Journal of Mathematical Modeling
Vol. -, No. -, 2026, pp. -. Research Article

JMM

A class of inversion-free iterative methods to find polar decomposition

Salman Sheikhi[†], **Hamid Esmaeili**[†*]

[†] *Department of Mathematics, Faculty of Science, Bu-Ali Sina University, Hamedan, Iran*

Email(s): s.sheikhi@sci.basu.ac.ir, esmaeili@basu.ac.ir

Abstract. This paper presents a novel approach for determining the polar decomposition of a complex (real) matrix, accompanied by a thorough examination of its convergence. The proposed class of methods' structure is inspired by Petković's idea for computing the Moore-Penrose inverse of a matrix using polynomials. This class of methods, rooted in matrix multiplications, circumvents the need for matrix inversion. Several categories of methods with varying convergence orders are introduced and scrutinized. The efficacy of these proposed methods is demonstrated through numerical experiments, comparing them with alternative approaches. The study focuses on random matrices of dimensions $n \times n$, where n takes values of 80, 100, 120, 150, 180, and 200 and several ill-conditioned matrices. The assessment includes key metrics such as the average number of iterations, matrix multiplications, and execution time for each method under consideration. The results affirm the efficiency of certain proposed methods in comparison to others.

Keywords: Polar decomposition, iterative method, convergence order, matrix multiplications.
AMS Subject Classification 2010: 15A23, 65F99.

1 Introduction

Polar decomposition has many applications in satellite tracking, factor analysis, computing the best orthogonal approximation for a matrix, finding the solution for orthogonal Procrustes problem in statistics, computing a nearest symmetric positive semi-definite matrix, and calculation of the second roots of a positive definite matrix [6, 8–10].

Consider an arbitrary matrix $A \in \mathbb{C}^{m\times n}$, $m \geq n$. A matrix U and a Hermitian positive semi-definite matrix H exist such that

$$A = UH, \quad U^*U = I_r, \quad H = (A^*A)^{1/2}, \tag{1}$$

*Corresponding author
Received: 06 February 2026/ Revised: 18 June 2026/ Accepted: 30 June 2026
DOI: 10.22124/jmm.2026.32920.2996

where $rank(A) = rank(U) = r$, is a polar decomposition of matrix A. When the rank of matrix A is equal to n, the matrix H becomes positive definite, and the existence of U is unique. For the case $n \geq m$, we can write

$$A = HU, \quad UU^* = I_r, \quad H = (AA^*)^{1/2},$$

and $A = HU$ is considered as the polar decomposition of the matrix A. It is a common practice to investigate polar decomposition when $m \geq n$, as exemplified in the paper under consideration. Polar decomposition, a crucial and practical matrix decomposition [11], extends the familiar polar representation of a complex number, $z = ae^{i\theta}$, to complex matrices. Here, $e^{i\theta}$ is linked to U, which possesses orthonormal columns, and a is associated with a positive semi-definite matrix H [1, 7]. Obtaining the polar decomposition through the singular value decomposition (SVD) [5] is a well-established strategy. Assuming there exists an SVD for matrix A,

$$A = P \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} Q^*,$$

in which $P \in \mathbb{C}^{m\times m}$ and $Q \in \mathbb{C}^{n\times n}$ are unitary matrices and also $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$ with non-zero singular values $0 < \sigma_r \leq \cdots \leq \sigma_1$. For matrices P and Q, let $P = [P_1, P_2]$ and $Q = [Q_1, Q_2]$ such that P_1 and Q_1 are the first r columns of P and Q, respectively. Then, $P_1^* P_1 = I_r$ and $Q_1^* Q_1 = I_r$ hold, so

$$A = (P_1 Q_1^*)(Q_1 \Sigma Q_1^*) = UH, \quad U = P_1 Q_1^*, \quad H = Q_1 \Sigma Q_1^*. \tag{2}$$

Several methods were introduced to compute polar decomposition [2–4, 12–15, 18–20]. Most of them compute U and then the symmetric matrix $H = \frac{1}{2}(U^*A + A^*U)$ is obtained. In [14], Kovarik presented the quadratically convergent iterative method as follows:

$$\begin{cases} U_0 = A, \quad R_k = I - U_k^* U_k \\ U_{k+1} = U_k(I + 0.5R_k), \quad k = 0, 1, \ldots. \end{cases} \tag{3}$$

Gander [4] applied Halley's method to compute polar decomposition with three-order convergence as follows:

$$\begin{cases} U_0 = A, \\ U_{k+1} = U_k(U_k^* U_k + 3I)(3U_k^* U_k + I)^{-1}, \quad k = 0, 1, \ldots, \end{cases} \tag{4}$$

in which $U_0 = A$ is a nonsingular matrix. Petcu and Popa [15], based on Kovarik's method, introduced the following iterative method:

$$\begin{cases} U_0 = A, \quad D_k = (I - U_k^* U_k)(I - 0.5U_k^* U_k), \\ U_{k+1} = U_k(I + D_k), \quad k = 0, 1, \ldots. \end{cases} \tag{5}$$

It has been demonstrated that equation (5) exhibits linear convergence, albeit with a convergence order of two. In a related context, Esmaeili [3] introduced an iterative method that avoids matrix inversion, outlined as follows:

$$\begin{cases} U_0 = A, \quad R_k = I - U_k^* U_k, \\ U_{k+1} = U_k(I + \frac{5}{4}R_k(\frac{2}{5}I + R_k)), \quad k = 0, 1, \ldots. \end{cases} \tag{6}$$

It achieves quadratic convergence and requires three matrix multiplications per iteration. In a study by the authors [20], they introduced an algorithm employing a sixth-order iteration

$$\begin{cases} U_0 = A, \quad Y_k = U_k^* U_k, \quad Z_k = Y_k Y_k, \quad W_k = Z_k Y_k, \quad T_k = W_k Y_k, \\ U_{k+1} = U_k(20I + 108Y_k + 108Z_k + 20W_k) \cdot (3I + 60Y_k + 130Z_k + 60W_k + 3T_k)^{-1}, \quad k = 0, 1, \ldots \end{cases} \tag{7}$$

until $\frac{\|U_{k+1}-U_k\|_\infty}{\|U_k\|_\infty}$ is less than $\zeta = 10^{-1}$. Then, it continues with the well-known Newton's method

$$U_{k+1} = \tfrac{1}{2}(U_k + U_k^{\dagger *}), \quad k = 0, 1, \ldots,$$

in which $U_k^\dagger$ is the Moore-Penrose inverse of U_k. Khaksar Haghani and Soleymani [12] introduced a fourth-order convergent method as follows:

$$\begin{cases} U_0 = A, \quad Y_k = U_k^* U_k, \quad Z_k = Y_k Y_k, \\ U_{k+1} = U_k(7I + Y_k)(I + 3Y_k)(I + 18Y_k + 13Z_k)^{-1}, \quad k = 0, 1, \ldots. \end{cases} \tag{8}$$

In [2], the authors presented the following sixth-order convergent method:

$$\begin{cases} U_0 = A, \quad Y_k = U_k^* U_k, \quad Z_k = Y_k Y_k, \quad W_k = Y_k Z_k, \quad T_k = Z_k Z_k, \\ U_{k+1} = U_k(36I + 314Y_k + 384Z_k + 66W_k) \\ \qquad\qquad \cdot(4I + 141Y_k + 435Z_k + 211W_k + 9T_k)^{-1}, \quad k = 0, 1, \ldots. \end{cases} \tag{9}$$

In [19], Soleymani, Khaksar Haghani and Shateyi introduced a seventh-order convergent method as follows:

$$\begin{cases} U_0 = A, \quad Y_k = U_k^* U_k, \quad Z_k = Y_k Y_k, \quad W_k = Y_k Z_k, \quad L_k = Y_k W_k, \\ U_{k+1} = U_k(765I + 7840Y_k + 12866Z_k + 4008W_k + 121L_k) \\ \qquad\qquad \cdot(81I + 3208Y_k + 12306Z_k + 8960W_k + 1045L_k)^{-1}, \quad k = 0, 1, \ldots. \end{cases} \tag{10}$$

Kiyoumarsi [13] proposed the following sixth-order convergent method:

$$\begin{cases} U_0 = A, \quad Y_k = U_k^* U_k, \quad Z_k = Y_k Y_k, \quad W_k = Y_k Z_k, \quad L_k = Y_k W_k, \\ U_{k+1} = U_k(28I + 146Y_k + 104Z_k + 10W_k) \\ \qquad\qquad \cdot(4I + 85Y_k + 155Z_k + 43W_k + L_k)^{-1}, \quad k = 0, 1, \ldots. \end{cases} \tag{11}$$

The methods presented in equations (3), (5), and (6) are based solely on matrix multiplications and exhibit a consistent structure. We will introduce a general form that encompasses these three methods and will compare this proposed class of methods to the methods discussed earlier, including some high-order methods.

In this paper, it is assumed that

$$\|A^* A\|_2 < 1, \tag{12}$$

otherwise the scaled matrix

$$A_{(new)} = \tau A, \qquad \tau = \frac{1}{\sqrt{\|A\|_1 \|A\|_\infty + 1}} \tag{13}$$

could be used. We note that

$$\sigma_i(\tau A) = \sqrt{\lambda_i(\tau^2 A^* A)} = \sqrt{\tau^2 \lambda_i(A^* A)} = \tau \sqrt{\lambda_i(A^* A)} = \tau \sigma_i(A).$$

So,

$$\tau A = P \begin{pmatrix} \tau\Sigma \\ 0 \end{pmatrix} Q^* = (P_1 Q_1^*)(Q_1(\tau\Sigma)Q_1^*) = U(\tau H),$$

then the factor U is invariant and the factor H scaled linearly.

