Electronic Transactions on Numerical Analysis.
Volume 65, pp. 326–346, 2026.

ISSN 1068–9613.
DOI: 10.1553/etna_vol65s326

ETNA
Kent State University and
Johann Radon Institute (RICAM)

A NEW STABLE AND INVERSION-FREE ITERATION FOR COMPUTING THE MATRIX SQUARE ROOT OF LARGE AND SPARSE MATRICES*

LI ZHU†, FENG WU†, KEQI YE†, YUELIN ZHAO†, JIQIANG HU†, AND WANXIE ZHONG†

Abstract. The objective of this research is to compute the principal matrix square root using a sparse approximation. A new stable inversion-free iteration (SIFI) is presented. An analysis of the sparsity and error of the matrices involved in the iterative process is given. Based on the bandwidth and error analysis, a more efficient algorithm combining the SIFI with a filtering technique is proposed. The computational efficiency and accuracy of the proposed method are demonstrated through the computation of the principal square root of various matrices, confirming its superiority over existing approaches.

Key words. matrix square root, iterative algorithm, error analysis, bandwidth analysis

AMS subject classifications. 65F45, 65F10, 65F35

1. Introduction.

The matrix square root is one of the most common matrix functions. Unlike scalars, matrices may not always have a square root. Sometimes a matrix has two or more square roots. For example, a nonsingular Jordan block has two square roots, and any involutory matrix is a square root of the identity matrix. If the matrix $\mathbf{A}$ is singular, then the existence of its square root depends on the structure of the Jordan blocks corresponding to the zero eigenvalues [9]. If the matrix $\mathbf{A}$ is real, then it may also lack real roots, for example, when $\mathbf{A}$ is the negative identity matrix. For matrices with different properties, a variety of theories have been developed.

It has been demonstrated that for any matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ with no nonpositive real eigenvalues, there is only one square root $\mathbf{X}$ whose spectrum lies in the open right half-plane. Such a matrix $\mathbf{X}$ is denoted by $\mathbf{A}^{1/2}$ and is called the principal square root [16]. The principal matrix square root plays an important role in many mathematical problems such as the definite generalized eigenvalue problem [10], polar decomposition [26], geometric mean [21], matrix sign function [24], and matrix logarithm function [1]. The matrix square root also plays an important role in many practical problems such as dynamics problems, deep learning [4], and machine vision [14, 22, 25].

Many algorithms have been developed for computing the principal square root of a matrix. One of the most famous algorithms is the Newton iteration [15]

$$\begin{cases} \mathbf{X}_k\mathbf{H}_k + \mathbf{H}_k\mathbf{X}_k = \mathbf{A} - \mathbf{X}_k^2, \\ \mathbf{X}_{k+1} = \mathbf{X}_k + \mathbf{H}_k, \end{cases} \qquad k = 0, 1, \ldots \tag{1.1}$$

By choosing a suitable initial value $\mathbf{X}_0$, a sequence $\{\mathbf{X}_k\}$ is obtained that converges to $\mathbf{A}^{1/2}$. This method has some ability to resist perturbations [4], but it requires solving a Sylvester equation in each iteration, which is expensive. If we set $\mathbf{X}_0 = \mathbf{A}$, then $\mathbf{X}_k$ commutes with $\mathbf{H}_k$, and (1.1) can be rewritten as

$$\mathbf{X}_{k+1} = \frac{1}{2}\left(\mathbf{X}_k + \mathbf{A}\mathbf{X}_k^{-1}\right), \qquad \mathbf{X}_0 = \mathbf{A}. \tag{1.2}$$

*Received February 28, 2024. Accepted June 8, 2026. Published online on July 21, 2026. Recommended by Dario Bini. Project supported by the National Natural Science Foundation of China grants (Nos. 11472076 and 51609034), the Science Foundation of Liaoning Province of China (No. 2021–MS–119), the Dalian Youth Science and Technology Star project (No. 2018RQ06), and the Fundamental Research Funds for the Central Universities grant (Nos. DUT20RC(5)009 and DUT20GJ216).

†Corresponding author: Feng Wu. Key Laboratory of Structural Analysis of Industrial Equipment, Department of Mechanics,Dalian University of Technology,Dalian 116024, People's Republic of China (vonwu@dlut.edu.cn).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

However, Higham [15] pointed out that the iterative scheme (1.2), despite avoiding the solution of the Sylvester equations in (1.1), is unstable in most cases.

To address the stability problem of (1.2), some modified versions have been proposed such as the Denman and Beavers (DB) method [13, 19] and the Meini iteration based on the cyclic reduction algorithm (CR) [23]. The DB method is based on the iteration of the matrix sign function and can be expressed as

$$\begin{cases} \mathbf{X}_0 = \mathbf{A}, \qquad \mathbf{Y}_0 = \mathbf{I}, \\ \mathbf{X}_{k+1} = \dfrac{1}{2}\left(\mathbf{X}_k + \mathbf{Y}_k^{-1}\right), \qquad k = 0, 1, \ldots, \\ \mathbf{Y}_{k+1} = \dfrac{1}{2}\left(\mathbf{Y}_k + \mathbf{X}_k^{-1}\right), \end{cases}$$

which generates a sequence $\{\mathbf{X}_k\}$ that converges to $\mathbf{A}^{1/2}$ and a sequence $\{\mathbf{Y}_k\}$ that converges to $\mathbf{A}^{-1/2}$. The scheme of the CR method is defined by

$$\begin{cases} \mathbf{Z}_0 = 2\left(\mathbf{I} + \mathbf{A}\right), \qquad \mathbf{Y}_0 = \mathbf{I} - \mathbf{A}, \\ \mathbf{Y}_{k+1} = -\mathbf{Y}_k \mathbf{Z}_k^{-1} \mathbf{Y}_k, \qquad k = 0, 1, \ldots \\ \mathbf{Z}_{k+1} = \mathbf{Z}_k - 2\mathbf{Y}_k \mathbf{Z}_k^{-1} \mathbf{Y}_k, \end{cases} \tag{1.3}$$

The sequence $\{\mathbf{Z}_k\}$ generated by (1.3) converges to $\frac{1}{4}\mathbf{A}^{1/2}$. In [20], Iannazzo derived an iteration based on Newton's method,

$$\begin{cases} \mathbf{X}_0 = \mathbf{A}, \qquad \mathbf{H}_0 = \dfrac{1}{2}\left(\mathbf{I} - \mathbf{A}\right), \\ \mathbf{X}_{k+1} = \mathbf{X}_k + \mathbf{H}_k, \qquad k = 0, 1, \ldots, \\ \mathbf{H}_{k+1} = -\dfrac{1}{2}\mathbf{H}_k \mathbf{X}_{k+1}^{-1} \mathbf{H}_k, \end{cases}$$

which is equivalent to (1.3) but generates a sequence $\{\mathbf{X}_k\}$ that converges directly to $\mathbf{A}^{1/2}$.

The above algorithms (DB, CR, IN) have been proved to be stable [17, 20, 23]. However, in the face of large-scale sparse matrices, these methods are time-consuming because they all require matrix inversion to some extent, which is costly. Designing a stable scheme that avoids matrix inversion is an important objective and is discussed in this paper.

Another limitation in computing the root of a large sparse matrix is the required memory, as the root of a large sparse matrix is often dense or full. Demko et al. [12], while studying the inverse of symmetric positive definite banded matrices, found that the absolute values of the elements of the final inverse matrix decay exponentially with increasing distance from the main diagonal, and the more diagonally dominant the matrix is, the stronger the decay. Benzi et al. [5] successfully extended this property to general matrix functions and proved that the elements of the matrix functions of all symmetric banded matrices decay exponentially with increasing distance from the main diagonal. In [6], they also summarized this decay property as a localization phenomenon and proposed that some of the smallest elements can be neglected in numerical calculations without affecting the computational error, which has a positive impact on the design of fast approximation algorithms. Inspired by this, Gao et al. [27] proposed a filtering technique and used it to compute large-scale sparse matrix exponentials, showing high efficiency. Wu et al. [28] introduced the ε-bandwidth to measure the sparsity of a matrix. If the ε-bandwidth of a matrix is much smaller than its dimension, then the matrix is treated as a nearly sparse matrix. For a matrix whose principal square root matrix is nearly sparse, it may be an effective way to combine the filtering technique with an existing iterative

ETNA
Kent State University and
Johann Radon Institute (RICAM)

scheme to construct an efficient method for the computation of the square root of large and sparse matrices.

