ACCURACY OF THE MATRIX SQUARE ROOT

ZHENGBO ZHOU*

March 3, 2023

Contents

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

1 Introduction

This project is aiming to compare the accuracy of different approaches for computing the principal square root of a Hermitian positive definite matrix. Throughout this work, we will always consider the principal square root.

1.1 Outline

We can compute the principal square root of any Hermitian positive definite matrix via the following algorithm proposed by Higham in [14, 2008, Alg. 6.21].

Algorithm 1 Given a Hermitian positive definite matrix $A \in \mathbb{C}^{n\times n}$ this algorithm computes $H = A^{1/2}$.

1: Compute the Cholesky factorization $A = R^T R$.
2: Compute the Hermitian polar factor H of R by applying any method (exploiting the triangularity of R).

Since we are focused on the Hermitian positive definite matrices, the Schur decomposition of A and its eigendecomposition coincide. Therefore, we can simplify the Schur method, discussed in Section 2.2.1, and get

Algorithm 2 Given a Hermitian positive definite matrix $A \in \mathbb{C}^{n\times n}$ this algorithm computes $H = A^{1/2}$.

1: Compute the eigendecomposition $A = Q\Lambda Q^*$.
2: Compute the matrix square root $A^{1/2} = Q\Lambda^{1/2}Q^*$.

We would like to assess the accuracy of these two algorithms. Suppose we are solving $H^2 = A$, where $A, H \in \mathbb{C}^{n\times n}$, and are both Hermitian positive definite. Let $\widehat{H}_1$, $\widehat{H}_2$ be the computed square root through Algorithm 1 and 2, respectively. We will compare the relative forward and backward error defined by

$$\text{forward error} := \frac{\|H - \widehat{H}_i\|}{\|H\|}, \quad \text{backward error} := \frac{\|A - \widehat{H}_i^2\|}{\|A\|}, \qquad i \in \{1, 2\}. \tag{1}$$

Clearly, the error depends on how we compute the Hermitian polar factor and the eigendecomposition.

1. For Algorithm 1, we can compute the Hermitian polar factor of R using
   (a) Scaled Newton method [14, 2008, Alg. 8.20] or other iterative approach. This can make use of the existing code by Higham [10].
   (b) The SVD approach. First compute the SVD $A = U\Sigma V^*$, and then $H = V\Sigma V^*$.
2. For Algorithm 2, the eigendecomposition can be computed by the MATLAB `eig` command. More accurate method such as the DV-SVD algorithm [6, 7, 2008] will also be consider but the numerical experiments will require Fortran.

1.2 Reproducible Research

In this section, we will present our testing environment and system specifications. All the experiments are conducted under the following environment:

- MATLAB Version: 9.13.0.2126072 (R2022b) Update 3.
- Device: Macbook Pro 13-inch, M1, 2020 Model, 16GB RAM, macOS Version 13.2.1.
- The random number generator will be controlled by `rng(1)`.

1.3 Motivation

The motivation of this work is based on the following experiment. We will use the following functions,

- `[U,H,k] = polar_newton(A)`: computes the polar decomposition of A by scaled Newton iteration [10].
- `[U,H] = polar_svd(A)`: computes the polar decomposition of A via singular value decomposition [10].
- `X = sqrtm(A)`: computes the principal square root of the matrix A. This is a MATLAB built-in function[1].
- `X = sqrt_chol_polar(A,method)`: computes the Hermitian positive definite square root of a Hermitian positive definite matrix A and with two options `'newton'` and `'svd'` to determine using `polar_newton` or `polar_svd`.
- `A = gallery('randsvd',n,-kappa)`: generates a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ with $\kappa_2(A) =$ `kappa`.

The experiment will generate random Hermitian positive definite matrices X first, and computes $A = X^2$. Then, we apply both Algorithm 1 and 2 to A and obtain $\widehat{X}_1$ and $\widehat{X}_2$. Finally, we compute the forward and backward error using (1).

Both Figure 1 and 2 show that the backward error is about the same regardless of the conditioning of the problem. However, if we focus on the forward error, $\|\widehat{X}_i - X\|/\|X\|$, then for ill conditioned A, the Algorithm 1 will produce a square root that is significantly more accurate than the one computed by Algorithm 2.

[1] `sqrtm`: https://uk.mathworks.com/help/matlab/ref/sqrtm.html

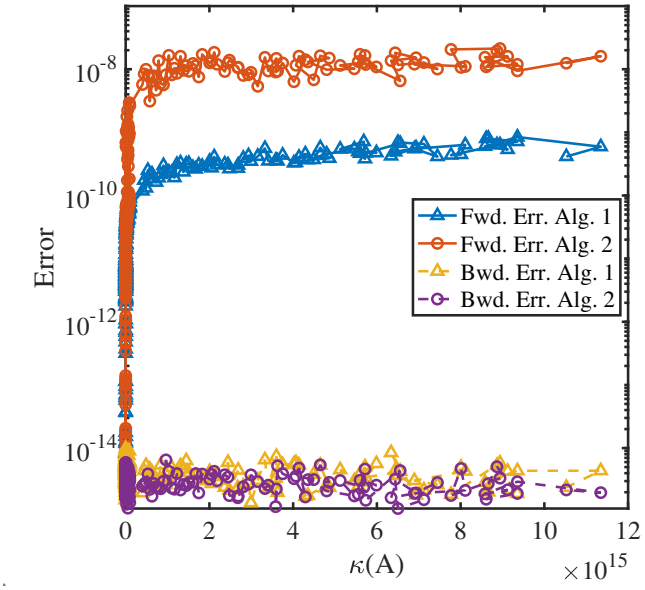

Figure 1: The forward errors and backward errors of Algorithm 1 and 2 apply on a random Hermitian positive definite matrix $A \in \mathbb{R}^{100\times 100}$ with different condition numbers. The Hermitian polar factor is computed via SVD approach. For more detail, please refer to Appendix A.1.

