Project : Computing the Polar Decomposition

Zhengbo Zhou*

September 22, 2022

1 Preliminaries

1.1 Vector Norms

Before introducing the matrix norm, a brief illustration of the vector norm is necessary.

Definition 1.1 (Vector Norm)**.** A vector norm on $\mathbb{C}^n$ is a function $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ such that it satisfies the following properties

1. $\|x\| \geq 0$ for all $x \in \mathbb{C}^n$,
2. $\|x\| = 0$ if and only if $x = 0$,
3. $\|\lambda x\| = |\lambda|\|x\|$ for all $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$,
4. $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{C}^n$.

Example 1.2. For $x \in \mathbb{C}^n$,

$$\begin{aligned} \text{1-norm}: \ \|x\|_1 &= \sum_{i=1}^{n} |x_i|, \\ \text{2-norm (Euclidean Norm)}: \ \|x\|_2 &= \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} = \sqrt{x^* x}, \\ \infty\text{-norm}: \ \|x\|_\infty &= \max_{1 \leq i \leq n} |x_i|. \end{aligned}$$

These are all special cases of the p-norm,

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \quad q \geq 1. \tag{1.1}$$

*School of Mathematics, University of Manchester, Manchester, M13 9PL, England (zhengbo.zhou@postgrad.manchester.ac.uk).

1.2 Matrix Norms

Definition 1.3 (Matrix Norms)**.** A matrix norm is a function $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}$ satisfies the analogues of the four properties in the Definition 1.1.

Example 1.4 (Frobenius norm)**.** For $A \in \mathbb{C}^{m\times n}$, the Frobenius norm is defined as

$$\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\right)^{1/2} = \left(\text{trace}(A^*A)\right)^{1/2}.$$

Example 1.5 (Subordinate matrix norm)**.** The matrix norm induced by a vector norm is called the subordinate norm. Suppose $\|\cdot\|$ is a vector norm, the corresponding subordinate matrix norm is defined as

$$\|A\| = \max_{\|x\|=1}\|Ax\|, \quad A \in \mathbb{C}^{m\times n}, \quad x \in \mathbb{C}^n.$$

Apply this definition into (1.1), we have the definition for the matrix p-norm

$$\|A\|_p = \max_{\|x\|_p=1}\|Ax\|_p.$$

The subordinate matrix norms for 1-, 2- and ∞-norms can be shown to have the following form

$$\begin{aligned}
\|A\|_1 &= \max_{1\le j\le n}\sum_{i=1}^{m}|a_{ij}|,\\
\|A\|_2 &= \left(\rho(A^*A)\right)^{1/2} = \sigma_{\max}(A),\\
\|A\|_\infty &= \max_{1\le i\le m}\sum_{j=1}^{n}|a_{ij}|,
\end{aligned}$$

where $\rho(A)$ represents the spectral radius of the matrix A, which is defined as the largest eigenvalue in magnitude of A and $\sigma_{\max}(A)$ represents the largest singular value of A.

We say a matrix norm $\|\cdot\|$ is consistent if for all $A, B \in \mathbb{C}^{m\times n}$, the following inequality holds whenever the product AB defines

$$\|AB\| \le \|A\|\|B\|.$$

The Frobenius norm and all subordinate norms are consistent.

1.3 Singular Value Decomposition

Theorem 1.6 (Singular value decomposition)**.** *If $A \in \mathbb{C}^{m\times n}$, $m \ge n$, then there exists two unitary matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ such that*

$$A = U\Sigma V^*, \quad \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{m\times n}, \tag{1.2}$$

where $\sigma_1, \ldots, \sigma_n$ are all non-negative and arranged in non-ascending order. We denote (1.2) as the singular value decomposition (SVD) of A and $\sigma_1, \ldots, \sigma_n$ are the singular values of A.

Alternatively, we can write the SVD of A as $U\Sigma_r V^*$ where the subscript r represents the number of nonzero elements on the diagonal of Σ. Therefore, if A has full column rank, we write $A = U\Sigma_n V$.

2 Polar Decomposition

Throughout this project, we focused on $A \in \mathbb{C}^{n\times n}$. In complex analysis, it is known that for any $\alpha \in \mathbb{C}$, we can write α in polar form, namely $\alpha = re^{i\theta}$. The polar decomposition is its matrix analogue.

Theorem 2.1 (Polar Decomposition [7, 2008, Theorem 8.1]). *Let $A \in \mathbb{C}^{m\times n}$ with $m \geq n$. There exists a matrix $U \in \mathbb{C}^{m\times n}$ with orthonormal columns and a unique Hermitian positive semidefinite matrix $H \in \mathbb{C}^{n\times n}$ such that $A = UH$. The matrix H is given by $H = (A^*A)^{1/2}$. If the matrix A is full rank, then H is Hermitian positive definite, and U is uniquely determined.*

Proof. Suppose $\operatorname{rank}(A) = r$, then let A has a SVD $A = P\Sigma_r V^*$. The polar decomposition of A can be formed in terms of the SVD:

$$A = P\begin{bmatrix} I_r & 0 \\ 0 & I_{m-r,n-r} \end{bmatrix} V^*V \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0_{n-r,n-r} \end{bmatrix} V^* =: UH.$$

where

$$U = P\begin{bmatrix} I_r & 0 \\ 0 & I_{m-r,n-r} \end{bmatrix} V^*, \quad H = V\begin{bmatrix} \Sigma_r & 0 \\ 0 & 0_{n-r,n-r} \end{bmatrix} V^*. \tag{2.1}$$

We can test the columns' orthonormality of U by performing

$$U^*U = V\begin{bmatrix} I_r & 0 \\ 0 & I_{n-r,m-r} \end{bmatrix} P^*P \begin{bmatrix} I_r & 0 \\ 0 & I_{m-r,n-r} \end{bmatrix} V^* = I_{n,n}.$$

The symmetry of H is obvious. Notice that, H and $\begin{bmatrix} \Sigma_r & 0 \\ 0 & 0_{n-r,n-r} \end{bmatrix}$ are unitarily similar, hence they share the same eigenvalues. Equivalently speaking, the eigenvalues of H are the singular values of A. From Theorem 1.6, the singular values of A are all real and non-negative, hence H is Hermitian positive semidefinite and it is uniquely determined via

$$(A^*A)^{1/2} = (H^*U^*UH)^{1/2} = \left(H^2\right)^{1/2} = H. \tag{2.2}$$

If A is full rank, namely $r = n$, then all the singular values of A are positive, and consequently H's eigenvalues are all positive which is equivalent as H is Hermitian positive definite. Clearly if H is nonsingular, then U is uniquely determined by $U = AH^{-1}$. □

We will refer to U as the unitary polar factor. If A is a square matrix and has a SVD $A = P\Sigma_r V$, then the expression for the polar factor in (2.1) can be simplified to $U = PV^*$. Furthermore, if $\operatorname{rank}(A) = n$, then $H = V\Sigma_n V^*$.

We can restrict our attention to nonsingular square matrices instead of general rectangular matrices. Let $A \in \mathbb{C}^{m\times n}$ with $m \geq n$ have the QR factorization $A = QR$ where $Q \in \mathbb{C}^{m\times n}$ and $R \in \mathbb{C}^{n\times n}$. We can find the polar decomposition $R = UH$ and $A = QU \cdot H$ becomes the polar decomposition of A. In addition, If $A \in \mathbb{C}^{n\times n}$ is singular, we can produce a complete orthogonal decomposition

$$A = P\begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} Q^*, \tag{2.3}$$

where P and Q are unitary, and $R \in \mathbb{C}^{r\times r}$ is nonsingular and upper triangular ([7, 2008, pp. 196], [1, 1999, Section 2.4.2.4] and [4, 1996, Definition 1.3.1.]). We can now compute the polar decomposition of R and assemble them together.

