Transactions of NAMP 24, (2026) 53-70

Transactions of
The Nigerian Association of
Mathematical Physics
Journal homepage: https://nampjournals.org.ng

COMPUTING HIGH SPEED CONVERGENT ITERATIVE METHODS FOR THE POLAR DECOMPOSITION.

Stephen Ehidiamhen Uwamusi[1] and Andrew Esaborlupia Uwamusi[2]

[1]*Department of Mathematics, Faculty of Physical Sciences, University of Benin, Benin City,Edo State, Nigeria.*

[2]*Department of Mathematics, College of Science, Federal University of Petroleum Resources, Effurun, Delta State, Nigeria.*

ARTICLE INFO

Article history:
Received xxxxx
Revised xxxxx
Accepted xxxxx
Available online xxxxx

Keywords:
Iteration of polar coordinates, Polar decomposition, Numerical methods.

ABSTRACT

This paper presents high speed convergent iterative methods for accelerating computation process in the polar decomposition of a matrix. As a first demonstration in our illustration, we split a complex number into real and imaginary components and simultaneously iterated to convergence using Halley's third order iterative method for polynomial equation. Utilizing the Newton and Halley's methods for computing polar decomposition of a matrix, we further accelerated convergence using matrix condition number and determinant of a matrix with a fast LU Factorization solver for matrix inversion. The described methods are backward stable.As a further insight of our study is the computation of zonal polynomials for these matrice., Numerical examples are demonstrated with these methods.

1. INTRODUCTION

The quest to compute the polar decomposition of a matrix by some fast iterative methods has been of much interest to Numerical analysts in the academic community for some time now. Even much more important is the need to modify these methods to accommodate a faster convergent technique leading to desired results in a reasonably clock time. Numerous applications of computing polar decomposition of a matrix range over aerospace computation, machine learning and other applied engineering sciences. This paper aims at providing fast iterative solvers in computing polar decomposition of a matrix provided that the matrix is diagonalizable [2].

1.1 Literature Review/ Preliminaries

In line with [4], a polar decomposition of a matrix (full rank) of $A \in \mathbb{C}^{n\times m}$ where, $n \geq m$, and an orthonormal matrix $U \in \mathbb{C}^{n\times n}$ with a unique Hermittian positive semi definite matrix $P \in \mathbb{C}^{m\times m}$ for which $A = UP, \quad U^*U = I_m$, was studied.

*Corresponding author: STEPHEN EHIDIAMHEN UWAMUSI
E-mail address: stephen.uwamusi@uniben.edu
https://doi.org/10.60787/tnamp.v24.668

The matrix P is a positive definite for which U is uniquely determined. The study of polar decomposition of a matrix has a wide ranging applications in the
measurement of a distance in the Frobenius norm, e.g., the nearest unitary matrix to $A^{m\times m}$ is the unitary factor U in the polar decomposition of A and whereas the farthest unitary matrix from A is minus this unitary matrix. Particularly in this direction, the polar decomposition works closely

with the singular value decomposition (SVD) which is defined by the equation

$$A = U\sum V^T \quad . \qquad (1.1)$$

In equation (1.1) the matrices U, V are orthonormal consisting of vectors corresponding to their singular values. This means that matrices U, V are unitary matrices consisting of eigenvectors so that $U \in \mathbb{C}^{n\times n}$, $V \in \mathbb{C}^{m\times m}$ where $\sum = diag(\sigma_1, \sigma_2, ..., \sigma_m)$, and $\sigma_1 > \sigma_2 > ... > \sigma_m \geq 0$ are arranged in their order of magnitudes.

Newton and Halley's methods [13, 14] are by far the fastest ways of computing the polar decomposition of a matrix and extends to a rectangular matrix as well. Inverse function theorem plays a major role in the convergence analysis of Newton and Halley's iterations in which the concept of homeomorphism is a useful tool. Previous work related to this work is the extraction of the P-th root of a matrix studied in [13].

Following [13,14] we introduce a prototype of Newton's method for computing square root of a diagonalizable matrix in the form:

$$X_{k+1} = \frac{1}{2}\left(X_k + X_k^{-T}\right), X_0 = A \qquad \text{(k=0,1,2,…)} \qquad (1.1)$$

Method of equation (1.1) may converge slowly when eigenvalues clustered around $\pm i$ where, $i = \sqrt{-1}$. The Pade's family of methods of order three [1,3,4,9,10] in which Halley's method belongs has advantage over Newton's method under this condition in overcoming this defect. It is given by the equation:

$$X_{k+1} = X_k\left(3I + X_k^T X_k\right)\left(I + 3X_k^T X_k\right)^{-1}, X_0 = A \quad \text{(k=0,1,2,…)} \qquad (1.2)$$

In the sense of [6], as eigenvalues cluster around $\pm i$, method of equation (1.1) leads to only computing the polar decomposition for the matrix $\frac{1}{2}\left(A + A^*\right)$ where A^* is a complex conjugate of the matrix A.

We speed up computation processes of these Newton's and Halley's methods [8] making use of basic characteristic techniques such as matrix condition number and determinant of a matrix. This takes into considerations coefficients of efficiencies of these methods. We discuss parameters scaling equivalents of Newton's and Halley's methods [1,3,4,,9,10,11, 12,15] as follows.

Letting

$$\varsigma_0 = \frac{1}{\sqrt{ab}}, \varsigma_1 = \sqrt{2\frac{\sqrt{ab}}{(a+b)}},$$

$a = \|A\|_2$, $b = \|A^{-1}\|_2$ and $\rho(\varsigma) = \frac{\left(\varsigma + \varsigma^{-1}\right)}{2}$, the scaled Newton method is expressed as:

$$X_{k+1} = \frac{1}{2}\left(\varsigma_k X_k + \left(\varsigma_k X_k\right)^{-T}\right), X_0 = A \; . \qquad (1.3)$$

Corresponding to equation (1.3) is inverse Newton's method for polar decomposition written as

$$Y_{k+1} = 2Y_k\left(I + Y_k^T Y_k\right)^{-1}, Y_0 = A \ , \qquad (1.4)$$

where $Y_k = X_k^{-T}$ .If we affix a weight to Y_k in the form $Y_k = \eta_k Y_{kk}$, method of equation (1.4) is rewritten in the form:

$$Y_{k+1} = 2\eta_k Y_k\left(I + \eta_k Y_k^T Y_k\right)^{-1} , Y_0 = A \qquad (1.5)$$

where,

$\eta_0 = \frac{1}{\sqrt{ab}}, \eta_1 = \sqrt{\frac{a+b}{2\sqrt{ab}}}$, $\eta_k = \sqrt{\rho(\eta_{k-1})}$, $k = 2,3,...$ Thus $Y_k \to U^{-T} = U$ as $k \to \infty$.

