Journal of Computational and Applied Mathematics 344 (2018) 572–584

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Rapid generalized Schultz iterative methods for the computation of outer inverses

Marko D. Petković [*], Mihailo A. Krstić, Kostadin P. Rajković

University of Niš, Faculty of Sciences and Mathematics, Višegradska 33, 18000 Niš, Serbia

ARTICLE INFO

Article history:
Received 30 April 2016
Received in revised form 30 January 2018

MSC:
15A09
47J25

Keywords:
Moore–Penrose inverse
Drazin inverse
Outer inverse
Iterative methods
Hyper-power methods
Convergence

ABSTRACT

We present a general scheme for the construction of new efficient generalized Schultz iterative methods for computing the inverse matrix and various matrix generalized inverses. These methods have the form $X_{k+1} = X_k p(AX_k)$, where A is $m \times n$ complex matrix and $p(x)$ is a polynomial. The construction procedure is general and can be applied to any number of matrix multiplications per iteration, denoted by θ. We use it to construct new methods for $\theta = 6$ matrix multiplications per iteration having (up to now) the highest computational efficiency among all other known methods. They are compared to several existing ones on a series of numerical tests. Finally, the numerical instability and the influence of roundoff errors is studied for an arbitrary generalized Schultz iterative method. These results are applicable to all considered new and existing particular iterative methods.

1. Introduction

Assume that $A \in \mathbb{C}^{m\times n}$ is a given matrix, while T and S are subspaces of $\mathbb{C}^n$ and $\mathbb{C}^m$, respectively, satisfying $AT \oplus S = \mathbb{C}^m$. In such a case, there exists a unique matrix $X = A^{(2)}_{T,S} \in \mathbb{C}^{n\times m}$ such that $XAX = X$, $\mathcal{R}(X) = T$ and $\mathcal{N}(X) = S$. Here, $\mathcal{R}(X)$ and $\mathcal{N}(X)$ denote the range and null-space of the matrix X. This unique matrix $X = A^{(2)}_{T,S}$ is known as *the outer inverse with prescribed range and null-space*. If $G \in \mathbb{C}^{n\times m}$ is a matrix such that $T = \mathcal{R}(G)$ and $S = \mathcal{N}(G)$, then the matrix $X = A^{(2)}_{T,S}$ is called *the outer G-inverse* or, simply, *the G-inverse*.

Taking some specific values of G, the matrix X is reduced to several well-known generalized inverses. For example, taking $G = A^*$ we obtain the *Moore–Penrose* inverse $X = A^\dagger$ satisfying the following set of Penrose equations:

$$(1)\ AXA = A, \quad (2)\, XAX = X, \quad (3)\ (AX)^* = AX, \quad (4)\ (XA)^* = XA.$$

When $m = n$ and A is regular, X is further reduced to the standard inverse matrix $X = A^{-1}$.

Assume that $m = n$ and $l = \mathrm{ind}(A)$ is the smallest integer such that $\mathrm{rank}(A^{l+1}) = \mathrm{rank}(A^l)$. Then, taking $G = A^l$ we get the *Drazin inverse* $X = A^D$, which is a unique solution to the following set of equations:

$$(1^l)\ A^lXA = A^l, \quad (2)\, XAX = X, \quad (5)\, AX = XA.$$

In the available literature, there are various methods for computing generalized inverses. Direct methods include SVD (Singular Value Decomposition) based methods, QR factorization [1], etc. On the other hand, a large number of different iterative methods have been studied for computing all kinds of generalized inverses.

* Corresponding author.
E-mail addresses: dexterofnis@gmail.com (M.D. Petković), mihailo1994@yahoo.com (M.A. Krstić), kosta.rajkovic@gmail.com (K.P. Rajković).

https://doi.org/10.1016/j.cam.2018.05.048
0377-0427/© 2018 Elsevier B.V. All rights reserved.

Generalizations of the well-known Schultz method

$$X_{k+1} = X_k(2I - AX_k) \tag{1.1}$$

for computing Moore–Penrose, Drazin and other outer inverses are given in [2–4], respectively. Hereafter I denotes the identity matrix of appropriate size. After that, a vast number of different iterative methods appeared in the literature, including [5–8]. All these methods are of the same type

$$X_{k+1} = X_k p(AX_k) \tag{1.2}$$

where $p(x)$ is a certain polynomial. They are known as *generalized Schultz iterative methods*. Putting $p(x) = 1 + (1-x) + \cdots + (1-x)^{r-1}$, one also gets the well-known hyper-power method of the order r given by

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

A systematic approach to the convergence analysis of all generalized Schultz iterative methods is adopted by Petković in [9]. The following corollary follows directly from Theorem 2.1 in [9] and provides necessary conditions for the convergence and the convergence order $r > 1$ of an arbitrary generalized Schultz iterative method of the form (1.2):

Corollary 1.1 ([9]). *The matrix method $X_{k+1} = X_k p(AX_k)$, $\deg p(x) = d$ converges to $X = A^{(2)}_{\mathcal{R}(G),\mathcal{N}(G)}$ and has the convergence order $r > 1$ if $\mathcal{N}(X_0) = \mathcal{N}(G)$, $\mathcal{R}(X_0) = \mathcal{R}(G)$, $\rho(AX - AX_0) < 1$ and*

$$\bar{p}(x) = p(1-x) = 1 + x + x^2 + \cdots + x^{r-1} + x^r v(x) \tag{1.4}$$

for some polynomial $v(x)$ of degree $d - r$.

In what follows, we always assume that $\bar{p}(x) = p(1-x)$. In order to estimate the performance and effectiveness of a particular method IM, the following value is used

$$E_c(IM) = r^{1/\theta} \tag{1.5}$$

where r is the convergence order and θ is the number of matrix multiplications per iteration. This value is known as *the computational efficiency* of the method IM. It was introduced by Traub [10] and is directly proportional to the total running time of the method (i.e. its computer implementation) required to obtain the result of the desired precision. Note that the Schultz method (1.1) has the computational efficiency $E_c((1.1)) = 2^{1/2}$, while the hyper-power method has $E_c((1.3)) = r^{1/r}$, which is maximized for $r = 3$. The corresponding third order hyper-power method

$$X_{k+1} = X_k(I + R_k + R_k^2), \qquad R_k = I - AX_k \tag{1.6}$$

will be henceforth referred to as **HP3**.

Given the generalized Schultz iterative method $X_{k+1} = X_k\bar{p}(R_k)$, the goal is to compute $\bar{p}(R_k)$ with the fewest matrix multiplications possible. It is equivalent to the computation of the polynomial $\bar{p}(x)$ with the smallest number of polynomial multiplication operations. This problem need not be solved for an arbitrary polynomial $\bar{p}(x)$. Indeed, one is to find the polynomial providing the maximum convergence order r, when θ is fixed. In other words, given an integer $\theta \geq 4$, one needs to find a polynomial $\bar{p}(x)$, which can be computed by θ polynomial multiplication operations and has the maximum possible number of first coefficients equal to 1.

In the paper [11], the previously stated problem for $\theta = 4, 5$ was solved, and two of the up to now most efficient high-order iterative methods were provided. The first is given by

$$\begin{aligned} R_k &= I - A\cdot X_k, \quad S_k = R_k\cdot R_k,\\ M_k &= I + R_k + S_k\cdot(I + R_k + S_k),\\ X_{k+1} &= X_k\cdot M_k \end{aligned} \tag{1.7}$$

having the order of convergence $r = 5$ with a total of $\theta = 4$ matrix multiplications per iteration. The second method is given by

$$\begin{aligned} R_k &= I - A\cdot X_k, \quad S_k = R_k\cdot R_k,\\ M_k &= \tfrac{7}{8}R_k + S_k\cdot\left(\tfrac{1}{2}R_k + S_k\right),\\ N_k &= \tfrac{11}{16}I - \frac{9}{8}R_k + \tfrac{3}{4}S_k + M_k,\\ T_k &= I + \tfrac{51}{128}R_k + \tfrac{39}{32}S_k + M_k\cdot N_k,\\ X_{k+1} &= X_k\cdot T_k, \end{aligned} \tag{1.8}$$

and has $r = 9$ with $\theta = 5$. These methods have the maximum possible convergence order with $\theta = 4$ and $\theta = 5$ matrix multiplications per iteration. Both methods are equivalent to the hyper-power methods of the corresponding convergence order. Henceforward, we denote these methods by **IHP5** and **IHP9**.

