Czechoslovak Mathematical Journal

Yunkun Chen; Xinghua Shi; Yi Min Wei
Convergence of Rump's method for computing the Moore-Penrose inverse

Czechoslovak Mathematical Journal, Vol. 66 (2016), No. 3, 859–879

Persistent URL: http://dml.cz/dmlcz/145876

Terms of use:

© Institute of Mathematics AS CR, 2016

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.

This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* http://dml.cz

Czechoslovak Mathematical Journal, 66 (141) (2016), 859–879

CONVERGENCE OF RUMP'S METHOD FOR COMPUTING THE MOORE-PENROSE INVERSE

Yunkun Chen, Xiamen, Xinghua Shi, Yimin Wei, Shanghai

(Received December 30, 2015)

In memory of Professor Miroslav Fiedler

Abstract. We extend Rump's verified method (S. Oishi, K. Tanabe, T. Ogita, S. M. Rump (2007)) for computing the inverse of extremely ill-conditioned square matrices to computing the Moore-Penrose inverse of extremely ill-conditioned rectangular matrices with full column (row) rank. We establish the convergence of our numerical verified method for computing the Moore-Penrose inverse. We also discuss the rank-deficient case and test some ill-conditioned examples. We provide our Matlab codes for computing the Moore-Penrose inverse.

Keywords: Moore-Penrose inverse; condition number; ill-conditioned matrix

MSC 2010: 65F05, 15A24

1. Introduction

The Moore-Penrose inverse is a useful tool in parallel sums with applications to electrical networks [1], [3], computing polar decompositions [8], the Tikhonov regularization and ill-posed problems [13], [18], [21], [31], [40], [41], [42], [43], the linear programming [3], the linear statistics model [27], the linear least squares problems [7], [10], [11], [12], [22], [37], [38], [39], [44] and the total least squares problems [45].

Now we provide preliminaries for the Moore-Penrose inverse and the singular value decomposition (SVD).

The research has been supported by the National Natural Science Foundation of China under grant 11271084.

Definition 1.1 ([1], [27], [38]). Let $A \in \mathbb{R}^{m\times n}$. The Moore-Penrose inverse $X = A^\dagger \in \mathbb{R}^{n\times m}$ is uniquely determined by the four matrix equations,

$$AXA = A, \quad XAX = X, \quad (AX)^{\mathrm{T}} = AX, \quad (XA)^{\mathrm{T}} = XA.$$

Lemma 1.2 ([2], [17] Singular value decomposition). *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with* $\operatorname{rank} r$. *Then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that*

$$A = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^{\mathrm{T}}, \quad \Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r), \tag{1.1}$$

where $U^{\mathrm{T}}U = I_m$, $V^{\mathrm{T}}V = I_n$ and $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_r > 0$ are called the singular values of A. The Moore-Penrose inverse of A can be expressed by

$$A^\dagger = V \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^{\mathrm{T}}.$$

Lemma 1.3 ([1], [3]). *If* $\operatorname{rank}(A) = m$ *(i.e., full row rank), then $A^\dagger = A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}$ and $AA^\dagger = I_m$; if* $\operatorname{rank}(A) = n$ *(i.e., full column rank), then $A^\dagger = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$ and $A^\dagger A = I_n$, where I_n is the identity matrix of order n.*

A number of numerical and symbolic algorithms, see [5], [6], [19], [20], [36] for computing the Moore-Penrose inverse of the structured and block matrices have been presented. Rump et al. develop the numerically verified methods for the matrix inversion, see [24], [26], [29], [33], the matrix equations in [25], the linear least squares problem and the under-determined linear system in [28]. The origin of Rump's method dates back to 1984. Rump did not publish it due to lack of analysis. The report in [34] gives only some computational results. Rump analysed his original algorithm in [29]. A modifed version was analysed by Oishi et al. in [26]. However, there is a significant change of the original method, namely that a perturbation of the size $\sqrt{\mathbf{u}}$ is introduced, rather than no perturbation or a perturbation of size $\mathbf{u}$ as in the original paper [29], where $\mathbf{u}$ denotes the machine precision.

An outline of this paper is organized as follows. We present our numerically verified method for computing the Moore-Penrose inverse in Section 2. The convergence of the verified algorithm is proved in Section 3. We also discuss the rank-deficient case in Section 4 and present some ill-conditioned examples in Section 5. We provide our Matlab codes for computing the Moore-Penrose inverse and detailed proofs of Theorems 3.1 and 3.2 in Appendix.

2. Verified algorithm

First, we introduce an accurate dot product calculation algorithm, see [23]. Let $A \in \mathbb{F}^{m\times n}$ and $B \in \mathbb{F}^{n\times l}$, where $\mathbb{F}$ is the set of double precision floating point numbers defined by IEEE 754 standard. Suppose that we have an accurate dot product algorithm with $G_i \in \mathbb{F}^{m\times l}$ $(i = 1, 2, \ldots, k)$ satisfying

$$\left|\sum_{i=1}^{k} G_i - AB\right| \leqslant \mathbf{C}_0\mathbf{u}^k|AB|, \tag{2.1}$$

where $\mathbf{u} = 2^{-53} \approx 1.1 \times 10^{-16}$ and $\mathbf{C}_0 = \mathcal{O}(1)$. We denote such an algorithm as

$$G_k = fl_{k,k}(AB) \quad \text{with } G_k := \sum_{i=1}^{k} G_i,\ G_i \in \mathbb{F}^{m\times l}.$$

If the product is executed in k-fold precision and stored in working precision in [26], then we can write $G = fl_{k,1}(A \cdot B)$.

Next we introduce the original algorithm for inverting arbitrarily ill-conditioned matrices which was developed by Oishi, Tanabe, Ogita, and Rump in [26]. The main idea of Rump's algorithm for inverting an extremely ill-conditioned matrix is that although inverting an arbitrarily ill-conditioned matrix in single or double precision does not produce meaningless numbers, it contains a lot of information, which could be used as preconditioners to compute the inversion of matrices. Here we apply simplified notation introduced above instead of the version of [26], Algorithm 1.

Algorithm 2.1 ([26], Algorithm 1)**.** Modified Rump's method I for inverting an extremely ill-conditioned matrix

$S_0 = A + \Delta A$;	% perturbation for A
$X_0 = \mathbf{inv}(S_0); R_1 = X_0$	
For $k = 1, 2, \ldots,$ until convergence	
$\quad S_k = fl_{k,1}(A \cdot R_k)$	% stored in working precision
$\quad \widetilde{S_k} = S_k + \Delta S_k$	% perturbation for S_k
$\quad X_k = \mathbf{inv}(\widetilde{S_k})$	% floating-point Inverse
$\quad R_{k+1} = fl_{k+1,k+1}(X_k \cdot R_k)$	% stored in $k+1$-fold precision
end	

Here '$\mathbf{inv}(B)$' is a built-in function in Matlab for inversion of B, $(\Delta S_k)_{ij} = r_{ij}\sqrt{\mathbf{u}}(|S_k|)_{ij}$ for an (i,j)-element of ΔS_k. Here we write $\kappa_\infty(A) = \|A\|_\infty\|A^{-1}\|_\infty$ for the ∞-norm. If $\kappa_\infty(S_k) > \mathbf{u}^{-1}$, then we choose r_{ij} as pseudo-random numbers distributed uniformly in $[-1, 1]$; otherwise, we select $r_{ij} = 0$. We store the result in the cell of matrices R_{k+1}. The algorithm converges if $\|R_k X_k - I\|_\infty \leqslant \varepsilon$. In [26], the authors present the proof that if the algorithm is convergent, then R_k converges to the inversion of the matrix A, under some reasonable assumptions.

Now we aim to compute the Moore-Penrose inversion of a full row rank extremely ill-conditioned matrix. Suppose that the matrix R is the approximation of $A^\dagger$; it is obvious that even if $\|AR - I\|_\infty \leqslant \varepsilon$, we could not guarantee that $\|R - A^\dagger\|_\infty$ is small enough. We need some more assumptions and obtain a new convergence result.