The modified Halley's method on the other hand is rewritten in the form:

$$X_{k+1} = X_k\left(a_k I + b_k X_k^T X_k\right)\left(I + c_k X_k^T X_K\right)^{-1}, \qquad X_0 = \frac{A}{a} \ , \ a = \|A\|_2 . \qquad (1.6)$$

Following we choose $a_K = \|A\|_2, b_k = \frac{1}{\|A\|_2}, c_k = a_k + b_k - 1$ applying *SVD* on equation (1.6) we have that:

$$X_{k+1} = U\sum_k\left(a_k I + b_k \sum_k^2\right)\left(I + c_k \sum_k^2\right)^{-1} V^T \ , (k = 0,1,2,...) . \qquad (1.7)$$

The behavior of singular values $\sigma_i(X_{k+1})$ is expressed in the form:

$$\sigma_i(X_{k+1}) = g_k(\sigma_i(X_k)) \ , \qquad (1.8)$$

$$g_k(x) = x\left(\frac{a_k + b_k}{1 + c_k x^2}\right) \ ; \ \ (0 < g_k(x) < 1) \ . \qquad (1.9)$$

The following facts on global minimizer of $g_k(x)$ is expressed by setting

$$a_k = h(\ell_k), b_k = \frac{(a_k - 1)^2}{4} , \qquad (1.10)$$

$$h(\ell) = \sqrt{1+d} + \frac{1}{2}\sqrt{8 - 4d + \frac{8(2-\ell^2)}{\ell^2\sqrt{1+d}}} , \qquad (1.11)$$

$$d = \sqrt[3]{\frac{4(1-\ell^2)}{\ell^4}} . \qquad (1.12)$$

$$\ell_0 = \frac{b}{a}, \ \ell_k = \frac{\ell_{k-1}\left(a_{k-1} + b_{k-1}\ell_{k-1}^2\right)}{1 + c_{k-1}\ell_{k-1}^2}, \quad (k = 1,2,...) \qquad (1.13)$$

To compute ℓ_k in the recurrence relation follows equation in the lower bound for the smallest singular value in the phase space matrices:

$$\ell_k = \frac{\ell_{k-1}\left(a_{k-1} + b_{k-1}\ell_{k-1}^2\right)}{\left(1 + c_{k-1}\ell_{k-1}^2\right)} \ , \ \ (\ell_0 \le \sigma_{\min}(X_0)), \qquad (1.14)$$

It should be noted that $K_2(A) \le u^{-1}$, for $u = 2^{-53} \approx 1.1 \times 10^{-16}$ in the unit round off for *IEEE* double precision arithmetic for any matrix.

Inverse free iteration process leads to further introduction of the *QR* decomposition for Halley's method of equation (1.6), we have the expression:

$$\begin{bmatrix} c_k X_k \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R ,$$

$$X_{k+1} = \frac{b_k}{c_k} X_k + \frac{1}{\sqrt{c_k}} \left(a_k - \frac{b_k}{c_k} \right) Q_1 Q_2 , \; k \geq 0 . \qquad (1.15)$$

wherefrom,

$$Q_1 Q_2^T = c_k X_k \left(I + c_k^2 X_k^2 X_k \right)^{-1} \quad , \quad X_0 = \frac{A}{\alpha}, \; \alpha = \|A\|_2 . \qquad (1.16)$$

The c_k is polynomial bounded in the form

$$c_k = a_k + \frac{(a_k - 1)}{4 - 1} = \frac{1}{4}\left((a_k - 1) + 4(a_k - 1)\right) = \frac{1}{4}(a_k - 1)(a_k + 3).$$

Therefore $\frac{b_k}{c_k} = \frac{a_k - 1}{a_k + 3}$. Hence the inequality $\frac{1}{3} \leq \frac{b_k}{c_k} \leq 1$ holds in the sense of [10].

2 Iteration for Complex number in the form polar coordinates with Halley's method.

The least order class in the Pade iteration family which includes Halley's iteration in this group for finding the matrix equation $X^2 - A = 0$ is given by the equation

$$X_{k+1} = X_k \left(3I + X_k^2 \right) \left(I + 3X_k^2 \right)^{-1}, \; X_0 = A. \qquad (2.1)$$

The following theorem holds true for the Pade iteration method in equation (2.1).

Theorem 2.1, [9]. Let A be a unitary matrix whose spectrum is contained in S_θ for some $\theta \in \left(0, \frac{\pi}{2} \right)$

For any number $n \in N$, the iteration

$$X_{k+1} = r_{2n+1}(X_k, \theta_k), \quad X_0 = A \quad , \qquad (2.2)$$

$$\theta_{k+1} = \left| \arg r_{2n+1}\left(e^{i\theta_k}, \theta_k \right) \right|, \theta_0 = 0 \quad , \qquad (2.3)$$

converges to $sign(A)$ with order of convergence $2n+1$.

We will be using equation (2.2) as informed decision leading to demonstration with the following example in computing complex number $z = re^{i\theta}$ expressed in the form polar coordinates. We make use of the cubically convergent Halley's iteration for solving polynomial equation $f(x) = 0$ .By realizing that a polar decomposition is similar to the complex number factorization $z = e^{i \arg(z)} |z|$, then we have the following as problem 2.1.

Problem 2.1.

Consider computing the Complex number

$$z = re^{i\theta} \quad , \qquad (2.4)$$

which comes from the representation of the complex number $z = u + iv$, where $i = \sqrt{-1}$.

Here, it is necessary to compute the modulus of z written as $|z| = r$ and, argument $\theta = \arctan \frac{y}{x}$ for $x \neq 0$. We are required to calculate the error in the estimation of the magnitude r and the argument θ at iteration n. Denoting respectively, the true values fo the magnitude and argument as r* and θ*. Then, the numerical tool box we are using could be any of the Newton or Halley's methods. But here in our calculations, using Halley's method for solving polynomial equation $f(x) = 0$ is adopted. One needs to show that the errors in the estimation of the magnitude and argument of the complex number decrease at a specific rate as the number of iteration increases.

Denoting the error in estimating the magnitude at iteration n as $e(r_n) = |r_n - r^*|$, where the r_n is estimate of the magnitude at the iteration n; r^* is the true magnitude. Similarly, we denote the

error in the estimation of the argument at n -th iteration as $e(\theta_n) = \left|\theta_n - \theta^*\right|$, then θ_n is the corresponding estimated magnitude of the argument at the iteration n.

Coupling these together, we then express the magnitude estimation error occurring in the radius (amplitude) at the $n+1$-th step in the form

$$e(r_{n+1}) = \left| r_{n+1} - r^* \right| \ , \tag{2.5}$$

and substituting the update equations for r_{n+1} as $(u_{n+1}^2 + v_{n+1}^2)^{\frac{1}{2}}$ and its corresponding true error as r*, we then have that:

$$e(r_{n+1}) = \left| (u_{n+1}^2 + v_{n+1}^2)^{\frac{1}{2}} - r^* \right| \ . \tag{2.6}$$

By setting

$$u_{n+1} = r_{n+1}\cos\theta_{n+1} \ , \ v_{n+1} = r_{n+1}\sin\theta_{n+1}, \tag{2.7}$$

the error at the nth step iteration is then written as

$$e(r_{n+1}) = \left| (r_{n+1}^2\cos_{n+1}^2\theta + r_{n+1}^2\sin_{n+1}^2\theta)^{\frac{1}{2}} - r^* \right|. \tag{2.8}$$

Using the trigonometric identity that $\cos^2(\theta) + \sin^2(\theta) = 1$, we simplified the equation (2.8) to read:

$$e(r_{n+1}) = \left| \left\{ \left(r_{n+1}^2\right)\left(\cos_{n+1}^2\theta + \sin_{n+1}^2\theta\right) \right\}^{\frac{1}{2}} - r* \right| \ . \tag{2.9}$$

This means that $e(r_{n+1}) = \left|r_{n+1} - r*\right|$ wherefrom, it holds that argument estimation error is

$e(\theta_{n+1}) = \left|\theta_{n+1} - \theta^*\right|$.

