^K + (I_{m\times m}\otimes Q_1)(AG)^{K-1} + \cdots + (I_{m\times m}\otimes Q_{K-1})(AG) + (I_{m\times m}\otimes Q_K)\right] \\ = \mathbb{O}_{m\,u\times m\,u}, \end{gathered}$$

and so

$$\begin{gathered} (AG)_g = \\ -(AG)_g(AG)\left[(AG)^{K-1} + (I_{m\times m}\otimes Q_1)(AG)^{K-2} + \cdots + (I_{m\times m}\otimes Q_{K-1})\right] \\ \cdot (I_{m\times m}\otimes Q_K)^{-1}. \end{gathered} \tag{3.2}$$

It follows from [31] that

$$G(AG)_g(AG) = G. \tag{3.3}$$

By (3.1) and (3.2) we have that

$$\begin{aligned}
&A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = \\
&-G(AG)_g(AG)\left[(AG)^{K-1} + (I_{m\times m}\otimes Q_1)(AG)^{K-2} + \cdots + (I_{m\times m}\otimes Q_{K-1})\right] \\
&\cdot(I_{m\times m}\otimes Q_K)^{-1} \\
&= -G\left[(AG)^{K-1} + (I_{m\times m}\otimes Q_1)(AG)^{K-2} + \cdots + (I_{m\times m}\otimes Q_{K-1})\right] \\
&\cdot(I_{m\times m}\otimes Q_K)^{-1} \\
&= -\left[(GA)^{K-1} + (I_{n\times n}\otimes Q_1)(GA)^{K-2} + \cdots + (I_{n\times n}\otimes Q_{K-1})\right] \\
&\cdot(I_{n\times n}\otimes Q_K)^{-1}G.
\end{aligned}$$

If $K = 0$, then $AG = \mathbb{O}_{u\,n\times u\,n}$. Obviously, $(AG)_g = \mathbb{O}_{u\,n\times u\,n}$ and, therefore, $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = \mathbb{O}_{n\,u\times m\,u}$. □

An algorithm for computing $A^{(2)}_{T,S}$ inverse based on the Cayley-Hamilton theorem, is proposed in [31]. We are giving its generalization on both block and rational matrices.

**Remark 3.1.** The weighted Moore-Penrose inverse $A^{\dagger}_{MN}$, the Moore-Penrose inverse $A^{\dagger}$, the Drazin inverse $A^D$, the group inverse $A^{\#}$ and the inverse $A^{-1}$ are the special cases of the generalized inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$, as it is defined in (1.1), (1.2) and (1.3):

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = \begin{cases} A^{\dagger}, & G = A^* \\ A^{\dagger}_{MN}, & G = N^{-1}A^*M \\ A^D, & G = A^k,\ k \geq \operatorname{ind}(A) \\ A^{\#}, & G = A \\ A^{-1}, & G = I \end{cases}$$

In accordance with Theorem 3.1 we give the following Algorithm 3.1 for computing outer inverses of a given block matrix. Let us mention that the matrix product $AG$ is calculated only once, at the initial step.

**Algorithm 3.1.** **Require:** $A = [A_{ij}] \in M_{m,n}(\mathbb{C}(x)^{u\times u})$ and $G = [A_{ij}] \in M_{n,m}(\mathbb{C}(x)^{u\times u})$.

```
1: Construct the sequence of $u \times u$ matrices $\{Q_0, Q_1, \dots, Q_u\}$ and the sequence of $m\,u \times m\,u$
   matrices $\{B_0, B_1, \dots, B_u\}$ by the following rules:

$$(3.4)\qquad \begin{array}{lll} A_0 = \mathbb{O}_{m\,u\times m\,u}, & Q_0 = -I_{u\times u}, & B_0 = I_{m\,u\times m\,u} \\ A_1 = AG, & Q_1 = \frac{\sum_{j=1}^{m}(A_1)_{jj}}{1}, & B_1 = A_1 - I_{m\times m}\otimes Q_1 \\ \dots & \dots & \dots \\ A_m = AGB_{m-1}, & Q_m = -\frac{\sum_{j=1}^{m}(A_m)_{jj}}{m}, & B_m = A_m + I_{m\times m}\otimes Q_m. \end{array}$$

2: Let $K = \max\{l : Q_l \neq \mathbb{O}_{u\times u}\}$.
3: if $K = 0$ then
4:     return $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} := \mathbb{O}_{n\,u\times m\,u}$
```

```
5: else
6:     return $A^{(2)}_{T,S} = GB_{K-1}(I_{m\times m} \otimes Q_K)^{-1}$.
7: end if
```

Dual algorithm can be formulated analogously.

## 4. Drazin inverse of block matrix

Wang and Lin in [26] obtained a representation of the Drazin inverse using the block Cayley–Hamilton theorem of matrix $A^{k+1}$. Also, the Drazin inverse can be obtained from the particular case $G = A^l$, $l \geq \mathrm{ind}(A)$ of Theorem 3.1. But, computation of matrix powers and, therefore, computation of the index is not appropriate for large scale matrices. Here we give an alternative representation obtained by the block Cayley–Hamilton theorem of the matrix $A$. Note that blocks must be pairisely commuting in order to meet condition of the block Cayley-Hamilton theorem, but not normal matrices. Proof of this theorem is an extension of the result from [8].

