mathematics

MDPI

Article

Calculating the Weighted Moore–Penrose Inverse by a High Order Iteration Scheme

Haifa Bin Jebreen

Mathematics Department, College of Science, King Saud University, Riyadh 11451, Saudi Arabia; hjebreen@ksu.edu.sa

Received: 16 July 2019; Accepted: 7 August 2019; Published: 10 August 2019

Abstract: The goal of this research is to extend and investigate an improved approach for calculating the weighted Moore–Penrose (WMP) inverses of singular or rectangular matrices. The scheme is constructed based on a hyperpower method of order ten. It is shown that the improved scheme converges with this rate using only six matrix products per cycle. Several tests are conducted to reveal the applicability and efficiency of the discussed method, in contrast with its well-known competitors.

Keywords: iteration scheme; Moore–Penrose; rectangular matrices; rate of convergence; efficiency index

MSC: 15A09; 65F30

1. Introduction

1.1. Background

Constructing and discussing different features of iterative schemes for the calculation of outer inverses is an active topic of current research in Applied Mathematics (for more details, refer to [1–3]). Many papers have been published in the field of outer inverses over the past few decades, each having their own domain of validity and usefulness. In fact, in 1920, Moore was a pioneer of this field and published seminal works about the outer inverse [4,5]. However, several deep works were published during the 1950s (as reviewed and observed in [4]). It is also noted that pseudo-inverse operator was first introduced by Fredholm in [6].

The method of partitioning (due to Greville) was a pioneering work in computing generalized inverses, which was re-introduced and re-investigated in [4,7]. This scheme requires a lot of operations and is subject to cancelation and rounding errors. Among the generalized inverses, the weighted Moore–Penrose (WMP) inverse is important, as it can be simplified to a pseudo-inverse, as well as a regular inverse. Several applications of computing the WMP inverse can be observed, with some discussion, in the recent literature [8,9]; including applications to the solution of matrix equations. See [10–13] for further discussions and applications.

Furthermore, for large matrices, or as long as the weight matrices in the process of computing the WMP inverse are ill-conditioned, symbolic computation of the current algorithms may not properly work due to several reasons, such as time consumption, requiring higher memory space, or instability. On the other hand, several numerical methods for the weighted Moore–Penrose (WMP) inverse are not stabie or possess slow convergence rates. Hence, it is necessary to investigate and extend novel and useful iterative matrix methods for such an objective; see, also, the discussions in [14,15].

1.2. Definition

Let us consider that M and N are two square Hermitian positive definite (HPD) matrices of sizes m and n ($m \leq n$) and $A \in \mathbb{C}^{m\times n}$. Then, there is a unique matrix X satisfying the following identities [16]:

1. $AXA = A$,
2. $XAX = X$,
3. $(MAX)^* = MAX$,
4. $(NXA)^* = NXA$.

Then, $X \in \mathbb{C}^{n\times m}$ is called the WMP inverse of A, and is shown by $A^{\dagger}_{MN}$. Noting that, as long as $M = I_{m\times m}$ and $N = I_{n\times n}$, then X is the Moore–Penrose (MP) inverse, or simply the pseudo-inverse of A, and we show it by $A^{\dagger}$ [17]. Furthermore, when the matrix A is non-singular, then the pseudo-inverse will be simplified to the regular inverse.

The weighted singular value decomposition (WSVD), first introduced in [18], is normally applied to define this generalized inverse. Consider that the rank of A is r. Then, we have $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$, satisfying the following relations:

$$U^*MU = I_{m\times m}, \tag{1}$$

and

$$V^*N^{-1}V = I_{n\times n}, \tag{2}$$

such that

$$A = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^*. \tag{3}$$

Thus, $A^{\dagger}_{MN}$ is furnished as follows:

$$A^{\dagger}_{MN} = N^{-1}V \begin{pmatrix} D^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^*M, \tag{4}$$

where we have a diagonal matrix $D = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, for $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, while σ_i^2 is the non-zero eigenvalue of $N^{-1}A^*MA$. In addition,

$$\|A\|_{MN} = \sigma_1, \quad \|A^{\dagger}_{MN}\|_{NM} = \frac{1}{\sigma_r}. \tag{5}$$

In this work, $A^{\#} = N^{-1}A^*M$ is used as the weighted matrix of the conjugate transpose of A. See [19] for more details.

