Linear Algebra and its Applications 484 (2015) 477–495

Contents lists available at ScienceDirect

Linear Algebra and its Applications

www.elsevier.com/locate/laa

On hyperpower family of iterations for computing outer inverses possessing high efficiencies

F. Soleymani [a], Predrag S. Stanimirović [b,*], F. Khaksar Haghani [a]

[a] *Department of Mathematics, Faculty of Basic Science, Shahrekord Branch, Islamic Azad University, Shahrekord, Iran*
[b] *University of Nis, Faculty of Sciences and Mathematics, Visegradska 33, 18000 Nis, Serbia*

ARTICLE INFO

Article history:
Received 10 November 2014
Accepted 11 July 2015
Available online 25 July 2015
Submitted by P. Semrl

MSC:
65F30

Keywords:
Outer inverse
Hyperpower iteration
Complexity
Efficiency index

ABSTRACT

Hyperpower iteration is a powerful family of iterative methods for finding outer inverses with arbitrary order of convergence $p \geq 2$. In this paper, we present several systematic algorithms for factorizations of the hyperpower iterative family of arbitrary orders with a view to reduce the necessary number of multiplications in each iterative step. Additionally, effective heuristics for factoring arbitrary higher orders hyperpower iteration are presented. The new formulations of the hyperpower iterative steps are convergent with higher computational efficiency indices.

1. Introduction

Let $A \in \mathbb{C}_r^{m\times n}$ be a complex $m \times n$ matrix of rank r, T be a subspace of $\mathbb{C}^n$ of dimension $t \leq r$ and S be a subspace of $\mathbb{C}^m$ of dimension $m-t$, then A has a $\{2\}$-inverse X with range $\mathcal{R}(X) = T$ and null space $\mathcal{N}(X) = S$ if and only if $AT \oplus S = \mathbb{C}^m$. In the

* Corresponding author.
E-mail addresses: fazl_soley_bsb@yahoo.com (F. Soleymani), pecko@pmf.ni.ac.rs (P.S. Stanimirović), haghani1351@yahoo.com (F. Khaksar Haghani).

http://dx.doi.org/10.1016/j.laa.2015.07.010
0024-3795/

case when the existence of this outer inverse is ensured, X is unique and it is denoted by $A^{(2)}_{T,S}$ (for theoretical and application aspects, see [3] and [5], respectively).

One of the most important family of iterative methods for computing $A^{(2)}_{T,S}$ is the hyperpower iteration. These iterations possess an arbitrary order $p \geq 2$ and are given by the standard form

$$X_{k+1} = X_k + X_k(I - AX_k) + \cdots + X_k(I - AX_k)^{p-1}, \tag{1.1}$$

which was fully discussed in [4]. This scheme could also be rewritten as

$$X_{k+1} = X_k \left(I + R_k + \cdots + R_k^{p-1}\right) = X_k \sum_{i=0}^{p-1} R_k^i, \quad R_k = I - AX_k. \tag{1.2}$$

The iteration (1.1), or equivalently (1.2), requires p matrix–matrix products to achieve pth order of convergence. Clearly, the choice $p = 2$ yields to the Schulz matrix iteration, expressed by

$$X_{k+1} = X_k(2I - AX_k) = X_k(I + R_k), \tag{1.3}$$

with a local quadratic convergence [2]. A deep discussion and improvement of the method (1.2) and its related issues were presented in [9] and [10].

We remark that for having a convergent sequence it is sufficient only to choose the initial matrix in the form $X_0 = \alpha G$, where α is an appropriately selected real parameter and G is a selected matrix of dimensions $n \times m$ and of rank s, $0 < s \leq r = \text{rank}(A)$. See [10] for detailed information.

The following two problems arise now in a natural way:

- Is it possible to construct an iterative hyperpower method of any order with lower complexity?
- And how to give a comparison of the hyperpower methods of different orders?

As a criterion for comparing two factorizations (simplifications) of a hyperpower method, we modify the comparison of two hyperpower methods from [1]. Let $\Upsilon_1(X^{(p)}_{k+1})$ and $\Upsilon_2(X^{(p)}_{k+1})$ be two factorizations of $X^{(p)}_{k+1}$. The factorization Υ_1 is better than the factorization Υ_2 if, after some iterative steps n_1 and n_2, respectively, the factorization Υ_1 achieves better accuracy using the same amount of matrix multiplications for both factorizations. Let $\theta_{\Upsilon_1}(p)$ and $\theta_{\Upsilon_2}(p)$ be the number of matrix multiplications required by these factorizations. By extending the comparison from [1], it is possible to conclude

$$n_1 \cdot \theta_{\Upsilon_1}(p) = n_2 \cdot \theta_{\Upsilon_2}(p),\ p^{n_1} \geq p^{n_2}, \tag{1.4}$$

which further implies $p^{\frac{1}{\theta_{\Upsilon_1}(p)}} \geq p^{\frac{1}{\theta_{\Upsilon_2}(p)}}$. Such a concept has a close link with the definition of computational efficiency index. In fact in the next section, we use this index for

comparing different methods of various orders which possess various matrix products per cycle as well.

The idea of reducing the number of multiplications in the calculation of a polynomial through factorizations and nested forms is not new. Actually, it has been widely used technique in numerical analysis. But, so far published results in the field of matrix generalized inversion are based on some particular heuristics. The authors of the paper [7] proposed accelerated hyperpower methods of orders 5 and 9 for the computation of outer inverse, which reduce the total number of matrix multiplications per iteration. A new 7th order method requiring only 5 matrix multiplications per iteration was obtained in [11]. An efficient factorization of the hyperpower method with the order of convergence 30 was recently constructed and analyzed in [8]. Our intention in this paper is to define a systematic and algorithmic approach in reducing the number of multiplications in the general hyperpower iterative scheme. Three rules for proper factorizations of the polynomial which defines the general hyperpower iterative scheme are expressed. Moreover, guaranteed improvements which can be obtained by these factorizations are determined.

In this paper, we propose factorized forms of an arbitrary pth order hyperpower iterative method for computing generalized inverses. In some cases, these factorizations may not be the most efficient. But, they are universal and applicable to arbitrary hyperpower iteration, without restrictions. In addition, they sometimes help in finding the most efficient heuristics for factorization. Particularly, a hyperpower iterative family of the orders $10 \leq p \leq 19$ is investigated in detail. Appropriate and simple factorizations allow achieving the pth order of convergence with a lower number of matrix–matrix products per cycle (instead of p matrix products needed in the standard form). This is a substantial improvement since matrix products are high computational cost operations. Furthermore, a non-standard way, known as non-normalized approach, for obtaining new factorizations with higher efficiency indices will also be discussed.

