Gauss–Jordan elimination method for computing outer inverses

Predrag S. Stanimirović[1], Marko D. Petković[2],

[1,2] *University of Niš, Faculty of Sciences and Mathematics, Višegradska 33, 18000 Niš, Serbia*[*]

E-mail: [1] pecko@pmf.ni.ac.rs, [2] dexterofnis@gmail.com

Abstract

This paper deals with the algorithm for computing outer inverse with prescribed range and null space, based on the choice of an appropriate matrix G and Gauss–Jordan elimination of the augmented matrix $[G \mid I]$. The advantage of such algorithms is the fact that one can compute various generalized inverses using the same procedure, for different input matrices. In particular, we derive representations of the Moore–Penrose inverse, the Drazin inverse as well as $\{2,4\}$ and $\{2,3\}$–inverses. Numerical examples on different test matrices are presented, as well as the comparison with well–known methods for generalized inverses computation.

AMS Subj. Class.: 15A09, 15A23.

Key words: Gauss–Jordan elimination; Generalized inverse; outer inverse; QR factorization.

1 Introduction

Using the usual notation, by $\mathbb{C}_r^{m\times n}$ we denote the set of all complex $m \times n$ matrices of rank r, and by I we denote the unit matrix of an appropriate order. Furthermore A^*, $\mathcal{R}(A)$, rank(A) and $\mathcal{N}(A)$ denote the conjugate transpose, the range, the rank and the null space of $A \in \mathbb{C}^{m\times n}$.

If $A \in \mathbb{C}_r^{m\times n}$, T is a subspace of $\mathbb{C}^n$ of dimension $t \leq r$ and S is a subspace of $\mathbb{C}^m$ of dimension $m-t$, then A has a $\{2\}$-inverse X such that $\mathcal{R}(X) = T$ and $\mathcal{N}(X) = S$ if and only if $AT \oplus S = \mathbb{C}^m$. In the case when the existence is ensured, X is unique and it is denoted by $A^{(2)}_{T,S}$. Outer generalized inverses with prescribed range and null-space are very important in matrix theory. They are used in constrcuing iterative methods for solving nonlinear equations [1, 8] as well as in statistics [4, 5]. Furthermore, outer inverses play an important role in stable approximations of ill-posed problems and in linear and nonlinear problems involving rank-deficient generalized inverses [7, 19]. Observing from the theoretical point of view, it is well known that the Moore-Penrose inverse and the weighted Moore-Penrose inverse $A^\dagger, A^\dagger_{M,N}$, the Drazin and the group inverse $A^D, A^\#$, as well as the Bott-Duffin inverse $A^{(-1)}_{(L)}$ and the generalized Bott-Duffin inverse $A^{(\dagger)}_{(L)}$ 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 statements are valid for a rectangular matrix A (see [1, 9, 16]):

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

where M, N are positive definite matrices of appropriate orders and $A^\sharp = N^{-1}A^*M$. For a given square matrix A the next identities are satisfied (see [1, 2, 3, 16]):

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

[*]The authors gratefully acknowledge support from the Research Project 174013 of the Serbian Ministry of Science

where $k = \mathrm{ind}(A)$. 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 (see [2, 16, 17]):

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

We study Gauss–Jordan elimination methods for computing various outer inverses of complex matrices. The oldest and best known among these methods is the method for calculating the inverse matrix. The Gauss–Jordan elimination method for computing the inverse of a nonsingular matrix A is based on the executing elementary row operations on the pair $[A \mid I]$ and its transformation into the block matrix $\left[I \mid A^{-1}\right]$ involving the inverse A^{-1}. A number of numerical methods are developed for computing various classes of outer inverses with prescribed range and null space. The Gauss–Jordan elimination method to compute the Moore–Penrose inverse is developed in [12]. The metod from [12] is based on two successive sets of elementary row operations. The first computes reduced row echelon form of A^*A:

$$E\left[A^*A \mid I\right] = \begin{bmatrix} E_1A^*A & E_1 \\ O & E_2 \end{bmatrix} \tag{1.4}$$

while the second provides the following transformation

$$\begin{bmatrix} E_1A^*A & E_1 \\ E_2 & O \end{bmatrix} \to \left[\begin{array}{c|c} I & \begin{bmatrix} E_1A^*A \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1 \\ O \end{bmatrix} \end{array}\right].$$

After that, Moore-Penrose inverse can be computed by

$$A^{\dagger} = \begin{bmatrix} E_1A^*A \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1 \\ O \end{bmatrix} A^*.$$

More general algorithm for computing $A_{T,S}^{(2)}$ inverses is introduced in [13]. This algorithm is very useful generalization of the method from [12]. The essence of this generalization consists in the replacement of the matrix A^* by an appropriate matrix G.

Several improvements of the algorithm from [12] are recently presented in [6]. First improvement from [6] assumes the initial transformation of the form

$$E\left[A^* \mid I\right] = \begin{bmatrix} E_1A^* & E_1 \\ O & E_2 \end{bmatrix}.$$

The second improvement exploits special structure of the matrix which is subject in Gauss Jordan transformation.

Two main goals of the present paper should be emphasized.

Firstly, motiveted by the modification introduced in [6], in the present paper we introduce corresponding modification of the algorithm introduced in [13]. This possibility is mentioned in the conclusion of the paper [6]. That type algorithms are able to compute various generalized inverses of matrix A, for different choice of an input matrix G.

Moreover, we observed that the algorithms introduced in [6, 11, 12, 13] are not accompained by adequate implementation and not tested on adequate test examples. The numerical properties of these algorithms are not studied in details so far. Our second goal is the implementation of described algorithms and the numerical experience derived applying the implementation.

The paper is organized as follows. Necessary preliminary results are surveyed in the next section. Our main algorithm is defined in the third section after necessary theoretical investigations. In Section 4 we presented an illustrative numerical example and explain our motivation for the corresponding improvements of the algorithm. These improvements save the computational time and increase numerical stability of the main algorithm. Exploiting our implementation in the programming language `C++`, in the last section we tested considered algorithms on randomly generated test matrices. Also, a series of numerical experiments corresponding to the Moore-Penrose inverse and the Drazin inverse are presented.

2 Preliminary results

There exist a number of full–rank representations for outer inverses of prescribed rank as well as for outer inverses with prescribed range and kernel. The following representations from [11, 18] will be useful for our results that follow.