Furthermore, a recently published paper [12] deals with the following generalization of the methods **IHP5** and **IHP9**, having $r = 17$ but with $\theta = 7$ matrix multiplications per iteration:

$$\begin{aligned} R_k &= I - A\cdot X_k, \quad S_k = R_k\cdot R_k, \quad M_k = S_k\cdot\left(\tfrac{1}{4}R_k + S_k\right),\\ Q_k &= (I+\delta_1R_k+\delta_2S_k+M_k)\cdot(I+\zeta_1R_k+\zeta_2S_k+M_k)+\eta_0I+\eta_1R_k+\eta_2S_k,\\ T_k &= (I+\theta_1R_k+\theta_2S_k+M_k)\cdot(I+\nu_1R_k+\nu_2S_k+M_k)+\kappa_0I+\kappa_1R_k+\kappa_2S_k,\\ X_{k+1} &= X_k\cdot(Q_k\cdot T_k+\gamma_0I+\gamma_1R_k+\gamma_2S_k+\gamma_4M_k). \end{aligned} \tag{1.9}$$

The values of all constants are given in [12]. The method in (1.9) will be denoted by **IHP17**.

The construction of all previously introduced iterative methods was independent for each particular method. In this paper, we present the general construction procedure, directly applicable to an arbitrary number of matrix multiplications per iteration θ. That scheme is described and analyzed in Section 2. It is then used in Section 3 to construct the optimal convergence order methods for $\theta = 6$. In Section 4 the results of different numerical tests are presented and the new methods are compared with the existing ones. Section 5 contains a theoretical analysis of the arbitrary generalized Schultz iterative method from [9] and its possible numerical instability issues. Finally, Section 6 concludes the paper.

2. General scheme for computing $\bar{p}(R_k)$

The main idea behind the methods **IHP5** and **IHP9** is to compute $I + R_k + R_k^2 + R_k^3 + R_k^4$ and $I + R_k + R_k^2 + \cdots + R_k^8$ with the fewest matrix multiplications possible. These are (matrix) values of the 4th and the 8th degree polynomial, so the smallest possible number of matrix multiplications was $\log_2 4 = 2$ and $\log_2 8 = 3$. Similarly, to evaluate the matrix value of the 2^{m-1}-degree polynomial $p(x)$, one needs at least $m-1$ matrix multiplication operations.

Our aim is to construct a general recurrent scheme for performing such a computation. It generates the sequence $(u_i(x))_{0\le i\le m}$ where $u_0(x) = 1$, $u_1(x) = x$, $u_i(x)$ is a 2^{i-1}-degree monic polynomial ($i \ge 1$) and $u_m(x) = p(x)$. Each $u_i(x)$ ($i = 2, 3, \dots, m$) is computed using $u_0(x), u_1(x), \dots, u_{i-1}(x)$ and exactly one (polynomial) multiplication operation.

The starting values are $u_0(x) = 1$, $u_1(x) = x$ and $u_2(x) = x^2$ since every second degree polynomial requires that one polynomial multiplication be computed and that it be computed as a linear combination of 1, x and x^2.

Assume that $i \ge 3$ and that we have already computed $u_0(x), u_1(x), \dots, u_{i-1}(x)$ with a total of $i-1$ multiplications. Without loss of generality, we may assume that in the ith multiplication step, $u_{i-1}(x)$ is multiplied by a linear combination of $u_0(x), u_1(x), \dots, u_{i-1}(x)$. The product is further modified by a certain linear combination of $u_0(x), u_1(x), \dots, u_{i-2}(x)$.

Therefore, the polynomial $u_i(x)$ is given by

$$\begin{aligned} u_i(x) = u_{i-1}(x)\cdot\big[&u_{i-1}(x) + a_{i,i-2}u_{i-2}(x) + \cdots + a_{i,0}u_0(x)\big]\\ &+ b_{i,i-2}u_{i-2}(x) + b_{i,i-3}u_{i-3}(x) + \cdots + b_{i,0}u_0(x) \end{aligned} \tag{2.1}$$

where $a_{i,0}, a_{i,1}, \dots, a_{i,i-2}$ and $b_{i,0}, b_{i,1}, \dots, b_{i,i-2}$ are unknown coefficients which have to be determined.

It is not difficult to see that the scheme (2.1) ($i = 3, 4, \dots, m$), together with the initial values $u_0(x) = 1$, $u_1(x) = x$ and $u_2(x) = x^2$, represents the most general $m-1$ polynomial multiplication scheme for computing a monic 2^{m-1}-degree polynomial $u_m(x)$. This scheme yields the construction of the generalized Schultz iterative matrix method with θ multiplications per iteration in the following way:

$$X_{k+1} = X_k\cdot c\cdot u_{\theta-1}(R_k), \qquad R_k = I - AX_k. \tag{2.2}$$

In other words, we choose $\bar{p}(x) = c\cdot u_{\theta-1}(x)$. Recall that two matrix multiplications are spent on computing $R_k = I - A\cdot X_k$ and $X_{k+1} = X_k\cdot\bar{p}(R_k)$. An additional constant c is required since the polynomial $\bar{p}(x)$ need not necessarily be monic. The unknown coefficients $a_{i,j}, b_{i,j}$ ($3 \le i \le \theta-1$ and $0 \le j \le i-2$) and c need to be determined such that the method has the highest convergence order r. In other words, we have to determine these coefficients such that

$$cu_{\theta-1}(x) = 1 + x + \cdots + x^{r-1} + x^r\hat{u}_{\theta-1}(x)$$

where r is as highest as possible and $\hat{u}_{\theta-1}(0) \ne 1$.

The complete algorithmic procedure for computing the unknown coefficients $a_{i,j}, b_{i,j}$ ($3 \le i \le \theta-1$ and $0 \le j \le i-2$) and c is summarized in Algorithm 2.1 .

Algorithm 2.1 Construction of the generalized Schultz iterative method with θ matrix multiplications per iteration.

```
1: Consider $c, x, a_{i,j}$ and $b_{i,j}$ as symbolic variables.
2: Set $u_0(x) := 1$, $u_1(x) := x$ and $u_2(x) := x^2$.
3: for $i := 3$ to $\theta - 1$ do
4:   $p(x) :=$ Expand $\left(u_{i-1}(x)\cdot\left[u_{i-1}(x) + a_{i,i-2}u_{i-2}(x) + \ldots + a_{i,0}u_0(x)\right]\right)$
5:   $u_i(x) := p(x) + b_{i,i-2}u_{i-2}(x) + b_{i,i-3}u_{i-3}(x) + \ldots + b_{i,0}u_0(x)$
6: end for
7: Find the maximum $r$ such that the system of polynomial equations, formed by equating the coefficients with
   $1, x, \dots, x^{r-1}$ in $c\cdot u_{\theta-1}(x)$ to 1, has a solution.
8: Return the unknown coefficients $a_{i,j}, b_{i,j}$ ($3 \le i \le \theta-1$ and $0 \le j \le i-2$) and $c$.
```

The best environment for the implementation of Algorithm 2.1 is any computer algebra system (CAS), such as `Mathematica`, `Maple`, `Maxima`, etc. The function `Expand` expands the polynomial expression given as a parameter and collects the terms involving the same powers of x. The variants of this function are available in all well-known CAS programs. Also, one can use any method for solving the system of polynomial equations (Groebner basis, minimization methods, etc.) and, practically, any solution is acceptable.

Note that both (1.7) and (1.8) can be written as a scheme of the type (2.2) for $\theta = 4, 5$. They have maximum possible values of the convergence order $r = 5, 9$.