1.3. Literature

Schulz-type methods for the calculation of the WMP inverse are sensitive to the choice of the initial value; that is, the initial choice of matrix must be close enough to the generalized inverse so as to guarantee the scheme to converge [20]. More precisely, convergence can only be observed if the starting matrix is chosen carefully. However, this starting value can be chosen simply for the case of the WMP inverse. The pioneering work in [21] gave several suggestions, along with deep discussions, about how to make such a choice quickly.

Let us, now, briefly provide some of the pioneering and most important matrix iterative methods for computing the WMP inverse.

The second–order Schulz scheme for finding the WMP inverse, requiring only two matrix products per computing cycle, is given by [22]:

$$X_{k+1} = X_k(2I - AX_k),\ k = 0, 1, 2, \cdots . \tag{6}$$

Throughout the work, I stands for the identity matrix, unless clearly stated otherwise.

An improvement of (6) with third-order convergence, known as Chebyshev's method, was discussed in [23] for computing $A^{\dagger}_{MN}$ as follows:

$$X_{k+1} = X_k(3I - AX_k(3I - AX_k)),\ k = 0, 1, 2, \cdots . \tag{7}$$

The authors in [23] proposed another third-order iterative formulation, having one more matrix multiplication, as follows:

$$X_{k+1} = X_k\left[I + \frac{1}{2}(I - AX_k)(I + (2I - AX_k)^2)\right],\ k = 0, 1, 2, \cdots . \tag{8}$$

It is necessary to recall that a general class of iteration schemes for computing the WMP inverse and some other kinds of other generalized inverses was discussed and investigated in [24] (Chapter 5) to have p-th order using a total of p matrix products. An example could be the following fourth-order iteration:

$$X_{k+1} = X_k(I + B_k(I + B_k(I + B_k))),\ k = 0, 1, 2, \cdots , \tag{9}$$

where $B_k = I - AX_k$. As another instance, a tenth-order matrix method could be furnished as follows [25]:

$$\begin{aligned} X_{k+1} \quad = \quad & X_k(I + B_k(I + B_k(I + B_k(I + B_k(I + B_k(I \\ & + B_k(I + B_k(I + B_k(I + B_k))))))))),\ k = 0, 1, 2, \cdots . \end{aligned} \tag{10}$$

1.4. Motivation and Organization

The main motivation behind proposing and extending new iterative methods for the WMP inverse is to apply them in practical large scale problems [26], as well as to improve the computational efficiency, which is directly linked to the concept of numerical analysis for designing new iterative expressions which are economically useful, in being able to reduce computational complexity and time requirements.

Hence, with this motivation at hand, to increase the computational efficiency index as well as to contribute in this field, the main focus of this work is to investigate a tenth-order method requiring only six matrix multiplications per cycle. We prove that this can provide an improvement of the computational efficiency index in calculating the WMP inverse.

The paper is organized as follows. Section 1 discusses the preliminaries and literature of this topic very briefly, to prepare the reader for the analytical discussions of Section 2, in which we describe an effective iteration formulation for the WMP inverse. It is investigated that the method needs only six matrix multiplications to reach its tenth order of convergence.

Concrete proofs of convergence are furnished in Section 3. Section 4 discusses the application of our formulation to the WMP inverses of many randomly generated matrices of various dimensions. Numerical evidence demonstrates the usefulness of this method for computing the WMP inverse, in terms of the elapsed computation time. Finally, several concluding remarks and comments are given in Section 5.

2. A High Order Scheme for the WMP Inverse

For the use of iterative methods, such as the ones described in Section 1, it is required to employ a starting value when computing the WMP inverse. As in [27], one general procedure to find this starting matrix is of the following form:

$$X_0 = \lambda A^{\#}, \tag{11}$$

where $A^{\#} = N^{-1}A^*M$ is the matrix of weighted conjugate transpose (WCT) for the input matrix A and

$$\lambda = \frac{1}{\sigma_1^2}. \tag{12}$$

Recall that, in (12), σ_1 is the the largest eigenvalue of $N^{-1}A^*MA$.

2.1. Derivation

Another reason for proposing a higher order method is that methods based on improvements of the Schulz iteration scheme are slow in the initial phase of iteration. This means that the convergence order cannot be observed at the beginning, it can be seen only after performing several iterates. On the other hand, by incorporating a stop condition using matrix norms, we can increase the elapsed time of executing the written programs for finding the WMP inverse.

