

Research Article

Improved Qrginv Algorithm for Computing Moore-Penrose Inverse Matrices

Alireza Ataei

Mathematics Department, Persian Gulf University, Bushehr 75169, Iran

Correspondence should be addressed to Alireza Ataei; ataei@pgu.ac.ir

Received 24 January 2014; Accepted 17 February 2014; Published 12 March 2014

Academic Editors: M. Hermann and M.-H. Hsu

Copyright © 2014 Alireza Ataei. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Katsikis et al. presented a computational method in order to calculate the Moore-Penrose inverse of an arbitrary matrix (including singular and rectangular) (2011). In this paper, an improved version of this method is presented for computing the pseudo inverse of an $m \times n$ real matrix A with rank $r > 0$. Numerical experiments show that the resulting pseudoinverse matrix is reasonably accurate and its computation time is significantly less than that obtained by Katsikis et al.

1. Introduction

Let $\mathbb{R}^{m \times n}$ denote the set of all $m \times n$ matrices over the field of real numbers, \mathbb{R} . The symbols A^T , $\text{rank}(A^T)$ will stand for the transpose and rank of $A \in \mathbb{R}^{m \times n}$, respectively. For a matrix $A \in \mathbb{R}^{m \times n}$, the Moore-Penrose inverse of A , denoted by A^\dagger , is the unique matrix $X \in \mathbb{R}^{n \times m}$ satisfying the following equations:

- (i) $AXA = A$,
- (ii) $XAX = X$,
- (iii) $(AX)^T = AX$,
- (iv) $(XA)^T = XA$.

A lot of works concerning generalized inverses have been carried out, in finite and infinite dimensions (e.g., [1–3]). There are several methods for computing the Moore-Penrose inverse matrix [2, 4–8]. In a recent article [9], an improved method for the computation of the Moore-Penrose inverse matrix provided.

In this paper, we aim to improve their method so that it can be used for any kind of matrices, square or rectangular, full rank or not. The numerical examples show that our method is competitive in terms of accuracy and is much faster than the commonly used methods and can also be used for large sparse matrices.

This paper is organized as follows. In Section 2 the improved version of this method is presented for computing

the pseudoinverse of an $m \times n$ real matrix A with rank $r > 0$. In Section 3, the numerical results of some test matrices are given. Section 4 is devoted to the concluding remarks.

2. Improved Qrginv Method (IMqrginv)

In [9], a method (qrginv) for computing the Moore-Penrose inverse of an arbitrary matrix was presented. They made use of the QR-factorization, as well as an algorithm based on a known reverse order law for generalized inverse matrices, and also they apply a method (ginv), presented in [4], based on a full rank Cholesky factorization of possibly singular symmetric positive matrices.

In the current paper, we improved *qrginv* algorithm using the QR-factorization by Gram-Schmidt orthonormalization (GSO) and Theorem 1 for faster computing Moore-Penrose inverse of arbitrary matrices (including singular and rectangular). We should note that we invoke *ginv* algorithm. In order to support and state our achievement we need to remind Gram-Schmidt orthonormalization (GSO) and the QR-factorization.

2.1. The Gram-Schmidt Procedure. Let us remind a generalization of the Gram-Schmidt orthonormalization process (shortly GSO) which is applied for singular matrices. Let $\mathcal{A} = \{a_1, a_2, \dots, a_n\} \subseteq \mathbb{R}^m$ be a set of vectors spanning a subspace V . This process generates a set of mutually

TABLE 1: Error and computational time results for random singular matrices.