The rest of this work proceeds as follows. The main results of this study are uncovered in Section 2, where some simplifications of (1.2) are given to reach the same order of convergence but via a lower number of matrix products. A discussion for improving the computational efficiency indices is also furnished. Section 3 is devoted to the derivation of a 17th-order method by consuming 7 matrix products only, which results in the efficiency index $17^{1/7} \approx 1.4989$. Section 4 presents the numerical behavior of the proposed formulations on a matrix problem in an academical manner. Finally, Section 5 ends the paper by reviewing some of the important results of this work.

2. Factorized forms of the hyperpower family

Computational efficiency of different matrix fixed-point type methods can be calculated in a prosperous manner by applying the definition of the efficiency index. The efficiency index (also known as the computational efficiency), of an iterative method

with the convergence order p that requires $\theta(p)$ matrix products, is computed by an extension of the Ostrowski's formula [6] as follows:

$$EI = p^{\frac{1}{\theta(p)}}. \tag{2.1}$$

Another variation of (2.1), known as logarithm of efficiency index, could be expressed as

$$LEI = \frac{\ln p}{\theta(p)}. \tag{2.2}$$

The above two indices, defined in (2.1) and (2.2), along with the informational efficiency index [12] are excellent tools for comparing matrix methods of various orders. These indices can respond clearly and theoretically to the two questions arisen in Section 1.

The basic iterative scheme (1.2) allows achieving the pth order of convergence by means of p matrix products. Therefore, computational efficiency index of the standard iterative scheme (1.2) is $EI = p^{\frac{1}{p}}$, $LEI = \frac{\ln p}{p}$.

Our aim is to develop iterative methods which accelerate the convergence of the hyperpower iteration with a minimal number of matrix products. Thus, toward presenting simple variants of (1.2) with lower complexity, we must decrease the number of matrix products. This could be done using proper factorizations of certain matrix polynomials. In what follows, we propose several proper factorizations of (1.2), applicable to arbitrary order p. Particularly, we look for optimal formulations of the hyperpower iterations for some higher values of the order p, $10 \leq p \leq 19$.

Note that several results for lower values of p have been published in the literature (such as [13]), and we refer the readers to them for further pointers.

Let us observe the matrix polynomials $P_k(p) = \sum_{i=0}^{p} R_k^i$. To avoid obscurity, the expression which defines the hyperpower method of the order p in the kth iterative step is denoted by $X_k^{(p)}$. Then (1.2) can be denoted shortly by

$$\begin{aligned} X_{k+1}^{(p)} = X_k^{(p)} P_k(p-1) = X_k^{(p)} \left(I + R_k P_k(p-2)\right), \quad p \geq 2, \\ P_k(0) = I, \ R_k = I - A X_k^{(p)}. \end{aligned} \tag{2.3}$$

We use (2.3) as the general rule which defines functionality between the hyperpower methods of the orders p and $p-1$. If $\theta(p)$ denotes the number of matrix multiplications required to compute the new iterative step in the hyperpower iteration (1.2) of the order p, then (2.3) clearly implies $\theta(1) = 0$, $\theta(2) = 2$, $\theta(p) = 1 + \theta(p-1)$, $p \geq 3$. This further implies $\theta(p) = p$, $p \geq 2$. Therefore, without smart factorizations, the computational efficiency index of the iterative scheme (1.2) changes according to the rule $EI = p^{\frac{1}{\theta(p)}} = p^{\frac{1}{\theta(p-1)+1}} = p^{\frac{1}{p}}$. But, using appropriate factorizations, it is possible to achieve smaller values for $\theta(p)$, i.e. greater values for $p^{\frac{1}{\theta(p)}}$.

We now define appropriate upper bounds for computational efficiency indices of the hyperpower family, which can be achieved using proposed factorizations of the general iterative scheme (1.2). In order to simplify the presentation, the notation

$$\Phi(l,t) = I + R_k^l + R_k^{2l} + \cdots + R_k^{t*l} = \sum_{i=0}^{t} R_k^{i*l} \tag{2.4}$$

will be used. Similarly, $O(P_k(p))$ (resp. $O(\Phi(l,t))$) stands for the number of matrix multiplications grasped during the computation of the polynomial $P_k(p)$ (resp. $\Phi(l,t)$). Besides, to generalize our investigations, let us denote by $\Upsilon(P_k(p))$ an arbitrary factorization Υ of $P_k(p)$ and the set of all possible factorizations of $P_k(p)$ by $\mathfrak{F}$.

Definition 2.1. The number of matrix multiplications required to compute $\Upsilon(P_k(p))$ is denoted by $O\left(\Upsilon(P_k(p))\right) = O_\Upsilon\left(P_k(p)\right)$. The computational complexity of a polynomial $P_k(p)$ is the minimal number of matrix multiplications spanned to compute $P_k(p)$ with respect to all possible factorizations $\Upsilon \in \mathfrak{F}$:

$$\Omega\left(P_k(p)\right) = \min_{\Upsilon \in \mathfrak{F}} \left\{O_\Upsilon\left(P_k(p)\right)\right\}.$$

Proposition 2.1. *The number of matrix multiplications necessary to generate $X_{k+1}^{(p)}$, defined by (2.3), is equal to*

$$\theta(p) = 1 + O\left(P_k(p-1)\right), \quad p \geq 2. \tag{2.5}$$

In the case when the factorization $\Upsilon(P_k(p-1))$ is used, the number of matrix multiplications $\theta(p)$ becomes

$$\theta_\Upsilon(p) = 1 + O_\Upsilon\left(P_k(p-1)\right), \quad p \geq 2. \tag{2.6}$$

Subsequently, the minimal efficiency index is equal to

$$\theta_{\min}(p) = 1 + \Omega(P_k(p-1)), \quad p \geq 2. \tag{2.7}$$

Lemma 2.1. *Let the polynomial $P_k(l-1)$ be computed, for an arbitrary integer $l \geq 2$. Then the minimal number of multiplications needed for the computation of $\Phi(l,t)$ satisfies $\Omega(\Phi(l,t)) \leq t$, for each $t \geq 1$.*

Proof. Indeed, since R_k^{l-1} is included in $P_k(l-1)$, it is necessary to perform an additional matrix–matrix multiplication to compute $R_k^l = R_k R_k^{l-1}$ and another $t-1$ multiplications to compute $R_k^{i*l} = R_k^{(i-1)\,l} R_k^l$, $i = 2, \ldots, t$. But, sometimes it is possible to find a factorization $\Upsilon(\Phi(l,t))$ which satisfies the inequality $O_\Upsilon(\Phi(l,t)) < t$. □

Example 2.1. Under the assumption that $P_k(2) = I + R_k + R_k^2$ is defined, it is clear that

$$O(\Phi(3,6)) = O\left(I + R_k^3 + R_k^6 + R_k^9 + R_k^{12} + R_k^{15} + R_k^{18}\right) = 6.$$

But, the factorization $\Upsilon\left(I+R_k^3+R_k^6+R_k^9+R_k^{12}+R_k^{15}+R_k^{18}\right)=I+(R_k^3+R_k^6+R_k^9)(I+R_k^9)$ gives

$$O_\Upsilon\left(I+R_k^3+R_k^6+R_k^9+R_k^{12}+R_k^{15}+R_k^{18}\right)=O\left(I+(R_k^3+R_k^6+R_k^9)(I+R_k^9)\right)=4.$$

Theorem 2.1 is the main result of this section and defines upper bounds for the complexity of the hyperpower family.