The updating formula for θ(n+1) and θ* is

$e(\theta_{n+1}) = \left|\arctan^2(v_{n+1}, u_{n+1}) - \theta^*\right|$.

The trigonometric identities $\arctan(v,u) = \arctan(v/u)$ and $\arctan(1/u) = arc\cot(u)$ are well known in the class of trigonometric problems. Hence we have thus simplified the equation to read:

$$e(\theta_{n+1}) = \left| \arctan\left(\frac{v_{n+1}}{u_{n+1}}\right) - \theta^* \right|.$$

Now, coupling together the relationship between the error at iteration n and n+1, triangular inequalities applied indicated that:

$$e(r_{n+1}) = \left| r_{n+1} - r^* \right| = \left| (u_{n+1}^2 + v_{n+1}^2)^{\frac{1}{2}} - r^* \right| \le \left| \ ((u_{n+1} - u^*)^2 + (v_{n+1} - v^*)^2)^{\frac{1}{2}} \ \right|, \text{(Triangle inequality),}$$

and

$$e(\theta_{n+1}) = \left|\arctan\left(\frac{v_{n+1}}{u_{n+1}}\right) - \theta^*\right| \le \left|\arctan\left(\frac{v_{n+1}}{u_{n+1}}\right) - u^*\right| \text{ (by Triangle inequality).}$$

We thus have succeeded in splitting the complex number $z = re^{i\theta}$ in the form of two simultaneous iterations into real space. We shall proceed further in this direction by utilizing the Halley's iteration of third order method as earlier mentioned at the beginning.

The cubically convergent iterative formula of Halley's method for finding roots of a polynomial equation $f(z) = 0$ is expressed in the form equation (2.7) in the form:

$$u_{n+1} = u_n - \frac{f(u_n) f'(u_n))}{f'(u_n)^2 - \frac{1}{2} f(u_n) f''(u_n)}; \qquad (2.10)$$

$$v_{n+1} = v_n - \frac{f(v_n) f'(v_n)}{f'(v_n)^2 - \frac{1}{2} f(v_n) f''(v_n)}. \qquad (2.11)$$

This is equivalent to writing that $f(r_n, \theta_n),\ n = 0,1,2,...$

$$r_{n+1} = (u_{n+1}^2 + v_{n+1}^2)^{\frac{1}{2}}, \qquad (2.12)$$

$$\theta_{n+1} = \arctan(v_{n+1}, u_{n+1}). \qquad (2.13$$

We display the iterative values for the radius r and argument θ, calculated using a cubically convergent Halley's method as shown in Table 1. It has an order three of convergence.

Table1: Showing results for computing with Halley's method the complex number $z = re^{i\theta}$.

Iteration n	0	1	2	3	4	5	6	7	8	9	10
Radius r_n	1	3.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
Argument θ_n	0.00	1.25	0.92	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93

As we can see, after just a few iterations, the estimated magnitude (radius) and argument values converged to the true values of 5 and 0.93, respectively. The angle θ is expressed in radians. This was first observed at the third iteration.

To visualize the convergence, we can plot the error in the estimated magnitude $|r_n - r^*|$ and the error in the estimated argument $|\theta_n - \theta^*|$ against the iteration number (n) on a graph. For this, we omit here.

The graph will show a decreasing trend in the errors as the iteration progresses, indicating the convergence of Halley's method. Thus the approximate solution computed for the given problem 2.1 is written as $z = 5e^{i0.93}$.This is equal to $z = 5\cos 0.93 + i5\sin 0.93 = 5(\cos 0.93 + i \sin 0.93)$.

2.1 - **The inverse function theorem**

Before going further we discuss the very fundamental inverse function theorem for an invertible matrix which is a significant tool for analysis in the linear bounded problems in Hilbert space.

Theorem1 , [2]. Let P and Q be Banach spaces in which both $A, B \in B(P,Q)$. Assuming further that A is invertible. If B satisfies

$$\|A - B\| < \frac{1}{\|A^{-1}\|}, \quad \text{then } B \text{ is also invertible and } \|B^{-1}\| \le \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\|A - B\|},$$

$$\|B^{-1} - A^{-1}\| \le \frac{\|A^{-1}\|^2 \|B - A\|}{1 - \|A^{-1}\|\|A - B\|}. \qquad (2.13)$$

The set of invertible map from P to Q is an open mapping in $B(P,Q)$ and, $A \to A^{-1}$ is continuous on the set. Thus

$$A^{-1} = \sum_{i=0}^{\infty} (I-A)^i = I + (I-A) + (I-A)^2 + ... + (I-A)^n + ... \leq \frac{1}{1-\|I-A\|}. \qquad (2.14)$$

2.2 The Zonal polynomials of a matrix.

Besides computing the polar decomposition of a diagonalizable matrix, we also discuss the associated zonal polynomials, a significant application in matrix analysis and its allied fields for the $n \times n$ matrix. Signifying notation with $C_k(A)$ to be the zonal polynomial [7,12] of a symmetric matrix of order n which corresponds to the partition $k = (k_1, k_2, ..., k_n), k_1 + k_2, ...k_n = k$, $k_1 \geq k_2 \geq ... \geq k_n \geq 0$. The first few values of $C_k(A)$ [7,12] are :

$C_{(1)}(A) = tr(A)$,

$C_{(2)}(A) = \frac{1}{3}\left[(trA)^2 + 2tr((A^2)\right]$,

$C_{(1^2)}(A) = \frac{2}{3}\left[(trA)^2 - tr(A)^2\right]$,

$C_{(3)}(A) = \frac{1}{15}\left[(trA)^3 + 6(trA)(trA^2) + 8tr(A^3)\right]$,

$C_{(2,1)}(A) = \frac{3}{5}\left[(trA)^3 + (trA)(trA^2) - 2tr(A^3)\right]$,

$C_{(1^3)}(A) = \frac{1}{3}\left[(trA)^3 - 3(trA)(trA^2) + 2tr(A^3)\right]$.

Making use of above expressions, we are able to write that :

$tr(A^2) = C_{(2)}(A) - \frac{1}{2}C_{(1^2)}(A)$,

$tr(A^3) = C_{(3)}(A) - \frac{1}{4}C_{(2,1)}(A) + \frac{1}{4}C_{(1^3)}(A)$,

$tr(A)tr(A^2) = C_{(3)}(A) + \frac{1}{6}C_{(2,1)}(A) - \frac{1}{2}C_{(1^3)}(A)$.