Theorem 2.2. *For $A \in \mathbb{C}^{m\times n}$, let $A = UH$ be its polar decomposition. A is normal if and only if U and H are commute.*

Proof. ($\Leftarrow$): If U and H are commute, then

$$\begin{aligned} AA^* &= (UH)(UH)^* \\ &= (HU)(HU)^* = HUU^*H^* = H^2. \\ A^*A &= H^2 = AA^*. \end{aligned}$$

Hence A is normal.

($\Rightarrow$): If A is normal, we have $AA^* = UHH^*U^* = UH^2U^*$ and $A^*A = H^2$. By taking the principal square root of both sides of $UH^2U^* = H^2$, we have $UHU^* = H$ (see [7, 2008, Theorem 1.26 and 1.29]). Since U is unitary, we have $UH = HU$, which completes the proof. □

Theorem 2.3. *For any $A \in \mathbb{C}^{n\times n}$ with $\operatorname{rank}(A) = n$, the following expression holds*

$$U = \frac{2}{\pi} A \int_0^\infty (t^2 I + A^*A)^{-1} \mathrm{d}t. \tag{2.4}$$

Proof. Let A have the SVD $A = P\Sigma V^*$. Then the unitary polar factor can be written as $U = PV^*$. Substituting them into the equation, we have

$$PV^* = \frac{2}{\pi} P\Sigma V^* \int_0^\infty (t^2 I + V\Sigma^*\Sigma V^*)^{-1} \mathrm{d}t.$$

Premultiply P^* and using the symmetry of Σ we have

$$V^* = \frac{2}{\pi} \Sigma V^* \int_0^\infty (t^2 I + V\Sigma^2 V^*)^{-1} \mathrm{d}t. \tag{2.5}$$

Since V is unitary, we can write V^* as V^{-1}, and using the formula $(AB)^{-1} = B^{-1}A^{-1}$ we can rewrite (2.5) as

$$\begin{aligned} V^* &= \frac{2}{\pi} \Sigma \int_0^\infty V^{-1}(t^2 I + V\Sigma^2 V^{-1})^{-1} \mathrm{d}t \\ &= \frac{2}{\pi} \Sigma \int_0^\infty \left(t^2 V + V\Sigma^2 V^{-1} V\right)^{-1} \mathrm{d}t \\ &= \frac{2}{\pi} \Sigma \int_0^\infty \left(t^2 V + V\Sigma^2\right)^{-1} \mathrm{d}t. \end{aligned}$$

Postmultiply $(V^*)^{-1} = V$ on both sides we have

$$\begin{aligned} I &= \frac{2}{\pi} \int_0^\infty \Sigma \left(t^2 V + V\Sigma^2\right)^{-1} (V^*)^{-1} \mathrm{d}t \\ &= \frac{2}{\pi} \int_0^\infty \Sigma \left(t^2 I + \Sigma^2\right)^{-1} \mathrm{d}t. \end{aligned}$$

Notice that, the matrix I and $\Sigma\left(t^2 I + \Sigma^2\right)^{-1}$ are all diagonal, therefore we can divide the problem element-wise

$$\frac{2}{\pi}\int_0^\infty \frac{\sigma_i}{t^2+\sigma_i^2}\mathrm{d}t = 1, \quad \text{for all } i = 1, \ldots, n. \tag{2.6}$$

The remaining task is to show that the integral from the left side of (2.6) is equal to 1. First notice that, since A is full rank, the singular values $\sigma_1, \ldots, \sigma_n$ are all positive. Then we can evaluate the integral for each $i = 1, \ldots, n$

$$\frac{2}{\pi}\int_0^\infty \frac{\sigma_i}{t^2+\sigma_i^2}\mathrm{d}t = \frac{2}{\pi}\sigma_i\int_0^\infty \frac{1}{t^2+\sigma_i^2}\mathrm{d}t = \frac{2}{\pi}\sigma_i\frac{1}{\sigma_i}\left[\arctan(t/\sigma_i)\right]_{t=0}^{t=\infty} = 1.$$

Hence we proved the expression (2.4) holds. □

3 Newton's Method

The Newton iteration for the unitary factor of a square matrix A can be derived by applying the Newton's method to the equation $X^*X = I_n$. Consider a function $F(\widetilde{X}) = \widetilde{X}^*\widetilde{X} - I_n$, then X is unitary if X is a zero of the function $F(\widetilde{X})$.

Suppose Y is an approximate solution to the equation $F(\widetilde{X}) = 0$, we can write $\widetilde{X} = Y + E$ where E is a "small perturbation" and substitute $\widetilde{X}$ into the equation $F(\widetilde{X}) = 0$

$$(Y+E)^*(Y+E) - I_n = Y^*Y + Y^*E + E^*Y + E^*E - I_n = 0.$$

Dropping the second order term, we get Newton iteration $Y^*Y + Y^*E + E^*Y - I_n = 0$ and this is the Sylvester equation for E [7, 2008, Problem 8.18]. We aim to solve E and update Y by $Y + E$, which gives the Newton iteration at the kth step:

$$\begin{cases} X_k^*X_k + X_k^*E_k + E_k^*X_k - I_n = 0, \\ X_{k+1} = X_k + E_k. \end{cases} \tag{3.1}$$

Assume that $X_k^*E_k$ is Hermitian, namely $X_k^*E_k = E_k^*X_k$, then we can further simplify the iteration by rewritten (3.1) as $X_k^*E_k = (I_n - X_k^*X_k)/2$ and the expression we got for $X_k^*E_k$ is indeed Hermitian, hence our assumption is valid. Therefore, we have a explicit expression for E_k, $E_k = (X_k^{-*} - X_k)/2$, and the iteration (3.1) becomes

$$X_{k+1} = X_k + \frac{1}{2}(X_k^{-*} - X_k) = \frac{1}{2}(X_k^{-*} + X_k)$$

choosing $X_0 = A$, we have our desired Newton iteration:

$$X_{k+1} = \frac{1}{2}(X_k^{-*} + X_k), \quad X_0 = A. \tag{3.2}$$

3.1 Convergence of the Newton's Method

Before looking into the convergence, an important property for the matrix 2-norm and the Frobenius norm should be presented.

Theorem 3.1. *The Frobenius norm and matrix 2-norm are the unitarily invariant norm. Namely, the following norm equality holds for these two norms*

$$\|UAV\| = \|A\|, \quad A \in \mathbb{C}^{n\times n},\ U \text{ and } V \text{ are unitary.}$$

Proof. By the definition of the Frobenius norm, we have

$$\|UAV\|_F^2 = \text{trace}\Big(UAVV^*A^*U^*\Big) = \text{trace}\Big(UAA^*U^*\Big). \tag{3.3}$$

By the properties $\text{trace}(AB) = \text{trace}(BA)$, (3.3) simplifies to

$$\|UAV\|_F^2 = \text{trace}\Big(A^*U^*UA\Big) = \text{trace}\Big(A^*A\Big) = \text{trace}\Big(AA^*\Big) = \|A\|_F^2. \tag{3.4}$$

Hence we finished the proof for the Frobenius norm.

