Journal of Mathematics Research; Vol. 6, No. 3; 2014
ISSN 1916-9795 E-ISSN 1916-9809
Published by Canadian Center of Science and Education

Computing Outer Inverses Using Complete Orthogonal Factorizations

Bilall I. Shaini[1]

[1] State University of Tetova, Rr. e Ilindenit, p.n., Tetovo, R. Macedonia

Correspondence: Bilall I. Shaini, State University of Tetova, Rr. e Ilindenit, p.n., Tetovo, R. Macedonia. E-mail: bilall.shaini@unite.edu.mk

Received: May 16, 2014 Accepted: June 10, 2014 Online Published: August 4, 2014

doi:10.5539/jmr.v6n3p91 URL: http://dx.doi.org/10.5539/jmr.v6n3p91

Abstract

Several full-rank representations of the $A^{(2)}_{T,S}$ inverse of a given constant complex matrix, which are based on various complete orthogonal factorizations, are introduced. Particularly, we introduce a full rank representation based on the Singular Value Decomposition (SVD) as well as on a combination of the QR decomposition and the SVD of the triangular matrix produced by the QR decomposition. Furthermore, representations based on the factorizations to a bidiagonal form are defined. The representations arising from reductions to bidiagonal form are applicable to real full row rank matrices. Illustrative numerical examples as well as an extensive numerical study are presented. A comparison of three introduced methods is presented.

Keywords: generalized inverse $A^{(2)}_{T,S}$, complete orthogonal factorizations, SVD factorization, QR factorization, full-rank representation

1. Introduction

Computation of generalized inverses by means of various matrix decompositions has been extensively investigated in the scientific literature.

A fast computational method for computing the Moore-Penrose inverse $A^{\dagger}$ based on the QR decomposition of the matrix A is introduced in (Katsikis et al., 2011). The QR decomposition is assumed to be defined as in Theorem 3.3.11 in (Watkins, 2002) and its extension to complex matrices is used from (Godall, 1993). An extension of the representation introduced in (Katsikis et al., 2011) to the set of outer inverses with prescribed range and null space is presented in (Stanimirović et al., 2012b).

Symbolic computation of $A^{(2)}_{T,S}$ inverses on the basis of the QDR decomposition of the matrix W is presented in (Stanimirović et al., 2012). The canonical form of the DMP inverse $A^{D}AA^{\dagger}$, of a square matrix A, is presented in (Malik & Thome, 2014). This representation of the DMP inverse is based on the Hartwig-Spindelböck matrix decomposition. The authors of the paper (Issa & Bdair, 2014) introduced some new bounds for the zeros of polynomials by using the QR and LU decompositions of the corresponding companion matrices. Recently, representations of $\{2,4\}$ and $\{2,3\}$ generalized inverses based on the SVD are considered in (Shaini & Hoxha, 2013).

In the present paper we develop several numerical algorithms for computing $A^{(2)}_{T,S}$ inverses. These algorithms are based on the full rank representation of an appropriately chosen $n \times m$ matrix W arising from various complete orthogonal factorizations of W.

Our second intention is to examine properties of the introduced methods. For this reason, the paper presents a greater number of numerical experiments in which the introduced methods are compared with each other.

The paper is organized as follows. The second section surveys some useful basic notions and notations concerning generalized inverses. In the third section we derive two numerical algorithms for computing outer inverses. These algorithms are based on the SVD factorization of an appropriately chosen matrix W as well as on the successive application of QR decomposition and SVD. Representations based on bidiagonal forms, applicable to real full row rank matrices, are introduced in Section 4. Numerical examples on various test matrices are presented in the last section.

2. Preliminaries

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

For any matrix A of the order $m\times n$ consider the following matrix equations in X, where $*$ denotes conjugate and transpose:

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

In the case $m=n$ we also consider the following equations

$$(5)\quad AX=XA \qquad (1^k)\quad A^{k+1}X=A^k.$$

For a sequence $\mathcal{S}$ of elements from the set $\{1,2,3,4,5,1^k\}$, the set of matrices obeying the equations labeled by the numbers collected in $\mathcal{S}$ is denoted by $A\{\mathcal{S}\}$. A matrix from $A\{\mathcal{S}\}$ is called an $\mathcal{S}$-inverse of A. The matrix $X=A^\dagger$ is said to be the Moore-Penrose inverse of A satisfying equations (1)–(4). The group inverse $A^\#$ is the unique $\{1,2,5\}$ inverse of A, and exists if and only if $\mathrm{ind}(A)=\min_k\{k|\,\mathrm{rank}(A^{k+1})=\mathrm{rank}(A^k)\}=1$. A matrix $X=A^D$ is said to be the Drazin inverse of A if (1^k) (for some positive integer k), (2) and (5) are satisfied. In the case $\mathrm{ind}(A)=1$, the Drazin inverse of A is equal to the group inverse $A^\#$ of A.

The rank of generalized inverse X is important, and it will be convenient to consider the subset $A\{i,j,k\}_s$ of $A\{i,j,k\}$, consisting $\{i,j,k\}$-inverses of rank s (see Ben-Israel & Greville, 2003).

If $A\in\mathbb{R}_r^{m\times n}$, T is a subspace of $\mathbb{R}^n$ of dimension $t\le r$ and S is a subspace of $\mathbb{R}^m$ 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 $AT\oplus S=\mathbb{R}^m$, in which case X is unique and it is denoted by $A^{(2)}_{T,S}$. The outer generalized inverses with prescribed range and null-space are of the special importance in matrix theory. The $\{2\}$-inverses have application in constructing the iterative methods for solving the nonlinear equations (Ben-Israel & Greville, 2003; Nashed, 1993) as well as in statistics (Getson & Hsuan, 1988; Husen & Langenberg, 1985). 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 inverse (Nashed, 1976; Zheng & Bapat, 2004). On the other hand, it is well known that the Moore-Penrose inverse $A^\dagger$ and the weighted Moore-Penrose inverse $A^\dagger_{M,N}$, the Drazin inverse A^D and the group inverse $A^\#$, 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 is valid for a rectangular matrix A (Ben-Israel & Greville, 2003):

$$A^\dagger=A^{(2)}_{\mathcal{R}(A^T),\mathcal{N}(A^T)},\quad A^\dagger_{M,N}=A^{(2)}_{\mathcal{R}(A^\sharp),\mathcal{N}(A^\sharp)}, \tag{1}$$

where M,N are positive definite matrices of appropriate orders and $A^\sharp=N^{-1}A^TM$. For a given square matrix A the next identities are satisfied (Ben-Israel & Greville, 2003; Chen, 1990; Wang et al., 2004):

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

In the next proposition we restate the full-rank representation of $\{2\}$-inverses with prescribed range and null space from (Shen & Cheng, 2007).