3. Methods with $\theta = 6$ multiplications per iteration

Our next goal is to consider the scheme (2.2) for $\theta = 6$ matrix multiplications per iteration and to find the maximum possible convergence order. This scheme has the form

$$X_{k+1} = X_k \cdot c \cdot u_5(R_k), \qquad R_k = I - AX_k \tag{3.3}$$

where

$$\begin{aligned} u_5(x) &= u_4(x)(u_4(x) + a_{5,3}u_3(x) + a_{5,2}x^2 + a_{5,1}x + a_{5,0}) \\ &\quad + b_{5,3}u_3(x) + b_{5,2}x^2 + b_{5,1}x + b_{5,0} \\ u_4(x) &= u_3(x)(u_3(x) + a_{4,2}x^2 + a_{4,1}x + a_{4,0}) \\ &\quad + b_{4,2}x^2 + b_{4,1}x + b_{4,0} \\ u_3(x) &= x^2(x^2 + a_{3,1}x + a_{3,0}) + b_{3,1}x + b_{3,0}. \end{aligned} \tag{3.4}$$

Expanding all products in (3.4) yields

$$u_5(x) = u_{5,0} + u_{5,1}x + u_{5,2}x^2 + \cdots + u_{5,15}x^{15} + x^{16}$$

where $u_{5,l}$ $(0 \le l \le 15)$ are multi-variable polynomials of $a_{i,j}$ and $b_{i,j}$ $(i = 3, 4, 5, b = 0, 1, \ldots, i-2)$. We need to determine the unknown coefficients $a_{i,j}$, $b_{i,j}$ and c such that

$$u_{5,l} = c^{-1}, \qquad l = 0, 1, \ldots, r-1 \tag{3.5}$$

and r is maximal. Using the Groebner basis computation, it is shown that the system (3.5) does not have a solution for $r = 17$ and $r = 16$. However, for $r = 15$, the system has many solutions and one of them is given by:

$$\begin{aligned} a_{3,0} &= 0.645082922061461 & b_{3,0} &= 0.43532078627935 \\ a_{3,1} &= 1.058661594262500 & b_{3,1} &= 0.22632676803681 \\ a_{4,0} &= 0.050654987162504 & b_{4,0} &= 0.42563167485906 \\ a_{4,1} &= 0.345901887114617 & b_{4,1} &= -0.75682522665618 \\ a_{4,2} &= -1.202519413928960 & b_{4,2} &= -1.62230203118978 \\ a_{5,0} &= 1.274524208649420 & b_{5,0} &= 2.72356048720756 \\ a_{5,1} &= 1.799910818770400 & b_{5,1} &= 5.02982915810813 \\ a_{5,2} &= 5.095088450188020 & b_{5,2} &= 2.63710149976585 \\ a_{5,3} &= -1.149108904227180 & b_{5,3} &= 7.52764810605388 \\ c &= 0.144930075923808. \end{aligned} \tag{3.6}$$

We also consider the special case of the system (3.5) when $c = 1$. In this case, the maximum value of r such that the system has a solution is $r = 14$, and the solution is given by

$$\begin{aligned} a_{3,0} &= 0.589305851677216 & b_{3,0} &= 0.13694492627385 \\ a_{3,1} &= -0.038317189491436 & b_{3,1} &= -0.24959247268375 \\ a_{4,0} &= 0.716088325159338 & b_{4,0} &= 0.31648994681425 \\ a_{4,1} &= 0.994592232369608 & b_{4,1} &= -0.20293695866733 \\ a_{4,2} &= -1.219543968940840 & b_{4,2} &= 0.73867616667272 \\ a_{5,0} &= -0.612715355555756 & b_{5,0} &= 0.99257143402746 \\ a_{5,1} &= 1.174304135325600 & b_{5,1} &= 0.72071414437193 \\ a_{5,2} &= -0.983452829557211 & b_{5,2} &= 1.10991297244531 \\ a_{5,3} &= -0.124571668920262 & b_{5,3} &= 0.67588545838602. \end{aligned} \tag{3.7}$$

Table 1
Exact and approximate computational efficiencies $E_c = r^{1/\theta}$, as well as the asymptotic storage complexities of the considered generalized Schultz iterative methods.

Method	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**
E_c	$3^{1/3}$	$5^{1/4}$	$9^{1/5}$	$14^{1/6}$	$15^{1/6}$	$17^{1/7}$
approx. E_c	1.4422	1.4953	1.5518	1.5525	1.5704	1.4989
Asymp. stor. compl.	$3m^2 + nm$	$4m^2 + nm$	$5m^2 + nm$	$6m^2 + nm$	$6m^2 + nm$	$6m^2 + nm$

The solutions (3.6) and (3.7), together with the expressions (3.3) and (3.4), define two generalized Schultz iterative methods of orders $r = 15$ and $r = 14$ using $\theta = 6$ matrix multiplications per iteration. In what follows, we refer to these methods as **IHP15** and **IHP14**.

4. Numerical results and performance testing

Recall that the computational efficiency E_c of the matrix iterative method is defined by $E_c = r^{1/\theta}$, where r is the convergence order of the method and θ is the number of matrix multiplications per iteration. The computational efficiency is proportional to the total number of matrix multiplications required to compute a generalized inverse to desired accuracy. Moreover, it is also asymptotically proportional to the running time of the method implementation, having in mind that the matrix multiplication operation is more computationally expensive than addition, subtraction, and multiplication by a scalar.

Let us mention that the method with θ matrix multiplications per iteration, produced by Algorithm 2.1 , needs to store exactly $\theta - 1$ temporary matrices of the size $m \times m$ per iteration. These matrices are: $R_k = I - A \cdot X_k$, $S_k = R_k \cdot R_k$, $u_3(R_k), \ldots, u_{\theta-1}(R_k)$. Additionally, one needs to store the matrix X_k of the size $n \times m$. The total storage complexity of the corresponding scheme is $\mathcal{O}((\theta - 1)m^2 + nm)$.

Table 1 shows the computational efficiencies and storage complexities of the new methods **IHP14** and **IHP15**, the state-of-the-art method **HP3** and the recently published methods **IHP5**, **IHP9** and **IHP17**.

It can be seen that **IHP14** and **IHP15** have the highest computational efficiencies, while E_c(**IHP9**) is very close to E_c(**IHP14**). The computational efficiency E_c is only a theoretical performance measure which does not take into account the effects of roundoff errors occurring in floating-point arithmetics. These roundoff errors can significantly degrade the performance of high-order methods and, additionally, make them diverge (Section 5).

4.1. Random matrices test

To obtain a real performance comparison, we need to test these methods on suitable test matrices. We use the same set of random test matrices as in [11]. All methods are implemented in the `Mathematica` package and tested in the cases of the ordinary inverse, the Moore–Penrose inverse ($G = A^*$), and the Drazin inverse ($G = A^l$, $l = \mathrm{ind}A$) computation.

The first test was done on 20 randomly generated double precision matrices for each dimension $n = 600, 800, 1000, 1200$ and each type of the inverse. All singular test matrices had the rank $\rho = 4n/5$, and in the case of the Drazin inverse, the matrices were taken such that $l = \mathrm{ind}A = 1$. More details on the algorithm for generating test matrices can be found in [11]. We used the following stopping criterion

$$\mathrm{res}(X_k) = \|R_k\|_F = \|I - AX_k\|_F < \epsilon = 10^{-10}$$

in the case of the ordinary inverse and

$$\mathrm{res}(X_k) = \max\{\|AX_kA - A\|_F, \|X_kAX_k - X_k\|_F\} < \epsilon = 10^{-10}$$

in the case of the Moore–Penrose and Drazin inverses. The methods were compared with respect to the total number of matrix multiplications required to compute the final result. This measure is independent of the implementation details. The results are given in Table 2. The bold values correspond to the best average total number of matrix multiplications.

It is evident from the table that the new methods **IHP14** and **IHP15** have the smallest total number of matrix multiplications in almost all cases. The results might vary with different choices of ϵ, but the final conclusion still remains.