By the definition of the matrix 2-norm, we have

$$\|UAV\|_2^2 = \rho(V^*A^*U^TUAV) = \rho(V^TA^*AV).$$

Notice that if V is unitary, then A^*A and V^*A^*AV are unitarily similar and share the same eigenvalues. Therefore $\rho(V^*A^*AV) = \rho(A^*A)$, therefore we have

$$\|UAV\|_2^2 = \rho(A^*A) = \|A\|_2^2.$$

Hence we proved the theorem. □

Theorem 3.2 (Convergence of the iteration (3.2), [7, 2008, Theorem 8.12])**.** *Let $A \in \mathbb{C}^{n\times n}$ be nonsingular. Then the Newton iterates X_k in (3.2) converges quadratically to the unitary polar factor U of A with*

$$\|X_{k+1} - U\| \le \frac{1}{2}\|X_k^{-1}\|\|X_k - U\|^2,$$

where $\|\cdot\|$ is any unitarily invariant norm.

Proof. Suppose A has a SVD $P\Sigma V^*$, where P and V are unitary, then by Theorem 2.1, the unitary polar factor is $U = PV^*$. To prove Theorem 3.2, the trick is to define a new sequence D_k where

$$D_k = P^*X_kV. \tag{3.5}$$

From iteration (3.2), we have

$$\begin{aligned} D_0 &= P^*X_0V = P^*AV = \Sigma, \\ D_{k+1} &= P^*X_{k+1}V \\ &= P^*\left(\frac{1}{2}(X_k^{-*} + X_k)\right)V \\ &= \frac{1}{2}\left(P^*X_k^{-*}V + P^*X_kV\right). \end{aligned} \tag{3.6}$$

Notice that, $P^*X_k^{-*}V = (P^*X_kV)^{-*}$, hence the iteration (3.6) becomes

$$D_{k+1} = \frac{1}{2}\left(D_k^{-*} + D_k\right), \quad D_0 = \Sigma. \tag{3.7}$$

Since A is full rank, D_0 is diagonal with positive diagonal entries, and therefore D_{k+1} is always defined since the determinant of D_{k+1} can never reduce to zero. The iteration will produce a sequence of diagonal matrices $\{D_k\}$ where we can write $D_k = \text{diag}(d_i^{(k)}), d_i^{(k)} > 0$ for $i = 1, 2, \ldots, n$. Notice that, we can use the same method as in the proof of Theorem 2.3 which decomposes (3.7) into

$$d_i^{(0)} = \sigma_i, \quad d_i^{(k+1)} = \frac{1}{2}\left(d_i^{(k)} + \frac{1}{d_i^{(k)}}\right), \quad \text{for } i = 1, 2, \ldots, n. \tag{3.8}$$

By an argument discussed in [5, 1964, pp. 84], we can write (3.8) as

$$d_i^{(k+1)} - 1 = \frac{1}{2d_i^{(k)}}\left(d_i^{(k)2} - 2d_i^{(k)} + 1\right) = \frac{1}{2d_i^{(k)}}\left(d_i^{(k)} - 1\right)^2. \tag{3.9}$$

This implies that the diagonal entries of D_k converge to 1 quadratically as $k \to \infty$ and this holds for all $i = 1, 2, \ldots, n$, therefore we can conclude that $\lim_{k\to\infty} D_k = I_n$ quadratically. Use the definition of D_k, (3.5), we have

$$\lim_{k\to\infty} X_k = \lim_{k\to\infty} PD_kV^* = PI_nV^* = PV^* = U,$$

where U is the unitary polar factor of the polar decomposition of A, and this convergence is quadratic.

Moreover, we can rewrite (3.9) in matrix form

$$\begin{bmatrix} d_1^{(k+1)} - 1 & & \\ & \ddots & \\ & & d_n^{(k+1)} - 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1/d_1^{(k)} & & \\ & \ddots & \\ & & 1/d_n^{(k)} \end{bmatrix}\begin{bmatrix} d_1^{(k)} - 1 & & \\ & \ddots & \\ & & d_n^{(k)} - 1 \end{bmatrix}^2$$

which is equivalent as

$$D_{k+1} - I_n = \frac{1}{2}D_k^{-1}(D_k - I_n)^2.$$

Taking the norm on both sides, we have

$$\|D_{k+1} - I_n\| = \frac{1}{2}\|D_k^{-1}(D_k - I_n)^2\| \le \frac{1}{2}\|D_k^{-1}\|\|D_k - I\|^2. \tag{3.10}$$

The final inequality comes from the fact that every unitarily invariant norm is subordinate [3, 1997, Proposition IV 2.4]. Also, the unitarily invariance implies the following three equalities,

- $\|D_{k+1} - I\| = \|PD_{k+1}V^* - PV^*\| = \|X_{k+1} - U\|$.
- $\|D_k^{-1}\| = \|VD_k^{-1}P^*\| = \|X_k^{-1}\|$.
- Using the same argument as 1, we have $\|D_k - I\| = \|X_k - U\|$.

As a result, (3.10) can be rewritten as

$$\|X_{k+1} - U\| \le \frac{1}{2}\|X_k^{-1}\|\|X_k - U\|^2.$$

Hence we proved the quadratic convergence of the Newton iteration (3.2). □

4 Newton–Schulz Iteration

The Newton iteration

$$X_{k+1} = \frac{1}{2}(X_k^{-*} + X_k), \quad X_0 = A,$$

requires the explicit form of a matrix inverse, and this can be avoided by using a one step of the Schulz iteration $Y_{k+1} = Y_k(2I_n - AY_k)$ [2, 1965] where Y_0 is an approximation to A^{-1}, and this iteration will converge to A^{-1}. To approximate the inverse of X_k^*, we have the one step of the Schulz iteration

$$X_k^{-*} \approx X_k(2I_n - X_k^* X_k).$$

This approach takes advantage of the fact that since the X_k converges quadratically, it is an increasingly good approximation to X_k^{-*}. Substituting this expression into the Newton iteration, we have the Newton–Schulz iteration [7, 2008, Section 8.3]

$$X_{k+1} = \frac{1}{2} X_k \left(3I - X_k^* X_k\right), \quad X_0 = A, \tag{4.1}$$

and this iteration will converge to the unitary polar factor of A.

4.1 Convergence of the Newton–Schulz Iteration

The convergence of the Newton–Schulz iteration can be proved using the similar approach as proof of Theorem 3.2. Let $P\Sigma V^*$ be the SVD of A and define $D_k = P^* X_k V$. It is obvious that $D_0 = \Sigma$ and

$$\begin{aligned} D_{k+1} &= P^* X_{k+1} V \\ &= \frac{1}{2} P^* \left(X_k \left(3I - X_k^* X_k\right)\right) V. \\ &= \frac{1}{2}(3P^* X_k V - P^* X_k X_k^* X_k V) \\ &= \frac{1}{2}(3P^* X_k V - P^* X_k V V^* X_k^* P P^* X_k V) \end{aligned}$$

From $D_k = P^* X_k V$, we have $D_k^* = V^* X_k^* P$ and then we can rewrite D_{k+1} as

$$D_{k+1} = \frac{1}{2}(3D_k - D_k D_k^* D_k). \tag{4.2}$$

Notice that $D_0 = \Sigma$ is a diagonal matrix, therefore it is obvious that all D_k for $k = 1, 2, \ldots$ are diagonal matrices. Therefore, (4.2) becomes

$$D_{k+1} = \frac{1}{2} D_k(3I_n - D_k^2), \quad D_0 = \Sigma. \tag{4.3}$$

Let $D_k = \operatorname{diag}\left(d_i^{(k)}\right)$ where $i = 1, 2, \ldots, n$. The iteration (4.3) can be written in an element-wise manner

$$d_i^{(k+1)} = \frac{1}{2} d_i^{(k)} \left(3 - \left(d_i^{(k)}\right)^2\right), \quad d_i^{(0)} = \sigma_i, \quad i = 1, 2, \ldots, n, \tag{4.4}$$

where σ_i is the ith singular value of A.

