Journal of Hyperstructures 6 *(1) (2017), 52-67.*
ISSN: 2322-1666 print/2251-8436 online

A FOURTH-ORDER ITERATIVE METHOD FOR COMPUTING THE MOORE-PENROSE INVERSE

HAMID ESMAEILI*, RAZIYEH ERFANIFAR AND MAHDIS RASHIDI

Abstract. In this study, a new fourth-order method to compute the Moore-Penrose inverse is proposed. Convergence analysis along with the error estimates of the method is investigated. Every iteration of the method involves four matrix multiplications. A wide set of numerical comparisons of the proposed method with nine higher order methods shows that the average number of matrix multiplications and the average CPU time of our method are considerably less than those of other methods.

Key Words: Moore-Penrose inverse; Iterative method; Schulz-type method; Fourth-order convergence; Matrix multiplication.

2010 Mathematics Subject Classification: Primary: 15A09; Secondary: 65F30, 65G50.

1. Introduction

Many higher order iterative methods have been developed to compute the Moore-Penrose inverse of a matrix. Iterative algorithms are a subject of current research (see, e.g., [10, 11, 17, 21, 24]), due to the importance of the topic in engineering and applied problems such as linear equations, statistical regression analysis, filtering, signal and image processing, and control of robot manipulators [5, 9, 14, 16].

In this article, we focus on presenting and demonstrating a new method as fast as possible with a close attention to reducing the computational time. To this end, we investigate a convergent iterative method to find the Moore-Penrose inverse, which could be viewed as an extension of

Received: 1 July 2016, Accepted: 02 February 2017. Communicated by Ali Taghavi;
∗Address correspondence to H. Esmaeili; Email: esmaeili@basu.ac.ir.

the famous Schulz method for such a purpose. It is proved that this method always converges with fourth-order, and every iteration involves four matrix multiplications. A theoretical discussion will also be given to show the behavior of proposed scheme.

In the simple case, when A is a $n \times n$ nonsingular complex matrix, to compute the matrix inverse, various iterative methods, called Schulz-type methods, already developed [1, 4, 7, 10, 12, 15, 18, 20, 23, 24], almost all of which are based on iterative solvers for the scalar equation $f(x) = \frac{1}{x} - a = 0$ applied to the matrix equation

$$f(X) = X^{-1} - A = 0.$$

We should also point out that even if the matrix A is singular, these methods converge to the Moore-Penrose inverse using a proper initial matrix. A full discussion on this feature of this type of iterative methods has been given in [1, 2]. So, upon this observation, we first assume that matrix A is a $n \times n$ nonsingular matrix.

The rest of this paper is organized as follows. Section 2 is devoted to presenting some existing iterative schemes to find matrix inverse. Also, we propose our new method in Section 2 and prove that it is fourth-order convergent. In Section 3, we show that one can use our method to find the Moore-Penrose inverse. In Section 4, some numerical examples are given to show the performance of the presented method compared with nine higher order methods. Finally, some conclusions are outlined in Section 5.

2. Schulz-type iterative methods

Suppose that A is a $n \times n$ nonsingular complex matrix. There are various iterative methods, called Schulz-type methods, to compute A^{-1}. In the sequel, we remind some of them.

Perhaps, the most frequently used iterative method to approximate A^{-1} is the famous Newton method

$$X_{k+1} = X_k(2I - AX_k), \quad k = 0, 1, 2, \dots, \tag{2.1}$$

originated in [15], in which I is the identity matrix with the same dimension as that of matrix A. Note that each iteration of (2.1) involves two matrix multiplications. Schulz in [15] found that the eigenvalues of $I - AX_0$ must have magnitudes less than 1 to ensure the convergence. Since the residuals $E_k = I - AX_k$ in each step (2.1) satisfy $\|E_{k+1}\| \leq \|A\| \, \|E_k\|^2$, Newton method is a second-order iterative method [1]. Similarly, in [10] the relation $\|A\varepsilon_{k+1}\| \leq \|A\varepsilon_k\|^2$ is verified for errors of the form $\varepsilon_k = X_k - A^{-1}$.