Theorem 2.1. *If the factorization Υ_{t-1}, defined by*

$$\Upsilon_{t-1}(P_k(t^j\,l-1))=P_k(l-1)\prod_{i=0}^{j-1}\Phi(t^i\,l,t-1),\quad l,j\geq 1,\ t\geq 2, \tag{2.8}$$

of $P_k(t^j\,l-1)$ is used, then the computational efficiency index of the hyperpower family (1.2) satisfies the following inequalities:

$$\begin{aligned}&\theta_{\Upsilon_2}(t^j)=t\,j,\quad j\geq 1,\ l=1,\ t\geq 2,\\ &\theta_{\Upsilon_{t-1}}(t^j\,l)=\theta(l)+t\,j\leq l+t\,j,\quad j\geq 1,\ j\geq 1,\ l\geq 2,\ t\geq 2.\end{aligned} \tag{2.9}$$

Proof. An application of (2.8) in the case $l>1$ produces

$$\begin{aligned}\theta_{\Upsilon_{t-1}}(t^j\,l)&=O\left(X_k^{(t^j\,l)}P_k(t^j\,l-1)\right)=O\left(X_k^{(t^j\,l)}P_k(l-1)\prod_{s=0}^{j-1}\Phi(t^s\,l,t-1)\right)\\ &=1+O\left(P_k(l-1)\right)+\sum_{s=0}^{j-1}\left(1+O\left(\Phi(t^s\,l,t-1)\right)\right).\end{aligned} \tag{2.10}$$

According to Lemma 2.1,

$$\Omega(\Phi(t^s\,l,t-1))\leq O(\Phi(t^s\,l,t-1))=O\left(\sum_{i=0}^{t-1}R_k^{i*t^s\,l}\right)=t-1.$$

Consequently, $\theta_{\Upsilon_{t-1}}(t^j\,l)=\theta(l)+t\,j$. Now it is sufficient to use $\theta(l)\leq l$ in the case $l\geq 2$.

In the case $l=1$ the proof can be completed using $P_k(l-1)=P_k(0)=I$, which implies $\theta(l)=0$. □

Example 2.2. Let us consider the case $p=36=3^2\,4$. Factorization Υ_2 gives

$$\begin{aligned}\Upsilon_2\left(X_{k+1}^{(3^2\,4)}\right)&=X_k^{(3^2\,4)}P_k(3)\sum_{i=0}^{8}R_k^{4i}\\ &=X_k^{(3^2\,4)}P_k(3)\Phi(4,2)\Phi(12,2)\end{aligned}$$

Table 1
Upper bounds $\theta(p)$ for the orders $p = 3^j\, l$.

$p = 3^j\, l$	$\theta(p)$	$p = 3^j\, l$	$\theta(p)$
$3 = 3^1$	3	$18 = 3^2\, 2$	8
$6 = 3^1\, 2$	5	$21 = 3^1\, 7$	10
$9 = 3^2$	6	$36 = 3^2\, 4$	10
$12 = 3^1\, 4$	7	$27 = 3^3$	9
$15 = 3^1\, 5$	8	$30 = 3^1\, 10$	13

Table 2
Upper bounds $\theta(p)$ for some of the orders $p = t^j\, l$, $t = 2, 4, 5, 7$.

$p = 2^j\, l$	$\theta(p)$	$p = 4^j\, l$	$\theta(p)$	$p = 5^j\, l$	$\theta(p)$	$p = 7^j\, l$	$\theta(p)$
$4 = 2^2$	4	$8 = 4^1\, 2$	6	$5 = 5^1$	5	$7 = 7^1$	7
$8 = 2^3$	6	$12 = 4^1\, 3$	7	$10 = 5^1\, 2$	7	$14 = 7^1\, 2$	9
$40 = 2^3\, 5$	11	$16 = 4^2$	8	$15 = 5^1\, 3$	8	$21 = 7^1\, 3$	10
$56 = 2^3\, 7$	13	$32 = 4^2\, 2$	10	$20 = 5^1\, 4$	9	$28 = 7^1\, 4$	11
$80 = 2^4\, 5$	13	$64 = 4^3$	12	$25 = 5^2$	10	$245 = 7^2\, 5$	19

and further

$$\theta_{\Upsilon_2}(3^2\, 4) = \theta(4) + 3 * 2 \leq 4 + 3 * 2 = 10.$$

Further verifications produce

$$\begin{aligned}\theta_{\min}(3^2\, 4) &= \theta_{\min}(4) + 3 * 2 = 1 + \Omega(P_k(3)) = 1 + \Omega(I + R_k + R_k^2 + R_k^3) + 6\\ &= 1 + \Omega\left((I + R_k)(I + R_k^2)\right) + 6\\ &= 3 + 6 = 9.\end{aligned}$$

Similarly,

$$\theta_{\Upsilon_2}(3^2) = O\left(X_k^{(3^2)} \Phi(4,2)\Phi(12,2)\right) = 1 + 2 + 1 + 2 = 3 * 2 = 6.$$

According to Theorem 2.1, in Table 1 we state upper bounds $\theta(p)$ for several orders $p = 3^j\, l$.

Example 2.3. According to Theorem 2.1, in Table 2 we state upper bounds $\theta(p)$ for different orders $p = t^j\, l$.

Beside the factorization (2.8), we observed the factorization used in [7]. Such a factorization is applicable for the orders $p = 2^q + 1$, $q \geq 2$ and it is defined by

$$\begin{aligned}X_{k+1}^{(p)} &= X_k^{(p)}\, P_k\left(2^q\right) = X_k\, U_{k_i}\left(2^{q-i-1}\right) V_{k_i}\left(2^{q-i-1}\right),\\ U_{k_i}\left(2^{q-i}\right) &= U'_{k_{i+1}}\left(2^{q-i-1}\right) U''_{k_{i+1}}\left(2^{q-i-1}\right) + \Gamma_{k_i}\left(2^{q-i-2}\right),\ i = 1, \ldots, q-1,\\ V_{k_i}\left(2^{q-i}\right) &= V'_{k_{i+1}}\left(2^{q-i-1}\right) V''_{k_{i+1}}\left(2^{q-i-1}\right) + \Sigma_{k_i}\left(2^{q-i-2}\right),\ i = 1, \ldots, q-1, \quad (2.11)\end{aligned}$$

where U_{k_i} and V_{k_i} are appropriately chosen polynomials of the order 2^{q-i} and Γ_{k_i}, Σ_{k_i} are corresponding corrections. The following upper bound is valid in this case.