Accordingly, to contribute and extend a high order matrix iteration scheme in this context, we first take into account a tenth-order scheme having ten matrix multiplications per cycle, as follows:

$$X_{k+1} \quad = \quad X_k(I + B_k + B_k^2 + \cdots + B_k^9). \tag{13}$$

Now, to develop the performance of (13), we factorize (13) to reduce the number of products. So, we can write

$$X_{k+1} = X_k\,(I + B_k)\,[(I - B_k + B_k^2 - B_k^3 + B_k^4)(I + B_k + B_k^2 + B_k^3 + B_k^4)]. \tag{14}$$

This formulation for the matrix iteration requires seven matrix products. However, it is possible to reduce this number of products by considering a more tight formulation for (14). Hence, we write

$$X_{k+1} = X_k\,(I + B_k)\,M_k, \tag{15}$$

where

$$M_k = [(I + \chi B_k^2 + B_k^4)(I + \kappa B_k^2 + B_k^4)]. \tag{16}$$

To find the unknown weighting coefficients in (15) and, more specifically, in (16), we need to solve a symbolic problem. As such, a Mathematica code [28] was employed to do such a task, as follows:

```
ClearAll["Global‘*"];
fact1 = (1 + a B^2 + B^4);
fact2 = (1 + b B^2 + B^4);
sol = fact1*fact2 + (c B^2) // Expand
S = Table[
s[i] = Coefficient[sol, B^i], {i, 2, 6, 2}
] // Simplify
Solve[
s[2] == 1 && s[4] == 1 && s[6] == 1, {a, b, c}
] // Simplify
{a, b, c} = {a, b, c} /. %[[1]] // Simplify
Chop@sol // Simplify
```

This was given only to ease understanding of the procedure of obtaining the coefficient. Now, we obtain:

$$\chi = \frac{1}{2}\left(1-\sqrt{5}\right), \qquad \kappa = \frac{1}{2}\left(1+\sqrt{5}\right). \tag{17}$$

This means that (15) requires only six matrix products per cycle to hit a convergence speed of ten.

2.2. Several Lemmas

Before providing the main results concerning the convergence analysis of the proposed scheme, we furnish the following lemmas, inspired by [29], which reveal how the iterates generated by (15) have some specific important relations and, then, show a relation between (4) and (15).

Lemma 1. *For $\{X_k\}_{k=0}^{k=\infty}$ produced by (15) using the starting matrix (11), for any $k \geq 0$, it holds that*

$$\begin{array}{l} (MAX_k)^* = MAX_k, \\ (NX_kA)^* = NX_kA, \\ X_kAA^{\dagger}_{MN} = X_k, \\ A^{\dagger}_{MN}AX_k = X_k. \end{array} \tag{18}$$

Proof. The proof can be done by employing mathematical induction. When $k = 0$ and X_0 is the suitable initial matrix, the first two relations in (18) are straightforward. Hence, we discuss the last two relations by applying the following identities:

$$(AA^{\dagger}_{MN})^{\#} = AA^{\dagger}_{MN}, \tag{19}$$

and

$$(A^{\dagger}_{MN}A)^{\#} = A^{\dagger}_{MN}A. \tag{20}$$

Accordingly, we have:

$$\begin{array}{rcl} X_0AA^{\dagger}_{MN} & = & \lambda A^{\#}AA^{\dagger}_{MN} \\ & = & \lambda A^{\#}(AA^{\dagger}_{MN})^{\#} \\ & = & \lambda A^{\#}(A^{\dagger}_{MN})^{\#}A^{\#} \\ & = & \lambda (AA^{\dagger}_{MN}A)^{\#} \\ & = & \lambda A^{\#} \\ & = & X_0, \end{array} \tag{21}$$

and also

$$\begin{array}{rcl} A^{\dagger}_{MN}AX_0 & = & \lambda A^{\dagger}_{MN}AA^{\#} \\ & = & \lambda (A^{\dagger}_{MN}A)^{\#}A^{\#} \\ & = & \lambda (A^{\#}(A^{\dagger}_{MN})^{\#}A^{\#}) \\ & = & \lambda (A(A^{\dagger}_{MN}A)^{\#} \\ & = & \lambda A^{\#} \\ & = & X_0. \end{array} \tag{22}$$