The methods were also tested on several matrices from the Matrix Market library [13], where the Moore–Penrose inverse was computed. These matrices were obtained from various real-world problems. The results are shown in Table 3.

As it can be seen from the table, the same conclusions remain valid in this case as well. It can be further noted that **IHP9** performs almost as well as **IHP15** and **IHP14**, sometimes even better (matrix `well1033.mtx`). This can also be noticed in the previous testing results (Table 2). The reason for such a behavior is the enhanced effect of numerical errors resulting from the use of double precision arithmetics, which causes a loss in the performance of the high order methods **IHP14**, **IHP15** and **IHP17**. It also suggests that some other methods with even larger computational efficiencies (and also a higher convergence order) will not introduce any considerable practical improvement and may only have a theoretical significance.

Table 2
Average total number of matrix multiplications required to obtain the corresponding (generalized) inverse to desired accuracy.

Inverse	$m = n$	r	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**
Ordinary inverse	600	600	63.	57.8	53.75	53.1	**52.2**	59.15
	800	800	68.1	63.	58.	57.6	**56.4**	64.4
	1000	1000	68.4	62.6	58.75	58.2	**57.**	63.
	1200	1200	71.1	64.8	59.75	59.1	**58.8**	66.15
Moore–Penrose inverse	600	480	38.1	36.	33.5	30.6	**30.**	35.
	800	640	39.	36.	35.	36.	**30.**	35.
	1000	800	39.	36.	35.	36.	**34.5**	35.
	1200	960	39.	36.	**35.**	36.	36.	**35.**
Drazin inverse	600	480	36.	32.	**30.**	**30.**	**30.**	35.
	800	640	36.	32.	**30.**	**30.**	**30.**	35.
	1000	800	36.	35.8	**30.**	**30.**	**30.**	35.
	1200	960	37.8	36.	33.	**30.**	**30.**	35.

Table 3
Total number of matrix multiplications required by each method to compute Moore–Penrose inverses of different matrices from Matrix Market [13].

Matrix	m	n	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**
bp___200.mtx	822	822	99.	92.	85.	84.	**78.**	91.
bp___400.mtx	822	822	96.	88.	80.	**78.**	**78.**	91.
cdde1.mtx	961	961	66.	60.	55.	**54.**	**54.**	63.
cdde2.mtx	961	961	45.	44.	40.	42.	**36.**	42.
cdde3.mtx	961	961	69.	64.	**60.**	**60.**	**60.**	63.
fidap001.mtx	216	216	75.	68.	65.	**60.**	**60.**	70.
illc1033.mtx	1033	320	75.	68.	65.	**60.**	**60.**	70.
olm500.mtx	500	500	90.	84.	75.	78.	**72.**	84.
orsirr_1.mtx	1030	1030	78.	72.	**65.**	66.	66.	70.
orsirr_2.mtx	886	886	75.	68.	65.	66.	**60.**	70.
well1033.mtx	1033	320	48.	44.	**40.**	42.	42.	49.

4.2. Fixed number of iterations test

We also did a test using a fixed number of iterations to compute the Moore–Penrose inverse, as it was done in [11]. The methods were run for several iterations such that all of them perform approximately the same total number of matrix multiplications. Specifically, the methods **HP3**, **IHP5**, **IHP9** and **IHP17** were run for 7, 5, 4 and 3 iterations, such that **HP3** and **IHP17** perform a total of 21 multiplications while **IHP5** and **IHP9** perform a total of 20 multiplications. The methods **IHP14** and **IHP15** were run for 3 iterations after they were initiated with one iteration of the Schultz method (the matrix $\bar{X}_0 = X_0(2I - AX_0)$ is taken as the initial matrix). In such a way, these methods also perform a total of 20 matrix multiplications. The residual norms of the obtained results were then compared for each method, as given in Table 4. In order to make a fair comparison, we used the same test matrices $A_1, A_2, \dots, A_8$ as in [11] (Subsection 5.2).

It is evident again that **IHP15** produces the smallest residual norms, whereas **IHP9** performs better than **IHP14** and **IHP17**. Having in mind that **IHP9** and **IHP14** have almost the same computational efficiencies, we can conclude that **IHP14** is more affected by numerical errors than **IHP9**. Although **IHP17** has a lower computational efficiency than **IHP14** and **IHP9**, it is also affected by errors such that it produces residual norms that are several orders of magnitude worse than **IHP9**, **IHP14** and **IHP15**.

The same test is repeated in multi-precision arithmetics (100 significant digits) and for the Drazin inverse computation. Again, we used the same set of matrices $\tilde{A}_1, \tilde{A}_2, \dots, \tilde{A}_8$ as in [11] (Subsection 5.2). Note that this test requires computation of the coefficients of the methods **IHP14** (3.7), **IHP15** (3.6) and **IHP17** (3.4) in multi-precision arithmetics. The coefficients of **IHP14** and **IHP15**, computed up to 200 significant digits, are given in Appendices A and B.

Similarly to [11], in order to increase the obtained final residual norms, we first applied the Schultz method $X_{k+1} = X_k(2I - AX_k)$ for a total of one iteration for $\tilde{A}_1, \tilde{A}_2$ ($n = 20, 30$), two iterations for $\tilde{A}_3, \tilde{A}_4, \tilde{A}_5$ ($n = 40, 50, 60$) and four iterations for $\tilde{A}_6, \tilde{A}_7, \tilde{A}_8$ ($n = 70, 100, 130$). The results are shown in Table 5.

One can see that the obtained performances are mostly in accordance with the E_c values from Table 1. The only exception is the fact that **IHP9** is again slightly better than **IHP14**. It leads us to the conclusion that in the case of similar computational efficiencies E_c, the more efficient method is the one with a lower convergence order.

Table 4
Residual norms for the Moore–Penrose inverse computation and a fixed number of iterations using double precision arithmetics.

	Method	$\|AXA-A\|$	$\|XAX-X\|$		Method	$\|AXA-A\|$	$\|XAX-X\|$
	HP3	1.79×10^{-5}	2.04×10^{-5}		**HP3**	1.36×10^{-3}	2.28×10^{-4}
A_1	**IHP5**	1.69×10^{-7}	1.93×10^{-7}	A_5	**IHP5**	5.21×10^{-5}	8.8×10^{-6}
$n=20$	**IHP9**	1.31×10^{-14}	2.38×10^{-14}	$n=60$	**IHP9**	3.69×10^{-10}	6.26×10^{-11}
$\rho=15$	**IHP14**	1.15×10^{-13}	1.33×10^{-13}	$\rho=35$	**IHP14**	2.4×10^{-9}	4.06×10^{-10}
	IHP15	2.6×10^{-14}	3.38×10^{-14}		**IHP15**	8.58×10^{-12}	1.46×10^{-12}
	IHP17	2.35×10^{-11}	2.68×10^{-11}		**IHP17**	1.08×10^{-7}	1.83×10^{-8}
	HP3	2.32×10^{-2}	2.35×10^{-2}		**HP3**	6.23×10^{-3}	1.02×10^{-3}
A_2	**IHP5**	4.64×10^{-3}	4.8×10^{-3}	A_6	**IHP5**	4.77×10^{-4}	7.81×10^{-5}
$n=30$	**IHP9**	1.29×10^{-5}	1.34×10^{-5}	$n=70$	**IHP9**	3.93×10^{-8}	6.43×10^{-9}
$\rho=20$	**IHP14**	2.86×10^{-5}	2.97×10^{-5}	$\rho=40$	**IHP14**	1.64×10^{-7}	2.68×10^{-8}
	IHP15	1.74×10^{-6}	1.81×10^{-6}		**IHP15**	1.83×10^{-9}	$3.\times10^{-10}$
	IHP17	2.17×10^{-4}	2.25×10^{-4}		**IHP17**	3.57×10^{-6}	5.85×10^{-7}
	HP3	5.3×10^{-4}	1.92×10^{-4}		**HP3**	1.74×10^{-1}	2.39×10^{-2}
A_3	**IHP5**	1.68×10^{-5}	6.09×10^{-6}	A_7	**IHP5**	4.7×10^{-2}	6.91×10^{-3}
$n=40$	**IHP9**	5.41×10^{-11}	1.96×10^{-11}	$n=100$	**IHP9**	4.81×10^{-4}	7.4×10^{-5}
$\rho=25$	**IHP14**	4.07×10^{-10}	1.48×10^{-10}	$\rho=60$	**IHP14**	8.46×10^{-4}	1.3×10^{-4}
	IHP15	1.01×10^{-12}	3.66×10^{-13}		**IHP15**	1.01×10^{-4}	1.56×10^{-5}
	IHP17	2.33×10^{-8}	8.45×10^{-9}		**IHP17**	4.25×10^{-3}	6.48×10^{-4}
	HP3	3.96×10^{-3}	1.12×10^{-3}		**HP3**	1.14×10^{-1}	7.11×10^{-3}
A_4	**IHP5**	2.82×10^{-4}	7.97×10^{-5}	A_8	**IHP5**	1.99×10^{-2}	1.28×10^{-3}
$n=50$	**IHP9**	1.76×10^{-8}	4.97×10^{-9}	$n=130$	**IHP9**	3.76×10^{-5}	2.46×10^{-6}
$\rho=30$	**IHP14**	7.7×10^{-8}	2.18×10^{-8}	$\rho=70$	**IHP14**	8.89×10^{-5}	5.8×10^{-6}
	IHP15	7.56×10^{-10}	2.14×10^{-10}		**IHP15**	4.57×10^{-6}	2.99×10^{-7}
	IHP17	1.83×10^{-6}	5.17×10^{-7}		**IHP17**	7.53×10^{-4}	4.89×10^{-5}