Theorem 2.2. *Let* $\|A(A+\Delta A)^\dagger - I_m\| < 1$ *and* $A \in \mathbb{R}^{m\times n}$ *with* $\operatorname{rank}(A) = m$. *Suppose that for the range space,* $\mathcal{R}[(\Delta A)^{\mathrm{T}}] \subseteq \mathcal{R}(A^{\mathrm{T}})$ *is satisfied. Then for any consistent norm* $\|\cdot\|$ *we have*

$$\frac{\|(A+\Delta A)^\dagger\|}{1 + \|A(A+\Delta A)^\dagger - I_m\|} \leqslant \|A^\dagger\| \leqslant \frac{\|(A+\Delta A)^\dagger\|}{1 - \|A(A+\Delta A)^\dagger - I_m\|}. \tag{2.2}$$

P r o o f. Denote $(A+\Delta A)^\dagger = R$ it is easy to see that AR is nonsingular due to $\operatorname{rank}(A+\Delta A) = m$, hence

$$\|(AR)^{-1}\| = \|[I_m - (I_m - AR)]^{-1}\| \leqslant \frac{1}{1 - \|I_m - AR\|}.$$

Furthermore,

$$\|R(AR)^{-1}\| \leqslant \|R\|\,\|(AR)^{-1}\| \leqslant \frac{\|R\|}{1 - \|AR - I_m\|}.$$

The key step is proving $R(AR)^{-1} = A^\dagger$. $\mathcal{R}\left[(\Delta A)^{\mathrm{T}}\right] \subseteq \mathcal{R}(A^{\mathrm{T}})$ is equivalent to $\Delta A = MA$ for an $m \times m$ matrix M. Hence $R = [(I_m + M)A]^\dagger$. The assumption $\|AR - I_m\| < 1$ ensures A, R and $I_m + M$ to have full rank. Then $R(AR)^{-1} = A^\dagger$ is directly verified.

It follows from $R = A^\dagger AR$ that

$$\|R\| \leqslant \|A^\dagger\|\,\|AR\| = \|A^\dagger\|\,\|AR - I_m + I_m\| \leqslant \|A^\dagger\|(1 + \|AR - I_m\|).$$

The proof is complete. □

Analogously, we can obtain similar results for the full column rank.

Corollary 2.3. *Let* $\|(A+\Delta A)^\dagger A - I_n\| < 1$ *and* $A \in \mathbb{R}^{m\times n}$ *with* $\operatorname{rank}(A) = n$. *Assume that* $\mathcal{R}(\Delta A) \subseteq \mathcal{R}(A)$, *then*

$$\frac{\|(A+\Delta A)^\dagger\|}{1 + \|(A+\Delta A)^\dagger A - I_n\|} \leqslant \|A^\dagger\| \leqslant \frac{\|(A+\Delta A)^\dagger\|}{1 - \|(A+\Delta A)^\dagger A - I_n\|}.$$

Now we present a verified algorithm to compute the Moore-Penrose inversion of a full row rank extremely ill-conditioned matrix such that the computational inverse R of A by Matlab does not satisfy $\|I - RA\| < 1$. In other words, in double precision the condition is larger that 1e+16, in fact, much larger. We need k-fold precision by Oishi, Tanabe, Ogita, and Rump, see [26].

Algorithm 2.4. Modified Rump's method I for the Moore-Penrose inversion of an extremely ill-conditioned matrix

$X_0 = A^{\mathrm{T}}, R_1 = X_0$;
For $k = 1, 2, \ldots,$ until convergence

$S_k = fl_{k,1}(A \cdot R_k)$	% stored in working precision
$\widetilde{S_k} = S_k + \Delta S_k$	% perturbation for S_k
$X_k = \mathbf{inv}(\widetilde{S_k})$	% floating-point Inverse
$R_{k+1} = fl_{k+1,k+1}(R_k \cdot X_k)$	% stored in k-fold precision

end

The choice of $(\Delta S_k)_{ij}$ is the same as in Algorithm 2.1. Moreover, the assumption $\Delta A = MA$ makes clear that our choice $R_1 = A^{\mathrm{T}}$ is good.

3. Convergence of verified algorithm

In this section we shall prove the convergence of Algorithm 2.5. Suppose that the dimensions of the problem m and n, satisfy $m\sqrt{\mathbf{u}} \ll 1$ and $n\sqrt{\mathbf{u}} \ll 1$. In this paper, we assume that $\mathbf{C}_i$, $i = 0, 1, 2, \ldots$, denote numbers of $\mathcal{O}(1)$ satisfying $\mathbf{C}_i\mathbf{u} \ll 1$ and $\mathbf{C}_i\sqrt{\mathbf{u}} \ll 1$. $\mathbf{c}_m$ is a number of $\mathcal{O}(m)$ satisfying $\mathbf{c}_m\mathbf{u} \ll 1$ and $\mathbf{c}_m\sqrt{\mathbf{u}} \ll 1$.

Denote $S_k := AR_k$; now S_k is an $m \times m$ nonsingular square matrix. We can obtain the following convergence theorems which are similar to [26].

Theorem 3.1. *Suppose that $\kappa_\infty(S_k) \geqslant \mathbf{u}^{-1}$, and some reasonable assumptions (which listed in Appendix) are satisfied. Then $\kappa_\infty(S_{k+1}) \leqslant \mathcal{O}(m)\sqrt{\mathbf{u}}\kappa_\infty(S_k) + \mathcal{O}(1)$.*

Since $\mathcal{O}(m)\sqrt{\mathbf{u}} \ll 1$, $\kappa_\infty(S_k)$ decreases as $\mathcal{O}((m\sqrt{\mathbf{u}})^k)\kappa_\infty(A)$ and finally $\kappa_\infty(S_k)$ becomes $\mathcal{O}(1)$, if k is sufficiently large. With some other assumptions (which are listed in Appendix), we can obtain the following theorem.

Theorem 3.2. *If $\kappa_\infty(S_k) = \mathcal{O}(1)$, then we can deduce that*

$$\|I - S_{k+1}\|_\infty = \mathbf{C}_{10}\sqrt{\mathbf{u}} + \varepsilon' \ll 1,$$

where $\varepsilon' \ll 1$.

We can denote by R_k the Moore-Penrose inverse of the matrix $A + \Delta A_k$, as we know that if k is large enough, then we have

$$\|I - A(A + \Delta A_k)\|_\infty \ll 1, \quad \text{tends to } 0. \tag{3.1}$$

Due to Theorem 2.2 and equation (3.1), we can claim that if k is large enough, then the result R_{k+1} of Algorithm 2.4 converges to the Moore-Penrose inverse $A^\dagger$.

4. Rank-deficient case

In this section, we discuss how to utilize Rump's method to compute the Moore-Penrose inverse for the rank-deficient matrix by the rank-revealing decomposition [4], [9], [14].

Definition 4.1 ([9], [14]). Suppose that $A \in \mathbb{R}^{m\times n}$ and $\operatorname{rank}(A) = r < \min\{m, n\}$. The rank-revealing decomposition of A is $A = XDY$, where $X \in \mathbb{R}^{m\times r}$, $D = \operatorname{diag}(d_1, d_2, \ldots, d_r)$, $Y \in \mathbb{R}^{r\times n}$. X and Y are of full column and row rank matrices, respectively.

It is easy to verify that

$$A^\dagger = Y^\dagger D^{-1} X^\dagger.$$

We first compute the Moore-Penrose inverse of full column (row) rank matrices X and Y, then we can obtain $A^\dagger$.