In Section 2, we unveil a generalized iterative approach designed for calculating matrix polar decomposition and eliminating the necessity for matrix inversion. Section 3 then establishes several categories of inventive iterative methods with diverse convergence orders derived from the generalized approach. Transitioning to Section 4, we undertake a comparative analysis between the proposed methods and prevailing techniques. Subsequently, we offer a detailed exposition of our results, including the average execution time, iteration count, and matrix multiplication count. These metrics are derived from experiments conducted on random matrices of varying dimensions.

2 A class of iterative methods and its properties

In this section, we commence by introducing a comprehensive inversion-free iterative method designed for obtaining polar decomposition and subsequently provide a proof of its convergence. Our objective is to present a generalized method, akin to the generalized Schultz iterative methods $X_{k+1} = X_k\phi(AX_k)$, initially proposed by Petković [16] for computing the Moore-Penrose inverse of a matrix A using the polynomial $\phi(x)$. We delve into the convergence analysis of this generalized method, exploring the necessary determinant conditions for its convergence. We show that, despite Petković's method, the polynomial from this generalized approach is represented in the form of an even polynomial function.

It is worth noting that the method (3) was formulated based on the following scalar iterative method:

$$x_{k+1} = x_k(1+\frac{1}{2}(1-x_k^2)) = x_k(\frac{3}{2}-\frac{1}{2}x_k^2) = x_k p_1(x_k^2).$$

Also, the method (5) is established based on the scalar iterative method

$$x_{k+1} = x_k(1+(1-x_k^2)(1-\frac{1}{2}x_k^2)) = x_k(2-\frac{3}{2}x_k^2-\frac{1}{2}x_k^4) = x_k p_2(x_k^2).$$

For the method (6), it is clear that it can be formed as the following scalar iterative method:

$$x_{k+1} = x_k(1+\frac{5}{4}(1-x_k^2)(\frac{2}{5}+1-x_k^2)) = x_k(\frac{11}{4}-3x_k^2+\frac{5}{4}x_k^4) = x_k p_3(x_k^2).$$

In general, according to all p_is, $i = 1,2,3$, a generalized scalar iterative method

$$x_{k+1} = x_k p(x_k^2), \tag{14}$$

can be considered, which is applied to find a solution of the equation $x = x^{-1}$ $(x \neq 0)$ or equivalently $x^2 = 1$. The defined polynomial of degree d is as follows:

$$p(x) = l_0 + l_1 x + \cdots + l_d x^d.$$

Utilizing the newly defined polynomial $p(x)$ and iterative method (14), we present a systematic proof for the convergence of the proposed generalized method. This proof is structured through several steps that draw upon the methodology established in Petković's strategy [16].

Proposition 1. *Assume that the iterative method* (14) *is convergent, it means* $x_k^2 \to 1$ *when* $k \to \infty$. *Then,* $p(1) = 1$ *holds.*

Proof. The convergence $\lim_{k\to\infty} x_k^2 = 1$ implies the convergence $\lim_{k\to\infty} x_k = 1$ with the assumption that x_k is positive. Taking the limit of both sides of (14) and noticing the continuously of polynomial $p(x)$ yield $1 = 1.p(1)$, we can get $p(1) = 1$. □

Therefore, through out this paper, we assume that $p(1) = 1$. Denote $e_k = 1 - x_k^2$ as the error sequence such that it is satisfied in

$$e_{k+1} = 1 - x_{k+1}^2 = 1 - x_k^2 p^2(x_k^2) = 1 - (1-e_k)p^2(1-e_k) = q(e_k),$$

in which $q(x) = 1 - (1-x)p^2(1-x)$ is a polynomial of degree $2d+1$. Since $p(1) = 1$, we have $q(0) = 0$.

Lemma 1. *The polynomial* $q(x)$ *satisfies in*

$$q(x) = -\sum_{i=0}^{2d} h_i \sum_{j=1}^{i+1} \binom{i+1}{j} (-1)^j x^j,$$

in which $p^2(x) = h(x) = \sum_{i=0}^{2d} h_i x^i$.

Proof. According to $p^2(x) = h(x) = \sum_{i=0}^{2d} h_i x^i$ and the definition of $q(x)$, we have

$$\begin{aligned} q(x) =& 1 - (1-x)p^2(1-x) \\ =& 1 - \sum_{i=0}^{2d} h_i (1-x)^{i+1} = 1 - \sum_{i=0}^{2d} h_i \sum_{j=0}^{i+1} \binom{i+1}{j} (-1)^j x^j. \end{aligned}$$

Since $p(1) = 1$, $p^2(1) = 1$ and $\sum_{i=0}^{2d} h_i = 1$ hold. In the last sum of the above equation, by separating the term related to $j = 0$ and applying $\sum_{i=0}^{2d} h_i = 1$, the desired result is obtained. □

Proposition 2. *Assume* $q(0) = 0$ *(*$p(1) = 1$*). If* $|q'(0)| < 1$ *(*$|2p'(1)+1| < 1$*), then there is an* $\varepsilon > 0$ *such that for every* $e_0 \in (-\varepsilon, \varepsilon)$, $e_k \to 0$ *(*$x_k^2 \to 1$*) when* $k \to \infty$.

Proof. The fixed point sequence $e_{k+1} = q(e_k)$ satisfies $q(0) = 0$. Therefore, there is a neighborhood $(-\varepsilon, \varepsilon)$ of 0 such that if $|q'(0)| < 1$ then $|q'(x)| < 1$ holds for every $x \in (-\varepsilon, \varepsilon)$. This means the fixed point sequence $e_{k+1} = q(e_k)$ is contractive, so converges to $x = 0$. □

In the sequel, we assume that $|q'(0)| < 1$ ($|2p'(1)+1| < 1$). Taking $t \geq 1$ as the order of the root $x = 0$ of the polynomial $q(x)$, we are to get

$$q(x) = x^t v(x),$$

in which $v(x)$ is a polynomial of degree $2d+1-t$ and satisfies in $v(0) \neq 0$. The definition of $q(x)$ results in

$$q(1) = 1 \Rightarrow v(1) = 1.$$

On the other hand, according to $q(x)$, $q_0 = q_1 = \cdots = q_{t-1} = 0$ and $q_t \neq 0$. Therefore,

$$v(x) = \sum_{i=0}^{2d+1-t} q_{t+i}x^i, \quad q(x) = x^t v(x) = \sum_{i=0}^{2d+1-t} q_{t+i}x^{t+i}.$$

Let $g(x) = xp(x^2)$. If the sequence $\{e_k\}_{k\geq 0}$ has the order of convergence equal to t, then $q^{(i)}(0) = 0$, $i = 1,\ldots,t-1$. Since $q'(0) = 0$ and $p(1) = 1$, we obtain $p'(1) = -\frac{1}{2}$. It can be observed that

$$g'(x) = p(x^2) + 2x^2p'(x^2) \Rightarrow g'(1) = 1 + 2(-\frac{1}{2}) = 0.$$

Accordingly,

$$p''(1) = \frac{3}{4}, \quad p'''(1) = -\frac{5}{8}, \ldots,$$

imply $g^{(i)}(1) = 0$, $i = 2,\ldots,t-1$. In below, we show that both fixed point sequences $e_{k+1} = q(e_k)$ and $x_{k+1} = g(x_k)$ have the same order of convergence.

Proposition 3. *The order of convergence, denoted by t, is identical for both the fixed point sequence $e_{k+1} = q(e_k)$ and the method (14).*

Proof. We have $e_{k+1} = q(e_k) = e_k^t v(e_k)$. It results

$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k^t} = \lim_{k\to\infty} \frac{(1-x_{k+1}^2)}{(1-x_k^2)^t} = v(0) \neq 0.$$

Therefore, the sequence $\{e_k\}_{k=0}^{\infty}$ and then the method (14) both have the same order of convergence equal to t. □

So, to find the determinable bound for ε in Proposition 2, define polynomials $\hat{q}(x)$ and $\hat{v}(x)$ by

$$\hat{v}(x) = \sum_{i=0}^{2d+1-t} |q_{t+i}|x^i, \quad \hat{q}(x) = x^t\hat{v}(x).$$

It is clear that $q(x) \leq \hat{q}(x)$ for every $x \geq 0$ and $\hat{q}(0) = 0$. Moreover, in the case $t = 1$, we have

$$\hat{v}(x) = \sum_{i=0}^{2d} |q_{1+i}|x^i, \quad q'(x) = (xv(x))' = \sum_{i=0}^{2d}(1+i)q_{1+i}x^i = q_1 + 2q_2x + 3q_3x^2 + \cdots + (1+2d)q_{1+2d}x^{2d},$$

and

$$\hat{v}(0) = |q_1| = |q'(0)| < 1.$$

Lemma 2. *The equation $\hat{q}(x) = x$ has the unique root γ in the interval $(0,+\infty)$. If $\hat{q}(x) = q(x)$, which means all coefficients of $q(x)$ are non-negative, then $\gamma = 1$, otherwise $0 < \gamma < 1$. In addition, $\hat{q}(x) < x$ holds for every $x \in (0,\gamma)$.*