Proposition 1 (Shen & Cheng, 2007) *Let $A\in\mathbb{C}_r^{m\times n}$, T be a subspace of $\mathbb{C}^n$ of dimension $s\le r$ and let S be a subspace of $\mathbb{C}^m$ of dimensions $m-s$. In addition, suppose that $W\in\mathbb{C}^{n\times m}$ satisfies $\mathcal{R}(W)=T,\mathcal{N}(W)=S$. Let W has an arbitrary full-rank decomposition, that is $W=FG$. If A has a $\{2\}$-inverse $A^{(2)}_{T,S}$, then:*

(1) GAF is an invertible matrix;

(2) $A^{(2)}_{T,S}=F(GAF)^{-1}G$.

3. Representations Based on QRD and SVD

In this section, it is assumed that $A\in\mathbb{C}_r^{m\times n}$ is an input matrix and $W\in\mathbb{C}_s^{n\times m}$, $0<s\le r$ is an arbitrary but fixed matrix. A complete orthogonal factorization of W is defined by

$$W=U\begin{bmatrix}T&0\\0&0\end{bmatrix}V^*, \tag{3}$$

where T is a square nonsingular matrix of dimension $s \times s$, $s = \text{rank}(W)$. It is assumed that the zero blocks in positions (1, 2) and (2, 2) of the block matrix in (3) may vanish. In order to simplify computations and derive a full-rank representation of W, it is necessary to consider partitions of matrices U and V into appropriate blocks

$$U = \begin{bmatrix} U_s & U_R \end{bmatrix}, \quad V = \begin{bmatrix} V_s & V_R \end{bmatrix}, \tag{4}$$

where U_s and V_s denote the first s columns of U and V, respectively. Then

$$W = U_s T V_s^*$$

is a full-rank factorization of W obtained from the complete orthogonal factorization (3).

In this section we will prove that the QR factorization and the SVD are various appearances of the complete orthogonal factorization. Also, in Section 4 we show that a complete orthogonal factorization of W can be derived from its reduction to a bidiagonal form.

Suppose that the QR factorization of W is of the form

$$WP = QR, \tag{5}$$

where P is an $m \times m$ permutation matrix, $Q \in \mathbb{C}^{n\times n}$, $Q^*Q = I_n$ and $R \in \mathbb{C}_s^{n\times m}$ is an upper trapezoidal matrix. Assume that Q and R are partitioned as

$$Q = \begin{bmatrix} Q_s & Q_R \end{bmatrix}, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ O & O \end{bmatrix} = \begin{bmatrix} R_1 \\ O \end{bmatrix}, \tag{6}$$

where Q_s consists of the first s columns of the matrix Q and $R_{11} \in \mathbb{C}^{s\times s}$ is nonsingular.

The idea to calculate the Moore-Penrose inverse using the QR decomposition is originated in (Katsikis et al., 2011): if $AP = QR$ is a QR factorization of A, then $A^\dagger = PR^\dagger Q^*$. A generalization of this result to the set of outer inverses is introduced in (Stanimirović et al., 2012b): if A has a $\{2\}$-inverse $A^{(2)}_{\mathcal{R}(W),\mathcal{N}(W)}$, then $R_1P^*AQ_s$ is an invertible matrix and

$$A^{(2)}_{\mathcal{R}(W),\mathcal{N}(W)} = Q_s(R_1P^*AQ_s)^{-1}R_1P^* = A^{(2)}_{\mathcal{R}(Q_s),\mathcal{N}(R_1P^*)}. \tag{7}$$

The QR factorization (6) is not a complete orthogonal factorization unless $R_{12} = 0$. Therefore, (7) is not derived from the complete orthogonal factorization. However, R_{12} can be eliminated by applying further orthogonal (or unitary) transformations from the right to the upper trapezoidal matrix $\begin{bmatrix} R_{11} & R_{12} \end{bmatrix}$:

$$\begin{bmatrix} R_{11} & R_{12} \end{bmatrix} Z = \begin{bmatrix} T_{11} & 0 \end{bmatrix}.$$

This gives the complete orthogonal factorization of W:

$$W = Q \begin{bmatrix} T_{11} & 0 \\ 0 & 0 \end{bmatrix} (PZ)^*. \tag{8}$$

Therefore, (8) is the complete orthogonal factorization of W arising from (5), where T is equal to T_{11}. Furthermore, $W = Q_sT_{11}(PZ)^*$ is the full-rank factorization of W (called the QR full-rank factorization). The following representation can be derived as an analogy of (7).

Corollary 1 *Let $A \in \mathbb{C}_r^{m\times n}$ be the given matrix and $W \in \mathbb{C}_s^{n\times m}$, $s \le r$ be selected. Assume that* (8) *the complete orthogonal factorization of W. If A has a* $\{2\}$*-inverse $A^{(2)}_{\mathcal{R}(W),\mathcal{N}(W)}$, then $T_{11}(PZ)^*AQ_s$ is an invertible matrix and*

$$A^{(2)}_{\mathcal{R}(W),\mathcal{N}(W)} = Q_s \left(T_{11}(PZ)^*AQ_s\right)^{-1} T_{11}(PZ)^* = A^{(2)}_{\mathcal{R}(Q_s),\mathcal{N}((PZ)^*)}. \tag{9}$$

Suppose that the SVD factorization of $W \in \mathbb{C}_s^{n\times m}$ is of the general form

$$W = U\Sigma V^*, \tag{10}$$

where $U \in \mathbb{C}_s^{n\times n}$ and $V \in \mathbb{C}_s^{m\times m}$ are column-orthogonal and $\Sigma \in \mathbb{C}_s^{n\times m}$ is a diagonal matrix with the singular values of W in descending order $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_s$ on the main diagonal.

The SVD is a complete orthogonal factorization in which the block T is the diagonal matrix with singular values of W on the main diagonal and U, V are left orthogonal. It is known that the SVD (10) can be presented in a more efficient form

$$W = U_s \Sigma_s V_s^*, \tag{11}$$

where

$$\begin{gathered} U = \begin{bmatrix} U_s & U_R \end{bmatrix},\ V = \begin{bmatrix} V_s & V_R \end{bmatrix},\ \Sigma = \begin{bmatrix} \Sigma_s & O \\ O & O \end{bmatrix}, \\ U_s \in \mathbb{C}^{m\times s},\ V_s \in \mathbb{C}^{n\times s},\ \Sigma_s = \operatorname{diag}\{\sigma_1, \ldots, \sigma_s\}. \end{gathered} \tag{12}$$

Further, discarding small singular values in Σ_s, it is possible to use the Truncated Singular Value Decomposition ($TSVD$) of W and generate a full-rank factorization of its approximation $W_{(t)}$, defined by

$$W_{(t)} = U_t \Sigma_t V_t^*,\ t \le s. \tag{13}$$

Immediately from Proposition and the full rank decomposition (13) we obtain the following representation of outer inverses.