Proposition 2.1. *Let $A\in\mathbb{C}_r^{m\times n}$, T be a subspace of $\mathbb{C}^n$ of dimension $s\leq r$ and let S be a subspace of $\mathbb{C}^m$ of dimension $m-s$. In addition, suppose that $G\in\mathbb{C}^{n\times m}$ satisfies $\mathcal{R}(G)=T$, $\mathcal{N}(G)=S$. Let G has an arbitrary full–rank decomposition, that is $G=UV$. If A has a $\{2\}$-inverse $A^{(2)}_{T,S}$, then:*

(1) [11] *GAF is an invertible matrix and*

$$A^{(2)}_{T,S}=U(VAU)^{-1}G=A^{(2)}_{\mathcal{R}(U),\mathcal{N}(V)}. \tag{2.1}$$

(2) [18] $\mathrm{ind}(AG)=\mathrm{ind}(GA)=1$ *and*

$$A^{(2)}_{T,S}=G(AG)^{\#}=(GA)^{\#}G. \tag{2.2}$$

According to known representations from [1, 10, 11, 14, 15] we state the next additional representations with respect to (1.1)–(1.3). These representations characterize the classes of $\{2\}$, $\{2,4\}$ and $\{2,3\}$ generalized inverses of known rank.

Proposition 2.2. *Let $A\in\mathbb{C}_r^{m\times n}$ be an arbitrary matrix and let $0<s\leq r$ be a positive integer. The following general representations for some classes of generalized inverses are valid:*

$$\begin{aligned}
&\text{(a)}\quad A\{2\}_s=\{A^{(2)}_{\mathcal{R}(U),\mathcal{N}(V)}=U(VAU)^{-1}G\mid U\in\mathbb{C}^{n\times s},\ V\in\mathbb{C}^{s\times m},\ \mathrm{rank}(VAU)=s\};\\
&\text{(b)}\quad A\{2,4\}_s=\left\{A^{(2,4)}_{\mathcal{R}((VA)^*),\mathcal{N}(V)}=(VA)^*\left(VA(VA)^*\right)^{-1}G\mid V\in\mathbb{C}_s^{s\times m}\right\}\\
&\qquad\qquad=\left\{(VA)^{\dagger}V\mid VA\in\mathbb{C}_s^{s\times n}\right\};\\
&\text{(c)}\quad A\{2,3\}_s=\left\{A^{(2,3)}_{\mathcal{R}(U),\mathcal{N}((AU)^*)}=U\left((AU)^*AU\right)^{-1}(AU)^*\mid U\in\mathbb{C}_s^{n\times s}\right\}\\
&\qquad\qquad=\left\{U(AU)^{\dagger}\mid AU\in\mathbb{C}_s^{m\times s}\right\};\\
&\text{(d)}\quad A\{1,2\}=A\{2\}_r.
\end{aligned}$$

Proposition 2.3. [18] *Let $A\in\mathbb{C}^{m\times n}$ be of rank r, let T be a subspace of $\mathbb{C}^n$ of dimension $s\leq r$, and let S be a subspace of $\mathbb{C}^m$ of dimension $m-s$. In addition, suppose that $G\in\mathbb{C}^{n\times m}$ satisfies $\mathcal{R}(G)=T$ and $\mathcal{N}(G)=S$. If A has $A^{(2)}_{T,S}$ then* $\mathrm{ind}(AG)=\mathrm{ind}(GA)=1$ *and*

$$A^{(2)}_{T,S}=G(AG)^{\#}=(GA)^{\#}G.$$

Sheng and Chen in [13] derived the following representation of the $A^{(2)}_{T,S}$ inverse corresponding to a particular choice of the matrix G

$$A^{(2)}_{T,S}=\begin{bmatrix}E_1GA\\E_2\end{bmatrix}^{-1}\begin{bmatrix}E_1\\O\end{bmatrix}G, \tag{2.3}$$

where the matrices E_1 and E_2 are defined in (1.4). The authors of the paper [13] derive an explicit expression for the group inverse $(GA)^{\#}$ and later, using this representation (2.2), established the representation (2.3).

Sheng and Chen in [13] also proposed the following Gauss-Jordan elimination algorithm for calculating the representation (2.3):

Algorithm 2.1 Computing the $A^{(2)}_{T,S}$ inverse of the matrix A using the Gauss–Jordan elimination. **(Algorithm GJATS2)**

Require: The matrix A of dimensions $m\times n$ and of rank r.

1: Confirm G and calculate GA.

2: Execute elementary row operations on the pair $[GA \mid I]$ to transform it into

$$E\,[GA \mid I] = \begin{bmatrix} E_1GA & E_1 \\ O & E_2 \end{bmatrix}.$$

3: Exchange the block of zeros with the corresponding block of the lower right–hand of the above 2×2 block matrix, then resuming elementary row operations on the pair

$$\begin{bmatrix} E_1GA & E_1 \\ E_2 & O \end{bmatrix}$$

to transform it into

$$[I \mid Y] = \left[\; I \; \middle| \; \begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1 \\ O \end{bmatrix} \right].$$

4: Compute the output

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = \begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1 \\ O \end{bmatrix} G = YG.$$

The particular case $G = A^*$ the representation (2.3) produces analogous representation of the Moore–Penrose inverse and Algorithm 2.1 reduces to the corresponding algorithm for computing the Moore–Penrose inverse. This representation and algorithm are proposed in [12]. Corresponding algorithm we denote by Algorithm **GJMP**.

On the other hand, the following improvement of Algorithm **GJMP** is recently published in [6]:

Algorithm 2.2 Computing the $A^\dagger$ using the Gauss–Jordan elimination.
(Algorithm GJMP1)

Require: The matrix A of dimensions $m \times n$ and of rank r.

1: Execute elementary row operations on the pair $[A^* \mid I]$ to get the reduced row echelon form

$$E\,[A^* \mid I] = \begin{bmatrix} E_1A^* & E_1 \\ O & E_2 \end{bmatrix} = \begin{bmatrix} B & E_1 \\ O & E_2 \end{bmatrix},$$

where the notation $B = E_1A^*$ is used.

2: Compute BA and form

$$\begin{bmatrix} BA & B \\ E_2 & O \end{bmatrix}$$

to transform it into

$$\left[I \mid A^\dagger\right] = \left[I \mid \begin{bmatrix} BA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} B \\ O \end{bmatrix}\right].$$

3: Return the output

$$A^\dagger = \begin{bmatrix} BA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} B \\ O \end{bmatrix}.$$

Our goal in the present paper is to improve Algorithm **GJATS2** in the same way as Algorithm **GJMP1** improves Algorithm **GJMP**. That gives a coherent set of numerical methods of similar type, which numerical properties are are also examined.

3 The algorithm

We start by proving the main theorem, which gives the representation of $A^{(2)}_{T,S}$ inverse corresponding to matrix G, using incomplete Gauss-Jordan elimination of the matrix $[G \mid I]$.