n	Method	Time	$\ AA^\dagger A - A\ _2$	$\ A^\dagger AA^\dagger - A^\dagger\ _2$	$\ (AA^\dagger)^T - (AA^\dagger)\ _2$	$\ (A^\dagger A)^T - (A^\dagger A)\ _2$
2^7	qrginv	0.0317	$3.4169e - 014$	$0.2638e - 016$	$5.7671e - 015$	$3.2645e - 015$
	IMqrginv	0.0137	$1.6175e - 014$	$3.7297e - 016$	$3.0344e - 015$	$2.5228e - 015$
2^8	qrginv	0.1176	$1.0052e - 013$	$9.9243e - 016$	$8.0286e - 015$	$4.0476e - 015$
	IMqrginv	0.0786	$2.9804e - 014$	$3.4041e - 016$	$3.8500e - 015$	$3.1148e - 015$
2^9	qrginv	1.0584	$1.3649e - 013$	$1.0186e - 015$	$1.1006e - 014$	$4.9494e - 015$
	IMqrginv	0.8236	$5.3456e - 014$	$5.0122e - 015$	$5.0122e - 015$	$3.8394e - 015$
2^{10}	qrginv	10.8336	$2.6266e - 013$	$1.0616e - 015$	$1.4943e - 014$	$5.9069e - 015$
	IMqrginv	8.2976	$8.8502e - 014$	$2.9117e - 016$	$6.3837e - 015$	$4.8854e - 015$
2^{11}	qrginv	81.1429	$4.9931e - 013$	$1.1850e - 015$	$2.1858e - 014$	$7.2568e - 015$
	IMqrginv	55.0746	$1.6676e - 013$	$2.5936e - 016$	$8.3743e - 015$	$6.5585e - 015$

orthonormal vectors such as $\mathcal{Q} = \{q_1, q_2, \dots, q_r\} \subseteq \mathbb{R}^m$ having the property that \mathcal{Q} is an orthonormalization basis for V . \mathcal{Q} is obtained using the Gram-Schmidt orthonormalization process (shortly GSO) as follows:

$$\begin{aligned}
 q_1 &= \frac{a_{c_1}}{\|a_{c_1}\|}, \quad \text{if } a_{c_1} \neq 0 = a_j \text{ for } 1 \leq j < c_1, \\
 \hat{q}_j &= a_j - \sum_{i=1}^{j-1} (a_j, q_i) q_i, \quad j = c_{k-1} + 1, c_{k-2} + 2, \dots, c_k, \\
 q_k &= \frac{\hat{q}_j}{\|\hat{q}_j\|}, \\
 \text{if } \hat{q}_{c_k} &\neq 0 = q_j, \text{ for } c_{k-1} + 1 \leq j < c_k, \quad k = 2, \dots, r.
 \end{aligned} \tag{1}$$

The integer r found by the GSO process is the dimension of the subspace V . The integers $\{c_1, \dots, c_r\}$ are the indices of a maximal linearly independent subset $\{a_{c_1}, \dots, a_{c_r}\}$ of \mathcal{A} .

2.2. The QR-Factorization. Let us remind the QR-factorization for arbitrary matrices (including singular and rectangular).

Let the orthonormal set $\{q_1, q_2, \dots, q_r\}$ be obtained from the set of vectors $\{a_1, a_2, \dots, a_n\}$ by the GSO process described in Section 2.1, and let

$$\begin{aligned}
 \tilde{Q} &= [q_1, q_2, \dots, q_r] \in \mathbb{R}^{m \times r}, \\
 A &= [a_1, a_2, \dots, a_n] \in \mathbb{R}^{m \times n},
 \end{aligned} \tag{2}$$

with $\text{rank}(A) = r > 0$, be the corresponding matrices. Then there exist matrices \tilde{A} , Q , and R such that

$$\tilde{A} = AP = QR, \tag{3}$$

where

- (i) P is a permutation matrix (therefore orthonormal);
- (ii) $Q = [\tilde{Q} \ Z] \in \mathbb{R}^{m \times m}$, where \tilde{Q} and Z denote matrices whose columns are an orthonormal basis of $R(A)$ and $N(A^T)$, respectively;
- (iii) $R = \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \in \mathbb{R}^{m \times n}$, where $\tilde{R} \in \mathbb{R}^{r \times n}$ is upper triangular matrix with $\text{rank}(\tilde{R}) = r$.

One obtains from (3)

$$\tilde{A} = [\tilde{Q} \ Z] \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} = \tilde{Q}\tilde{R}. \tag{4}$$

It follows that \tilde{A} has a $\tilde{Q}\tilde{R}$ -factorization. A nonzero matrix can be expressed as the product of a matrix of full column rank and a matrix of full row rank. In fact, for given $A \in \mathbb{R}^{m \times n}$ ($\text{rank}(A) = r > 0$) there exist matrices $F \in \mathbb{R}^{m \times r}$ and $G \in \mathbb{R}^{r \times n}$ such that $A = FG$ [2]. Such factorization, which is the so-called a full rank factorization, turns out to be a powerful tool in the study of generalized inverses.

