Available online at http://ijim.srbiau.ac.ir/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol. 10, No. 2, 2018 Article ID IJIM-00835, 9 pages
Research Article

An Efficient Schulz-type Method to Compute the Moore-Penrose Inverse

H. Esmaeili *†, R. Erfanifar ‡, M. Rashidi §

Received Date: 2016-08-29 Revised Date: 2017-09-27 Accepted Date: 2018-05-17

Abstract

A new Schulz-type method to compute the Moore-Penrose inverse of a matrix is proposed. Every iteration of the method involves four matrix multiplications. It is proved that this method always converge with fourth-order. A wide set of numerical comparisons shows that the average number of matrix multiplications and the average CPU time of our method are considerably less than those of other methods. For each of sizes $n \times n$ and $n \times (n+10)$, $n = 100, 200, 300, 400$, ten random matrices were chosen to make these comparisons.

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

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., [11, 12, 18, 23, 25]), 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, 10, 15, 17].

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

In the simple case, when A is a $n \times n$ nonsingular matrix, to compute the matrix inverse, various iterative methods, called Schulz-type methods, were developed [1, 4, 6, 9, 11, 13, 16, 19, 22, 24, 25], 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

*Corresponding author. esmaeili@basu.ac.ir, Tel: +(98)9121028244
†Department of Mathematics, Bu-Ali Sina University, Hamedan, Iran.
‡Department of Mathematics, Malayer University, Malayer, Iran.
§Department of Mathematics, Bu-Ali Sina University, Hamedan, Iran.

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].

The rest of this paper is organized as follows. Section 2 is devoted to presenting some existing iterative schemes to find the Moore-Penrose inverse. We propose our new method in Section 3 and prove that it is fourth-order convergent. In Section 4, some numerical examples are given to show the performance of the presented method compared with other higher order methods. For each of sizes $n \times n$ and $n \times (n+10)$, $n = 100, 200, 300, 400$, ten random matrices were chosen to make these comparisons. Finally, some conclusions are outlined in Section 5.

2 Schulz-type iterative methods

The MoorePenrose inverse of a matrix $A \in \mathbb{C}^{m\times n}$, denoted by $A^\dagger \in \mathbb{C}^{n\times m}$, is a unique matrix X satisfying the following four Penrose equations

$$AXA = A, \quad (AX)^* = AX,$$
$$XAX = X, \quad (XA)^* = XA,$$

where A^* is the conjugate transpose of A. There are various iterative methods, called Schulz-type methods, to compute $A^\dagger$. In the sequel, we recall some of them.

Perhaps, the most frequently used iterative method to approximate $A^\dagger$ is the famous Newton method

$$\begin{aligned} B_k &= AX_k \\ X_{k+1} &= X_k(2I - B_k), \end{aligned} \tag{2.1}$$

originated in [16], in which I is the $m\times m$ identity matrix. Schulz in [16] found that the eigenvalues of $I - AX_0$ must have magnitudes less than 1 to ensure the convergence. Since the residuals $R_k = I - AX_k$ in each step (2.1) satisfy $\|R_{k+1}\| \leq \|A\|\,\|R_k\|^2$, the Newton method is a second-order iterative method [1]. Similarly, in [11] the relation $\|AE_{k+1}\| \leq \|AE_k\|^2$ is verified for errors of the form $E_k = X_k - A^\dagger$.

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

$$\begin{aligned} B_k &= AX_k \\ X_{k+1} &= X_k(3I - B_k(3I - B_k)), \end{aligned} \tag{2.2}$$

and also proposed another iterative method to find $A^\dagger$ of the same order as given in

$$\begin{aligned} B_k &= AX_k \\ X_{k+1} &= X_k[I + 0.5(I - B_k) \\ &\qquad (I + (2I - B_k)^2)]. \end{aligned} \tag{2.3}$$