Theorem 3.1. *Let $A \in \mathbb{C}_r^{m\times n}$ is given matrix. Let $G \in \mathbb{C}_s^{n\times m}$ is given matrix satisfying $0 < s \leq r$. Assume that the condition $AT \oplus S = \mathbb{C}^m$ is satisfied in the case $T = \mathcal{R}(G), S = \mathcal{N}(G)$. Let*

$$\begin{bmatrix} E_1G \\ O \end{bmatrix} = \begin{bmatrix} B \\ O \end{bmatrix}$$

be the reduced row echelon form of G and E is the product of all the elementary matrices derived corresponding to s pivoting steps of Gauss-Jordan elimination on $[G \mid I]$ satisfying

$$E\,[G \mid I] = \begin{bmatrix} B & E_1 \\ O & E_2 \end{bmatrix}.$$

Then the matrix

$$\begin{bmatrix} BA \\ E_2 \end{bmatrix} \tag{3.1}$$

is nonsingular and

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = \begin{bmatrix} BA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} B \\ O \end{bmatrix} = \begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1G \\ O \end{bmatrix}. \tag{3.2}$$

Proof. Denote the first s rows of E by E_1. By E_2 we denote the remaining $n - s$ columns of E. It follows that

$$EG = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} G = \begin{bmatrix} E_1G \\ O \end{bmatrix} = \begin{bmatrix} B \\ O \end{bmatrix}, \tag{3.3}$$

where the notation $B = E_1G$ is used for the sake of simplification. We also have

$$E_2G = O,$$

which implies $\mathcal{R}(G) \subset \mathcal{N}(E_2)$. Due to the fact

$$\dim(\mathcal{N}(E_2)) + \dim(\mathcal{R}(E_2)) = \dim(\mathcal{N}(E_2)) + \operatorname{rank}(E_2) = n$$

and $\operatorname{rank}(E_2) = n - s$ we have

$$\dim(\mathcal{N}(E_2)) = n - (n - s) = s = \operatorname{rank}(G) = \dim(\mathcal{R}(G)),$$

and later

$$\mathcal{N}(E_2) = \mathcal{R}(G). \tag{3.4}$$

Since the identity $\mathcal{R}(G) = T$ holds, we have

$$\mathcal{N}(E_2) = \mathcal{R}(G) = \mathcal{R}(A^{(2)}_{T,S}),$$

which further implies

$$E_2A^{(2)}_{T,S} = O. \tag{3.5}$$

On the other hand, we have

$$BAA^{(2)}_{T,S} = B.$$

Indeed, if $G = UV$ is a full–rank factorization of G, according to Proposition 2.1 we obtain

$$A^{(2)}_{T,S} = U(VAU)^{-1}V$$

and

$$\begin{aligned} BAA^{(2)}_{T,S} &= E_1(UV)AU(VAU)^{-1}V \\ &= E_1UV = E_1G \\ &= B. \end{aligned}$$

The last identity in conjunction with (3.5) implies

$$\begin{bmatrix} BA \\ E_2 \end{bmatrix} A^{(2)}_{T,S} = \begin{bmatrix} B \\ O \end{bmatrix}. \tag{3.6}$$

In order to complete the proof it is necessary to verify invertibility of the matrix

$$\begin{bmatrix} BA \\ E_2 \end{bmatrix}.$$

Let $x \in \mathbb{C}^n$ satisfy

$$\begin{bmatrix} E_1GA \\ E_2 \end{bmatrix} x = 0.$$

Then immediately follows $E_2x = E_1GAx = 0$. The condition $E_2x = 0$ implies

$$x \in \mathcal{N}(E_2) = \mathcal{R}(G) = \mathcal{R}(GA). \tag{3.7}$$

From $E_1GAx = 0$, taking into account (3.4), we have

$$x \in \mathcal{N}(E_1GA) = \mathcal{N}(GA). \tag{3.8}$$

According to assertions (3.7) and (3.8) and Proposition 2.3 we have

$$x \in \mathcal{R}(GA) \cap \mathcal{N}(GA) = \{0\} \Rightarrow x = 0, \tag{3.9}$$

which completes the proof. □

According to the representation introduced in Theorem 3.1, we introduce the following algorithm for computing $A^{(2)}_{T,S}$ inverses:

Algorithm 3.1 Computing the $A^{(2)}_{T,S}$ using the Gauss–Jordan elimination.
(Algorithm GJATS2PM)

Require: The complex matrix A of dimensions $m \times n$ and of rank r.

1: Choose a complex matrix G of dimensions $n \times m$ and of rank $0 < s \leq r$.

2: Perform elementary row operations on the pair $[G|\ I]$ to get the reduced row echelon form

$$E\,[G \mid I] = \begin{bmatrix} E_1G & E_1 \\ O & E_2 \end{bmatrix} = \begin{bmatrix} B & E_1 \\ O & E_2 \end{bmatrix}.$$

3: Compute BA and form the block matrix

$$\begin{bmatrix} BA & B \\ E_2 & O \end{bmatrix}.$$

Transform this matrix into

$$[I \mid X] = \left[\ I \ \middle|\ \begin{bmatrix} BA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} B \\ O \end{bmatrix}\ \right]$$

applying the Gauss–Jordan elimination

4: Return

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = X = \begin{bmatrix} BA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} B \\ O \end{bmatrix}.$$

It is possible to use Algorithm **GJATS2PM** to compute the common six important generalized inverses, for a different choice of input matrices.

Corollary 3.1. *For a given matrix $A \in \mathbb{C}_r^{m\times n}$ and arbitrarily chosen matrix $G \in \mathbb{C}_s^{n\times m}$ the following statements are valid for the generalized inverse $A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ produced by Algorithm* **GJATS2PM***:*

$$\begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1G \\ O \end{bmatrix} = A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = \begin{cases} A^{\dagger}, & G = A^*; \\ A^{\dagger}_{M,N}, & G = A^{\sharp}; \\ A^{D}, & G = A^l,\ l \geq \mathrm{ind}(A); \\ A^{\#}, & G = A; \\ A^{(-1)}_{(L)}, & \mathcal{R}(G) = L,\ \mathcal{N}(G) = L^{\perp}; \\ A^{(\dagger)}_{(L)}, & \mathcal{R}(G) = S,\ \mathcal{N}(G) = S^{\perp}. \end{cases} \tag{3.10}$$

Furthermore, using representations from [15], we derive Gauss–Jordan elimination methods for generating $\{2,4\}$ i $\{2,3\}$–inverses.