Li et al. in [8] investigated the following third-order method, known as Chebyshev method,

$$X_{k+1} = X_k(3I - AX_k(3I - AX_k)), \tag{2.2}$$

and also proposed another iterative method to find A^{-1} of the same order as given in

$$X_{k+1} = X_k(I + 0.5(I - AX_k)(I + (2I - AX_k)^2)). \tag{2.3}$$

We observe that each iteration of (2.2) and (2.3) contain 3 and 4 matrix multiplications, respectively. Toutounian and Soleymani [23] proposed the following fourth-order method that involves 5 matrix multiplications:

$$X_{k+1} = 0.5X_k(9I - AX_k(16I - AX_k(14I - AX_k(6I - AX_k)))). \tag{2.4}$$

Krishnamurthy and Sen [6] provided the following fourth-order method that contains 4 matrix multiplications:

$$\begin{aligned} Y_k &= I - AX_k, \\ X_{k+1} &= X_k(I + Y_k(I + Y_k(I + Y_k))). \end{aligned} \tag{2.5}$$

As another example, a ninth-order method, could be presented as

$$X_{k+1} = X_k(I+Y_k(I+Y_k(I+Y_k(I+Y_k(I+Y_k(I+Y_k(I+Y_k(I+Y_k)))))))).$$

The number of matrix multiplications of the above method can be reduced from 9 to 7 if it is rewritten as follows:

$$\begin{aligned} B_k &= Y_k^2, \\ C_k &= B_k^2, \\ D_k &= C_k^2, \\ X_{k+1} &= X_k((I + Y_k)(I + B_k)(I + C_k) + D_k). \end{aligned} \tag{2.6}$$

Soleymani et al. [19] provided the following sixth-order method that contains 5 matrix multiplications:

$$\begin{aligned} B_k &= AX_k \\ S_k &= B_k(-I + B_k) \\ X_{k+1} &= X_k(2I - B_k)(3I - 2B_k + S_k)(I + S_k). \end{aligned} \tag{2.7}$$

Soleymani and Stanimirović [18] investigated the following ninth-order method that has 7 matrix multiplications in each iteration:

$$\begin{aligned} B_k &= AX_k \\ S_k &= -7I + B_k(9I + B_k(-5I + B_k)) \\ T_k &= B_kS_k \\ X_{k+1} &= -0.125X_kS_k(12I + T_k(6I + T_k)). \end{aligned} \tag{2.8}$$

Also, Soleymani et al. [20] proposed another ninth-order method, again involving 7 matrix multiplications in each iteration, as

$$\begin{aligned} B_k &= AX_k \\ S_k &= 3I + B_k(-3I + B_k) \\ T_k &= B_kS_k \\ X_{k+1} &= -\tfrac{1}{9}\,X_kS_k(-29I + T_k(33I + T_k(-15I + 2T_k))). \end{aligned} \tag{2.9}$$

Remark 2.1. All of the above methods are initiated by

$$X_0 = \alpha A^*, \quad 0 < \alpha < \frac{2}{\sigma_1^2}, \tag{2.10}$$

in which σ_1 denote the largest singular value of A. There are another choices for X_0 too. A discussion on choosing the initial approximation X_0 is given in [2, 11]. Perhaps, in general case, the simplest choice for X_0 is

$$X_0 = \alpha A^*, \tag{2.11}$$

in which α is an appropriate constant.

Beside the above methods, we investigate our method to find Moore-Penrose inverse as follows:

$$X_{k+1} = X_k\left[9I - 26(AX_k) + 34(AX_k)^2 - 21(AX_k)^3 + 5(AX_k)^4\right].$$

We can rewrite our method in the way

$$\begin{aligned} B_k &= AX_k \\ C_k &= B_k^2 \\ X_{k+1} &= X_k\left[9I - 26B_k + C_k(34I - 21B_k + 5C_k)\right]. \end{aligned} \tag{2.12}$$

Note that every iteration of the method (2.12) involves four matrix multiplications. In the sequel, we prove that the method (2.12) is fourth-order convergent.

Let us consider the following singular value decomposition of the matrix A:

$$A = USV^*, \quad S = diag(\sigma_1, \ldots, \sigma_n), \quad \sigma_1 \geq \cdots \geq \sigma_n > 0, \tag{2.13}$$

where U and V are unitary matrices. Using

$$X_0 = \beta A^*, \tag{2.14}$$

in which β is a constant, we can deduce that each iterate of the method (2.12) has a singular value decomposition of the form

$$X_k = VS_kU^*, \quad S_k = diag(s_1^{(k)}, \ldots, s_n^{(k)}),$$

where

$$S_0 = \beta S,$$

and

$$S_{k+1} = S_k \left[9I - 26SS_k + 34(SS_k)^2 - 21(SS_k)^3 + 5(SS_k)^4\right]. \tag{2.15}$$

Therefore, the diagonal matrices $R_k = SS_k = diag(r_1^{(k)}, \ldots, r_n^{(k)})$ satisfy

$$R_{k+1} := g(R_k) = 9R_k - 26R_k^2 + 34R_k^3 - 21R_k^4 + 5R_k^5,$$

that means

$$r_i^{(k+1)} = g(r_i^{(k)}) = 9r_i^{(k)} - 26r_i^{(k)2} + 34r_i^{(k)3} - 21r_i^{(k)4} + 5r_i^{(k)5}. \tag{2.16}$$

In the following theorem, we show that the sequences (2.16) are fourth-order convergent to $r_i = 1$ for any $r_i^{(0)} \in (0, 1+\gamma)$, in which γ is a suitable constant.