The following theorem is due to C. C MacDuffe [10] who apparently was the first one to point out that a full rank factorization of a matrix A leads to an explicit formula for its Moore-Penrose inverse, A^\dagger .

Theorem 1. *If $A \in \mathbb{R}^{m \times n}$ matrix, with $\text{rank}(A) = r > 0$, has a full rank factorization $A = FG$, then*

$$A^\dagger = G^T (F^T A G^T)^{-1} F^T. \tag{5}$$

As a direct consequence of Theorem 1, we have the following.

Corollary 2. *Let $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) = r > 0$, and $\tilde{A} = \tilde{Q}\tilde{R}$ be the $\tilde{Q}\tilde{R}$ -factorization of \tilde{A} . Then*

$$A^\dagger = P^T (\tilde{R}^T) (\tilde{R}\tilde{R}^T)^{-1} \tilde{Q}^T. \tag{6}$$

Proof. With $A = \tilde{A}$, $F = \tilde{Q}$, and $G = \tilde{R}$ in Theorem 1 one obtains

$$(\tilde{A})^\dagger = (AP)^\dagger = \tilde{R}^T (\tilde{R}\tilde{R}^T)^{-1} \tilde{Q}^T, \tag{7}$$

and consequently

$$A^\dagger = P\tilde{R}^T (\tilde{R}\tilde{R}^T)^{-1} \tilde{Q}^T. \tag{8}$$

□

The function *IMqrginv* 4 provided all implementation details of the above solution, in MATLAB code. To calculate

TABLE 2: Error and computational time results for *chow* (rank = 199, cond = $5.9407e + 135$).

Method	Time	$\ AA^\dagger A - A\ _2$	$\ A^\dagger AA^\dagger - A^\dagger\ _2$	$\ (AA^\dagger)^T - (AA^\dagger)\ _2$	$\ (A^\dagger A)^T - (A^\dagger A)\ _2$
qrginv	0.0376	$3.6711e - 013$	$1.7331e - 013$	$4.4765e - 013$	$2.4702e - 013$
IMqrginv	0.0187	$4.0038e - 013$	$1.7331e - 013$	$2.4448e - 013$	$2.4741e - 013$

TABLE 3: Error and computational time results for *cycol* (rank = 50, cond = $3.7045e + 048$).

Method	Time	$\ AA^\dagger A - A\ _2$	$\ A^\dagger AA^\dagger - A^\dagger\ _2$	$\ (AA^\dagger)^T - (AA^\dagger)\ _2$	$\ (A^\dagger A)^T - (A^\dagger A)\ _2$
qrginv	0.2816	$1.5114e - 013$	$2.3446e - 017$	$1.1512e - 015$	$8.8889e - 016$
IMqrginv	0.2334	$8.1308e - 014$	$1.4034e - 017$	$1.0225e - 015$	$8.2643e - 016$

TABLE 4: Error and computational time results for *gearmat* (rank = 199, cond = $1.8074e + 016$).

Method	Time	$\ AA^\dagger A - A\ _2$	$\ A^\dagger AA^\dagger - A^\dagger\ _2$	$\ (AA^\dagger)^T - (AA^\dagger)\ _2$	$\ (A^\dagger A)^T - (A^\dagger A)\ _2$
qrginv	0.0377	$4.6584e - 015$	$3.0532e - 013$	$1.1996e - 013$	$2.1253e - 014$
IMqrginv	0.0165	$2.8959e - 015$	$3.3357e - 013$	$7.7888e - 014$	$2.1380e - 014$

TABLE 5: Error and computational time results for *kahan* (rank = 199, cond = $1.9055e + 024$).

Method	Time	$\ AA^\dagger A - A\ _2$	$\ A^\dagger AA^\dagger - A^\dagger\ _2$	$\ (AA^\dagger)^T - (AA^\dagger)\ _2$	$\ (A^\dagger A)^T - (A^\dagger A)\ _2$
qrginv	0.0279	$1.9877e - 005$	$4.1177e - 009$	$8.8330e - 001$	$5.4162e - 014$
IMqrginv	0.0099	$1.9877e - 005$	$3.8389e - 009$	$8.8330e - 001$	$1.0398e - 014$

TABLE 6: Error and computational time results for *lotkin* (rank = 19, cond = $3.8210e + 021$).