**Theorem 4.1.** *Let $A \in M_n(\mathbb{C}(x)^{u\times u})$ has the form (2.1) where $A_{ij} \in \mathbb{C}^{u\times u}$ are pairwisely commuting, and let*

$$f(x) = det[I_{n\times n} \otimes S - A] = S^n + Q_1 S^{n-1} + \cdots + Q_{n-1}S + Q_n$$

*be the characteristic polynomial of $A$ where $S \in \mathbb{C}(x)^{u\times u}$ is the matrix (block) eigenvalue of $A$. Also, consider the following sequence of $n\,u \times n\,u$ constant matrices defined by coefficients $Q_i$ and powers of $A$:*

$$B_j = A^j + (I_{n\times n} \otimes Q_1)A^{j-1} + \cdots + (I_{n\times n} \otimes Q_{j-1})A + I_{n\times n} \otimes Q_j, \quad j = 0, \dots, n. \tag{4.1}$$

*Let $r$ denotes the smallest integer such that $B_r = \mathbb{O}_{n\,u\times n\,u}$, let $t$ denotes the largest integer satisfying $Q_t \neq \mathbb{O}_{u\times u}$, and let $k = r - t$. Then the Drazin inverse of $A$ is defined as*

$$A^D = (-1)^{k+1}(I_{n\times n} \otimes Q_t)^{-k-1}A^k B^{k+1}_{t-1}. \tag{4.2}$$

*Proof.* According from (4.1) we have

$$B_j = AB_{j-1} + I_{n\times n} \otimes Q_j. \tag{4.3}$$

Since $t$ denote the largest integer such that $Q_t \neq \mathbb{O}_{u\times u}$ and $\rho(Q_t) = u$, we have from (4.3) that

$$B_{t+h} = A^h B_t \qquad (h = 0, 1, \dots). \tag{4.4}$$

Let $k = r - t$. Then, from (4.3) and (4.4) it follows

$$\begin{aligned} A^k(AB_{t-1} + I_{n\times n} \otimes Q_t) &= A^k B_t = B_{t+k} = B_{t+r-t} = B_r = \mathbb{O}_{n\,u\times n\,u} \\ &\Longrightarrow A^{k+1}B_{t-1} = -(I_{n\times n} \otimes Q_t)A^k. \end{aligned}$$

It is obvious from (4.1) that $B_{t-1}$ is polynomial in $A$. Now, let define $q(A)$ in the following way

$$q(A) = (I_{n\times n} \otimes Q_t)^{-1} B_{t-1}. \tag{4.5}$$

Let us show that

$$X = (-1)^{k+1} A^k [q(A)]^{k+1} \tag{4.6}$$

satisfies $(1^k), (2), (5)$.

Since $A$ commutes with $B_j$, $(1^k)$ is satisfied. Also, we have that

$$A^{k+1} q(A) = A^{k+1} (I_{n\times n} \otimes Q_t)^{-1} B_{t-1} = -A^k. \tag{4.7}$$

By successive substitution of (4.7), it is easily verified that

$$\begin{aligned} AX^2 &= A^{2k+1}[q(A)]^{2k+2} = A^k[A^{k+1}q(A)][q(A)]^{2k+1} \\ &= -A^{2k}[q(A)]^{2k+1} = \cdots = (-1)^{k+1}A^k[q(A)]^{k+1} = X, \\ XA^{k+1} &= (-1)^{k+1}A^{2k+1}[q(A)]^{k+1} = (-1)^{k+1}A^k[A^{k+1}q(A)]q(A)^k \\ &= (-1)^k A^{2k}[q(A)]^k = \cdots = A^k. \end{aligned}$$

Therefore (4.6) gives the Drazin pseudoinverse and substitution of (4.5) in (4.6) gives (4.2). □

Also, in [8] is proposed algorithm for computation of the Drazin inverse of $A$, which avoids explicit matrix powering. Here we give an extension of this algorithm to the set of block matrices.

**Algorithm 4.1.** **Require:** $A(x) = [A_{ij}] \in M_n(\mathbb{C}(x)^{u\times u})$.

```
1: Construct the sequence of $n \times n$ matrices $\{Q_0, Q_1, \dots, Q_n\}$ and the sequence of $nu \times nu$
   matrices $\{B_0, B_1, \dots, B_n\}$ in the following way:
```

$$\begin{array}{lll} A_0 = \mathbb{O}_{nu\times nu}, & Q_0 = I_{u\times u}, & B_0 = I_{nu\times nu} \\ A_1 = AB_0, & Q_1 = -\frac{\sum_{j=1}^n (A_1)_{jj}}{1}, & B_1 = A_1 + I_{n\times n} \otimes Q_1 \\ \dots & \dots & \dots \\ A_n = AB_{n-1}, & Q_n = -\frac{\sum_{j=1}^n (A_n)_{jj}}{n}, & B_n = A_n + I_{n\times n} \otimes Q_n. \end{array} \tag{4.8}$$

```
2: Let $t = \max\{l : Q_l \neq \mathbb{O}_{u\times u}\}$,   $r = \min\{l : B_l = \mathbb{O}_{nu\times nu}\}$,   $k = r - t$.
3: **if** $k = 0$ **then**
4:     **return** $A^D := \mathbb{O}_{nu\times nu}$
5: **else**
6:     **return** $A^D$ defined by (4.2).
7: **end if**
```

**Remark 4.1.** Computation of the Drazin inverse requires determination of $\mathrm{ind}(A)$, which is a time consuming job and numerically instable process because of the usage of matrix powers $A^k$. To avoid this difficulty, we present a simplified method to obtain integer $l$ satisfying $l \geq \mathrm{ind}(A)$. More precisely, we compute successive matrix powers $A^{2^k}$ by the squaring method: $A^2 = A \cdot A, A^4 = A^2 \cdot A^2$ etc. The squaring is stopped when the condition $\mathrm{rank}(A^{2^k}) = \mathrm{rank}(A^{2^{k+1}})$ is satisfied and $l$ is chosen by $l = 2^k$.