Theorem 2.2. *For any initial point $r^{(0)} \in (0, 1+\gamma)$, the sequence $r^{(k+1)} = g(r^{(k)})$ is fourth-order convergent to $r = 1$, in which the function $g(r)$ is defined by (2.16) and $\gamma = 0.53$.*

Proof. We can find that the real fixed points and the critical points of $g(r)$ as follows:

$$\begin{aligned} g(r) = r \quad &\Longrightarrow \quad r = 0,\ 1,\ 1+\gamma, \\ g'(r) = 0 \quad &\Longrightarrow \quad r = 0.36\,,\ 1,\ 1,\ 1, \end{aligned}$$

in which

$$\gamma = \frac{1}{15}\left[1 + \sqrt[3]{316 + 30\sqrt{114}} - \frac{14}{\sqrt[3]{316 + 30\sqrt{114}}}\right] \approx 0.53.$$

Noting $g''(0.36) = -6.55 < 0$ and $g^{(4)}(1) = 96 > 0$, we can deduce that 0.36 is a local maximizer and 1 is a local minimizer of $g(r)$. On the other hand, $g(0) = 0 < 1 = g(1)$ and $g(0.36) \approx 1.13 < 1 + \gamma = g(1 + \gamma)$. Therefore, $\underline{r} = 0, 1$ and $\overline{r} = 0.36, 1 + \gamma$ are minimizer and maximizer of $g(r)$ in the interval $[0, 1 + \gamma]$, respectively. Moreover, the interval $[0, 1 + \gamma]$ maps into itself by the function $g(r)$.

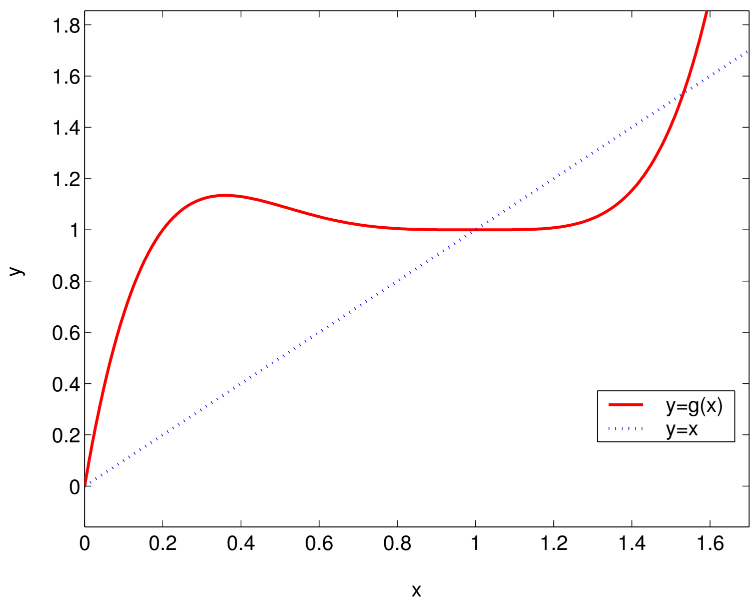

FIGURE 1. Graphs of the line $y = x$ and the function $y = g(x)$.

Considering an arbitrary initial point $r^{(0)} \in (0, 1 + \gamma)$, one can easily obtain the following considerations (For clarification, see Figure 1):

- The unique solution of the equation $g(r) = 1$ in the interval $[0, 1)$ is $\frac{1}{5}$.
- $g(r)$ is increasing in the interval $(0, \frac{1}{5})$. Therefore, if $r^{(k)} \in (0, \frac{1}{5})$, for some k, then there exists an index $k_0 \geq k$ such that either $r^{(k_0)} = \frac{1}{5}$, and so $r^{(k_0+1)} = 1$, or $r^{(k_0+1)} \in (\frac{1}{5}, 1)$.
- If $r^{(k)} \in (\frac{1}{5}, 1)$, for some k, then $r^{(k+1)} \in (1, 1 + \gamma)$.
- If $r^{(k)} \in (1, 1 + \gamma)$, for some k, then the sequence $\{r^{(k+\ell)}\}_{\ell \geq 1} \subseteq [1, 1 + \gamma)$ is a strictly decreasing sequence converging to $r = 1$.