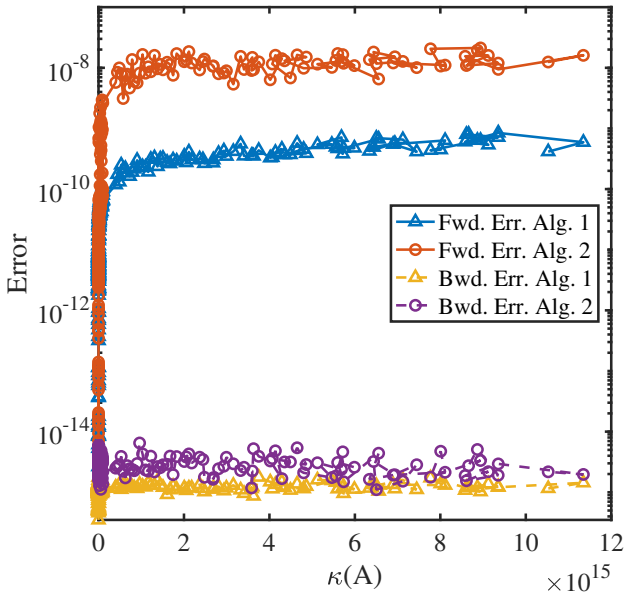

Figure 2: The forward errors and backward errors of Algorithm 1 and 2 apply on a random Hermitian positive definite matrix $A \in \mathbb{R}^{100\times 100}$ with different condition numbers. The Hermitian polar factor is computed via iterative approach. For more detail, please refer to Appendix A.1.

2 Background Information

In this section, we will provide a brief introduction on the matrix sign function, the matrix square roots and the polar decomposition. Especially on the iterations that associated with them. For more comprehensive analysis and results, see [14, 2008, Chap. 6 and 8].

In order to analyze the matrix square roots and the polar decomposition, we are need the singular value decomposition.

Theorem 2.1 (Singular value decomposition)**.** *If $A \in \mathbb{C}^{m\times n}$, $m \geq n$, then there exists two unitary matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ such that*

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

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

The number of nonzero singular values of A is the rank of A. However, the number of eigenvalues can be less than the rank of a matrix. For example, consider $A = \left[\begin{smallmatrix} 0 & 2 \\ 0 & 0 \end{smallmatrix}\right]$, then $\Lambda(A) = \{0, 0\}$ and $\Sigma(A) = \{2, 0\}$, whereas $\operatorname{rank}(A) = 1$.

The singular values of A is uniquely determined by the eigenvalues of A^*A, so the diagonal factor Σ is unique up to permutation of its diagonal entries.

2.1 Matrix Sign Function

Before introducing matrix square root and the polar decomposition, we need the matrix sign function. In this section, we will give a brief introduction on the matrix sign function. Several iterations for computing it will be presented. We will see, in Section 2.2 and Section 2.3, the matrix sign function plays an important role.

2.1.1 Definitions and Properties

The matrix sign function is an analogue of the scalar sign function defined for $z \in \mathbb{C}$

$$\operatorname{sign}(z) = \begin{cases} 1, & \operatorname{Re} z > 0, \\ -1, & \operatorname{Re} z < 0. \end{cases}$$

Notice that $\operatorname{sign}(z)$ is undefined when z lies on the imaginary axis. Therefore, throughout this section, $A \in \mathbb{C}^{n\times n}$ is assumed to have no eigenvalues on the imaginary axis, so that $\operatorname{sign}(A)$ is defined. Notice that, this condition restrict A to be nonsingular.

If $A = ZJZ^{-1}$ is a Jordan canonical form arranged such that $J = \operatorname{diag}(J_1, J_2)$, where the eigenvalues of $J_1 \in \mathbb{C}^{p\times p}$ lie in the open left half-plane and those of $J_2 \in \mathbb{C}^{q\times q}$ lie in the open right half-plane, then

$$\operatorname{sign}(A) = Z \begin{bmatrix} -I_p & 0 \\ 0 & I_q \end{bmatrix} Z^{-1}.$$

Clearly, A is commute with $\operatorname{sign}(A)$.

To get another expression of $\operatorname{sign}(A)$, consider the decomposition $A = SN$, where $S = \operatorname{sign}(A)$. Then $N^2 = S^{-1}AS^{-1}A = (SS)^{-1}(A^2) = A^2$. We denote this representation as the matrix sign decomposition [12, 1994, Sec. 2],

$$A = SN, \quad S = \operatorname{sign}(A), \quad N = (A^2)^{1/2}.$$

Hence, we have the concise formula

$$\operatorname{sign}(A) = A(A^2)^{-1/2},$$

where $(\cdot)^{1/2}$ denotes the principal square root, see Theorem 2.4.

The next result is useful for connecting the matrix sign function with the matrix square root and the polar decomposition.

Theorem 2.2 ([16, 2005, Lemma 4.3]). *Let $A, B \in \mathbb{C}^{n\times n}$ and suppose that AB has no eigenvalues on $\mathbb{R}^-$. Then*

$$\operatorname{sign}\left(\begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & C \\ C^{-1} & 0 \end{bmatrix},$$

where $C = A(BA)^{-1/2}$.

Proof. It is obvious that,†(Not so obvious, try to be more precise) AB share the same eigenvalues as BA [18, 2013, Thm. 1.3.22]. Then consider the matrix $P = \left[\begin{smallmatrix} 0 & A \\ B & 0 \end{smallmatrix}\right]$, and this matrix certainly will not have eigenvalues on the imaginary axis, otherwise $P^2 = \left[\begin{smallmatrix} AB & 0 \\ 0 & BA \end{smallmatrix}\right]$ will have eigenvalues on $\mathbb{R}^-$. Therefore $\operatorname{sign}(P)$ is defined and using the definition $\operatorname{sign}(A) = A(A^2)^{-1/2}$, we have

$$\begin{aligned} \operatorname{sign}\left(\begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}\right) &= \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}\left(\begin{bmatrix} AB & 0 \\ 0 & BA \end{bmatrix}\right)^{-1/2} \\ &= \begin{bmatrix} 0 & A(BA)^{-1/2} \\ B(AB)^{-1/2} & 0 \end{bmatrix} =: \begin{bmatrix} 0 & C \\ D & 0 \end{bmatrix} \end{aligned} \tag{2}$$