Lemma 4.1. *If $\sigma_i \in (0, \sqrt{3})$, then the iteration (4.4) converges quadratically to 1.*

Proof. If $\sigma_i = 1$, then the iteration converges immediately. To simplify the notation, we rewrite the iteration (4.4) as

$$d_{k+1} = f(d_k) = \frac{1}{2} d_k (3 - d_k^2). \tag{4.5}$$

Case 1: If $d_k \in (0, 1)$, then $(3 - d_k^2)/2 > 1$ which gives $d_k(3 - d_k^2)/2 > d_k$. Differentiate $f(d_k)$ once we have $f'(d_k) = -d_k^2 + (3 - d_k^2)/2$ and this will always be positive for $d_k \in (0, 1)$. Therefore, if we start with $d_k \in (0, 1)$, then $d_k < f(d_k) = d_{k+1} < f(1) = 1$, namely, start from $d_0 \in (0, 1)$, then the iteration (4.5) will converges to 1.

Case 2: If $d_k \in (1, \sqrt{3})$, then by similar approach we have the following

$$f(1) = 1, \quad f(\sqrt{3}) = 0, \quad f'(d_k) < 1 \quad \text{for } d_k \in (1, \sqrt{3}).$$

Therefore, the function f will monotonically decreasing from 1 to 0 within the interval $(1, \sqrt{3})$ and $\sup_{d_k \in (1, \sqrt{3})} f(d_k) = 1$. Hence, if $d_k \in (0, \sqrt{3})$, then $f(d_k)$ will belongs to $(0, 1)$, then the iteration (4.5) converges to 1 as discussed in case 1.

The properties of the function $f(d_k)$ in both case 1 and case 2 can be seen from the Figure 1 which shows that for $d_k \in (0, 1)$, the function f monotonically increasing from 0 to 1 and for $d_k \in (1, \sqrt{3})$, the function f monotonically decreasing from 1 to 0.

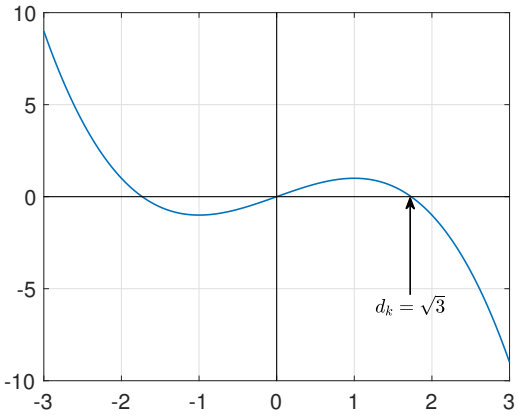

FIG. 1: *The plot of $f(d_k)$ defined in (4.5) over $[-3, 3]$. Three zeros of $f(d_k)$ are $d_k = -\sqrt{3}, 0$ and $\sqrt{3}$.*

Other parts of the real line may still converges to 1 but the interval of converges are too short to consider and control (for example, if we start with $d_0 = 2.2$, then $d_1 = f(d_0) = -2.0240$ and $d_2 = 1.1097 \in (0, \sqrt{3})$ which will converges to 1).

Remain to show that the convergence is of order 2. By consider the difference $d_{k+1} - 1$ (similar approach as (3.9)), we have

$$d_{k+1} - 1 = -\frac{1}{2}(d_k - 1)^2 (d_k + 2) \tag{4.6}$$

which implies quadratic convergence. □

Theorem 4.2 (Convergence of the Newton–Schulz iteration (4.1)). *For $A \in \mathbb{C}^{n\times n}$ of rank n which has the polar decomposition $A = UH$, if $\sigma_i(A) \in (0, \sqrt{3})$ or equivalently $\|A\|_2 < \sqrt{3}$, then the Newton–Schulz iteration*

$$X_{k+1} = \frac{1}{2}X_k(3I - X_k^*X_k), \quad X_0 = A,$$

converges to the unitary factor U quadratically as $k \to \infty$ and

$$\|X_{k+1} - U\| \leq \frac{1}{2}\|X_k + 2U\|\|X_k - U\|^2 \tag{4.7}$$

holds for any unitarily invariant norm.

Proof. By Lemma 4.1, the diagonal entries of D_k converge to 1 quadratically and this is equivalent as D_k converges to I_n quadratically as $k \to \infty$. Using the definition of D_k, we conclude that X_k converges to U, the unitary polar factor of A, quadratically.

From (4.6) and the proof of Theorem 3.2, we have

$$\begin{aligned} \|D_{k+1} - I_n\| &= \frac{1}{2}\|(D_k - I_n)(D_k - I_n)(D_k + 2I_n)\| \\ &\leq \frac{1}{2}\|D_k - I_n\|^2\|D_k + 2I_n\|. \\ \Rightarrow \|X_{k+1} - U\| &\leq \frac{1}{2}\|X_k - U\|^2\|X_k + 2U\|. \end{aligned}$$

Hence we proved the quadratic convergence of the Newton-Schulz iteration. □

5 Operation Count

In this section, we will evaluate the operation count for one step of Newton's method and one step of the Newton–Schulz iteration. We measure the computational cost by the number of scalar multiplications, divisions and additions/subtractions. Each of these operations are called a *flop* [6, 2002, p. 3].

5.1 Operation Count for Newton's Method

Recall that the Newton's method is

$$X_{k+1} = \frac{1}{2}(X_k^{-*} + X_k), \quad X_0 = A.$$

Each step requires one matrix inversion, one scalar-matrix multiplication and one matrix-matrix addition. The final two operations cost n^2 flops respectively. To compute X_k^{-1}, we use the following routine

1. Compute the LU decomposition with pivoting $PX_k = LU$, where P is a permutation matrix, L is a lower triangular matrix and U is an upper triangular matrix.
2. Compute the inverse of U using a sequence of backward substitutions.
3. Solve the lower triangular system $XL = U^{-1}$ for X.

4. Compute the inverse $X_k^{-1} = XP$.

Step 1 requires $2/3n^3+O(n^2)$ flops, step 2 requires $1/3n^3+O(n^2)$ flops, step 3 requires n^3 flops, and step 4 requires no flops. Therefore, one step of Newton's method requires $2n^3 + O(n^2)$ flops.

5.2 Operation Count for Newton–Schulz Iteration

Recall that the Newton–Schulz iteration is

$$X_{k+1} = \frac{1}{2}X_k(3I - X_k^*X_k), \quad X_0 = A.$$

If we rewrite this iteration as

$$X_{k+1} = \frac{3}{2}X_k - \frac{1}{2}X_kX_k^*X_k, \quad X_0 = A,$$

then clearly, we require two matrix-matrix multiplications and one matrix-matrix subtraction per iteration. Since the latter one requires only $O(n^2)$ flops, the majority of the computational cost will be the matrix level multiplications. Notice that the product $X_k^*X_k$ is a symmetric positive definite matrix, hence we can exploit its symmetry during computation. Instead of $2n^3$, we only need

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=1}^{n} 2 = n^3 + O(n^2)$$

to compute $X_k^*X_k$. Together with the product $X_k(X_k^*X_k)$, the total cost of the Newton–Schulz iteration per step will be $3n^3 + O(n^2)$ flops.

5.3 Conclusion

Base on previous discussions, we conclude

1. For Newton's method, which involves matrix inversion, requires $2n^3 + O(n^2)$ flops per step while using the LU decomposition with pivoting.