This paper is concerned with the numerical computation of the principal square root of a large-scale sparse matrix. In Section 2, a new stable inversion-free iteration (SIFI) is developed, and a stability proof is presented. In Section 3, we point out the applicability of the filtering technique via a bandwidth analysis. In Section 4, the filtering algorithm is introduced into the proposed SIFI, and the adaptive selection of the filtering threshold is discussed in detail. Numerical experiments are given in Section 5 to test the accuracy and efficiency of the proposed method. Conclusions are summarized in Section 6.

2. A stable inversion-free iteration. The problem of finding the principal square root of a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ involves solving the following equation:

$$\mathbf{X}^2 - \mathbf{A} = \mathbf{0}. \tag{2.1}$$

Applying Newton's method to this equation inevitably introduces matrix inversions. For a large-scale sparse matrix, the inverse matrix is usually dense, which makes the computation very expensive. To avoid or reduce matrix inversions, the authors in [2] have recommended the following equation:

$$\mathbf{X}^{-2} - \mathbf{A}^{-1} = \mathbf{0}. \tag{2.2}$$

Applying Newton's method to (2.2) gives the Newton update:

$$\mathbf{X}_{k+1}\mathbf{X}_k^{-1} + \mathbf{X}_k^{-1}\mathbf{X}_{k+1} = 2\mathbf{I} + \mathbf{I} - \mathbf{X}_k\mathbf{A}^{-1}\mathbf{X}_k.$$

If we impose the commutativity condition $\mathbf{A}\mathbf{X}_k = \mathbf{X}_k\mathbf{A}$ (which holds for all iteration indices k if $\mathbf{A}\mathbf{X}_0 = \mathbf{X}_0\mathbf{A}$), then $\mathbf{X}_k$ commutes with $\mathbf{A}^{-1}$, and the above equation simplifies to

$$\begin{cases} \mathbf{X}_{k+1} = \mathbf{X}_k\left(\mathbf{I} + \frac{1}{2}\mathbf{Y}_k\right), \\ \mathbf{Y}_{k+1} = \mathbf{I} - \mathbf{A}^{-1}\mathbf{X}_{k+1}^2, \qquad k \geq 0. \end{cases} \tag{2.3}$$

Actually, the above iteration is a variant of the inverse Newton iteration [3]. If we replace $\mathbf{A}^{-1}$ with $\mathbf{A}$ and maintain the commutativity condition, then this yields a sequence $\mathbf{X}_k \to \mathbf{A}^{-1/2}$, as $k \to \infty$.

2.1. How to choose a suitable $\mathbf{X_0}$. According to [7], if $\mathbf{A}\mathbf{X}_0 = \mathbf{X}_0\mathbf{A}$, one has

$$\mathbf{Y}_{k+1} = \frac{3}{4}\mathbf{Y}_k^2 + \frac{1}{4}\mathbf{Y}_k^3. \tag{2.4}$$

Then, taking the norm of both sides of (2.4) yields

$$\|\mathbf{Y}_{k+1}\| \leq \frac{3}{4}\|\mathbf{Y}_k\|^2 + \frac{1}{4}\|\mathbf{Y}_k\|^3,$$

for any consistent norm. Therefore, if we can find such a $\mathbf{X}_0$ which commutes with $\mathbf{A}$ and satisfies $\|\mathbf{Y}_0\| < 1$, then the iteration (2.3) will generate the sequence $\{\mathbf{X}_k\}$ converging to $\mathbf{A}^{1/2}$, and the auxiliary sequence $\{\mathbf{Y}_k\}$ will converge to the zero matrix as $k \to \infty$. The most common selection is $\mathbf{X}_0 = \mathbf{A}$, which, however, has a great limitation: it requires $\|\mathbf{I} - \mathbf{A}\| < 1$ to converge [17]. To overcome this limitation, we here recommend the following choice:

$$\mathbf{X}_0 = \sqrt{\frac{\alpha}{\|\mathbf{A}\|}}\mathbf{A}, \qquad \mathbf{Y}_0 = \mathbf{I} - \frac{\alpha}{\|\mathbf{A}\|}\mathbf{A},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

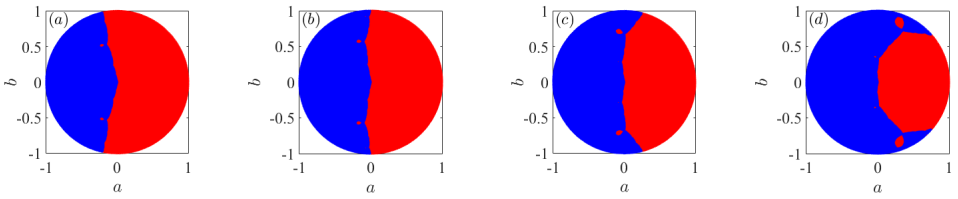


FIG. 2.1. *Convergence region (red) of the iteration (2.3) for different α: (a) $\alpha = 0.1$; (b) $\alpha = 0.5$; (c) $\alpha = 1$; (d) $\alpha = 2$.*

where $\alpha > 0$ is a scaling parameter. For any matrix $\mathbf{A}$ with no nonpositive real eigenvalues, we can always choose a sufficiently small $\alpha > 0$ such that $\|\mathbf{Y}_0\| = \left\| I - \frac{\alpha}{\|\mathbf{A}\|}\mathbf{A} \right\| < 1$, which guarantees the convergence of iteration (2.3) to $\mathbf{A}^{1/2}$. In addition, if we set

$$\mathbf{X}_0 = \sqrt{\frac{\alpha}{\|\mathbf{A}\|}}\mathbf{I}, \qquad \mathbf{Y}_0 = \mathbf{I} - \frac{\alpha}{\|\mathbf{A}\|}\mathbf{A},$$

and replace $\mathbf{A}^{-1}$ with $\mathbf{A}$ in (2.3), then this yields a sequence $\mathbf{X}_k \to \mathbf{A}^{-1/2}$, as $k \to \infty$. According to [18, Theorem 4.15], which establishes superlinear convergence of this iteration for nonsingular matrices, the convergence region of the iteration (2.4) depends on the following scalar iteration:

$$y_{k+1} = \frac{3}{4}y_k^2 + \frac{1}{4}y_k^3, \qquad y_0 = 1 - \alpha a - \alpha b\mathrm{i}, \tag{2.5}$$

where a and b are the real and imaginary parts of a certain eigenvalue $z = a + b\mathrm{i}$ of $\|\mathbf{A}\|^{-1}\mathbf{A}$, respectively. Due to $\rho(\mathbf{A}) \le \|\mathbf{A}\|$, we have $|a + b\mathrm{i}| \le 1$. Obviously, when the eigenvalues of $\|\mathbf{A}\|^{-1}\mathbf{A}$ are all in the set

$$\left\{ z : \ z \in \mathbb{C}, \text{ and } \left| z - \frac{1}{\alpha} \right| < \frac{1}{\alpha} \right\}, \tag{2.6}$$

we obtain $|y_0| < 1$, and the scalar iteration (2.5) converges. Therefore, when the eigenvalues of $\|\mathbf{A}\|^{-1}\mathbf{A}$ are all real positive, taking $\alpha = 2$ ensures convergence of the iteration (2.3). In fact, the actual convergence region is larger than (2.6). Figure 2.1 displays the convergence regions when α takes different values. The figure is obtained as follows: at first, we select 10^6 initial points $y_0^{(i)}$ randomly in the complex region $|z| < 1$ and then substitute these initial points into (2.5) and iterate for 200 steps. If, for the i-th initial point $y_0^{(i)}$, $\left| y_{200}^{(i)} \right| \le 10^{-10}$, then $y_0^{(i)}$ is marked in red; otherwise $y_0^{(i)}$ is marked in blue. As can be seen in Figure 2.1, the value of α significantly affects the size of the convergence region. When $\alpha = 0.5$, all red points (representing convergent initial values) lie within the open right half-plane of the complex plane, meaning that any matrix whose normalized eigenvalues $\|\mathbf{A}\|^{-1}\mathbf{A}$ are all in the open right half-plane will converge under this iteration. A similar approach to the initial value selection can also be used for the Newton-Schulz iteration [18]. The original Newton-Schulz iteration for the matrix square root is as follows:

$$\begin{cases} \mathbf{X}_0 = \mathbf{A}, \qquad \mathbf{Y}_0 = \mathbf{I}, \\ \mathbf{X}_{k+1} = \frac{1}{2}\mathbf{X}_k\left(3\mathbf{I} - \mathbf{Y}_k\mathbf{X}_k\right), \\ \mathbf{Y}_{k+1} = \frac{1}{2}\left(3\mathbf{I} - \mathbf{Y}_k\mathbf{X}_k\right)\mathbf{Y}_k. \end{cases} \tag{2.7}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Its convergence condition is $\|\mathbf{I}-\mathbf{A}\| < 1$, and it requires the commutativity condition $\mathbf{A}\mathbf{X}_0 = \mathbf{X}_0\mathbf{A}$ (which holds for all k if satisfied initially) to simplify the Newton update to this form. To extend its convergence region, we can use the same scaled initial value approach. Set $\mathbf{X}_0 = \sqrt{\frac{\alpha}{\|\mathbf{A}\|}}\mathbf{A}, \mathbf{Y}_0 = \sqrt{\frac{\alpha}{\|\mathbf{A}\|}}\mathbf{I}$. Then the sequence $\{\mathbf{X}_k\}$ generated by (2.7) converges to $\mathbf{A}^{1/2}$, and the sequence $\{\mathbf{Y}_k\}$ converges to $\mathbf{A}^{-1/2}$. According to [18], if $\mathbf{X}_k$ commutes with $\mathbf{Y}_k$ (which is guaranteed by the initial commutativity condition $\mathbf{A}\mathbf{X}_0 = \mathbf{X}_0\mathbf{A}$), then $\mathbf{X}_k = \mathbf{A}^{-1}\mathbf{Y}_k$ follows by induction. Let

$$\mathbf{R}_k = \mathbf{I} - \mathbf{A}^{-1}\mathbf{X}_k^2 = \mathbf{I} - \mathbf{X}_k\mathbf{Y}_k.$$

Then we have

$$\begin{aligned}\mathbf{R}_{k+1} &= \mathbf{I} - \mathbf{Y}_{k+1}\mathbf{X}_{k+1} = \mathbf{I} - \frac{1}{4}\mathbf{X}_k(3\mathbf{I} - \mathbf{Y}_k\mathbf{X}_k)^2\mathbf{Y}_k \\ &= \mathbf{I} - \frac{1}{4}\mathbf{X}_k(2\mathbf{I} + \mathbf{R}_k)^2\mathbf{Y}_k = \mathbf{I} - \frac{1}{4}\left(\mathbf{I} - \mathbf{R}_k\right)\left(2\mathbf{I} + \mathbf{R}_k\right)^2 \\ &= \mathbf{I} - \left(\mathbf{I} + \mathbf{R}_k + \frac{1}{4}\mathbf{R}_k^2 - \mathbf{R}_k - \mathbf{R}_k^2 - \frac{1}{4}\mathbf{R}_k^3\right) = \frac{3}{4}\mathbf{R}_k^2 + \frac{1}{4}\mathbf{R}_k^3.\end{aligned} \tag{2.8}$$

Obviously, equations (2.4) and (2.8) are the same, which means the Newton-Schulz iteration shares the same convergence condition as iteration (2.3).

2.2. Stability analysis. Although iteration (2.3) requires only one matrix inversion, i.e., $\mathbf{A}^{-1}$, it is unstable, similar to the iteration generated directly by the Newton method for (2.1), as will be proved below.

The iteration (2.3) can be written as a fixed-point iteration $\mathbf{X}_{k+1} = F(\mathbf{X}_k)$, where

$$F(\mathbf{X}) = \mathbf{X}\left(\mathbf{I} + \frac{1}{2}\left(\mathbf{I} - \mathbf{A}^{-1}\mathbf{X}^2\right)\right) = \frac{3}{2}\mathbf{X} - \frac{1}{2}\mathbf{A}^{-1}\mathbf{X}^3.$$

The Fréchet derivative of F at the solution $\mathbf{X}^* = \mathbf{A}^{1/2}$ is

$$L_F(\mathbf{X}^*;\mathbf{H}) = \frac{3}{2}\mathbf{H} - \frac{1}{2}\left(\mathbf{A}^{-1}\mathbf{X}^*\mathbf{H}\mathbf{X}^* + A^{-1}\mathbf{X}^{*2}\mathbf{H}\right) = \frac{1}{2}\mathbf{H} - \frac{1}{2}\mathbf{A}^{-1/2}\mathbf{H}\mathbf{A}^{1/2}.$$

Applying the vec operator yields the Jacobian matrix

$$\mathbf{J} = \frac{1}{2}\left[\mathbf{I} - \left(\mathbf{A}^{\mathrm{T}}\right)^{1/2} \otimes \mathbf{A}^{-1/2}\right].$$

The stability condition of the above iterative scheme is $\rho(\mathbf{J}) \leq 1$. However, according to [18, Equation (6.24)], for Hermitian positive definite matrices, the condition $\rho(\mathbf{J}) \leq 1$ holds only if $\kappa_2(\mathbf{A}) < 9$. Thus, iteration (2.3) is conditionally stable.

2.3. The proposed iteration. According to the above analysis, iteration (2.3) cannot guarantee stability. In this section, we present a new stable iterative scheme. According to (2.4), $\mathbf{Y}_{k+1}$ can also be generated from $\mathbf{Y}_k$, and thus the iterative scheme can be rewritten as

$$\mathbf{X}_{k+1} = \mathbf{X}_k\left(\mathbf{I} + \frac{1}{2}\mathbf{Y}_k\right), \qquad \mathbf{Y}_{k+1} = \mathbf{Y}_k^2\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\mathbf{Y}_k\right). \tag{2.9}$$

Obviously, if we do not take into consideration round-off errors, then the schemes (2.3) and (2.9) are equivalent, so the convergence analysis in Section 2.2 is also applicable to (2.9).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

However, if round-off errors are considered, then we find that iteration (2.9) is numerically stable.

Let $F(\mathbf{X},\mathbf{Y}) = \left[\mathbf{X}\left(I+\frac{1}{2}\mathbf{Y}\right), \mathbf{Y}^2\left(\frac{3}{4}\mathbf{I}+\frac{1}{4}\mathbf{Y}\right)\right]^{\mathrm{T}}$ be the iteration map. Its Fréchet derivative at the fixed point $\mathbf{X}=\mathbf{A}^{1/2}, \mathbf{Y}=\mathbf{0}$ is

$$\begin{bmatrix} \mathbf{I} & \mathbf{A}^{1/2} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

The eigenvalues of this block matrix are 1 and 0, both with magnitude ≤ 1. Therefore, the proposed iteration (2.9) is unconditionally numerically stable for all matrices $\mathbf{A}$ with no nonpositive real eigenvalues.

It can be seen that the computation of $\mathbf{Y}_{k+1}$ depends only on $\mathbf{Y}_k$, and no longer involves $\mathbf{A}^{-1}$. Also, the initial values $\mathbf{X}_0 = \sqrt{0.5/\|\mathbf{A}\|}\mathbf{A}$, $\mathbf{Y}_0 = \mathbf{I} - 0.5\mathbf{A}/\|\mathbf{A}\|$ do not involve the inverse $\mathbf{A}^{-1}$. Hence, the iterative scheme (2.9) combined with the initial matrix $\mathbf{X}_0 = \sqrt{0.5/\|\mathbf{A}\|}\mathbf{A}$ is a stable iteration that fully avoids matrix inversion. As a matrix inversion costs much more computational time than matrix multiplication for large-scale sparse matrices, avoiding matrix inversion can reduce computation time effectively.

3. Sparsity of the principal matrix square root. Although the iterative algorithm (2.9) is stable and avoids computing the matrix inverse, it is still not applicable to the computation of the principal square root of large-scale matrices due to the limitations of computational cost and memory. However, it will be shown in this section that there are some sparse matrices whose principal square roots are still sparse, or nearly sparse, with the help of the bandwidth theory proposed in [28]. We first introduce the definitions of matrix bandwidth, real bandwidth, and ε-bandwidth.