Moreover, we impose the fact that $(\operatorname{sign}(P))^2 = I$, gives $D = C^{-1}$. □

2.1.2 Newton's Method

The most widely used iteration for computing the sign function is the Newton iteration, which first mentioned by J. D. Roberts [20, 1980, Sec. 1.3] when he was solving the matrix Lyapunov equation and the algebraic Riccati equation. The Newton iteration for the Sign function can be obtained by applying the first order approximation to the function $F(X) = X^2 - I$. Let Y be an approximation to the solution $X^2 = I$, then define the correction term $E = S - Y$, where $S = \operatorname{sign}(A)$. Then $I = S^2 = (Y + E)^2 = Y^2 + EY + YE + E^2$. Since the Newton's method is a first order approximation method, therefore, we drop the second order term in E and give $I = Y^2 + EY + YE$. Assembling these, we have the Newton iteration for the matrix sign function,

$$\left.\begin{aligned} E_kY_k + Y_kE_k &= I - Y_k^2 \\ Y_{k+1} &= Y_k + E_k \end{aligned}\right\} \quad k = 0, 1, 2, \ldots, \quad Y_0 \text{ is given.} \tag{3}$$

However, this method does not have practical use, since we need to solve a Sylvester equation during each iteration which usually costs $O(n^3)$ where n is the dimension of Y_k [8, 1979, Sec. III]. Instead, we point out that $E_k = \frac{1}{2}(Y_k^{-1} - Y_k)$ is a solution to the Sylvester equation $E_k Y_k + Y_k E_k = I - Y_k^2$. Hence we have the Newton iteration for the matrix sign function

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \qquad X_0 = A. \tag{4}$$

The next theorem [14, 2008, Thm. 5.6] gives the convergence of the above iteration

Theorem 2.3 (convergence of the Newton iteration for matrix sign function)**.** *Let $A \in \mathbb{C}^{n\times n}$ have no pure imaginary eigenvalues. Then the Newton iterates X_k in (4) converge quadratically to $S = \operatorname{sign}(A)$, with*

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

for any consistent norm.

Proof. Consider an complex eigenvalue $\lambda = x + \mathrm{i}y$, then the iteration mapping will map this eigenvalue to

$$\lambda' = \lambda + \frac{1}{\lambda} = x + \mathrm{i}y + \frac{1}{x + \mathrm{i}y} = \left(x + \frac{x}{x^2 + y^2}\right) + \mathrm{i}\left(y - \frac{y}{x^2 + y^2}\right).$$

The mapped eigenvalue λ' will remain in the same half plane as λ and never become a pure imaginary eigenvalue as long as λ is not pure imaginary. Therefore, all X_k are well-defined and nonsingular.

Since both X_k and S are functions of A, X_k and S are commute by [14, 2008, Thm. 1.13]. Furthermore, we can consider the difference

$$\begin{aligned} X_{k+1} - S &= \frac{1}{2}(X_k + X_k^{-1} - 2S) \\ &= \frac{1}{2}X_k^{-1}(X_k^2 - 2X_kS + I) \\ &= \frac{1}{2}X_k^{-1}(X_k - S)^2 \quad \text{by commutativity.} \end{aligned}$$

By taking any consistent norm and using $\|AB\| \le \|A\|\|B\|$, we have the desired norm inequality and this inequality displays the quadratic convergence. □

2.1.3 Other Iterations

One straightforward variant of the Newton iteration (4) is the Newton–Schulz iteration. By considering removing the inversion in the Newton iteration, one can utilize the one-step Schulz iteration [24, 1933], $X_{k+1} = X_k(2I - AX_k)$, to estimate the inversion each step. The benefit of using Schulz iteration is that, as we iterates X_k from A towards $\operatorname{sign}(A)$, X_k is an increasingly good approximation to its inverse. Hence, to estimate X_k^{-1}, we use $X_k^{-1} \approx X_k(2I - X_kX_k)$, and we got the Newton–Schulz iteration for the matrix sign function

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

This iteration is multiplication-rich and keeps the quadratic convergence of Newton's method. However, it is only locally convergent for $\|I - A^2\| < 1$. The convergence result for the Newton–Schulz iteration was analyzed in terms of Padé iteration by Kenney and Laub [19, 1991, Thm. 5.3]. †(Any other convergence proof except the Padé approximation?)

This "approximate matrix inverse using Schulz iteration" approach was first introduced by Schreiber and Parlett [23, 1988, Sec. 3.2.1]. Higham and Schreiber [17, 1990] employed just one Schulz iteration for the polar decomposition and shown its convergence. Later, Kenney and Laub [19, 1991] shown that the Newton–Schulz is actually a Padé approximant.

For non-pure imaginary $z \in \mathbb{C}$, the $\operatorname{sign}(z)$ can be written as

$$\operatorname{sign}(z) = \frac{z}{(z^2)^{1/2}} = \frac{z}{(1-(1-z^2))^{1/2}} = \frac{z}{(1-\xi)^{1/2}},$$

where $\xi = 1 - z^2$. The task of approximating $\operatorname{sign}(z)$ leads to approximating $h(\xi) = (1-\xi)^{-1/2}$. The theory of the Padé approximants of $h(\xi)$ is based on results for hypergeometric functions [2, 1975, Chap. 5]. Kenny and Laub provided a family of rational iteration functions which are the $[k/m]$ Padé approximants to $h(\xi)$. These functions are rational functions P_{km}/Q_{km} where $\deg(P_{km}) = k$, $\deg(Q_{km}) = m$, and

$$h(\xi) - \frac{P_{km}(\xi)}{Q_{km}(\xi)} = O(\xi^{k+m+1}).$$

It turns out that the $[1/0]$ approximant gives the Newton–Schulz iteration and the $[1/1]$ approximant gives the Halley's method. Here, we will not discuss these in full detail, but one may refer to the original paper [19, 1991] or the book by Higham [14, 2008, Chap. 5.4].