2. For Newton–Schulz iteration, which involves matrix multiplications, requires $3n^3 + O(n^2)$ flops per step.

Therefore, ignoring operation counts, the matrix multiplication needs to be 1.5 faster than matrix inversion for Newton–Schulz to be faster than Newton.

6 Implementation

In this section, we will present an algorithm that computes the polar decomposition of A using a mixture of the Newton iteration and the Newton–Schulz iteration. The motivation is that matrix multiplication is very fast on high-performance computers. The outline will be the following: Given a nonsingular matrix $A \in \mathbb{C}^{n\times n}$, starts with the Newton iteration and switches to the Newton–Schulz iteration when the latter is guaranteed to converge.

This will produce the unitary polar factor U of A, and then H can be constructed by $H = U^*A$. In order to ensure the symmetry for H, we calculate H via

$$H = \frac{1}{2}\left(U^*A + A^*U\right).$$

We need to determine an appropriate condition for us to switch from Newton to Newton–Schulz. From Theorem 4.2, the Newton–Schulz iteration will converge if the input X_k satisfies $\|X_k\|_2 < \sqrt{3}$. However, this is rather expensive to compute, and we should view the convergence in the following alternative way.

Theorem 6.1. *For $A \in \mathbb{C}^{n\times n}$ of rank n which has the polar decomposition $A = UH$ and given the Newton–Schulz iteration*

$$X_{k+1} = \frac{1}{2}X_k(3I - X_k^*X_k), \quad X_0 = A.$$

If we define $R_k = I - X_k^*X_k$, then R_k satisfies

$$R_{k+1} = \frac{3}{4}R_k^2 + \frac{1}{4}R_k^3.$$

Proof. Write $R_{k+1} = I - X_{k+1}^*X_{k+1}$, then substitute $X_{k+1} = \frac{3}{2}X_k - \frac{1}{2}X_kX_k^*X_k$ into the expression of R_{k+1}. The rest are just brute-force substitutions and simplifications, and we will omit the rest. □

We can use R_{k+1} to measure the distance between the iterator X_k and the desired limit U based on the following lemma.

Lemma 6.2. *Let $A \in \mathbb{C}^{n\times n}$ have the polar decomposition $A = UH$. Then*

$$\frac{\|A^*A - I\|}{1 + \sigma_1(A)} \le \|A - U\| \le \frac{\|A^*A - I\|}{1 + \sigma_n(A)}, \tag{6.1}$$

for any unitarily invariant norm.

Proof. Let A have the SVD $A = P\Sigma V^*$. Then we have two equalities

- $A^*U = U^*A$.
- $\|A - U\| = \|\Sigma - I\|$.

Based on the first equality, we can write $A^*A - I = (A-U)^*(A+U)$, then take a unitarily invariant norm on both sides

$$\begin{aligned}\|A^*A - I\| &= \|(A-U)^*(A+U)\| \\ &\le \|A-U\|\|A+U\|_2 \quad [9,\ 1991,\ \text{Cor. } 3.5.10] \\ &= (1+\sigma_1(A))\|A-U\|,\end{aligned}$$

which gives the lower bound. For the upper bound, we first write $A + U = U(H + I)$, and then we have

$$A^*A - I = (A-U)^*U(H+I) \quad \Rightarrow \quad (A^*A - I)(H+I)^{-1} = (A-U)^*U.$$

Taking any unitarily invariant norm on both sides, we have

$$\begin{aligned}\|(A-U)^*U\| = \|A-U\| &= \|(A^*A-I)(H+I)^{-1}\| \\ &\leq \|A^*A-I\|\|(H+I)^{-1}\|_2 \\ &= \frac{\|A^*A-I\|}{1+\sigma_n(A)}.\end{aligned}$$

Hence we proved the inequality (6.1). □

This result shows that the two measures of orthonormality $\|A^*A-I\|$ and $\|A-U\|$ are essentially equivalent, hence the residual $\|X_k^*X_k-I\|$ of an iterate is essentially the same as the error $\|U-X_k\|$.

6.1 Switching Condition

Instead of switching from Newton iteration to Newton–Schulz iteration at the kth step where $\|X_k\|_2 < \sqrt{3}$, we can use Theorem 6.1 to characterise this absolute convergence. If $\|R_k\| < 1$, then

$$\|R_{k+1}\| \leq \frac{3}{4}\|R_k\|^2 + \frac{1}{4}\|R_k\|^3 < \frac{3}{4}\|R_k\| + \frac{1}{4}\|R_k\| = \|R_k\|,$$

which implies that the Newton–Schulz iteration will certainly converge. Therefore, we can compute R_k at each step, once $\|R_k\| \leq \theta < 1$, we can switch to Newton–Schulz. In [8, 1990, p. 652], it suggests, to ensure fast convergence, θ should not be too close to 1 and here we choose $\theta = 0.6$ which is used by [8].

Notice that since the 2-norm and the ∞-norm are equivalent, hence it is reasonable to use ∞-norm for the errors even though the infinity norm is not a unitarily invariant norm.

6.2 Termination Condition

Suppose η is some positive tolerance, a natural way of terminating an iteration is the following: Define the relative difference $\delta_{k+1} = \|X_{k+1}-X_k\|/\|X_{k+1}\|$ between two successive iterators, we stop the iteration once $\delta_{k+1} \leq \eta$. However, as suggested by [7, 2008, Sec. 4.9.2], in floating point arithmetic there is typically no guarantee that δ_k will reach the tolerance η. The solution is to add another criterion which terminates the iteration once the relative change has not decreased by a factor of at least 2.

In addition, as suggested by [7, 2008, p. 208], $\delta_{k+1} \leq \eta$ tends to terminate the iteration too late. Instead, we can use the following condition:

$$\|X_{k+1}-X_k\|_\infty \leq (2\eta)^{1/2}n^{1/2},$$

as suggested by [7, 2008, p. 208] and [10, 2003, App. C].

6.3 Algorithm

In summary, the computation of the polar decomposition of $A \in \mathbb{C}^{n\times n}$ can be summarized into the Algorithm 1.

Algorithm 1 Given $A \in \mathbb{C}^{n\times n}$ of rank n. This algorithm computes the polar decomposition $A = UH$ using both the Newton iteration and the Newton–Schulz iteration. η is a positive tolerance which is the machine epsilon at double precision by default.

```
1: Let X = A, switch = false
2: tol = √(2η)√n, maxiteration = 10^6
3: for i = 1 : maxiteration do
4:     Compute R_1 = ‖X*X − I‖_∞
5:     if R_k ≤ 0.6 then
6:         switch = true
7:     end if
8:     if switch then
9:         Store X → X_store
10:        Update X using Newton–Schulz iteration
11:        Compute δ_{i+1} = ‖X − X_store‖_∞/‖X‖_∞
12:        % termination condition
13:        if δ_{i+1} < tol OR (δ_{i+1} > δ_i/2 AND i ≠ 1) then
14:            Break
15:        end if
16:    else
17:        Store X → X_store
18:        Update X using Newton iteration
19:        Compute δ_{i+1} = ‖X − X_store‖_∞
20:    end if
21: end for
```

In line 2, we add a `maxiteration` to prevent the algorithm from going forever. At the beginning of the algorithm, the relative error between successive iteration can have $\delta_{i+1} \approx \delta_i$, which will stop the algorithm immediately. Instead we put the termination condition, line 13–15, inside the Newton–Schulz iteration. The algorithm can be implemented using MATLAB as shown in Appendix A.