## 5. Complexity of algorithms

In this section we compare computational complexities of Algorithm 3.1 and the usual Leverier-Faddeev algorithm as well as complexities of Algorithm 4.1 and Grevile's algorithm from [8]. Let us denote by $A(n)$ the complexity of the addition/subtraction of $n \times n$ matrices. These operations can be computed in time $A(n) = \mathcal{O}(n^2)$. By $T_r(n)$ we denote the complexity of the algorithm for computing the trace of $n \times n$ matrix. It is clear that its computational complexity is $T_r(n) = \mathcal{O}(n)$. Also, let us denote by $M(m,n,k)$ the complexity of multiplying $m \times n$ matrix by $n \times k$ matrix. Without using any rapid method for matrix multiplication its computational complexity is $M(m,n,k) = \mathcal{O}(mnk)$. The simpler notation $M(n)$ is used instead of $M(n,n,n)$. Also by $K_p(m,n,p,q)$ we denote the complexity of the Kronecker product of matrices of the order $m \times n$ and $p \times q$. It can be easily verified that $K_p(m,n,p,q) = \mathcal{O}(mnpq)$.

We denote $E_{BLF}$ and $E_{LF}$ computational complexities of of Algorithm. 3.1 and usual Leverier-Faddeev algorithm. Computational complexity of each step in Algorithm. 3.1 has order $M(m\,u) + A(u) + K_p(m,m,u,u) + A(m\,u)$. Since algorithm has $m$ steps, for its effectiveness it can be terminated after $K$ steps, when we find $K = \max\{l : Q_l \neq \mathbb{O}_{u\times u}\}$.

Complexity of Algorithm 3.1 is of the order

$$\begin{aligned} E_{BLF} &= K\Big(M(m\,u) + A(u) + K_p(m,m,u,u) + A(m\,u)\Big) \\ &= K\Big(\mathcal{O}(m^3\,u^3) + \mathcal{O}(u^2) + \mathcal{O}(m^2\,u^2) + \mathcal{O}(m^2\,u^2)\Big) \\ &\approx \mathcal{O}(K\,m^3\,u^3). \end{aligned}$$

On the other hand, computational complexity of the usual Leverier-Faddeev algorithm is of the order

$$\begin{aligned} E_{LF} &= k\,(M(m\,u, n\,u, m\,u) + T_r(m\,u) + A(m\,u)) \\ &= u\,K\Big(\mathcal{O}(m^2\,n\,u^3) + \mathcal{O}(m\,u) + \mathcal{O}(m^2\,u^2)\Big) \\ &\approx \mathcal{O}(K\,m^2\,n\,u^4). \end{aligned}$$

Let us denote $E_{BGreville}$ and $E_{Greville}$ computational complexities of of Algorithm 4.1 and Grevile's algorithm from [8]. Scanning these Algorithms in a similar way, it is not difficult to verify that their computational complexities are same as of Algorithm 3.1 and the usual Leverier-Faddeev algorithm. The only difference is that we use the square matrix as input and in that case $m = n$. We derive the following computational complexities:

$$\begin{aligned} E_{BLF} &\approx \mathcal{O}(K\,m^3\,u^3) \quad , E_{LF} \approx \mathcal{O}(K\,m^2\,n\,u^4) \\ E_{BGreville} &\approx \mathcal{O}(K\,n^3\,u^3) \quad , E_{Greville} \approx \mathcal{O}(K\,n^3\,u^4). \end{aligned}$$

We conclude that Algorithm 3.1 has approximately $u$ times smaller computational complexity than the usual Leverier-Faddeev algorithm, where $u$ is the dimension of block in block matrix. Also Algorithm 4.1 has $u$ times smaller computational complexity than the usual Greville algorithm.

**Remark 5.1.** Therefore, block Leverrier-Faddeev algorithm is $u$ times faster than the usual Leverrier-Faddeev algorithm. Therefore, it is desirable to provide that $u$ be as greater as possible. Therefore, the best improvement can be expected in the case $m = n = 2$, i.e. when $A$ is $2 \times 2$ block matrix.

## 6. Examples

In this section we compare CPU times for evaluation of the Moore-Penrose and the Drazin inverse on some test matrices by means of the usual Leverier-Faddeev algorithm (LF shortly) and Grevile's [8] algorithm and introduced algorithms (Alg. 3.1) and (Alg. 4.1). Values are derived using built-in function *Timing*[] in *Mathematica* and *tic* and *toc* command in *Matlab*. Experiments are done on an Intel(R) Core(TM)2DUO CPU T6600 @ 2.20 GHz 64/32-bit system with 4 GB RAM memory.