COMPUTATIONAL RESULTS

3.1 Computer implementation.

In an attempt at implementing above mentioned numerical methods we programmed a C ++ (and python) open source project polar decomposition that implements efficient versions of these functions.

Below is a python code [16] which generates a random matrix of size k and prints results. The generated matrix is a sample test matrix problem for the Newton and Halley's operators. The algorithm is given below.

```
Import numpy as np
# Generate a random matrix of size k
np.random.seed(42)
A=np.random.rand(k,k)
# perform polar decomposition using Newton's method
U=np.eye(k)    # Initial guess for orthogonal matrix U
tolerance=1e-6  #  convergence tolerance
```

```
max_iterations=100  #  maximum number of iterations:
converged=False
iterations=0
#  Create a list to store the intermediate results
intermediate_results=[ ]
while not converged and iterations < max_iterations:
      V=np.linalg.inv(U) @A   #  Calculate V using the inverse of U
      U_new = (U + V^T)/ 2.0    # Update U
      error=np.linalg.norm(U_new –U)   #  Calculate the error
      if  error<tolerance:
            converged=True
      U=U_new
      iterations+=1
     #  Store the iteration number and error in the intermediate results list
     Intermediate_ results.append((iterations, error))
Print intermediate results in tabular form
```

3.2 Results.

Table 3.1 shows a randomly generated matrix of order 8 obtained from the Matrix market by importing Sci-python for our use.

TABLE 3.1: Sample randomly generated Matrix A from the matrix market for numerical experiments.

Matrix A=

$$\begin{bmatrix} 0.5488135 & 0.71518937 & 0.60276338 & 0.54488318 & 0.4236548 & 0.64589411 & 0.43758721 & 0.891773 \\ 0.96366276 & 0.38344152 & 0.79172504 & 0.52889492 & 0.56804456 & 0.92559664 & 0.07103606 & 0.0871293 \\ 0.0202184 & 0.83261985 & 0.77815675 & 0.87001215 & 0.97861834 & 0.7991586 & 0.46147936 & 0.78052918 \\ 0.11827443 & 0.63992102 & 0.14335329 & 0.94466892 & 0.52184832 & 0.41466194 & 0.26455561 & 0.77423369 \\ 0.45615033 & 0.56843395 & 0.0187898 & 0.6176355 & 0.61209572 & 0.616934 & 0.94374808 & 0.6818203 \\ 0.3595079 & 0.43703195 & 0.6976312 & 0.06022547 & 0.66676672 & 0.67063787 & 0.21038256 & 0.1289263 \\ 0.31542835 & 0.36371077 & 0.57019677 & 0.43860151 & 0.98837384 & 0.10204481 & 0.20887676 & 0.16130952 \\ 0.65310833 & 0.2532916 & 0.46631077 & 0.24442559 & 0.15896958 & 0.11037514 & 0.65632959 & 0.13818295 \end{bmatrix}$$

The following results were computed and displayed below according to their numerical methods used for the values of Unitary matrices and corresponding Hermittian matrices.

TABLE 3.2: Newton's Method: Unitary Matrix (U)=

$$\begin{bmatrix} 0.20411988 & 0.4522557 & 0.38004077 & -0.11214485 & -0.21214479 & 0.66048918 & -0.24671944 & 0.26706623 \\ 0.50607635 & -0.02620399 & -0.05650647 & -0.21849292 & -0.06007407 & -0.47266413 & -0.48148962 & -0.44125491 \\ 0.37100856 & -0.304194462 & 0.0663003 & 0.25843961 & 0.16872525 & -0.0075416 & 0.52413767 & -0.60266434 \\ 0.3488741 & -0.24636949 & -0.07163029 & 0.49290228 & 0.43706791 & -0.12012179 & -0.43476911 & -0.41962045 \\ 0.33371435 & -0.1799194 & 0.0336644 & 0.55784463 & -0.42534872 & -0.07362365 & 0.48785293 & 0.29233219 \\ 0.23540511 & 0.39687266 & -0.74870811 & 0.03107605 & 0.07387216 & 0.01145147 & -0.28064822 & 0.30923533 \\ 0.28197348 & 0.38893449 & 0.31311868 & 0.35574562 & -0.60316199 & -0.23330011 & -0.02818011 & -0.32957357 \\ 0.54207671 & 0.1841618 & 0.0770955 & -0.52032716 & 0.35617674 & 0.07998478 & 0.16946855 & 0.43722799 \end{bmatrix}$$

TABLE 3.3: Corresponding Positive Semidefinite Hermitian Matrix (P)=

$$\begin{bmatrix}
0.72487911 & 0.95104836 & 0.78973024 & 0.69492443 & 0.47448591 & 0.76212691 & 0.67993268 & 1.02079133 \\
0.95104836 & 1.23687775 & 1.00275104 & 0.89391307 & 0.6067511 & 1.01229344 & 0.90296031 & 1.3484438 \\
0.78973024 & 1.00275104 & 0.87492607 & 0.74661845 & 0.49050973 & 0.84828164 & 0.75444796 & 1.13217815 \\
0.69492443 & 0.89391307 & 0.74661845 & 0.6669701 & 0.45543346 & 0.73214209 & 0.65204022 & 0.9839977 \\
0.47448591 & 0.6067511 & 0.49050973 & 0.45543346 & 0.32746453 & 0.52999246 & 0.46897661 & 0.71254842 \\
0.76212691 & 1.01229344 & 0.84828164 & 0.73214209 & 0.52999246 & 0.89971178 & 0.80013039 & 1.19485776 \\
0.67993268 & 0.9029603 & 0.75444796 & 0.65204022 & 0.46897661 & 0.80013039 & 0.71680886 & 1.07282962 \\
1.02079133 & 1.3484438 & 1.13217815 & 0.9839977 & 0.71254842 & 1.19485776 & 1.07282962 & 1.60770855
\end{bmatrix}$$

TABLE 3.4: Halley's method yielding Unitary matrix (U)=

$$\begin{bmatrix}
0.2411987 & 0.45225572 & 0.38004077 & -0.11214485 & -0.21214479 & 0.66048919 & -0.24671944 & 0.26706622 \\
0.50607635 & -0.02620399 & -0.05650647 & -0.21849292 & -0.06007406 & -0.47266414 & -0.48148962 & -0.44125491 \\
0.37100856 & -0.30419462 & 0.0663003 & 0.25843962 & 0.16872524 & -0.0075416 & 0.52413767 & -0.60266434 \\
0.3488741 & -0.2463695 & -0.07163029 & 0.49290228 & 0.43706791 & -0.12012179 & -0.43476911 & -0.41962045 \\
0.33371435 & -0.1799194 & 0.0336644 & 0.55784463 & -0.42534872 & -0.07362365 & 0.48785293 & 0.29233219 \\
0.2354051 & 0.39687266 & -0.74870811 & 0.03107605 & 0.07387216 & 0.01145147 & 0.166200609 & 0.3900305 \\
0.30575569 & 0.26815108 & 0.49363506 & -0.16334772 & 0.4789246 & -0.50743393 & 0.18041864 & 0.1112070 \\
0.54207671 & 0.18416179 & 0.076709955 & -0.52032716 & 0.35617673 & 0.07998477 & 0.16946854 & 0.43722799
\end{bmatrix}$$