2.1.4 Stability of Sign Iterations

For $A \in \mathbb{C}^{n\times n}$ that has no pure imaginary eigenvalues, by [14, 2008, Thm. 5.13], if the iteration function $X_{k+1} = g(X_k)$ is superlinearly convergent to $\operatorname{sign}(X_0)$ for all X_0 sufficient close to $\operatorname{sign}(A)$, and g is independent of X_0, then the iteration is stable in the sense that the Fréchet derivative of the matrix sign function at $\operatorname{sign}(A)$ has bounded powers.

However, the Newton iteration (4) can be numerically unstable is not always bounded by a modest multiple of the condition number $\kappa_{\text{sign}}(A)$[2], and this is shown by applying the Newton iteration to a matrix with eigenvalues close to the imaginary axis [14, 2008, pp. 126-127]. This instability was also pointed by Bai and Demmel [1, 1998, Sec. 2]. They observed that when the eigenvalues of A close to the pure-imaginary axis, the Newton iteration and its variations are very slowly convergent and may be mis-convergent.

We can conduct the similar test from [14, 2008, pp. 126-127]

```
clc; clear; close all; rng(1);
d = 1/3;
Q = rand_or(16); % generate random orth. mtx using qmult
T = triu(randn(16)); T_diag = abs(diag(T));
T_diag(1:8) = d * T_diag(1:8);
T_diag(9:16) = -1 * d * T_diag(9:16);
T = T - diag(diag(T)) + diag(T_diag);
A = Q * T * Q';
[X,k] = signm_newton(A);
fun = @(x) x/abs(x);
k_sign = funm_condest_fro(A,fun); % estimate the condition number
```

[2]The condition number is defined via [14, 2008, p. 110]

The functions `signm_newton` at line 9 and `funm_condest_fro` at line 11 are from [10]. With $d = 1/3$, the approximate condition number is $\kappa_{\text{sign}} \approx 2.1\text{e}8$. However, $\min_k \|S - X_k\|_\infty / \|S\|_\infty \approx 7.8\text{e-}2$, which greatly exceeds $\kappa_{\text{sign}} u \approx 2.4\text{e-}8$.

2.2 Matrix Square Roots

We will concerned with the principal square root [14, 2008, Thm. 1.29]. We denote by $\mathbb{R}^-$ the closed negative real axis.

Theorem 2.4 (principal square root). *Let $A \in \mathbb{C}^{n\times n}$ have no eigenvalues on $\mathbb{R}^-$. There is a unique square root X of A all of whose eigenvalues lie in the open right half-plane, and it is a primary matrix function of A. We refer to X as the principal square root of A and write $X = A^{1/2}$.*

2.2.1 Schur Method

Schur method, which initially presented by Björck and Hammarling [4, 1983] and extended to compute a real square root of a real matrix in real arithmetic by Higham [11, 1987], can computes all the primary square root of a given matrix provided its eigenvalues does not lie on $\mathbb{R}^-$.

Algorithm 3 (Schur method) Given a nonsingular $A \in \mathbb{C}^{n\times n}$ this algorithm computes $X = \sqrt{A}$ via a Schur decomposition, where $\sqrt{\cdot}$ denotes any primary square root.

1: Compute a (complex) Schur decomposition $A = QTQ^*$.
2: $u_{ii} = \sqrt{t_{ii}},\ i = 1\colon n$
3: **for** $j = 2\colon n$ **do**
4: **for** $i = j-1\colon -1\colon 1$ **do**
5: $$u_{ij} = \frac{t_{ij} - \sum_{k=i+1}^{j-1} u_{ik}u_{kj}}{u_{ii} + u_{jj}}$$
6: **end for**
7: **end for**
8: $X = QUQ^*$

The cost of this algorithm is $25n^3$ for the Schur decomposition [9, 2013, Alg. 7.5.2] plus $n^3/3$ for U and $3n^3$ to form X, which result in $28\frac{1}{3}n^3$ flops in total.

Following the language of [13, 2002, Sec. 3.1], if the computed diagonal elements satisfy $\widehat{u}_{ii} = \sqrt{t_{ii}}(1+\delta_i)$, where $|\delta_i| \le u$. Then, we have

$$\widehat{U}^2 = T + \Delta T, \qquad |\Delta T| \le \widetilde{\gamma}_n |\widehat{U}|^2,$$

where the inequality is to be interpreted elementwise. Computation of the Schur decomposition by the QR algorithm is backward stable process [9, 2013, Sec. 7.5.6], and we have

$$\widehat{X}^2 = A + \Delta A, \qquad \|\Delta A\|_F \le \widetilde{\gamma}_{n^3} \|\widehat{X}\|_F^2.$$

Using the expression $\alpha_F(\widehat{X}) = \|\widehat{X}\|_F^2 / \|A\|_F$, we can rewrite the expression as

$$\frac{\|A - \widehat{X}^2\|_F}{\|A\|_F} \le \widetilde{\gamma}_{n^3} \alpha_F(\widehat{X}).$$

This analysis confirms the result in [4, 1983], which also shown that if α is not large, then the computed square root is the exact square root of a matrix close to A.

A summary of the extension of the Schur method by Higham [11, 1987, Sec. 4.3] can be found in [14, 2008, Sec. 6.2]. Later in 2013, Deadman, Higham and Ralha [5, 2013] provided blocked Schur algorithms. MATLAB currently uses the Schur method with recursive blocking [15, 2020, Sec. 4] described in [5, 2013].

Notice that, for Hermitian matrices, the Schur decomposition becomes the eigendecomposition, and the Schur method becomes Algorithm 2.

2.2.2 Newton's Method

2.3 Polar Decomposition

Theorem 2.5 (Polar decomposition)**.** *Let $A \in \mathbb{C}^{n\times n}$ be nonsingular. There exist a unique unitary matrix $U \in \mathbb{C}^{n\times n}$, and a unique Hermitian positive definite matrix $H \in \mathbb{C}^{n\times n}$ such that $A = UH$. The matrix H is determined by*

$$(A^*A)^{1/2} = (H^*U^*UH)^{1/2} = (H^2)^{1/2} = H.$$

Moreover, U is uniquely determined by $U = AH^{-1}$.