Theorem 2.2. *The computational efficiency index of the hyperpower family satisfies the equality*

$$\theta(2^q+1) = 1+(q+1), \tag{2.12}$$

if the factorization (2.11) of $P_k(2^q)$ is used.

Proof. The factorization (2.11) implies the following

$$\begin{aligned}\theta(2^q+1) &= 1+1+O\left(U_{k_1}\left(2^{q-1}\right)\right)\\ &= 1+2+O\left(U_{k_2}\left(2^{q-2}\right)\right)\\ &= 1+q-1+\theta(2)\\ &= 1+(q+1),\end{aligned} \tag{2.13}$$

which is (2.12). □

By generalizing the factorization (2.11), it is possible to define the following factorizations for $X_{k+1}^{(p)} = X_k^{(p)} P_k\left(3^q\right)$, applicable on the orders $p = 3^q+1$, $q \geq 2$:

$$\begin{aligned}&P_k\left(3^{q-i}\right) = U_{k_i}\left(3^{q-i-1}\right) V_{k_i}\left(3^{q-i-1}\right) W_{k_i}\left(3^{q-i-1}\right),\\ &U_{k_i}\left(3^{q-i}\right) = U'_{k_{i+1}}\left(3^{q-i-1}\right) U''_{k_{i+1}}\left(3^{q-i-1}\right) U'''_{k_{i+1}}\left(3^{q-i-1}\right) + \Gamma_{k_i}\left(3^{q-i-2}\right),\\ &\quad i=1,\ldots,q-1,\\ &V_{k_i}\left(3^{q-i}\right) = V'_{k_{i+1}}\left(3^{q-i-1}\right) V''_{k_{i+1}}\left(3^{q-i-1}\right) V'''_{k_{i+1}}\left(3^{q-i-1}\right) + \Sigma_{k_i}\left(3^{q-i-2}\right),\\ &\quad i=1,\ldots,q-1,\\ &W_{k_i}\left(3^{q-i}\right) = W'_{k_{i+1}}\left(3^{q-i-1}\right) W''_{k_{i+1}}\left(3^{q-i-1}\right) W'''_{k_{i+1}}\left(3^{q-i-1}\right) + \Lambda_{k_i}\left(3^{q-i-2}\right),\\ &\quad i=1,\ldots,q-1,\end{aligned} \tag{2.14}$$

where U_{k_i}, V_{k_i} and W_{k_i} are appropriately chosen polynomials of the order 3^{q-i} and Γ_{k_i}, Σ_{k_i} and Λ_{k_i} are corresponding corrections. The following upper bound is valid in this case.

Theorem 2.3. *The computational efficiency index of the hyperpower family satisfies the equalities:*

$$\theta(3^q+1) = 1+(q+1) \tag{2.15}$$

if the factorization (2.14) of $P_k(3^q)$ is used.

Remark 2.1. Thinking the same way, it seems interesting to define factorizations efficient on the orders $p = t^q + 1$, $q \geq 2$, $t \geq 2$ which satisfy $\theta(t^q + 1) = 1 + (q + 1)$.

Now, we study some of the important factorizations of the hyperpower scheme according to the above-mentioned discussions.

2.1. Case of $p = 10$

In this case, we propose the following variant for the hyperpower iteration, at which only 7 matrix products are required:

$$\begin{aligned} X_{k+1}^{(10)} &= X_k^{(10)} \left(I + R_k P_k(8)\right) \\ &= X_k^{(10)} \left(I + R_k P_k(2) \left(I + R_k^3 + R_k^6\right)\right) \\ &= X_k^{(10)} \left(I + R_k \left(I + R_k + R_k^2\right) \left(I + R_k^3 + R_k^6\right)\right). \end{aligned} \tag{2.16}$$

This results in $EI((2.16)) = 10^{1/7} \approx 1.3894$.

If (2.16) is considered in the form

$$X_{k+1}^{(10)} = X_k^{(10)} \left(I + \left(R_k + R_k^2 + R_k^3\right) \left(I + R_k^3 + R_k^6\right)\right), \tag{2.17}$$

then only 6 matrix products are required, so that the efficiency index increased to $EI((2.16)) = 10^{1/6} \approx 1.4677$.

2.2. Case of $p = 11$

Here, using (2.17) and applying the rule (2.3), we present a new formulation for the case of $p = 11$ via proper factorizations as comes next

$$X_{k+1}^{(11)} = X_k^{(11)} \left(I + R_k \left(I + \left(R_k + R_k^2 + R_k^3\right) \left(I + R_k^3 + R_k^6\right)\right)\right), \tag{2.18}$$

which improves the efficiency index to $11^{1/7} \approx 1.4085$.

2.3. Case of $p = 12$

Our new formulation for $p = 12$ is as follows:

$$X_{k+1}^{(12)} = X_k^{(12)} (I + R_k) \left(I + R_k^2\right) \left(I + R_k^2 + R_k^4\right) \left(I - R_k^2 + R_k^4\right), \tag{2.19}$$

where its computational efficiency index is $12^{1/7} \approx 1.4261$.

Following (2.8), one can verify

$$X_{k+1}^{(3l)} = X_k^{(3l)} P_k(3l - 1) = X_k^{(3l)} P_k(l - 1) \Phi(l, 2)$$

and the iterative process (2.19) can be reorganized in the following way

$$\begin{aligned} X_{k+1}^{(12)} &= X_k^{(12)} P_k(11) = X_k^{(12)} P_k(3) \left(I + R_k^4 + R_k^8\right) \\ &= X_k^{(12)} \left(I + R_k + R_k^2 + R_k^3\right) \left(I + R_k^4 + R_k^8\right), \end{aligned} \tag{2.20}$$

with the same computational efficiency index as in (2.19).

2.4. Case of $p = 13$

In this case, after many factorizations we are able to present the following variant for the hyperpower iteration:

$$\begin{aligned} X_{k+1}^{(13)} = X_k^{(13)} &\left(I + R_k(I + R_k)\left(I + (-I + R_k)R_k\right)\left(I + R_k^2\right)\left(I + R_k + R_k^2\right)\right. \\ &\left. \times \left(I - R_k^2 + R_k^4\right)\right). \end{aligned} \tag{2.21}$$

The iterative expression (2.21) can further be simplified as follows:

$$X_{k+1}^{(13)} = X_k^{(13)} \left(I + (R_k + R_k^2)\left(I - R_k + R_k^2\right)(I + R_k^2)(I + R_k + R_k^2)(I - R_k^2 + R_k^4)\right), \tag{2.22}$$

with the efficiency index $13^{1/8} \approx 1.3779$, which is higher than $13^{1/13} \approx 1.2181$ of the standard form (1.2).