Lemma 1 *Let $A \in \mathbb{C}_r^{m\times n}$ be the given matrix and $W \in \mathbb{C}_s^{n\times m}$, $s \le r$ be selected. If* (13) *is the full-rank factorization of $W_{(t)}$ then the following is valid*

$$A^{(2)}_{\mathcal{N}(U_t),\mathcal{N}(V_t^*)} = U_t \left(\Sigma_t V_t^* A U_t\right)^{-1} \Sigma_t V_t^*,\ t \le s. \tag{14}$$

Algorithm 1 defines the method for computing the outer inverse of A which is based on the SVD full-rank decomposition of W.

Algorithm 1 Computing the $A^{(2)}_{T,S}$ inverse of the matrix A using (14).
(Algorithm SVDATS2)

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

1: Choose arbitrary but fixed $n \times m$ matrix W of rank $s \le r$.
2: Compute the SVD full-rank decomposition of the matrix $W_{(t)}$ in the form (13).
3: Solve the matrix equation

$$\Sigma_t V_t^* A U_t X = \Sigma_t V_t^*$$

with respect to unknown matrix X.
4: Compute the output

$$A^{(2)}_{\mathcal{N}(U_t),\mathcal{N}(V_t^*)} = U_t X.$$

We also define representations based on a combination of the QR decomposition of A and the SVD of the triangular matrix R. This representation is known from (Moor, 1991, 1992). In applications where $m \gg n$, it is often a good idea to use the SVD of the triangular factor that appears in the QR decomposition of A (see Moor, 1991):

$$\begin{aligned} A &= QR \\ &= Q\left(U^R \Sigma^R \left(V^R\right)^*\right) \\ &= \left(QU_r^R\right)\left(\Sigma_r^R \left(V_r^R\right)^*\right), \end{aligned} \tag{15}$$

where U_r^R contains first r columns of U^R, $\Sigma_r = \operatorname{diag}\{\sigma_1^R, \ldots, \sigma_r^R\}$, V_r^R contains first r columns of V^R.

The full-rank factorization (15) of A gives us idea to apply SVD of the triangular factor arising from the QR decomposition of $W \in \mathbb{R}_s^{n\times m}$, $0 < s \le r$:

$$WP \ = \ Q_W R_W = Q_W (U \Sigma V^*). \tag{16}$$

Later, one can use the following truncated form of (16):

$$W_{(t)} = (Q_W U_t)\Sigma_t V_t^* P^*,\ 0 < t \le s \le r, \tag{17}$$

where U_t contains first t columns of U, $\Sigma_t = \mathrm{diag}\{\sigma_1^R, \ldots, \sigma_t^R\}$ is a diagonal matrix defined by the first t singular values $\sigma_1^R, \ldots, \sigma_t^R$ of R_W and V_t contains first t columns of V. Clearly, (17) is a full-rank factorization of $W_{(t)}$. Therefore, the induced outer inverse of A with the range $\mathcal{R}(Q_W)$ and null space $\mathcal{N}(V_t^* P^*)$ is defined by

$$A^{(2)}_{\mathcal{R}(Q_W),\mathcal{N}(V_t^* P^*)} = (Q_W U_t)\left(\Sigma_t V_t^* P^* A Q_W U_t\right)^{-1} \Sigma_t V_t^* P^*. \tag{18}$$

Algorithm 2 defines the method for computing outer inverse of A which is defined in (18).

Algorithm 2 Computing the $A^{(2)}_{T,S}$ inverse of the matrix A using the full-rank representation (18).
(Algorithm QRSVDATS2)

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

1: Choose arbitrary but fixed $n \times m$ matrix W of rank $s \leq r$.
2: Compute the QR decomposition of the matrix W in the form (16).
3: Compute the $TSVD$ decomposition of the matrix R_W and derive the factorization (17) of $W_{(t)}$.
4: Solve the matrix equation

$$\Sigma_t V_t^* P^* A Q_W U_t X = \Sigma_t V_t^* P^*$$

with respect to unknown matrix X.
5: Compute the output

$$A^{(2)}_{\mathcal{R}(Q_W),\mathcal{N}(V_t^* P^*)} = Q_W U_t X.$$

4. Representations Based on Bidiagonal Forms

In this section we generalize two algorithms from (Smoktunowicz & Wróbel, 2012). The first algorithm from (Smoktunowicz & Wróbel, 2012) is restated in Algorithm 3.

Algorithm 3 Computing $A^\dagger$ of a full-column rank matrix A.
(Algorithm V (Bidiag1) from (Smoktunowicz & Wróbel, 2012))

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

1: Use Golub-Kahan algorithm to reduce A to a bidiagonal form $A = UBV^T$, where $U \in \mathbb{R}^{m\times n}$ is left orthogonal (i.e. $U^T U = I_n$), $V \in \mathbb{R}^{n\times n}$ is orthogonal and $B \in \mathbb{R}^{n\times n}$ is bidiagonal (i.e. the nonzero elements can be located on the main diagonal or on the superdiagonal only).
2: Solve the equation $BY = U^T$ for Y by back substitution.
3: Compute the output

$$X = VY = A^\dagger = VB^{-1}U^T.$$

The second algorithm from (Smoktunowicz & Wróbel, 2012) is restated in Algorithm 4.

Algorithm 4 Computing $A^\dagger$ of a full-column rank matrix A.
(Algorithm VI (Bidiag2) from (Smoktunowicz & Wróbel, 2012))

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

1: Find the QR decomposition of A ($A = QR$).
2: Bidiagonalize R, as described in Algorithm V: $R = UBV^T$, where $U \in \mathbb{R}^{n\times n}$ is orthogonal (i.e. $U^T U = I_n$), $V \in \mathbb{R}^{n\times n}$ is orthogonal, and $B \in \mathbb{R}^{n\times n}$ is bidiagonal.
3: Solve the equation $BY = (QU)^T$ for Y by back substitution.
4: Compute the output

$$X = VY = A^\dagger = VB^{-1}(QU)^T.$$

A complete orthogonal factorization of W can be obtained starting from its bidiagonal form. Introduced generalizations of Algorithm 3 and Algorithm 4 are applicable to real full row rank matrices. If A is of the order $m \times n$ and of rank $m \leq n$, introduced algorithms generate the set of outer inverses of A of rank m, denoted by $A\{2\}_m$. The bidiagonal form of the matrix $W \in \mathbb{R}_m^{n\times m}$, defined in (Ralha, 2003), is used to compute outer inverses of A. As it is stated in (Smoktunowicz & Wróbel, 2012), the assumption $\mathrm{rank}(W) = m$ guarantees invertibility of B. This means that $W = UBV^T$ is a complete orthogonal factorization of A. In the sequel we define a method to obtain a complete orthogonal factorization of $W \in \mathbb{R}_m^{n\times m}$ starting from its bidiagonal form.