Proof. Full proof for general $A \in \mathbb{C}^{m\times n}$ can be found in [14, 2008, Thm 8.1]. Let $A \in \mathbb{C}^{n\times n}$ be nonsingular, and $A = P\Sigma Q^*$. Then, $H = Q\Sigma Q^*$ and $U = PQ^*$. We can check H and U by

$$\begin{aligned} H^2 &= Q\Sigma\Sigma Q^* = Q\Sigma P^*P\Sigma Q = (P\Sigma Q^*)^*(P\Sigma Q) = A^*A, \\ U &= AH^{-1} = P\Sigma Q^*Q\Sigma^{-1}Q^* = PQ^*. \end{aligned}$$

The uniqueness of H is ensured by [14, 2008, Cor. 1.30]. □

We will refer to U as the unitary polar factor and H as the Hermitian polar factor. It is sufficient to only consider the square, nonsingular matrices for the polar decomposition. 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 $R \in \mathbb{C}^{n\times n}$. Then find the polar decomposition $R = UH$ and $A = QU \cdot H$ becomes the polar decomposition of A. In addition, if $A \in \mathbb{C}^{m\times n}$ is singular, we can compute a complete orthogonal decomposition

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

where P, Q are unitary, and $R \in \mathbb{C}^{r\times r}$ is nonsingular and upper triangular. For more details of this decomposition, you may refer to Björck [3, 1996, Def. 1.3.1]. Then, we may compute the polar decomposition of R and assemble them together, according to [17, 1990, Sec. 2],

$$\begin{aligned} A &= P \begin{bmatrix} UH & 0 \\ 0 & 0 \end{bmatrix} Q^* \\ &= P \begin{bmatrix} U & 0 \\ 0 & I_{m-r,n-r} \end{bmatrix} \begin{bmatrix} H & 0 \\ 0 & 0 \end{bmatrix} Q^* \\ &= P \begin{bmatrix} U & 0 \\ 0 & I_{m-r,n-r} \end{bmatrix} Q^*Q \begin{bmatrix} H & 0 \\ 0 & 0 \end{bmatrix} Q^* \\ &\equiv U_A H_A, \end{aligned}$$

where $I_{m-r,n-r}$ is the $(m-r) \times (n-r)$ identity matrix.[3]

The fundamental connection between the matrix sign function and the polar decomposition is obtained from Theorem 2.2 by setting $B = A^*$,

$$\operatorname{sign}\left(\begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & A(A^*A)^{-1/2} \\ (A^*A)^{1/2}A^{-1} & 0 \end{bmatrix} = \begin{bmatrix} 0 & U \\ U^* & 0 \end{bmatrix}.$$

[3]The identity matrix is $I_{m,n} = (\delta_{ij}) \in \mathbb{R}^{m \times n}$. [14, 2008, App. B.1]

3 LR Algorithm

3.1 Introduction

Theorem 3.1. *For $A = A^T \in \mathbb{R}^{n\times n}$, the following conditions are equivalent,*

(a) *A is positive definite, i.e. $x^TAx > 0$ for all nonzero $x \in \mathbb{R}^n$.*

(b) *There is a unique upper triangular matrix $R \in \mathbb{R}^{n\times n}$ with positive diagonal elements such that $A = R^TR$. This is called the Cholesky factorization of A.*

(c) *All the eigenvalues of A are positive.*

(d) $\det(A_k) > 0$, *$k = 1\colon n$ where A_k is the leading principal submatrix of order k.*

Proposition 3.2. *If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, then $a_{ii} > 0$ for all i.*

Proof. Taking $x = e_i$, where e_i is the ith column of the identity matrix. Then $e_i^TAe_i = a_{ii} > 0$. □

The LR algorithm is developed by Rutishauser when he was working on the quotient–difference algorithm. The algorithm can be summarized as follows, given $A \in \mathbb{C}^{n\times n}$, then let $A_0 = A$, and do the following iteration,

$$\left.\begin{aligned} A_k &= L_kU_k, \\ A_{k+1} &= U_kL_k, \end{aligned}\right\} \quad \text{for } k = 0, 1, \dots.$$

This has been shown by Rutishauser and Wilkinson that A_k converges to an upper triangular matrix where the diagonal entries are the eigenvalues of A. This can be seen from the following trick,

$$A_{k+1} = U_kL_k = L_k^{-1}L_kU_kL_k = L_k^{-1}A_kL_k.$$

We can do this process inductively, and finally we conclude that A_{k+1} is a similar transformation of A, where

$$A_{k+1} = (L_kL_{k-1}\cdots L_1)A(L_kL_{k-1}\cdots L_1)^{-1}.$$

3.2 Cholesky LR Algorithm

The algorithm in Section 3.1 can be modified for symmetric positive definite matrices by replacing the LU factorization by the Cholesky factorization.

Algorithm 4 Cholesky LR Algorithm.

1: $A_1 = A$
2: **for** $k = 1, 2, \dots$ until converge **do**
3: $\quad A_k = R_k^TR_k \quad$ *%Cholesky factorization*
4: $\quad A_{k+1} = R_kR_k^T$
5: **end for**

It can be seen, inductively, that

$$A_{k+1} = (R_k\cdots R_1)A(R_k\cdots R_1)^{-1}. \tag{5}$$

The next result gives the convergence of the Cholesky LR algorithm [22, 1990, Sec. 12.6].

Theorem 3.3. *If A is symmetric positive definite, then for all the matrices A_k that generated iteratively by Algorithm 4, one has*

$$\lim_{n\to\infty} A_k = \mathrm{diag}(\lambda_1, \ldots, \lambda_n),$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A.

Proof. Since

$$A_{k+1} = R_k R_k^T = R_k(R_k^T R_k)R_k^{-1} = R_k A_k R_k^{-1},$$

all matrices A_k are similar to one another. Therefore, *if* the limit matrix is diagonal matrix, then its diagonal elements are eigenvalues of A. Remain to prove that the limit matrix, $\lim_{k\to\infty} A_k$ is diagonal.