Method	Time	$\ AA^\dagger A - A\ _2$	$\ A^\dagger AA^\dagger - A^\dagger\ _2$	$\ (AA^\dagger)^T - (AA^\dagger)\ _2$	$\ (A^\dagger A)^T - (A^\dagger A)\ _2$
qrginv	0.0244	$8.2512e - 006$	$1.2717e - 011$	$4.4898e - 002$	$3.9689e - 009$
IMqrginv	0.0102	$8.2512e - 006$	$3.2435e - 009$	$4.4898e - 002$	$1.2636e - 011$

the rank of \tilde{R} , one needs only the number of its columns having at least one value above a tolerance level in absolute terms. This tolerance is set to be equal to 10^{-5} , which is also used by Katsikis et al. [9], and turns out to provide accurate results.

3. Numerical Examples

In this section, we compare the performance of the proposed method (*IMqrginv* function) to that of Katsikis et al. [9] for the computation of Moore-Penrose inverse matrices. Testing *qrginv* and *IMqrginv* was performed separately for random singular and singular matrices with “large” condition number from the Matrix Computation Toolbox (see [11]). We also tested the proposed method for some sparse matrices and obtained very fast and accurate results. Specifically, MATLAB 7.11 (R2010b) Service Pack 3 version of the software was used on an Intel Core 2 (Duo) 8400 Processor running 2.26 GHz with 3 GB of RAM memory using Windows Vista professional 32-bit operating system.

3.1. Random Singular Matrices. We are computing the performance of the proposed method *IMqrginv* to that of [9] (*qrginv* function). In the same way of [4] we tested on a series of random singular matrices of size $m \times n$, with $n = 2^k, k =$

$7, \dots, 11$, and $m = 2n$, which are rank deficient, with rank $r = 7n/8$. In addition, the accuracy of the results is examined with the matrix 2-norm in error matrices corresponding to the four properties characterizing the Moore-Penrose inverses shown in Table 1. The computation error is less than 10^{-12} per coefficient in the error matrices, in all cases. The computation time (in seconds) is reported in Table 1. We observe that the computation time of *IMqrginv* method is substantially less than that of the *qrginv* method.

3.2. Singular Matrices. In this section we use a set of singular test matrices that includes singular matrices of size 200×200 , obtained from the function *matrix* in the Matrix Computation Toolbox [11] (which includes test matrices from Matlab itself). The condition number of test matrices ranges from order 10^{16} to 10^{135} . Since the matrices are of relatively small size and so as to measure the time needed for each algorithm to compute the Moore-Penrose inverse accurately, each algorithm runs 100 distinct times. The reported time is the mean time over these 100 replications. For each case, the time responses together with the error results are presented in Tables 2, 3, 4, 5, 6, 7, 8, and 9. We observe that the computation time of *IMqrginv* method is shorter than the *qrginv* method for all matrices and is proved to be more efficient.

TABLE 7: Error and computational time results for *prolate* (rank = 117, cond = 4.7489e + 017).

Method	Time	$\ AA^+A - A\ _2$	$\ A^+AA^+ - A^+\ _2$	$\ (AA^+)^T - (AA^+)\ _2$	$\ (A^+A)^T - (A^+A)\ _2$
qrginv	0.0276	1.3837e - 006	1.2998e - 007	4.7715e - 002	4.7317e - 011
IMqrginv	0.0138	1.3837e - 006	1.1842e - 007	4.7715e - 002	4.7401e - 011

TABLE 8: Error and computational time results for *hilb* (rank = 20, cond = 4.4158e + 021).

Method	Time	$\ AA^+A - A\ _2$	$\ A^+AA^+ - A^+\ _2$	$\ (AA^+)^T - (AA^+)\ _2$	$\ (A^+A)^T - (A^+A)\ _2$
qrginv	0.0247	7.7880e - 006	1.1444e - 008	1.0053e - 001	5.5974e - 012
IMqrginv	0.0117	7.7880e - 006	1.1184e - 008	1.0053e - 001	5.5636e - 012

TABLE 9: Error and computational time results for *magic* (rank = 3, cond = 5.3481e + 019).