Iterative process (2.21) can also be reorganized in the following way

$$\begin{aligned} X_{k+1}^{(13)} &= X_k^{(13)} P_k(12) = X_k^{(13)} \left(I + R_k P_k(11)\right) = X_k^{(13)} \left(I + R_k P_k(3) \left(I + R_k^4 + R_k^8\right)\right) \\ &= X_k^{(13)} \left(I + (R_k + R_k^2 + R_k^3 + R_k^4)\left(I + R_k^4 + R_k^8\right)\right), \end{aligned} \tag{2.23}$$

with the higher computational efficiency index $13^{1/7} \approx 1.44256$. Let us state that it is possible to derive (2.23) immediately applying (2.3) on (2.20).

2.5. Case of $p = 14$

We first deduce the following formulation for this case

$$\begin{aligned} X_{k+1}^{(14)} = X_k^{(14)} &\left(I + R_k \left(I + R_k(I + R_k)\left(I + (-I + R_k)R_k\right)\left(I + R_k^2\right)\left(I + R_k + R_k^2\right)\right.\right. \\ &\left.\left. \times \left(I - R_k^2 + R_k^4\right)\right)\right), \end{aligned} \tag{2.24}$$

while further simplification results in

$$X_{k+1}^{(14)} = X_k^{(14)} \left(I + R_k + (R_k^3 - R_k)(I + R_k^2 + R_k^4)\right) \left(I + (R_k + R_k^2)(I + R_k^2 + R_k^4)\right), \tag{2.25}$$

where it hits the computational efficiency index $14^{1/8} \approx 1.3908$.

But, an alternative and simpler variant of (2.25) can be derived following (2.23). It is given in the form

$$\begin{aligned} X_{k+1}^{(14)} &= X_k^{(14)} \left(I + R_k + \left(R_k^2 + R_k^3 + R_k^4 + R_k^5\right) \left(I + R_k^4 + R_k^8\right)\right) \\ &= X_k^{(14)} (I + R_k) \left(I + \left(R_k^2 + R_k^4\right) \left(I + R_k^4 + R_k^8\right)\right), \end{aligned} \tag{2.26}$$

and possesses the better computational efficiency index $14^{1/7} \approx 1.45791$.

As could be seen in all the derived methods up to now and the subsequent formulations, the number of matrix products is lower than the order of convergence, i.e. a clear advance in contrast to the standard forms extracted from (1.2) has been obtained.

2.6. *Case of* $p = 15$

The new variant for the case $p = 15$ can be written in what follows:

$$\begin{aligned} X_{k+1}^{(15)} &= X_k^{(15)} \left(I + R_k + R_k^2\right) \left(I + R_k + R_k^2 + R_k^3 + R_k^4\right) \\ &\quad \times \left(I + R_k (R_k^4 - I)(I - R_k^2 + R_k^3)\right), \end{aligned} \tag{2.27}$$

where it hits the computational efficiency index $15^{1/9} \approx 1.3510$.

Another application of the above discussions leads to

$$\begin{aligned} X_{k+1}^{(15)} &= X_k^{(15)} P_k(14) = X_k^{(15)} P_k(2) \left(I + R_k^3 + R_k^6 + R_k^9 + R_k^{12}\right) \\ &= X_k^{(15)} \left(I + R_k + R_k^2\right) \left(I + R_k^3 + R_k^6 + R_k^9 + R_k^{12}\right), \end{aligned} \tag{2.28}$$

whose computational efficiency index is $15^{1/8} \approx 1.4029$.

The best heuristic is initiated by a factorization of (2.27) which can be derived following (2.26) and the principle (2.3). It is given in the form

$$X_{k+1}^{(15)} = X_k^{(15)} \left(I + (R_k + R_k^2) \left(I + \left(R_k^2 + R_k^4\right) \left(I + R_k^4 + R_k^8\right)\right)\right), \tag{2.29}$$

which possesses better computational efficiency index $15^{1/7} \approx 1.47236$.

2.7. *Case of* $p = 16$

For this case, we propose the following new variant

$$\begin{aligned} X_{k+1}^{(16)} &= X_k^{(16)} P_3(k) \left(I + R_k^4 + R_k^8 + R_k^{12}\right) \\ &= X_k^{(16)} \left(I + R_k + R_k^2 + R_k^3\right) \left(I + R_k^4 + R_k^8 + R_k^{12}\right) \\ &= X_k^{(16)} (I + R_k) \left(I + R_k^2\right) \left(I + R_k^4\right) \left(I + R_k^8\right), \end{aligned} \tag{2.30}$$

where its efficiency index is $16^{1/8} \approx 1.4142$.

2.8. Case of $p = 17$

Now we further improve the computational order of convergence using the same number of matrix products as in the previous case. In fact, by applying (2.3) on (2.30), after some algebraic factorizations, we are able to present the following two methods

$$\begin{aligned} X_{k+1}^{(17)} &= X_k^{(17)} \left(I + (R_k + R_k^2) \left(I + R_k^2\right) \left(I + R_k^4\right) \left(I + R_k^8\right)\right) \\ &= X_k^{(17)} \left(I + \left(R_k + R_k^2 + R_k^3 + R_k^4\right) \left(I + R_k^4 + R_k^8 + R_k^{12}\right)\right), \end{aligned} \tag{2.31}$$

whose efficiency index would be $17^{1/8} \approx 1.4279$.

2.9. Case of $p = 18$

In this case, we are able to improve the computational efficiency index even much more than 1.4279 of the previous case using elegant factorizations. We obtain the following alternative

$$\begin{aligned} X_{k+1}^{(18)} &= X_k^{(18)} P_k(17) = X_k^{(18)} P_k(5) \left(I + R_k^6 + R_k^{12}\right) \\ &= X_k^{(18)} (I + R_k) \left(I + R_k^2 + R_k^4\right) \left(I + R_k^6 + R_k^{12}\right), \end{aligned} \tag{2.32}$$

where its efficiency index is $18^{1/8} \approx 1.4351$.

2.10. Case of $p = 19$

As the last studied method in this paper, we consider $p = 19$. After many factorizations, we obtained the following alternative method for the hyperpower iteration

$$X_{k+1}^{(19)} = X_k^{(19)} \left(I + (R_k + R_k^2) \left(I + R_k^2 + R_k^4\right) \left(I + R_k^6 + R_k^{12}\right)\right). \tag{2.33}$$

In (2.33), the order of convergence is 19 while it requires only 8 matrix products per computing cycle to proceed. This means that the computational efficiency index of (2.33) is $19^{1/8} \approx 1.4449$ which is much higher than the case of standard hyperpower iteration, i.e. $19^{1/19} \approx 1.1676$.

We here stop and do not pursue the simplification of higher values of $p \geq 20$. This could be considered for future works.