Proof. Noticing the equation $\hat{q}(x) = x$, one can obtained $x^t\hat{v}(x) = x$, so $x^{t-1}\hat{v}(x) = 1$. Take $s(x) = x^{t-1}\hat{v}(x) - 1$. Since the coefficients of $\hat{v}(x)$ are non-negative and the degree of $s(x)$ is equal to $2d \geq 2$, the coefficients of $s'(x)$ are non-negative as well and $s'(x) > 0$ holds for every $x > 0$. Hence, the function $s(x)$ is strictly increasing for every $x > 0$ and satisfies $s(+\infty) = +\infty$. On the other hand,

$$s(0) = \begin{cases} |q_1| - 1, & t = 1 \\ -1, & t > 1. \end{cases}$$

Notice that in the case $t = 1$, according to the assumption $|q_1| = |q'(0)| < 1$, $s(0) < 0$ consistently holds. Further, since $\hat{v}(1) \geq v(1) = 1$, we have $\hat{v}(1) - 1 \geq 0$. Thus, it results in $s(1) = \hat{v}(1) - 1 \geq 0$. It means that the strictly increasing function $s(x)$ has a unique root γ in the interval $(0, 1]$. If $\hat{q}(x) = q(x)$, then $\hat{q}(1) = q(1) = 1$ resulting in $\hat{v}(1) = 1$. Therefore, $s(1) = 0$ is obtained, it means $\gamma = 1$ is the unique root of $s(x)$. Otherwise, $\gamma \in (0, 1)$ is derived. Assume that $x \in (0, \gamma)$. Since $s(x)$ is strictly increasing on this interval, we obtain $s(x) < 0$. Hence, $x^{t-1}\hat{v}(x) < 1$ and multiplying both sides of the inequality by $x > 0$ result in $\hat{q}(x) = x^t\hat{v}(x) < x$. □

Lemma 3. *If the sequence* $\{f_k\}$ *satisfying* $0 \leq f_0 < \gamma$ *and* $0 \leq f_{k+1} \leq \hat{q}(f_k)$, *then* $\lim_{k\to\infty} f_k = 0$ *holds.*

Proof. Using mathematical induction, we show $0 \leq f_k < \gamma$ and $0 \leq f_{k+1} \leq f_k$, for all k. Since $f_0 \in (0, \gamma)$, it is true that $0 \leq f_1 \leq \hat{q}(f_0) < f_0 < \gamma$. Suppose that $f_{k-1} < \gamma$ and $0 \leq f_k \leq f_{k-1}$. As a result, $f_k < \gamma$ and $0 \leq f_{k+1} \leq \hat{q}(f_k) < f_k < \gamma$ are obtained. So, $0 \leq f_{k+1} \leq f_k < \gamma$ completes the induction and shows that the sequence $\{f_k\}_{k\geq 0}$ is decreasing. On the other hand, since $0 \leq f_k < \gamma$, the sequence $\{f_k\}_{k\geq 0}$ is bounded from below by zero. Therefore, it is convergent to a f, which satisfies in $0 \leq f < \gamma$ and $f \leq \hat{q}(f)$. Now, it is shown $f = 0$. Let $f \neq 0$. Since $f \leq \hat{q}(f)$ and $\hat{q}(f) < f$ are true, hence $f < f$, which is a contradiction. □

Since $|e_{k+1}| = |q(e_k)| \leq \hat{q}(|e_k|) \leq |e_k|$ holds for all k, taking $f_k = |e_k|$, we have the following corollary for the convergence of the method (14).

Corollary 1. *The method* (14) *is convergent if* $|e_0| < \gamma$.

Therefore, the value of bound ε in Proposition 2 is equal to γ.

Now, to find coefficients of the polynomial $p(x)$ such that the method (14) has the order of convergence t, from $q(x) = 1 - (1-x)p^2(1-x)$, the following can be immediately obtained:

$$p^2(1-x) = \frac{1-q(x)}{1-x} = \frac{1-x^t v(x)}{1-x} = \frac{1-x^t}{1-x} + \frac{1-v(x)}{1-x}x^t.$$

Regarding the definition of $q(x)$, it is easy to see $q(1) = 1$ and $v(1) = 1$. Thus, it is evident that $v(x) = 1 - (1-x)u(x)$, in which $u(x)$ is a polynomial of degree $2d - t$. Accordingly, the following condition is obtained:

$$p^2(1-x) = 1 + x + x^2 + \cdots + x^{t-1} + x^t u(x). \tag{15}$$

All results of this section are abridged in the following theorem and the convergence of the iterative method (14) is proven under certain assumption.

Theorem 1. *Assume $x_k \neq 0$ and the given polynomial $p(x)$, which $p(1) = 1$. The method (14) locally converges to 1, if $|2p'(1)+1| < 1$ and $|1-x_0^2| < \gamma$. In addition, if $p(x)$ is given by (15), then the method (14) has convergence order t.*

Using the convergent iterative method (14), we define a new generalized method

$$U_{k+1} = U_k p(U_k^* U_k), \quad U_0 = A, \tag{16}$$

to find the factor U of the polar decomposition of the matrix A. In the following theorem, the convergence of the method (16) to find the polar decomposition of the matrix A is proven by using the singular value decomposition under the assumption (12).

Theorem 2. *Let $A \in \mathbb{C}^{m\times n}$ is an arbitrary matrix with rank $r \leq n$ and has non-zero singular values $0 < \sigma_r \leq \cdots \leq \sigma_2 \leq \sigma_1 \leq 1$ with condition $1-\gamma < \frac{1}{\|(A^\dagger)^* A^\dagger\|_2}$. Then, the sequence $\{U_k\}_{k\geq 0}$, generated by the method (16) with the initial matrix $U_0 = A$, that converges to the unitary factor U of the polar decomposition of the matrix A with the order of convergence equal to t.*

Proof. Assume the matrix A has the following SVD:

$$A = P \begin{pmatrix} D_0 \\ 0 \end{pmatrix} Q^*,$$

in which P and Q are unitary matrices and $D_0 = diag(d_1^{(0)}, d_2^{(0)}, \ldots, d_r^{(0)}, 0, \ldots, 0)$ is a diagonal matrix that $d_i^{(0)} = \sigma_i$, $i = 1, \ldots, r$. For $k = 0$, since $U_0 = A$, we have

$$P^* U_0 Q = \begin{pmatrix} D_0 \\ 0 \end{pmatrix}.$$

Take $D_{k+1} = D_k p(D_k^2)$. Then, the following comes forth:

$$P^* U_{k+1} Q = \begin{pmatrix} D_{k+1} \\ 0 \end{pmatrix},$$

in which $D_{k+1} = diag(d_1^{(k+1)}, d_2^{(k+1)}, \ldots, d_r^{(k+1)}, 0, \ldots, 0)$. Based on (14),

$$d_i^{(k+1)} = d_i^{(k)} p((d_i^{(k)})^2), \quad i = 1, 2, \ldots, r.$$

According to the condition $1-\gamma < \frac{1}{\|(A^\dagger)^* A^\dagger\|_2}$, it can be concluded that $1-\gamma < \sigma_r^2$ and then the non-zero values of diagonal matrix D_0 are satisfied in $|1-(d_i^{(0)})^2| < \gamma$, $i = 1, \ldots, r$. Thus, based on Theorem 1, the sequence $\{d_i^{(k)}\}_{k\geq 0}$ converges to 1 with the order of convergence equal to t. Hence, $D_k \to \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix}$ when $k \to \infty$. Therefore,

$$\lim_{k\to\infty} P^* U_k Q = \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix},$$

it means $\lim_{k\to\infty} U_k = P \begin{pmatrix} I_r & 0 \\ 0 & 0 \end{pmatrix} Q^* = P_1 Q_1^*$ and the sequence $\{U_k\}_{k\geq 0}$ converges to the factor U with the order of convergence equal to t. □

3 New iterative methods

In this section, we present several new iterative methods and subsequently introduce a strategy aimed at enhancing the order of convergence with a reduction in the number of required matrix multiplications as the computational costs. The derivation of the polynomial $p(x)$ is facilitated through the utilization of equation (15). To achieve this, we scrutinize the process of determining the polynomial $p(x)$ based on a generic form. Initially, iterative methods with first and second orders are obtained, followed by the introduction of a comprehensive model for higher orders. Assuming that the polynomial $p(x)$ complies with the conditions outlined in equation (15), the ensuing analysis unfolds the implications of this satisfaction,

$$p^2(x) = 1 + (1-x) + (1-x)^2 + \cdots + (1-x)^{t-1} + (1-x)^t u(1-x).$$

Hence, in the sequel, we compute $p(1-z)$ and by taking $z = 1-x$, the desired result is obtained. If $p(x)$ is the polynomial of degree of d, then the degree of $p^2(1-x)$ is $2d$. Therefore, we assume

$$p^2(1-x) = c^2(\alpha_0 + \alpha_1 x + \cdots + \alpha_{d-1}x^{d-1} + x^d)^2.$$

Since $p(1) = 1$, it is clear that $c^2\alpha_0^2 = 1$ always holds. To acquire the class of iterative methods with order one, we consider the polynomial $p(1-x)$ as follows:

$$p(1-x) = c(\alpha_0 + x).$$

Applying the condition (15), when $t = 1$, we obtain

$$p^2(1-x) = c^2(\alpha_0+x)^2 = c^2(\alpha_0^2 + 2\alpha_0 x + x^2) = c^2\alpha_0^2 + x(2c^2\alpha_0 + c^2 x) = 1 + xu(x).$$

Choosing $c > 0$ implies $\alpha_0 = \frac{1}{c}$. So, the class of iterative methods with order one is obtained as follows:

$$x_{k+1} = x_k p(x_k^2) = cx_k(\frac{1}{c} + (1-x_k^2)), \quad c > 0.$$

Applying the condition $|2p'(1)+1| < 1$, the valid range for c is $0 < c < 1$. The most appropriate value of c is determined when $2p'(1)+1 = 0$, meaning $c = \frac{1}{2}$. In this case, if we take $x_{k+1} = \frac{1}{2}x_k(2+(1-x_k^2)) = g_1(x_k)$, then it is observed that $g_1'(1) = 0$. So, the following iterative method

$$\begin{cases} U_0 = A, \quad R_k = I - U_k^* U_k \\ U_{k+1} = \frac{1}{2}U_k(2I + R_k), \quad k = 0, 1, \ldots, \end{cases} \tag{17}$$

with order one is derived, which is the method (3). Assuming $A \in \mathbb{C}^{m\times n}$, each matrix multiplication costs $\mathrm{O}(mn^2)$ flops, then this method costs $(2 \times \mathrm{O}(mn^2))$ flops in each iteration.