Table 5
Residual norms for the Drazin inverse computation and a fixed number of iterations using double precision arithmetics.

	Method	$\|AXA-A\|$	$\|XAX-X\|$		Method	$\|AXA-A\|$	$\|XAX-X\|$
	HP3	5.28×10^{-10}	4.42×10^{-8}		**HP3**	5.93×10^{-8}	4.69×10^{-6}
$\tilde{A}_1$	**IHP5**	1.43×10^{-13}	1.20×10^{-11}	$\tilde{A}_5$	**IHP5**	1.20×10^{-10}	9.51×10^{-9}
$n=20$	**IHP9**	1.23×10^{-26}	1.03×10^{-24}	$n=60$	**IHP9**	1.65×10^{-20}	1.30×10^{-18}
$\rho=15$	**IHP14**	3.24×10^{-24}	2.71×10^{-22}	$\rho=35$	**IHP14**	8.87×10^{-19}	7.01×10^{-17}
	IHP15	2.66×10^{-30}	2.23×10^{-28}		**IHP15**	2.19×10^{-23}	1.73×10^{-21}
	IHP17	2.28×10^{-20}	1.91×10^{-18}		**IHP17**	8.86×10^{-16}	7.01×10^{-14}
	HP3	1.11×10^{-39}	1.06×10^{-38}		**HP3**	2.39×10^{-19}	1.84×10^{-17}
$\tilde{A}_2$	**IHP5**	3.54×10^{-56}	3.39×10^{-55}	$\tilde{A}_6$	**IHP5**	6.27×10^{-27}	4.81×10^{-25}
$n=30$	**IHP9**	$<10^{-100}$	$<10^{-100}$	$n=70$	**IHP9**	1.05×10^{-54}	8.10×10^{-53}
$\rho=20$	**IHP14**	$<10^{-100}$	$<10^{-100}$	$\rho=40$	**IHP14**	1.09×10^{-48}	8.37×10^{-47}
	IHP15	$<10^{-100}$	$<10^{-100}$		**IHP15**	9.04×10^{-62}	6.94×10^{-60}
	IHP17	$<10^{-100}$	$<10^{-100}$		**IHP17**	2.21×10^{-41}	1.70×10^{-39}
	HP3	3.17×10^{-14}	1.88×10^{-12}		**HP3**	2.54×10^{-16}	2.22×10^{-14}
$\tilde{A}_3$	**IHP5**	1.23×10^{-19}	7.32×10^{-18}	$\tilde{A}_7$	**IHP5**	1.36×10^{-22}	1.18×10^{-20}
$n=40$	**IHP9**	1.89×10^{-39}	1.12×10^{-37}	$n=100$	**IHP9**	1.43×10^{-45}	1.25×10^{-43}
$\rho=25$	**IHP14**	1.91×10^{-35}	1.13×10^{-33}	$\rho=60$	**IHP14**	8.83×10^{-41}	7.70×10^{-39}
	IHP15	1.03×10^{-44}	6.11×10^{-43}		**IHP15**	1.47×10^{-51}	1.28×10^{-49}
	IHP17	6.02×10^{-30}	3.58×10^{-28}		**IHP17**	1.50×10^{-34}	1.31×10^{-32}
	HP3	8.87×10^{-11}	5.47×10^{-9}		**HP3**	1.96×10^{-14}	1.69×10^{-12}
$\tilde{A}_4$	**IHP5**	8.60×10^{-15}	5.32×10^{-13}	$\tilde{A}_8$	**IHP5**	6.75×10^{-20}	5.82×10^{-18}
$n=50$	**IHP9**	1.85×10^{-29}	1.15×10^{-27}	$n=130$	**IHP9**	6.53×10^{-40}	5.63×10^{-38}
$\rho=30$	**IHP14**	1.11×10^{-26}	6.90×10^{-25}	$\rho=70$	**IHP14**	7.38×10^{-36}	6.36×10^{-34}
	IHP15	1.66×10^{-33}	1.03×10^{-31}		**IHP15**	3.20×10^{-45}	2.76×10^{-43}
	IHP17	1.99×10^{-22}	1.23×10^{-20}		**IHP17**	2.60×10^{-30}	2.24×10^{-28}

Table 6
Total number of matrix multiplications required to obtain the inverse matrix $A_{F,n}^{-1}$ to the accuracy 10^{-10}.

n	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**
100	54	48	45	48	42	49
300	66	60	55	54	54	63
500	72	64	60	60	60	70
700	75	68	65	66	60	70
900	78	72	65	66	66	70
1100	81	72	70	66	66	77
1300	81	76	70	72	66	77

Table 7
Total running time (in seconds) of the **BiCGStab** method (`Mathematica` implementation) for solving all linear systems $A_{F,n}x_{l,n} = g_{l,n}$ $(l = 1, 2, \ldots, 30)$, preconditioned by the matrix obtained using different iterative methods.

n	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**
700	1.97	1.53	0.80	0.53	0.47	1.00
800	1.36	1.06	0.48	0.52	0.38	0.78
900	1.08	0.53	0.34	0.41	0.28	0.56
1000	1.25	0.55	0.36	0.33	0.28	0.97
1100	1.17	0.83	0.41	0.41	0.38	0.50
1200	1.75	1.06	0.55	0.56	0.45	0.66
1300	1.47	0.70	0.50	0.47	0.42	0.86

4.3. Application to solving Fredholm integral equation

Finally, we use generalized Schultz iterative methods for solving the following Fredholm integral equation

$$\int_0^1 K(s,t)x(t) = g_l(s), \quad s \in [0,1] \tag{4.8}$$

where

$$K(s,t) = \begin{cases} s(1-t) & \text{for } s \le t, \\ t(1-s) & \text{for } s > t, \end{cases} \quad g_l(s) = \frac{s}{3}(s^3 - ls + 1), \quad l \ge 0.$$

The same problem for $l = 2$ is considered in [14]. Using the generalized mid-point quadrature formula in the points $t_i = (i-1/2)/n$ $(i = 1, 2, \ldots, n)$ for the integral in (4.8), and evaluating the equation at the points $s_j = (j-1/2)/n$, $j = 1, 2, \ldots, n$, we obtain the following linear system

$$A_{F,n}x_n = g_{l,n} \tag{4.9}$$

where $\Delta t = 1/n$, $A_{F,n} = [K(t_i, s_j)\Delta t]_{1\le i,j\le n} \in \mathbb{R}^{n\times n}$, $x_n = [x(i\Delta t)]_{1\le i\le n}^T$ and $g_{l,n} = [g_l(j\Delta t)]_{1\le j\le n}^T$.