Proposition 2 (Ralha, 2003) *Let $W \in \mathbb{R}^{n\times m}$ satisfy $n \geq m = \text{rank}(W)$. There exist orthogonal $U \in \mathbb{R}^{n\times n}$, orthogonal $V \in \mathbb{R}^{m\times m}$ and upper bidiagonal matrix*

$$B = \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} \in \mathbb{R}^{n\times m},$$

where the block $\tilde{B}$ is bidiagonal, such that the following factorization of W holds:

$$W = UBV^T. \tag{19}$$

Theorem 1 *Let $A \in \mathbb{R}_m^{m\times n}$ be a given matrix satisfying $n \geq m$ and $W \in \mathbb{R}_n^{n\times m}$ be a selected but fixed matrix. Let* (19) *be the bidiagonal form of W. Let U be partitioned into blocks*

$$U = \begin{bmatrix} U_m & U_R \end{bmatrix},$$

where U_m denotes the first m rows of U. Then

$$W = U_m \tilde{B} V^T \tag{20}$$

is a complete orthogonal factorization of W. Also,

$$W = FG, \;\; F = U_m, \; G = \tilde{B}V^T \tag{21}$$

is a full-rank factorization of Wand

$$A^{(2)}_{\mathcal{R}(U_m),\mathcal{N}(V^T)} = U_m \left(\tilde{B}V^T A U_m\right)^{-1} \tilde{B}V^T. \tag{22}$$

Using the representation (22) from Theorem we are in a position to state Algorithm 5.

Algorithm 5 Computing the $A^{(2)}_{T,S}$ inverse of $A \in \mathbb{R}^{m\times n}$ using (22).
(Algorithm Bidiag1ATS2)

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

1: Choose arbitrary but fixed $n \times m$ matrix W of rank m.
2: Compute the factorization of the matrix W in the form (20).
3: Solve the matrix equation

$$\tilde{B}V^T A U_m X = \tilde{B}V^T$$

with respect to unknown matrix X.
4: Compute the output

$$A^{(2)}_{\mathcal{R}(U_m),\mathcal{N}(V^T)} = U_m X.$$

In order to generalize Algorithm 4, in the sequel we generalize the idea of the reduction to bidiagonal form in two stages. If $W \in \mathbb{R}_m^{n\times m}$, the generalization is justified in the case when n is much larger than m. A QR decomposition of $W \in \mathbb{R}_m^{n\times m}$ is performed in the first stage:

$$W = QR = \begin{bmatrix} Q_m & Q_R \end{bmatrix} \begin{bmatrix} R_m \\ 0 \end{bmatrix}, \tag{23}$$

where $R_m \in \mathbb{R}^{m\times m}$ is upper triangular and Q_m denotes the matrix constructed from the first m columns of Q. The second stage requires reduction of the relatively small matrix R_m into a bidiagonal form

$$R_m = \tilde{U}\tilde{B}\tilde{V}^T. \tag{24}$$

All matrices included in the decomposition (24) are of the order $m \times m$, $\tilde{U}, \tilde{V}$ are orthogonal and $\tilde{B}$ is bidiagonal. Decompositions (23) and (24) imply the following representation of outer inverses of A.

Theorem 2 *Let $A \in \mathbb{R}_m^{m\times n}$ be given and $W \in \mathbb{R}_m^{n\times m}$ be chosen matrix. If* (23) *is a QR decomposition of W and* (24) *is a bidiagonal form of R_m, then*

$$A^{(2)}_{\mathcal{R}(Q_m),\mathcal{N}(V^T)} = Q_m \tilde{U} \left(\tilde{B}\tilde{V}^T A Q_m \tilde{U}\right)^{-1} \tilde{B}\tilde{V}^T. \tag{25}$$

Proof. We have

$$\begin{aligned} W &= \begin{bmatrix} Q_m & Q_R \end{bmatrix} \begin{bmatrix} \tilde{U}\tilde{B}\tilde{V}^T \\ 0 \end{bmatrix} \\ &= Q_m \tilde{U}\tilde{B}\tilde{V}^T. \end{aligned} \tag{26}$$

Since

$$W = FG, \;\; F = Q_m\tilde{U}, \; G = \tilde{B}\tilde{V}^T$$

is a full-rank factorization of W, the proof immediately follows from the representation of outer inverses with prescribed range and null space from (Shen & Cheng, 2007). □

Algorithm 6 Computing the $A^{(2)}_{T,S}$ inverse of the matrix $A \in \mathbb{R}^{m\times n}$ using (25).
(Algorithm Bidiag2ATS2)

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

1: Choose arbitrary but fixed $n \times m$ matrix W of rank m.
2: Compute the factorization of the matrix W in the form (26).
3: Solve the matrix equation

$$\tilde{B}\tilde{V}^T A Q_m \tilde{U} X = \tilde{B}\tilde{V}^T$$

with respect to unknown matrix X.
4: Compute the output

$$A^{(2)}_{\mathcal{R}(Q_m),\mathcal{N}(V^T)} = Q_m\tilde{U}X.$$

5. Numerical Experience

Example 1 In this illustrative example we consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 4 & 6 & 2 \\ 2 & 3 & 4 & 5 & 3 \\ 3 & 4 & 5 & 6 & 4 \\ 4 & 5 & 6 & 7 & 6 \\ 6 & 6 & 7 & 7 & 8 \end{bmatrix}$$

of rank 4 and choose the matrix

$$W = \begin{bmatrix} 13 & 1 & 0 & 0 & 39 & 0 \\ 17 & 3 & 0 & 0 & 51 & 0 \\ 21 & 4 & 0 & 0 & 63 & 0 \\ 25 & 6 & 0 & 0 & 75 & 0 \\ 19 & 2 & 0 & 0 & 57 & 0 \end{bmatrix}$$

of rank 2. The QR decomposition of W is defined by

$$\{Q,R,P\} = \left\{ \begin{bmatrix} 0.299425 & -0.532976 \\ 0.391555 & -0.0122351 \\ 0.483686 & 0.103891 \\ 0.575817 & 0.624632 \\ 0.437621 & -0.561096 \end{bmatrix}, \begin{bmatrix} 43.4166 & 7.73898 & 0. & 0. & 130.25 & 0. \\ 0. & 2.47148 & 0. & 0. & 7.105\times 10^{-15} & 0. \end{bmatrix}, P = I_6 \right\}.$$

Truncated SVD of the order 2 of the matrix R is defined by the ordered triple

$$\{U,S,V\} = \left\{ \begin{bmatrix} -0.999999 & 0.00101179 \\ -0.00101179 & -0.999999 \end{bmatrix}, \begin{bmatrix} 137.513 & 0. \\ 0. & 2.46756 \end{bmatrix}, \begin{bmatrix} -0.315726 & 0.0178024 \\ -0.0562962 & -0.998414 \\ 0. & 0. \\ 0. & 0. \\ -0.947179 & 0.0534072 \\ 0. & 0. \end{bmatrix} \right\}$$

Outer generalized inverse defined in (18) is equal to

$$A^{(2)}_{\mathcal{R}(Q_W),\mathcal{N}(V_t^*P^*)} = \begin{bmatrix} 0.0453361 & -0.215651 & 0. & 0. & 0.136008 & 0. \\ 0.00990099 & -0.0049505 & 0. & 0. & 0.029703 & 0. \\ 0.00364773 & 0.0420358 & 0. & 0. & 0.0109432 & 0. \\ -0.0317874 & 0.252736 & 0. & 0. & -0.0953622 & 0. \\ 0.0505472 & -0.227028 & 0. & 0. & 0.151641 & 0. \end{bmatrix}.$$

In the continuation of this section we will give a series of examples in which we study behavior of introduced representations.