Subsequently, now the relation is valid for $k > 0$, then we discuss that it will still be true for $k+1$. Taking our matrix iteration (15) into consideration, we have:

$$\begin{array}{rcl} (MAX_{k+1})^* & = & (MA(X_k\,(I+B_k)\,[(I+\chi B_k^2+B_k^4)(I+\kappa B_k^2+B_k^4)]))^* \\ & = & [MAX_k\left(I+B_k+B_k^2+B_k^3+B_k^4+B_k^5+B_k^6+B_k^7+B_k^8+B_k^9\right)]^* \\ & = & MA[X_k\left(I+B_k+B_k^2+B_k^3+B_k^4+B_k^5+B_k^6+B_k^7+B_k^8+B_k^9\right)] \\ & = & MAX_{k+1}, \end{array}$$

using that

$$(M(AX_k))^* = MAX_k, \tag{23}$$

M is a Hermitian positive definite matrix ($M^* = M$), and similar facts, such as:

$$\begin{array}{rcl} (M(AX_k)^2)^* & = & (M(AX_k)(AX_k))^* \\ & = & (AX_k)^*(M(AX_k))^* \\ & = & (AX_k)^*(M(AX_k)) \\ & = & (AX_k)^*M^*(AX_k) \\ & = & (M(AX_k))^*(AX_k) \\ & = & M(AX_k)(AX_k) \\ & = & M(AX_k)^2. \end{array} \tag{24}$$

Hence, the first relation in (18) is true for $k+1$, and the 2nd relation could be investigated similarly. For the other relation in (18), by employing the assumption that

$$X_k A A^{\dagger}_{MN} = X_k, \tag{25}$$

and (15), we have:

$$\begin{array}{rcl} X_{k+1}AA^{\dagger}_{MN} & = & (X_k\,(I+B_k)\,[(I+\chi B_k^2+B_k^4)(I+\kappa B_k^2+B_k^4)])AA^{\dagger}_{MN} \\ & = & (X_k + X_kB_k + X_kB_k^2 + X_kB_k^3 + X_kB_k^4 + X_kB_k^5 + X_kB_k^6 \\ & & +X_kB_k^7 + X_kB_k^8 + X_kB_k^9)AA^{\dagger}_{MN} \\ & = & X_kAA^{\dagger}_{MN} + X_kB_kAA^{\dagger}_{MN} + X_kB_k^2AA^{\dagger}_{MN} + X_kB_k^3AA^{\dagger}_{MN} + X_kB_k^4AA^{\dagger}_{MN} \\ & & +X_kB_k^5AA^{\dagger}_{MN} + X_kB_k^6AA^{\dagger}_{MN} + X_kB_k^7AA^{\dagger}_{MN} + X_kB_k^8AA^{\dagger}_{MN} + X_kB_k^9AA^{\dagger}_{MN} \\ & = & (X_k + X_kB_k + X_kB_k^2 + X_kB_k^3 + X_kB_k^4 + X_kB_k^5 \\ & & +X_kB_k^6 + X_kB_k^7 + X_kB_k^8 + X_kB_k^9) \\ & = & X_{k+1}. \end{array}$$

Therefore, the third relation in (18) is valid for $k+1$. The final relation could be investigated in a similar way, and the result follows. The proof is, thus, complete. □

Lemma 2. *Employing the assumptions of Lemma 1 and (3), then for (15) we have:*

$$(V^{-1}N)X_k(M^{-1}(U^*)^{-1}) = diag(T_k, 0), \tag{26}$$

where T_k is a diagonal matrix, $V^*N^{-1}V = I_{n\times n}$, $U^*MU = I_{m\times m}$, $V \in \mathbb{C}^{n\times n}$, $U \in \mathbb{C}^{m\times m}$, and $A = U\Sigma V^*$.