Noting the above considerations, we can conclude that the sequence $r^{(k+1)} = g(r^{(k)})$ is convergent to $r = 1$. On the other hand,

$$g'(1) = g''(1) = g'''(1) = 0$$

implies that the convergence is fourth-order (See [3]). $\square$

Considering Theorem 2.2, we conclude that if $\beta\sigma_1^2 = r_1^{(0)} \in (0, 1.53)$, then $\beta\sigma_i^2 = r_i^{(0)} \in (0, 1.53)$, for all i, and

$$\lim_{k\to\infty} R_k = I.$$

Hence,

$$\lim_{k\to\infty} S_k = S^{-1},$$

so

$$\lim_{k\to\infty} X_k = VS^{-1}U^* = A^{-1}.$$

Moreover, the order of convergence is four. Therefore, the following theorem is proved.

Theorem 2.3. *Consider the nonsingular matrix A, and suppose that σ_1^2 denotes the largest singular value of A. Moreover, assume that the initial approximation X_0 is defined by (2.14), in which*

$$0 < \beta < \frac{1.53}{\sigma_1^2}\,. \tag{2.17}$$

Then, the sequence $\{X_k\}_{k\geq 0}$ generated by (2.12) converges to the inverse matrix A^{-1} with fourth-order.

Remark 2.4. Consider the initial matrix X_0 according to (2.14), with β from (2.17). Since σ_1^2 is a (the) largest singular value of A, we have $\sigma_1^2 = \|A\|_2^2 \leq \|A\|_1 \|A\|_\infty$. Therefore, the selection

$$\beta = \frac{1}{\|A\|_1 \|A\|_\infty} \tag{2.18}$$

satisfies both in (2.17) and (2.10).

3. Moore-Penrose inverse

The Moore-Penrose inverse of a $m \times n$ complex matrix A, denoted by $A^{\dagger}$, is a unique $n \times m$ matrix X satisfying the following four Penrose equations:

$$AXA = A, \quad XAX = X, \quad (AX)^* = AX, \quad (XA)^* = XA, \tag{3.1}$$

where A^* is the conjugate transpose of A.

Now, suppose that $rank(A) = r \leq \min\{m, n\}$ and consider the singular value decomposition of A as follows:

$$A = U \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} V^*, \quad S = diag(\sigma_1, \ldots, \sigma_r), \quad \sigma_1 \geq \cdots \geq \sigma_r > 0.$$

It is well known that

$$A^{\dagger} = V \begin{bmatrix} S^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^*.$$

Let us now extend the contributed method (2.12) for calculating the Moore-Penrose inverse $A^{\dagger}$. That is, we must analytically reveal that the sequence $\{X_k\}_{k\geq 0}$ generated by the iterative Schulz-type method (2.12), tends to the Moore-Penrose inverse as well.

Using mathematical induction, it would be easy to check that the iterates produced at each cycle of (2.12) satisfy the following relations:

$$\begin{aligned} (AX_k)^* &= AX_k, & (X_kA)^* &= X_kA, \\ A^{\dagger}AX_k &= X_k, & X_kAA^{\dagger} &= X_k. \end{aligned} \tag{3.2}$$

If we take $X_0 = \beta A^*$, which β is defined by (2.17), then

$$X_0 = \alpha A^* = V \begin{bmatrix} S_0 & 0 \\ 0 & 0 \end{bmatrix} U^*,$$

where

$$S_0 = \beta S.$$

is a diagonal matrix. Therefore,

$$V^* X_0 U = \begin{bmatrix} S_0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Now, the principle of mathematical induction and (3.2) lead to

$$V^* X_k U = \begin{bmatrix} S_k & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.3}$$

in which S_k is a diagonal matrix satisfying the following relation:

$$S_{k+1} = S_k \left[9I - 26SS_k + 34(SS_k)^2 - 21(SS_k)^3 + 5(SS_k)^4\right]. \tag{3.4}$$

This is the same as (2.15) and the proof of the Theorem 2.2 show that

$$\lim_{k\to\infty} S_k = S^{-1}.$$

Therefore,

$$\lim_{k\to\infty} X_k = V \begin{pmatrix} \mathrm{S}^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^* = A^\dagger.$$

Moreover, the order of convergence is four. Hence, we have the following theorem.