Example 2 In this example we compare algorithms for computing outer inverses based on three different factorizations:

- Algorithm QRATS2 based on the *QR* decomposition from (Stanimirović et al., 2012b);

- Algorithm QRSVDATS2 based on the representation (18) and

- Algorithm SVDATS2 based on the *SVD* from (Shaini & Hoxha, 2013).

The following code in the programming package *Mathematica* is applied:

```
(* n=100 or n=200 or n=300 or n=400 or n=500 or n=600 or n=700 or n=800 *)
Clear[S];
S[n_] := Module[{i, j, mat = Table[a, {i, n}, {j, n}]},
      For[i = 1, i <= n, i++,
         If[OddQ[i], mat[[i, i]] = a + 1, mat[[i, i]] = a - 1 ];
       ];
      mat[[1, n]] = mat[[n, 1]] = a + 1;
           Return[mat];
  ]

a=1;

LQR = {}; LQRSVD = {}; LSVD = {};
TQR = {}; TQRSVD = {}; TSVD = {};

For[k = 1, k <= 50, k++,
 F = Table[RandomReal[], {i, n}, {j, 2}];     G = Table[RandomReal[], {i, 2}, {j, n}];
 W = F.G;
 s = MatrixRank[W];

 (* Algorithm QRATS2 *)
 {Q, R, P} = QRDecomposition[W, Pivoting -> True] // N;
 Q = Transpose[Q];
 Q1 = Transpose[Take[Transpose[Q], s]];   R1 = Take[R, s];
 XQR = Timing[Q1.Inverse[R1.Transpose[P].A.Q1].R1.Transpose[P]];

 (* Algorithm QRSVDATS2 *)
 {U, S, V} = SingularValueDecomposition[R, s] // N;
 XQRSVD = Timing[Q.U.Inverse[S.Transpose[V].Transpose[P].A.Q.U].S.Transpose[V].Transpose[P]];

 (* Algorithm SVDATS2 *)
 {U, S, V} = SingularValueDecomposition[W, s] // N;
 XSVD = Timing[U.Inverse[S.Transpose[V].A.U].S.Transpose[V]];

 qr = Norm[XQR[[2]].A.XQR[[2]] - XQR[[2]]];
 qrsvd = Norm[XQRSVD[[2]].A.XQRSVD[[2]] - XQRSVD[[2]]];
 svd = Norm[XSVD[[2]].A.XSVD[[2]] - XSVD[[2]]];

 AppendTo[LQR, qr];   AppendTo[LQRSVD, qrsvd];   AppendTo[LSVD, svd];
 AppendTo[TQR, XQR[[1]]];  AppendTo[TQRSVD, XQRSVD[[1]]];  AppendTo[TSVD, XSVD[[1]]];
 ]
```

The smallest values in the tables are written in bold font.

In Table 1 and Table 2 we arrange the results obtained applying the above *Mathematica* code on the test function $S[n]$ from (Zielke, 1986), in the case $a = 1$. Dimensions n of test matrices are equal to $n = 100, 200, 300, 400, 500,$

600, 700, 800. For each matrix $S[n]$ we generate 50 random matrices W of rank 2. Two criteria are used for testing the algorithms: precision and CPU time. Let us denote by $X_k = QRATS2(S[n])$, $Y_k = QRSVDATS2(S[n])$, $Y_k = SVDATS2(S[n])$ the results generated applying QRD, the combination of QRD and SVD of R and $TSVD$ for kth generated matrix W_k, $k = 1, \ldots, 50$. The columns 2,3,4 in Table 1 denote the sum of generated norms:

$$SQR_n = \sum_{k=1}^{50} \|X_k A X_k - X_k\|, SQRSVD_n = \sum_{k=1}^{50} \|Y_k A Y_k - Y_k\|, SSVD_n = \sum_{k=1}^{50} \|Z_k A Z_k - Z_k\|.$$

Table 1. Comparison of three algorithms for computing outer inverses

n	SQR_n	$SQRSVD_n$	$SSVD_n$
100	$3.043527549586 \times 10^{-11}$	$\mathbf{1.454904489449 \times 10^{-11}}$	$1.799685579904 \times 10^{-11}$
200	$6.42517329481 \times 10^{-11}$	$4.418754168002 \times 10^{-11}$	$\mathbf{4.205730328071 \times 10^{-11}}$
300	$1.812027062984 \times 10^{-10}$	$5.292579107393 \times 10^{-10}$	$1.039752882034 \times 10^{-10}$
400	$2.713609669991 \times 10^{-9}$	$\mathbf{5.292579107393 \times 10^{-10}}$	$5.956789277588 \times 10^{-10}$
500	$2.995818463930 \times 10^{-9}$	$1.314409320131 \times 10^{-9}$	$\mathbf{1.163914125347 \times 10^{-9}}$
600	$1.434233650545 \times 10^{-9}$	$3.91646747789 \times 10^{-10}$	$\mathbf{3.902297496071 \times 10^{-10}}$
700	$1.258711792555 \times 10^{-9}$	$5.1970729464 \times 10^{-10}$	$\mathbf{4.57213168965 \times 10^{-10}}$
800	$3.92444897696 \times 10^{-9}$	$\mathbf{2.728576549705 \times 10^{-9}}$	$4.672371872085 \times 10^{-9}$

Performances of compared methods are tested by using the so-called performance profile, introduced in (Dolan & Moore, 2002). The underlying performance metric is defined by the accuracy and the CPU time. Following the notations given in the paper (Dolan & Moore, 2002), the number of solvers is $n_s = 3$ (the solvers are $SQR_n, SQRSVD_n, SSVD_n$) and the number of numerical experiments is $n_p = 8$ (each exploiting 50 randomly matrices W and accordingly generated outer inverses). By $i_{p,s}$ we denote the achieved accuracy obtained by applying the method s on the problem p. The quantity

$$r_{p,s} = \frac{i_{p,s}}{\min\{i_{p,s} : s \in \{SQR_n, SQRSVD_n, SSVD_n\}\}}$$

is called the performance ratio. Finally, the performance of the solver s is defined by the following cumulative distribution function

$$\rho_s(\tau) = \frac{1}{n_p} \text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}, \ s \in \{SQR_n, SQRSVD_n, SSVD_n\}$$

where $\tau \in \mathbb{R}$ and $\mathcal{P}$ represents the set of problems.

Figure 1 shows the performance profiles for $SQR_n, SQRSVD_n, SSVD_n$ regarding the accuracy, using data arranged in Table 1.