3. A factorization with higher efficiency

Here we stir up interests to an eminent issue in the development of our formulations. We have found the formulations depicted in Section 2 *without imposing any weights*. It is meant that the coefficient of any matrices involved in the formulations (2.16)–(2.33) are ± 1. This could be called as a *normalized approach*, at which there is no emerging cost for scalar-to-matrix products.

On the other hand, there is a modus operandi in which we could reduce the number of matrix–matrix products, while imposing further scalar-to-matrix multiplications. Following this, the coefficients are not anymore normal and it is thus named as *non-normalized approach*. Such a technique has recently been discussed in [7].

In this section, we state a method of order $16+1$ and the index $17^{1/7} \approx 1.4989$. Although the scheme possesses a higher computational index of efficiency at the first sight, it includes some multiplications of abnormal scalar-to-matrices as discussed below.

This strategy is established upon the fact that a hyperpower formulation of order 2^q+1 ($q \geq 2$) can be re-written as if at least $(q+1)+1$ matrix–matrix multiplications (mmm) are required (see Theorem 2.2). For example, in case of the hyperpower iteration of order 5, we have the following:

$$X_{k+1}^{(5)} = X_k^{(5)}(I + R_k + R_k^2 + R_k^3 + R_k^4), \tag{3.1}$$

and obtain a simple factorization

$$X_{k+1}^{(5)} = X_k^{(5)}(I + R_k + R_k^2 + R_k^2(R_k + R_k^2)), \tag{3.2}$$

which requires only $(q+1)+1 = (2+1)+1 = 4$ mmm and possesses the order $2^q+1 = 5$. We again remark that this is the only method of this type whose coefficients are normal and it possesses the efficiency index $5^{1/4} \approx 1.4953$.

The case of $q = 3$ has been discussed in [7]. Here we focus on the case of $q = 4$ to construct a method of convergence order 17. However, it should be stated that, in practice, obtaining the optimal bound of 6 mmm for such a case is really hard to implement. And we here find a formulation, which requires 7 mmm. Clearly, this would be a more economic formulation than the method (2.31).

Toward this goal, we start by considering the hyperpower iteration of order 17 in what follows:

$$X_{k+1}^{(17)} = X_k^{(17)} P_k(17) = X_k(I + R_k + R_k^2 + \cdots + R_k^{16}). \tag{3.3}$$

This formulation can be simplified as follows (we omit the index k for simplicity in writing only):

Table 3
Values for the parameters γ_i in exact and floating-point arithmetic in (3.4).

Parameters	γ_0	γ_1	γ_2	γ_3	γ_4
Exact values	$\frac{5\,685\,192\,828\,231}{2\,399\,141\,888\,000}$	$\frac{296\,142\,499}{2\,306\,867\,200}$	$\frac{211\,930\,891}{576\,716\,800}$	$\frac{7\,337\,251}{10\,485\,760}$	$4\gamma_3$

$$\begin{aligned} P(R) = &\left(\alpha_0 I + \alpha_1 R + \alpha_2 R^2 + \alpha_3 R^3 + \alpha_4 R^4 + \alpha_5 R^5 + \alpha_6 R^6 + (1/2)R^7 + R^8\right) \\ &\times \left(\beta_0 I + \beta_1 R + \beta_2 R^2 + \beta_3 R^3 + \beta_4 R^4 + \beta_5 R^5 + \beta_6 R^6 + (1/2)R^7 + R^8\right) \\ &+ \gamma_0 I + \gamma_1 R + \gamma_2 R^2 + \gamma_3 R^3 + \gamma_4 R^4, \end{aligned} \tag{3.4}$$

where α_i, β_i and γ_i (for any i) are some real constants that must be found. Taking into account

$$\beta_6 = \alpha_6,\ \beta_5 = \alpha_5,\ \beta_3 = \alpha_3,\ \gamma_4 = 4\gamma_3, \tag{3.5}$$

we come by to a system of fifteen nonlinear equations with fifteen unknowns which must be solved symbolically in a computer algebra system (CAS). Doing this, yields to a solution, e.g., $\alpha_0 = \frac{9295}{16\,384} - \frac{690\,969\sqrt{\frac{17}{715}}}{81\,920}$ and $\beta_0 = \frac{690\,969\sqrt{\frac{17}{715}}}{81\,920} + \frac{9295}{16\,384}$. Some of the important results are given in Table 3.

Now, the second aim is to reduce the number of mmm involved in the process of computation of matrix polynomials

$$\begin{aligned} Q(R) &= \alpha_0 I + \alpha_1 R + \alpha_2 R^2 + \alpha_3 R^3 + \alpha_4 R^4 + \alpha_5 R^5 + \alpha_6 R^6 + (1/2)R^7 + R^8 \\ T(R) &= \beta_0 I + \beta_1 R + \beta_2 R^2 + \beta_3 R^3 + \beta_4 R^4 + \beta_5 R^5 + \beta_6 R^6 + (1/2)R^7 + R^8. \end{aligned}$$

Therefore, we again simplify the procedure by a similar approach as above and by finding the appropriate real constants:

$$\begin{aligned} Q(R) = &\left(I + \delta_1 R + \delta_2 R^2 + 1/4R^3 + R^4\right)\left(I + \zeta_1 R + \zeta_2 R^2 + 1/4R^3 + R^4\right) \\ &+ \eta_0 I + \eta_1 R + \eta_2 R^2, \end{aligned} \tag{3.6}$$

wherein δ_i, ζ_i and η_i (for any i) are real constants that must be found. Following a similar methodology as above and taking into consideration $\eta_0 = \alpha_0 - 1$, we have a system of six nonlinear equations and six unknowns. Solving this gives the results illustrated in Tables 4–6. Similarly for $T(R)$, we have

$$\begin{aligned} T(R) = &\left(I + \theta_1 R + \theta_2 R^2 + 1/4R^3 + R^4\right)\left(I + \vartheta_1 R + \vartheta_2 R^2 + 1/4R^3 + R^4\right) \\ &+ \kappa_0 I + \kappa_1 R + \kappa_2 R^2, \end{aligned} \tag{3.7}$$

wherein the unknowns are summarized in Tables 7–9 while $\kappa_0 = \beta_0 - 1$.

Table 4
Values for the parameters δ_i, in exact and floating-point arithmetic in (3.6).

Parameters	δ_1	δ_2
Exact values	$\frac{5}{128}\left(3-\frac{119}{\sqrt{1853+8\sqrt{12\,155}}}\right)$	$\frac{1}{32}\left(5-\sqrt{1853+8\sqrt{12\,155}}\right)$

Table 5
Values for the parameters ζ_i, in exact and floating-point arithmetic in (3.6).