Corollary 3.2. *Let $A\in\mathbb{C}_r^{m\times n}$ be the given matrix, $s \leq r$ be a given integer. Assume that the conditions of Theorem 3.1 are satisfied. Then the following statements are valid:*

(a) *If $G = UV$ is arbitrary full-rank factorization of G, then expression* (3.2) *produces*

$$\begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1G \\ O \end{bmatrix} = A^{(2)}_{\mathcal{R}(U),\mathcal{N}(V)} = U(VAU)^{-1}V \in A\{2\}_s. \tag{3.11}$$

(b) *In the case $G = (VA)^*V\in\mathbb{C}_s^{n\times m}$ expression* (3.2) *produces*

$$\begin{aligned} \begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1G \\ O \end{bmatrix} &= A^{(2,4)}_{\mathcal{R}((VA)^*),\mathcal{N}(V)} \\ &= (VA)^*(VA(VA)^*)^{-1}V = (VA)^{\dagger}V \in A\{2,4\}_s. \end{aligned} \tag{3.12}$$

(c) *In the case $G = U(AU)^*\in\mathbb{C}_s^{n\times m}$ the following holds:*

$$\begin{aligned} \begin{bmatrix} E_1GA \\ E_2 \end{bmatrix}^{-1} \begin{bmatrix} E_1G \\ O \end{bmatrix} &= A^{(2,3)}_{\mathcal{R}(U),\mathcal{N}((AU)^*)} \\ &= U((AU)^*AU)^{-1}(AU)^* = U(AU)^{\dagger} \in A\{2,3\}_s. \end{aligned} \tag{3.13}$$

4 Example and improvement of Algorithm GJATS2PM

In this section we illustrate Algorithm **GJATS2PM** on one numerical example and show one improvement that has been used in our implementation. The idea for such improvement is briefly mentioned in [6]. Here we give more details, including the explicit formulation of the modified steps of Algorithm **GJATS2PM**.

Example 4.1. *Consider the following matrices $A \in \mathbb{C}^{7\times 6}$ and $G \in \mathbb{C}^{6\times 7}$:*

$$A = \begin{bmatrix} 0. & 5. & 1. & 8. & 5. & 4. \\ 3. & 8. & 8. & 7. & 1. & 8. \\ 7. & 5. & 3. & 0. & 0. & 3. \\ 1. & 1. & 9. & 4. & 1. & 0. \\ 0. & 3. & 5. & 5. & 6. & 6. \\ 1. & 6. & 4. & 3. & 0. & 6. \\ 7. & 8. & 5. & 3. & 8. & 7. \end{bmatrix}, \qquad G = \begin{bmatrix} 54. & 81. & 18. & 18. & 153. & 18. & 45. \\ 30. & 21. & 10. & 10. & 49. & 2. & 17. \\ 24. & 24. & 8. & 8. & 50. & 4. & 16. \\ 48. & 36. & 16. & 16. & 82. & 4. & 28. \\ 42. & 69. & 14. & 14. & 128. & 16. & 37. \\ 54. & 81. & 18. & 18. & 153. & 18. & 45. \end{bmatrix}.$$

It is not difficult to determine ranks of these matrices: $\mathrm{rank}(A) = 6$, $\mathrm{rank}(G) = 2$. *We apply Algorithm* **GJATS2PM** *to compute $A^{(2)}_{T,S}$ inverse corresponding to the matrix G. After performing Gauss-Jordan*

elimination up to the $n = 7th$ column on the matrix $[G \mid I] \in \mathbb{C}^{6\times 13}$, we obtain

$$\begin{bmatrix} B & E_1 \\ O & E_2 \end{bmatrix}$$

$$= \left[\begin{array}{ccccccc|ccccccc} 1. & 0. & 0.3333 & 0.3333 & 0.5833 & -0.1667 & 0.3333 & 0.2129 & 0. & 0. & 0. & -0.2500 & 0. \\ 0. & 1. & 0. & 0. & 1.5000 & 0.3333 & 0.3333 & -0.1296 & 0. & 0. & 0. & 0.1667 & 0. \\ \hline 0. & 0. & 0. & 0. & 0. & 0. & 0. & -2. & 0. & 1. & 0. & 2. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & -5.5556 & 0. & 0. & 1. & 6. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & -3.6667 & 1. & 0. & 0. & 4. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & -1. & 0. & 0. & 0. & 0. & 1. \end{array}\right] \tag{4.14}$$

Rows of the matrix $[G \mid I]$ are permuted according to the following permutation of rows:

$$\mathbf{row} = (1, 5, 3, 4, 2, 6).$$

Next step is to form the matrix:

$$\begin{bmatrix} E_2 & O \\ BA & B \end{bmatrix}$$

$$= \left[\begin{array}{cccccc|ccccccc} -2. & 0. & 1. & 0. & 2. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -5.5556 & 0. & 0. & 1. & 6. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -3.6667 & 1. & 0. & 0. & 4. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -1. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ \hline 4.8333 & 10.4167 & 8.9167 & 12.7500 & 11.5000 & 9.8333 & 1. & 0. & 0.3333 & 0.3333 & 0.5833 & -0.1667 & 0.3333 \\ 5.6667 & 17.1667 & 18.5000 & 16.5000 & 12.6667 & 21.3333 & 0. & 1. & 0. & 0. & 1.5000 & 0.3333 & 0.3333 \end{array}\right] \tag{4.15}$$

and continue Gauss-Jordan elimination. Note that both matrices (4.14) *and* (4.15) *are rounded on 4 decimals. In fact, double precision is used for representation of all intermediate results. Hence, we finally get*

$$[I \mid X]$$

$$= \left[\begin{array}{cccccc|ccccccc} 1. & 0. & 0. & 0. & 0. & 0. & -4.38857 & 2.84571 & -1.46286 & -1.46286 & 1.70857 & 1.68000 & -0.51429 \\ 0. & 1. & 0. & 0. & 0. & 0. & 3.89587 & -2.50540 & 1.29862 & 1.29862 & -1.48550 & -1.48444 & 0.46349 \\ 0. & 0. & 1. & 0. & 0. & 0. & 1.21651 & -0.77841 & 0.40550 & 0.40550 & -0.45799 & -0.46222 & 0.14603 \\ 0. & 0. & 0. & 1. & 0. & 0. & 5.60000 & -3.60000 & 1.86667 & 1.86667 & -2.13333 & -2.13333 & 0.66667 \\ 0. & 0. & 0. & 0. & 1. & 0. & -4.99683 & 3.23492 & -1.66561 & -1.66561 & 1.93757 & 1.91111 & -0.58730 \\ 0. & 0. & 0. & 0. & 0. & 1. & -4.38857 & 2.84571 & -1.46286 & -1.46286 & 1.70857 & 1.68000 & -0.51429 \end{array}\right]. \tag{4.16}$$

Now the matrix X represents the inverse $A^{(2)}_{T,S}$ corresponding to matrix G.