### 6.1. Hadamard matrices

Hadamard matrices $H_n$ of order $n = 2^k, k = 1, 2, \ldots$ are defined as

$$H_n = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{6.1}$$

CPU times of the evaluation are given in the following Table 6.1. Notation $H_n^\dagger$ (Alg. 3.1) denotes application of Algorithm 3.1 in the case $G = A^T$. Similarly, $H_n^\dagger$ (LF) denotes the Moore-Penrose inverse obtained by Lemma 1.1. Values for the column $H_n^D$ (Alg. 3.1) are derived applying Algorithm 3.1 in the case $G = A^k$, $k \geq \text{ind}(A)$ and $H_n^D$ (LF) denotes the Drazin inverse obtained by Lemma 1.3 in the case $G = A^k$, $k \geq \text{ind}(A)$, applying Leverier-Faddeev algorithm for its computation. Values for the column $H_n^D$ (Alg. 4.1) are obtained by Algorithm 4.1 and $H_n^D$ (Grevile [8]) denotes the Drazin inverse obtained by Lemma 1.2.

Table 6.1: Comparing CPU times on Hadamard matrices

| $(2\times 2\times n)$ | $H_n^\dagger$ (Alg. 3.1) | $H_n^\dagger$ (LF) | $H_n^D$ (Alg. 3.1) | $H_n^D$ (LF) |
|---|---|---|---|---|
| $2\times 2\times 2$ | 0 | 0 | 0 | 0 |
| $2\times 2\times 4$ | 0 | 0 | 0 | 0 |
| $2\times 2\times 8$ | 0 | 0.015 | 0 | 0.013 |
| $2\times 2\times 16$ | 0.032 | 0.124 | 0.028 | 0.117 |
| $2\times 2\times 32$ | 0.125 | 1.279 | 0.109 | 1.128 |
| $2\times 2\times 64$ | 0.593 | 16.255 | 0.577 | 15.957 |
| $2\times 2\times 128$ | 4.431 | 235.249 | 4.327 | 228.471 |
| $2\times 2\times 256$ | 36.021 | 3308.591 | 33.628 | 3257.478 |
| $2\times 2\times 512$ | 428.129 | – | 421.387 | – |
| $2\times 2\times 1024$ | 3576.011 | – | 3432.416 | – |

Table 6.2: Comparing CPU times on Hadamard matrices

| $(2\times 2\times n)$ | $H_n^D$ (Alg. 4.1) | $H_n^D$ (Grevile [8]) |
|---|---|---|
| $2\times 2\times 2$ | 0 | 0 |
| $2\times 2\times 4$ | 0 | 0 |
| $2\times 2\times 8$ | 0 | 0.016 |
| $2\times 2\times 16$ | 0.015 | 0.078 |
| $2\times 2\times 32$ | 0.063 | 1.575 |
| $2\times 2\times 64$ | 0.671 | 24.523 |
| $2\times 2\times 128$ | 4.103 | 416.961 |
| $2\times 2\times 256$ | 31.863 | – |
| $2\times 2\times 512$ | 397.129 | – |
| $2\times 2\times 1024$ | 3237.771 | – |

### 6.2. Test block matrix

Zielke [33] has generated block test matrices in the following way:

$$V_{2n} = \begin{pmatrix} V_n & V_n \\ V_n & -V_n \end{pmatrix},\; n = 2^k,\; k = 0, 1, 2, \ldots \tag{6.2}$$

with

$$V_2 = \begin{pmatrix} a & b \\ b & -a \end{pmatrix}.$$

In the following table we compare CPU times for evaluation of the Moore-Penrose and the Drazin inverse by means of the usual Leverier-Faddeev and Grevile algorithm [8], respectively, as well as the introduced algorithms for block matrices. Since test matrices are given in the symbolic form, the implementation written in *Mathematica* is used to generate these results.

In the following Table 6.2 we use the same notation as in Table 6.1.

Table 6.3: CPU times on the test block matrix $V_n$

| $(2\times 2\times n)$ | $H_n^\dagger$ (Alg. 3.1) | $H_n^\dagger$ (LF) | $H_n^D$ (Alg. 3.1) | $H_n^D$ (LF) |
|---|---|---|---|---|
| $2\times 2\times 2$ | 0.015 | 0.015 | 0.013 | 0.013 |
| $2\times 2\times 4$ | 0.031 | 0.047 | 0.030 | 0.042 |
| $2\times 2\times 8$ | 0.094 | 0.187 | 0.092 | 0.169 |
| $2\times 2\times 16$ | 0.328 | 1.171 | 0.323 | 1.165 |
| $2\times 2\times 32$ | 1.981 | 7.722 | 1.898 | 7.697 |
| $2\times 2\times 64$ | 18.486 | 85.348 | 18.378 | 83.135 |
| $2\times 2\times 128$ | 150.261 | 973.368 | 137.162 | 966.152 |

Table 6.4: CPU times on the test block matrix $V_n$

| $(2\times 2\times n)$ | $H_n^D$ (Alg. 3.1) | $H_n^D$ (LF) | $H_n^D$ (Alg. 4.1) | $H_n^D$ (Grevile [8]) |
|---|---|---|---|---|
| $2\times 2\times 2$ | 0.013 | 0.013 | 0 | 0.016 |
| $2\times 2\times 4$ | 0.030 | 0.042 | 0.016 | 0.031 |
| $2\times 2\times 8$ | 0.092 | 0.169 | 0.062 | 0.421 |
| $2\times 2\times 16$ | 0.323 | 1.165 | 0.312 | 4.634 |
| $2\times 2\times 32$ | 1.898 | 7.697 | 1.685 | 50.326 |
| $2\times 2\times 64$ | 18.378 | 83.135 | 11.466 | 765.356 |
| $2\times 2\times 128$ | 137.162 | 966.152 | 107.868 | – |

## 6.3. Randomly generated test matrices

In this section we construct random numerical matrices that satisfy initial conditions of Theorems 3.1, where $G = A^*$ and $G = A^k$ and compare times of evaluation using algorithms implemented in programming language *Matlab* First we compare times of evaluation of Moore-Penrose inverse between Leverier-Faddeev algorithm and our modification of that algorithm (Alg. 3.1) and then times of evaluation of Drazin inverse between these algorithms. Test matrix $A$ must have blocks that normal and commutative, therefore blocks must be simultaneously diagonalisable by a unitary matrix. We construct blocks of $A$ in the following way:

$$A_{ij} = UDU^{-1},\ \ i = 1,\dots,m,\ j = 1,\dots,n,$$

where the matrix $U \in \mathbb{C}^{u\times u}$ is random unitary matrix which is the same for all blocks, and the matrix $D \in \mathbb{C}^{u\times u}$ is a random diagonal matrix.