To obtain a class of iterative methods with order two, we consider the function $p(1-x)$ as follows:

$$p(1-x) = c(\alpha_0 + \alpha_1 x + x^2) = c(\alpha_0 + x(\alpha_1 + x)).$$

Using (15), we have

$$p^2(1-x) = c^2(\alpha_0 + \alpha_1 x + x^2)^2$$

$$
\begin{aligned}
&= c^2((\alpha_0^2)+(2\alpha_0\alpha_1)x+(\alpha_1^2+2\alpha_0)x^2+(2\alpha_1)x^3+x^4)\\
&= c^2\alpha_0^2+c^2(2\alpha_0\alpha_1)x+x^2(c^2(\alpha_1^2+2\alpha_0)+2c^2\alpha_1 x+c^2x^2)\\
&= 1+x+x^2u(x),
\end{aligned}
$$

which yields $c^2\alpha_0^2=1$ again and also $c^2(2\alpha_0\alpha_1)=1$. Thus, the coefficients $\alpha_0=\frac{1}{c}$ and $\alpha_1=\frac{1}{2c}$ are calculated with $c>0$. Take

$$x_{k+1}=cx_k(\frac{1}{c}+\frac{1}{2c}(1-x_k^2)+(1-x_k^2)^2)=g_2(x_k).$$

The function $g_2(x)$ has the local maximizer $x=\sqrt{\frac{3+2c}{10}}$ on the interval $(0,1)$ when $c>\frac{3}{8}$. On the other hand,

$$g_2(x)-x=x(1-x^2)(\frac{1}{2}+c(1-x^2))=0,$$

which yields the fixed points of the function $g_2(x)$ as follows:

$$x=\pm1,0,\pm\sqrt{\frac{2c+1}{2c}}.$$

For the method $x_{k+1}=g_2(x_k)$ to be convergent, the function $g_2(x)$ must map the interval $(0,\sqrt{\frac{2c+1}{2c}})$ into itself, so $g_2(\sqrt{\frac{3+2c}{10}})<\sqrt{\frac{2c+1}{2c}}$. Therefore, $0<c<1.47223$ and the following class of iterative methods with order two is obtained:

$$
\begin{cases}
U_0=A, \quad R_k=I-U_k^*U_k,\\
U_{k+1}=cU_k(R_k(R_k+\frac{1}{2c}I)+\frac{1}{c}I), \quad k=0,1,\ldots,
\end{cases}
\tag{18}
$$

which contains only three matrix multiplications and costs $(2\times O(mn^2)+O(n^3))$ flops in each iteration. To find a suitable c, using $c_i=0.1+0.1i$, $i=0,1,\ldots,13$, different methods based on each c_i are compared for random matrices. The value $c=1.3$ outperforms others. It should be noted that based on this class of iterative methods, the methods (5) and (6) are obtained by choosing $c=\frac{1}{2}$ and $c=\frac{5}{4}$, respectively.

In the general case, take

$$p(1-x)=c(\alpha_0+\alpha_1x+\cdots+\alpha_{d-1}x^{d-1}+x^d).$$

To achieve higher order methods, based on Theorem 1, we apply the condition (15) for above $p(1-x)$ with $d\geq1$. Then, a nonlinear system in terms of c is created. For $c>0$, we can solve the system by forward substitution. For example, the coefficients of the polynomial $p(1-x)$ of degrees $d=1,\ldots,8$ are calculated as follows:

$$
\begin{aligned}
&\alpha_0=\frac{1}{c}, \quad \alpha_1=\frac{1}{2c}, \quad \alpha_2=\frac{3}{8c}, \quad \alpha_3=\frac{5}{16c}, \quad \alpha_4=\frac{35}{128c},\\
&\alpha_5=\frac{63}{256c}, \quad \alpha_6=\frac{231}{1024c}, \quad \alpha_7=\frac{429}{2048c}.
\end{aligned}
\tag{19}
$$

According to these coefficients, for $d=3$, we can define

$$g_3(x)=cx(\frac{1}{c}+\frac{1}{2c}(1-x^2)+\frac{3}{8c}(1-x^2)^2+(1-x^2)^3).$$

For every $c>0$, $g_3'(1)=0$ and $g_3''(1)=0$ consistently hold. Similar to the case of $d=1$, the most appropriate value for c is obtained when $g_3'''(1)=0$, resulting $c=\frac{5}{16}$. According to this value of c, we have $g_3^{(4)}(1)=-105$ and the following third-order iterative method is attained:

$$\begin{cases} U_0=A, \quad R_k=I-U_k^*U_k, \quad B_k=R_k^2, \\ U_{k+1}=\frac{1}{16}U_k(B_k(5R_k+6I)+8R_k+16I), \quad k=0,1,\ldots. \end{cases} \tag{20}$$

This method contains four matrix multiplications and costs $(2\times \mathrm{O}(mn^2)+2\times \mathrm{O}(n^3))$ flops per iteration.

It is observed that methods with even order convergence perform better. Thus, we introduce a strategy to obtain classes of iterative methods with even orders ($t\geq 4$) and at least required matrix multiplications. Similar to $p(1-x)=c(x(x+\alpha_1)+\alpha_0)$ and the technique used in [17], assume the polynomials w_i are as follows:

$$\begin{aligned} w_i(x)=w_{i-1}(x)[w_{i-1}(x)+a_{i,i-2}w_{i-2}(x)+\cdots+a_{i,0}w_0(x)] \\ +b_{i,i-2}w_{i-2}(x)+b_{i,i-3}w_{i-3}(x)+\cdots+b_{i,0}w_0(x), \end{aligned} \tag{21}$$

in which $w_0(x)=1$, $w_1(x)=x$, and $w_2(x)=x^2$. Now, take $p(1-x)=cw_{\theta-1}(x)$. Using (21) and based on (14), we can obtain classes of iterative methods which contain θ matrix multiplications. To this end, if

$$w_{\theta-1}(x)=\sum_{i=0}^{d}\alpha_i x^i, \quad \theta=4,5,6,\ldots, \quad d=4,8,16,\ldots, \tag{22}$$

then the condition (15) implies some results similar to (19) with assumption $c>0$. Now, to find coefficients $a_{i,j}$ and $b_{i,j}$, where $j=i-l$, $l=i-2,\ldots,0$ and $i=\theta-1,\ldots,3$, a nonlinear system must be solved.

For $\theta=4$, the unknown coefficients are identical to those in (19). Thus,

$$\begin{aligned} p(1-x)=cw_3(x)&=c(w_2(x)(w_2(x)+\frac{5}{16c}w_1(x)+\frac{3}{8c}w_0(x))+\frac{1}{2c}w_1(x)+\frac{1}{c}w_0(x)) \\ &=c(x^2(x^2+\frac{5}{16c}x+\frac{3}{8c})+\frac{1}{2c}x+\frac{1}{c}), \end{aligned}$$

is applied to obtain a fourth-order method. Take $x_{k+1}=cx_kw_3(1-x_k^2)=g_4(x_k)$. The points $x=0$ and $x=1$ are two of the fixed points of the function $g_4(x)$ when $x\geq 0$. Moreover, this function has another fixed point $\beta_4>1$, when $c>0$ and $x\geq 0$. If $c>\frac{1680}{6144}$, then there is a local maximizer of the function $g_4(x)$ on the interval $(0,1)$, say ζ. In order to find the range of c, we apply the condition $g_4(\zeta)<\beta_4$, which guarantees that the function $g_4(x)$ maps the interval $(0,\beta_4)$ into itself. It yields $0<c<2.387437$. The class of fourth-order iterative methods is obtained as follows:

$$\begin{cases} U_0=A, \quad R_k=I-U_k^*U_k, \quad B_k=R_k^2, \\ U_{k+1}=cU_k(B_k(B_k+\frac{5}{16c}R_k+\frac{3}{8c}I)+\frac{1}{2c}R_k+\frac{1}{c}I), \quad k=0,1,\ldots, \end{cases} \tag{23}$$

which contains four matrix multiplications and costs $(2\times \mathrm{O}(mn^2)+2\times \mathrm{O}(n^3))$ flops in each iteration.

A suitable $c=1.8$ is obtained by using a similar accomplished numerical comparison for quadratic method (18).