Theorem 3.1. *Let A be a $m \times n$ complex matrix of rank r and suppose that σ_1 is the largest singular value of A. Moreover, assume that the initial approximation X_0 is defined by (2.14), in which β is defined by (2.17). Then, the sequence $\{X_k\}_{k\geq 0}$ generated by (2.12) converges to the Moore-Penrose inverse $A^\dagger$ with fourth-order.*

Remark 3.2. If $m \leq n$, then we apply (2.12) in the same form, in which I denotes the $m\times m$ identity matrix. On the other hand, for case $m > n$ we must apply (2.12) with A^* instead of A and use the $n \times n$ identity matrix. So, for the case $m > n$, we compute $(A^*)^\dagger$, that is $(A^\dagger)^*$.

We can study the convergence properties of the algorithm (2.12) using the error matrix $E_k = X_k - A^\dagger$. The matrix formula representing E_{k+1} is a sum of possible zero-order term consisting of a matrix which does not depend upon E_k, one or more first-order matrix terms in which E_k or E_k^* appears only once, one or more second-order terms in which E_k and E_k^* appear at least twice, and so on [13]. To compute error estimates, first note that

$$A^\dagger A E_k = E_k, \quad E_k A A^\dagger = E_k$$

according to (3.2). Therefore,

$$\begin{aligned}
X_k(AX_k) &= A^\dagger + 2E_k + E_kAE_k,\\
X_k(AX_k)^2 &= A^\dagger + 3E_k + 3E_kAE_k + (E_kA)^2E_k,\\
X_k(AX_k)^3 &= A^\dagger + 4E_k + 6E_kAE_k + 4(E_kA)^2E_k + (E_kA)^3E_k,\\
X_k(AX_k)^4 &= A^\dagger + 5E_k + 10E_kAE_k + 10(E_kA)^2E_k + 5(E_kA)^3E_k\\
&\quad + (E_kA)^4E_k.
\end{aligned}$$

Now, substituting $X_k = A^\dagger + E_k$ in (2.12) results in

$$\begin{aligned} A^\dagger + E_{k+1} &= X_{k+1} \\ &= 9X_k - 26X_kAX_k + 34X_k(AX_k)^2 - 21X_k(AX_k)^3 \\ &\qquad + 5X_k(AX_k)^4 \\ &= A^\dagger + 7(E_kA)^3E_k + 8(E_kA)^4E_k, \end{aligned}$$

such that

$$E_{k+1} = 7(E_kA)^3E_k + 8(E_kA)^4E_k. \tag{3.5}$$

Hence, we have the following theorem.

Theorem 3.3. *Let A be a $m \times n$ nonzero complex matrix. If the initial approximation X_0 is defined by (2.14), with β from (2.17), then*

$$\|A(X_0 - A^\dagger)\| < 1,$$

and iterative method (2.12) converges to $A^\dagger$ with fourth-order. Its first, second, third, fourth and fifth order error terms are given by

$$\begin{aligned} error_1 &= error_2 = error_3 = 0, \\ error_4 &= 7(E_kA)^3E_k, \\ error_5 &= 8(E_kA)^4E_k. \end{aligned} \tag{3.6}$$

in which $E_k = X_k - A^\dagger$ denotes the error matrix.

Proof. Take $P = AA^\dagger$ and $S = I - AX_0$. Then, $P^2 = P$ and

$$\begin{aligned} PS &= AA^\dagger(I - AX_0) = AA^\dagger - AX_0 \\ &= AA^\dagger - AX_0AA^\dagger = (I - AX_0)AA^\dagger = SP. \end{aligned}$$

On the other hand, it is proved [22] that for $n \times n$ matrices M and N such that $M^2 = M$ and $MN = NM$, one has

$$\rho(MN) \le \rho(N).$$

Consequently, we attain

$$\begin{aligned} \rho(A(X_0 - A^\dagger)) &= \rho(A(\beta A^* - A^\dagger)) \\ &\le \rho(I - \beta AA^*) \\ &= \max_{1\le i\le r} |1 - \lambda_i(\beta AA^*)| \\ &= \max_{1\le i\le r} |1 - \beta\sigma_i^2|. \end{aligned}$$

By using (2.17), we conclude that

$$\|AE_0\| = \|A(X_0 - A^\dagger)\| \leq \rho(A(X_0 - A^\dagger)) \leq \max_{1\leq i\leq r} |1 - \beta\sigma_i^2| < 1.$$