Notation $H_n^\dagger$ (Alg. 3.1) denotes application of Algorithm 3.1 in the case $G = A^*$ and $H_n^\dagger$ (LF) denotes the Moore-Penrose inverse obtained by Lemma 1.3 in the case $G = A^*$ applying Leverier-Faddeev algorithm for its computation. Other notations are the same as in previous subsections.

CPU times obtained by various algorithms are arranged in the following tables.

Table 6.5: CPU times on arbitrary matrices

| $(m \times n \times u)$ | $A^\dagger$ (Alg. 3.1) | $A^\dagger$ (LF) |
|---|---|---|
| $2 \times 3 \times 2$ | 0.000 | 0.000 |
| $4 \times 7 \times 3$ | 0.001 | 0.003 |
| $5 \times 9 \times 4$ | 0.003 | 0.007 |
| $8 \times 11 \times 5$ | 0.011 | 0.081 |
| $13 \times 17 \times 6$ | 0.154 | 1.616 |
| $16 \times 21 \times 7$ | 0.271 | 39.586 |
| $23 \times 29 \times 8$ | 2.009 | 614.253 |
| $28 \times 36 \times 9$ | 4.126 | 2728.189 |
| $32 \times 45 \times 10$ | 9.563 | 8498.012 |
| $41 \times 52 \times 11$ | 31.543 | – |
| $45 \times 55 \times 12$ | 60.435 | – |
| $50 \times 60 \times 13$ | 109.747 | – |
| $56 \times 67 \times 14$ | 212.441 | – |
| $60 \times 70 \times 15$ | 335.853 | – |

Table 6.6: CPU times on arbitrary matrices

| $(n \times n \times u)$ | $A^D$ (Alg. 3.1) | $A^D$ (LF) |
|---|---|---|
| $2 \times 2 \times 2$ | 0.000 | 0.000 |
| $4 \times 4 \times 3$ | 0.000 | 0.000 |
| $5 \times 5 \times 4$ | 0.000 | 0.000 |
| $8 \times 8 \times 5$ | 0.015 | 0.077 |
| $13 \times 13 \times 6$ | 0.122 | 1.423 |
| $16 \times 16 \times 7$ | 0.405 | 39.129 |
| $23 \times 23 \times 8$ | 1.775 | 607.315 |
| $28 \times 28 \times 9$ | 4.160 | 2698.371 |
| $32 \times 32 \times 10$ | 11.446 | 8371.418 |
| $41 \times 41 \times 11$ | 32.649 | – |
| $45 \times 45 \times 12$ | 64.491 | – |
| $50 \times 50 \times 13$ | 136.638 | – |
| $56 \times 56 \times 14$ | 228.354 | – |
| $60 \times 60 \times 15$ | 326.692 | – |

Test matrix $A$ must have blocks that are pairwisely commuting during the evaluation of the Drazin inverse by means of the Grevile algorithm and our modification of that algorithm (Alg. 4.1). Therefore, we construct blocks of $A$ in the following way:

$$A_{ij} = SDS^{-1}, i = 1, \dots, n, \ j = 1, \dots, n.$$

where the matrix $S \in \mathbb{C}^{u \times u}$ is random invertible matrix which is same for all blocks and the matrix $D \in \mathbb{C}^{u \times u}$ is randomly generated diagonal matrix.

Comparative CPU times are arranged in the following table.

Table 6.7: CPU times on arbitrary matrices

| $(n \times n \times u)$ | $A^D$ (Alg. 4.1) | $A^D$ (Grevile [8]) |
|---|---|---|
| $2 \times 2 \times 2$ | 0.000 | 0.000 |
| $4 \times 4 \times 3$ | 0.000 | 0.000 |
| $5 \times 5 \times 4$ | 0.000 | 0.000 |
| $8 \times 8 \times 5$ | 0.006 | 0.031 |
| $13 \times 13 \times 6$ | 0.037 | 0.624 |
| $16 \times 16 \times 7$ | 0.124 | 2.386 |
| $23 \times 23 \times 8$ | 0.591 | 16.863 |
| $28 \times 28 \times 9$ | 2.000 | 247.184 |
| $32 \times 32 \times 10$ | 4.323 | 998.555 |
| $41 \times 41 \times 11$ | 11.336 | 2969.567 |
| $45 \times 45 \times 12$ | 21.778 | 7548.900 |
| $50 \times 50 \times 13$ | 40.656 | – |
| $56 \times 56 \times 14$ | 77.311 | – |
| $60 \times 60 \times 15$ | 116.914 | – |

## 7. Conclusion

We present an algorithm for computing $A^{(2)}_{T,S}$ inverse as well as an algorithm for computing the Drazin inverse of a given block matrix. These algorithms are based on the Leverrier-Faddeev algorithm and the block Cayley-Hamilton theorem. We prove theorems which provides these algorithms. We also present and compare computational complexities of these and standard algorithms. Algorithms are implemented in both programming languages *Mathematica* and *Matlab*, and tested on several classes of test examples. Numerical examples which compare our algorithms with corresponding algorithms on element-wise given matrices are presented.