The matrix $A_{F,n}$ is regular and its ordinary inverse is computed using all previously considered iterative methods. Table 6 shows the total number of matrix multiplications required by each method to compute the inverse matrix $A_{F,n}^{-1}$ to the accuracy $\epsilon = 10^{-10}$. The shown results imply that the **IHP15** method requires the smallest total number of matrix multiplications to achieve the required accuracy ϵ. The second best result is obtained either by **IHP14** or **IHP9**.

Moreover, we did another test where all iterative methods were run for a certain number of iterations, and the resulting matrices were used as preconditioners for the **BiCGStab** method for solving the linear system (4.9). We first ran the **HP3** method until it reached the accuracy $\epsilon = 25$. Then all other methods were run such that they perform the maximum possible number of matrix multiplications, which is not larger than one for the **HP3** method. All resulting matrices were used as preconditioners for the **BiCGStab** method (default Krylov space method for the `LinearSolve` function in `Mathematica`). The total running times of the preconditioned **BiCGStab** method are shown in Table 7. One can see that **IHP15** produces the minimal running time, while either **IHP14** or **IHP9** has the second minimal running time.

5. Influence of floating-point errors on generalized Schultz iterative methods

The results of the testing in the previous section (especially in Section 4.2) show that all considered methods are influenced by numerical errors produced by floating-point arithmetics. In this section, we consider one possible effect of

Table 8
Values $|p(0)|$ for different generalized Schultz iterative methods.

Method	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**
$\|p(0)\|$	3	5	9	14.7737	15.7587	17

roundoff errors on an arbitrary generalized Schultz iterative method $X_{k+1} = X_k p(AX_k)$, when computing an outer G-inverse. This effect is already considered in [15] and [6], for two special methods.

Assume that a roundoff error occurs in the sth iteration, such that $\tilde{X}_s = X_s + \Delta_s$ is computed instead of X_s. The matrix Δ_s is the error matrix. Further iterations are computed as usual, $\tilde{X}_{k+1} = \tilde{X}_k p(A\tilde{X}_k)$, for $k \geq s$. The following theorem shows that under certain conditions on the matrix Δ_s (i.e. the matrix $\tilde{X}_s$) and the polynomial $p(x)$, the sequence $\tilde{X}_k$ diverges.

Theorem 5.1. *Assume that $\mathcal{N}(\tilde{X}_s A\tilde{X}_s) \supset \mathcal{N}(\tilde{X}_s)$ and $|p(0)| > 1$. Then the resulting method*

$$\tilde{X}_{k+1} = \tilde{X}_k p(A\tilde{X}_k), \qquad k \geq s \tag{5.10}$$

diverges and $\|\tilde{X}_k\| \geq c \cdot |p(0)|^{k-s}$, where $c > 0$ is a constant which depends only on $\tilde{X}_s$. The matrix norm $\|\cdot\|$ is induced by the corresponding vector norm.

Proof. Consider an arbitrary $y \in \mathcal{N}(\tilde{X}_s A\tilde{X}_s) \setminus \mathcal{N}(\tilde{X}_s)$ and let $p(x) = a_0 + a_1 x + \cdots + a_d x^d$. We prove by mathematical induction that $y \in \mathcal{N}(\tilde{X}_k A\tilde{X}_k) \setminus \mathcal{N}(\tilde{X}_k)$ and $\tilde{X}_k y = a_0^{k-s}\tilde{X}_s y$ for all $k \geq s$. The statement trivially holds for $k = s$. Assume also that it is valid for some $k \geq s$. Then

$$\begin{aligned}\tilde{X}_{k+1}y &= \tilde{X}_k p(A\tilde{X}_k)y = \tilde{X}_k(a_0 I + a_1 A\tilde{X}_k + \cdots + a_d (A\tilde{X}_k)^d)y\\ &= a_0\tilde{X}_k y + a_1\tilde{X}_k A\tilde{X}_k y + a_2\tilde{X}_k A\tilde{X}_k A\tilde{X}_k y + \cdots + (\tilde{X}_k A)^{d-1}\tilde{X}_k A\tilde{X}_k y\\ &= a_0\tilde{X}_k y.\end{aligned}$$

The last equation is satisfied since $\tilde{X}_k A\tilde{X}_k y = 0$ by assumption and $\tilde{X}_k(A\tilde{X}_k)^l = (\tilde{X}_k A)^{l-1}\tilde{X}_k A\tilde{X}_k$. Now, using the induction hypothesis, we get directly

$$\tilde{X}_{k+1}y = a_0\tilde{X}_k y = a_0 \cdot a_0^{k-s}\tilde{X}_s y = a_0^{k+1-s}\tilde{X}_s y.$$

This also proves $\tilde{X}_{k+1}y \neq 0$ i.e. $y \notin \mathcal{N}(\tilde{X}_{k+1})$ since $a_0 \neq 0$. Furthermore,

$$\tilde{X}_k A\tilde{X}_{k+1}y = \tilde{X}_k A(a_0\tilde{X}_k y) = a_0\tilde{X}_k A\tilde{X}_k y = 0$$

and

$$\begin{aligned}\tilde{X}_{k+1}A\tilde{X}_{k+1}y &= \tilde{X}_k p(A\tilde{X}_k)A\tilde{X}_{k+1}y\\ &= \sum_{l=0}^{d} a_l\tilde{X}_k(A\tilde{X}_k)^l A\tilde{X}_{k+1}y\\ &= a_0\tilde{X}_k A\tilde{X}_{k+1}y + \sum_{l=1}^{d} a_l(\tilde{X}_k A)^{l-1}\tilde{X}_k A\tilde{X}_{k+1}y = 0.\end{aligned}$$

Hence, $y \in \mathcal{N}(\tilde{X}_{k+1}A\tilde{X}_{k+1}) \setminus \mathcal{N}(\tilde{X}_{k+1})$, which completes the proof by mathematical induction.

Now, since $|a_0| = |p(0)| > 1$ by assumption, we get

$$\|\tilde{X}_k\| \geq \frac{\|\tilde{X}_k y\|}{\|y\|} = |a_0|^{k-s}\frac{\|\tilde{X}_s y\|}{\|y\|}.$$

Taking $c = \|\tilde{X}_s y\|/\|y\|$ and $k \to +\infty$ (using again $|a_0| > 1$), the proof follows. □

Having in mind that the error matrix Δ_s is generated by roundoff errors, its structure is mostly random, so one can believe that $\tilde{X}_s = X_s + \Delta_s$ is (almost surely) a full row or column rank matrix. Under the assumption that A is not a full row or column rank matrix, we have $\operatorname{rank}\tilde{X}_s > \operatorname{rank}A$, which then implies $\mathcal{N}(\tilde{X}_s A\tilde{X}_s) \supset \mathcal{N}(\tilde{X}_s)$.

It can be seen by direct check that the condition $|p(0)| = |\bar{p}(1)| > 1$ is satisfied for all methods considered in the previous section. These values are given in Table 8.

Table 9
Testing results on several ill-conditioned test matrices from Matrix Market.