Proof. Assume that $T_0 = \lambda D$ and that σ_i^2 are the non-zero eigenvalues of the matrix $N^{-1}A^*MA$, while $D = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, $\sigma_i > 0$ for any i. Thus, we can write that:

$$\begin{array}{rcl} T_{k+1} := \varphi(T_k) & = & T_k\,(I+(I-DT_k))\,[(I+\chi(I-DT_k)^2+(I-DT_k)^4) \\ & & \times(I+\kappa(I-DT_k)^2+(I-DT_k)^4)]. \end{array} \tag{27}$$

Applying mathematical induction, one can write that

$$\begin{array}{rcl} (V^{-1}N)X_0(M^{-1}(U^*)^{-1}) & = & \lambda(V^{-1}N)A^{\#}(M^{-1}(U^*)^{-1}) \\ & = & \lambda(V^{-1}N)N^{-1}A^*(MM^{-1}(U^*)^{-1}) \\ & = & \lambda(V^{-1}N)N^{-1}V\text{diag}(D,0)U^*(MM^{-1}(U^*)^{-1}) \\ & = & \text{diag}(\lambda D, 0). \end{array} \tag{28}$$

In addition, when (31) is satisfied, then using (15), one can get that:

$$\begin{aligned}(V^{-1}N)X_{k+1}(M^{-1}(U^*)^{-1}) &= (V^{-1}N)X_k(M^{-1}(U^*)^{-1})\\ &\times\left(2I-(V^{-1}N)AX_k(M^{-1}(U^*)^{-1})\right)\\ &\times[(I+\chi(V^{-1}N)(I-AX_k)^2(M^{-1}(U^*)^{-1})\\ &+(V^{-1}N)(I-AX_k)^4(M^{-1}(U^*)^{-1}))\\ &\times(I+\kappa(V^{-1}N)(I-AX_k)^2(M^{-1}(U^*)^{-1})\\ &+(V^{-1}N)(I-AX_k)^4)(M^{-1}(U^*)^{-1})].\end{aligned} \tag{29}$$

Using the fact that $A = U^*MU\mathrm{diag}(D,0) = V^*NV$, one attains

$$(V^{-1}N)X_{k+1}(M^{-1}(U^*)^{-1}) = \mathrm{diag}(\varphi(T_k),0), \tag{30}$$

which shows that (27) is a diagonal matrix. This completes the proof. □

3. Error Analysis

The objective of this section is to provide a matrix analysis for the convergence of the iteration scheme (15).

Theorem 1. *Let us consider that A is an $m \times n$ matrix whose WSVD is provided by (4). Furthermore, assume that the starting value is given by (11). Thus, the matrix sequence from (15) tends to $A^{\dagger}_{MN}$.*

Proof. In light of (4), to prove our convergence for the WMP inverse, we now just need to prove that

$$\lim_{k\to\infty}(V^{-1}N)X_k(M^{-1}(U^*)^{-1}) = \mathrm{diag}(D^{-1},0). \tag{31}$$

It is obtained, using Lemmas 1 and 2, that

$$T_k = \mathrm{diag}(\tau_1^{(k)},\tau_2^{(k)},\ldots,\tau_r^{(k)}), \tag{32}$$

where

$$\tau_i^{(0)} = \lambda\sigma_i \tag{33}$$

and

$$\begin{aligned}\tau_i^{(k+1)} &= \tau_i^{(k)}\left(2I+\sigma_i\tau_i^{(k)}\right)[(I\\ &+\chi(\sigma_i\tau_i^{(k)})^2+(\sigma_i\tau_i^{(k)})^4)(I\\ &+\kappa(\sigma_i\tau_i^{(k)})^2+(\sigma_i\tau_i^{(k)})^4)].\end{aligned} \tag{34}$$

The sequence produced by (34) is the result of employing (15) in calculating the zero σ_i^{-1} of the function

$$\phi(\tau) = \sigma_i - \tau^{-1}, \tag{35}$$

using the starting condition $\tau_i^{(0)}$.

We observe that convergence to σ_i^{-1} can be achieved, as long as

$$0 < \tau_i^{(0)} < \frac{2}{\sigma_i}, \tag{36}$$

which results in a criterion on λ (the selection in formula (12) has now been shown). Hence, $\{T_k\} \to \Sigma^{-1}$, and (31) is satisfied. It is now clear that $\{X_k\}_{k=0}^{k=\infty} \to A^{\dagger}_{MN}$ when $k \to \infty$. This concludes the proof. □

4. Computational Tests

In this section, our aim is to study the efficiency of the proposed approach for calculating the WMP inverse computationally and analytically. To do this, we considered several competitors from the literature in our comparisons, such as those from (6), (7), (10), and (15), denoted by "SM2", "CM3", "KMS10", and "PM10", respectively.