Previous example leads to one possible improvement of Algorithm **GJATS2PM**. The matrix E_2 consists of several columns which remain unchanged during the first Gauss-Jordan elimination. Those columns are $3, 4, 2$ and 6, or $\mathbf{row}(3), \mathbf{row}(4), \mathbf{row}(5)$ and $\mathbf{row}(6)$. In the general case, columns with indices $\mathbf{row}(s+1), \mathbf{row}(s+2), \ldots, \mathbf{row}(n)$ ($s = \mathrm{rank}(G)$) remain unchanged and correspond to appropriate columns of the identity matrix.

Putting those columns at first s positions, matrix (4.15) becomes

$$\left[\begin{array}{cccccc|ccccccc} 1. & 0. & 0. & 0. & -2. & 2. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. & -5.5556 & 6. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. & -3.6667 & 4. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & -1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ \hline 8.9167 & 12.75 & 10.4167 & 9.8333 & 4.8333 & 11.5 & 1. & 0. & 0.3333 & 0.3333 & 0.5833 & -0.1667 & 0.3333 \\ 18.5 & 16.5 & 17.1667 & 21.3333 & 5.6667 & 12.6667 & 0. & 1. & 0. & 0. & 1.5 & 0.3333 & 0.3333 \end{array}\right]. \tag{4.17}$$

Choosing the main diagonal elements as pivots, in the first $n - r = 4$ steps of Gauss-Jordan elimination, we significantly reduce the number of required arithmetic operations. That is since we do not need to update a submatrix consisting of the first $n - r = 4$ rows and columns of the matrix (4.15). Continuing Gauss-Jordan elimination on the matrix (4.17) we obtain the same matrix X (as in (4.16)), but with permuted rows, according to the permutation $(\mathbf{row}(s+1), \mathbf{row}(s+2), \ldots, \mathbf{row}(n), \mathbf{row}(1), \mathbf{row}(2), \ldots, \mathbf{row}(s))$ as we used for columns of the matrix (4.15).

Shown idea can be used for all input matrices A and G. For a matrix $M \in \mathbb{C}^{m\times n}$ and appropriate permutations $\mathbf{p}$ and $\mathbf{q}$, denote by $M_{\mathbf{p},\bullet}$ and $M_{\bullet,\mathbf{q}}$ matrices formed by permutation of rows and columns according to permutations $\mathbf{p}$ and $\mathbf{q}$ respectively. Assume that

$$\mathbf{row} = (\mathbf{row}(1), \mathbf{row}(2), \ldots, \mathbf{row}(n))$$

is the permutation of rows obtained during Gauss-Jordan elimination procedure. Since there is no pivot element belong to rows $\mathbf{row}(s+1), \mathbf{row}(s+2), \ldots, \mathbf{row}(n)$, columns of the matrix I (in initial matrix $[G \mid I]$) with the same indices are unchanged during the elimination process. Consider the permutation of columns

$$\mathbf{col} = (\mathbf{row}(s+1), \mathbf{row}(s+2), \ldots, \mathbf{row}(n), \mathbf{row}(1), \mathbf{row}(2), \ldots, \mathbf{row}(s))$$

and form

$$\begin{bmatrix} (E_2)_{\bullet,\mathbf{col}} & O \\ (BA)_{\bullet,\mathbf{col}} & B \end{bmatrix}. \tag{4.18}$$

The first s rows and columns of the matrix (4.18) form the identity matrix $I_{s\times s}$. Hence, by choosing the first r main diagonal elements as pivot elements in second Gauss-Jordan elimination, only last $n-r$ rows and $n+m-r$ columns of matrix (4.18) need to be updated in each step. After the elimination is done, the final matrix has the form $[I \mid \tilde{X}]$, where $\tilde{X} = X_{\mathbf{col},\bullet}$. Now, matrix X is easily computed by applying the row permutation $\mathbf{col}^{-1}$ on the matrix $\tilde{X}$, i.e. $X = \tilde{X}_{\mathbf{col}^{-1},\bullet}$.

According to the above discussion, we can formulate the following improvement of Algorithm **GJATS2PM**.

Algorithm 4.2 Improved algorithm for computing the $A^{(2)}_{T,S}$ using the Gauss–Jordan elimination. **(Algorithm GJATS2PMimp)**

Require: The complex matrix A of dimensions $m \times n$ and of rank r.

1: Choose a complex matrix G of dimensions $n \times m$ and of rank $0 < s \leq r$.

2: Execute elementary row operations on the pair $[G|\ I]$ to get the reduced row echelon form

$$E\,[G \mid I] = \begin{bmatrix} E_1G & E_1 \\ O & E_2 \end{bmatrix} = \begin{bmatrix} B & E_1 \\ O & E_2 \end{bmatrix}.$$

During the elimination, maintain the permutation of rows $\mathbf{row} = (\mathbf{row}(1), \mathbf{row}(2), \ldots, \mathbf{row}(n))$.

3: Form the permutation of columns

$$\mathbf{col} = (\mathbf{row}(r+1), \mathbf{row}(r+2), \ldots, \mathbf{row}(n), \mathbf{row}(1), \mathbf{row}(2), \ldots, \mathbf{row}(r)).$$

4: Compute BA and form

$$\begin{bmatrix} (E_2)_{\bullet,\mathbf{col}} & O \\ (BA)_{\bullet,\mathbf{col}} & B \end{bmatrix},$$

to transform it into

$$\left[I \mid \tilde{X}\right] = \left[\begin{array}{c|c} I & \begin{bmatrix} (E_2)_{\bullet,\mathbf{col}} \\ (BA)_{\bullet,\mathbf{col}} \end{bmatrix}^{-1} \begin{bmatrix} O \\ B \end{bmatrix} \end{array}\right].$$

5: Compute $\mathbf{col}^{-1}$ and return

$$A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)} = X = \tilde{X}_{\mathbf{col}^{-1},\bullet}.$$

Note that, in the same way, we can construct an improvement of Algorithm **GJATS2**, which will be denoted by Algorithm **GJATS2imp**.