7 Numerical Testings

7.1 Experiment Setup

To test the correctness of our code, we examine $\|A - UH\|_\infty/\|A\|_\infty$ and $\|U^*U - I\|_\infty$. In addition, I will compare our computed solution U with the reference solution U_{ref} which is computed at quadruple precision using Multiprecision Computing Toolbox [1]. Our experiments are running under the following environment:

- CPU : Intel(R) Core(TM) i5-9600KF CPU @ 3.70GHz
- GPU : NVIDIA GeForce RTX 2060
- MATLAB Version : 9.12.0.2039608 (R2022a) Update 5

[1] https://www.advanpix.com/

- Multiprecision Computing Toolbox : 4.9.0 Build 14753
- Random Number Generator : `rng(1)`

We will test our MATLAB code in Appendix A using the following matrices:

- `randn(n)` : An $\mathbb{R}^{n\times n}$ matrix containing pseudorandom values drawn from the standard normal distribution.
- `eye(8)` : The 8×8 identity matrix.
- `hilb(6)` : The 6×6 matrix with elements $1/(i+j-1)$, which is a famous example of a badly conditioned matrix.
- `magic(6)` : An 6×6 matrix constructed from the integers 1 through 36 with equal row, column, and diagonal sums.
- `hadamard(8)` : A Hadamard matrix of order 8, that is, a matrix H with elements $+1$ or -1 such that $H^*H = 8I$.

7.2 Results and Comments

After applying the Algorithm 1 on the above test matrices, we have the data shown in Table 1. The testing routine can be found in Appendix B, and we produce the figures contain how $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ and $\|X^*X - I\|_\infty$ change with respect to the iteration. The figures (except for `eye(8)`) can be found in Appendix D. We have the following comments on the results:

- `randn(n)`: The algorithm works perfect, and we indeed have a polar decomposition $A = UH$, and the unitary polar factor is unitary as expected.
- `eye(8)`: The algorithm converges after one iteration since the input A is the same as the expected result U, which are both I_8. The polar decomposition for I_8 is then $I_8 \cdot I_8$.
- `hilb(6)`: Hilbert matrix is badly conditioned as seen from the Table 1, $\kappa_\infty(H_6)$ is $O(10^7)$. Due to bad conditioning, although $\|H^*H - I\|_\infty$ can be small, but after one iteration, the matrix $X_1 = (H^{-*} + H)/2$ can have a large $\|X_1^*X_1 - I\|_\infty$ as shown in Figure 5 and the following output.

```
>> rng(1); H = hilb(6); norm(H'*H-eye(6),"inf")
ans =
   3.1932e+00
>> X = 0.5*(inv(H)'+H); norm(X'*X-eye(6),"inf")
ans =
   2.7423e+13
```

  Therefore, the Newton's method requires a lot of iterations in order to drag the quantity $\|X_k^*X_k - I\|_\infty$ back to 0.6.

- `magic(6)`: The magic square is a singular matrix, and therefore our algorithm is not applicable. Nevertheless, we can run our code on this matrix anyway. Notice that the quantity $\|A - UH\|_\infty / \|A\|_\infty$ is large compared to the machine epsilon of double precision because A is a badly conditioned matrix. The computed H is positive semidefinite as expected from Definition 2.1.

- `hadamard(8)`: The Hadamard matrix of order 8 is well conditioned, and the result is perfect. Since the input Hadamard matrix A has the property that $A^*A = 8I_8$, therefore by Theorem 2.1, the positive definite part has the form $H = (A^*A)^{1/2} = \sqrt{8}I \approx 2.8284I_8$. The computed result indeed satisfies this structure with

$$\|H_{\text{comp}} - H_{\text{expect}}\|_\infty \approx 8.8818 \times 10^{-16}.$$

TABLE 1: *Results for applying* `poldec.m` *in Appendix A to the test matrices.*

Matrix A	$\kappa_\infty(A)$	Iteration	$\frac{\|A - UH\|_\infty}{\|A\|_\infty}$	$\|U^*U - I\|_\infty$	$\|U - U_{\text{ref}}\|_\infty$
`randn(20)`	1.6827e+03	8	3.1315e-16	4.6783e-16	5.6639e-16
`randn(50)`	1.0280e+03	9	6.8817e-16	8.3942e-16	1.5430e-15
`randn(100)`	2.1124e+03	9	1.1056e-15	1.1314e-15	2.3256e-15
`eye(8)`	1	1	0	0	0
`hilb(6)`	2.9070e+07	28	1.3028e-16	2.2303e-16	1.1334e-16
`magic(6)`	9.9980e+17	58	4.4338e-03	4.2653e-16	3.3307e-16
`hadamard(8)`	8	7	2.4980e-16	3.0175e-16	3.8858e-16

8 Further Application

We can utilize the polar decomposition to compute the square root of a symmetric positive definite matrix. Given a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$. We can compute the Cholesky factor R of A such that $A = R^TR$. Then apply Algorithm 1 to R, and we have $R = UH$ where U is unitary (in this case, orthogonal) and H is symmetric positive definite. Substitute back into the Cholesky decomposition of A, we have

$$A = R^TR = (UH)^T(UH) = H^TU^TUH = H^2.$$

Therefore, the symmetric positive definite part of the Cholesky factor of A is the square root of A for any symmetric positive definite matrix A.

The implementation is straightforward using MATLAB builtin function `chol` which computes the Cholesky factor, and the function `polsqrt.m` can be found in Appendix C. A simple test is carried out below:

```
>> rng(1); A = gallery('randsvd',50,-100);
>> H = polsqrt(A); disp(norm(H*H - A));
    2.9638e-16
```

The result shows that our code is valid. Here `gallery('randsvd',50,-100)` generates a symmetric positive definite matrix $A \in \mathbb{R}^{50\times 50}$ with fixed condition number 100.

References

[1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, Third ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. https://doi.org/10.1137/1.9780898719604. (Cited on page 4.)

[2] A. Ben-Israel, An iterative method for computing the generalized inverse of an arbitrary matrix, *Mathematics of Computation* **19** no. 91 (1965), 452–455. https://doi.org/10.1090/s0025-5718-1965-0179915-5. (Cited on page 8.)

[3] R. Bhatia, *Matrix Analysis*, Springer, New York, 1997. (Cited on page 7.)

[4] Å. Björck, *Numerical Methods for Least Squares Problems*, Society for Industrial and Applied Mathematics, 1996. https://doi.org/10.1137/1.9781611971484. (Cited on page 4.)

[5] P. Henrici, *Elements of Numerical Analysis*, John Wiley, New York, 1964. (Cited on page 7.)

[6] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, second ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. https://doi.org/10.1137/1.9780898718027. (Cited on page 10.)

[7] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. https://doi.org/10.1137/1.9780898717778. (Cited on pages 3, 4, 5, 6, 8, and 13.)

[8] N. J. Higham and R. S. Schreiber, Fast polar decomposition of an arbitrary matrix, *SIAM J. Sci. Statist. Comput.* **11** no. 4 (1990), 648–655. https://doi.org/10.1137/0911038. (Cited on page 13.)

[9] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, 1991. https://doi.org/10.1017/CBO9780511840371. (Cited on page 12.)

[10] A. Kiełbasiński, Numerical behaviour of higham's scaled method for polar decomposition, *Numerical Algorithms* **32** no. 2/4 (2003), 105–140. https://doi.org/10.1023/a:1024098014869. (Cited on page 13.)