TABLE 3.5: Halley's method yielding Positive Semi-definite Hermitian matrix (P)=

$$\begin{bmatrix}
0.72487911 & 0.95104836 & 0.78973024 & 0.69492443 & 0.76212691 & 0.67993268 & 1.02079133 \\
0.95104836 & 1.23687775 & 1.00275104 & 0.89391307 & 0.60675111 & 1.01229344 & 0.9029603 & 1.3484438 \\
0.78973024 & 1.00275104 & 0.87492607 & 0.74661845 & 0.49050973 & 0.84828164 & 0.75444796 & 1.13217815 \\
0.69492443 & 0.89391307 & 0.74661845 & 0.6669701 & 0.455433460 & 0.73214209 & 0.65204022 & 0.9839977 \\
0.47448591 & 0.6067511 & 0.49050973 & 0.45543346 & 0.32746453 & 0.52999246 & 0.46897661 & 0.71254842 \\
0.76212691 & 1.01229344 & 0.84828164 & 0.73214209 & 0.52999246 & 0.89971178 & 0.80013039 & 1.19485776 \\
0.67993268 & 0.9029603 & 0.75444796 & 0.65204022 & 0.46897661 & 0.80013039 & 0.71680886 & 1.07282962 \\
1.02079133 & 1.3484438 & 1.13217815 & 0.9839977 & 0.71254842 & 1.19485776 & 1.07282962 & 1.60770855
\end{bmatrix}$$

The next set of results following presents the weighted quadratically Newton method and weighted cubically Halley's method for polar decomposition using Matrix determinant and Condition number of their matrices. We denoted results for the Matrix Polar Decomposition using Matrix determinant as weighted Quadratically Newton's method (QWNM) and weighted Cubically Halley's method (CWHM) for the given random matrix of order 8. Tables 3.6 & 3.7 have the results for weighted Quadratically Newton's method which uses matrix determinant.

TABLE3.6: Quadratically Weighted Newton's method (QWNM): Unitary Matrix (U)=

$$\begin{bmatrix}
0.56804594 & -0.37653729 & -0.11542451 & -0.30597629 & 0.22234875 & 0.006770276 & -0.01609762 & 0.4329641 \\
-0.43510415 & 0.4672972 & -0.09129393 & 0.20747753 & 0.2448293 & -0.28739918 & 0.60775882 & 0.0862087 \\
-0.43013319 & 0.05844948 & 0.6087042 & 0.08480189 & -0.10015752 & 0.46355894 & -0.16800595 & -0.29841477 \\
-0.2169248 & 0.17739648 & 0.30888535 & 0.39631034 & -0.14664526 & 0.16286955 & -0.67321514 & -0.06715745 \\
0.17034382 & -0.37948083 & -0.15627214 & 0.12333401 & 0.42658629 & -0.01807541 & 0.21777644 & -0.60185694 \\
0.08169246 & 0.36516812 & 0.44606363 & -0.03332267 & 0.03934919 & -0.73315695 & -0.0320148 & -0.23205653 \\
-0.15206614 & 0.60113615 & -0.11461299 & -0.54259633 & -0.18953091 & -0.05895619 & -0.08449894 & 0.50031348 \\
0.38900341 & 0.07983567 & -0.28620445 & 0.01621899 & -0.56726363 & -0.21618453 & 0.50134428 & 0.12227498
\end{bmatrix}$$

TABLE 3.7: Quadratically Weighted Newton's method (QWNM) yields Positive Semidefinite Hermitian Matrix (P)=

$$\begin{bmatrix}
3.75433657 & 2.21630549 & 2.06070054 & 2.27994997 & 2.38971533 & 2.26000347 & 2.44123651 & 3.14917263 \\
2.21630549 & 2.78260137 & 1.78107385 & 2.08936226 & 2.17574151 & 1.99776368 & 2.01970689 & 2.59879535 \\
2.06070054 & 1.78107385 & 2.34698454 & 1.76416262 & 1.75721412 & 1.80833813 & 1.76376203 & 2.28922148 \\
2.27994997 & 2.08936226 & 1.76416262 & 2.42013942 & 2.01229361 & 1.94467783 & 2.21726141 & 2.68592748 \\
2.38971533 & 2.17574151 & 1.75721412 & 2.01229361 & 2.4568938 & 2.05524173 & 2.18768878 & 2.82939316 \\
2.26000347 & 1.99776368 & 1.80833813 & 1.94467783 & 2.05524173 & 2.27247897 & 1.87063513 & 2.53113634 \\
2.44123651 & 2.01970689 & 1.76376203 & 2.21726141 & 2.18768878 & 1.87063513 & 2.74001277 & 2.928618 \\
3.14917263 & 2.59879535 & 2.28922148 & 2.68592748 & 2.82939316 & 2.53113634 & 2.928618 & 3.93428429
\end{bmatrix}$$

TABLE 3.8: Cubically Weighted Halley's Method (CWHM) yields Unitary Matrix (U)=

$$\begin{bmatrix}
0.65949619 & -0.19329666 & -.028066943 & -0.30820252 & 0.21964777 & 0.20609246 & 0.06490746 & -0.38916266 \\
0.08445451 & 0.54869892 & -0.2660302 & 0.13466756 & -0.08627366 & -0.31547714 & -0.12205035 & 0.64709183 \\
0.03389212 & -0.07933152 & 0.38304617 & 0.006952 & -0.06430929 & 0.05358886 & 0.73638042 & -0.53466261 \\
-0.21371685 & 0.20898223 & 0.28403026 & 0.57299379 & -0.23969433 & -0.0994467 & -0.43016691 & -0.43630064 \\
0.06908915 & -0.21499409 & -0.11757829 & 0.05183468 & 0.47808336 & -0.69590837 & 0.0280829 & -0.24410254 \\
0.19077788 & -0.46655425 & 0.5723207 & -0.12345018 & -0.1090414 & 0.31766925 & -0.21734633 & -0.43331962 \\
0.01822694 & 0.13736159 & -0.44128953 & -0.5892225 & -0.34101195 & -0.03257271 & -0.0237138 & -0.54705143 \\
-0.31064308 & -0.22566992 & 0.26685406 & -0.08220778 & 0.48616542 & -0.32717282 & 0.24417153 & 0.40503697
\end{bmatrix}$$

TABLE 3.9: Cubically Weighted Halley's method (CWHM) yields Positive Semidefinite Hermitian Matrix (P)=

$$\begin{bmatrix}
3.1918726 & 0.512696 & -0.5634457 & -0.51978794 & 0.02817192 & -0.23467424 & -0.21250992 & -0.3357201 \\
0.51269616 & 1.26053644 & 0.39861021 & 0.1773459 & 0.05107402 & -0.07485699 & -0.15788015 & -0.37992809 \\
-0.5634457 & 0.39861021 & 1.23231933 & 0.35826872 & 0.10321405 & -0.04854467 & -0.21447647 & -0.47055984 \\
-0.51978794 & 0.1773459 & 0.35826872 & 1.31226904 & 0.13231896 & -0.03900168 & -0.26170984 & -0.59428256 \\
0.02817192 & 0.05107402 & 0.10321405 & 0.13231896 & 0.90072402 & -0.01156396 & -0.0421253 & -0.11361866 \\
-0.23467424 & -0.07485699 & -0.04854467 & -0.03900168 & -0.01156396 & 0.7636873 & 0.01745729 & -0.07673967 \\
-0.21250992 & -0.15788015 & -0.21447647 & -0.26170984 & -0.0421253 & 0.01745729 & 0.81053161 & -0.14067253 \\
-0.3357201 & -0.37992809 & -0.47055984 & -0.59428256 & -0.11361866 & -0.07673967 & -0.14067253 & 1.23460178
\end{bmatrix}$$

The following Table below displays results for Cubically Weighted Halley's method which uses matrix condition number.