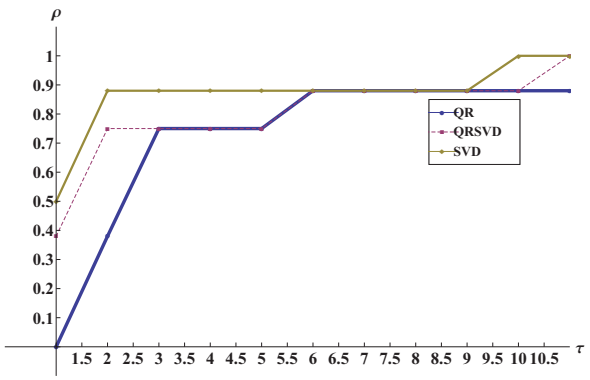

Figure 1. Performance profile regarding the accuracy presented in Table 1

It is observable from Figure 1 that $SSVD_n$ and $SQRSVD_n$ methods show better performances compared to SQR_n:

$$\rho_{SSVD_n}(\tau) \geq \rho_{SQRSVD_n}(\tau) \geq \rho_{SQR_n}(\tau), \ 0 \leq \tau \leq 1.$$

This means that Algorithm $SVDATS2$ has the highest probability of being the optimal solver with respect to the numerical accuracy. Also, Algorithm $QRSVDATS2$ is better than Algorithm $QRATS2$.

Let us denote by $TQR_k, TQRSVD_k, TSVD_k$ CPU times spanned by applying QRD, combination of QRD and SVD of R and $TSVD$ for a selected n and kth generated matrix W_k, $k = 1, \ldots, 50$. The columns 2, 3, 4 in Table 2 denote the sums of CPU times

$$STQR_n = \sum_{k=1}^{50} TQR_k,\ STQRSVD_n = \sum_{k=1}^{50} TQRSVD_k,\ STSVD_n = \sum_{k=1}^{50} XTSVD_k.$$

Table 2. CPU time of three algorithms for computing outer inverses

n	$STQR_n$	$STQRSVD_n$	$STSVD_n$
100	0.0312002	**0.0000001**	**0.0000001**
200	0.0468003	**0.0000001**	**0.0000001**
300	0.0624004	**0.0156001**	0.0312002
400	0.1248008	0.0624004	**0.0312002**
500	0.202801	0.1716011	**0.0936006**
600	0.3432022	0.2028013	**0.1936006**
700	0.3588023	0.3588023	**0.2340015**
800	0.5304034	0.5460035	**0.312002**

Figure 2 illustrates data arranged in Table 2 and shows the performance profiles for the methods $QRATS2$, $QRSVDATS2, SVDATS2$ regarding the spanned CPU time.

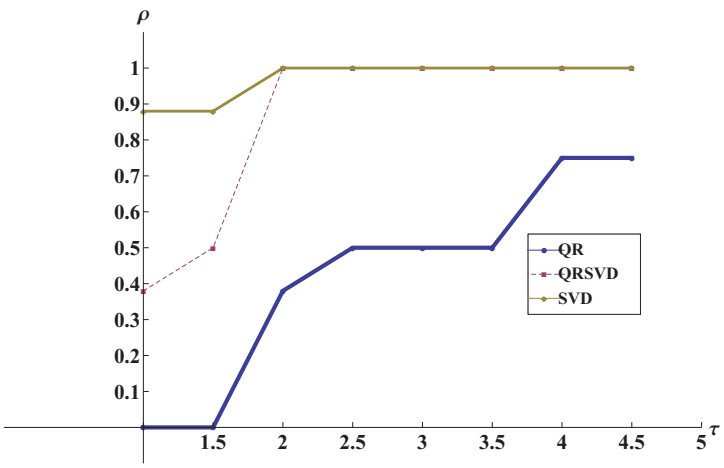

Figure 2. Performance profile regarding the CPU time presented in Table 2

Figure 2 leads to the same conclusion as Figure 1:

$$\rho_{STSVD_n}(\tau) \geq \rho_{STQRSVD_n}(\tau) \geq \rho_{STQR_n}(\tau),\ 0 \leq \tau \leq 1.$$

Example 3 In this example we compare the same algorithms as in Example 2 on the numerical computation of the Moore-Penrose inverse. Instead of randomly generated matrices we use the matrices `W=Transpose[A]=Transpose[S[n]]`, $n = 10, 30, 50, 70, 90, 110, 130, 150, 170, 190$.

The results are arranged in Table 3. Let us denote by

$$X_n = QRATS2(S[n]),\ Y_n = QRSVDATS2(S[n]),\ Z_n = SVDATS2(S[n])$$

the results generated applying QRD, the combination of QRD and SVD of R and $TSVD$, respectively. The columns 2, 3, 4 in Table 3 contain the generated matrix norms

$$QR_n = \|X_nAX_n - X_n\|,\ QRSVD_n = \|Y_nAY_n - Y_n\|,\ SVD_n = \|Z_nAZ_n - Z_n\|.$$

Table 3. Accuracy of three algorithms for computing the Moore-Penrose inverse of $A = S[n]$

n	QR_n	$QRSVD_n$	SVD_n
10	$1.26979128026215 \times 10^{-14}$	$\mathbf{2.034260922393099 \times 10^{-15}}$	$2.08463132435095 \times 10^{-15}$
30	$4.552256280073264 \times 10^{-13}$	$\mathbf{1.400571444654255 \times 10^{-14}}$	$1.69407576911329 \times 10^{-14}$
50	$3.211712544885929 \times 10^{-12}$	$\mathbf{3.948851248647643 \times 10^{-14}}$	$8.49949726133356 \times 10^{-14}$
70	$2.074587304154639 \times 10^{-11}$	$1.405943038980567 \times 10^{-13}$	$\mathbf{9.335752912707724 \times 10^{-14}}$
90	$7.562701917834577 \times 10^{-11}$	$\mathbf{1.848439361484347 \times 10^{-13}}$	$2.52063399770323 \times 10^{-13}$
110	$1.611344139340718 \times 10^{-10}$	$\mathbf{2.995396809335819 \times 10^{-13}}$	$4.933328358230868 \times 10^{-13}$
130	$3.10808433007285 \times 10^{-10}$	$\mathbf{5.76950819729177 \times 10^{-13}}$	$8.547396331319857 \times 10^{-13}$
150	$9.706992066606753 \times 10^{-10}$	$\mathbf{6.703513511452163 \times 10^{-13}}$	$1.01164533157711 \times 10^{-12}$
170	$1.537548362472706 \times 10^{-9}$	$\mathbf{9.427111973617442 \times 10^{-13}}$	$1.39830112820996 \times 10^{-12}$
190	$1.763047487102721 \times 10^{-9}$	$\mathbf{1.368421169609527 \times 10^{-12}}$	$2.423576221191562 \times 10^{-12}$

From Table 3 it is possible to conclude the following:

1) The method $QRATS2$ is the worst solver.