Note that all computations were done in Mathematica 11.0 [30] and the time is reported in seconds. The hardware used was a CPU Intel Core i5 2430-M with 16 GB of RAM.

We know that the efficiency index is expressed by [31]:

$$EI = \rho^{\frac{1}{\kappa}}, \tag{37}$$

where ρ and κ stand for the speed and the whole cost in each cycle, respectively.

As such, the efficiency index of different methods (6–10) and (15) are reported by: $2^{\frac{1}{2}} \simeq 1.414$, $3^{\frac{1}{3}} \simeq 1.442$, $3^{\frac{1}{4}} \simeq 1.316$, $4^{\frac{1}{4}} \simeq 1.414$, $10^{\frac{1}{10}} \simeq 1.258$, and $10^{\frac{1}{6}} \simeq 1.467$, respectively. Clearly, our investigated iterative expression has better a index and can be more useful in finding the WMP inverse.

Example 1. *[29] The purpose of this experiment was to examine the calculation of WMP inverses for 10 uniform randomly provided* $m1 \times n1 = 200 \times 210$ *matrices, as follows:*

```
SeedRandom[12]; no = 10; m1 = 200; n1 = 210;
ParallelTable[A[k] = RandomReal[{1}, {m1, n1}];, {k, no}];
```

where the ten various HPD matrices M and N were given by:

```
ParallelTable[MM[k] = RandomReal[{2}, {m1, m1}];, {k, no}];
ParallelTable[MM[k] = Transpose[MM[k]].MM[k];, {k, no}];
ParallelTable[NN[k] = RandomReal[{3}, {n1, n1}];, {k, no}];
ParallelTable[NN[k] = Transpose[NN[k]].NN[k];, {k, no}];
```

The results by applying the stop termination

$$||X_{k+1} - X_k||_2 \leq 10^{-10}, \tag{38}$$

are reported in Tables 1 and 2, based on the number of iterations, elapsed CPU time (in seconds), and $X_0 = \frac{1}{\sigma_1^2} A^{\#}$. As can be observed from the results, the best scheme in terms of number of iterations and time was (15).

Table 1. Comparison based on the number of iterations and the required mean in Experiment 1.

Methods	SM2	CM3	KMS10	PM10
A_1	68	43	22	22
A_2	69	44	22	22
A_3	67	43	21	21
A_4	71	46	23	23
A_5	72	46	23	23
A_6	72	46	23	23
A_7	66	42	21	21
A_8	78	50	25	25
A_9	63	41	20	20
A_{10}	69	44	22	22
Mean	69.5	44.5	22.2	22.2

Table 2. Comparison based on the elapsed CPU time and its mean in Experiment 1.

Methods	SM2	CM3	KMS10	PM10
A_1	1.4954	1.04826	0.996155	0.755317
A_2	1.4563	1.08006	0.984057	0.767785
A_3	1.37301	1.03847	0.967427	0.720294
A_4	1.53201	1.10927	1.0365	0.789994
A_5	1.50908	1.10164	0.998853	0.794098
A_6	1.51215	1.11421	1.03361	0.823177
A_7	1.39481	1.00116	0.915244	0.743779
A_8	1.62742	1.24438	1.12434	0.87007
A_9	1.32916	0.999683	0.903072	0.709523
A_{10}	1.49738	1.05736	0.985084	0.764156
Mean	1.47267	1.07945	0.994434	0.773819

Example 2. *The iterative methods were compared for five randomly generated dense* $m1 \times n1 = 500 \times 500$ *matrices produced in Mathematica environment by the following piece of code:*

```
m1 = 500; n1 = 500; no = 5; SeedRandom[12];
ParallelTable[A[k] = RandomReal[{0, 1}, {m1, n1}];, {k, no}];
ParallelTable[MM[k] = RandomReal[{0, 1}, {m1, m1}];, {k, no}];
ParallelTable[MM[k] = Transpose[MM[k]].MM[k];, {k, no}];
ParallelTable[NN[k] = RandomReal[{0, 1}, {n1, n1}];, {k, no}];
ParallelTable[NN[k] = Transpose[NN[k]].NN[k];, {k, no}];
```

Here, we applied the stopping condition

$$||X_{k+1} - X_k||_\infty \leq 10^{-10}, \tag{39}$$

with a change in the initial approximation as $X_0 = \frac{1.5}{\sigma_1^2} A^{\#}$. Noting that the weights M and N were very ill-conditioned, as we had produced them to be. We report the results in Tables 3 and 4, which reveal that the novel approach was superior to the existing solvers.