Matrix	n	**HP3**	**IHP5**	**IHP9**	**IHP14**	**IHP15**	**IHP17**	**SVD**
olm1000.mtx	1000	3.86×10^{-10}	3.89×10^{-10}	3.89×10^{-10}	3.81×10^{-10}	4.13×10^{-10}	3.68×10^{-10}	1.66×10^{-9}
olm2000.mtx	2000	1.25×10^{-9}	1.24×10^{-9}	1.25×10^{-9}	1.25×10^{-9}	1.24×10^{-9}	1.23×10^{-9}	1.04×10^{-8}
olm500.mtx	500	7.89×10^{-11}	8.18×10^{-11}	7.72×10^{-11}	8.36×10^{-11}	6.93×10^{-11}	7.91×10^{-11}	2.65×10^{-10}
plat1919.mtx	1919	6.39	1.35	0.38	0.33	0.33	0.42	1.00
plat362.mtx	362	1.67	3.85×10^{-5}	3.93×10^{-5}	3.86×10^{-5}	3.32×10^{-5}	3.93×10^{-5}	8.73×10^{-5}
sherman2.mtx	1080	8.79	1.03×10^{-8}	9.58×10^{-9}	1.03×10^{-8}	1.03×10^{-8}	1.07×10^{-8}	1.57×10^{-4}
steam2.mtx	600	5.21×10^{-14}	4.22×10^{-14}	4.95×10^{-14}	2.65×10^{-10}	9.06×10^{-14}	5.24×10^{-14}	1.67×10^{-8}

According to Theorem 5.1, all these methods will converge for some iterations, and after that they will start diverging. When both A and G are full-rank matrices, the condition $\mathcal{N}(\tilde{X}_s A \tilde{X}_s) \supset \mathcal{N}(\tilde{X}_s)$ cannot be satisfied. However, accumulating roundoff errors causes the final result to be inaccurate. In that sense, we test all methods on some ill-conditioned matrices from Matrix Market [13]. Each method is run for several iterations. We show the best obtained residual norm $\|I - AX_k\|_F$. The results are shown in Table 9. Additionally, the last column shows the residual norms obtained using the Singular Value Decomposition (**SVD**) based method (`Mathematica` build-in function `Pseudoinverse`).

It can be seen from Table 9 that for every test matrix all obtained residual norms are approximately of the same order of magnitude. They are also comparable to (or sometimes less than) the residual norm obtained by the **SVD** based method. This means that all tested methods are shown to be equally (un)stable on all test matrices. In other words, the high convergence order of the methods **IHP14**, **IHP15** and **IHP17** did not additionally reduce the performances of these methods on the ill-conditioned matrices.

6. Conclusion and further research

Having in mind all previous discussion, we can conclude that the newly constructed method **IHP15** is currently the most efficient generalized Schultz iterative method for computing outer G-inverses. Its (theoretically) highest computational efficiency E_c is confirmed by numerical testing, where it achieves the best results on both double precision and multiple precision arithmetics.

The general scheme (2.1)–(2.2) can be used in the same way to construct efficient methods with an arbitrary number of matrix multiplications per iteration θ. Such methods would have the convergence order of up to $2^{\theta-2}$ and, perhaps, even higher computational efficiency, but, unfortunately, they will most probably be only theoretically important. Consider, for example $\theta = 7$. In order to provide a method with an E_c higher than $E_c(\textbf{IHP15}) = 15^{1/6} \approx 1.5704$, it must have the convergence order $r \geq 24$, which will most probably result in its practical inapplicability. However, from the theoretical point of view, the following problem is definitely interesting for future research.

Problem 6.1. Find a way to construct a method of the form (2.1)–(2.2) having the highest possible convergence order r, for a given number of matrix multiplications per iteration θ.

On the other hand, it would be interesting to explore the stability of the arbitrary generalized Schultz iterative method $X_{k+1} = X_k p(AX_k)$ when the conditions of the instability theorem (Theorem 5.1) are not met, especially when $|p(1)| < 1$ or $p(1) = 0$. We also leave this problem for future research.

Acknowledgments

The authors would like to thank professors Predrag S. Stanimirović and Aleksandar S. Cvetković for useful discussion on this topic. We also gratefully acknowledge the help by Professor David D. Pokrajac from Delaware State University, since a part of the time-consuming numerical computation was done on the computers from his research laboratory.

Moreover, the authors wish to thank the anonymous reviewers for the careful reading and valuable comments which improved the quality of the paper. This study is supported by the Serbian Ministry of Education and Science, Serbia, Grant No 174013.

Appendix A. Coefficients of IHP15 method in multiprecision arithmetics

$$\begin{aligned}
c &= 0.144930075923807570678172370025672076908118035215628454264338657802222600 2125\\
&\ 652038430033104624049371457227147211472532821502295511189444824619624561388870\\
&\ 711831413501556819400525013572912661632624 7684\\
a_{3,0} &= 0.6450829220614610138647060270484373728459055693070258186128550605323468048601\\
&\ 2364479137102487405633623195982428323070154841024881651895590308148431008801 98\\
&\ 368408287606750726346456272275433388737952 9180\\
a_{3,1} &= 1.0586615942624956438158333079957432428406523558483868564053442415828764501439\\
&\ 2275127270534820777552250012626070065777606797707618918296794420268246663906 49\\
&\ 14519765837703885145754158410447943925224 7255\\
a_{4,0} &= 0.0506549871625042783426434814906721546379954799427165214107475789465359055604\\
&\ 0467223614722868864319949285734451998815802612436729416984796358942687415703 14\\
&\ 09439088390973196160392069025990805855161986159\\
a_{4,1} &= 0.3459018871146173374677172838297682708583904393355524447591598880020263293388\\
&\ 5063923474197418899353704839374618795895526358366270704377335182688993347919 37\\
&\ 9156469983412203366474528069321342996321100121\\
a_{4,2} &= -1.2025194139289593766197381962576145544266311878173653606913667950962887494 25\\
&\ 2946994296972443825658814710728852881428216327699581052276686163729806361939 88\\
&\ 2776468166739977404809159402624493281366867092\\
a_{5,0} &= 1.2745242086494158687138847817388378991860918592694371317007021502809014435946\\
&\ 3484982064343273088179395133310315245138421368778833294838760036715464509292 50\\
&\ 391175425787706440188271544743292296611176712\\
a_{5,1} &= 1.7999108187703980588929782092312522536326987373780069557112491556921855571245\\
&\ 4623681557512536988204731843531585490734404747101397284717851545750355474766 10\\
&\ 292326321629498145641629077545970210706185486\\
a_{5,2} &= 5.0950884501880239616807378436158903632780281966201660358119891536035312895857\\
&\ 2160903004879859749592168326189168176578132222253232810596873252105093064311 23\\
&\ 606431200251792619220977080792068238847465550\\
a_{5,3} &= -1.1491089042271791658579731158570725427150641985678297323150460531118840367 89\\
&\ 5982214904087823832595982287190465471973306643325351867436218334125527468715 28\\
&\ 9285751826927098569439032213470916928358918647\\
b_{3,0} &= 0.4353207862793513988155581385391670206891946872594844513000067261884229363928\\
&\ 8211944763765932913942430424300235741973012975144312651275084090227306142205 79\\
&\ 9301635272881386796751291059569768472214718139\\
b_{3,1} &= 0.2263267680368166248654209679922426688801056124682974757238767807618729826103\\
&\ 8921752607774914405722682338376201431981189003870870134588420469067126104718 51\\
&\ 3222270294828070228919227080285148032314053298\\
b_{4,0} &= 0.4256316748590594999645640862172512360560150634641786069420760472067162815248\\
&\ 1262576123033252601092677112799058922296436015098793794853370310086499118655 40\\
&\ 6276485554512207540524206382343582838401132521\\
b_{4,1} &= -0.7568252266561805019372688892799004031049828939701498400401036069813500430 42\\
&\ 1586902247741493449996593674865357704306777137630970262071764829807715889571 67\\
&\ 20450230720255882259991390854547949071145852913\\
b_{4,2} &= -1.6223020311897785555923636326670254204755665472720681743375597664037125420 16\\
&\ 2927277762765491301925159839662109469972720267113174913745321201240454536391 40\\
&\ 9526043401384218782201651097152030838244431950\\
b_{5,0} &= 2.7235604872075580897235250748697719805263892710981995932891537210763709183385\\
&\ 3583766238400466682725858190629484003959765638373402988162205808018693205715 04\\
&\ 568551491658443186063974056464270541310219855\\
b_{5,1} &= 5.0298291581081260726490273908507148146609159818089750114396955753343062912052\\
&\ 2081367019462028509008342963890165746655692743577828072462343073892954183821 12\\
&\ 012969362548834113039923163920305136569951088\\
b_{5,2} &= 2.6371014997658525256201147266645001271977959325104544537679256048756690238174\\
&\ 8892769761586846767703149767081082215006557561181919162199895629807608392604 79\\
&\ 326760403217328320667157488528732009768563609\\
b_{5,3} &= 7.5276481060538817566651074074288556553697434410386814332535141155072788129227\\
&\ 3916916377199020257561771118345533388360920620423908978724333861262026412476 84\\
&\ 585706538428822358528366627686706967885949785.
\end{aligned}$$