For $\theta=5$, based on (21), we have

$$w_3(x)=x^2[x^2+a_{3,1}x+a_{3,0}]+b_{3,1}x+b_{3,0},$$
$$w_4(x)=w_3(x)[w_3(x)+a_{4,2}x^2+a_{4,1}x+a_{4,0}]+b_{4,2}x^2+b_{4,1}x+b_{4,0},$$

in which unknown coefficients $a_{i,j}$ and $b_{i,j}$, $i=4,3$ and $j=2,1,0$, are computed by using (22) and obtained coefficients in (19). Having expanded $w_4(x)$, we have the following nonlinear system:

$$\begin{cases} b_{3,0}^2+a_{4,0}b_{3,0}+b_{4,0}=\frac{1}{c}\\ 2b_{3,1}b_{3,0}+a_{4,1}b_{3,0}+a_{4,0}b_{3,1}+b_{4,1}=\frac{1}{2c}\\ b_{3,1}^2+2a_{3,0}b_{3,0}+a_{4,2}b_{3,0}+a_{4,1}b_{3,1}+a_{4,0}a_{3,0}+b_{4,2}=\frac{3}{8c}\\ 2a_{3,1}b_{3,0}+2a_{3,0}b_{3,1}+a_{4,2}b_{3,1}+a_{4,1}a_{3,0}+a_{4,0}a_{3,1}=\frac{5}{16c}\\ a_{3,0}^2+2a_{3,1}b_{3,1}+2b_{3,0}+a_{4,2}a_{3,0}+a_{4,1}a_{3,1}+a_{4,0}=\frac{35}{128c}\\ 2b_{3,1}+2a_{3,1}a_{3,0}+a_{4,2}a_{3,1}+a_{4,1}=\frac{63}{256c}\\ 2a_{3,0}+a_{3,1}^2+a_{4,2}=\frac{231}{1024c}\\ 2a_{3,1}=\frac{429}{2048c} \end{cases}$$

To solve the above system, define

$$\eta_1=\frac{429}{4096c},\quad \eta_2=\frac{231}{1024c}-\eta_1^2,\quad \eta_3=\frac{63}{256c}-\eta_1\eta_2,$$
$$\eta_4=\frac{35}{128c}-\eta_1\eta_3,\quad \eta_5=\frac{5}{16c}-\eta_1\eta_4.$$

Then,

$$\begin{cases} a_{3,1}=\eta_1,\\ a_{4,2}=\eta_2-2a_{3,0},\\ a_{4,1}=\eta_3-2b_{3,1}\\ a_{4,0}=\eta_4-a_{3,0}(\eta_2-a_{3,0})-2b_{3,0},\\ b_{4,0}=\frac{1}{c}-b_{3,0}(\eta_4-a_{3,0}(\eta_2-a_{3,0})-b_{3,0}),\\ b_{4,1}=\frac{1}{2c}-b_{3,0}(\eta_3-2b_{3,1})-b_{3,1}(\eta_4-a_{3,0}(\eta_2-a_{3,0})),\\ b_{4,2}=\frac{3}{8c}-b_{3,1}(\eta_3-b_{3,1})-b_{3,0}(\eta_2-2a_{3,0})-a_{3,0}(\eta_4-a_{3,0}(\eta_2-a_{3,0})),\\ \eta_1a_{3,0}^2+a_{3,0}(\eta_3-\eta_1\eta_2-2b_{3,1})+\eta_2b_{3,1}=\eta_5. \end{cases} \tag{24}$$

Determining $b_{3,1}$, we find an acceptable solution for the quadratic equation

$$\eta_1a_{3,0}^2+a_{3,0}(\eta_3-\eta_1\eta_2-2b_{3,1})+\eta_2b_{3,1}=\eta_5, \tag{25}$$

which is in terms of $a_{3,0}$. Observe that the discriminant of (25) is as follows:

$$\Delta = (\eta_3 - \eta_1\eta_2 - 2b_{3,1})^2 - 4\eta_1(\eta_2 b_{3,1} - \eta_5).$$

If $\Delta \geq 0$, then (25) has at least a solution and using this condition, we have

$$\Delta = 4b_{3,1}^2 - 4\eta_3 b_{3,1} + (\eta_3 - \eta_1\eta_2)^2 + 4\eta_1\eta_5 \geq 0,$$

which shows $\Delta = 0$ is a quadratic equation in terms of $b_{3,1}$. The inequality $\Delta \geq 0$ is true if the discriminant of the equation $\Delta = 0$ is non-positive. Taking non-positive the discriminant of the equation $\Delta = 0$, we obtain an inequality in terms of c that implies $c > 0$. If the discriminant of the equation $\Delta = 0$ equals to zero, then $b_{3,1} = \frac{\eta_3}{2}$ is a solution of the equation $\Delta = 0$ and the inequality $\Delta \geq 0$. Thus, the following solution can be obtained for the equation (25):

$$a_{3,0} = \frac{\eta_1\eta_2 + \sqrt{\eta_1^2\eta_2^2 - 4\eta_1(\frac{\eta_2\eta_3}{2} - \eta_5)}}{2\eta_1}.$$

Take

$$g_5(x) = xp(1-x^2) = cx(\alpha_0 + \alpha_1(1-x^2) + \cdots + \alpha_7(1-x^2)^7 + (1-x^2)^8),$$

in which the coefficients α_i are obtained according to (19). Now, choosing $b_{3,0} = 0$ yields that two functions $g_5(x)$ and $cxw_4(1-x^2)$ are equal, when $c > 0$, and then all coefficients in (24) can be obtained in terms of c. On the other hand, when $0 < c < 3.7275$, the function $g_5(x)$ maps the interval $(0, \beta_5)$ into itself, in which $\beta_5 \geq 1$ is a fixed point of the function $g_5(x)$. According to the obtained coefficients $a_{i,j}$ and $b_{i,j}$, $i = 4, 3$ and $j = 2, 1, 0$, the following class of eight-order iterative methods is acquired:

$$\begin{cases} U_0 = A, \quad R_k = I - U_k^* U_k, \quad B_k = R_k^2, \\ w_3(R_k) = B_k[B_k + a_{3,1}R_k + a_{3,0}I] + b_{3,1}R_k + b_{3,0}I, \\ w_4(R_k) = w_3(R_k)[w_3(R_k) + a_{4,2}B_k + a_{4,1}R_k + a_{4,0}I] + b_{4,2}B_k + b_{4,1}R_k + b_{4,0}I, \\ U_{k+1} = cU_k w_4(R_k), \quad k = 0, 1, \ldots, \end{cases}$$

which contains only five matrix multiplications and costs $(2 \times \mathrm{O}(mn^2) + 3 \times \mathrm{O}(n^3))$ in each iteration. Similar to the two previously introduced methods, using numerical comparison, we choose $c = 3.4$. Then, after rounding the computation to four decimal places, we obtain

$$\begin{cases} U_0 = A, \quad R_k = I - U_k^* U_k, \quad B_k = R_k^2, \\ w_3(R_k) = B_k[B_k + 0.0308R_k + 1.7162I] + 0.0340R_k, \\ w_4(R_k) = w_3(R_k)[w_3(R_k) - 3.3671B_k + 2.9116I] - 4.8879B_k + 0.0482R_k + 0.2941I, \\ U_{k+1} = 3.4U_k w_4(R_k), \quad k = 0, 1, \ldots. \end{cases} \tag{26}$$

In the sequel, we list functions $y = g_i(x)$, $i = 1, \ldots, 5$, with considered values of c are as follows:

$$g_1(x) = \frac{x}{2}(2 + (1-x^2)),$$
$$g_2(x) = \frac{x}{10}(10 + 5(1-x^2) + 13(1-x_k^2)^2),$$
$$g_3(x) = \frac{x}{16}(16 + 8(1-x^2) + 6(1-x^2)^2 + 5(1-x^2)^3),$$

$$g_4(x) = \frac{x}{80}(80+40(1-x^2)+30(1-x^2)^2+25(1-x^2)^3+144(1-x^2)^4),$$
$$g_5(x) = \frac{x}{10240}(10240+5120(1-x^2)+3840(1-x^2)^2+3200(1-x^2)^3+2800(1-x^2)^4$$
$$+2520(1-x^2)^5+2310(1-x^2)^6+2145(1-x^2)^7+34816(1-x^2)^8).$$

The graph of this functions are illustrated in Figure 1, which declare $x=0$ and $x=1$ are fixed points of these functions and it is clear that the function $g_5(x)$ increase faster than others on the interval $(0,\beta_5)$. This feature of $g_5(x)$ is the important motivation of using it for the method (26).

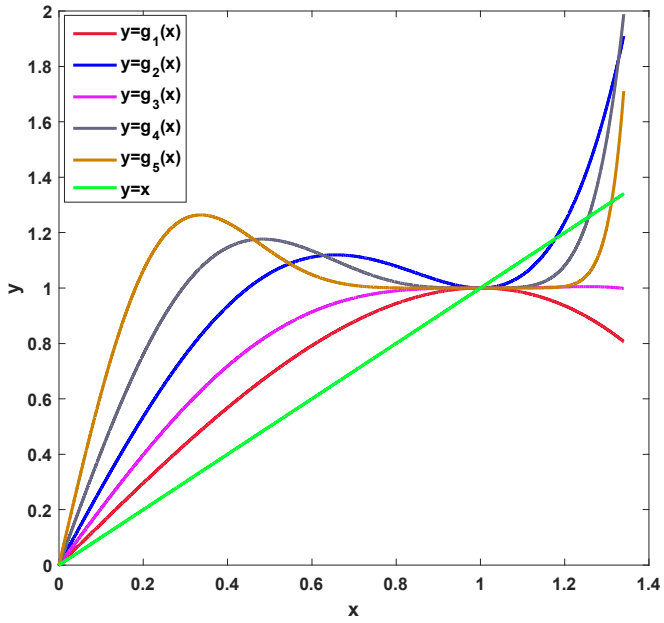

Figure 1: The graph of functions $y = g_i(x)$, $i = 1,\ldots,5$ and $y = x$

4 Numerical experiments

In this section, to demonstrate the performance of methods, some examples are given, in which the matrix U of the polar decomposition $A = UH$ is computed. All matrices A used in examples follow the assumptions (12) and (13). The calculations of this part have been run by MATLAB R2015b software on a PC with specifications AMD, E2-1800, APU, 1.70 GHz. In examples, methods (3) (M_1), (4) (M_2), (5) (M_3), (6) (M_4), (7) (M_5), (8) (M_6), (9) (M_7), (10) (M_8), (11) (M_9), (18) (M_{10}), (20) (M_{11}), (23) (M_{12}), and (26) (M_{13}) are compared for different matrices. The stop criterion is

$$\frac{\|U_{k+1}-U_k\|_1}{\|U_k\|_1} < 0.5\times 10^{-8},$$

and the maximum number of iterations is set to 200. Moreover, the factor U is obtained using the SVD by MATLAB and then

$$Error = \|U_{k+1}-U\|_F,$$

is computed for each solution of methods, which $\|.\|_F$ is the Frobenius norm. The average elapsed execution time (CPU), the average number of iterations (Iter), the average number of matrix multiplications (TMM), and the average *Error* as the precision (Error) are reported for each method. Although matrix inversion is computationally expensive than matrix multiplication, we assume equal computational costs for these operations to facilitate a simpler exposition and a more straightforward comparison of the proposed methods. All reported results in tables are rounded to two decimal places of scientific notation.