Table 3. Comparison based on the number of iterations and the required mean in Experiment 2.

Methods	SM2	CM3	KMS10	PM10
A_1	98	61	30	30
A_2	86	55	27	27
A_3	83	53	26	26
A_4	85	54	27	27
A_5	81	52	26	26
Mean	86.6	55.	27.2	27.2

Table 4. Comparison based on the elapsed time and its mean in Experiment 2.

Methods	SM2	CM3	KMS10	PM10
A_1	7.89745	7.20885	12.1801	7.11963
A_2	6.90346	6.6397	10.933	6.50042
A_3	2.34013	2.23622	3.75341	2.20977
A_4	2.23133	2.15679	3.78848	2.23819
A_5	2.44316	2.26733	3.79153	2.2391
Mean	4.36311	4.10178	6.88929	4.06142

One other application of (15), aside from computing the WMP inverse, is in finding good approximate inverse pre-conditioners for Krylov methods when tackling large sparse linear system of

equations (see, e.g., [29]). In fact, to apply our scheme in such environments, we can employ several commands, such as `SparseArray[]` for handling sparse matrices.

The main advantage of the proposed method is the improvement of convergence order obtained by improving the computational efficiency index. Although this computational efficiency index improvement was not observed to be drastic, in solving practical problems in higher dimensions it leads to a clear reduction of computation time.

5. Ending Notes

We have investigated a tenth order iterative method for computing the WMP inverse requiring only six matrix products. The WMP inverse has many applications, from the numerical solution of non-linear equations (those involving singular linear systems [32]) to direct engineering applications. Clearly, the efficiency index will reach $10^{1/6} \simeq 1.46$, which is better than the Newton–Schulz and Chebyshev methods for calculating the WMP inverse. The convergence order of the scheme was supported and upheld analytically. The extension of this improved version of the hyperpower family for computing other types generalized inverses, such as outer and inner inverses, under special criteria and initial matrices provides a direction for future works in this active topic of research.

Funding: This research project was supported by a grant from the "Research Center of the Female Scientific and Medical Colleges", Deanship of Scientific Research, King Saud University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bin Jebreen, H.; Chalco-Cano, Y. An improved computationally efficient method for finding the Drazin inverse. *Disc. Dyn. Nat. Soc.* **2018**, *2018*, 6758302. [CrossRef]
2. Niazi Moghani, Z.; Khanehgir, M.; Mohammadzadeh Karizaki, M. Explicit solution to the operator equation $AXD + FX^*B = C$ over Hilbert C*-modules. *J. Math. Anal.* **2019**, *10*, 52–64.
3. Stanimirović, P.S.; Katsikis, V.N.; Srivastava, S.; Pappas, D. A class of quadratically convergent iterative methods. *RACSAM* **2019**, 1–22. [CrossRef]
4. Ben-Israel, A.; Greville, T.N.E. *Generalized Inverses: Theory and Applications*, 2nd ed.; Springer: New York, NY, USA, 2003.
5. Godunov, S.K.; Antonov, A.G.; Kiriljuk, O.P.; Kostin, V.I. *Guaranteed Accuracy in Numerical Linear Algebra*; Springer: Dordrecht, The Netherlands, 1993.
6. Fredholm, I. Sur une classe d'équations fonctionnelles. *Acta Math.* **1903**, *27*, 365–390. [CrossRef]
7. Wang, G.R. A new proof of Grevile's method for computing the weighted M-P inverse. *J. Shangai Norm. Univ.* **1985**, *3*, 32–38.
8. Bakhtiari, Z.; Mansour Vaezpour, S. Positive solutions to the system of operator equations $T_iX = U_i$ and $T_iXV_i = U_i$. *J. Math. Anal.* **2016**, *7*, 102–117.
9. Xia, Y.; Chen, T.; Shan, J. A novel iterative method for computing generalized inverse. *Neural Comput.* **2014**, *26*, 449–465. [CrossRef]
10. Courriee, P. Fast computation of Moore-Penrose inverse matrices. *arXiv preprint* **2008**, arXiv:0804.4809.
11. Lu, S.; Wang, X.; Zhang, G.; Zhou, X. Effective algorithms of the Moore-Penrose inverse matrices for extreme learning machine. *Intell. Data Anal.* **2015**, *19.4*, 743–760. [CrossRef]
12. Sheng, X.; Chen, G. The generalized weighted Moore-Penrose inverse. *J. Appl. Math. Comput.* **2007**, *25*, 407–413. [CrossRef]
13. Soleymani, F.; Soheili, A.R. A revisit of stochastic theta method with some improvements. *Filomat* **2017**, *31*, 585–596. [CrossRef]
14. Söderström, T.; Stewart, G.W. On the numerical properties of an iterative method for computing the Moore-Penrose generalized inverse. *SIAM J. Numer. Anal.* **1974**, *11*, 61–74. [CrossRef]
15. Stanimirović, P.S.; Ciric, M.; Stojanović, I.; Gerontitis, D. Conditions for existence, representations, and computation of matrix generalized inverses. *Complexity* **2017**, *2017*, 6429725. [CrossRef]