DEFINITION 3.1. *For a sparse matrix $\mathbf{A}\in\mathbb{C}^{n\times n}$, whose elements are denoted by a_{ij}, the bandwidth $l(\mathbf{A})$ of the matrix $\mathbf{A}$ is defined by*

$$l(\mathbf{A}) = \max_{\Omega(\mathbf{A})}(j-i,0) + \max_{\Omega(\mathbf{A})}(i-j,0),$$

where $\Omega(\mathbf{A}) := \{(i,j) : a_{ij}\neq 0\}$.

DEFINITION 3.2. *For a sparse matrix $\mathbf{A}\in\mathbb{C}^{n\times n}$ and any $\varepsilon>0$, the following holds:*

(a) The real bandwidth $\lambda(\mathbf{A})$ is

$$\lambda(\mathbf{A}) = \min_{\mathbf{P}\in P} l(\mathbf{P}\mathbf{A}\mathbf{P}^T).$$

Here P is the set of all permutation matrices. Essentially, the matrix $\mathbf{P}$ permutes the columns and rows of $\mathbf{A}$ to make it banded.

(b) The ε-bandwidth $\beta(\varepsilon,\mathbf{A})$ is

$$\beta(\varepsilon,\mathbf{A}) = \min_{\mathbf{B}\in\Gamma(\varepsilon)}\lambda(\mathbf{A}-\mathbf{B}), \qquad \Gamma(\varepsilon) := \left\{\mathbf{B} : \mathbf{B}\in\mathbb{C}^{n\times n}, \text{ and } \|\mathbf{B}\|\leq\varepsilon\|\mathbf{A}\|\right\}.$$

For the real bandwidth and ε-bandwidth, we have the following lemma [28]:

LEMMA 3.3. *Let $\mathbf{A},\mathbf{B}\in\mathbb{C}^{n\times n}$ be sparse matrices and $p_m(\mathbf{A}) = \sum_{i=0}^{m} c_i\mathbf{A}^i$, $c_i\in\mathbb{C}$, where m is an arbitrary positive integer. Then,*

(a) $\lambda(p_m(\mathbf{A}))\leq m\lambda(\mathbf{A})$;

(b) $\beta(\varepsilon_p, p_m(\mathbf{A}))\leq m\lambda(\mathbf{A}-\mathbf{B})$, where

$$\varepsilon_p = \|p_m(\mathbf{A})\|^{-1}\|p_m(\mathbf{A}) - p_m(\mathbf{A}-\mathbf{B})\|.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

The sparsity of $\mathbf{A}^{1/2}$ can be measured in terms of the real bandwidth or the ε-bandwidth. In fact, we have the following theorems:

THEOREM 3.4. *For any sparse matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ without nonpositive real eigenvalues and any $\varepsilon > 0$, if $y_0 = \left\|\mathbf{I} - 0.5\|\mathbf{A}\|^{-1}\mathbf{A}\right\| < 1$, then*

$$\beta(\varepsilon, \mathbf{A}^{1/2}) \le \nu(\varepsilon, y_0)\,\lambda(\mathbf{A}),$$
$$\nu(\varepsilon, y_0) = \min\left\{ s,\ s \in \mathbb{N},\ \textit{and},\ \sum_{k=s}^{\infty}\left|\frac{d_j}{j!}\right| y_0^j \le \sqrt{0.5}\varepsilon \right\},$$

where $d_j = \prod_{i=0}^{j-1}\left(\frac{1}{2} - i\right)$.

Proof. The Taylor series for $\mathbf{A}^{1/2}$ can be expressed as

$$\mathbf{A}^{1/2} = \frac{\sqrt{\|\mathbf{A}\|}}{\sqrt{0.5}}\left(\frac{0.5}{\|\mathbf{A}\|}\mathbf{A}\right)^{1/2} = \frac{\sqrt{\|\mathbf{A}\|}}{\sqrt{0.5}}\left(\mathbf{I} - \mathbf{Y}_0\right)^{1/2} = \frac{\sqrt{\|\mathbf{A}\|}}{\sqrt{0.5}}\sum_{j=0}^{\infty}\frac{d_j}{j!}(-\mathbf{Y}_0)^j,$$

where $d_j = \prod_{i=0}^{j-1}\left(\frac{1}{2} - i\right)$ and $\mathbf{Y}_0 = \mathbf{I} - 0.5\|\mathbf{A}\|^{-1}\mathbf{A}$. Thus, there is a polynomial approximation of $\mathbf{A}^{1/2}$

$$f_N(\mathbf{Y}_0) = \frac{\sqrt{\|\mathbf{A}\|}}{\sqrt{0.5}}\sum_{j=0}^{N}\frac{d_j}{j!}(-\mathbf{Y}_0)^j,$$

whose real bandwidth satisfies $\lambda\left(f_N\left(Y_0\right)\right) \le N\lambda(\mathbf{A})$, according to Lemma 3.3(a). If N satisfies

$$\begin{aligned}\frac{\sqrt{\|\mathbf{A}\|}}{\sqrt{0.5}\left\|\mathbf{A}^{1/2}\right\|}\left\|\sum_{j=N+1}^{\infty}\frac{d_j}{j!}(-\mathbf{Y}_0)^j\right\| &\le \frac{1}{\sqrt{0.5}}\left\|\sum_{j=N+1}^{\infty}\frac{d_j}{j!}(-\mathbf{Y}_0)^j\right\| \\ &\le \frac{1}{\sqrt{0.5}}\sum_{j=N+1}^{\infty}\left|\frac{d_j}{j!}\right|\|\mathbf{Y}_0\|^j < \varepsilon,\end{aligned} \tag{3.1}$$

then

$$\frac{\left\|\mathbf{A}^{1/2} - f_N(\mathbf{Y}_0)\right\|}{\left\|\mathbf{A}^{1/2}\right\|} \le \varepsilon.$$

Obviously, the minimum of N satisfying (3.1) is $\nu(\varepsilon, y_0)$. Let $\mathbf{B}_0 = \frac{\sqrt{\|\mathbf{A}\|}}{\sqrt{0.5}}\sum_{j=N+1}^{\infty}\frac{d_j}{j!}(-\mathbf{Y}_0)^j$. Then, according to (3.1), we have $\|\mathbf{B}_0\| \le \varepsilon\left\|\mathbf{A}^{1/2}\right\|$, followed by $\mathbf{B}_0 \in \Gamma(\varepsilon)$. According to Definition 3.2, we have

$$\lambda\left(f_N\left(\mathbf{Y}_0\right)\right) = \lambda(\mathbf{A}^{1/2} - \mathbf{B}_0) \ge \min_{\mathbf{B}\in\Gamma(\varepsilon)}\lambda(\mathbf{A}^{1/2} - \mathbf{B}) = \beta(\varepsilon, \mathbf{A}^{1/2}).$$

Finally, according to Lemma 3.3(a),

$$\beta(\varepsilon, \mathbf{A}^{1/2}) \le \lambda\left(f_N\left(\mathbf{Y}_0\right)\right) \le N\lambda(\mathbf{A}) \le \nu(\varepsilon, y_0)\,\lambda(\mathbf{A}). \qquad \square$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

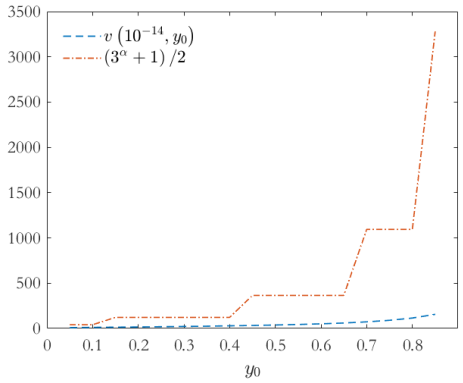


FIG. 3.1. *Numerical comparison of* $v\left(\varepsilon, y_0\right)$ *and* $\left(3^{\alpha}+1\right)/2$.