Now, we can immediately derive (3.6) from (3.5). Furthermore, (3.5) results in

$$AE_{k+1} = 7(AE_k)^4 + 8(AE_kA)^5.$$

Hence,

$$\|AE_{k+1}\| \leq (7 + 8\|AE_k\|)\,\|AE_k\|^4,$$

and therefore $\|AE_k\| \to 0$, since $\|AE_0\| < 1$. On the other hand,

$$\|E_{k+1}\| = \|A^\dagger AE_{k+1}\| \leq \|A^\dagger\|\,\|AE_{k+1}\| \leq \|A^\dagger\|\,(7 + 8\|AE_k\|)\,\|AE_k\|^4$$

results in

$$\|E_{k+1}\| \leq \left[\|A^\dagger\|\,\|A\|^4\,(7 + 8\|A\|\,\|E_k\|)\right]\|E_k\|^4.$$

Consequently, $X_k \to A^\dagger$ and the order of convergence is four. $\square$

4. Numerical experiments

In this section, we will make some numerical comparisons of our proposed method (2.12) with other methods presented here. To this end, we focus on the total number of matrix multiplications and CPU times required for convergence. Table 1 denotes convergence order and number of matrix multiplications in any iteration of different methods.

Table 1. Convergence order and number of matrix multiplications for different methods

Method	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)	(2.7)	(2.8)	(2.9)	(2.12)
Convergence order	2	3	3	4	4	9	6	9	9	4
Matrix multiplications	2	3	4	5	4	7	5	7	7	4

We present three different types of tests. Test 1 is devoted to comparing the schemes to find the inverse of some randomly generated dense square matrices, Test 2 gives some comparison to find the Moore-Penrose inverse of dense matrices, and Test 3 gives some comparison to find the Moore-Penrose inverse of some randomly generated large sparse matrices. All tests were carried out with a Matlab code while the computer specifications are Microsoft Windows XP Intel(R), Pentium(R) 4, CPU

2.60 GHz, with 2 GB of RAM. We use the initial matrix X_0 defined in (2.14), with β from (2.17). The stop criterion is

$$\frac{\|X_{k+1} - X_k\|_\infty}{1 + \|X_k\|_\infty} < 10^{-7}$$

and the maximum number of iterations is set to 100 in our written codes as the maximum number of cycle for the methods is considered in comparisons.

Test 1. In this test, we compute the inverse of dense nonsingular square matrix A, where 10 matrices of the sizes $m \times m$, $m = 100, 200, 300, 400, 500$, are randomly generated as follows:

$$A = 100\, rand(m, m) - 10\, rand(m, m).$$

Average number of matrix multiplications and average of CPU times required for convergence are listed in Table 2. In this table, DIM, MAT and TIME denote the size of A, average values of matrix multiplications and elapsed times in seconds, respectively.

Table 2. Average values of matrix multiplications and elapsed times to compute the inverse of a dense nonsingular square matrix by different methods

Methods	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)	(2.7)	(2.8)	(2.9)	(2.12)
DIM:100 × 100										
MAT	63.2	61.5	73.2	77.5	66.0	76.3	65.5	72.8	74.9	46.8
TIME	0.05	0.04	0.07	0.05	0.05	0.08	0.05	0.05	0.05	0.04
DIM:200 × 200										
MAT	66.2	63.9	76.4	81.0	68.8	79.8	68.0	75.6	79.1	48.8
TIME	0.33	0.32	0.48	0.40	0.34	0.56	0.35	0.38	0.40	0.30
DIM:300 × 300										
MAT	70.0	67.50	81.20	85.0	72.80	84.0	72.0	79.8	82.6	51.2
TIME	1.22	1.17	1.76	1.50	1.25	2.06	1.28	1.39	1.44	1.13
DIM:400 × 400										
MAT	72.2	69.9	82.8	88.0	75.2	86.8	74.5	82.6	83.3	52.0
TIME	2.94	2.84	4.20	3.60	3.03	4.98	3.07	3.35	3.39	2.68
DIM:500 × 500										
MAT	73.0	70.8	84.0	88.5	76.4	87.5	75.5	84.0	85.4	53.2
TIME	5.77	5.60	8.30	7.04	5.99	9.79	6.04	6.63	6.76	5.33

From Table 2, we can see that our method (2.12) is more better than others both in matrix multiplications and CPU time. The worst one is fourth-order method (2.4). Note that the ninth-order method (2.8) acts like the third-order method (2.3). [Also, the sixth-order method (2.7) acts like the fourth-order method (2.5)]. The third-order method (2.2) and the second-order method (2.1) are better than other higher order methods, although they are not comparable with our method (2.12). Hence, we can consider the scheme (2.12) as the fastest method.