Lemma 4.2 ([4], [9]). *Let $A = XDY$ be the rank-revealing decomposition of A, $\widehat{X}$, $\widehat{D} = \operatorname{diag}(\widehat{d}_1, \widehat{d}_2, \ldots, \widehat{d}_r)$ and let $\widehat{Y}$ be the factors computed by a certain algorithm. These factors satisfy*

$$\frac{\|\widehat{X} - X\|}{\|X\|} \leqslant p(m,n)\mathbf{u}, \qquad \frac{\|\widehat{Y} - Y\|}{\|Y\|} \leqslant p(m,n)\mathbf{u}, \tag{4.1}$$
$$\text{and} \quad \frac{|\widehat{d}_i - d_i|}{|d_i|} \leqslant p(m,n)\mathbf{u}, \quad i = 1, 2, \ldots, n,$$

where $p(m,n)$ is a modestly growing function of m and n, i.e., a function bounded by a low degree polynomial in m and n, such that $\max\{\kappa_2(X), \kappa_2(Y)\}p(m,n)\mathbf{u} < 1/2$, where the condition numbers are $\kappa_2(X) = \|X\|_2\|X^\dagger\|_2$ and $\kappa_2(Y) = \|Y\|_2\|Y^\dagger\|_2$.

Theorem 4.3. *Suppose that $A \in \mathbb{R}^{m\times n}$, $m \leqslant n$, and $\operatorname{rank}(A) = r < m$. Let $A = XDY$ be the rank-revealing decomposition of A. Let $\widehat{X}$, $\widehat{D} = \operatorname{diag}(\widehat{d}_1, \widehat{d}_2, \ldots, \widehat{d}_r)$ and*

$\widehat{Y}$ *be the computed factors which satisfy* (4.1). *Then we compute* $\widehat{A}^{\dagger} = \widehat{Y}^{\dagger}\widehat{D}^{-1}\widehat{X}^{\dagger}$. *Dropping the second order terms, it follows that*

$$\|\widehat{A}^{\dagger} - A^{\dagger}\|_2 \leqslant [2\kappa_2(X) + 2\kappa_2(Y) + \kappa_2(D)]\frac{\kappa_2(X)\kappa_2(Y)\kappa_2(D)}{\|X\|_2\|Y\|_2\|D\|_2}p(m,n)\mathbf{u}.$$

P r o o f. Let $\widehat{X} = X + \Delta X$, $\widehat{D} = D + \Delta D$, $\widehat{Y} = Y + \Delta Y$. It follows from [12] that dropping the second order terms we obtain,

$$\widehat{X}^{\dagger} = X^{\dagger} - X^{\dagger}(\Delta X)X^{\dagger} + (X^{\mathrm{T}}X)^{-1}(\Delta X)^{\mathrm{T}}(I_m - XX^{\dagger}), \tag{4.2}$$

$$\widehat{Y}^{\dagger} = Y^{\dagger} - Y^{\dagger}(\Delta Y)Y^{\dagger} + (I_n - Y^{\dagger}Y)(\Delta Y)^{\mathrm{T}}(YY^{\mathrm{T}})^{-1}, \tag{4.3}$$

$$\widehat{D}^{-1} = D^{-1} - D^{-1}(\Delta D)D^{-1}. \tag{4.4}$$

Next, applying (4.3) and (4.4), we can estimate the approximation $\|\widehat{A}^{\dagger} - A^{\dagger}\|_2$. Dropping the second and higher order terms, we obtain

$$\begin{aligned}\widehat{A}^{\dagger} - A^{\dagger} = & -Y^{\dagger}D^{-1}[X^{\dagger}(\Delta X)X^{\dagger} - (X^{\mathrm{T}}X)^{-1}(\Delta X)^{\mathrm{T}}(I_m - XX^{\dagger})] \\ & - [Y^{\dagger}(\Delta Y)Y^{\dagger} - (I_n - Y^{\dagger}Y)(\Delta Y)^{\mathrm{T}}(YY^{\mathrm{T}})^{-1}]D^{-1}X^{\dagger} \\ & - Y^{\dagger}D^{-1}(\Delta D)D^{-1}X^{\dagger}.\end{aligned}$$

Since $\kappa_2(X) = \|X\|_2\|X^{\dagger}\|_2$ and due to (4.1), we have

$$\begin{aligned}\|\widehat{A}^{\dagger} - A^{\dagger}\|_2 \leqslant\ & 2\|Y^{\dagger}\|_2\|D^{-1}\|_2\|X^{\dagger}\|_2^2\|\Delta X\|_2 \\ & + 2\|X^{\dagger}|_2\|D^{-1}\|_2\|Y^{\dagger}\|_2^2\|\Delta Y\|_2 + \|X^{\dagger}\|_2\|Y^{\dagger}\|_2\|D^{-1}\|_2^2\|\Delta D\|_2 \\ \leqslant\ & 2\|Y^{\dagger}\|_2\|D^{-1}\|_2\|X^{\dagger}\|_2^2\|X\|_2p(m,n)\mathbf{u} \\ & + 2\|X^{\dagger}\|_2\|D^{-1}\|_2\|Y^{\dagger}\|_2^2\|Y\|_2p(m,n)\mathbf{u} \\ & + \|X^{\dagger}\|_2\|Y^{\dagger}\|_2\|D^{-1}\|_2^2\|D\|_2p(m,n)\mathbf{u} \\ =\ & [2\kappa_2(X) + 2\kappa_2(Y) + \kappa_2(D)]\frac{\kappa_2(X)\kappa_2(Y)\kappa_2(D)}{\|X\|_2\|Y\|_2\|D\|_2}p(m,n)\mathbf{u}.\end{aligned}$$

□

5. Numerical examples

Now we present some extremely ill-conditioned examples for computing the Moore-Penrose inverse.

5.1. Ill-conditioned examples.

Example 5.1 ([32])**.** Let $A \in \mathbb{R}^{3\times 4}$ with an element $0 \neq \varepsilon \in \mathbb{R}$. It is easy to check the determinant $\det(AA^{\mathrm{T}}) = 6\varepsilon^2$ and A is of full row rank. Here we use Matlab function '**pinv**' and Algorithm 2.5 to compute the Moore-Penrose inverse of the matrix A. If ε is close to zero, then we compare the relative error of $\|R - A^{\dagger}\|_{\infty}/\|A^{\dagger}\|_{\infty}$.

Let

$$A = \begin{pmatrix} 0 & -1 & 0 & -1 \\ -1 & 1 & 1 & -1 \\ 0 & 1 & \varepsilon & 1 \end{pmatrix}, \quad \text{and} \quad A^{\dagger} = \frac{1}{6\varepsilon}\begin{pmatrix} 2 & -2\varepsilon & 2 \\ -2-3\varepsilon & 2\varepsilon & -2 \\ 6 & 0 & 6 \\ 2-3\varepsilon & -2\varepsilon & 2 \end{pmatrix}.$$

We present the numerical result. It follows from the results in Table 1 that we can compute the exact value of $A^{\dagger}$ by Algorithm 2.4 after only two iterations.

	$\varepsilon = 2^{0}$	$\varepsilon = 2^{-5}$	$\varepsilon = 2^{-10}$	$\varepsilon = 2^{-20}$
pinv	2.9497e−016	1.0142e−014	3.2135e−013	3.2927e−010
Iteration 1 of Algorithm 2.5	9.1552e−017	2.9233e−015	9.2802e−014	9.5053e−011
Iteration 2 of Algorithm 2.5	0	0	0	0

Table 1. Relative error and number of iteration results.

Example 5.2 ([46], page 153)**.** Let $A \in \mathbb{R}^{5\times 7}$ with $\operatorname{rank}(A) = 5$. The maximum and minimum singular values are $\sigma_{\max} = 5.9161|a|$ and $\sigma_{\min} = 1/\sqrt{2}|a|$, respectively. We can compute $\kappa_2(A) \approx 8.3|a|^2$. Let $a = 1 \times 10^{15}$ so that $\kappa_2(A) \approx 8.3 \times 10^{30}$.