Example 1. In this example, for each n, $n = 80, 100, 120, 150, 180, 200$, we compute the factor U of ten generated random matrices as follows:

$$A := (rand(n,n) - rand(n,n))/n.$$

The obtained results of the methods are reported in Table 1, Figure 2, and Figure 3.

Table 1: The comparison of methods for various matrices with $n = 80$, 100, 120, 150, 180, 200 in Example 1

		$n=80$					$n=100$		
Method	CPU	Iter	TMM	Error	Method	CPU	Iter	TMM	Error
M_1	3.64×10^{-2}	24.3	48.6	4.56×10^{-14}	M_1	6.19×10^{-2}	24.2	48.4	3.11×10^{-14}
M_2	1.82×10^{-1}	10.5	42	3.10×10^{-14}	M_2	2.12×10^{-1}	10.5	42	3.54×10^{-14}
M_3	3.34×10^{-2}	15.4	46.2	3.39×10^{-14}	M_3	5.93×10^{-2}	15.4	46.2	3.36×10^{-14}
M_4	3.09×10^{-2}	13.6	40.8	3.39×10^{-14}	M_4	5.51×10^{-2}	13.5	40.5	3.09×10^{-14}
M_5	1.55×10^{-1}	10.5	49.5	8.55×10^{-14}	M_5	2.50×10^{-1}	10.4	48.8	9.87×10^{-14}
M_6	1.14×10^{-1}	6.9	41.4	4.58×10^{-14}	M_6	1.39×10^{-1}	6.7	40.2	4.84×10^{-14}
M_7	1.26×10^{-1}	6.2	43.4	2.16×10^{-13}	M_7	1.45×10^{-1}	6.1	42.7	2.09×10^{-13}
M_8	1.17×10^{-1}	5.9	41.3	3.12×10^{-13}	M_8	1.31×10^{-1}	5.9	41.3	3.01×10^{-13}
M_9	1.11×10^{-1}	6.4	44.8	1.55×10^{-13}	M_9	1.65×10^{-1}	6.6	46.2	1.09×10^{-13}
M_{10}	2.90×10^{-2}	13.5	40.5	2.54×10^{-14}	M_{10}	5.03×10^{-2}	13.5	40.5	3.08×10^{-14}
M_{11}	4.89×10^{-2}	14.3	57.2	3.13×10^{-14}	M_{11}	7.69×10^{-2}	14.3	57.2	3.28×10^{-14}
M_{12}	2.61×10^{-2}	9.5	38	2.73×10^{-14}	M_{12}	4.64×10^{-2}	9.5	38	2.94×10^{-14}
M_{13}	2.63×10^{-2}	7.2	36	4.97×10^{-14}	M_{13}	4.69×10^{-2}	7.5	37.5	5.28×10^{-14}
		$n=120$					$n=150$		
Method	CPU	Iter	TMM	Error	Method	CPU	Iter	TMM	Error
M_1	1.02×10^{-1}	24.1	48.2	4.65×10^{-14}	M_1	1.70×10^{-1}	25.4	50.8	8.97×10^{-14}
M_2	1.63×10^{-1}	10.2	40.8	4.61×10^{-14}	M_2	3.38×10^{-1}	10.8	43.2	7.30×10^{-14}
M_3	9.28×10^{-2}	15.2	45.6	5.03×10^{-14}	M_3	1.88×10^{-1}	16	48	9.48×10^{-14}
M_4	8.77×10^{-2}	13.7	41.1	4.23×10^{-14}	M_4	1.45×10^{-1}	14	42	6.01×10^{-14}
M_5	2.80×10^{-1}	10.3	48.1	9.52×10^{-14}	M_5	4.85×10^{-1}	10.8	51.6	1.18×10^{-13}
M_6	1.41×10^{-1}	6.9	41.4	5.46×10^{-14}	M_6	3.13×10^{-1}	7.2	43.2	8.35×10^{-14}
M_7	1.26×10^{-1}	6.1	42.7	2.72×10^{-13}	M_7	2.76×10^{-1}	6.2	43.4	3.48×10^{-13}
M_8	1.29×10^{-1}	6	42	4.4×10^{-13}	M_8	2.50×10^{-1}	6.2	43.4	4.41×10^{-13}
M_9	1.31×10^{-1}	6.6	46.2	1.47×10^{-13}	M_9	3.00×10^{-1}	7	49	1.32×10^{-13}
M_{10}	8.05×10^{-2}	13.9	41.7	4.30×10^{-14}	M_{10}	1.52×10^{-1}	14.2	42.6	7.02×10^{-14}
M_{11}	1.10×10^{-1}	14.2	56.8	4.19×10^{-14}	M_{11}	2.05×10^{-1}	15	60	7.46×10^{-14}
M_{12}	7.51×10^{-2}	9.7	38.8	4.18×10^{-14}	M_{12}	1.49×10^{-1}	10	40	7.13×10^{-14}
M_{13}	7.49×10^{-2}	7.6	38	6.43×10^{-14}	M_{13}	1.34×10^{-1}	7.8	39	8.23×10^{-14}
		$n=180$					$n=200$		
Method	CPU	Iter	TMM	Error	Method	CPU	Iter	TMM	Error
M_1	2.79×10^{-1}	25.5	51	7.32×10^{-14}	M_1	3.77×10^{-1}	26.4	52.8	5.66×10^{-14}
M_2	4.31×10^{-1}	11	44	6.02×10^{-14}	M_2	5.14×10^{-1}	11.3	45.2	5.61×10^{-14}
M_3	2.79×10^{-1}	15.9	47.7	6.6×10^{-14}	M_3	3.56×10^{-1}	16.7	50.1	5.35×10^{-14}
M_4	2.68×10^{-1}	13.8	41.4	5.86×10^{-14}	M_4	3.24×10^{-1}	14.5	43.5	6.17×10^{-14}
M_5	6.93×10^{-1}	11	53	1.01×10^{-13}	M_5	8.32×10^{-1}	11	53	1.35×10^{-13}
M_6	3.58×10^{-1}	7	42	8.14×10^{-14}	M_6	4.29×10^{-1}	7.4	44.4	7.12×10^{-14}
M_7	3.46×10^{-1}	6.2	43.4	2.55×10^{-13}	M_7	4.06×10^{-1}	6.4	44.8	3.32×10^{-13}
M_8	3.24×10^{-1}	6.1	42.7	4.20×10^{-13}	M_8	3.99×10^{-1}	6.4	44.8	4.65×10^{-13}
M_9	3.73×10^{-1}	7	49	1.45×10^{-13}	M_9	4.40×10^{-1}	7.1	49.7	1.65×10^{-13}
M_{10}	2.31×10^{-1}	13.9	41.7	7.06×10^{-14}	M_{10}	3.11×10^{-1}	14.3	42.9	5.12×10^{-14}
M_{11}	3.35×10^{-1}	15	60	6.92×10^{-14}	M_{11}	4.49×10^{-1}	15.7	62.8	5.65×10^{-14}
M_{12}	2.17×10^{-1}	9.9	39.6	5.40×10^{-14}	M_{12}	2.95×10^{-1}	10	40	5.47×10^{-14}
M_{13}	2.07×10^{-1}	7.7	38.5	7.58×10^{-14}	M_{13}	2.95×10^{-1}	8.1	40.5	1.01×10^{-13}

Based on Table 1, Figure 2, and Figure 3, we note that the methods (23) (M_{12}) and (26) (M_{13}) have lower CPU and TMM than other methods with acceptable accuracy.