## 8. Implementation

In this section we present programs in *Mathematica* that compute the Moore-Penrose inverse of $m \times n$ block matrix and the Drazin inverse of $n \times n$ block matrix, where all blocks are of dimensions $u \times u$. We also give programs that construct test matrix for computing these pseudoinverses. We use implementation in *Mathematica* for symbolic computations of outer inverses.

```
MoorePenroseBlockLF[A_, m_, n_, u_] :=
  Module[{A1, B, Z, Ap, Q, Bp, i, j, K, Inv},
   A1 = ConjugateTranspose[A]; B = Simplify[A.A1];
   Z = 0*IdentityMatrix[m*u];
   Ap[0] = Z; Q[0] = -IdentityMatrix[u]; Bp[0] = IdentityMatrix[m*u];
   For[i = 1, i <= m, i++,
    Ap[i] = Simplify[B.Bp[i - 1]];
    Q[i] =
     Simplify[
      Sum[Ap[i][[(j - 1)*u + 1 ;; j*u  ,   (j - 1)*u + 1 ;;
          j*u    ]], {j, 1, m}]/i];
    Bp[i] =
     Simplify[Ap[i] - KroneckerProduct[IdentityMatrix[m], Q[i]]];
    ];
   K = m; While[ToString[Det[Q[K]]] == ToString[0], K--];
   Inv =
    Simplify[
     A1.Bp[K - 1].Inverse[KroneckerProduct[IdentityMatrix[m], Q[K]]]];
   Print[MatrixForm[Inv]];
   If[((Simplify[A.Inv.A] == A) && (Simplify[Inv.A.Inv] ==
        Inv) && (Simplify[ConjugateTranspose[A.Inv]] ==
        Simplify[A.Inv]) && (Simplify[ConjugateTranspose[Inv.A]] ==
        Simplify[Inv.A])),
    Print["Inverse satisfies Moore-Penrose's equations"]];
   ];


DrazinBlockLF[A_, m_, n_, u_] :=
  Module[{A1, B, Z, Ap, Q, Bp, i, j, K, Inv},
   r = 1;
   While[
    MatrixRank[MatrixPower[A, r]] != MatrixRank[MatrixPower[A, 2*r]],
    r := 2*r];
   A1 = MatrixPower[A, r]; B = Simplify[A.A1];
   Z = 0*IdentityMatrix[m*u];
   Ap[0] = Z; Q[0] = -IdentityMatrix[u]; Bp[0] = IdentityMatrix[m*u];
   For[i = 1, i <= m, i++,
    Ap[i] = Simplify[B.Bp[i - 1]];
    Q[i] =
     Simplify[
      Sum[Ap[i][[(j - 1)*u + 1 ;; j*u  ,   (j - 1)*u + 1 ;;
          j*u    ]], {j, 1, m}]/i];
    Bp[i] =
     Simplify[Ap[i] - KroneckerProduct[IdentityMatrix[m], Q[i]]];
    ];
   K = m; While[ToString[Det[Q[K]]] == ToString[0], K--];
   Inv =
    Simplify[
     A1.Bp[K - 1].Inverse[KroneckerProduct[IdentityMatrix[m], Q[K]]]];
   Print[MatrixForm[Inv]];
   If[((Simplify[Inv.A.Inv] == Inv) && (Simplify[Inv.A] ==
        Simplify[A.Inv]) && (Simplify[MatrixPower[A, K].Inv.A] ==
        Simplify[MatrixPower[A, K]]) ),
    Print["Inverse satisfies Drazin's equations"]];
   ];


DrazinBlockGrevile[A_, n_, u_] :=
  Module[{Z, AD, Q, B, i, j, t, r, k, Inv},
   Z = 0*IdentityMatrix[n*u];
   AD[0] = Z; Q[0] = IdentityMatrix[u]; B[0] = IdentityMatrix[n*u];
   For[i = 1, i <= n, i++,
    AD[i] = Simplify[A.B[i - 1]];
    Q[i] =
     Simplify[-Sum[
         AD[i][[(j - 1)*u + 1 ;; j*u  , (j - 1)*u + 1 ;; j*u ]], {j,
          1, n}]/i];
    B[i] = Simplify[AD[i] + KroneckerProduct[IdentityMatrix[n], Q[i]]];
    ];
   t = n; While[ToString[Det[Q[t]]] == ToString[0], t--];
   r = 0; While[ToString[B[r]] != ToString[Z], r++];
   k = r - t;
   Inv =
    Simplify[(-1)^(k - 1)*
      MatrixPower[
        KroneckerProduct[IdentityMatrix[n],
         Q[t]], (-k - 1)].MatrixPower[A, k].MatrixPower[
        B[t - 1], (k + 1)]];
   Print[MatrixForm[Inv]];
   If[((Simplify[Inv.A.Inv] == Inv) && (Simplify[Inv.A] ==
        Simplify[A.Inv]) && (Simplify[MatrixPower[A, k].Inv.A] ==
        Simplify[MatrixPower[A, k]]) ),
    Print["Inverse satisfies Drazin's equations"]];
   ];


TestMatLF[m_, n_, u_] :=
  Module[{U, i, j, k, D},
   U = Orthogonalize[
     Table[RandomInteger[{-10, 10}] +
       I*RandomInteger[{-10, 10}], {u}, {u}]];
   A = Table[1, {i, m}, {j, n}];
   For[i = 1, i <= m, i++,
    For[j = 1, j <= n, j++,
      D = IdentityMatrix[u];
      For[k = 1, k <= u, k++, D[[k, k]] = RandomInteger[{-10, 10}]];
      A[[i, j]] = U.D.Inverse[U];
      ];
    ];
   A = ArrayFlatten[A];
   ];


TestMatGrevile[n_, u_] :=
  Module[{S, i, j, k, D},
   S = Table[RandomInteger[{-10, 10}], {u}, {u}];
   A = IdentityMatrix[n];
   For[i = 1, i <= n, i++,
    For[j = 1, j <= n, j++,
      D = IdentityMatrix[u];
      For[k = 1, k <= u, k++, D[[k, k]] = RandomInteger[{-10, 10}]];
      A[[i, j]] = S.D.Inverse[S];
      ];
    ];
   A = ArrayFlatten[A];
   ];
```