Toutounian and Soleymani [24] proposed the following fourth-order method:

$$\begin{aligned} B_k &= AX_k \\ X_{k+1} &= 0.5X_k[9I - B_k(16I - \\ &\qquad B_k(14I - B_k(6I - B_k)))]. \end{aligned} \tag{2.4}$$

Krishnamurthy and Sen [7] provided the following fourth-order method:

$$X_{k+1} = X_k(I + Y_k(I + Y_k(I + Y_k))), \tag{2.5}$$

in which $Y_k = I - AX_k$. As another example, a ninth-order method could be presented as

$$\begin{aligned} X_{k+1} = X_k[I + Y_k(I + Y_k(I + Y_k(I + \\ Y_k(I + Y_k(I + Y_k(I + Y_k)))))))]. \end{aligned}$$

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

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

Soleymani et al. [21] provided the following sixth-order method:

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

Soleymani and Stanimirović [19] investigated the following ninth-order method:

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

Also, Soleymani et al. [22] proposed another ninth-order method as

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

Recently, Esmaeili and Pirnia [6] investigate a quadratically convergent method as follows:

$$\begin{aligned} B_k &= AX_k \\ X_{k+1} &= X_k(5.5I - B_k(8I - 3.5B_k)). \end{aligned} \tag{2.10}$$

Although their method is not a higher method, numerical experiments showed that (2.10) is very effective than other methods both in number of matrix multiplications and CPU time.

To start each of methods, we need an initial matrix X_0. A discussion on choosing the initial approximation X_0 is given in [2, 14]. Perhaps, in general, the simplest choice for X_0 is

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

in which β is a suitable real number.

3 The New Method

In this paper, we would like to propose a fourth-order class of Schulz-type methods to find $A^\dagger$ such that is more effective than all of above methods in terms of number of matrix multiplications and CPU time. To this end, we consider the following iterative class of methods:

$$\begin{aligned} B_k &= AX_k \\ X_{k+1} &= X_k \left(aI + bB_k + cB_k^2 + dB_k^3 + eB_k^4\right), \end{aligned} \tag{3.12}$$

in which a, b, c, d, e are parameters. Note that every iteration of the method (3.12) involves four matrix multiplications. In the sequel, we prove that the method (3.12) is fourth-order convergent to $A^\dagger$ for appropriate chooses of parameters.

Note that, using mathematical induction, it would be easy to check that the iterates produced at each cycle of (3.12) satisfy the following relations:

$$\begin{aligned} (AX_k)^* &= AX_k, \quad A^\dagger AX_k = X_k, \\ (X_kA)^* &= X_kA, \quad X_kAA^\dagger = X_k. \end{aligned} \tag{3.13}$$

We can study the convergence properties of the algorithm (3.12) 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 [15]. To compute error estimates, first note that

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

according to (3.13). Therefore,

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

Now, substituting $X_k = A^\dagger + E_k$ in (3.12), we have

$$\begin{aligned} A^\dagger + E_{k+1} = &(a+b+c+d+e)A^\dagger \\ &+(a+2b+3c+4d+5e)E_k \\ &+(b+3c+6d+10e)E_kAE_k \\ &+(c+4d+10e)(E_kA)^2E_k \\ &+(d+5e)(E_kA)^3E_k \\ &+e(E_kA)^4E_k. \end{aligned}$$

Fourth-order convergent is obtained when

$$\begin{cases} a+b+c+d+e = 1 \\ a+2b+3c+4d+5e = 0 \\ b+3c+6d+10e = 0 \\ c+4d+10e = 0, \end{cases}$$

that result in

$$\begin{aligned} a &= 4+e, \quad b = -(6+4e), \\ c &= 4+6e, \quad d = -(1+4e) \end{aligned}.$$

So, we have

$$E_{k+1} = (e-1)(E_kA)^3E_k + e(E_kA)^4E_k. \tag{3.14}$$

Hence, the following theorem can be obtained.