2) The method $QRSVDATS2$ reaches the best numerical precision. The only exception is observed in the case $n = 70$.

Example 4 In this example we continue comparison of the same algorithms for numerical computation of outer inverse. We again use the matrix $A = S[n]$ and randomly generated matrices W of rank $\text{rank}(W) = n/2$. These matrices can be generated by the next *Mathematica* code:

```
F = Table[RandomReal[], {i, n}, {j, n/2}];
G = Table[RandomReal[], {i, n/2}, {j, n}];
W = F.G;
```

The results are arranged in Table 4. Dimensions n are equal to $n = 10, 30, 50, 70, 90, 110, 130, 150, 170, 190$. Let us denote by

$$X_n = QRATS2(S[n]),\ Y_n = QRSVDATS2(S[n]),\ Z_n = SVDATS2(S[n])$$

the results generated applying QRD, combination of QRD and SVD of R and $TSVD$. The columns 2, 3, 4 in Table 3 contain values of the matrix norms

$$QR_n = \|X_nAX_n - X_n\|,\ QRSVD_n = \|Y_nAY_n - Y_n\|,\ SVD_n = \|Z_nAZ_n - Z_n\|.$$

Table 4. Accuracy of three algorithms for computing outer inverses of the matrix $A = S[n]$

n	QR_n	$QRSVD_n$	SVD_n
100	$6.482337614683532 \times 10^{-11}$	$2.336597321351282 \times 10^{-12}$	$\mathbf{2.25369380715732 \times 10^{-12}}$
200	$1.009070836847863 \times 10^{-10}$	$\mathbf{4.835373145662078 \times 10^{-12}}$	$9.532052089466464 \times 10^{-12}$
300	$1.431377648899222 \times 10^{-9}$	$\mathbf{7.52993615694554 \times 10^{-11}}$	$2.239950472891654 \times 10^{-10}$
400	$8.858839542548144 \times 10^{-9}$	$\mathbf{2.172761850286787 \times 10^{-10}}$	$2.525893041322895 \times 10^{-10}$
500	$6.042211807189602 \times 10^{-9}$	$3.297464621913326 \times 10^{-10}$	$\mathbf{3.281406609564898 \times 10^{-10}}$
600	$2.880744866586528 \times 10^{-7}$	$1.292875855137494 \times 10^{-8}$	$\mathbf{1.125846642264049 \times 10^{-8}}$
700	$2.289567222443336 \times 10^{-6}$	$\mathbf{1.342219034202508 \times 10^{-8}}$	$1.797521293862002 \times 10^{-8}$
800	$4.740735235660559 \times 10^{-8}$	$\mathbf{1.537611159244433 \times 10^{-9}}$	$1.771785004312878 \times 10^{-9}$
900	$1.446198521363777 \times 10^{-8}$	$\mathbf{5.4865896018928 \times 10^{-10}}$	$6.560027712609556 \times 10^{-10}$
1000	$1.433314506856117 \times 10^{-8}$	$1.10417479414053 \times 10^{-10}$	$\mathbf{7.640356337951447 \times 10^{-11}}$
2000	$1.156190659420065 \times 10^{-7}$	$\mathbf{9.368815439069883 \times 10^{-10}}$	$1.112831852598413 \times 10^{-9}$
3000	$3.630671988979119 \times 10^{-7}$	$\mathbf{7.875340741156123 \times 10^{-10}}$	$1.995284420020749 \times 10^{-9}$
4000	0.00001541881512652355	$3.562343444447241 \times 10^{-8}$	$\mathbf{3.05838545101564 \times 10^{-8}}$
5000	$4.028422642321919 \times 10^{-6}$	$3.980279026633443 \times 10^{-8}$	$\mathbf{1.396301973029279 \times 10^{-8}}$

Comparing the results included in Table 4, it is possible to conclude the following:

1) The method $QRATS2$ is again the worst solver regarding the numerical precision.

2) The methods $QRSVDATS2$ and $SVDATS2$ reach much better numerical precision.

Example 5 Finally, the same algorithms are compared in numerical computation of outer inverses of the Lauchli matrix. The Lauchli matrix is a $(n+1)\times n$ matrix defined by (see Higham, 1988)

$$L(n,\mu)=\begin{bmatrix} 1 & 1 & \dots & 1 \\ \mu & & & \\ & \mu & & \\ & & \ddots & \\ & & & \mu \end{bmatrix}.$$

In this example, the matrix A is equal to $A = L(n, 0.2)$. We again use randomly generated matrices W satisfying $\text{rank}(W) = n/2$. The results are arranged in Table 5.

Table 5. Accuracy of three algorithms for computing the outer inverses of the Lauchli matrix

n	QR_n	$QRSVD_n$	SVD_n
100	$\mathbf{1.028604544233837\times 10^{-11}}$	$1.425460861122351\times 10^{-11}$	$1.142795986871279\times 10^{-11}$
200	$\mathbf{7.100072558578722\times 10^{-12}}$	$1.824218048288609\times 10^{-11}$	$1.689039220659226\times 10^{-11}$
300	$3.931602647539584\times 10^{-10}$	$\mathbf{1.522893445477722\times 10^{-10}}$	$3.067952338830401\times 10^{-10}$
400	$5.172628286663662\times 10^{-11}$	$\mathbf{3.515510616280827\times 10^{-11}}$	$4.156475258052759\times 10^{-11}$
500	$4.645784213245661\times 10^{-10}$	$\mathbf{1.315563869323691\times 10^{-10}}$	$3.028534791228661\times 10^{-10}$
600	$1.689506089025813\times 10^{-10}$	$\mathbf{1.018132753134374\times 10^{-10}}$	$1.172500764749819\times 10^{-10}$
700	$1.134125813525221\times 10^{-9}$	$5.572822660688449\times 10^{-10}$	$\mathbf{2.571534358928769\times 10^{-9}}$
800	$5.647648157231927\times 10^{-10}$	$\mathbf{1.945489973975069\times 10^{-10}}$	$2.76608506543413\times 10^{-10}$
900	$4.477591961216335\times 10^{-10}$	$4.434485330379887\times 10^{-10}$	$\mathbf{2.129986321753629\times 10^{-10}}$
1000	$5.175937950315212\times 10^{-10}$	$\mathbf{4.066683452807805\times 10^{-10}}$	$5.50032373267487\times 10^{-10}$
2000	$1.019238815039415\times 10^{-9}$	$7.941173235725879\times 10^{-10}$	$\mathbf{7.09726704984236\times 10^{-10}}$
3000	$1.803956394814176\times 10^{-8}$	$5.854369183636487\times 10^{-9}$	$\mathbf{3.522502016638002\times 10^{-9}}$
4000	$4.642390180619101\times 10^{-9}$	$4.484612755803374\times 10^{-9}$	$\mathbf{3.34756148108135\times 10^{-9}}$
5000	$2.528918230390071\times 10^{-6}$	$4.265161501638872\times 10^{-6}$	$\mathbf{6.238001463545644\times 10^{-7}}$

Comparing the results included in Table 5, it is possible to conclude the following.