THEOREM 3.5. *For any matrix* $\mathbf{A} \in \mathbb{C}^{n\times n}$ *with no nonpositive real eigenvalues, the approximation* $\mathbf{X}_j$ *of the square root of* $\mathbf{A}$ *generated by iteration* (2.9) *satisfies*

$$\lambda(\mathbf{X}_j) \leq \frac{3^j+1}{2}\lambda(\mathbf{A}), \qquad j = 0, 1, 2, \ldots$$

Proof. According to Lemma 3.3(a) and considering that $\mathbf{X}_0 = \sqrt{\frac{\alpha}{\|\mathbf{A}\|}}\mathbf{A}$, we have $\lambda(\mathbf{X}_0) \leq \frac{3^0+1}{2}\lambda(\mathbf{A})$. Suppose that $\lambda(\mathbf{X}_i) \leq \frac{3^i+1}{2}\lambda(\mathbf{A})$ holds for i. Considering the iteration (2.9), $\mathbf{X}_{i+1} = \mathbf{X}_i + \mathbf{A}^{-1}\mathbf{X}_i^3$, the iterate $\mathbf{X}_{i+1}$ is a matrix polynomial in $\mathbf{A}$ of degree $\frac{1}{2}(3^i+1)$. Then, using Lemma 3.3(a), we have $\lambda(\mathbf{X}_{i+1}) \leq \frac{3^{i+1}+1}{2}\lambda(\mathbf{A})$. Setting $i = j$, we obtain $\lambda(\mathbf{X}_j) \leq \frac{3^j+1}{2}\lambda(\mathbf{A})$, $j = 0, 1, 2, \ldots$ ∎

According to Theorem 3.5, for any given relative error bound $\varepsilon > 0$, the real bandwidth of the approximate matrix root calculated by (2.9) satisfies

$$\lambda(\mathbf{X}_\alpha) \leq \frac{3^\alpha+1}{2}\lambda(\mathbf{A}), \qquad \alpha = \min\left\{ s : \frac{\left\|\mathbf{A}^{1/2} - \mathbf{X}_s\right\|}{\left\|\mathbf{A}^{1/2}\right\|} \leq \varepsilon \right\}.$$

Additionally, according to Theorem 3.4, $\beta(\varepsilon, \mathbf{A}^{1/2}) \leq \nu\left(\varepsilon, y_0\right)\lambda(\mathbf{A})$.

We now provide a numerical example to compare the upper bound for the real bandwidth of the approximation matrix computed by iteration (2.9) with that for the ε-bandwidth of $\mathbf{A}^{1/2}$ with $\varepsilon = 10^{-14}$. The results are displayed in Figure 3.1. It can be seen that when $\varepsilon = 10^{-14}$, we have

$$\nu\left(\varepsilon, y_0\right) \ll \frac{3^\alpha+1}{2}.$$

The numerical comparison shown in Figure 3.1 demonstrates that for the matrix computed by (2.9), there exists a sparser matrix approximation with a similar accuracy. This property is very helpful for the computation of large-scale sparse matrix square roots.

4. SIFI combined with filtering. Obviously, if (2.9) is used directly, then the results will be dense due to multiple matrix multiplications, and the amount of computation and required storage are very large. Therefore, it is necessary to conduct an in-depth research on how to obtain sparse approximate square roots. In this section, we improve the iteration (2.9) by introducing an adaptive filtering technique to obtain a more efficient iterative method.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

4.1. Error analysis considering filtering. Because the sparsity of $\mathbf{X}_0 = \sqrt{0.5/\|\mathbf{A}\|}\mathbf{A}$ and $\mathbf{Y}_0 = \mathbf{I} - 0.5\mathbf{A}/\|\mathbf{A}\|$ is the same as that of $\mathbf{A}$, we consider them to be sufficiently sparse, meaning that they do not need to be filtered. Although both $\mathbf{X}_0$ and $\mathbf{Y}_0$ are sparse, the matrices $\mathbf{X}_1 = \mathbf{X}_0(\mathbf{I} + 0.5\mathbf{Y}_0)$ and $\mathbf{Y}_1 = \frac{3}{4}\mathbf{Y}_0^2 + \frac{1}{4}\mathbf{Y}_0^3$ can become dense because the real bandwidth is tripled due to the matrix multiplication. Since the process of computing $\mathbf{X}_1$ involves a matrix multiplication, some small elements may be generated that do not affect the overall accuracy of $\mathbf{X}_1$, which means that there may exist a sparser matrix yielding the same precision as $\mathbf{X}_1$. Therefore, we apply filtering to $\mathbf{X}_1$ and ignore elements close to zero to obtain a sparser matrix $\widehat{\mathbf{X}}_1 = \mathbf{X}_1 - \mathbf{F}_1$, where $\mathbf{F}_1$ represents the matrix consisting of nearly zero elements that are dropped after $\mathbf{X}_1$ is filtered. Similarly, filtering is used as well to compute $\mathbf{Y}_1$, and its computation can be divided into

$$\widehat{\mathbf{P}}_0 = \mathbf{Y}_0^2 - \Delta\mathbf{P}_0, \qquad \widehat{\mathbf{Y}}_1 = \widehat{\mathbf{P}}_0\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\mathbf{Y}_0\right) - \mathbf{E}_1,$$

where $\Delta\mathbf{P}_0$ and $\mathbf{E}_1$ represent the matrices of nearly zero elements that are dropped after filtering $\mathbf{Y}_0^2$ and $\widehat{\mathbf{P}}_0\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\mathbf{Y}_0\right)$, respectively. The subsequent calculations are based on $\widehat{\mathbf{X}}_1$ and $\widehat{\mathbf{Y}}_1$. Generally, the new iteration with filtering (denoted by SIFI_F) can be expressed as follows:

$$\begin{cases} \bar{\mathbf{X}}_{k+1} = \widehat{\mathbf{X}}_k\left(\mathbf{I} + \frac{1}{2}\widehat{\mathbf{Y}}_k\right), & \widehat{\mathbf{X}}_{k+1} = \bar{\mathbf{X}}_{k+1} - \mathbf{F}_{k+1}, \\ \mathbf{P}_k = \widehat{\mathbf{Y}}_k^2, & \widehat{\mathbf{P}}_k = \mathbf{P}_k - \Delta\mathbf{P}_k, \\ \bar{\mathbf{Y}}_{k+1} = \widehat{\mathbf{P}}_k\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\widehat{\mathbf{Y}}_k\right), & \widehat{\mathbf{Y}}_{k+1} = \bar{\mathbf{Y}}_{k+1} - \mathbf{E}_{k+1}. \end{cases} \tag{4.1}$$

Obviously, it can be expected that appropriate constraints for $\mathbf{F}_{k+1}$, $\Delta\mathbf{P}_k$, and $\mathbf{E}_{k+1}$ are needed to improve the computational efficiency as much as possible and to ensure that the error of the final result does not exceed a given tolerance bound. Let

$$\Delta\mathbf{X}_k = \widehat{\mathbf{X}}_k - \mathbf{X}_k, \qquad \Delta\mathbf{Y}_k = \widehat{\mathbf{Y}}_k - \mathbf{Y}_k, \qquad k = 0, 1, 2, \ldots, \tag{4.2}$$

where $\mathbf{X}_k$ and $\mathbf{Y}_k$ are the matrices computed without filtering. They both commute with $\mathbf{A}$. Substituting (4.2) into the first equation of (4.1) and ignoring small higher-order quantities, we have

$$\begin{aligned} \bar{\mathbf{X}}_{k+1} = \widehat{\mathbf{X}}_k\left(\mathbf{I} + \frac{1}{2}\widehat{\mathbf{Y}}_k\right) &= (\mathbf{X}_k + \Delta\mathbf{X}_k)\left[\mathbf{I} + \frac{1}{2}(\mathbf{Y}_k + \Delta\mathbf{Y}_k)\right] \\ &= \mathbf{X}_{k+1} + \Delta\mathbf{X}_k\left(\mathbf{I} + \frac{1}{2}\mathbf{Y}_k\right) + \frac{1}{2}\mathbf{X}_k\Delta\mathbf{Y}_k. \end{aligned} \tag{4.3}$$