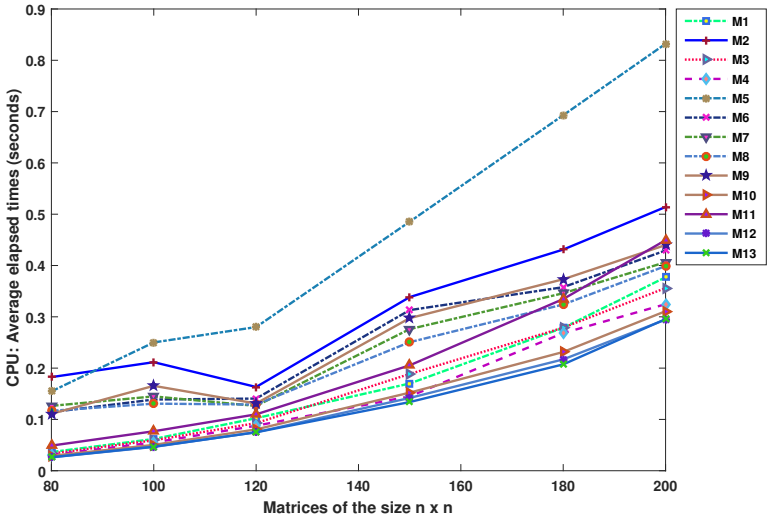

Figure 2: The comparison of CPU of methods for various matrices with $n = 80, 100, 120, 150, 180, 200$ in Example 1

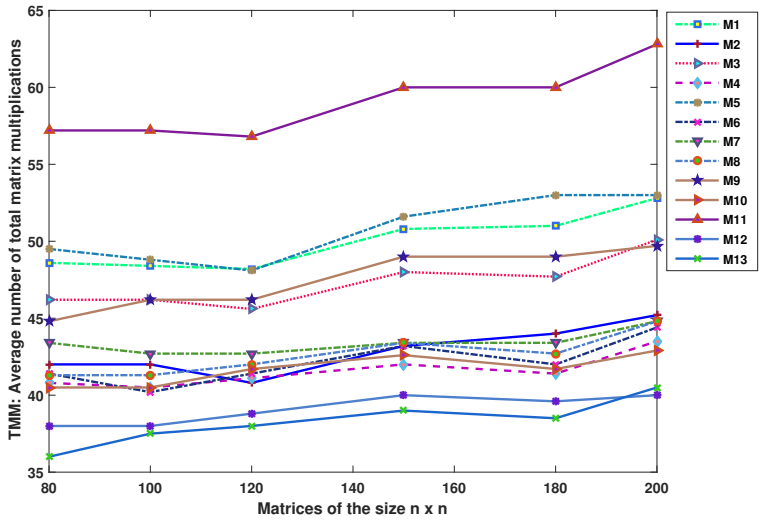

Figure 3: The comparison of TMM of methods for various matrices with $n = 80, 100, 120, 150, 180, 200$ in Example 1

For the next example, we define the residual value of polar decomposition as follows:

$$Res = \frac{\|A - U_{k+1}H_{k+1}\|_F}{\|A\|_F},$$

in which $H_{k+1} = \frac{1}{2}(A^*U_{k+1} + U_{k+1}^*A)$. We report the average Res of methods for ill-conditional matrices

Table 2: The comparison of methods for matrices with dimension 80×80 in Example 2

	Matrix: A_1	$K(A_1) = 8.13 \times 10^{22}$				Matrix: A_2	$K(A_2) = 2.89 \times 10^{18}$			
Method	CPU	Iter	TMM	Res	Method	CPU	Iter	TMM	Res
M_1	8.03×10^{-2}	48.5	97	1.69×10^{-15}	M_1	1.59×10^{-1}	114	228	1.45×10^{-15}
M_2	1.12×10^{-1}	19.3	77.2	1.56×10^{-15}	M_2	2.91×10^{-1}	45	180	2.32×10^{-15}
M_3	7.32×10^{-2}	29.4	88.2	1.37×10^{-15}	M_3	1.54×10^{-1}	67	201	1.29×10^{-15}
M_4	5.77×10^{-2}	23.3	69.9	1.15×10^{-15}	M_4	1.12×10^{-1}	49	147	2.10×10^{-15}
M_5	2.88×10^{-1}	33.2	90.2	8.22×10^{-15}	M_5	2.24×10^{-1}	30	186	2.41×10^{-14}
M_6	9.18×10^{-2}	11.8	70.8	2.73×10^{-15}	M_6	1.56×10^{-1}	25	150	1.66×10^{-14}
M_7	8.58×10^{-2}	10.4	72.8	2.00×10^{-14}	M_7	1.49×10^{-1}	22	154	6.21×10^{-13}
M_8	8.23×10^{-2}	10.2	71.4	3.43×10^{-14}	M_8	1.41×10^{-1}	21	147	9.29×10^{-13}
M_9	9.24×10^{-2}	11.5	80.5	7.29×10^{-15}	M_9	1.65×10^{-1}	24	168	7.06×10^{-14}
M_{10}	5.85×10^{-2}	23	69	1.14×10^{-15}	M_{10}	1.13×10^{-1}	49	147	1.82×10^{-15}
M_{11}	9.29×10^{-2}	27.9	111.6	2.26×10^{-15}	M_{11}	1.89×10^{-1}	66	264	1.73×10^{-15}
M_{12}	5.39×10^{-2}	16.1	64.4	1.17×10^{-15}	M_{12}	1.11×10^{-1}	36	144	2.75×10^{-15}
M_{13}	5.40×10^{-2}	12.8	64	4.21×10^{-15}	M_{13}	9.73×10^{-2}	27	135	2.53×10^{-14}
	Matrix: A_3	$K(A_3) = 1.04 \times 10^{47}$				Matrix: A_4	$K(A_4) = 5.51 \times 10^{19}$			
Method	CPU	Iter	TMM	Res	Method	CPU	Iter	TMM	Res
M_1	–	–	–	–	M_1	1.74×10^{-1}	123	246	2.06×10^{-15}
M_2	6.37×10^{-1}	123	492	7.34×10^{-16}	M_2	2.46×10^{-1}	47	188	5.30×10^{-15}
M_3	4.11×10^{-1}	195	585	3.89×10^{-16}	M_3	1.60×10^{-1}	70	210	1.87×10^{-15}
M_4	2.78×10^{-1}	135	405	3.80×10^{-16}	M_4	1.14×10^{-1}	51	153	2.99×10^{-15}
M_5	4.78×10^{-1}	73	487	8.81×10^{-15}	M_5	2.38×10^{-1}	30	186	3.43×10^{-14}
M_6	4.42×10^{-1}	70	420	7.06×10^{-15}	M_6	1.88×10^{-1}	27	162	3.40×10^{-14}
M_7	4.09×10^{-1}	62	434	8.98×10^{-14}	M_7	1.70×10^{-1}	23	161	7.10×10^{-13}
M_8	3.98×10^{-1}	60	420	1.28×10^{-13}	M_8	1.57×10^{-1}	22	154	2.77×10^{-12}
M_9	4.45×10^{-1}	69	483	2.68×10^{-14}	M_9	1.74×10^{-1}	26	182	1.01×10^{-13}
M_{10}	2.78×10^{-1}	131	393	2.82×10^{-16}	M_{10}	1.16×10^{-1}	53	159	2.26×10^{-15}
M_{11}	5.11×10^{-1}	181	724	3.75×10^{-16}	M_{11}	1.93×10^{-1}	67	268	3.47×10^{-15}
M_{12}	2.73×10^{-1}	97	388	4.92×10^{-16}	M_{12}	1.04×10^{-1}	37	148	3.96×10^{-15}
M_{13}	2.63×10^{-1}	72	360	2.14×10^{-15}	M_{13}	9.96×10^{-2}	28	140	2.50×10^{-14}

	Matrix: A_5	$K(A_5) = 1.70 \times 10^{20}$		
Method	CPU	Iter	TMM	Res
M_1	1.72×10^{-1}	124.7	249.4	1.90×10^{-15}
M_2	2.43×10^{-1}	46.9	187.6	3.57×10^{-15}
M_3	1.51×10^{-1}	75	225	1.69×10^{-15}
M_4	1.13×10^{-1}	54	162	1.79×10^{-15}
M_5	2.45×10^{-1}	31	193	2.26×10^{-14}
M_6	1.85×10^{-1}	27.2	163.2	2.35×10^{-14}
M_7	1.70×10^{-1}	23.6	165.2	3.93×10^{-13}
M_8	1.53×10^{-1}	22.8	159.6	1.02×10^{-12}
M_9	1.74×10^{-1}	26.7	186.9	7.14×10^{-14}
M_{10}	1.15×10^{-1}	53.1	159.3	1.57×10^{-15}
M_{11}	2.02×10^{-1}	70.8	283.2	2.95×10^{-15}
M_{12}	1.10×10^{-1}	38.7	154.8	2.21×10^{-15}
M_{13}	1.06×10^{-1}	29.4	147	1.35×10^{-14}

used in Example 2.

Example 2. Consider the ill-conditioned matrices A_i, $i = 1, \ldots, 5$, with sizes 80×80 and 100×100. For each dimension, ten random matrices are generated. The average condition number for each matrix A_i is computed as $K(A_i)$. The results include the average *Res* are reported in Tables 2 and 3. Some methods that do not reach the solution with iterations less than the maximum iteration (equal to 200), are reported by "–" in tables.