Let

$$A = \begin{pmatrix} a+1 & a+2 & a+2 & a+3 & a+4 & a & a-1 \\ a+2 & a+2 & a+3 & a+4 & a+5 & a+1 & a-1 \\ a+2 & a+3 & a+4 & a+5 & a+6 & a+1 & a-1 \\ a+3 & a+4 & a+5 & a+5 & a+6 & a+2 & a+1 \\ a+4 & a+5 & a+6 & a+6 & a+7 & a+3 & a+2 \end{pmatrix},$$

and

$$A^{\dagger} = \frac{1}{12}\begin{pmatrix} 4 & 16 & -22 & 6a+16 & -6a-8 \\ 8 & -10 & 4 & -10 & 8 \\ -12 & 0 & 0 & 36 & -24 \\ -4 & -10 & 22 & -6a-34 & 6a+20 \\ 8 & 8 & -14 & 6a+8 & -6a-4 \\ -8 & -2 & 14 & -6a-26 & 6a+16 \\ 4 & -2 & -4 & -2 & 4 \end{pmatrix}.$$

The numerical result is listed in Table 2.

k	$\Vert\widetilde{S_k}\Vert_\infty$	$\Vert X_k\Vert_\infty$	$\Vert I_m - \widetilde{S_k}X_k\Vert_\infty$	$\Vert I_n - R_kA\Vert_\infty$	$\frac{\Vert R_k - A^{\dagger}\Vert_\infty}{\Vert A^{\dagger}\Vert_\infty}$
$k=1$	3.5000e+030	3.5527e−015	2.1731e+000	1.9394e+000	1.0000e+000
$k=2$	1.1731e+000	3.8160e+016	5.9898e+000	1.4602e+001	1.0000e+000
$k=3$	6.5488e+001	1.4500e+016	7.4065e+000	5.2841e+001	1.0000e+000
$k=4$	6.4065e+001	6.7598e+014	5.3312e−001	1.4475e+001	3.7971e−001
$k=5$	1.5311e+001	1.2530e+000	7.8873e−017	2.7149e−017	1.1827e−016
$k=6$	1.0000e+000	1.0000e+000	2.8297e−034	6.6126e−017	8.8388e−017

Table 2. Example 5.2.

Example 5.3 ([46], page 150)**.** Let $A \in \mathbb{R}^{6\times 7}$ with $\operatorname{rank}(A) = 6$. The maximum and minimum singular values are $\sigma_{\max} = \sqrt{42}|a|$ and $\sigma_{\min} = \sqrt{2/49}/|a|$, respectively. We can compute $\kappa_2(A) \approx \mathcal{O}(|a|^2)$. In this example, we select $a = 1 \times 10^{15}$.

Let

$$A = \begin{pmatrix} a+5 & a+3 & a+2 & a+4 & a+3 & a+2 & a+1 \\ a+3 & a+4 & a+2 & a+3 & a+3 & a+2 & a \\ a+2 & a+2 & a+2 & a+2 & a+2 & a+1 & a+1 \\ a+4 & a+3 & a+2 & a+3 & a+3 & a+2 & a+1 \\ a+3 & a+3 & a+2 & a+3 & a+2 & a+2 & a+1 \\ a+2 & a+2 & a+1 & a+2 & a+2 & a & a-1 \end{pmatrix},$$

and

$$A^{\dagger} = \frac{1}{4}\begin{pmatrix} -4a-12 & -4a-12 & -4a-8 & 4a+16 & 4a+12 & 4a+8 \\ -3a-9 & -3a-6 & -3a-5 & 3a+9 & 3a+9 & 3a+5 \\ -5a-11 & -5a-10 & -5a-3 & 5a+11 & 5a+11 & 5a+7 \\ 4a+16 & 4a+12 & 4a+8 & -4a-20 & -4a-12 & -4a-8 \\ 4a+12 & 4a+12 & 4a+8 & -4a-12 & -4a-16 & -4a-8 \\ 3a+5 & 3a+6 & 3a+1 & -3a-5 & -3a-5 & -3a-5 \\ a+3 & a+2 & a+3 & -a-3 & -a-3 & -a-3 \end{pmatrix}.$$

The numerical result is shown in Table 3.

k	$\|\widetilde{S_k}\|_\infty$	$\|X_k\|_\infty$	$\|I_m - \widetilde{S_k}X_k\|_\infty$	$\|I_n - R_kA\|_\infty$	$\frac{\|R_k - A^\dagger\|_\infty}{\|A^\dagger\|_\infty}$
$k=1$	4.2000e+031	8.9775e−023	8.5425e+000	8.5425e+000	1.0000e+000
$k=2$	7.5425e+000	3.6029e+016	4.6093e+000	3.6479e+000	1.0000e+000
$k=3$	2.6479e+000	5.6867e+017	1.8928e+002	2.0025e+002	1.0000e+000
$k=4$	1.9925e+002	2.8177e+014	3.3036e+000	5.1361e+000	1.0000e+000
$k=5$	4.1363e+000	2.6385e+010	3.6051e−006	4.1939e−006	1.6027e−006
$k=6$	1.0000e+000	1.0000e+000	2.1665e−016	1.9621e−016	1.6646e−016
$k=7$	1.0000e+000	1.0000e+000	1.0625e−016	1.5601e−016	6.8464e−017

Table 3. Example 5.3.

5.2. Comparative results. Here we select a different number a in Example 5.2, we compare the relative error of Algorithm 2.5 with SVD based algorithms (Original SVD, Truncated SVD and Regularized SVD). Independently of the choice of a, the relative error of Algorithm 2.5 can be smaller than 10^{-11}. We list the number of iterations for Algorithm 2.5 to converge.

	$a=10^3$	$a=10^4$	$a=10^7$	$a=10^8$	$a=10^{15}$
Error-svd-original	1.9957e−010	4.0150e−008	9.3846e−003	1.0000	1.0000
Error-svd-truncated	1.9957e−010	4.0150e−008	9.3846e−003	3.2891e−001	1.0000
Error-svd-regularized	2.5518e−010	1.7566e−008	9.3846e−003	7.4549e−001	1.0000
Number of Iteration	2	2	3	3	5

Table 4. Relative error and number of iteration (Algorithm 2.5).

Next we choose different initial guess for $A^\dagger$ and compare the relative error results in Example 5.2, see Table 5. Here $\Delta A_{ij} = r_{ij}\mathbf{u}A_{ij}$ and Algorithm 2.5 is convergent for $R_1 = A^{\mathrm{T}}$.

	$R_1 = \mathrm{randn}(n,m)$	$R_1 = \mathbf{pinv}(A+\Delta A)$	$R_1 = A^{\mathrm{T}}$
$a=1$	0.7394	1.9517e−016	0
$a=10^4$	0.6984	8.7328e−013	1.1900e−016
$a=10^8$	0.2587	0.7101	1.2197e−016
$a=10^{15}$	0.3901	1.0000	1.4369e−016

Table 5. Relative error of different initial gueses.

Example 5.4 ([35]). We compute the *'glued matrices'* introduced by Smoktunowicz, Barlow, and Langou, and compare the computational residual and error of direct algorithms in [36] with Algorithm 2.5. This matrix A is given by the following Matlab code, with different values of the parameter c:

```
randn(state,0)
m=24; n=2; B=hilb(m);
A1=ones(m,n)-B(:,1:n)*c;
B=pascal(m); A2=B(:,1:n);
A3=randn(m,n)-A1;
A4=A1+1.1e-7*randn(m,n);
A5=A2-1.1e-7*randn(m,n);
B=magic(m); A6=B(:,1:n);
A=[A1 A2 A6+A2 A3 A4 A5-A4]';
```

Now we compare the residual and error of direct algorithms, see [36] with Algorithm 2.4. Since we do not know the exact Moore-Penrose inverse of A, we define the residual and error from [36] as follows:

$$\mathrm{res}_{\mathrm{Algorithm}} = \frac{\|A\widetilde{X}_{\mathrm{Algorithm}} - I_m\|_2}{\|A\|_2\|\widetilde{X}_{\mathrm{Algorithm}}\|_2}, \tag{5.1}$$

$$e_{\mathrm{Algorithm}} = \frac{\|\widetilde{X}_{\mathrm{Algorithm}} - \mathbf{pinv}(A)\|_2}{\mathbf{u}\|\mathbf{pinv}(A)\|_2\kappa_2(A)}. \tag{5.2}$$