16. Gulliksson, M.E.; Wedin, P.A.; Wei, Y. Perturbation identities for regularized Tikhonov inverse and weighted pseudo inverse. *BIT* **2000**, *40*, 513–523. [CrossRef]
17. Roy, F.; Gupta, D.K.; Stanimirović, P.S. An interval extension of SMS method for computing weighted Moore-Penrose inverse. *Calcolo* **2018**, *55*, 15. [CrossRef]
18. Van Loan, C.F. Generalizing the singular value decomposition. *SIAM J. Numer. Anal.* **1976**, *13*, 76–83. [CrossRef]
19. Zhang, N.; Wei, Y. A note on the perturbation of an outer inverse. *Calcolo* **2008**, *45*, 263–273. [CrossRef]
20. Ghorbanzadeh, M.; Mahdiani, K.; Soleymani, F.; Lotfi, T. A class of Kung-Traub-type iterative algorithms for matrix inversion. *Int. J. Appl. Comput. Math.* **2016**, *2*, 641–648. [CrossRef]
21. Pan, V.Y. *Structured Matrices and Polynomials: Unified Superfast Algorithms*; BirkhWauser: Boston, MA, USA; Springer: New York, NY, USA, 2001.
22. Schulz, G. Iterative Berechnung der Reziproken matrix. *Z. Angew. Math. Mech.* **1933**, *13*, 57–59. [CrossRef]
23. Li, H.-B.; Huang, T.-Z.; Zhang, Y.; Liu, X.-P.; Gu, T.-X. Chebyshev-type methods and preconditioning techniques. *Appl. Math. Comput.* **2011**, *218*, 260–270. [CrossRef]
24. Krishnamurthy, E.V.; Sen, S.K. *Numerical Algorithms—Computations in Science and Engineering*; Affiliated East-West Press: New Delhi, India, 1986.
25. Sen, S.K.; Prabhu, S.S. Optimal iterative schemes for computing Moore-Penrose matrix inverse. *Int. J. Syst. Sci.* **1976**, *8*, 748–753. [CrossRef]
26. Grevile, T.N.E. Some applications of the pseudo-inverse of matrix. *SIAM Rev.* **1960**, *3*, 15–22. [CrossRef]
27. Huang, F.; Zhang, X. An improved Newton iteration for the weighted Moore-Penrose inverse. *Appl. Math. Comput.* **2006**, *174*, 1460–1486. [CrossRef]
28. Sánchez León, J.G. *Mathematica Beyond Mathematics: The Wolfram Language in the Real World*; Taylor & Francis Group: Boca Raton, FL, USA, 2017.
29. Zaka Ullah, M.; Soleymani, F.; Al-Fhaid, A.S. An efficient matrix iteration for computing weighted Moore-Penrose inverse. *Appl. Math. Comput.* **2014**, *226*, 441–454. [CrossRef]
30. Trott, M. *The Mathematica Guide-Book for Numerics*; Springer: New York, NY, USA, 2006.
31. Ostrowski, A.M. Sur quelques transformations de la serie de LiouvilleNewman. *CR Acad. Sci. Paris* **1938**, *206*, 1345–1347.
32. Soheili, A.R.; Soleymani, F. Iterative methods for nonlinear systems associated with finite difference approach in stochastic differential equations. *Numer. Algor.* **2016**, *71*, 89–102. [CrossRef]

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).