Consider the following ill-conditioned matrices for the second example with their MATLAB commands. Assume a random vector $v = rand(n, 1)$, then

A_1: Hankel matrix ($A_1 = hankel(v)$),

A_2: Hilbert matrix ($A_2 = hilb(n)$),

A_3: Pascal matrix ($A_3 = pascal(n)$),

Table 3: The comparison of methods for matrices with dimension 100×100 in Example 2

	Matrix: A_1	$K(A_1) = 6.31 \times 10^{20}$		
Method	CPU	Iter	TMM	Res
M_1	1.46×10^{-1}	56.9	113.8	2.19×10^{-15}
M_2	1.74×10^{-1}	22.5	90	1.71×10^{-15}
M_3	1.26×10^{-1}	34.4	103.2	1.55×10^{-15}
M_4	1.10×10^{-1}	29.6	88.8	1.47×10^{-15}
M_5	4.70×10^{-1}	40.4	93.2	2.90×10^{-4}
M_6	1.43×10^{-1}	13.4	80.4	3.08×10^{-15}
M_7	1.46×10^{-1}	12.1	84.7	2.09×10^{-14}
M_8	1.30×10^{-1}	11.5	80.5	4.08×10^{-13}
M_9	1.46×10^{-1}	13.1	91.7	7.40×10^{-15}
M_{10}	1.15×10^{-1}	28.9	86.7	1.43×10^{-15}
M_{11}	1.66×10^{-1}	32.3	129.2	2.54×10^{-15}
M_{12}	9.80×10^{-2}	19.1	76.4	1.32×10^{-15}
M_{13}	9.28×10^{-2}	14.3	71.5	4.86×10^{-15}

	Matrix: A_2	$K(A_2) = 3.47 \times 10^{19}$		
Method	CPU	Iter	TMM	Res
M_1	2.96×10^{-1}	114	228	1.35×10^{-15}
M_2	3.50×10^{-1}	44	176	2.80×10^{-15}
M_3	2.87×10^{-1}	70	210	2.25×10^{-15}
M_4	1.95×10^{-1}	49	147	1.94×10^{-15}
M_5	3.72×10^{-1}	30	186	2.62×10^{-14}
M_6	2.59×10^{-1}	25	150	1.96×10^{-14}
M_7	2.36×10^{-1}	21	147	3.84×10^{-13}
M_8	2.30×10^{-1}	21	147	1.34×10^{-12}
M_9	2.73×10^{-1}	25	175	5.23×10^{-14}
M_{10}	2.14×10^{-1}	55	165	2.21×10^{-15}
M_{11}	3.43×10^{-1}	66	264	2.85×10^{-15}
M_{12}	1.84×10^{-1}	36	144	3.21×10^{-15}
M_{13}	1.86×10^{-1}	28	140	2.18×10^{-14}

	Matrix: A_3	$K(A_3) = 4.06 \times 10^{59}$		
Method	CPU	Iter	TMM	Res
M_1	–	–	–	–
M_2	1.19	147	588	5.87×10^{-16}
M_3	–	–	–	–
M_4	6.20×10^{-1}	163	489	2.73×10^{-16}
M_5	1.03	89	599	6.75×10^{-15}
M_6	8.80×10^{-1}	84	504	6.15×10^{-15}
M_7	8.28×10^{-1}	74	518	5.69×10^{-14}
M_8	8.06×10^{-1}	72	504	6.42×10^{-14}
M_9	8.98×10^{-1}	83	581	2.05×10^{-14}
M_{10}	6.10×10^{-1}	159	477	6.18×10^{-16}
M_{11}	–	–	–	–
M_{12}	5.95×10^{-1}	117	468	4.67×10^{-16}
M_{13}	5.75×10^{-1}	87	435	2.03×10^{-15}

	Matrix: A_4	$K(A_4) = 2.28 \times 10^{19}$		
Method	CPU	Iter	TMM	Res
M_1	3.13×10^{-1}	119	238	2.47×10^{-15}
M_2	3.52×10^{-1}	45	180	5.38×10^{-15}
M_3	2.83×10^{-1}	71	213	2.66×10^{-15}
M_4	2.26×10^{-1}	51	153	3.04×10^{-15}
M_5	3.90×10^{-1}	30	186	4.52×10^{-14}
M_6	2.73×10^{-1}	26	156	3.54×10^{-14}
M_7	2.55×10^{-1}	22	154	8.48×10^{-13}
M_8	2.47×10^{-1}	22	154	2.88×10^{-12}
M_9	2.89×10^{-1}	25	175	1.07×10^{-13}
M_{10}	2.08×10^{-1}	51	153	2.64×10^{-15}
M_{11}	3.68×10^{-1}	70	280	4.70×10^{-15}
M_{12}	2.02×10^{-1}	37	148	4.83×10^{-15}
M_{13}	1.88×10^{-1}	28	140	2.48×10^{-14}

	Matrix: A_5	$K(A_5) = 3.46 \times 10^{19}$		
Method	CPU	Iter	TMM	Res
M_1	3.18×10^{-1}	124.7	249.4	2.18×10^{-15}
M_2	3.63×10^{-1}	47.3	189.2	4.38×10^{-15}
M_3	2.81×10^{-1}	74.9	224.7	1.94×10^{-15}
M_4	2.07×10^{-1}	54.3	162.9	1.98×10^{-15}
M_5	4.13×10^{-1}	31.5	196.5	2.61×10^{-14}
M_6	2.96×10^{-1}	27.5	165	2.79×10^{-14}
M_7	2.67×10^{-1}	23.6	165.2	4.25×10^{-13}
M_8	2.57×10^{-1}	23	161	1.02×10^{-12}
M_9	2.94×10^{-1}	26.9	188	8.16×10^{-14}
M_{10}	2.09×10^{-1}	53.5	160.5	1.94×10^{-15}
M_{11}	3.67×10^{-1}	71.2	284.8	3.29×10^{-15}
M_{12}	2.02×10^{-1}	39.6	158.4	2.64×10^{-15}
M_{13}	2.01×10^{-1}	30.3	151.5	1.58×10^{-14}

A_4**:** Lotkin matrix ($A_4 = gallery('lotkin', n)$),

A_5**:** Vandermonde matrix ($A_5 = vander(v)$).

In Table 2 and 3, superior performance of methods (23) (M_{12}) and (26) (M_{13}) can be concluded. Moreover, reported values of *Res* show that this methods find prime approximates for the polar decomposition of ill-conditioned matrices rapidly.

Upon reviewing the results of the examples, it is evident that equations (23) (M_{12}) and (26) (M_{13}) exhibit a lower number of matrix multiplications and a shorter elapsed execution time compared to other methods. Consequently, their computational costs are more economical than alternative methods, leading to a faster computation of solutions.

5 Conclusion

In this paper, we present a class of iterative methods for determining the matrix U in the polar decomposition $A = UH$, exploring its convergence characteristics. By proposing a comprehensive strategy,

we introduce several classes of iterative methods, each exhibiting distinct orders of convergence for polar decomposition. Notably, higher-order methods are derived within this framework. Importantly, these methods circumvent matrix inversion, relying solely on matrix multiplication. Application of these methods to diverse matrices reveals that the convergent methods of order four and eight, denoted as (23) and (26), respectively, require only four and five matrix multiplications. Strikingly, these methods outperform others in terms of elapsed execution time and the total number of matrix multiplications. Additionally, as they abstain from matrix inversion, their computational costs are substantially lower, rendering them particularly suitable for ill-conditioned matrices.

Competing interests

The authors declare that they have no conflict of interests.

Funding

Not applicable.

Data availability

Not applicable.

Acknowledgements

We sincerely thank the anonymous reviewers for their valuable comments.

References

[1] L. Autonne, *Sur les groupes lineaires, reels et orthogonaux*, Bull. Sot. Math. France **30** (1902) 121-134.

[2] A. Cordero, J.R. Torregrosa, *A sixth-order iterative method for approximating the polar decomposition of an arbitrary matrix*, J. Comput. Appl. Math. **318** (2017) 591-598.

[3] H. Esmaeili, *A class of iterative methods for computing polar decomposition*, Int. J. Comput. Math. **88** (2011) 207-220.

[4] W. Gander, *Algorithms for polar decomposition*, SIAM J. Sci. Stat. Comput. **11** (1990) 1102-1115.

[5] G. Golub, C. Van Loan, *Matrix Computations, fourth edition*, Baltimore: The Johns Hopkins University Press, 2013.

[6] J.C. Gower, G.B. Dijksterhuis, *Procrustes problems*, Oxford University Press, 2004.

[7] N.J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, 2008.

[8] N.J. Higham, *Computing the polar decomposition - with applications*, SIAM J. Matrix Anal. Appl. **7** (1986) 1160-1174.

[9] N.J. Higham, *Matrix nearness problems and applications*, in Applications of Matrix Theory, M. J. C. Gover, S. Barnett, eds., Oxford University Press, (1989) 1-27.

[10] N.J. Higham, *Computing a nearest symmetric positive semidefinite matrix*, Linear Algebra Appl. **103** (1988) 103-118.

[11] N.J. Higham, D.S. Mackey, N. Mackey, F. Tisseur, *Computing the polar decomposition and the matrix sign decomposition in matrix groups*, SIAM J. Matrix Anal. Appl. **25** (2004) 1178-1192.

[12] F. Khaksar Haghani, F. Soleymani, *On a fourth-order matrix method for computing polar decomposition*, Comp. Appl. Math. **34** (2015) 389-399.

[13] F. Kiyoumarsi, *Some new high-order computational methods for polar decomposition of complex matrices*, Iran. J. Sci. Technol. Trans. A: Sci. **42** (2018) 2293-2299.

[14] Z. Kovarik, *Some iterative methods for improving orthogonality*, SIAM J. Numer. Anal. **7** (1970) 386-389.

[15] D. Petcu, C. Popa, *A new version of Kovarik's approximate orthogonalization algorithm without matrix inversion*, Int. J. Comput. Math. **82** (2005) 1235-1246.

[16] M.D. Petković, *Generalized Schultz iterative methods for the computation of outer inverses*, Comput. Math. Appl. **67(10)** (2014) 1837-1847.

[17] M.D. Petković, M.A. Krstić, K.P. Rajković, *Rapid generalized Schultz iterative methods for the computation of outer inverses*, J. Comput. Appl. Math. **344** (2018) 572-584.

[18] S. Sheikhi, H. Esmaeili, *A new iterative method to find polar decomposition*, Commun. Appl. Math. Comput. **7** (2025) 2243-2256.

[19] F. Soleymani, F. Khaksar Haghani, S. Shateyi, *Several numerical methods for computing unitary polar factor of a matrix*, Adv. Differ. Equ. **2016(4)** (2016).

[20] F. Soleymani, P.S. Stanimirović, I. Stojanović. *A novel iterative method for polar decomposition and matrix sign function*, Discrete Dyn. Nat. Soc. **2015(1)** (2015) 649423.