Let $s_\ell^{(k)}$ denotes the sum of ℓ first diagonal elements of A_k, i.e.

$$s_\ell^{(k)} = \sum_{i=1}^{\ell} a_{ii}^{(k)},$$

then certainly, by Proposition 3.2,

$$0 < s_1^{(k)} < s_2^{(k)} < \cdots < s_{n-1}^{(k)} < s_n^{(k)},$$

where $\mathrm{trace}(A) = s = s_n^{(k)}$ is independent of k, since the $\mathrm{trace}(A)$ is the summation of the eigenvalues of A. Using $A_k = R_k^T R_k$, we have

$$a_{jj}^{(k)} = \sum_{i=1}^{j} (r_{ij}^{(k)})^2,$$

while from $A_{k+1} = R_k R_k^T$, we have

$$a_{ii}^{(k+1)} = \sum_{j=i}^{n} (r_{ij}^{(k)})^2.$$

Summing over j and i from 1 to ℓ, respectively, we have

$$s_\ell^{(k)} = \sum_{j=1}^{\ell}\sum_{i=1}^{j} (r_{ij}^{(k)})^2 = \sum_{i=1}^{\ell}\sum_{j=i}^{\ell} (r_{ij}^{(k)})^2,$$

$$s_\ell^{(k+1)} = \sum_{i=1}^{\ell}\sum_{j=i}^{n} (r_{ij}^{(k)})^2,$$

and thus

$$s_\ell^{(k+1)} - s_\ell^{(k)} = \sum_{i=1}^{\ell}\sum_{j=\ell+1}^{n} (r_{ij}^{(k)})^2. \tag{6}$$

Thus, the sequence $\{s_\ell^{(k)}\}$ is monotone in k, more importantly, the sequence is bounded by $\mathrm{trace}(A)$, hence it must converge. By (6), we have

$$s_\ell^{(k+1)} - s_\ell^{(k)} = \sum_{i=1}^{\ell}\sum_{j=\ell+1}^{n} (r_{ij}^{(k)})^2 \to 0,$$

which implies $r_{ij}^{(k)} \to 0$ for $i = 1, \dots, \ell$ and $j = \ell + 1, \dots, n$. Since This is true for all ℓ, we have

$$\lim_{n\to\infty} r_{ij}^{(k)} = 0 \quad \text{for } i < j. \tag{7}$$

Moreover, consider the difference

$$s_\ell^{(k)} - s_{\ell-1}^{(k)} = a_\ell^{(k)} = \sum_{i=1}^{\ell} (r_{i\ell}^{(k)})^2.$$

As $k \to \infty$, for all $i < j$, $r_{ij} = 0$, therefore

$$\lim_{k\to\infty} (s_\ell^{(k)} - s_{\ell-1}^{(k)}) = \lim_{k\to\infty} (r_{\ell\ell}^{(k)})^2. \tag{8}$$

Hence, $\lim_{k\to\infty} R_k$ exist by (8) and is a diagonal matrix by (7). Finally,

$$\lim_{n\to\infty} A_k = \lim_{k\to\infty} R_k^T R_k$$

is clearly diagonal. □

3.3 Problem

I will now formulate a problem that I encountered. From the relationship between A_{k+1} and A,

$$A_{k+1} = (R_k \cdots R_1) A (R_k \cdots R_1)^{-1},$$

we have

$$A_1 = (R_1^{-1} \cdots R_k^{-1}) A_{k+1} (R_k \cdots R_1).$$

Then by Theorem 3.3, we take the limits at both sides and have

$$A_1 = T^{-1} \Lambda T, \qquad T = \lim_{k\to\infty} (R_k \cdots R_1), \quad \Lambda = \lim_{k\to\infty} A_{k+1}.$$

However, T, T^{-1} and Λ are all upper triangular, then why the resulting matrix is a full matrix? Since no matter how we taking the limit (i.e. adding $R_{k+1}, R_{k+2}, \dots$ to P), the matrix T will remain its triangular structure.

Moreover, since A is symmetric positive definite, then it is unitarily diagonalizable, $A = Q\Lambda Q^T$, where Q is unitary and Λ is a diagonal matrix with eigenvalues of A on its diagonal. On the other hands, from previous construction, we have $A = T\Delta T^{-1}$, where T is upper triangular. Therefore, we have

$$T\Delta T^{-1} = Q\Lambda Q^*.$$

We can permute the matrix Q such that $\Lambda = \Delta$, and

$$T\Delta T^{-1} = Q\Delta Q^*, \quad \Longrightarrow \quad \Delta = T^{-1} Q \Delta Q^* T.$$

This implies $T^{-1}Q = Q^*T = I$, and gives $T = Q$. *But does this told us that T will converge to a unitary matrix?*

3.4 Practical Consideration of Cholesky LR Algorithm

Purely using Algorithm 4 is not feasible to construct the eigendecomposition.

From Section 3.2, we see that all the A_k have the same eigenvalues since they are all similar. Moreover, the products,

$$M_k = R_k \cdots R_1, \quad \text{and} \quad N_k = R_1^T \cdots R_k^T$$

are the transformation matrices which transform A_1 into A_{k+1}, namely

$$A_{k+1} = M_k A_1 M_k^{-1} = N_k^{-1} A_1 N_k.$$

Using the rule $R_k^T R_k = R_{k-1} R_{k-1}^T$ from Algorithm 4, we have the expression

$$R_k^T R_k \cdots R_1 = R_{k-1} R_{k-2} \cdots R_3 R_2 R_1 R_1^T R_1,$$

and now, we form the product,

$$N_k M_k = R_1^T \cdots R_k^T R_k \cdots R_1 = (R_1^T R_1)^k. \tag{9}$$

Notice that, N_k and M_k are lower and upper triangular matrices respectively, and the diagonal entries of these matrices are positive. Therefore the kth transformation matrices are the Cholesky factors of A^k.