Corresponding implementation in *Matlab* is also presented in the following code. We use this code for numerical computations.

```
function [time] = MoorePenroseBlockLF(A,m,n,u)
A1=A'; B=A*A1; Z = zeros(m*u);
Ap=Z; Q=-eye(u); Bp=eye(m*u);
for i=2:(m+1)
    Ap(:,:,i)=B*Bp(:,:,i-1);
    Q(:,:,i)=zeros(u);
    for j=1:m
        Q(:,:,i)=Q(:,:,i)+Ap((j-1)*u+1:j*u,(j-1)*u+1:j*u,i);
    end
    Q(:,:,i)=Q(:,:,i)/(i-1);
    Bp(:,:,i)=Ap(:,:,i)-kron(eye(m),Q(:,:,i));
end
k=m+1;
while (det(Q(:,:,k))==0) & (k>1)
    k=k-1;
end
fprintf('MoorePenroseBlockLF(k) = %f \n',k-1);
Inv = A1*Bp(:,:,k-1)*inv(kron(eye(m),Q(:,:,k)));


function [time] = DrazinBlockLF(A,n,u)
r=1;
while (rank(A^r)~=rank(A^(2*r))) do   r=2*r;
end
A1=A^r; B=A*A1; Z = zeros(n*u);
Ap=Z; Q=-eye(u); Bp=eye(n*u);
for i=2:(n+1)
    Ap(:,:,i)=B*Bp(:,:,i-1);
    Q(:,:,i)=zeros(u);
    for j=1:n
        Q(:,:,i)=Q(:,:,i)+Ap((j-1)*u+1:j*u,(j-1)*u+1:j*u,i);
    end
    Q(:,:,i)=Q(:,:,i)/(i-1);
    Bp(:,:,i)=Ap(:,:,i)-kron(eye(n),Q(:,:,i));
end
k=n+1;
while (det(Q(:,:,k))==0) & (k>1)     k=k-1;
end
Inv = A1*Bp(:,:,k-1)*inv(kron(eye(n),Q(:,:,k)));


function [time] = DrazinBlockGrevile(A,n,u)
Z = zeros(n*u);   AD=Z; Q=eye(u); B=eye(n*u);
for i=2:(n+1)
    AD(:,:,i)=A*B(:,:,i-1);
    Q(:,:,i)=zeros(u);
    for j=1:n
        Q(:,:,i)=Q(:,:,i)+AD((j-1)*u+1:j*u,(j-1)*u+1:j*u,i);
    end
    Q(:,:,i)=-Q(:,:,i)/(i-1);
    B(:,:,i)=AD(:,:,i)+kron(eye(n),Q(:,:,i));
end
t=n+1;
while (det(Q(:,:,t))==0) & (t>1)        t=t-1;
end
r=1;
while (any(any(B(:,:,r)~=Z))) & (r<=n)  r=r+1;
end
```

```
k=r-t;
Inv = (-1)^(k+1)*(kron(eye(n),Q(:,:,t)))^(-k-1)*(A^k)*(B(:,:,t-1)^(k+1));

function [A] = TestMatLF(m,n,u)
U=orth(randint(u,u,[-10,10])+i*randint(u,u,[-10,10]));
A=[];
for x=1:m
    B=[];
    for y=1:n
      D= eye(u);
      for k=1:u
          D(k,k)=randint(1,1,[-10,10]);
      end
      B=horzcat(B,U*D*inv(U));
    end
    A=vertcat(A,B);
end

function [A] = TestMatGrevile(n,u)
S=randint(u,u,[-10,10]);
A=[];
for i=1:n
    B=[];
    for j=1:n
      D= eye(u);
      for k=1:u
          D(k,k)=randint(1,1,[-10,10]);
      end
      B=horzcat(B,S*D*inv(S));
    end
    A=vertcat(A,B);
end
```

## REFERENCES

1. S. Barnett: *Leverrier's algorithm: a new proof and extensions*, SIAM J. Matrix Anal. Appl. **10** (1989), 551–556.
2. A. Ben-Israel and T.N.E. Greville: *Generalized inverses: theory and applications*, Springer, New York, NY, USA, 2nd edition, 2003.
3. Y. Chen: *The generalized Bott–Duffin inverse and its application*, Linear Algebra Appl. **134** (1990), 71–91.
4. Y. Chen: *Finite Algorithms for the* (2)*-Generalized Inverse* $A^{(2)}_{T,S}$, Linear and Multilinear Algebra **40** (1995), 61–68.
5. H.P. Decell: *An application of the Cayley-Hamilton theorem to generalized matrix inversion*, SIAM Review **7** No 4 (1965), 526–528.
6. M.P. Drazin: *Pseudo-inverse in associative rings and semigroups*, Amer. Math. Monthly **65** (1958), 506–514.
7. D.K. Faddeev and V.N. Faddeeva; *Computational Methods of Linear Algebra*, Freeman, San Francisko, 1963.
8. T.N.E. Grevile: *The Souriau-Frame algorithm and the Drazin pseudoinverse*, Linear Algebra Appl. **6** (1973), 205–208.