Theorem 3.1 *Let A be a $m \times n$ nonzero complex matrix. Moreover, suppose that the initial approximation X_0 is defined by (2.11). If the real number β is chose such that*

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

then the iterative method (3.12) converges to $A^\dagger$ with fourth-order. Its first, second, third, fourth and fifth order error terms are given by

$$\begin{aligned} error_1 &= error_2 = error_3 = 0, \\ error_4 &= (e-1)(E_kA)^3E_k, \\ error_5 &= e(E_kA)^4E_k. \end{aligned} \tag{3.15}$$

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

Proof. We can immediately derive (3.15) from (3.14). Furthermore, (3.14) results in

$$AE_{k+1} = (e-1)(AE_k)^4 + e(AE_k)^5.$$

Hence,

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

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

$$\begin{aligned} \|E_{k+1}\| &= \|A^\dagger AE_{k+1}\| \leq \|A^\dagger\|\,\|AE_{k+1}\| \\ &\leq \|A^\dagger\|(e-1+e\|AE_k\|)\,\|AE_k\|^4 \end{aligned}$$

results in

$$\begin{aligned} &\|E_{k+1}\| \leq \\ &\left[\|A^\dagger\|\,\|A\|^4(e-1+e\|A\|\,\|E_k\|)\right]\|E_k\|^4. \end{aligned}$$

Consequently, $X_k \to A^\dagger$ and the order of convergence is four. □ Now, suppose that $rank(A) = r \leq \min\{m,n\}$ and consider the singular value decomposition of A as follows:

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

It is well known that

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

If we take X_0 as (2.11), then

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

where

$$S_0 = \beta S$$

is a diagonal matrix. Therefore,

$$V^*X_0U = \begin{bmatrix} S_0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Now, the principle of mathematical induction and (3.13) lead to

$$V^*X_kU = \begin{bmatrix} S_k & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.16}$$

in which $S_k = diag(s_1^{(k)},\ldots,s_r^{(k)})$ is a diagonal matrix satisfying the following relation:

$$\begin{aligned} S_{k+1} = S_k[&(4+e)I - (6+4e)SS_k \\ &+(4+6e)(SS_k)^2 - (1+4e)(SS_k)^3 \\ &+e(SS_k)^4]. \end{aligned} \tag{3.17}$$

Therefore, the diagonal matrices $D_k := SS_k = diag(d_1^{(k)},\ldots,d_r^{(k)})$, where $d_i^{(k)} = \sigma_i s_i^{(k)}$, satisfy

$$\begin{aligned} D_{k+1} := g(D_k) = &(4+e)D_k - (6+4e)D_k^2 \\ &+(4+6e)D_k^3 - (1+4e)D_k^4 + eD_k^5, \end{aligned}$$

that means

$$\begin{aligned} d_i^{(k+1)} = g(d_i^{(k)}) = &(4+e)d_i^{(k)} \\ &-(6+4e)d_i^{(k)2} + (4+6e)d_i^{(k)3} \\ &-(1+4e)d_i^{(k)4} + ed_i^{(k)5}. \end{aligned} \tag{3.18}$$

In the following theorem, we show that, for $e = 8$, the sequences (3.18) are fourth-order convergent to $d_i = 1$ for any $d_i^{(0)} \in (0, 1.45)$.

Theorem 3.2 *For any initial point $d^{(0)} \in (0, 1.45)$, the sequence $d^{(k+1)} = g(d^{(k)})$ is fourth-order convergent to $d = 1$, in which the function $g(x)$ is defined by*

$$g(x) = 12x - 38x^2 + 52x^3 - 33x^4 + 8x^5. \tag{3.19}$$

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

$$\begin{aligned} g(x) = x \quad &\Longrightarrow \quad x = 0,\ 1,\ 1.45, \\ g'(x) = 0 \quad &\Longrightarrow \quad x = 0.3,\ 1,\ 1,\ 1. \end{aligned}$$