Due to the slow convergence of the Cholesky LR algorithm, M_k or N_k is impossible to form in practice via Cholesky factorization of A^k where k is the number of iterations. We will denote $\mathsf{off}(A)$ as

$$\mathsf{off}(A) = \sqrt{\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} a_{ij}^2}, \qquad A \in \mathbb{R}^{n\times n},$$

which is the Frobenius norm of the off diagonal entries of A. Here is a simple program that implements the Algorithm 4,

```
function [i,A] = chol_lr(A)
%CHOL_LR Cholesky LR iteration for diagonalization.
maxiter = 1e3; n = size(A,1); tol = eps/2;
for i = 1:maxiter
  R = chol(A); % A = R' * R;
  A = R * R';
  if off(A) <= tol * n
    break;
  end
end
end
```

We test the function using a 5×5 symmetric positive definite matrix A with $\kappa(A) = 5$ generated by `gallery('randsvd')` with different modes. In addition, we fixed the random number generator by `rng(1)`.

Table 1: Iterations required for `chol_lr` to converge on different test matrices.

Test matrix	Iterations required
`gallery("randsvd",5,-5,1)`	46
`gallery("randsvd",5,-5,2)`	45
`gallery("randsvd",5,-5,3)`	175

The results from Table 1 show that even for such a small and well-conditioned matrix, the algorithm requires about 50 iterations to converge. If we concentrated on mode 1, the transformation matrix will become the Cholesky factor of A^{46}. Let us denote the eigenvalues of A by $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A) > 0$. Consider the 2-norm condition number of A

$$\kappa_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)} = \frac{\lambda_1(A)}{\lambda_n(A)},$$

where the final equality is due to A being a symmetric positive definite matrix. Let us consider the eigenvalues of A^{46}. Let $A = Q\Lambda Q^*$ be its eigendecomposition, then

$$A^{46} = Q\Delta Q^*, \qquad \Delta = \begin{bmatrix} \lambda_1(A)^{46} & & & \\ & \lambda_2(A)^{46} & & \\ & & \ddots & \\ & & & \lambda_n(A)^{46} \end{bmatrix}.$$

Therefore, $\kappa(A^{46}) = (\lambda_1(A)/\lambda_n(A))^{46}$ will be extremely large if $\lambda_1(A)/\lambda_n(A) > 1$. This implies that the computed Cholesky factors of A^{46} can be very inaccurate due to its ill-conditioning. This is assured by our previous experiments: for the matrix generated by `gallery('randsvd',5,-5,1)`, the `chol_lr` requires 46 iterations to converge. Then, the condition number of A^{46} will be about 5.1e18. By calling `chol`, we observe that A^{46} lost its positive definiteness due to its high condition number.

An alternative way of computing the transformation matrices will be accumulate them during the iteration,

Algorithm 5 Cholesky LR algorithm for $A \in \mathbb{R}^{n\times n}$. The algorithm will accumulate upper triangular matrices T_1 and T_2 such that the output matrix can be written as $\widetilde{A} = T_1 A T_2$.

$A_1 = A, P = I$
for $k = 1, 2, \ldots,$ until converged **do**
  $A_k = R_k^T R_k$
  $T_1 = R_k T_1$
  $T_2 = T_2 R_k^{-1}$
  $A_{k+1} = R_k R_k^T$
  if $\mathsf{off}(A_k) \leq nu\|A_1\|$ **then**
    Break
  end if
end for
$\widetilde{A} = A_{k+1}$

If the iteration converged after k steps, we have

$$\widetilde{A} = T_1 A T_2, \qquad \widetilde{A} = A_{k+1}, \quad T_1 = R_k \cdots R_1, \quad T_2 = R_1^{-1} \cdots R_k^{-1}.$$

which is the same as (5).

We would like to obtain a lower bound for $\kappa_2(T_1)$. From (9), $T_1^T T_1 = A^k$, and consider

the condition number of A^k using any consistent norm,

$$\begin{aligned}\kappa(A^k) &= \|A^k\|\|(A^k)^{-1}\| \\ &= \|T_1^T T_1\|\|T_1^{-1}(T_1^{-1})^T\| \\ &\leq \|T_1\|\|T_1\|\|T_1^{-1}\|\|T_1^{-1}\| \\ &= \kappa(T_1)^2.\end{aligned}$$

This gives $\kappa(T_1) \geq \sqrt{\kappa(A^k)}$. From previous analysis, $\kappa(A^k)$ can be extremely large which may cause T_1 and T_2 become ill-conditioned.

We can conduct numerical experiments by applying Algorithm 5 to the test matrices in Table 1.

Table 2: Condition of accumulated triangular matrix T_1 from applying Algorithm 5 to different test matrices and the error $\|T_1 A T_2 - \widetilde{A}\|$.

Test Matrix	$\kappa_2(T_1)$	$\|T_1 A T_2 - \widetilde{A}\|$
`gallery("randsvd",5,-5,1)`	1.2e16	5.4e0
`gallery("randsvd",5,-5,2)`	5.3e15	1.7e0
`gallery("randsvd",5,-5,3)`	1.4e61	1.1e46

Table 2 presents the condition number for the accumulated T_1 and we observe that these are too ill-conditioned to work with. Moreover, we cannot guarantee the equality $\widetilde{A} = T_1 A T_2$ in practice due to the ill-conditioning of T_1.

We can do the same experiments using quadruple precision via the MATLAB package, Symbolic Math Toolbox. [4] The results are perfect for mode 1 and 2 in the sense that

$$T_1 T_2 = T_2 T_1 = I, \quad \widetilde{A} = T_1 A T_2.$$

However, for mode 3, due to this extreme ill-conditioned system, the quadruple precision does not improve the results. The experiment shows that after 70 iterations, T_1 is no longer acted as the inverse of T_2 in the sense that $\|T_1 T_2 - I\| \geq nu$.

To improve this algorithm and stick at using Cholesky factorization and the matrix multiplication, we should speed up the convergence by introducing shifts.

I don't think there is a way of cooperate Cholesky LR iteration and the Jacobi algorithm due to the Cholesky LR iteration is not using unitarily similar transformations. Moreover, the ill-conditioned triangular transformation matrices are also the problem. In the literature, both the original paper by Rutishauser [21, 1958] and his lecture notes [22, 1990] are suggesting that LR algorithm is used for determining the eigenvalues only.