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

10. A.J. Getson and F.C. Hsuan: *{2}-inverses and their Statistical applications*, Lecture Notes in Statistics **47**, Springer, New York, NY, USA, 1988.

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

12. T. Kaczorek: *New extensions of the Cayley–Hamilton theorem with applicattions*, Proceeding of the 19th European Conference on Modelling and Simulation, 2005.

13. T. Kaczorek: *An Existence of the Cayley-Hamilton Theorem for Singular 2-D Linear Systems with Non-Square Matrices*, Bulletin of the Polish Academy of Sciences. Technical Sciences **43**(1) (1995), 39–48.

14. T. Kaczorek: *Generalization of the Cayley-Hamilton Theorem for Non-Square Matrices*, International Conference of Fundamentals of Electronics and Circuit Theory XVIII-SPETO, Gliwice, 1995, pp. 77–83.

15. T. Kaczorek: *An Existence of the Caley-Hamilton Theorem for Non-Square Block Matrices*, Bulletin of the Polish Academy of Sciences. Technical Sciences **43**(1) (1995), 49–56.

16. T. Kaczorek: *An Extension of the Cayley-Hamilton Theorem for a Standard Pair of Block Matrices*, Applied Mathematics and Computation Sciences **8**(3) (1998), 511–516.

17. T. Kaczorek: *Extension of the Cayley-Hamilton theorem to continuoustime linear systems with delays*, Int. J. Appl. Math. Comput. Sci. **15**(2) (2005), 231–234.

18. T. Kaczorek: *An extension of the CayleyHamilton theorem for nonlinear timevarying systems*, Int. J. Appl. Math. Comput. Sci. **16**(1) (2006), 141–145.

19. N.P. Karampetakis: *Computation of the generalized inverse of a polynomial matrix and applications*, Linear Algebra Appl. **252** (1997), 35–60.

20. N.P. Karampetakis, P.S. Stanimirović and M.B. Tasić: *On the computation of the Drazin inverse of a polynomial matrix*, Far East J. Math. Sci. (FJMS) **26(1)** (2007), 1–24.

21. R. Penrose: *A generalized inverse for matrices*, Proc. Cambridge Philos. Soc. **52** (1956), 17–19.

22. A. Paz: *An application of the Cayley-Hamilton theorem to matrix polynomials in several variables*, Linear and Multilinear Algebra **15** (1984), 161–170.

23. P.S. Stanimirović and M.B. Tasić: *Drazin inverse of one-variable polynomial matrices*, Filomat, Niš **15** (2001), 71–78.

24. P.S. Stanimirović: *A finite algorithm for generalized inverses of polynomial and rational matrices*, Appl. Math. Comput. **144** (2003) 199–214.

25. J. Vitoria: *A block–Cayley–Hamilton theorem*, Bulletin Mathematique **26(71)** (1982), 93–97.

26. G. Wang and L. Qiu: *Some New Applications of the Block–Cayley–Hamilton Theorem*, J. of Shangai Teachers Univ. (Natural Sciences) **27** (1998), 8–15, In Chinesse.

27. G. Wang: *A finite algorithm for computing the weighted Moore-Penrose inverse $A^{\dagger}_{M,N}$*, Appl. Math. Comput. **23** (1987), 277–289.

28. G. Wang, Y. Wei and S. Qiao: *Generalized Inverses: Theory and Computations*, Science Press, Beijing/New York, 2004.

29. Y. Wei and H. Wu: *The representation and approximation for the generalized inverse $A^{(2)}_{T,S}$*, Appl. Math. Comput. **135** (2003), 263–276.

30. Y. Yu and G. Wang: *On the generalized inverse* $A^{(2)}_{T,S}$ *over integral domains*, Aust. J. Math. Appl. **4** (2007), 1. Article 16, 1–20.

31. Y. Yu and G. Wang: *DFT calculation for the* {2}*-inverse of a polynomial matrix with prescribed image and kernel*, Applied Math. Comput. **215** (2009), 2741–2749.

32. B. Zheng and R. B. Bapat: *Generalized inverse* $A^{(2)}_{T,S}$ *and a rank equation*, Appl. Math. Comput. **155** (2004), 407-415.

33. G. Zielke: *Report on Test Matrices for Generalized Inverses*, Computing **36** (1986), 105–162.

Predrag S. Stanimirović
Faculty of Science and Mathematics
Department of Mathematics and Informatics
Višegradska 33
18000 Niš, Serbia
pecko@pmf.ni.ac.rs

Aleksandar S. Ranđelović
Faculty of Science and Mathematics
Department of Mathematics and Informatics
Višegradska 33
18000 Niš, Serbia
randjelovicaca@gmail.com