Because $\widehat{\mathbf{X}}_k = \bar{\mathbf{X}}_k - \mathbf{F}_k$, (4.3) can be written as

$$\widehat{\mathbf{X}}_{k+1} = \bar{\mathbf{X}}_{k+1} - \mathbf{F}_{k+1} = \mathbf{X}_{k+1} + \Delta\mathbf{X}_k\left(\mathbf{I} + \frac{1}{2}\mathbf{Y}_k\right) + \frac{1}{2}\mathbf{X}_k\Delta\mathbf{Y}_k - \mathbf{F}_{k+1}.$$

Combining the above equation with (4.2) yields

$$\Delta\mathbf{X}_{k+1} = \Delta\mathbf{X}_k\left(\mathbf{I} + \frac{1}{2}\mathbf{Y}_k\right) + \frac{1}{2}\mathbf{X}_k\Delta\mathbf{Y}_k - \mathbf{F}_{k+1}. \tag{4.4}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

According to (4.1) and (4.2), $\widehat{\mathbf{P}}_k$ can be expressed as

$$\begin{aligned}\widehat{\mathbf{P}}_k &= \mathbf{P}_k - \Delta\mathbf{P}_k = \widehat{\mathbf{Y}}_k^2 - \Delta\mathbf{P}_k = (\mathbf{Y}_k + \Delta\mathbf{Y}_k)^2 - \Delta\mathbf{P}_k \\ &= \mathbf{Y}_k^2 + \mathbf{Y}_k\Delta\mathbf{Y}_k + \Delta\mathbf{Y}_k\mathbf{Y}_k - \Delta\mathbf{P}_k + \Delta\mathbf{Y}_k^2\,.\end{aligned} \tag{4.5}$$

Based on $\widehat{\mathbf{P}}_k$, we compute $\bar{\mathbf{Y}}_{k+1}$ by

$$\bar{\mathbf{Y}}_{k+1} = \widehat{\mathbf{P}}_k\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\widehat{\mathbf{Y}}_k\right) = \widehat{\mathbf{P}}_k\left[\frac{3}{4}\mathbf{I} + \frac{1}{4}\left(\mathbf{Y}_k + \Delta\mathbf{Y}_k\right)\right].$$

Substituting (4.5) into the above equation and ignoring small higher-order quantities, we obtain

$$\bar{\mathbf{Y}}_{k+1} = \mathbf{Y}_{k+1} + \frac{1}{4}\mathbf{Y}_k^2\Delta\mathbf{Y}_k + \left(\mathbf{Y}_k\Delta\mathbf{Y}_k + \Delta\mathbf{Y}_k\mathbf{Y}_k - \Delta\mathbf{P}_k\right)\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\mathbf{Y}_k\right).$$

Because $\widehat{\mathbf{Y}}_k = \bar{\mathbf{Y}}_k - \mathbf{E}_k$, we have

$$\begin{aligned}\widehat{\mathbf{Y}}_{k+1} &= \bar{\mathbf{Y}}_{k+1} - \mathbf{E}_{k+1} \\ &= \mathbf{Y}_{k+1} + \frac{1}{4}\mathbf{Y}_k^2\Delta\mathbf{Y}_k + \left(\mathbf{Y}_k\Delta\mathbf{Y}_k + \Delta\mathbf{Y}_k\mathbf{Y}_k - \Delta\mathbf{P}_k\right)\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\mathbf{Y}_k\right) - \mathbf{E}_{k+1}.\end{aligned}$$

Combining the above equation with (4.2) gives

$$\Delta\mathbf{Y}_{k+1} = \frac{1}{4}\mathbf{Y}_k^2\Delta\mathbf{Y}_k + \left(\mathbf{Y}_k\Delta\mathbf{Y}_k + \Delta\mathbf{Y}_k\mathbf{Y}_k - \Delta\mathbf{P}_k\right)\left(\frac{3}{4}\mathbf{I} + \frac{1}{4}\mathbf{Y}_k\right) - \mathbf{E}_{k+1}. \tag{4.6}$$

Due to $\widehat{\mathbf{X}}_0 = \mathbf{X}_0, \widehat{\mathbf{Y}}_0 = \mathbf{Y}_0$, we have $\Delta\mathbf{X}_0 = \mathbf{0}$, $\Delta\mathbf{Y}_0 = \mathbf{0}$. Combining (4.4) and (4.6) yields

$$\begin{aligned}\Delta\mathbf{X}_{k+1} &= \Delta\mathbf{X}_k\left(\mathbf{I} + 0.5\mathbf{Y}_k\right) + 0.5\mathbf{X}_k\Delta\mathbf{Y}_k - \mathbf{F}_{k+1}, \\ \Delta\mathbf{Y}_{k+1} &= 0.25\mathbf{Y}_k^2\Delta\mathbf{Y}_k + \left(\mathbf{Y}_k\Delta\mathbf{Y}_k + \Delta\mathbf{Y}_k\mathbf{Y}_k - \Delta\mathbf{P}_k\right)\left(0.75\mathbf{I} + 0.25\mathbf{Y}_k\right) - \mathbf{E}_{k+1}, \\ \Delta\mathbf{X}_0 &= \mathbf{0}, \qquad \Delta\mathbf{Y}_0 = \mathbf{0}.\end{aligned}$$

From the above equation, we have

$$\begin{aligned}\|\Delta\mathbf{X}_k\| &\le \left(\|\mathbf{I}\| + 0.5\left\|\mathbf{Y}_{k-1}\right\|\right)\|\Delta\mathbf{X}_{k-1}\| + 0.5\|\mathbf{X}_{k-1}\|\,\|\Delta\mathbf{Y}_{k-1}\| + \|\mathbf{F}_k\|\,, \\ \|\Delta\mathbf{Y}_k\| &\le \left(1.5\|\mathbf{I}\| + 0.75\left\|\mathbf{Y}_{k-1}\right\|\right)\|\mathbf{Y}_{k-1}\|\,\|\Delta\mathbf{Y}_{k-1}\| \\ &\quad + \left(0.75\|\mathbf{I}\| + 0.25\left\|\mathbf{Y}_{k-1}\right\|\right)\|\Delta\mathbf{P}_{k-1}\| + \|\mathbf{E}_k\|\,.\end{aligned} \tag{4.7}$$

Rewrite (4.7) in matrix form,

$$\mathbf{U}_k \le \mathbf{T}_{k-1}\mathbf{U}_{k-1} + \mathbf{B}_{k-1}\mathbf{V}_{k-1}, \qquad k = 1, 2, \ldots, \tag{4.8}$$

where the inequality is meant componentwise and

$$\begin{aligned}\mathbf{U}_k &= \begin{bmatrix}\|\Delta\mathbf{X}_k\| \\ \|\Delta\mathbf{Y}_k\|\end{bmatrix}, \\ \mathbf{T}_{k-1} &= \begin{bmatrix}\|\mathbf{I}\| + 0.5\left\|\mathbf{Y}_{k-1}\right\| & 0.5\|\mathbf{X}_{k-1}\| \\ 0 & \left(1.5\|\mathbf{I}\| + 0.75\left\|\mathbf{Y}_{k-1}\right\|\right)\|\mathbf{Y}_{k-1}\|\end{bmatrix}, \\ \mathbf{B}_{k-1} &= \begin{bmatrix}0 & 0 & 1 \\ 0.75\|\mathbf{I}\| + 0.25\left\|\mathbf{Y}_{k-1}\right\| & 1 & 0\end{bmatrix}, \\ \mathbf{V}_{k-1} &= \begin{bmatrix}\|\Delta\mathbf{P}_{k-1}\| \\ \|\mathbf{E}_k\| \\ \|\mathbf{F}_k\|\end{bmatrix}.\end{aligned} \tag{4.9}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

According to (4.8), if $\mathbf{U}_q$ is known, then the error vector $\mathbf{U}_k$ satisfies the following inequality:

$$\mathbf{U}_k \le \mathbf{Q}_{k,q}\mathbf{U}_q + \sum_{i=q+1}^{k} \mathbf{Q}_{k,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1}, \tag{4.10}$$

where

$$\mathbf{Q}_{k,i} = \begin{cases} \mathbf{T}_{k-1}\cdots\mathbf{T}_i\,, & i = 1,\ 2,\ \ldots,\ k-1, \\ \mathbf{I}, & i = k, \end{cases}$$

which provides a bound for the accumulated filtering errors.