Test 2. In this test, we compute the Moore-Penrose inverse of randomly generated dense matrix A of the size $m \times n$, $n = m + 50$, as follows:

$$A = 100\, rand(m, n) - 10\, rand(m, n).$$

Again, for each $m = 100, 200, 300, 400, 500$, we have performed 10 tests and compared the average values of matrix multiplications and elapsed times in seconds. The results of comparisons are reported in Table 3, in terms of the number of matrix multiplications and the computational time (in seconds).

From Table 3, we can see that our method (2.12) is more better than others both in matrix multiplications and CPU time. The worst one is ninth-order method (2.6). Note that the ninth-order method (2.8) acts like the third-order method (2.3). [Also, the sixth-order method (2.7) acts like the fourth-order method (2.5)]. The third-order method (2.2) and the second-order method (2.1) are better than other higher order methods, although they are not comparable with our method (2.12). Hence, we can consider the scheme (2.12) as the fastest method.

Table 3. Average values of matrix multiplications and elapsed times to compute the Moore-Penrose inverse of a dense rectangular matrix by different methods

Methods	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)	(2.7)	(2.8)	(2.9)	(2.12)
DIM:100 $\times$ 150										
MAT	36.2	36.0	44.0	45.0	40.0	49.0	40.0	43.4	48.3	31.6
TIME	0.04	0.03	0.04	0.04	0.03	0.05	0.04	0.04	0.04	0.04
DIM:200 $\times$ 250										
MAT	41.6	41.4	48.0	51.0	44.0	54.6	45.0	49.0	49.7	32.0
TIME	0.26	0.24	0.33	0.28	0.24	0.40	0.26	0.27	0.27	0.22
DIM:300 $\times$ 350										
MAT	44.4	44.7	52.0	55.0	48.0	56.0	49.5	56.0	56.0	36.0
TIME	0.89	0.87	1.20	1.03	0.90	1.41	0.96	1.05	1.05	0.85
DIM:400 $\times$ 450										
MAT	46.4	45.0	56.0	60.0	48.8	56.0	50.0	56.0	56.0	36.0
TIME	2.12	1.98	2.99	2.57	2.09	3.29	2.21	2.39	2.40	1.95
DIM:500 $\times$ 550										
MAT	48.2	48.3	56.4	60.5	52.0	63.0	50.5	56.7	59.5	38.0
TIME	4.22	4.10	5.82	5.02	4.30	7.21	4.30	4.69	4.93	3.98

Test 3. This experiment evaluates the applicability of the new method (2.12) to find Moore-Penrose inverse of 10 random large sparse matrices of the size 1000×1500 containing approximately 6000 nonzero entries

as follows:

$$A = sprand(1000, 1500, 0.004).$$

The results of comparisons are reported in Table 4, in terms of number of matrix multiplications and computational time (in seconds).

Table 4. Average values of matrix multiplications and elapsed times to compute the Moore-Penrose inverse of a 1000×1500 matrix by different methods

Methods	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)	(2.7)	(2.8)	(2.9)	(2.12)
MAT	47.4	46.5	56.0	58.5	50.8	60.9	50.5	55.3	58.8	36.8
TIME	85.35	91.38	115.64	120.01	103.33	129.42	113.30	123.83	133.64	74.36

From Table 4, we can see that our method (2.12) is more better than others both in matrix multiplications and CPU time. The worst one is ninth-order method (2.6). Note that the ninth-order method (2.8) acts like the third-order method (2.3). [Also, the sixth-order method (2.7) acts like the fourth-order method (2.5)]. The third-order method (2.2) and the second-order method (2.1) are better than other higher order methods, although they are not comparable with our method (2.12). Hence, we can consider the scheme (2.12) as the fastest method.

5. Conclusions

In this paper, we proposed a new method to find the Moore-Penrose inverse. It is proved that this method is fourth-order convergent. A wide set of random numerical experiments showed that its number of matrix multiplications and CPU time are considerably less than those of other higher methods. So, our method could be considered as a fast method.

Acknowledgments

The authors would like to thank the referees for their useful comments and constructive suggestions which substantially improved the quality of this paper.

References

[1] A. Ben-Israel, D. Cohen, *On iterative computation of generalized inverses and associated projections*, SIAM J. Numer. Anal. **3** (1966), 410–419.

[2] A. Ben-Israel, T.N.E. Greville, *Generalized Inverses*, Second ed., New York: Springer (2003).