5 Numerical experiments

It is realistic to expect that two successive applications Gauss-Jordan elimination procedures contribute to bad conditioning of numerical algorithms. We implemented Algorithm **GJATS2imp** and Algorithm **GJATS2PMimp** in programming language `C++` and tested on randomly generated test matrices. Note that papers [12, 13], where Algorithm **GJATS2** is introduced, does not contain any numerical experiments. Same situation is in the paper [6] of J. Ji introducing the special case of Algorithm **GJATS2PM** for computing Moore-Penrose inverse $A^\dagger$ of matrix A. Hence, in this paper, we provide testing results for both Algorithm **GJATS2imp** and **GJATS2PMimp**. Those results include the special cases of Moore-Penrose and Drazin inverse, obtained for the choice $G = A^*$ and $G = A^k$ ($k = \mathrm{ind}(A) = \min\{l \in \mathbb{N} \mid \mathrm{rank}(A^{l+1}) = \mathrm{rank}(A^l)\}$), respectively. In the case of Moore-Penrose inverse, we compared our algorithms with `Matlab` function `pinv` which implements well-known SVD (Singular Value Decomposition) method.

Code is compiled by `Microsoft Visual Studio 2010` compiler with default compiler settings. All generated matrices had unit norm, but different values of rank.

Furthermore, we list the following two more issues we used in the implementation of both Algorithm **GJATS2imp** and Algorithm **GJATS2PMimp**:

1. While performing Gauss-Jordan elimination, we first select non-zero entries in pivot row and column and update only those fields contained in the cross product of those entries. This improvement is based on the fact that (in both algorithms) Gauss-Jordan elimination is applied on matrices containing non-negligible number of zero elements.

2. Note that the matrix B in Algorithm **GJATS2PMimp** has exactly s unit columns and others have at least $n - s$ zeros. This fact can be used to significantly reduce the number of operations (also running time) needed to compute product of matrices B and A. In other words, we can reduce the number of multiplications to
$$\#\{b_{ij} \mid b_{ij} \neq 0,\ i = 1, 2, \ldots, s,\ j = 1, 2, \ldots, m\} \cdot n.$$
where $B = [b_{ij}]_{1 \leq i \leq s, 1 \leq j \leq n}$. Similar fact can be used for the last step of Algorithm **GJATS2imp**.

5.1 Numerical experiments for the Moore-Penrose inverse

In the case $G = A^*$ the resulting matrix X is the Moore-Penrose inverse $A^\dagger$ of the matrix A. Tables 1 and 2 show maximal error norms for matrices $A \in \mathbb{C}^{n \times n}$ with $\mathrm{rank}(A) = n/2$ and $\mathrm{rank}(A) = 10$, obtained for 20 different randomly generated test matrices.

We see that both algorithms give satisfactory results, while Algorithm **GJATS2PMimp** is better, average by 3 orders of magnitude.

Average running times of our algorithms and `pinv` function from `Matlab` are shown in Table 3. Testings are done on Intel Core-i5 720 processor (without multi-core optimization) with 4 GB of RAM. All presented times are in seconds and obtained by averaging times on 20 test matrices.

It can be seen that Algorithm **GJATS2PMimp** outperforms Algorithm **GJATS2imp** in all test cases. One possible reason for such behavior is the fact that Algorithm **GJATS2imp** needs to compute the product A^*A (i.e. GA), where both A^* and A are not sparse.

In the case of low-rank matrices ($\mathrm{rank}(A) = 10$) we see that Algorithm **GJATS2PMimp** also outperforms `pinv`, while for $\mathrm{rank}(A) = n/10$ results are comparable each to other. In the case $\mathrm{rank}(A) = n/2$, both algorithms are slower than `pinv`.

According to the above discussion, we can conclude that Algorithm **GJATS2PMimp** is the best choice for computing $A^\dagger$ of low-rank matrices.

5.2 Numerical experiments for the Drazin inverse

Algorithms **GJATS2imp** and **GJATS2PMimp** are tested in the case $G = A^k$ ($k = \mathrm{ind}A$) where the result matrix X is Drazin inverse A^D of the matrix A. Running times, as well as the residual vectors are shown in tables 4 and 5.

Numerical results show that Algorithm **GJATS2PMimp** also outperformed Algorithm **GJATS2imp** both in the result accuracy and running time. Especially this is the case for low-rank matrices where Algorithm **GJATS2PMimp** gives the best performance.

5.3 Numerical experiments for randomly generated matrix G

Finally, we show numerical results in the case when both matrices A and G are chosen randomly. In such case, obtained matrix X is only $\{2\}$ inverse of A and therefore, we only show the norm of $\|XAX - X\|$. At it can be seen from tables 6, 7 and 8, Algorithm **GJATS2PMimp** clearly outperforms Algorithm **GJATS2imp** in all test cases. Both algorithms have smaller running times for matrices with lower rank. It is worth noting that accuracy also depends on the rank of both A and G and it is drastically reduced when $\text{rank}(A) \approx \text{rank}(G)$.

6 Conclusions

Two main objectives are achieved in the present paper. First, we define several improvements of the algorithm for generating the outer inverses with prescribed range and null space from [13]. Our improvements follow corresponding modifications of the algorithm from [12] which are presented in [6]. In this way, our paper represents a continuation of results given in [6, 11, 12, 13]. Defined algorithm represents an another algorithm for computing outer inverses with prescribed range and null space as well as an algorithm based on the Gauss–Jordan elimination procedure.

In addition, the paper presents a numerical study on the properties of algorithms based on the Gauss–Jordan elimination and aimed in computation of generalized inverses. For this purpose, we give a set of numerical examples to compare these algorithms with several well–known methods for computing the Moore–Penrose inverse and the Drazin inverse.

In this paper we searched for the answer to the important question: how good are methods based on two Gauss–Jordan eliminations? Our numerical experience indicates that the answer depends on the type of inverse which is being computed and the rank of both matrices A and G:

- In the case of Moore-Penrose inverse ($G = A^T$), methods are stable and fast for low-rank matrices A. Both running time and accuracy are degraded for higher rank matrices.
- In the case of Drazin inverse ($G = A^k$, $k = \text{ind}(A)$), running times are similar to the previous case, while accuracy is more reduced with increase of $\text{rank}(A)$.
- Finally, in the general case of arbitrary A and G, we also see that better results are obtained in cases when $\text{rank}(G)$ is small, while accuracy is reduced when $\text{rank}(A) \approx \text{rank}(G)$.

Hence, methods based on Gauss–Jordan elimination are most practically applicable as a unique tool for computation of arbitrary low–rank generalized inverses of matrix A.

References

[1] A. Ben-Israel, T.N.E. Greville, *Generalized inverses: Theory and Applications*, Second Ed., Springer, 2003.

[2] Y. Chen, *The generalized Bott–Duffin inverse and its application*, Linear Algebra Appl. **134** (1990), 71–91.

[3] R.E. Cline, *Inverses of rank invariant powers of a matrix*, SIAM J. Numer. Anal. **5** (1968), 182–197.

[4] A.J. Getson, F.C. Hsuan, {2}-*Inverses and their Statistical Applications*, Lecture Notes in Statistics **47**, Springer, Berlin, 1988.