4.2. How to select the adaptive filtering thresholds. The upper bounds for $\|\Delta\mathbf{P}_{k-1}\|$, $\|\mathbf{E}_k\|$, and $\|\mathbf{F}_k\|$ are discussed here to ensure the accuracy of the computational results. After performing several iterations, the norm of the absolute error can be expressed as

$$\left\|\mathbf{A}^{1/2} - \widehat{\mathbf{X}}_k\right\| = \left\|\mathbf{A}^{1/2} - \mathbf{X}_k + \mathbf{X}_k - \widehat{\mathbf{X}}_k\right\| \le \left\|\mathbf{A}^{1/2} - \mathbf{X}_k\right\| + \|\Delta\mathbf{X}_k\|\,. \tag{4.11}$$

Since $\mathbf{Y}_k = \mathbf{I} - \mathbf{A}^{-1}\mathbf{X}_k^2$, we have

$$\mathbf{A}^{1/2} = \mathbf{X}_k(\mathbf{I} - \mathbf{Y}_k)^{-0.5} = \mathbf{X}_k\left(\mathbf{I} + 0.5\mathbf{Y}_k + O\left(\mathbf{Y}_k^2\right)\right).$$

Substituting the above equation into (4.11) yields

$$\left\|\mathbf{A}^{1/2} - \widehat{\mathbf{X}}_k\right\| \le \|0.5\mathbf{X}_k\mathbf{Y}_k\| + \|\Delta\mathbf{X}_k\| + O\left(\|\mathbf{Y}_k\|^2\right).$$

Since we want the iterative residual to be below the tolerance error limit after M more iterations, we also have

$$\left\|\mathbf{A}^{1/2} - \widehat{\mathbf{X}}_{k+M}\right\| \le \|0.5\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\| + \|\Delta\mathbf{X}_{k+M}\| + O\left(\left\|\mathbf{Y}_{k+M}\right\|^2\right). \tag{4.12}$$

Let $\mathbf{e} = [1, 0]^{\mathrm{T}}$. According to (4.10), we have

$$\|\Delta\mathbf{X}_{k+M}\| \le \mathbf{e}^{\mathrm{T}}\mathbf{U}_{k+M} = \mathbf{e}^{\mathrm{T}}\left(\mathbf{Q}_{k+M,k}\mathbf{U}_k + \sum_{i=k+1}^{k+M} \mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1}\right).$$

Substituting the above equation into (4.12) gives

$$\begin{aligned}\left\|\mathbf{A}^{1/2} - \widehat{\mathbf{X}}_{k+M}\right\| \le{}& \|0.5\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\| \\ &+ \mathbf{e}^{\mathrm{T}}\left(\mathbf{Q}_{k+M,k}\mathbf{U}_k + \sum_{i=k+1}^{k+M} \mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1}\right) + O\left(\left\|\mathbf{Y}_{k+M}\right\|^2\right).\end{aligned}$$

If the absolute error is required to be less than the given tolerance error ε_{tol}, then the following inequality should hold:

$$\begin{aligned}&\|0.5\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\| + \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,k}\mathbf{U}_k \\ &\quad + \sum_{i=k+1}^{k+M} \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1} + O\left(\left\|\mathbf{Y}_{k+M}\right\|^2\right) \le \varepsilon_{\text{tol}},\end{aligned} \tag{4.13}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where $\mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1}$ denotes the error accumulation due to the three filterings in each iteration.

In order to avoid large errors in the final result due to an overly aggressive selection of the filtering threshold and to avoid an increase in computational time due to an uneven selection of the filtering threshold, we take an empirical approach and require the error accumulation to be the same for each iteration. Then we have

$$c_{i,1}\left\|\Delta\mathbf{P}_{i-1}\right\| + c_{i,2}\left\|\mathbf{E}_i\right\| + c_{i,3}\left\|\mathbf{F}_i\right\| \le \frac{\phi_{k,M}}{M}, \tag{4.14}$$

with $\phi_{k,M} = \varepsilon_{\text{tol}} - 0.5\left\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\right\| - \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,k}\mathbf{U}_k$ and

$$\begin{aligned} c_{i,1} &= \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\begin{bmatrix}1 & 0 & 0\end{bmatrix}^{\mathrm{T}},\\ c_{i,2} &= \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\begin{bmatrix}0 & 1 & 0\end{bmatrix}^{\mathrm{T}},\\ c_{i,3} &= \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\begin{bmatrix}0 & 0 & 1\end{bmatrix}^{\mathrm{T}}. \end{aligned}$$

As can be seen from (4.14), $\phi_{k,M}$ should be positive; this can be achieved by choosing a suitable M. We explain this in detail in the subsequent section.

If we also use the idea that the errors accumulate identically, then this requires

$$\begin{aligned} \left\|\Delta\mathbf{P}_{i-1}\right\| &\le \frac{\phi_{k,M}}{Kc_{i,1}M}, \qquad & \left\|\mathbf{E}_i\right\| &\le \frac{\phi_{k,M}}{Kc_{i,2}M},\\ \left\|\mathbf{F}_i\right\| &\le \frac{\phi_{k,M}}{Kc_{i,3}M}, \qquad & K &= \sum_{j=1}^{3}\operatorname{sign}(c_{i,j}), \end{aligned} \tag{4.15}$$

which is actually the adaptive filtering threshold and can be determined by simply estimating the required number of residual iterations from the current iteration to final convergence.

4.3. How to estimate the number of residual iterations. According to (4.15), M must satisfy $\phi_{k,M} > 0$, namely,

$$0.5\left\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\right\| + \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,k}\mathbf{U}_k < \varepsilon_{\text{tol}}.$$

For notational convenience, we denote $x_0 = \|\mathbf{X}_k\|$ and $y_0 = \|\mathbf{Y}_k\|$. Then, we perform the following scalar iteration:

$$\begin{cases} x_{m+1} = x_m + 0.5x_m y_m, \\ y_{m+1} = y_m^2\left(0.75 + 0.25y_m\right), \end{cases} \qquad m = 0, 1, \ldots, M-1. \tag{4.16}$$

In this study we use $x_M y_M$ to estimate $\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\|$ as well as $\widetilde{\mathbf{Q}}_{k+M,k}$ to estimate $\mathbf{Q}_{k+M,k}$, where

$$\widetilde{\mathbf{Q}}_{k+M,k} = \widetilde{\mathbf{T}}_{k+M-1}\cdots\widetilde{\mathbf{T}}_k, \qquad \widetilde{\mathbf{T}}_{k+m} = \begin{bmatrix} 1+0.5y_m & 0.5x_m \\ 0 & \left(1.5+0.75y_m\right)y_m \end{bmatrix}. \tag{4.17}$$

Thus, the number of residual iterations can be written as

$$M(k) = \min\left\{m: \ \ m\in\mathbb{N} \ \text{ and } \ 0.5x_m y_m + \mathbf{e}^{\mathrm{T}}\widetilde{\mathbf{Q}}_{k+m,k}\mathbf{U}_k \le \varepsilon_{\text{tol}}\right\}. \tag{4.18}$$

In practical computations, due to the inclusion of filtering, the matrices obtained in our iteration are actually $\widehat{\mathbf{X}}_k, \widehat{\mathbf{Y}}_k$. We obtain the exact $\mathbf{X}_k$ and $\mathbf{Y}_k$ only without filtering. Thus,

ETNA
Kent State University and
Johann Radon Institute (RICAM)

when we apply the filtering technique, the exact $\mathbf{X}_k$ and $\mathbf{Y}_k$ are unknown. However, since the filtering threshold is usually small, it can be expected that the difference between $\widehat{\mathbf{X}}_k$ and $\mathbf{X}_k$, or between $\widehat{\mathbf{Y}}_k$ and $\mathbf{Y}_k$, is very small. Therefore, we use $\widehat{\mathbf{X}}_k$ and $\widehat{\mathbf{Y}}_k$ instead, and then estimate x_m and y_m based on (4.16). In theory, the total number of iterations required for convergence is a constant, and hence $k + M(k)$ must be a constant. With an increase of k, $M(k)$ will decrease. However, using x_m and y_m to estimate $\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\|$ and $M(k)$ is quite conservative, and it cannot guarantee that $k + M(k)$ is a constant. In the first few steps of the iteration, we use x_m and y_m to estimate $\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\|$, which necessarily leads to a large difference between $\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\|$ and $x_m y_m$. As the iteration proceeds, this difference becomes smaller and smaller, and $k + M(k)$ will be close to a constant. When $k + M(k)$ becomes a constant, we consider the current estimate of M to be accurate, and the iteration will converge after M iterations.