Noting $g''(0.3) < 0$ and $g^{(4)}(1) > 0$, we can deduce that 0.3 is a local maximizer and 1 is a local minimizer of $g(x)$. On the other hand, $g(0) = 0 < 1 = g(1)$ and $g(0.3) \approx 1.34 < 1.45 = g(1.45)$. Therefore, $\underline{x} = 0, 1$ and $\overline{x} = 0.3, 1.45$ are minimizers and maximizers of $g(x)$ in the interval $[0, 1.45]$, respectively. Moreover, the interval $[0, 1.45]$ maps into itself by the function $g(x)$. Considering an arbitrary initial point $d^{(0)} \in (0, 1.45)$, one can easily obtain the following considerations (For clarification, see Figure 1):

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)	(2.10)	(3.20)
Convergence order	2	3	3	4	4	9	6	9	9	2	4
Matrix multiplications	2	3	4	5	4	7	5	7	7	3	4

Table 2: Average values of matrix multiplications and elapsed times for different methods

Methods	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)	(2.7)	(2.8)	(2.9)	(2.10)	(3.20)
DIM:100 × 100											
MAT	59.8	57.9	70.4	74.0	63.2	72.8	63.0	69.3	70.7	46.7	43.6
TIME	0.047	0.038	0.060	0.050	0.047	0.072	0.040	0.050	0.053	0.034	0.036
DIM:100 × 110											
MAT	44.6	43.8	52.4	55.5	47.6	56.0	48.0	55.3	56.0	39.6	35.6
TIME	0.037	0.033	0.0547	0.036	0.038	0.053	0.039	0.039	0.041	0.032	0.031
DIM:200 × 200											
MAT	65.2	63.6	76.0	80.5	68.4	79.8	67.5	74.9	77.7	51.4	46.8
TIME	0.319	0.316	0.464	0.400	0.330	0.560	0.338	0.367	0.386	0.260	0.292
DIM:200 × 210											
MAT	50.8	49.2	60.0	62.0	53.2	63.0	54.5	58.8	63.0	42.7	37.6
TIME	0.270	0.248	0.380	0.314	0.264	0.445	0.284	0.297	0.325	0.239	0.234
DIM:300 × 300											
MAT	70.0	76.8	81.2	85.5	73.2	84.7	72.0	79.1	82.6	51.9	49.2
TIME	1.225	1.184	1.764	1.505	1.263	2.078	1.281	1.380	1.444	0.916	1.086
DIM:300 × 310											
MAT	53.2	51.6	62.0	65.5	56.4	64.4	55.5	63.0	63.7	44.5	40.0
TIME	0.956	0.917	1.363	1.166	0.988	1.589	1.002	1.114	1.116	0.983	0.899
DIM:400 × 400											
MAT	73.6	71.7	84.8	89.5	76.8	88.9	75.5	83.3	86.1	53.3	51.6
TIME	2.991	2.916	4.306	3.675	3.100	5.121	3.119	3.386	3.505	2.189	2.667
DIM:400 × 410											
MAT	56.6	55.8	66.4	70.0	60.0	70.0	60.0	65.1	70.0	46.9	40.4
TIME	2.364	2.308	3.406	2.892	2.455	4.034	2.511	2.675	2.883	2.012	2.111

- The unique solution of the equation $g(x) = 1$ in the interval $[0, 1)$ is $\frac{1}{8}$.
- The function $g(x)$ is increasing in the interval $(0, \frac{1}{8})$. Therefore, if $d^{(k)} \in (0, \frac{1}{8})$, for some k, then there exists an index $k_0 \geq k$ such that either $d^{(k_0)} = \frac{1}{8}$, and so $d^{(k_0+1)} = 1$, or $d^{(k_0+1)} \in (\frac{1}{8}, 1)$.
- If $d^{(k)} \in (\frac{1}{8}, 1)$, for some k, then $d^{(k+1)} \in (1, 1.45)$.
- If $d^{(k)} \in (1, 1.45)$, for some k, then the sequence $\{d^{(k+\ell)}\}_{\ell \geq 1} \subseteq [1, 1.45)$ is a strictly decreasing sequence converging to $d = 1$.