Method	Time	$\ AA^+A - A\ _2$	$\ A^+AA^+ - A^+\ _2$	$\ (AA^+)^T - (AA^+)\ _2$	$\ (A^+A)^T - (A^+A)\ _2$
qrginv	0.0154	1.5837e - 009	4.4922e - 009	9.4641e - 014	6.3953e - 015
IMqrginv	0.0035	1.4929e - 009	4.8349e - 009	4.7537e - 014	6.0546e - 015

TABLE 10: Test problem information.

Matrix\property	m	n	nnz	Cond
WELL1033	1033	320	4732	Not available
WELL1850	1850	712	8758	Not available
ILCC1033	1033	320	4732	Not available
ILCC1850	10850	712	8758	Not available
WATT1	1856	1856	11360	5.4e + 009
GR-30-30	900	900	4322	3.8e + 002
ADD20	2395	2395	17319	1.76e + 004
NOSE3	960	960	8402	7.3e + 004
SHERMAN1	1000	1000	3750	2.3e + 004

3.3. *Matrix-Market Sparse Matrices.* For sparse matrices, we have chosen some matrices from Matrix-Market collection [11]. We follow the same method as in [8], and we have the rank deficient matrices as

$$A_Z \equiv [A \ Z], \quad (9)$$

where A is one of the chosen matrices and Z is a zero matrix of order $m \times 100$. The chosen matrices with their properties are shown in Table 10. The results of the methods are presented in Table 11. We observe that the Moore-Penrose inverses obtained by *IMqrginv* are reasonably accurate in all cases; the computation time required by the *IMqrginv* method is significantly less than the time required by the *qrginv* methods. On the other hand, we can see that the accuracy computation of the *IMqrginv* method is less than the *qrginv* method; however, in some cases, the accuracy of the results of both methods is low. We can conclude that *IMqrginv* method is a robust and efficient tool for obtaining the Moore-Penrose inverse of large sparse and rank deficient matrices.

4. Conclusion

In this paper we have presented a new method, called *IMqrginv*, for the fast computation of Moore-Penrose inverse

of singular square, rectangular, full, or sparse matrices. This method is based on the GSO method and Theorem 1. Invoking the *ginv* function we improved *qrginv* methods presented by Katsikis et al. [9]. We have compared the performance of the proposed method *IMqrginv* to the *qrginv* method. Numerical examples (see Tables 1–9 and 11) show that the proposed method is reasonably accurate, and its computation time is less than that of pseudoinverses obtained by the *qrginv*. Hence we conclude that the *IMqrginv* algorithm is a robust and efficient tool for computing the Moore-Penrose inverse of arbitrary matrices (including singular and rectangular).

Appendix

MATLAB Code of the “IMqrginv” Function

Function IMqrginv = IMqrginv(A):

- (1) $[m, n] = \text{size}(A)$,
- (2) $[Q, R, P] = \text{qr}(A)$,
- (3) $r = \text{sum}(\text{any}(\text{abs}(R) > 1e - 005, 2))$,
- (4) $Q = Q(:, 1 : r)$,
- (5) $R = R(1 : r, :)$,
- (6) $\text{IMqrginv} = P * (R^T / ((R * R^T)) * Q^T)$.

TABLE II: Error and computational time results for *magic* (rank = 3, cond = 5.3481e + 019).