[4] Simple examples of using quadruple precision in MATLAB can be found at https://nhigham.com/2017/08/31/how-fast-is-quadruple-precision-arithmetic/

A Supplementary Codes

A.1 Code for Figure 1 and 2

```
clear; close all; clc; rng(1);
addpath("~/Dropbox/matlab/mftoolbox/");
kappa = [1e2:1e2:1e3,2e3:1e3:1e4,2e4:1e4:1e5,...
  2e5:1e4:1e6,2e6:1e5:1e7,2e7:1e6:1e8];
for i = 1:length(kappa)
  fprintf("%d/%d\n", i, length(kappa));
  X = gallery("randsvd", 1e2, -1 * kappa(i)); % Symm. pd. sqrt X
  A = X * X';
  [X1] = sqrt_chol_polar(A,'svd'); % Polar
  [X2] = sqrtm(A); % Schur
  fwd_err1(i) = norm(X1 - X)/norm(X);    % use Alg. 1
  fwd_err2(i) = norm(X2 - X)/norm(X);    % use Alg. 2
  bwd_err1(i) = norm(X1^2 - A)/norm(A); % use Alg. 1
  bwd_err2(i) = norm(X2^2 - A)/norm(A); % use Alg. 2
  condi(i) = cond(A,2);
end

function X = sqrt_chol_polar(A,method)
R = chol(A);
switch method
  case "newton"
    [~,X,~] = polar_newton(R); % newton
  case "svd"
    [~,X] = polar_svd(R); % svd
end
end
```

References

[1] Zhaojun Bai and James Demmel. Using the matrix sign function to compute invariant subspaces. *SIAM Journal on Matrix Analysis and Applications*, 19(1):205–225, 1998. (Cited on p. 8.)

[2] George A. Baker. *Essentials of Padé approximants*. Academic Press, New York, 1975. xi+306 pp. ISBN 0-12-074855-X. (Cited on p. 8.)

[3] Åke Björck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, jan 1996. xvii+408 pp. ISBN 978-0-89871-360-2. (Cited on p. 10.)

[4] Åke Björck and Sven Hammarling. A Schur method for the square root of a matrix. *Linear Algebra and its Applications*, 52–53:127–140, 1983. (Cited on pp. 9 and 10.)

[5] Edvin Deadman, Nicholas J. Higham, and Rui Ralha. Blocked Schur algorithms for computing the matrix square root. In *Applied Parallel and Scientific Computing: 11th International Conference, PARA 2012, Helsinki, Finland*, P. Manninen and P. Öster, editors, volume 7782 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Germany, 2013, pages 171–182. (Cited on p. 10.)

[6] Zlatko Drmač and Krešimir Veselić. New fast and accurate Jacobi SVD algorithm. I. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1322–1342, 2008. (Cited on p. 2.)

[7] Zlatko Drmač and Krešimir Veselić. New fast and accurate Jacobi SVD algorithm. II. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1343–1362, 2008. (Cited on p. 2.)

[8] Gene H. Golub, Steve G. Nash, and Charles F. Van Loan. A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Transactions on Automatic Control*, 24(6):909–913, 1979. (Cited on p. 7.)

[9] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins studies in the mathematical sciences. 4th edition, Johns Hopkins University Press, Baltimore, MD, USA, 2013. ISBN 978-1-4214-0794-4. (Cited on p. 9.)

[10] Nicholas J. Higham. The Matrix Function Toolbox. `http://www.ma.man.ac.uk/~higham/mftoolbox`. (Cited on pp. 2, 3, and 9.)

[11] Nicholas J. Higham. Computing real square roots of a real matrix. *Linear Algebra and its Applications*, 88–89:405–430, 1987. (Cited on pp. 9 and 10.)

[12] Nicholas J. Higham. The matrix sign decomposition and its relation to the polar decomposition. *Linear Algebra and its Applications*, 212–213:3–20, 1994. (Cited on p. 6.)

[13] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2nd edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, January 2002. xxx+680 pp. ISBN 0-89871-521-0. (Cited on p. 9.)

[14] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-46-7. (Cited on pp. 2, 5, 7, 8, 9, 10, and 11.)

[15] Nicholas J. Higham and Edvin Hopkins. A catalogue of software for matrix functions. Version 3.0. MIMS EPrint 2020.7, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, March 2020. 24 pp. (Cited on p. 10.)

[16] Nicholas J. Higham, D. Steven Mackey, Niloufer Mackey, and Françoise Tisseur. Functions preserving matrix groups and iterations for the matrix square root. *SIAM Journal on Matrix Analysis and Applications*, 26(3):849–877, 2005. (Cited on p. 6.)

[17] Nicholas J. Higham and Robert S. Schreiber. Fast polar decomposition of an arbitrary matrix. *SIAM Journal on Scientific and Statistical Computing*, 11(4):648–655, 1990. (Cited on pp. 8 and 10.)

[18] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. 2nd edition, Cambridge University Press, Cambridge, UK, 2013. ISBN 9780521839402. (Cited on p. 6.)

[19] Charles Kenney and Alan J. Laub. Rational iterative methods for the matrix sign function. *SIAM Journal on Matrix Analysis and Applications*, 12(2):273–291, 1991. (Cited on p. 8.)

[20] J. Douglas Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *International Journal of Control*, 32(4):677–687, 1980. (Cited on p. 6.)

[21] Heinz Rutishauser. Solution of eigenvalue problems with the LR-transformation. *National Bureau of Standards Applied Mathematics Series*, (49):47–81, 1958. (Cited on p. 17.)

[22] Heinz Rutishauser. *Lectures on Numerical Mathematics*. Birkhäuser, Boston, MA, USA, 1990. xvi+546 pp. Edited by Martin Gutknecht, with the assistance of Peter Henrici, Peter Uiuchli, Hans-Rudolf Schwarz. Translated by Walter Gautschi. ISBN 978-0-8176-3491-9. (Cited on pp. 12 and 17.)

[23] Robert S. Schreiber and Beresford Parlett. Block reflectors: Theory and computation. *SIAM Journal on Numerical Analysis*, 25(1):189–205, 1988. (Cited on p. 8.)

[24] Günther Schulz. Iterative Berechung der reziproken Matrix. *Zeitschrift für Angewandte Mathematik und Mechanik*, 13(1):57–59, 1933. (Cited on p. 7.)