The results are shown in Tables 6 and 7. It is obvious that we can compute the Moore-Penrose inverse of the *'glued matrices'* more accurate than that of [36].

c	$\kappa_2(A)$	e_{QR}	$e_{\mathrm{QR_{pivot}}}$	$e_{\mathrm{QR_{CGS2}}}$	$e_{\mathrm{Alg2.4}}$
1	1.44e+10	3.92e+2	5.21e+0	3.92e+2	4.31e−3
10^{-1}	1.37e+10	6.97e+3	5.66e+0	6.97e+3	5.68e−3
10^{-2}	1.25e+10	4.30e+4	5.66e+0	4.40e+4	6.48e−3
10^{-3}	1.29e+10	4.76e+5	5.79e+0	4.76e+5	3.20e−3
10^{-4}	1.36e+10	3.58e+6	5.81e+0	3.58e+6	2.14e−3
10^{-5}	1.20e+10	4.23e+7	5.25e+0	4.23e+7	8.40e−3
10^{-6}	5.58e+10	1.01e+8	1.20e+0	1.01e+8	2.99e−2
10^{-8}	6.62e+12	1.26e+8	1.55e+0	1.26e+8	3.38e−2

Table 6. Relative error of different initial gueses.

c	$\kappa_2(A)$	$\mathrm{res}_{\mathrm{QR}}$	$\mathrm{res}_{\mathrm{QR}_{\mathrm{pivot}}}$	$\mathrm{res}_{\mathrm{QR}_{\mathrm{CGS2}}}$	$\mathrm{res}_{\mathrm{QR}_{\mathrm{Bdiag1}}}$	$\mathrm{res}_{\mathrm{Bdiags2}}$	$\mathrm{res}_{\mathrm{SVD}}$	$\mathrm{res}_{\mathrm{Alg2.4}}$
1	1.44e+10	5.11e−15	1.14e−16	6.39e−15	1.33e−16	6.17e−17	5.39e−17	9.10e−27
10^{-1}	1.37e+10	3.26e−14	7.75e−17	5.21e−14	8.68e−17	8.15e−17	4.21e−17	1.08e−26
10^{-2}	1.25e+10	3.55e−13	2.63e−17	1.17e−12	8.14e−17	1.19e−16	5.21e−17	9.74e−27
10^{-3}	1.29e+10	4.63e−12	4.27e−17	9.25e−12	1.14e−16	7.29e−17	3.88e−17	7.44e−27
10^{-4}	1.36e+10	3.81e−11	8.02e−17	3.13e−11	1.19e−16	5.77e−17	5.78e−17	1.01e−26
10^{-5}	1.20e+10	2.58e−10	1.05e−16	8.87e−10	1.50e−16	9.44e−17	9.02e−17	6.77e−27
10^{-6}	5.58e+10	4.45e−10	6.04e−17	8.73e−10	6.38e−17	1.85e−16	1.44e−16	1.80e−27
10^{-8}	6.62e+12	1.21e−9	6.09e−17	3.63e−10	1.63e−16	1.40e−16	1.03e−16	2.28e−29

Table 7. Relative error of different initial gueses.

6. Appendix

6.1. Matlab code for Algorithm 2.5. The sub-function 'accdot' and 'ProdKL' (with a bit modification) are from INTLAB V5.6, see [30].

```
function R = MPInvIllco(A)
%  MP Inverse of Full rank extremely ill-conditioned matrices
%  The output R is stored in matrices R_1,...,R_k
[m,n] = size(A);
if (m>n)
   A=A';m=n;flag=1;
end
R=A';
kmax=15;res=0;
for k=1:kmax
    preres=res;
    S = ProdKL(A,R,k+1,1);
    X = inv(S);
    while any(any(isinf(X)))
          X = inv(C.*(1+sqrt(eps)*randn(m)));
    end
    R = ProdKL(R,X,k+1,k+1);
    res=norm(accdot(A,R,-1,eye(m)),'inf');
    if (abs(preres-res)<1e-16) break; end
end

while (flag)
```

```
    for i=1:length(R)
        R{i}=R{i}';
    end
end

return
```

6.2. The Proof of Theorem 3.1. It follows from [26] that we can estimate the approximation

$$|\widetilde{S_k} - S_k| \leqslant \mathbf{C}_1\sqrt{\mathbf{u}}|S_k|, \tag{6.1}$$

where $\mathbf{C}_1 = \mathcal{O}(1)$.

Using (6.1), we have the upper bound

$$|\widetilde{S_k}| \leqslant \frac{1}{1 - \mathbf{C}_1\sqrt{\mathbf{u}}}|S_k|. \tag{6.2}$$

In this section, we show that

$$\kappa_\infty(S_{k+1}) = \mathcal{O}(m)\sqrt{\mathbf{u}}\kappa_\infty(S_k) + \mathcal{O}(1), \tag{6.3}$$

provided that $\kappa_\infty(S_k) \leqslant \mathbf{u}^{-1}$.

First, we estimate $\|S_{k+1}\|_\infty$.

Let $\Gamma := \widetilde{S_k} - S_k$, it follows from (6.1) and (6.2) that

$$\|\Gamma\|_\infty \leqslant \mathbf{C}_1\sqrt{\mathbf{u}}\|S_k\|_\infty \leqslant \mathbf{C}_1'\sqrt{\mathbf{u}}\|\widetilde{S_k}\|_\infty, \tag{6.4}$$

where $\mathbf{C}_1' := \mathbf{C}_1/(1 - \mathbf{C}_i\sqrt{\mathbf{u}})$.

The difference between S_k (which is almost singular) and $\widetilde{S_k}$ is of order $\sqrt{\mathbf{u}}\|\widetilde{S_k}\|_\infty$. This implies that (cf. [10])

$$\kappa_\infty(\widetilde{S_k}) = \mathbf{C}_2\mathbf{u}^{-1/2}. \tag{6.5}$$

Now we need some assumptions given by [26].

Assumption 1. $\mathbf{C}_2 = \mathcal{O}(1)$.

It implies that

$$\kappa_\infty(\widetilde{S_k}) = \mathbf{C}_2\mathbf{u}^{-1/2} \ll \mathbf{u}^{-1}. \tag{6.6}$$

In the previous section, numerical examples show that Assumption 1 is satisfied in many cases.

Assumption 2.
$$\|I - \widetilde{S_k} X_k\|_\infty = \varepsilon \ll 1.$$

It follows from Assumption 2 that $\widetilde{S_k}^{-1}$ exists. Then we derive that

$$\begin{aligned}\|X_k - \widetilde{S_k}^{-1}\|_\infty &= \|\widetilde{S_k}^{-1}(I - \widetilde{S_k} X_k)\|_\infty \\ &\leqslant \|\widetilde{S_k}^{-1}\|_\infty \|I - \widetilde{S_k} X_k\|_\infty \\ &\leqslant \frac{\|X_k\|_\infty}{1 - \|I - \widetilde{S_k} X_k\|_\infty} \|I - \widetilde{S_k} X_k\|_\infty \\ &= \frac{\varepsilon}{1-\varepsilon} \|X_k\|_\infty.\end{aligned}$$

From the above equation, we can bound

$$\|X_k\|_\infty \leqslant \|\widetilde{S_k}^{-1}\|_\infty + \|X_k - \widetilde{S_k}^{-1}\|_\infty \leqslant \|\widetilde{S_k}^{-1}\|_\infty + \frac{\varepsilon}{1-\varepsilon}\|X_k\|_\infty. \tag{6.7}$$

It follows that

$$\|X_k\|_\infty \leqslant \frac{\|\widetilde{S_k}^{-1}\|_\infty}{1 - \varepsilon/(1-\varepsilon)} = \mathbf{C}_3 \|\widetilde{S_k}^{-1}\|_\infty, \tag{6.8}$$

where $\mathbf{C}_3 = (1-\varepsilon)/(1-2\varepsilon) = \mathcal{O}(1)$.