TABLE 3.10: Cubically Weighted Halley's Method (CWHM) yields Unitary Matrix (U)=

$$\begin{bmatrix}
0.65949619 & -0.19329666 & -0.28066943 & -0.30820252 & 0.21964777 & 0.20609246 & 0.06490746 & -0.38916266 \\
0.08445451 & 0.54869892 & -0.2660302 & 0.13466756 & -0.08627366 & -0.31547714 & -0.12205035 & 0.64709183 \\
0.03389212 & -0.07933152 & 0.38304617 & 0.0069525 & -0.06430929 & 0.05358886 & 0.73638042 & -0.53466261 \\
-0.21371685 & 0.20898223 & 0.28403026 & 0.57299379 & -0.23969433 & -0.0994467 & -0.43016691 & -0.43630064 \\
0.06908915 & -0.21499409 & -0.11757829 & 0.05183468 & 0.47808336 & -0.69590937 & 0.0280829 & -0.24410254 \\
0.19077788 & -0.46655425 & 0.5723207 & -0.12345018 & -0.1090414 & 0.31766925 & -0.21734633 & -0.43331962 \\
0.01822694 & 0.13736159 & -0.44128953 & -0.5892225 & -0.34101195 & -0.03257271 & -0.0237138 & -0.54705143 \\
-0.31064308 & -0.22566992 & 0.26685406 & -0.08220778 & 0.48616542 & -0.32717282 & 0.24417153 & 0.40503697
\end{bmatrix}$$

TABLE 3.11: Cubically convergent Halley's method yields Semidefinite Hermitian Matrix (P)=

$$\begin{bmatrix}
3.1918726 & 0.51269616 & -0.5634457 & -0.51978794 & 0.02817192 & -0.23467424 & -0.21250992 & -0.3357201 \\
0.51269616 & 1.26053644 & 0.39861021 & 0.1773459 & 0.05107402 & -0.07485699 & -0.15788015 & -0.37992809 \\
-0.5634457 & 0.39861021 & 1.23231933 & 0.35826872 & 0.10321405 & -0.04854467 & -0.21447647 & -0.47055984 \\
-0.51978794 & 0.1773459 & 0.35826872 & 1.31226904 & 0.13231896 & -0.03900168 & -0.26170984 & -0.59428256 \\
0.02817192 & 0.05107402 & 0.10321405 & 0.13231896 & 0.90072402 & -0.01156396 & -0.0421253 & -0.11361866 \\
-0.23467424 & -0.07485699 & -0.04854467 & -0.03900168 & -0.01156396 & 0.7636873 & 0.01745729 & -0.07673967 \\
-0.21250992 & -0.15788015 & -0.21447647 & -0.026170984 = & -.0421253 & 0.01745729 & 0.81053161 & -0.14067253 \\
-0.3357201 & -0.37992809 & -0.47055984 & -0.59428256 & -0.11361866 & -0.07673967 & -0.14067253 & 1.23460178
\end{bmatrix}$$

TABLE 3.12: Shows results computed from equation (1.3) the Pade's family

Iteration 1: $X^{(1)} =$

$$\begin{bmatrix}
2.827006160.233015470.31038446 & -0.00921032 & -0.07690304 & -0.25840321 & -0.015341771 & -0.30149479 \\
-0.041207680.94450717 & -0.125561550.0234574 & -0.0234574 & -0.10668744 & -0.08986693 & -0.08768057 & -0.05251277 \\
0.13719615 & -0.070386140.879862470.01699777 & -0.058899230.02426515 & -0.07726356 & -0.1138769 \\
0.04220175 & -0.002381620.027615690.96417679 & -0.072146710.03449732 & -0.07544957 & -0.12033143 \\
-0.034055080.07032434 & -0.057183190.047068710.99665996 & -0.040705570.01016989 & -0.02368149 \\
0.04344023 & -0.066413140.02808315 & -0.068493170.036825470.995190970.043293910.08011326 \\
-0.01573647 & -0.01210136 & -0.05252706 & -0.031693720.005927620.038609930.99309782 & -0.00309056 \\
0.048095390.00964859 & 0.08871729 & -0.08229957 & -0.025409610.07009061 & -0.001349320.9935793
\end{bmatrix}$$

Iteration 2: $X^{(2)} =$

$$\begin{bmatrix}
2.96244154\ 0.23804682\ 0.34836719 & -0.01486034 & -0.10249419 & -0.26126274 & -0.15215517 & -0.30754577 \\
-0.04477846\ 0.93819966 & -0.12183565\ 0.02235571 & -0.10460325 & -0.0861334 & -0.08728729 & -0.05004999 \\
0.14181015 & -0.07255235\ 0.87722017\ 0.01555929 & -0.06045038\ 0.0217706 & -0.08026672 & -0.11198688 \\
0.04503546 & -0.00059271 & 0.03366322\ 0.95682013 & -0.06661664\ 0.03624874 & -0.06861078 & -0.10761898 \\
-0.03599833\ 0.07204568 & -0.0600784\ 0.04654418\ 0.99299342 & -0.03841128\ 0.1062725 & -0.02357642 \\
0.04500625 & -0.06787242\ 0.02773833 & -0.06532643\ 0.03477536\ 0.99131291\ 0.04314766\ 0.07654178 \\
-0.01630116 & -0.01327936 & -0.05403303 & -0.03065371\ 0.00630826\ 0.03609503\ 0.99131947 & -0.00235906 \\
0.04861897\ 0.00989143\ 0.09081083 & -0.08036015 & -0.0245614\ 0.06643754 & -0.00098937\ 0.99175833
\end{bmatrix}$$

Corresponding values of Y_k for k-Iterations

Iteration 1. $Y^{(1)} =$

$$\begin{bmatrix}
2.82700616\ 0.23301547\ 0.31038446 & -0.00921032 & -0.07690304 & -0.25840321 & -0.15341771 & -0.30149479 \\
-004120768\ 0.94450717 & -0.12556155\ 0.0234574 & -0.10668744 & -0.08986693 & -0.08768057 & -0.05251277 \\
0.13719615 & -0.07038614\ 0.87986247\ 0.01699777 & -0.05889923\ 0.02426515 & -0.07726356 & -0.1138769 \\
0.04220175 & -0.00238162\ 0.02761569\ 0.96417679 & -0.07214671\ 0.03449732 & -0.07544957 & -0.12033143 \\
-0.03405508\ 0.070342434 & -0.05718319\ 0.04706871\ 0.99665996 & -0.04070557\ 0.01016989 & -0.02368149 \\
0.04344023 & -0.06641314\ 0.02808315 & -0.06849317\ 0.03682547\ 0.99519097\ 0.04329391\ 0.08011326 \\
-0.01573647 & -0.01210136 & -0.05252706 & -0.03169372\ 0.00592762\ 0.03860993\ 0.99309782 & -0.00309056 \\
0.04809536\ 0.00964859\ 0.08871729 & -0.08229957 & -0.02540961\ 0.07009061 & -0.00134932\ 0.9935793
\end{bmatrix}$$