Parameters	ζ_1	ζ_2
Exact values	$\frac{5}{128}\left(3+\frac{119}{\sqrt{1853+8\sqrt{12\,155}}}\right)$	$\frac{1}{32}\left(\sqrt{1853+8\sqrt{12\,155}}+5\right)$

Table 6
Values for the parameters η_i, in exact and floating-point arithmetic in (3.6).

Parameters	η_1	η_2
Exact values	$\frac{3(83\sqrt{12\,155}-935)}{112\,640}$	$\frac{4\,165\,826\sqrt{12\,155}-273\,766\,385}{3\,199\,324\,160}$

Table 7
Values for the parameters θ_i, in exact and floating-point arithmetic in (3.6).

Parameters	θ_1	θ_2
Exact values	$\frac{5}{128}\left(3+\frac{119}{\sqrt{1853-8\sqrt{12\,155}}}\right)$	$\frac{1}{32}\left(\sqrt{1853-8\sqrt{12\,155}}+5\right)$

Table 8
Values for the parameters ϑ_i, in exact and floating-point arithmetic in (3.6).

Parameters	ϑ_1	ϑ_2
Exact values	$\frac{5}{128}\left(3-\frac{119}{\sqrt{1853-8\sqrt{12\,155}}}\right)$	$\frac{1}{32}\left(5-\sqrt{1853-8\sqrt{12\,155}}\right)$

Table 9
Values for the parameters κ_i, in exact and floating-point arithmetic in (3.7).

Parameters	κ_1	κ_2
Exact values	$-\frac{3(935+83\sqrt{12\,155})}{112\,640}$	$\frac{-273\,766\,385-4\,165\,826\sqrt{12\,155}}{3\,199\,324\,160}$

Accordingly, the proposed 17th-order formulation (denoted by APM17) can be implemented using the notations $Q(R_k)=S(R_k)D(R_k)+\eta_0 I+\eta_1 R_k+\eta_2 R_k^2$ and $T(R_k)=L(R_k)J(R_k)+\kappa_0 I+\kappa_1 R_k+\kappa_2 R_k^2$, as follows:

$$X_{k+1}^{(17)}=X_k^{(17)}\left[Q(R_k)T(R_k)+\gamma_0 I+\gamma_1 R_k+\gamma_2 R_k^2+\gamma_4 R_k^2\left(\frac{1}{4}R_k+R_k^2\right)\right], \quad (3.8)$$

whereas the matrix $R_k^2(\frac{1}{4}R_k+R_k^2)$ must be computed once per cycle and used each time in $S(R_k)$, $D(R_k)$, $L(R_k)$ and $J(R_k)$. In addition, the method must be implemented using the coefficients obtained by exact arithmetic to clearly reveal the super-fast convergence speed.

The proposed formulation (3.8) is interesting since it reaches the convergence order 17 by only 7 mmm, and beats the efficiency index $17^{1/7} \approx 1.4989$.

4. Numerical aspects

Although the computational indices used in Section 2 revealed the superiority of the proposed formulations in both theoretical and computational aspects, it is required to put on show the high convergence speed of the proposed variants for finding outer inverses. To this aim, in this section we consider a theoretical example and apply different new formulations of Section 2 for finding outer inverses.

The programming package Mathematica [15] is used in the implementation of the methods.

For numerical comparisons we have used the methods PM10, PM11, PM12, PM13, PM14, PM15, PM16, PM17, PM18, and PM19 corresponding to (2.17), (2.18), (2.19), (2.23), (2.26), (2.29), (2.30), (2.31), (2.32), and (2.33), respectively.

We note that we used 1500 fixed point digits in our calculations to *clearly* show the high convergence rate of the proposed formulations. Although in all practical problems the machine precision (double precision) is enough, here our focus is to support the theoretical findings of the previous sections.

For the sake of completeness, we restate definition of the Moore–Penrose inverse (for more details see [3]). The Moore–Penrose inverse $A^\dagger \in \mathbb{C}^{n\times m}$ of the matrix $A \in \mathbb{C}^{m\times n}$ is the unique matrix satisfying the following four equations

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

where $(.)^*$ represents the conjugate transpose.

Example 4.1 *(Academical test).* We consider an academical example $A = \begin{pmatrix} 1 & 0 & 0 & -6 \\ 2 & 6 & 0 & -6 \\ 7 & 8 & 9 & -6 \end{pmatrix}$, where its exact Moore–Penrose inverse is

$$A^\dagger = \begin{pmatrix} \frac{28}{1931} & -\frac{143}{3862} & \frac{84}{1931} \\ -\frac{653}{3862} & \frac{1335}{7724} & -\frac{14}{1931} \\ \frac{57}{1931} & -\frac{249}{1931} & \frac{171}{1931} \\ -\frac{1903}{11\,586} & -\frac{143}{23\,172} & \frac{14}{1931} \end{pmatrix}.$$

The results of comparisons are given in Table 10 using the stopping criterion $\|X_k - A^\dagger\|_2 \leq \epsilon = 10^{-100}$. The initial matrix is $X_0 = \frac{1}{\sigma_1^2}A^*$, where σ_1 denotes the largest singular value of A.

We present the formula of computational order of convergence for finding outer inverses in what follows [14]:

Table 10
Results of comparisons for Example 4.1.

Methods	$\|X_1 - A^\dagger\|$	$\|X_2 - A^\dagger\|$	$\|X_3 - A^\dagger\|$	$\|X_4 - A^\dagger\|$	ρ	LEI
PM10	1.96753×10^{-1}	5.14812×10^{-3}	7.74329×10^{-19}	4.58879×10^{-177}	10.	0.3837
PM11	1.88947×10^{-1}	2.20015×10^{-3}	1.17403×10^{-24}	1.17271×10^{-258}	11.	0.3425
PM12	1.81451×10^{-1}	8.67148×10^{-4}	1.23055×10^{-31}	8.20721×10^{-366}	12.	0.3549
PM13	1.74252×10^{-1}	3.1519×10^{-4}	6.99366×10^{-40}	2.21009×10^{-503}	13.	0.3664
PM14	1.67339×10^{-1}	1.05655×10^{-4}	1.69046×10^{-49}	1.21785×10^{-676}	14.	0.3770
PM15	1.607×10^{-1}	3.26624×10^{-5}	1.36307×10^{-60}	$2.764213 \times 10^{-891}$	15.	0.3868
PM16	1.54325×10^{-1}	9.31201×10^{-6}	2.8758×10^{-73}	$1.96886 \times 10^{-1153}$	16.	0.3465
PM17	1.48202×10^{-1}	2.44837×10^{-6}	1.24521×10^{-87}	$1.26903 \times 10^{-1469}$	17.	0.3541
APM17	1.48202×10^{-1}	2.44837×10^{-6}	1.24521×10^{-87}	$1.26903 \times 10^{-1469}$	17.	0.4047
PM18	1.42323×10^{-1}	5.93675×10^{-7}	8.67931×10^{-104}	0.	18.	0.3612
PM19	1.36676×10^{-1}	1.32757×10^{-7}	7.63836×10^{-122}	0.	19.	0.3680

$$\rho = \frac{\ln\left(\frac{\|X_{k+1} - A^{(2)}_{T,S}\|}{\|X_k - A^{(2)}_{T,S}\|}\right)}{\ln\left(\frac{\|X_k - A^{(2)}_{T,S}\|}{\|X_{k-1} - A^{(2)}_{T,S}\|}\right)}, \tag{4.1}$$

wherein $\|\cdot\|$ denotes the 2-norm.