Noting the above assertions, we can conclude that the sequence $d^{(k+1)} = g(d^{(k)})$ is convergent to $d = 1$. On the other hand, $g'(1) = g''(1) = g'''(1) = 0$ implies that the convergence is fourth-order (See [3]). □

Using the iteration function (3.19), we obtain the following iterative method to find $A^{\dagger}$:

$$X_{k+1} = X_k[12I - 38(AX_k) + 52(AX_k)^2 - 33(AX_k)^3 + 8(AX_k)^4],$$

which can be written as follows:

$$\begin{aligned} B_k &= AX_k \\ C_k &= B_k^2 \\ X_{k+1} &= X_k[12I - 38B_k \\ &\quad + C_k(52I - 33B_k + 8C_k)]. \end{aligned} \tag{3.20}$$

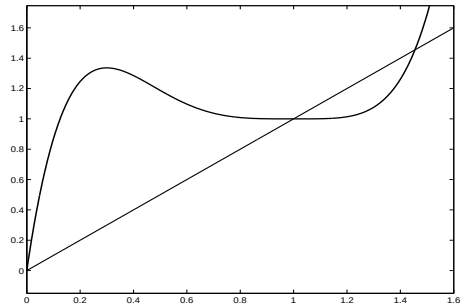


Figure 1: Graphs of the line $y = x$ and the function $y = g(x)$.

Considering Theorem 3.2, we conclude that if $\beta\sigma_1^2 = d_1^{(0)} \in (0, 1.45)$, then $\beta\sigma_i^2 = d_i^{(0)} \in (0, 1.45)$, for all i, and

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

Hence,

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

so

$$\lim_{k\to\infty} X_k = A^{\dagger}.$$

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

Theorem 3.3 *Consider the $m \times n$ complex matrix A of rank r, and suppose that σ_1^2 denotes the largest singular value of A. Moreover, assume that the initial approximation X_0 is defined by (2.11), in which*

$$0 < \beta < \frac{1.45}{\sigma_1^2}. \tag{3.21}$$

Then, the sequence $\{X_k\}_{k\geq 0}$ generated by (3.20) converges to $A^{\dagger}$ with fourth-order.

Remark 3.1 Consider the initial matrix X_0 given in (2.11), with β from (3.21). 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

$$\beta = \frac{1}{\|A\|_1 \|A\|_\infty} \tag{3.22}$$

satisfies (3.21). Furthermore, it is proved in [4] that for such a β, we have $\|A(X_0 - A^{\dagger})\| < 1$.

Note that every iteration of the method (3.20) involves four matrix multiplications. Although (3.20) is not a higher order scheme, the numerical experiments show that its total number of matrix multiplications and its CPU time are considerably less than those of other methods. So, method (3.20) will be the fastest one among considered iterative methods in this article.

The Schulz-type iterations, including (3.20), are strongly numerically stable, that is, they have the self-correcting characteristic and are essentially based upon matrix multiplication per an iterative step. The iterative scheme (3.20) could be combined efficiently with sparse techniques in order to reduce the computational load of matrix multiplications per step.

Remark 3.2 If $m \leq n$, then we apply (3.20) in the same form, in which I denotes the $m \times m$ identity matrix. On the other hand, for $m > n$ we must apply (3.20) 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 $(A^{\dagger})^*$.

Theorem 3.4 *If the same assumptions as in Theorem 3.3 are considered, then the use of the iterative method (3.20) for finding the Moore-Penrose generalized inverse has an asymptotical stability.*

Proof. The steps of proving the asymptotic stability of (3.20) are similar to those taken for a general family of methods in [20]. Hence, the proof is omitted. □

4 Numerical experiments

In this section, we will make some numerical comparisons of our proposed method (3.20) 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 the number of matrix multiplications in any iteration of different methods.

All tests were carried out with Matlab, while the computer specifications are Microsoft Windows XP Intel(R), Pentium(R) 4, CPU 2.60 GHz, with 2 GB of RAM.