[5] F. Husen, P. Langenberg, A. Getson, *The {2}-inverse with applications to satistics*, Linear Algebra Appl. **70** (1985), 241-248.

[6] J. Ji, *Gauss-Jordan elimination methods for the Moore-Penrose inverse of a matrix*, Linear Algebra Appl. (2012), http://dx.doi.org/10.1016/j.laa.2012.05.017.

[7] M.Z. Nashed, *Generalized Inverse and Applications*, Academic Press, New York, 1976.

[8] M.Z. Nashed, X. Chen, *Convergence of Newton-like methods for singular operator equations using outer inverses*, Numer. Math. **66** (1993), 235-257.

[9] R. Piziak, P. L. Odell, *Full Rank Factorization of Matrices*, Mathematics Magazine **72** (1999), 193–201.

[10] C.R. Rao, S.K. Mitra, *Generalized Inverse of Matrices and its Applications*, John Wiley & Sons, Inc., New York, London, Sydney, Toronto, 1971.

[11] X. Sheng, G. Chen, *Full–rank representation of generalized inverse $A^{(2)}_{T,S}$ and its applications*, Comput. Math. Appl. **54** (2007), 1422–1430.

[12] X. Sheng, G.L. Chen, *A note of computation for M–P inverse $A^{\dagger}$*, Int. J. Comput. Math. **87** (2010), 2235–2241.

[13] X. Sheng, G.L. Chen, Y. Gong, *The representation and computation of generalized inverse $A^{(2)}_{T,S}$*, J. Comput. Appl. Math. **213** (2008), 248–257.

[14] P.S. Stanimirović, *Block representations of $\{2\},\{1,2\}$ inverses and the Drazin inverse*, Indian J. Pure Appl. Math. **29** (1998), 1159–1176.

[15] P.S. Stanimirović, D.S. Cvetković-Ilić, S. Miljković, M. Miladinović, *Full-rank representations of $\{2,4\},\{2,3\}$-inverses and successive matrix squaring algorithm*, Appl. Math. Comput. **217** (2011), 9358–9367.

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

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

[18] Y. Wei, *A characterization and representation of the generalized inverse $A^{(2)}_{T,S}$ and its applications*, Linear Algebra Appl. **280** (1998), 79–86.

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

n	$\|AXA-A\|$	$\|XAX-X\|$	$\|AX-(AX)^T\|$	$\|XA-(XA)^T\|$
300	3.339e-008	8.835e-007	3.479e-009	1.172e-008
350	4.155e-008	1.899e-007	2.561e-009	8.668e-008
400	1.531e-007	2.844e-007	3.897e-009	1.318e-007
450	7.013e-008	1.515e-007	2.070e-009	3.073e-007
500	3.199e-007	4.253e-007	5.348e-009	1.273e-007
550	2.713e-008	6.711e-007	7.852e-009	7.348e-007
600	1.622e-008	5.577e-007	1.707e-009	1.281e-007
650	3.644e-007	1.374e-006	1.419e-008	4.342e-007
700	3.764e-006	1.515e-006	1.525e-008	2.146e-006

a) The case rank$A = n/2$.

n	$\|AXA-A\|$	$\|XAX-X\|$	$\|AX-(AX)^T\|$	$\|XA-(XA)^T\|$
300	4.382e-012	2.216e-013	2.963e-013	4.002e-012
350	1.160e-012	1.299e-013	1.242e-013	4.003e-012
400	1.943e-012	1.423e-013	1.669e-013	3.319e-012
450	2.237e-012	1.864e-013	1.811e-013	2.992e-012
500	4.140e-012	1.571e-013	2.347e-013	4.421e-012
550	1.003e-011	1.584e-013	1.215e-012	8.514e-012
600	5.490e-012	2.536e-013	1.455e-012	2.307e-011
650	1.577e-012	1.985e-013	3.212e-013	4.745e-012
700	1.122e-011	2.997e-013	4.213e-013	3.457e-012

b) The case rank$A = 10$.

Table 1: Maximal error norms of the result of Algorithm **GJATS2imp**, on random matrices.

n	$\|AXA-A\|$	$\|XAX-X\|$	$\|AX-(AX)^T\|$	$\|XA-(XA)^T\|$
300	5.275e-012	5.035e-010	3.298e-011	2.654e-011
350	1.276e-011	4.166e-008	1.030e-009	1.028e-009
400	4.204e-012	2.215e-009	5.602e-011	5.719e-011
450	1.057e-011	1.944e-008	4.954e-010	4.769e-010
500	9.138e-012	3.988e-008	8.342e-010	8.623e-010
550	6.269e-012	1.736e-009	3.723e-011	3.704e-011
600	1.647e-012	4.279e-009	1.160e-010	1.286e-010
650	3.299e-011	1.964e-007	3.558e-009	4.000e-009
700	8.328e-010	3.680e-009	1.370e-009	1.505e-009

a) The case rank$A = n/2$.

n	$\|AXA-A\|$	$\|XAX-X\|$	$\|AX-(AX)^T\|$	$\|XA-(XA)^T\|$
300	1.938e-012	3.021e-015	1.016e-013	1.130e-013
350	1.397e-012	1.141e-015	5.099e-014	5.784e-014
400	1.279e-012	8.597e-016	4.093e-014	4.293e-014
450	2.930e-012	1.793e-015	9.655e-014	8.348e-014
500	1.117e-011	5.935e-015	3.084e-013	3.239e-013
550	2.989e-011	1.118e-014	7.070e-013	8.488e-013
600	3.423e-012	8.750e-016	7.263e-014	9.354e-014
650	4.555e-012	1.192e-015	9.920e-014	1.026e-013
700	8.849e-012	2.323e-015	2.003e-013	2.069e-013

b) The case rank$A = 10$.