Iteration 2: $Y^{(2)} =$

$$\begin{bmatrix}
2.95748130.238132680.3421437 & -0.014807 & -0.10246612 & -0.26122811 & -0.0.152116668 & -0.307493 \\
-0.044750560.93824233 & -0.121842620.022362 & -0.10459571 & -0.08614407 & -0.08728875 & -0.05006129 \\
0.14180362 & -0.072551630.87721890.01556123 & -0.060452820.02177334 & -0.08026744 & -0.11198815 \\
0.04503702 & -0.000592510.033665530.95681855 & -0.066617710.03625017 & -0.06860966 & -0.10762985 \\
-0.035996060.07204106 & -0.060074150.046541590.99299571 & -0.038411210.01062778 & -0.02357723 \\
0.04500516 & -0.06786990.02773771 & -0.065328090.03477670.991313960.043148490.07654383 \\
-0.01630066 & -0.01327876 & -0.05403226 & -0.030654370.006308470.03609550.99132019 & -0.00235959 \\
0.048618390.009891270.09081026 & -0.08036088 & -0.024562 & 0.06643819 & -0.000989740.99175879
\end{bmatrix}$$

TABLE 3.13: The zonal polynomials (Power sums) using the trace of the matrix from Table 1.

Zonal polynomial/ k	1	2	3	4
$Z_k(A) = Tr(A)$	4.6311	23.6126	152.6126	989.2512
Zonal polynomial k	5	6	7	8
$Z_k(A)$	6417.1003	41971.6236	274199.3492	1791202.8357

DISCUSSION

We implemented the polar decomposition formula for the matrix using the Newton's method and Halley's method. Also implemented were the variants of Newton's method and Halley's method whereby we used the determinant of the matrix and the Condition number of the matrix to speed up computation processes. The so called variants of these two methods of Newton and Halley iterations are termed Quadratically Weighted Newton method (QWNM) and Cubically Weighted Halley's Method (CWHM) respectively.

Because $X_k = U\sum_k V^T$ as the SVD of X_k for $h(\lambda_k)$ it follows that $[\sigma_{\min}(X_k), \sigma_{\max}(X_k)] \subset [h(\lambda_k), 1] \subset (0,1]$ as it pertains to equation (1.7) in which the interval of values hold. It thus follows that $\sigma_i(X_{k+1}) = g_k(\sigma_i(X_k))$ where g_k is as defined in equation (1.9). This means that $[\sigma_{\min}(X_{k+1}), \sigma_{\max}(X_{k+1})] \subseteq \left[\min_{h(\lambda_k)\le x\le 1} g_k(x), \max_{h(\lambda_k)\le x\le 1} g_k(x)\right]$. This expression gives the lower bound for the singular value of the matrix X_k, $k \ge 1$. The situation where X_k is a square matrix is a limiting case. In this later condition the matrices U_* and V_* are the orthonormal matrices corresponding to their respective eigenvectors.

In a situation where the determinant of a matrix vanishes it is recommended to regularize the matrix using the technique of Tikhonov regularization parameter in the form
$(A + \tau I)$ where τ is a parameter lying between (0,1]

Computational efficiency of these methods can be executed using definition of the computational order of convergence in the sense of [8]. The quantity in the sense of [8] which describes the Q_R -order of convergence is given by

$$Q_R = \frac{\ln(R_{k+1}/R_k)}{\ln(R_k/R_{k-1})} \quad , k = 0,1,... \qquad (4.1)$$

where it is defined that

$$R_{k+1} = \frac{\|X_{k+1} - X_k\|_\infty}{\|X_k\|_\infty} < \varepsilon \quad . \qquad (4.2)$$

The value of ε is a pre-assigned tolerance threshold with which iteration is halted to a stop. Nevertheless detail descriptions of Q_R are omitted due to [8]. The backward error for the polar decomposition $A = UP_d$ is given by $\frac{\|A - UP_d\|_F}{\|A\|_F}$ while the orthogonality measure is

$\max\left(\frac{\|U^TU - I\|_F}{\sqrt{n}}, \frac{\|P_d^TP_d - I\|_F}{\sqrt{n}}\right)$. Considering equation (1.1) written as $A = U\sum V^T$ it is pertinent to note that its backward error is $\frac{\|A - U\sum V^T\|_F}{\|A\|_F}$ and the accompanying orthogonality measure is

$\max\left(\frac{\|U^TU - I\|_F}{\sqrt{n}}, \frac{\|V^TV - I\|_F}{\sqrt{n}}\right)$.

All computed results are displayed in Tabular forms yielding the unitary matrices (U) and their accompanying Positive semi-definite matrices (P) for each of Newton's and Halley's methods respectively as shown in Tables 3.2-3.9. We used the determinant of a matrix and matrix condition numbers in the formulations which helped in speeding up the computation process yielding high qualitative results. We further computed with equations (1.2 and 1.3) whose results are presented in Tables 3.12 for a class of Pade's family of methods. Thus the Quadratically Weighted Newton Method (QWNM) and Cubically Weighted Halley's Method (CWHM) require solving linear system per iteration using the LU Factorization techniques to invert a matrix in the polar decomposition. The arithmetical costs involve are: Newton's method ($O(n^2)$) and Halley's method ($O(n^3)$). This again demonstrated that the cubically convergent Halley's method has superiority over the quadratically Newton's method

CONCLUSION

The paper studied high speed computational methods for computing the Polar decomposition of a matrix by using Newton's methods and Halley's method. We solved a complex number $z = re^{i\theta}$ represented in the form polar coordinate using Halley's third order convergent method for solving polynomial equation $f(x) = 0$. The analysis was given in terms of error bounds for the radius and argument in the complex number. In the matrix case, computing a polar decomposition using numerical iterative schemes led to incorporating into formulations the uses of determinant of a matrix and matrix condition number for speeding up computation processes. It requires solving a linear system per iteration using LU Factorization approach. The results computed are of high quality as shown in Tables 3.2-3.12. A class of zonal polynomials for a sample symmetric matrix was computed equals the trace of the matrix.

Funding statement: Not applicable.

Author Statement: The author declares no conflict of interest.

The author is grateful for the Fund received from Kogi State University, Anyigba under TETFUND grant of 2016.

REFERENCES

[1] Benner,p., Nakatsukasa, Y. & Penke,C. 2021 Stable and efficient computation of generalized polar decomposition, arXiv: 2104.06659v1 [Math NA].