A Code for Algorithm 1

```
function [X, H, its, delta, Rk] = poldec(A, TOL)
%POLDEC Polar decomposition.
% [U, H, ITS, DELTA, RK] = poldec(A) computes the polar decomposition
% A = U*H of the square, nonsingular matrix A. ITS is the number of
% iterations for convergence. DELTA is the relative error between
% successive iterations. RK is the norm(X'X - I) at each iteration.
[n,m] = size(A);
if n ~= m
    error('Input matrix is not square!');
end
if nargin == 2
    err = TOL;
else
    err = eps("double");
end
I = eye(n); term_tol = sqrt(2*err)*sqrt(n); maxiter = 1e6;
X = A; switched = false;
delta = zeros(maxiter,1); Rk = zeros(maxiter,1);
for its = 1:1:maxiter
    % check if switch to Newton Schulz
    Rk(its) = norm(X'*X - I,"inf");
    if Rk(its) <= 0.6
        switched = true;
    end
    % apply appropriate iterations
    if switched % Newton Schulz iteration
        tmp = X;
        X = 1.5*X - 0.5*X*(X'*X);
        delta(its+1) = norm(X - tmp,"inf")/norm(X,"inf"); % store error
        % termination condition
        if delta(its+1) < term_tol || (delta(its+1) > delta(its)/(2) &&
    its ~= 1)
            break;
        end
    else % Newton iteration
        tmp = X;
        X = 0.5*(inv(X)' + X);
        delta(its+1) = norm(X - tmp,"inf")/norm(X,"inf"); % store error
    end
end
H = 0.5*(X'*A + A'*X); % construct H
end
```

B Testing Routine

```
% testing routine
clc; clear; close all; format short e
rng(1);
print = true;
A = hadamard(8); A_mp = mp(A, 34);
term_tol = sqrt(2*eps("double"))*sqrt(size(A,1));
[U_d,H_d,its_d,del_d,Rk_d] = poldec(A);
[U_q,H_q,its_q,del_q,Rk_q] = poldec(A,5e-34);
% printing results
fprintf("Condition number = %.4d\n", cond(A,"inf"));
fprintf("Iteration required = %d\n", its_d);
tmp1 = norm(A - U_d*H_d, "inf")/norm(A,"inf");
fprintf("norm(A-UH)/norm(A) = %.4d\n", tmp1);
tmp2 = norm(U_d'*U_d - eye(size(A,1)));
fprintf("norm(U'*U-I) = %.4d\n", tmp2);
tmp3 = norm(U_d - U_q,"inf");
fprintf("norm(U-U_ref) = %.4d\n", tmp3);

if print
    figure('Renderer', 'painters', 'Position', [200 200 900 900])
    semilogy(1:its_d, del_d(2:its_d+1),'-^k', 'LineWidth', 2);
    hold on;
    semilogy(1:its_d, Rk_d(1:its_d),'-^r', 'LineWidth', 2);
    semilogy(1:its_d, ones(1,its_d).*term_tol, '--b', 'LineWidth', 2);
    legend("$\|X_{k+1}-X_{k}\|_{\infty}/\|X_{k+1}\|_{\infty}$", ...
        "$\|X_k^*X_k - I\|_{\infty}$", ...
        "Terminate tolerance", ...
        "Location","southwest", ...
        "Interpreter", "latex");
    xlabel("Iteration");
    axis square;
    set(gca, "FontSize", 30)
end
```

C Routine for Matrix Square Root

```
function H = polsqrt(A)
%POLSQRT Matrix square root using polar decomposition.
% H = poldec(A) computes the matrix square root of the symmetric
   positive
% definite matrix A, such that norm(H^2 - A) is small.
R = chol(A);
[~,H] = poldec(R);
end
```

D Figures

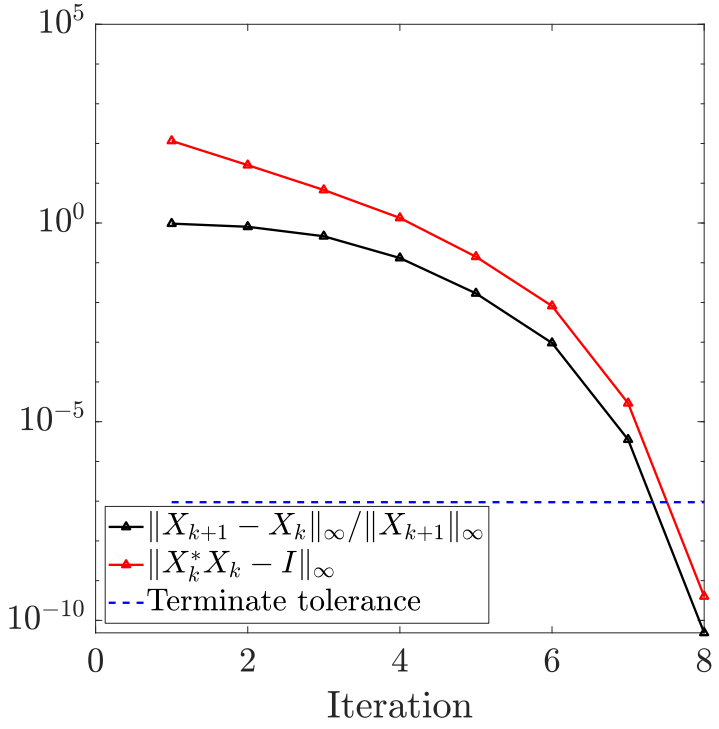

FIG. 2: *Behaviour of* $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ *and* $\|X^*X - I\|_\infty$ *with respect to the iteration when applying* `poldec` *on* `randn(20)`.

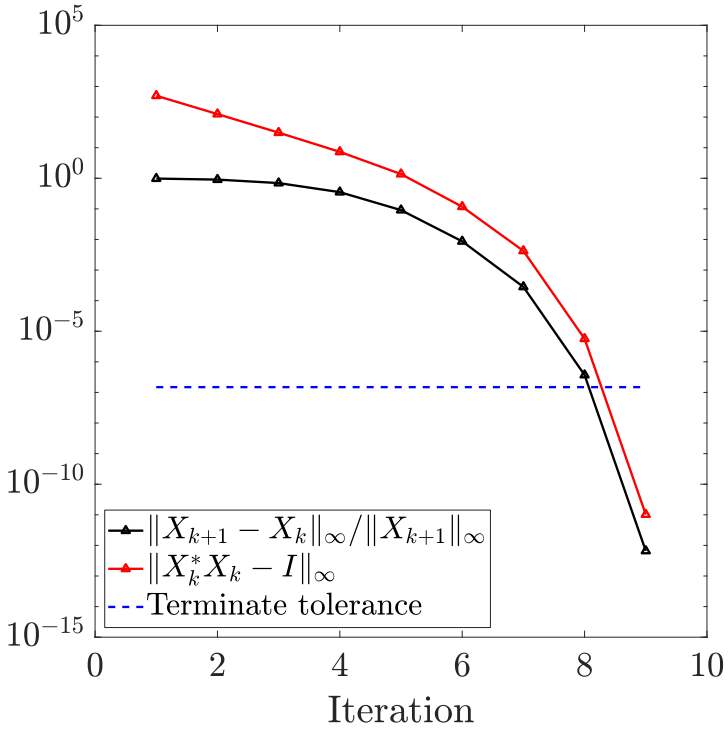

FIG. 3: *Behaviour of* $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ *and* $\|X^*X - I\|_\infty$ *with respect to the iteration when applying* `poldec` *on* `randn(50)`.