1) The method $QRATS2$ reaches the minimal precision two times, in the first two cases.

2) The methods $QRSVDATS2$ and $SVDATS2$ generate the results with much better numerical precision. Both of them reach the minimal precision six times.

3) $QRATS2$ is the best solver for small dimensions n, $QRSVDATS2$ for medium values and $SVDATS2$ for greatest dimensions.

Example 6 The matrix $A = L(n, 0.000002)$ is ill–conditioned and causes serious numerical problems in computation of required outer inverses. The results are obtained using randomly generated matrices W of rank $\text{rank}(W) = n/2$ and arranged in Table 6.

Table 6. Accuracy of three algorithms for computing the outer inverses of the ill-conditioned Lauchli matrix

n	QR_n	$QRSVD_n$	SVD_n
100	0.395788193577041	0.4493449896897091	**0.1430739500991127**
200	0.1101693065477999	**0.09052408752219679**	0.1517284556203095
300	**0.5757304019521979**	0.9945688854731312	0.7355659072586593
400	0.5008982407371158	**0.2071745759326693**	0.473611267749282
500	0.8474063213377292	0.6275335481487543	**0.3640134520309316**
600	0.3563584901595677	0.3442714329338474	**0.1503166152086566**
700	6.652359033095027	2.89843803437463	**2.299765261435784**
800	**19.43181229016723**	32.86722282380996	37.35266633821223
900	5.962766800197418	**3.555777641729863**	8.543932380583575
1000	14.55693372724122	**3.973128971933598**	5.542061014703714
2000	22.84253966856699	**13.65738063338715**	41.93288432714076
3000	5.6061607170558	62.15805382380937	**21.85354632399693**
4000	**210.4520364390122**	613.7966082641098	721.2117782329316
5000	50.09409738330062	**29.81137054768079**	31.69036850919142

The results included in Table 6 show that $QRSVDATS2$ is the best solver six times, $SVDATS2$ five times and $QRATS2$ reaches the best values three times.

6. Conclusion

According to performed numerical experiments we conclude the following.

1) The SVD representation of outer inverses with prescribed range and null space is the most efficient in the case of low rank matrices W.

2) The combination of QR and SVD decomposition of outer inverses with prescribed range and null space is the most efficient in the case $W = A^T$.

3) The combination of QR decomposition and SVD is a significant improvement of the QR decomposition method from (Stanimirović et al., 2012b).

Also, in the present paper we show that the factorizations based on a bidiagonal form are useful in representation of outer inverses of matrices with full column rank.

References

Ben-Israel, A., & Greville, T. N. E. (2003). *Generalized inverses: Theory and Applications* (2nd ed.). Springer.

Chen, Y. (1990). The generalized Bott-Duffin inverse and its application. *Linear Algebra Appl., 134*, 71-91. http://dx.doi.org/10.1016/0024-3795(90)90007-Y

De Moor, B. (1991). Generalizations of the singular value and QR decompositions. *Signal Processing, 25*, 135-146.

De Moor, B., & Dooren, P. V. (1992). Generalizations of the singular value and QR decompositions. *SIAM J. Matrix Anal. Appl., 13*, 993-1014.

Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math. Program., 91*, 201-213. http://dx.doi.org/10.1007/s101070100263

Getson, A. J., & Hsuan, F. C. (1988). {2}-Inverses and their Statistical Applications. *Lecture Notes in Statistics, 47*. Berlin: Springer.

Goodall, C. R. (1993). Computation Using the QR Decomposition. In C. R. Rao (Ed.), *Handbook of Statistics* (Vol. 9, pp. 467-508).

Higham, N. J. (n.d.). *The Matrix Computation Toolbox*. Retrieved from http://www.ma.man.ac.uk/ higham/mctoolbox

Husen, F., Langenberg, P., & Getson, A. (1985). The {2}-inverse with applications to statistics. *Linear Algebra Appl., 70*, 241-248. http://dx.doi.org/10.1016/0024-3795(85)90055-2

Issa, A. S., & Bdair, O. M. (2014). Some Bounds of the Zeros of Polynomials Based on the QR-Decomposition

and the LU-Decomposition of the Frobenius Companion Matrix. *Journal of Mathematics Research, 6* 33-41. http://dx.doi.org/10.5539/jmr.v6n1p33

Katsikis, V. N., Pappas, D., & Petralias, A. (2011). An improved method for the computation of the Moore-Penrose inverse matrix. *Appl. Math. Comput., 217*, 9828-9834. http://dx.doi.org/10.1016/j.amc.2011.04.080

Malik, S. B., & Thome, N. (2014). On a new generalized inverse for matrices of an arbitrary index. *Appl. Math. Comput., 226*, 575-580. http://dx.doi.org/10.1016/j.amc.2013.10.060

Nashed, M. Z. (1976). *Generalized Inverse and Applications*. New York: Academic Press.

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

Ralha, R. (2003). One-sided reduction to bidiagonal form. *Linear Algebra Appl., 358*, 219-238. http://dx.doi.org/10.1016/S0024-3795

Shaini, B. I., & Hoxha, F. (2013). Computing generalized inverses using matrix factorizations. *Facta Universitatis (Niš), Ser. Math. Inform., 28*, 335-351.

Sheng, X., & Chen, G. (2007). Full-rank representation of generalized inverse $A^{(2)}_{T,S}$ and its applications. *Comput. Math. Appl., 54*, 1422-1430. http://dx.doi.org/10.1016/j.camwa.2007.05.011

Smoktunowicz, A., & Wróbel, I. (2012). Numerical aspects of computing the Moore-Penrose inverse of full column rank matrices. *BIT Numer. Math., 52*, 503-524. http://dx.doi.org/10.1007/s10543-011-0362-0

Stanimirović, P. S., Pappas, D., Katsikis, V. N., & Stanimirović, I. P. (2012a). Computation of $A^{(2)}_{T,S}$-inverses using QDR factorization. *Linear Algebra Appl., 437* 1317-1331. http://dx.doi.org/10.1016/j.laa.2012.04.026

Stanimirović, P. S., Pappas, D., Katsikis, V. N., & Stanimirović, I. P. (2012b). Full-rank representations of outer inverses based on the QR decomposition. *Appl. Math. Comput., 218*, 10321-10333. http://dx.doi.org/10.1016/j.amc.2012.04.011

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

Watkins, D. (2002). *Fundamentals of Matrix Computations*. New York: Wiley-Interscience.

Zheng, B., & Bapat, R. B. (2004). Generalized inverse $A^{(2)}_{T,S}$ and a rank equation. *Appl. Math. Comput., 155*(2), 407-415. http://dx.doi.org/10.1016/S0096-3003(03)00786-0

Zielke, G. (1986). Report on test matrices for generalized inverses. *Computing, 36*, 105-162. http://dx.doi.org/10.1007/BF02238196