[2] Bjorck, A. 2009 Numerical methods in Scientific Computing Vol. 2, SIAM, Philadelphia.

[3] Farquhar, N.E., Moroney,T.J., Yang, Q. & Turner,I.W. 2016 GPU accelerated vector products with applications to exponential integrators and fractional diffusion, SIAM Journal of Scientific Computing, 38(3), C127-C149.

[4] Higham,N.I. 1986 Computing the polar decomposition- with applications, SIAM. J. Sci. Stat. Comput. Vol 7, No. 4, pp. 1160-1174.

[5] Gander, W. 1990 Algorithms for the polar decomposition, SIAM J. Sci. Comput., 11, pp. 1102-1115.

[6] Gwalik,E.S. 2020 Iterations for the Unitary Sign decomposition and the Unitary eigendecomposition, arXiv:2011.12449v1 [Math.NA].

[7] James,A.T. 1964 ‘’Distributions of matrix variates and latent roots derived from normal samples,’’ Annals of Mathematical Statistics, Vol. 35, pp. 475-501.

[8] Ortega J.M. & Rheinboldt,W.C. 2000 Iterative solution of Nonlinear equations in several variables, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, USA.

[9] Nakatsukasa, Y. & Freund,R. 2016 Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev’s functions, SIAM J. Numer. Anal. Vol. 58, Issue: 3, pp. 461-493.

[10] Nakatsuksa, Y. & Higham, N.J. 2021 Backward stability of iterations for computing the polar decomposition. SIAM J. Matrix Anal. Appl. Vol. 33, No. 2, pp. 460-479.

[11] Nakatsukasa,Y., Bai, Z. & Gygi,F. 2010 Optimising the Halley’s iteration for computing the matrix polar decomposition. SIAM J. Matrix Anal. Appl. Vol. 31, No. 5, pp. 2700-2720.

[12] Nagar,D.K., Joshi, L. & Gupta,A.K. 2012 International Scholarly Research Network, Matrix variate Pareto distribution of second kind.ISRN Probability and Statistics Vol. 2012 Article ID789273,20 Pages. DOI:10.54021/2012/789273.

[13] Uwamusi,S.E. 2017 Extracting p-th root of a matrix with positive eigenvalues via Newton and Halley’s methods, University of Ilorin Journal of Science Vol. 4(1) , pp. 1-16. ISSN:2408-4840.

[14] Uwamusi,S.E. 2024 Rotation matrix and Angles of rotation in the polar decomposition, Earthline Journalof Mathematical Sciences, E-ISSN:2581-8147, Vol 14(1), pp. 63-74. https:doi.org/10.34198/ejms14124.063074

[15] Wu,L., Laeuchli,J.& Kalantzis,V. 2016 A. Stathopoulos and E. Gallopoulos, Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse, Journal of Computational Physics, 326, pp. 828-844.

[16] https://chat.openai.com/share/c14412f1-9d87-4cd3-8353-f45683ce14a2.

APPENDIX

METHOD 1: CODES FOR COMPUTING MATRIX POLAR DECOMPOSITION USING NEWTON'S AND HALLEY'S METHODS.

```
import  numpy as np
# Set the random seed for reproducibility
np.random.seed(0)
# Generate a random matrix of order 8
A=np.random.rand(8,8)
# Function to compute the Matrix Polar Decomposition using Newton's method
def polar_decomposition_newton(A):
      n=A.shape[0]
      X=A.copy( )
      # Iteration loop
      for_ in range (20): # Adjust the number of iterations as needed
           Y=0.5*(X+np.linalg.inv(X.T))
                X=0.5*(X+np.linalg.inv(Y.T))
           U=Y
           P=np.dot(U.T,A)
           return U,P
# Function to compute the Matrix Polar Decomposition using Halley's method
def  polar _decomposition halley(A):
   n=A.shape[0]
  X=A.copy( )
  # Iteration loop
  for _ in range(20): # Adjust the number of iterations as needed
            X_inv=np.linalg.inv(X)
            X_sq=np.dot(X,X)
            X_cube=np.dot(X_sq,X)
            F=np.dot(X_sq,A)-X
            J=4*X_cube-4*X
            X=X+np.dot(X,F)/np.linalg.norm(F,'fro')**2-np.dot(X,np.dot(X,F))
   U=X
   P=np.dot(U.T,A)
   return U,P
   # Perform the Matrix Decomposition using Newton's method
```

```
U_newton, P_newton=polar decomposition_newton(A)
# Perform the Matrix Polar Decomposition using Halley's method
U_halley, P_halley=polar decomposition_halley(A)
#  Print the original matrix
print( " Original Matrix:" )
print (A)
print (  )
# Print the results using Newton's method
print  ( " -----------------" )
 Print ( " Unitary Matrix (U):" )
 Print (U_newton)
 Print (  )
 Print ( " positive Semidefinite Hermittian Matrix (P) :'')
 Print ( P_newton)
 Print (  )
 # Print the results using Halley's method
 print ( "Halley's method:")
 print ( " -----------------" )
 print ( " Unitary Matrix (U) : " )
 print ( U_halley)
 print (  )
 print ( " Positive Semidefinite Hermittian Matrix (P) : " )
 print ( P_halley)
 print (  )
```

Remarks: Note that when you run this code, it will generate a random matrix 'A' of order 8. It then performs the Matrix Polar Decomposition using both Newton's method and Halley's method. The original matrix 'A" , along with unitary matrix 'U' and positive semidefinite Hermittian matrix 'P' obtained using each method are displayed in tabular form.

METHOD 2: CODES FOR COMPUTING FORMULAR

$$X_{k+1} = X_k(3I - X_k^2)(I + 3X_k^2)^{-1},\ (k = 0,1,...) \quad \text{(Halley's method)}$$

AND

$$X_{k+1} = 0.5(X_k + X_k^{-T}), X_0 = A,\ \ (k = 0,1,...) \quad \text{(Newton's method).}$$

```
python

import  numpy as np

# Generate a random matrix of 8
A=np.random.rand(8,8)

# Initialize X(k) with A
X=A
# Iterate until convergence
while True:
```

```
 X_squared=np.dot(X,X)
 numerator=3*np.eye(8)-X_squared
 denominator=np.linalg.inv(np.eye(8)+3*X_squared)
 X_next=np.dot(X,dot(numerator,denominator))

  check for convergence
  if np.allclose(X,X_next):
  P=np.dot(U.T,A)

   return U,P
# perform the Matrix Polar Decomposition using Newton's method
U_newton, P_newton=polar decomposition_newton (A)

# Perform the Matrix Polar Decomposition using Halley's method
U_halley,P_halley=polar_decomposition_halley(A)

# Print the result
print (" Matrix Polar Decomposition using Newton's method: " )
print (Unitary Matrix (U) : " )
print (  )
print ( " Positive Semidefinite Hermittian Matrix (P) : " )
print (  )

print( '' Matrix Polar Decomposition using Halley's method : '' )
print ( '' Unitary Matrix (U) : '' )
print ( U_halley)
print (  )
print ( '' Positive Semidefinite Hermittian Matrix (P) : '' )
print (P_halley)
print (   )
```