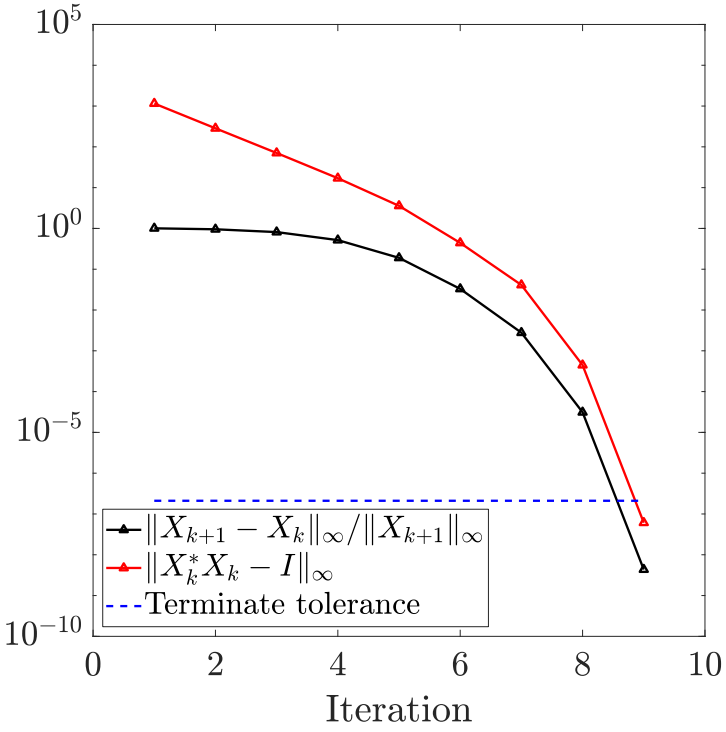

FIG. 4: *Behaviour of* $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ *and* $\|X^*X - I\|_\infty$ *with respect to the iteration when applying* `poldec` *on* `randn(100)`.

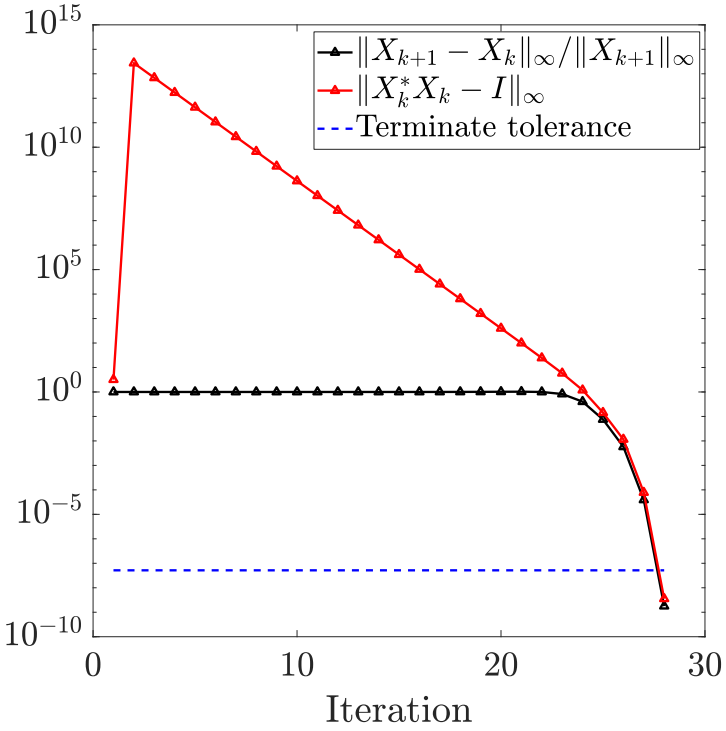

FIG. 5: *Behaviour of* $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ *and* $\|X^*X - I\|_\infty$ *with respect to the iteration when applying* `poldec` *on* `hilb(6)`.

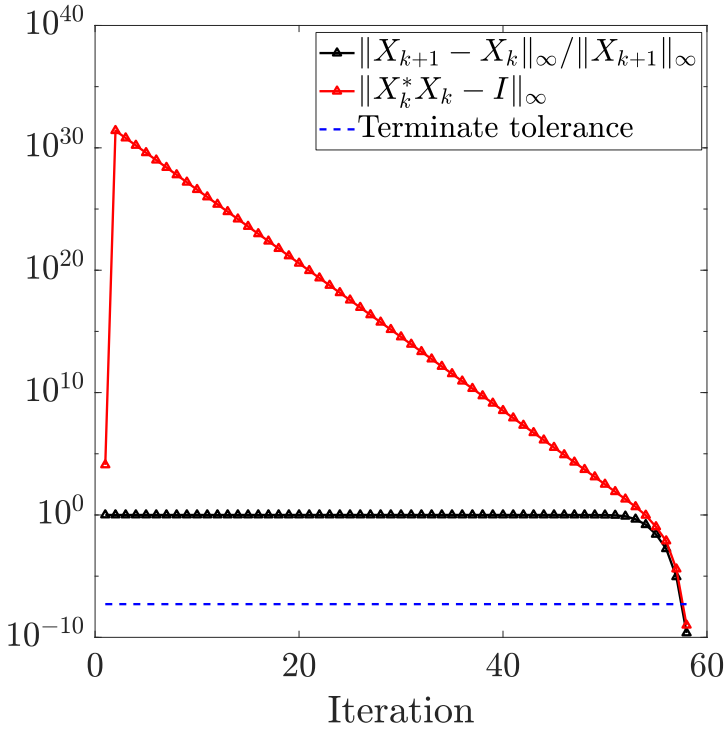

FIG. 6: *Behaviour of* $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ *and* $\|X^*X - I\|_\infty$ *with respect to the iteration when applying* `poldec` *on* `magic(6)`.

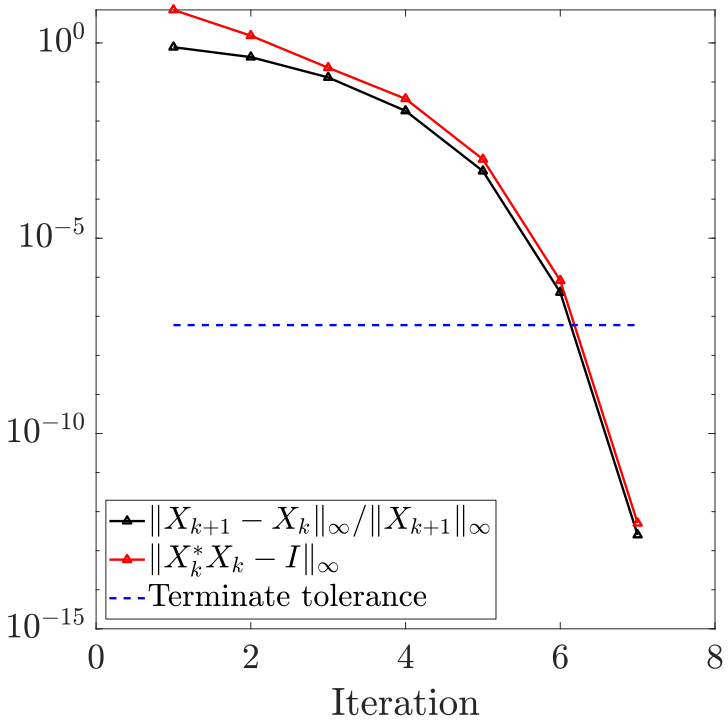

FIG. 7: *Behaviour of* $\|X_{k+1} - X_k\|_\infty / \|X_{k+1}\|_\infty$ *and* $\|X^*X - I\|_\infty$ *with respect to the iteration when applying* `poldec` *on* `hadamard(8)`.