Appendix B. Coefficients of IHP14 method in multiprecision arithmetics

$$
\begin{aligned}
a_{3,0} &= 0.994592232369608356925137322350336799469986942202824911070905622507859682008\\
&\quad 3975875746471889626310393658344590693395733481938242833881664367883723368239841\\
&\quad 711668595208179699655474955577312186833539081 2\\
a_{3,1} &= 0.589305851677215924021899043593129864520897392146507415971963604689315855236 9\\
&\quad 1789359014133204233523423675476260410214087285063821250090744446349708509707 44\\
&\quad 3580886482027078604250710922821742781534412580\\
a_{4,0} &= 0.7160883251593376608456738396627034971140578435978571171144393775566651414658\\
&\quad 2550361333949790072271100869745950074962424854218547002588997989481311859736 72\\
&\quad 90600642343594315083492924090378385101111173 07\\
a_{4,1} &= -1.219543968940840145859093978450108310849512788717048705863498277042462172750\\
&\quad 2008756784391675515182433060536872911405080818410302667510780828992782735687 46\\
&\quad 785419154168459739507974351075343945282351525 5\\
a_{4,2} &= -0.038317189491436174741552015086083592921228510252837188781611272692906844190\\
&\quad 6747413276380386809889620887616327474465128743573360494030423693000814448235 12\\
&\quad 5650125899661292505498504442352714329874862146 69\\
a_{5,0} &= -0.612715355555755968012847741588229749901778337013678890754718730517743342585\\
&\quad 7322491798167829573250075549325399389565008450615138247387000573970904449529 42\\
&\quad 0452041558185758728163293851819583754968299653 8\\
a_{5,1} &= 1.174304135325604801739200467827569970694925784599512396081009098529863546267 7\\
&\quad 6263774520252243303454559600668840609832814255308269276388057807872369148322 93\\
&\quad 94713343679356503565474437468828984657499494 0\\
a_{5,2} &= -0.983452829557210825938841070405975087036903291238472408433289405932793027806\\
&\quad 3044040889474505047086601309790121082029009604381154618952927346307992059697 94\\
&\quad 4287633477052021711586749286009733669979541455 8\\
a_{5,3} &= -0.124571668920262155487295945351336765174115074691626160617143046219664263083\\
&\quad 1292293508575219387058846067422186882158905780226043136387226523343596218986 92\\
&\quad 7128774183200991839561551388163233829225706077 4\\
b_{3,0} &= 0.136944926273856539661432973487597569592348834570416752156830122554305075580 6\\
&\quad 3674727451166316167983643735099370375723268464738985477079479380830397514027 94\\
&\quad 697188213098953744093368845887329250527947962 5\\
b_{3,1} &= -0.249592472683751625847738497575364953705686818204077520412928749981130562074\\
&\quad 1378674599415433248377572078252445761905456318114881468331872247229400074577 90\\
&\quad 2109059105723978696612255576829175525283780886 4\\
b_{4,0} &= 0.316489946814256747835459736476676184566372645945105767262945663516015230287 2\\
&\quad 4140546779228713970347109646281622730509794292137963183874285379575655019695 03\\
&\quad 257734208073351321489985044600215925823064651 0\\
b_{4,1} &= -0.202936958667333649957223347350150390119464540427586567010652518993019244227\\
&\quad 0684780076726710923871200826945156048103001597567855458934056443840040932207 38\\
&\quad 2016213171232793826508046290668780474495808171 1\\
b_{4,2} &= 0.738676166672721676707152046033499129005164313400620563647172529177801235864 0\\
&\quad 0142512193945986828550583261668166092536156370531571681451068079364326335873 30\\
&\quad 101944784275243816701320194648030074370775835 2\\
b_{5,0} &= 0.992571434027460899687041791139845095186467256460317196492039069227289242087 1\\
&\quad 1204872288791090212878430461898292246463962055682620655664072759196305329069 22\\
&\quad 691801096230924843734712138442941472046848696 5\\
b_{5,1} &= 0.720714144371934137025459264317528266365132078645112513776446289221424787543 3\\
&\quad 1993314758056187753032233436072363153737631286303647091210930514819210153334 26\\
&\quad 150672203114796362200829639020221929779902579 3\\
b_{5,2} &= 1.109912972445306289346528911848878005998115161665479863389784062772025131982 6\\
&\quad 8407202194239095157833732426184783804385253126040144221122239247811335784187 37\\
&\quad 87831108181290339831034240267670206404858778 5\\
b_{5,3} &= 0.675885458386025640212864704230015035642064702412243595092654425334434214043 3\\
&\quad 7570237523564702391796942664026140060996912034711964834599213292895093076183 11\\
&\quad 381121060375808760160693643407643207791983890 0.
\end{aligned}
$$

References

[1] V.N. Katsikis, D. Pappas, A. Petralias, An improved method for the computation of the Moore–Penrose inverse matrix, Appl. Math. Comput. 217 (2011) 9828–9834.
[2] L. Chen, E.V. Krishnamurthy, I. Macleod, Generalized matrix inversion and rank compuation by successive matrix powering, Parallel Comput. 20 (1994) 297–311.
[3] P.S. Stanimirović, D.S. Cvetković-Ilić, Successive matrix squaring algorithm for computing outer inverses, Appl. Math. Comput. 203 (2008) 19–29.
[4] Y. Wei, Successive matrix squaring algorithm for computing the Drazin inverse, Appl. Math. Comput. 108 (2000) 67–75.
[5] H. Chen, Y. Wang, A family of higher–order convergent iterative methods for computing the Moore–Penrose inverse, Appl. Math. Comput. 218 (2011) 4012–4016.
[6] M.D. Petković, P.S. Stanimirović, Two improvements of the iterative method for computing Moore–Penrose inverse based on Penrose equations, J. Comput. Appl. Math. 267 (2014) 61–71.
[7] A.R. Soheili, F. Soleymani, M.D. Petković, On the computation of weighted Moore–Penrose inverse using a high-order matrix method, Comput. Math. Appl. 66 (11) (2013) 2344–2351.
[8] 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.
[9] M.D. Petković, Generalized Schultz iterative methods for the computation of outer inverses, Comput. Math. Appl. 67 (10) (2014) 1837–1847.
[10] J.F. Traub, Iterative Methods for the Solution of Equations, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.
[11] M.D. Petković, M.S. Petković, Hyper–power methods for the computation of outer inverses, J. Comput. Appl. Math. 278 (2015) 110–118.
[12] F. Soleymani, P.S. Stanimirović, F.K. Haghani, On hyperpower family of iterations for computing outer inverses possessing high efficiencies, Linear Algebra Appl. 484 (2015) 477–495.
[13] Matrix Market, National Institute of Standards and Technology, Gaithersburg, MD. Available online from http://arxiv.org///math.nist.gov/MatrixMarket.
[14] F.S.V. Bazan, Simple and efficient determination of the Tikhonov regularization parameter chosen by the generalized discrepancy principle for discrete ill-posed problems, J. Sci. Comput. 63 (2015) 163–184.
[15] M.D. Petković, P.S. Stanimirović, Iterative method for computing Moore–Penrose inverse based on Penrose equations, J. Comput. Appl. Math. 235 (2011) 1604–1613.