Table 2: Maximal error norms of the result of Algorithm **GJATS2PMimp**, on random matrices.

n	`pinv`	**GJATS2imp**	**GJATS2PMimp**
300	0.016	0.040	0.006
350	0.022	0.064	0.008
400	0.031	0.092	0.011
450	0.044	0.128	0.015
500	0.057	0.173	0.020
550	0.082	0.235	0.023
600	0.098	0.293	0.026
650	0.122	0.373	0.032
700	0.128	0.461	0.037

rank$(A) = 10$

n	`pinv`	**GJATS2imp**	**GJATS2PMimp**
300	0.019	0.048	0.014
350	0.023	0.076	0.020
400	0.032	0.111	0.030
450	0.043	0.158	0.047
500	0.056	0.217	0.060
550	0.071	0.288	0.079
600	0.112	0.377	0.101
650	0.130	0.472	0.129
700	0.144	0.592	0.164

rank$(A) = n/10$

n	`pinv`	**GJATS2imp**	**GJATS2PMimp**
300	0.022	0.097	0.057
350	0.030	0.151	0.089
400	0.043	0.226	0.131
450	0.059	0.319	0.185
500	0.075	0.439	0.263
550	0.096	0.584	0.343
600	0.124	0.761	0.434
650	0.148	0.953	0.551
700	0.172	1.198	0.701

rank$(A) = n/2$

Table 3: Comparison of the running times for `Matlab` function `pinv`, Algorithm **GJATS2imp** and Algorithm **GJATS2PMimp**. All times are in seconds.

n	Running time [s]	$\|A^{k+1}X - A^k\|$	$\|XAX - X\|$	$\|AX - XA\|$
300	0.056	4.666e-007	7.661e-003	1.789e-005
350	0.086	1.578e-008	5.237e-005	1.408e-007
400	0.129	9.349e-007	8.507e-003	2.957e-006
450	0.183	5.138e-007	4.327e-002	2.154e-005
500	0.256	2.154e-007	1.033e-002	1.856e-005
550	0.332	9.233e-007	5.782e-004	5.745e-007
600	0.425	4.101e-007	1.576e-001	2.103e-005
650	0.541	9.463e-008	7.659e-004	4.400e-007
700	0.698	7.869e-008	6.755e-004	1.282e-006

a) Case rank$(A) = n/2$.

n	Running time [s]	$\|A^{k+1}X - A^k\|$	$\|XAX - X\|$	$\|AX - XA\|$
300	0.006	2.501e-011	6.186e-009	3.357e-010
350	0.009	5.713e-009	2.125e-006	9.657e-008
400	0.011	2.912e-010	3.992e-008	1.641e-009
450	0.015	3.037e-010	3.718e-008	1.238e-009
500	0.018	2.923e-010	5.487e-008	2.038e-009
550	0.022	4.160e-010	2.910e-007	2.062e-009
600	0.027	1.570e-008	2.142e-006	3.913e-008
650	0.032	2.424e-010	1.584e-007	7.107e-009
700	0.038	7.359e-010	1.463e-006	1.697e-008

b) Case rank$(A) = 10$.

Table 4: Running times and maximal error norms of the result of Algorithm **GJATS2PMimp** for random matrices.

n	Running time [s]	$\|A^{k+1}X - A^k\|$	$\|XAX - X\|$	$\|AX - XA\|$
300	0.097	7.519e-006	5.708e-003	6.331e-005
350	0.149	1.248e-006	1.432e-003	1.364e-005
400	0.222	1.011e-005	8.619e-002	1.722e-005
450	0.326	1.233e-005	1.792e-003	6.304e-005
500	0.438	2.347e-005	7.228e-003	8.896e-005
550	0.591	4.670e-005	1.400e-003	4.472e-004
600	0.759	5.907e-005	4.975e-003	1.184e-005
650	0.947	4.620e-006	7.662e-002	2.380e-004
700	1.202	2.145e-006	1.836e-003	4.503e-004

a) Case rank$A = n/2$.

n	Running time [s]	$\|A^{k+1}X - A^k\|$	$\|XAX - X\|$	$\|AX - XA\|$
300	0.041	1.840e-010	5.096e-009	9.251e-010
350	0.063	5.626e-010	9.591e-008	6.579e-009
400	0.091	1.853e-009	5.527e-007	7.432e-009
450	0.133	7.938e-010	1.233e-007	3.885e-009
500	0.177	4.871e-010	3.989e-008	3.767e-009
550	0.230	4.396e-009	2.045e-007	8.048e-008
600	0.305	2.577e-009	4.200e-007	3.045e-008
650	0.393	1.755e-009	1.126e-007	1.384e-008
700	0.462	8.892e-009	7.409e-007	6.244e-008

b) Case rank$(A) = 10$.

Table 5: Running times and maximal error norms of the result of Algorithm **GJATS2imp** for random matrices.

n	**GJATS2imp**		**GJATS2PMimp**	
	Running time [s]	$\|XAX - X\|$	Running time [s]	$\|XAX - X\|$
300	0.095	8.364e-009	0.057	1.073e-010
350	0.151	1.169e-008	0.087	4.560e-010
400	0.226	2.719e-009	0.129	1.203e-010
450	0.312	2.186e-007	0.185	1.071e-010
500	0.439	7.630e-001	0.262	6.420e-010
550	0.602	6.427e-008	0.334	3.717e-010
600	0.738	3.383e-005	0.431	1.117e-006
650	0.941	1.953e-008	0.578	9.035e-010
700	1.192	3.983e-008	0.719	3.476e-010

Table 6: Running times and maximal error norms of results of Algorithms **GJATS2imp** and **GJATS2PMimp** for randomly generated matrices A and G with rank$(A) = n$ and rank$(G) = n/2$.

n	**GJATS2imp**		**GJATS2PMimp**	
	Running time [s]	$\|XAX - X\|$	Running time [s]	$\|XAX - X\|$
300	0.039	8.936e-007	0.006	3.939e-006
350	0.063	8.309e-006	0.008	6.899e-007
400	0.094	1.868e-007	0.012	1.113e-006
450	0.129	2.835e-007	0.015	8.278e-007
500	0.175	8.142e-008	0.020	4.975e-008
550	0.230	2.054e-007	0.023	3.943e-007
600	0.294	4.490e-007	0.023	5.761e-007
650	0.385	1.708e-006	0.036	2.072e-007
700	0.460	4.150e-007	0.040	3.417e-006

Table 7: Running times and maximal error norms of results of Algorithms **GJATS2imp** and **GJATS2PMimp** for randomly generated matrices A and G with $\text{rank}(A) = n/2$ and $\text{rank}(G) = 10$.

n	**GJATS2imp**		**GJATS2PMimp**	
	Running time [s]	$\|XAX - X\|$	Running time [s]	$\|XAX - X\|$
300	0.096	1.074e-001	0.056	4.654e-003
350	0.151	2.426e-003	0.087	1.641e-005
400	0.225	7.318e-002	0.130	1.077e-003
450	0.317	1.130e+000	0.204	3.251e-003
500	0.440	2.739e-001	0.295	2.423e-002
550	0.587	1.085e+001	0.364	4.872e-001
600	0.738	6.261e+000	0.469	3.024e-001
650	0.943	1.383e+000	0.603	3.619e-002
700	1.190	4.178e-001	0.739	1.240e-001

Table 8: Running times and maximal error norms of results of Algorithms **GJATS2imp** and **GJATS2PMimp** for randomly generated matrices A and G with $\text{rank}(A) = n/2$ and $\text{rank}(G) = n/2$.