[3] R.L. Burden, J.D. Faires, *Numerical Analysis*, 9th Ed. Brooks/Cole, Cengage Learning, Boston (2011).

[4] H. Chen, Y. Wang, *A family of higher-order convergent iterative methods for computing the Moore-Penrose inverse*, Appl. Math. Comput. **218** (2011), 4012–4016.

[5] A. Cichocki, B. Unbehauen, *Neural networks for optimization and signal processing*, New York: John Wiley & Sons (1993).

[6] E.V. Krishnamurthy, S.K. Sen, *Numerical Algorithms: Computations in Science and Engineering*, New Delhi, India: Affiliated East-West Press (1986).

[7] W. Li, Z. Li, *A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix*, Appl. Math. Comput. **215** (2010), 3433–3442.

[8] H. B. Li, T.Z. Huang, Y. Zhang, X.P. Liu, T.X. Gu, *Chebyshev-type methods and preconditioning techniques*, Appl. Math. Comput. **218** (2001), 260–270.

[9] S. Miljković, M. Miladinović, P., Stanimirović, I. Stojanović, *Application of the pseudo-inverse computation in reconstruction of blurred images*, Filomat **26** (2012), 453–465.

[10] H.S. Najafi, M.S. Solary, *Computational algorithms for computing the inverse of a square matrix, quasi-inverse of a non-square matrix and block matrices*, Appl. Math. Comput. **183** (2006), 539–550.

[11] V.Y. Pan, R. Schreiber, *An improved Newton iteration for the generalized inverse of a matrix with applications*, SIAM J. Sci. Stat. Comput. **12** (1991), 1109–1131.

[12] V.Y. Pan, *Newton's iteration for matrix inversion, advances and extensions, matrix methods: theory algorithms and applications*, Singapore: World Scientific (2010).

[13] W.H. Pierce, *A self-correcting matrix iteration for the Moore-Penrose inverse*, Linear Algebra Appl. **244** (1996), 357-363.

[14] P. Roland, P.G. Beim, *Inverse problems in neural field theory*, SIAM J. Appl. Dynam. Sys. **8** (2009), 1405–1433.

[15] G. Schulz, *Iterative Berechmmg der reziproken Matrix*, Z. Angew. Math. Mech. **13** (1933), 57–59.

[16] L. Sciavicco, B. Siciliano, *Modelling and control of robot manipulators*, London: Springer–Verlag (2000).

[17] X. Sheng, G. Chen: *The generalized weighted Moore-Penrose inverse*, J. Appl. Math. Comput. **25** (2007), 407–413.

[18] F. Soleymani, P.S. Stanimirović, *A Higher Order Iterative Method for Computing the Drazin Inverse*, The Scientific World Journal Volume 2013, Article ID 708647, 11 pages, http://dx.doi.org/10.1155/2013/708647.

[19] F. Soleymani, P.S. Stanimirović, M.Z. Ullah, *An accelerated iterative method for computing weighted Moore-Penrose inverse*, Appl. Math. Comput. **222** (2013), 365–371.

[20] F. Soleymani, H. Salmani, M. Rasouli, *Finding the Moore-Penrose inverse by a new matrix iteration*, J. Appl. Math. Comput. **47** (2015), 33-48.

[21] S. Srivastava, D.K. Gupta, *A higher order iterative method for $A^{(2)}_{T,S}$*, J. Appl. Math. Comput. **46** (2014), 147–168.

[22] P.S. Stanimirović, D.S. Cvetković-Ilić, *Successive matrix squaring algorithm for computing outer inverses*, Appl. Math. Comput. **203** (2008), 19–29.

[23] F. Toutounian, F. Soleymani, *An iterative method for computing the approximate inverse of a square matrix and the Moore-Penrose inverse of a non-square matrix*, Appl. Math. Comput. **224** (2013), 671–680.

[24] L. Weiguo, L. Juan, Q. Tiantian, *A family of iterative methods for computing Moore-Penrose inverse of a matrix*, Linear Algebra Appl. **438** (2013), 47-56.

H. Esmaeili
Department of Mathematics, Bu-Ali Sina University, Hamedan, Iran
`Email: esmaeili@basu.ac.ir`

R. Erfanifar
Department of Mathematics, Bu-Ali Sina University, Hamedan, Iran
`Email: rerfani@basu.ac.ir`

M. Rashidi
Department of Mathematics, Bu-Ali Sina University, Hamedan, Iran
`Email: m.rashidi94@basu.ac.ir`