Since we use Matlab '**inv**' function, according to [26] we have

$$\|I - \widetilde{S_k} X_k\|_\infty \leqslant \mathbf{c}_m \mathcal{O}(\mathbf{u}) \|X_k\|_\infty \|\widetilde{S_k}\|_\infty, \tag{6.9}$$

where $\mathbf{c}_m = \mathcal{O}(m)$.

From (6.6), (6.8) and (6.9), we can achieve that

$$\|I - \widetilde{S_k} X_k\|_\infty \leqslant \mathbf{c}_m \mathcal{O}(\mathbf{u}) \kappa_\infty(\widetilde{S}) = \mathbf{c}_m C_4 \sqrt{\mathbf{u}}. \tag{6.10}$$

Assumption 2 is equivalent to

Assumption 3. $\mathbf{C}_4 = \mathcal{O}(1)$ satisfying $\mathbf{c}_m \mathbf{C}_4 \sqrt{\mathbf{u}} \ll 1$.

Lemma 6.1. *Suppose that Assumptions* 1 *and* 3 *are satisfied, then we have*

$$\|I - S_k X_k\|_\infty \leqslant \mathbf{C}_6, \tag{6.11}$$

where $\mathbf{C}_6 := \mathbf{C}_2 \mathbf{C}_3 (\mathbf{C}_1 + \mathbf{c}_m \mathcal{O}(1) \sqrt{\mathbf{u}})$.

P r o o f. Using (6.4), (6.8), and (6.9), we have

$$\begin{aligned}
\|I - S_kX_k\|_\infty &= \|I - (S_k - \widetilde{S_k} + \widetilde{S_k})X_k\|_\infty \\
&\leqslant \|(S_k - \widetilde{S_k})X_k\|_\infty + \|I - \widetilde{S_k}X_k\|_\infty \\
&\leqslant \mathbf{C}_1\sqrt{\mathbf{u}}\|X_k\|_\infty\|\widetilde{S_k}\|_\infty + \mathbf{c}_m\mathcal{O}(u)\|X_k\|_\infty\|\widetilde{S_k}\|_\infty \\
&\leqslant (\mathbf{C}_1 + \mathbf{c}_m\mathcal{O}(u))\|X_k\|_\infty\|\widetilde{S_k}\|_\infty \\
&= \mathbf{C}_5\sqrt{\mathbf{u}}\kappa_\infty(\widetilde{S_k}),
\end{aligned} \tag{6.12}$$

where $\mathbf{C}_5 = \mathbf{C}_3(\mathbf{C}_1 + \mathbf{c}_m\mathcal{O}(1)\mathbf{u})$. This equation and (6.5) prove the lemma. □

It follows from Lemma 6.1 that

$$\|S_kX_k\|_\infty = \|S_kX_k - I\|_\infty + \|I\|_\infty = 1 + \|S_kX_k - I\|_\infty \leqslant 1 + \mathbf{C}_6. \tag{6.13}$$

Then we can derive a relationship between S_{k+1} and X_kS_k:

$$|S_{k+1} - S_kX_k| = |AR_{k+1} - AR_kX_k| = |A(R_{k+1} - R_kX_k)| \leqslant |A||R_{k+1} - R_kX_k|. \tag{6.14}$$

Since $R_{k+1} = fl_{k,k}(R_k \cdot X_k)$, we have

$$|R_{k+1} - R_kX_k| \leqslant \mathbf{C}_7\mathbf{u}^{k+1}|R_kX_k|, \tag{6.15}$$

where $\mathbf{C}_7 = \mathcal{O}(1)$.

From (6.14) and (6.15), we have

$$|S_{k+1} - S_kX_k| \leqslant \mathbf{C}_7\mathbf{u}^{k+1}|A| \cdot |R_k| \cdot |X_k|. \tag{6.16}$$

Then

$$\|S_{k+1}\|_\infty \leqslant \|S_kX_k\|_\infty + \mathbf{u}^{k+1}\alpha, \tag{6.17}$$

where

$$\alpha := \mathbf{C}_7\||A| \cdot |R_k| \cdot |X_k|\|_\infty. \tag{6.18}$$

Now we introduce

Assumption 4. $\mathbf{u}^{k+1}\alpha \ll 1$.

If this assumption is not satisfied, then we modify Algorithm 2.5 as follows:

Algorithm 6.2. Modified Rump's method II for the Moore-Penrose inversion of an extremely ill-conditioned matrix

$R_1 = A^{\mathrm{T}}$, $X_0 = R_1$;

For $k = 1, 2, \ldots,$ until convergence

$S_k = fl_{(k-1)p+1,1}(A \cdot R_k)$ % stored in working precision

$\widetilde{S_k} = S_k + \Delta S_k$ % perturbation for S_k

$X_k = \mathbf{inv}(\widetilde{S_k})$ % floating-point Inverse

$R_{k+1} = fl_{(kp+1),(kp+1)}(R_{(k-1)p+1} \cdot X_k)$ % stored in $(kp+1)$-fold precision

end

Here '$\mathbf{inv}(B)$' is a built-in function in Matlab for the inversion of B, $(\Delta S_k)_{ij} = r_{ij}\sqrt{\mathbf{u}}(|S_k|)_{ij}$ for (i,j)-entry of ΔS_k. Here we denote $\kappa(S_k) = \|S_k\|_\infty \|S_k^{-1}\|_\infty$. If $\kappa_\infty(S_k) > \mathbf{u}^{-1}$, then we choose r_{ij} as pseudorandom numbers distributed uniformly in $[-1, 1]$; otherwise, we choose $r_{ij} = 0$.

Thus Assumption 4 becomes

Assumption 5. $\mathbf{u}^{kp+1}\alpha \ll 1$.

This assumption is satisfied for sufficiently large $p \in \mathbb{N}$ (integer). Without loss of generality, we can assume that Assumption 4 is satisfied.

Under Assumption 4, it can be seen from (6.16) that

$$\|S_{k+1}\|_\infty = \|S_k X_k\|_\infty + \varepsilon, \tag{6.19}$$

where $\varepsilon \ll 1$.

Now, we estimate $\|S_{k+1}^{-1}\|_\infty$.

Let $\Delta = X_k^{-1} - \widetilde{S_k}$, from (6.4) and (6.10) we have

$$\begin{aligned}
\|\Delta\|_\infty &= \|X_k^{-1} - \widetilde{S_k}\|_\infty \\
&= \|(I - \widetilde{S_k} X_k) X_k^{-1}\|_\infty \\
&\leqslant \|I - \widetilde{S_k} X_k\|_\infty \|X_k^{-1}\|_\infty \\
&\leqslant \|I - \widetilde{S_k} X_k\|_\infty \frac{\|\widetilde{S_k}\|_\infty}{1 - \|I - \widetilde{S_k} X_k\|_\infty} \\
&\leqslant \frac{\mathbf{c}_m \mathbf{C}_4 \sqrt{\mathbf{u}}}{1 - \mathbf{c}_m \mathbf{C}_4 \sqrt{\mathbf{u}}} \|\widetilde{S_k}\|_\infty \\
&\leqslant \mathbf{c}_m \mathbf{C}_8 \sqrt{\mathbf{u}} \|S_k\|_\infty,
\end{aligned}$$

where $\mathbf{C}_8 := \mathbf{C}_1' \mathbf{C}_4 / (1 - \mathbf{c}_m \mathbf{C}_4 \sqrt{\mathbf{u}}) = \mathcal{O}(1)$.