We used the initial matrix X_0 defined in (2.11), with β from (3.21). The stop criterion was

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

and the maximum number of iterations was set to 100. Following [6, 21, 24], for each of sizes $n \times n$ and $n \times (n+10)$, $n = 100, 200, 300, 400$, we have performed 10 random tests and compared average values of matrix multiplications and elapsed times in seconds. The results are listed in Table 2, where DIM, MAT, and TIME denote the size of A, average values of matrix multiplications and elapsed times in seconds, respectively.

From Table 2, we observe that the method (3.20) is much better than others both in matrix multiplications and CPU time. The worst one is the ninth-order method (2.4). The third-order method (2.2) and the second-order method (2.1) are better than the higher order methods, although they are not comparable with our method. We can almost arrange these methods in the form

$$\{(2.4),(2.6)\} < \{(2.3),(2.8),(2.9)\} < \{(2.5),(2.7)\} < \{(2.1)\} < \{(2.2)\} \ll \{(2.10)\} < \{(3.20)\},$$

meaning that methods belonging to a set have a similar efficiency, while "<" denotes less efficiency.

5 Conclusions

In this paper, we proposed a new Schulz-type method to find the Moore-Penrose inverse. It was proved that the method converge with fourth-order. Although our method is not a higher order scheme, a wide set of random numerical experiments showed that the required number of matrix multiplications and CPU time is considerably less than those of higher order methods. So, our method could be considered as a fast method.

References

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

[2] A. Ben-Israel, T. N. E. Greville, Generalized Inverses, 2nd ed., CMS Books Math./OuvragesMath.SMC 15, Springer, *New York*, (2003).

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

[4] H. Chen, Y. Wang, A family of higher-order convergent iterative methods for computing the MoorePenrose 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] H. Esmaeili, A. Pirnia, An efficient quadratically convergent iterative method to find the Moore-Penrose inverse, *Int. J. Comp. Math.* 94 (2017) 1079-1088.

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

[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] 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.

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

[11] 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.

[12] 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.

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

[14] V. Y. Pan, Structured Matrices and Polynomials, Unified Superfast Algorithms, Birkhauser-Springer, *New York*, (2001).

[15] P. Roland, P. B. Graben, Inverse problems in neural field theory, *SIAM J. Appl. Dynam. Sys.* 8 (2009) 1405-1433.

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

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

[18] X. Sheng, G. Chen, The generalized weighted MoorePenrose inverse, *J. Appl. Math. Comput.* 25 (2007) 407-413.

[19] 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/.

[20] F. Soleymani, P. S. Stanimirović, A note on the stability of a p-th order iteration for finding generalized inverses, *Appl. Math. Lett.* 28 (2014) 77-81.

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

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

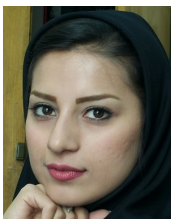
[23] S. Srivastava, D.K. Gupta, A higher order iterative method for $A^{(2)}_{T,S}$, *Appl. Math. Comput.*, (2013) http://dx.doi.org/10.1007/s12190-013-0743-4/.

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

[25] L. Weiguo, L. Juan, Q. Tiantian, A family of iterative methods for computingMoorePenrose inverse of a matrix, *Linear Algebr. Appl.* 438 (2013) 47-56.

Hamid Esmaeili is Associate Professor of applied mathematics and Faculty of Sciences, Bu-Ali Sina University, Hamedan, Iran. His research interests include Optimizatoin, Numerical Analysis, and Numerical Linear Algebra. He has published more than 30 papers in international journals and conferences.

Raziyeh Erfanifar is now PhD student of applied mathematics, Malayer University, Malayer, Iran. Her research interests include fractional calculus, numerical analysis, and numerical linear algebra. He has published some papers in international journals and conferences.



Mahdis Rashidi is graduate MS student of Bu-Ali Sina University, Hamedan, Iran. Her research interests include numerical linear algebra and Matrix Equations.