The calculated value ρ estimates the theoretical order of convergence well when a pathological behavior of the iterative method (for instance, slow convergence at the beginning of the implemented iterative method, etc.) does not exist.

From Table 10, we can see that the computational order of convergence overwhelmingly supports the theoretical order of convergence.

The logarithms of efficiency indices are given in the last column of Table 10. They manifest that the case of $p = 15$ is the most attractive formulation and advance of the hyperpower iteration, using the normalized approach.

For $A \in \mathbb{C}^{n\times n}$, the smallest nonnegative integer k satisfying $\text{rank}(A^{k+1}) = \text{rank}(A^k)$ is called the index of A, and it is denoted by $\text{ind}(A) = k$. Then the matrix $X \in \mathbb{C}^{n\times n}$ satisfying

$$(1^k) \quad A^{l+1}X = A^l,\ l \geq \text{ind}(A), \quad (2) \quad XAX = X, \quad (5) \quad AX = XA$$

is called the Drazin inverse of A, and it is denoted by $X = A^D$ (see, for example, [3]).

Example 4.2. Let consider the matrix $A = \begin{pmatrix} -2 & 1 & 0 \\ 4 & -2 & 1 \\ -8 & 4 & -2 \end{pmatrix}$, whose exact Drazin inverse is

$$A^D = A^l \left(A^{2l+1}\right)^\dagger A^l = \begin{pmatrix} -1 & \frac{1}{2} & \frac{3}{8} \\ -1 & \frac{1}{2} & \frac{1}{2} \\ 2 & -1 & -1 \end{pmatrix}.$$

Table 11
Results of comparisons for Example 4.2.

Methods	$\|X_1 - A^D\|$	$\|X_2 - A^D\|$	$\|X_3 - A^D\|$	$\|X_4 - A^D\|$	ρ
PM10	3.13419	2.53663	0.231142	9.123449×10^{-12}	10.
PM11	3.13666	2.39873	0.0957746	3.937338×10^{-17}	11.
PM12	3.1378	2.25629	0.0332911	$3.5442617 \times 10^{-24}$	12.
PM13	3.13777	2.11103	0.00955377	3.191607×10^{-33}	13.
PM14	3.13671	1.96464	0.00222769	$1.2937808 \times 10^{-44}$	14.
PM15	3.13475	1.8187	0.000415369	9.966513×10^{-59}	15.
PM16	3.13201	1.67465	0.0000609502	5.777995×10^{-76}	16.
PM17	3.12858	1.53383	6.926956×10^{-6}	9.365678×10^{-97}	17
PM18	3.12455	1.39739	6.00067×10^{-7}	$1.4793221 \times 10^{-121}$	18.
PM19	3.12	1.26633	3.8995×10^{-8}	$7.4445347 \times 10^{-151}$	19.

The results presented in Table 11 are obtained using the stopping criterion $\|X_k - A^D\|_2 \leq \epsilon = 10^{-95}$ and the initial matrix defined by $X_0 = \frac{1}{2\|A\|^l}$, $l = \text{ind}(A)$.

According to the results arranged in Table 11, one can observe that the computational order of convergence also supports the theoretical order of convergence.

5. Discussion

The hyperpower family was originally introduced by Ben-Israel in [2]. It was based on Schulz's method and was investigated by numerous authors such as [16]. In this paper, we have constructed some new formulations for the hyperpower iteration for values $10 \leq p \leq 19$.

It has been shown that the new formulations are convergent and possess higher computational efficiency indices in contrast to the standard forms. They have been derived such that one needs fewer number of matrix products per full cycle.

We moreover discussed on the non-normalized approach of deriving hyper-power schemes. In this way, we have derived a method of order 17 possessing the efficiency index 1.4989.

Acknowledgements

The second author gratefully acknowledge support from the Research Project 174013 of the Serbian Ministry of Science. The authors are grateful to anonymous reviewers for careful reading and valuable comments which improved the quality of the paper.

References

[1] M. Altman, An optimum cubically convergent iterative method of inverting a linear bounded operator in Hilbert space, Pacific J. Math. 10 (1960) 1107–1113.
[2] A. Ben-Israel, An iterative method for computing the generalized inverse of an arbitrary matrix, Math. Comp. 19 (1965) 452–455.
[3] A. Ben-Israel, T.N.E. Greville, Generalized Inverses, second ed., Springer, New York, 2003.
[4] J.-J. Climent, N. Thome, Y. Wei, A geometrical approach on generalized inverses by Neumann-type series, Linear Algebra Appl. 332–334 (2001) 533–540.

[5] X. Liu, H. Jin, Y. Yu, Higher-order convergent iterative method for computing the generalized inverse and its application to Toeplitz matrices, Linear Algebra Appl. 439 (2013) 1635–1650.
[6] A.M. Ostrowski, Solutions of Equations and Systems of Equations, Academic Press, New York, London, 1966.
[7] M.D. Petković, M.S. Petković, Hyper-power methods for the computation of outer inverses, J. Comput. Appl. Math. 278 (2015) 110–118.
[8] M. Sharifi, M. Arab, F. Khaksar Haghani, Finding generalized inverses by a fast and efficient numerical method, J. Comput. Appl. Math. 279 (2015) 187–191.
[9] P.S. Stanimirović, Self-correcting iterative methods for computing {2}-inverses, Arch. Math. 39 (2003) 27–36.
[10] P.S. Stanimirović, F. Soleymani, A class of numerical algorithms for computing outer inverses, J. Comput. Appl. Math. 263 (2014) 236–245.
[11] F. Soleymani, An efficient and stable Newton-type iterative method for computing generalized inverse $A^{(2)}_{T,S}$, Numer. Algorithms 69 (3) (2015) 569–578.
[12] F. Soleymani, A fast convergent iterative solver for approximate inverse of matrices, Numer. Linear Algebra Appl. 21 (2014) 439–452.
[13] 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.
[14] F. Soleymani, On finding robust approximate inverses for large sparse matrices, Linear Multilinear Algebra 62 (2014) 1314–1334.
[15] M. Trott, The Mathematica Guidebook for Numerics, Springer, New York, NY, USA, 2006.
[16] C. Xu-Zhou, R.E. Hartwig, The hyperpower iteration revisited, Linear Algebra Appl. 233 (1996) 207–229.