It follows from (6.4) that

$$\begin{aligned}\|(S_kX_k)^{-1}\|_\infty &= \|(S_k(S_k+\Delta+\Gamma)^{-1})^{-1}\|_\infty \\ &= \|I+(\Delta+\Gamma)S_k^{-1}\|_\infty \\ &\leqslant 1+\|S_k^{-1}\|_\infty(\|\Gamma\|_\infty+\|\Delta\|_\infty) \\ &\leqslant 1+(\mathbf{C}_1'+\mathbf{c}_m\mathbf{C}_8)\sqrt{\mathbf{u}}\|S_k^{-1}\|_\infty\|S_k\|_\infty \\ &\leqslant 1+(\mathbf{C}_1'+\mathbf{c}_m\mathbf{C}_8)\sqrt{\mathbf{u}}\kappa_\infty(S_k).\end{aligned}$$

For nonsingular matrices P and Q, we drop the second order terms, it is well known that

$$\|P^{-1}-Q^{-1}\| = \|P^{-1}(P-Q)Q^{-1}\| \leqslant \|P-Q\|\|P^{-1}\|\|Q^{-1}\|.$$

From (6.16), we get

$$(6.20)\qquad \begin{aligned}\|S_{k+1}^{-1}-(S_kX_k)^{-1}\|_\infty &\leqslant \|S_{k+1}-S_kX_k\|_\infty\|S_{k+1}^{-1}\|_\infty\|(S_kX_k)^{-1}\|_\infty \\ &\leqslant \mathbf{u}^{k+1}\beta\|S_{k+1}^{-1}\|_\infty,\end{aligned}$$

where $\beta := \mathbf{C}_7\||A|\cdot|R_k|\cdot X_k\|_\infty\|(S_kX_k)^{-1}\|_\infty$.

From (6.20), we have

$$(6.21)\qquad \begin{aligned}\|S_{k+1}^{-1}\|_\infty &\leqslant \|S_{k+1}^{-1}-(S_kX_k)^{-1}\|_\infty+\|(S_kX_k)^{-1}\|_\infty \\ &\leqslant \mathbf{u}^{k+1}\beta\|S_{k+1}^{-1}\|_\infty+\|(S_kX_k)^{-1}\|_\infty.\end{aligned}$$

Let the following assumption hold:

Assumption 6. $\mathbf{u}^{k+1}\beta \ll 1$.

Then we have

$$(6.22)\qquad \|S_{k+1}^{-1}\|_\infty \leqslant (1-\mathbf{u}^{k+1}\beta)^{-1}\|(S_kX_k)^{-1}\|_\infty.$$

If Assumption 6 is not satisfied, then we use the modified Rump's method II (Algorithm 6.2). Namely, we introduce

Assumption 7. $\mathbf{u}^{kp+1}\beta \ll 1$.

This assumption is satisfied, if we choose a sufficiently large $m \in \mathbb{N}$, then (6.22) becomes

$$(6.23)\qquad \|S_{k+1}^{-1}\|_\infty \leqslant (1-\mathbf{u}^{kp+1}\beta)^{-1}\|(S_kX_k)^{-1}\|_\infty.$$

Without loss of generality, we can assume that Assumption 6 is satisfied. Then

$$\|S_{k+1}^{-1}\|_\infty \leqslant \mathbf{C}_9 |(S_k X_k)^{-1}\|_\infty, \tag{6.24}$$

where $\mathbf{C}_9 = \mathcal{O}(1)$.

From (6.13), (6.19) and (6.24), we have

$$\begin{aligned}\kappa_\infty(S_{k+1}) &= \|S_{k+1}\|_\infty \|S_{k+1}^{-1}\|_\infty \\ &\leqslant (\|S_k X_k\|_\infty + \varepsilon)\mathbf{C}_9 \|(S_k X_k)^{-1}\|_\infty \\ &\leqslant (1 + \mathbf{C}_6 + \varepsilon)\mathbf{C}_9 (1 + \mathbf{C}_1' c_m C_8 \sqrt{\mathbf{u}} \kappa_\infty(S_k)) \\ &\leqslant \mathcal{O}(m)\sqrt{\mathbf{u}}\kappa_\infty(S_k) + \mathcal{O}(1).\end{aligned}$$

If Assumptions 1, 3, 4, and 6 (or Assumptions 1, 3, 5, and 7) are satisfied, then we can prove Theorem 3.1.

6.3. The Proof of Theorem 3.2. In this section, we need to prove that $\|I - S_{k+1}\|_\infty = \mathcal{O}(\sqrt{\mathbf{u}})$ if $\kappa_\infty(S_k) = \mathcal{O}(1)$. Since $\|S_k - \widetilde{S_k}\|_\infty \leqslant \mathbf{C}_1 \sqrt{\mathbf{u}} \|S_k\|_\infty$, we can estimate $\kappa_\infty(\widetilde{S_k}) \approx \kappa_\infty(S_k) = \mathcal{O}(1)$. Then we can expect that X_k becomes a good approximation inverse of $\widetilde{S_k}$ satisfying

$$\|I - \widetilde{S_k} X_k\|_\infty \ll 1. \tag{6.25}$$

This implies that there exists $\mathbf{C}_{10} = \mathcal{O}(1)$ such that

$$\|X_k\|_\infty \leqslant \mathbf{C}_{10} \|\widetilde{S_k}^{-1}\|_\infty. \tag{6.26}$$

Then, from (6.12) we have

$$\|I - S_k X_k\|_\infty \leqslant \mathbf{C}_5 \sqrt{\mathbf{u}} \kappa_\infty(\widetilde{S_k}) = \mathbf{C}_{11} \sqrt{\mathbf{u}}. \tag{6.27}$$

Thus, from (6.16) and (6.27) we have

$$\|I - S_{k+1}\|_\infty \leqslant \|I - S_k X_k\|_\infty + \|S_k X_k - S_{k+1}\|_\infty \leqslant \mathbf{C}_{11} \sqrt{\mathbf{u}} + \mathbf{u}^{k+1}\alpha, \tag{6.28}$$

where α is defined in (6.18). Since $\kappa_\infty(\widetilde{S_k}) = \mathcal{O}(1)$, we introduce

Assumption 8. $\mathbf{C}_{11} = \mathcal{O}(1)$.

Furthermore, we assume that k is so large that the following inequality holds:

Assumption 9. $\mathbf{u}^{k+1}\alpha \ll 1$.

If this assumption does not hold, then we use the modified Rump's method II (Algorithm 6.2), and

$$\|I - S_{k+1}\|_\infty \leqslant \mathbf{C}_{10}\sqrt{\mathbf{u}} + \mathbf{u}^{kj+1}\alpha \tag{6.29}$$

holds. If j is large enough, then the following inequality holds:

Assumption 10. $\mathbf{u}^{kj+1}\alpha \ll 1$.

Without loss of generality, we can assume that Assumption 9 is satisfied. If Assumptions 8 and 9 (or Assumptions 8 and 10) are satisfied, then we can prove Theorem 3.2.

Acknowledgement. Y. Wei would like to thank Professors M. Fielder and S. Rump for their reprints [16], [15], [25], [26] and to Professors N. Castro-González, A. Cichocki, K. Hayami, S. Oishi, S. Rump and K. Tanabe for useful discussions. We thank Professor Frank Hall and the referee for very detailed comments which give a simplified proof of Theorem 2.2.