Method	Method	Time	$\ AA^{\dagger}A - A\ _2$	$\ A^{\dagger}AA^{\dagger} - A^{\dagger}\ _2$	$\ (AA^{\dagger})^T - (AA^{\dagger})\ _2$	$\ (A^{\dagger}A)^T - (A^{\dagger}A)\ _2$
WELL1033_Z	qrginv	0.6277	2.1186e - 013	3.6375e - 011	1.4406e - 011	1.8819e - 013
	IMqrginv	0.3300	3.1283e - 014	2.5635e - 011	2.4875e - 012	9.1501e - 014
WELL1850_Z	qrginv	3.8271	1.0218e - 012	5.2163e - 011	5.6680e - 011	3.5487e - 013
	IMqrginv	1.7559	4.0066e - 014	6.3726e - 012	1.9053e - 012	7.7633e - 014
ILCC1033_Z	qrginv	0.6119	1.4956e - 010	1.1845e - 005	1.1674e - 006	6.9918e - 011
	IMqrginv	0.3786	2.3305e - 011	8.1774e - 006	1.5766e - 008	5.6012e - 010
ILCC1850_Z	qrginv	3.9197	1.9075e - 011	1.9254e - 008	9.4935e - 009	1.1185e - 011
	IMqrginv	1.6791	2.2511e - 013	9.5637e - 009	1.2945e - 010	6.6275e - 012
WATT1_Z	qrginv	10.5110	7.1357e - 007	1.8762e - 005	6.1068	7.4875e - 009
	IMqrginv	3.1018	7.1357e - 007	6.4114e - 006	6.1068	7.4875e - 009
GR-30-30_Z	qrginv	3.1095	2.3077e - 011	5.3574e - 011	3.5492e - 010	3.7356e - 013
	IMqrginv	1.6834	3.2708e - 013	1.2685e - 011	4.9654e - 012	2.3249e - 013
ADD20_Z	qrginv	60.9176	2.1032e - 008	4.9866e - 004	2.7022e - 004	1.4014e - 008
	IMqrginv	59.0933	9.3835e - 011	2.0676e - 005	8.6123e - 007	1.0353e - 009
NOS3_Z	qrginv	4.3836	1.6638e - 009	2.6309e - 009	5.4663e - 008	2.1929e - 011
	IMqrginv	3.0214	8.6964e - 011	8.4269e - 011	9.9464e - 010	2.1778e - 011
SHERMAN1_Z	qrginv	4.3150	1.0363e - 009	4.5703e - 007	7.4492e - 007	3.0278e - 010
	IMqrginv	1.3069	1.6398e - 012	3.2617e - 007	1.4722e - 009	5.6796e - 012

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The author deeply indebted to referees whose comments helped in improving the paper. Also, the author would like to thank Professor Davod Khojasteh Salkuyeh from Guilan University for his worth comments on this paper.

References

- [1] S. L. Campbell and C. D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Pitman (Advanced Publishing Program), Boston, Mass, USA, 1979, reprinted by Dover, 1991.
- [2] A. Ben-Israel and T. N. E. Greville, *Generalized Inverse: Theory and Applications*, Springer, New York, NY, USA, 2nd edition, 2003.
- [3] M. A. Rakha, "On the Moore-Penrose generalized inverse matrix," *Applied Mathematics and Computation*, vol. 158, no. 1, pp. 185–200, 2004.
- [4] P. Courrieu, "Fast computation of moore-penrose inverse matrices," *Nueural Information Processing-Letters and Review*, vol. 8, pp. 25–29, 2005.
- [5] W. Guo and T. Huang, "Method of elementary transformation to compute Moore-Penrose inverse," *Applied Mathematics and Computation*, vol. 216, no. 5, pp. 1614–1617, 2010.
- [6] S. Karanasios and D. Pappas, "Generalized inverses and special type operator algebras," *Facta Universitatis*, no. 21, pp. 41–48, 2006.
- [7] V. N. Katsikis and D. Pappas, "Fast computing of the Moore-Penrose inverse matrix," *Electronic Journal of Linear Algebra*, vol. 17, pp. 637–650, 2008.
- [8] F. Toutounian and A. Ataei, "A new method for computing Moore-Penrose inverse matrices," *Journal of Computational and Applied Mathematics*, vol. 228, no. 1, pp. 412–417, 2009.
- [9] V. N. Katsikis, D. Pappas, and A. Petralias, "An improved method for the computation of the Moore-Penrose inverse matrix," *Applied Mathematics and Computation*, vol. 217, no. 23, pp. 9828–9834, 2011.
- [10] C. C. MacDuffe, *The Theory of Matrices*, Chelsea Publishing, New York, NY, USA, 1959.
- [11] Matrix Market, *National Institute of Standards and Technology*, MatrixMarket, Gaithersburg, Md, USA, <http://math.nist.gov/>.