References

[1] *A. Ben-Israel, T. N. E. Greville*: Generalized Inverses. Theory and Applications. Springer, New York, 2003.
[2] *Å. Björck*: Numerical Methods for Least Squares Problems. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1996.
[3] *S. L. Campbell, C. D. Meyer*: Generalized Inverses of Linear Transformations. Dover Publications, New York, 1991.
[4] *N. Castro-González, J. Ceballos, F. M. Dopico, J. M. Molera*: Accurate solution of structured least squares problems via rank-revealing decompositions. SIAM J. Matrix Anal. Appl. *34* (2013), 1112–1128.
[5] *L. Chen, E. V. Krishnamurthy, I. Madeod*: Generalised matrix inversion and rank computation by successive matrix powering. Parallel Comput. *20* (1994), 297–311.
[6] *P. Courrieu*: Fast computation of Moore-Penrose inverse matrices. Neural Information Processing *8* (2005), 25–29.
[7] *F. Cucker, H. Diao, Y. Wei*: On mixed and componentwise condition numbers for Moore-Penrose inverse and linear least squares problems. Math. Comput. *76* (2007), 947–963.
[8] *F. Cucker, H. Diao, Y. Wei*: Smoothed analysis of some condition numbers. Numer. Linear Algebra Appl. *13* (2006), 71–84.
[9] *J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, Z. Drmač*: Computing the singular value decomposition with high relative accuracy. Linear Algebra Appl. *299* (1999), 21–80.
[10] *J. W. Demmel, Y. Hida, X. S. Li, E. J. Riedy*: Extra-precise iterative refinement for overdetermined least squares problems. ACM Trans. Math. Software *35* (2009), Art. 28, 32 pages.
[11] *J. Demmel, N. J. Higham*: Improved error bounds for underdetermined system solvers. SIAM J. Matrix Anal. Appl. *14* (1993), 1–14.

[12] *H. Diao, Y. Wei*: On Frobenius normwise condition numbers for Moore-Penrose inverse and linear least-squares problems. Numer. Linear Algebra Appl. *14* (2007), 603–610.
[13] *H. Diao, Y. Wei, S. Qiao*: Structured condition numbers of structured Tikhonov regularization problem and their estimations. J. Comput. Appl. Math. *308* (2016), 276–300.
[14] *F. M. Dopico, J. M. Molera*: Accurate solution of structured linear systems via rank-revealing decompositions. IMA J. Numer. Anal. *32* (2012), 1096–1116.
[15] *M. Fiedler*: Moore-Penrose biorthogonal systems in Euclidean spaces. Linear Algebra Appl. *362* (2003), 137–143.
[16] *M. Fiedler, T. L. Markham*: A characterization of the Moore-Penrose inverse. Linear Algebra Appl. *179* (1993), 129–133.
[17] *G. H. Golub, C. F. Van Loan*: Matrix Computations. Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, 2013.
[18] *M. Gulliksson, P. Å. Wedin, Y. Wei*: Perturbation identities for regularized Tikhonov inverses and weighted pseudoinverses. BIT *40* (2000), 513–523.
[19] *J. Jones, N. P. Karampetakis, A. C. Pugh*: The computation and application of the generalized inverse via Maple. J. Symb. Comput. *25* (1998), 99–124.
[20] *V. N. Katsikis, D. Pappas*: Fast computing of the Moore-Penrose inverse matrix. Electron. J. Linear Algebra (electronic only) *17* (2008), 637–650.
[21] *Z.-C. Li, H.-T. Huang, Y. Wei, A. H.-D. Cheng*: Effective Condition Number for Numerical Partial Differential Equations. Alpha Science International, Oxford, 2014.
[22] *Z. Li, Q. Xu, Y. Wei*: A note on stable perturbations of Moore-Penrose inverses. Numer. Linear Algebra Appl. *20* (2013), 18–26.
[23] *T. Ogita, S. M. Rump, S. Oishi*: Accurate sum and dot product. SIAM J. Sci. Comput. *26* (2005), 1955–1988.
[24] *T. Ohta, T. Ogita, S. M. Rump, S. Oishi*: Numerical verification method for arbitrarily ill-conditioned linear systems. Transactions of the Japan Society for Industrial and Applied Mathematics *15* (2005), 269–287.
[25] *S. Oishi, S. M. Rump*: Fast verification of solutions of matrix equations. Numer. Math. *90* (2002), 755–773.
[26] *S. Oishi, K. Tanabe, T. Ogita, S. M. Rump*: Convergence of Rump's method for inverting arbitrarily ill-conditioned matrices. J. Comput. Appl. Math. *205* (2007), 533–544.
[27] *C. R. Rao, S. K. Mitra*: Generalized Inverses of Matrices and Its Applications. Wiley Series in Probability and Mathematical Statistics, Wiley & Sons, New York, 1971.
[28] *S. M. Rump*: Verified bounds for least squares problems and underdetermined linear systems. SIAM J. Matrix Anal. Appl. *33* (2012), 130–148.
[29] *S. M. Rump*: Inversion of extremely ill-conditioned matrices in floating-point. Japan J. Ind. Appl. Math. *26* (2009), 249–277.
[30] *S. M. Rump*: INTLAB—Interval Laboratory, the Matlab toolbox for verified computations, Version 5.3. Institute for Reliable Computing, Hamburg, 2006.
[31] *S. M. Rump*: Ill-conditioned matrices are componentwise near to singularity. SIAM Rev. *41* (1999), 102–112.
[32] *S. M. Rump*: Ill-conditionedness need not be componentwise near to ill-posedness for least squares problems. BIT *39* (1999), 143–151.
[33] *S. M. Rump*: A class of arbitrarily ill-conditioned floating-point matrices. SIAM J. Matrix Anal. Appl. *12* (1991), 645–653.
[34] *S. M. Rump*: Approximate Inverses of Almost Singular Matrices Still Contain Useful Information, Technical Report 90.1. Faculty for Information and Communications Sciences, TU Hamburg, Harburg, 1990.
[35] *A. Smoktunowicz, J. Barlow, J. Langou*: A note on the error analysis of classical Gram-Schmidt. Numer. Math. *105* (2006), 299–313.

[36] *A. Smoktunowicz, I. Wróbel*: Numerical aspects of computing the Moore-Penrose inverse of full column rank matrices. BIT *52* (2012), 503–524.
[37] *G. W. Stewart*: On the perturbation of pseudo-inverses, projections and linear least squares problems. SIAM Rev. *19* (1977), 634–662.
[38] *G. Wang, Y. Wei, S. Qiao*: Generalized Inverses: Theory and Computations. Science Press, Beijing, 2004.
[39] *P. Å. Wedin*: Perturbation theory for pseudo-inverses. BIT, Nord. Tidskr. Inf.-behandl. *13* (1973), 217–232.
[40] *Y. Wei*: Generalized inverses of matrices, Chapter 27. Handbook of Linear Algebra (L. Hogben, ed.). Chapman & Hall/CRC Press, Boca Raton, 2014, pp. 27-1–27-15.
[41] *Y. Wei, J. Ding*: Representations for Moore-Penrose inverses in Hilbert spaces. Appl. Math. Lett. *14* (2001), 599–604.
[42] *Y. Wei, H. Wu*: Expression for the perturbation of the weighted Moore-Penrose inverse. Comput. Math. Appl. *39* (2000), 13–18.
[43] *Y. Wei, P. Xie, L. Zhang*: Tikhonov regularization and randomized GSVD. SIAM J. Matrix Anal. Appl. *37* (2016), 649–675.
[44] *W. Xu, Y. Wei, S. Qiao*: Condition numbers for structured least squares problems. BIT *46* (2006), 203–225.
[45] *L. Zhou, L. Lin, Y. Wei, S. Qiao*: Perturbation analysis and condition numbers of scaled total least squares problems. Numer. Algorithms *51* (2009), 381–399.
[46] *G. Zielke*: Report on test matrices for generalized inverses. Computing *36* (1986), 105–162.

Authors' addresses: Yunkun Chen, School of Mathematical Sciences, Xiamen University, Shi Ming Nan Road, No. 422, Xiamen 361005, Fujian, P. R. China, and School of Mathematics and Computer Science, Guizhou Normal University, Bao Shan Bei Road, No. 116, Guiyang 550001, Guizhou, P. R. China, e-mail: cyk2009sd@163.com; Xinghua Shi, School of Mathematical Sciences, Fudan University, Handan Road, No. 220, Shanghai, 200433, P. R. China, e-mail: xhshi1984@gmail.com; Yimin Wei, School of Mathematical Sciences, Fudan University, Handan Road, No. 220, Shanghai, 200433, P. R. China, and Shanghai Key Laboratory of Contemporary Applied Mathematics, Handan Road, No. 220, Shanghai, 200433, P. R. China. e-mail: ymwei@fudan.edu.cn.