According to Section 2.1, as long as $\rho(\mathbf{Y}_0) < 1$, the iteration converges, but sometimes $\|\mathbf{Y}_0\|$ may be larger than 1, which leads to M having no solution in the estimate (4.18), and the adaptive filtering thresholds (4.15) cannot be used. In this case, a fixed threshold is recommended as a supplement, namely, when $\|\mathbf{Y}_k\| > 0.96$, it is required that

$$\|\Delta\mathbf{P}_{k-1}\| \le 0.01\varepsilon_{\text{tol}}, \qquad \|\mathbf{E}_k\| \le 0.01\varepsilon_{\text{tol}}, \qquad \|\mathbf{F}_k\| \le 0.01\varepsilon_{\text{tol}}.$$

4.4. Adaptive algorithm using a relative error bound. In the actual computation, a relative error bound rather than an absolute error bound is often used as the tolerance error. In the previous section, we employed an absolute error bound as the tolerance error and stated the corresponding adaptive filtering threshold. If the relative error bound is selected as the tolerance error ε_{tol}, then according to (4.13), this requires that

$$\begin{aligned}&\|0.5\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\| + \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,k}\mathbf{U}_k \\ &\quad + \sum_{i=k+1}^{k+M} \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1} + O\left(\left\|\mathbf{Y}_{k+M}\right\|^2\right) \le \left\|\mathbf{A}^{1/2}\right\|\varepsilon_{\text{tol}},\end{aligned} \tag{4.19}$$

where $\left\|\mathbf{A}^{1/2}\right\|$ can be evaluated in terms of the following lemma:

LEMMA 4.1. *For the matrices $\mathbf{X}_k$ and $\mathbf{Y}_k$ in the iterative scheme (2.9), if $\|\mathbf{Y}_k\| \le 1$, then*

$$\left\|\mathbf{A}^{1/2}\right\| \ge a_k := \max\left(\sqrt{\|\mathbf{A}\|}, \ \frac{\|\mathbf{X}_k\|}{2 - \sqrt{1 - \|\mathbf{Y}_k\|}}\right).$$

Proof. According to $\mathbf{Y}_k = \mathbf{I} - \mathbf{A}^{-1}\mathbf{X}_k^2$, we have

$$\mathbf{X}_k = \mathbf{A}^{1/2}\left(\mathbf{I} - \mathbf{Y}_k\right)^{1/2} = \mathbf{A}^{1/2}\left(\mathbf{I} - \frac{1}{2}\mathbf{Y}_k - \frac{1}{8}\mathbf{Y}_k^2 - \dots\right),$$

which means

$$\begin{aligned}\|\mathbf{X}_k\| &\le \left\|\mathbf{A}^{1/2}\right\|\left(1 + \frac{1}{2}\|\mathbf{Y}_k\| + \frac{1}{8}\|\mathbf{Y}_k\|^2 + \dots\right) \\ &= \left\|\mathbf{A}^{1/2}\right\|\left(2 - \sqrt{1 - \|\mathbf{Y}_k\|}\right).\end{aligned} \tag{4.20}$$

As $\|\mathbf{A}\| = \left\|\left(\mathbf{A}^{1/2}\right)^2\right\| \le \left\|\left(\mathbf{A}^{1/2}\right)\right\|^2$, it follows that $\left\|\mathbf{A}^{1/2}\right\| \ge \sqrt{\|\mathbf{A}\|}$, and combining this with (4.20) completes the proof. ∎

ETNA
Kent State University and
Johann Radon Institute (RICAM)

In the actual computation, a_k is evaluated by using

$$a_k = \max\left(\sqrt{\|\mathbf{A}\|},\ \frac{\left\|\widehat{\mathbf{X}}_k\right\|}{2-\sqrt{1-\left\|\widehat{\mathbf{Y}}_k\right\|}}\right).$$

Using Lemma 4.1, (4.19) can be rewritten as

$$\begin{aligned}&\|0.5\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\| + \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,k}\mathbf{U}_k \\ &\quad + \sum_{i=k+1}^{k+M} \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,i}\mathbf{B}_{i-1}\mathbf{V}_{i-1} + O\left(\left\|\mathbf{Y}_{k+M}\right\|^2\right) \le a_k\varepsilon_{\text{tol}}.\end{aligned}$$

By the same derivation as in Section 4.2, the upper bounds used for the filtering are

$$\|\Delta\mathbf{P}_{i-1}\| \le \frac{\widetilde{\phi}_{k,M}}{Kc_{i,1}M}, \qquad \|\mathbf{E}_i\| \le \frac{\widetilde{\phi}_{k,M}}{Kc_{i,2}M},$$
$$\|\mathbf{F}_i\| \le \frac{\widetilde{\phi}_{k,M}}{Kc_{i,3}M}, \qquad K = \sum_{j=1}^{3}\operatorname{sign}(c_{i,j}),$$

where

$$\widetilde{\phi}_{k,M} = a_k\varepsilon_{\text{tol}} - 0.5\left\|\mathbf{X}_{k+M}\mathbf{Y}_{k+M}\right\| - \mathbf{e}^{\mathrm{T}}\mathbf{Q}_{k+M,k}\mathbf{U}_k.$$

4.5. Specific algorithm of the proposed method. The SIFI_F method is summarized in Algorithm 1 and Algorithm 2. In lines 7, 9, and 11 of Algorithm 2, we utilize the filtering algorithm proposed in [28] to filter the matrix and obtain a sparser matrix. If the absolute error bound is used, then lines 16–18 in Algorithm 2 should be deleted.

5. Numerical experiments. To test the accuracy of the proposed algorithms, we define the following relative error:

$$\text{err} = \frac{\left\|\mathbf{X}^2 - \mathbf{A}\right\|}{\|\mathbf{A}\|}.$$

Due to the difficulty of evaluating the 2-norm of a large-scale matrix, the 1-norm is applied here. We performed the experiments in MATLAB version R2021b, on a computer with Microsoft Windows 10 21H1 and Intel(R) Core (TM)i7-10750H CPU @ 2.60GHz, and 31.6GB of RAM. In this section, we refer to the iteration (2.9) as SIFI and to Algorithm 2 as SIFI_F. The code of SIFI_F, as well as the matrices used in the third experiment in Section 5.3 are available for download from the website `https://www.rocewea.com/8.html` for interested readers.

5.1. Performance of the SIFI. In the first experiment, we compare the proposed SIFI method with the original algorithm without filtering, i.e., the iterative scheme (2.3), in order to test the effect of matrix inversion on the computational efficiency. When iteration (2.3) is used to compute $\mathbf{A}^{-1}$, there are two ways to implement it:

(a) $\mathbf{Y}_k = \mathbf{I} - \mathbf{A}^{-1}\mathbf{X}_k^2$, where $\mathbf{A}^{-1}$is computed at the beginning once and for all with the MATLAB command $\texttt{inv}\,(\mathbf{A})$;

(b) Set $\mathbf{X}_0 = \sqrt{\frac{\alpha}{\|\mathbf{A}\|}}\mathbf{I}$, $\mathbf{Y}_0 = \mathbf{I} - \frac{\alpha}{\|\mathbf{A}\|}\mathbf{A}$, $\mathbf{Y}_k = \mathbf{I} - \mathbf{A}\mathbf{X}_k^2$. Then, at the end of the iteration, the approximation of the square root is obtained by inverting $\mathbf{X}_k$ by means of the command $\mathbf{X}_k = \texttt{inv}\,(\mathbf{X}_\texttt{k})$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 1 Determination of the adaptive thresholds
$[p_{\text{k,ft}}, e_{\text{k,ft}}, f_{\text{k,ft}}]$ = adaptive_thre$[\|\widehat{\mathbf{X}}_{k-1}\|, \|\widehat{\mathbf{Y}}_{k-1}\|, \mathbf{U}_{k-1}, \mathbf{B}_{k-1}, C, \varepsilon_{\text{tol}}, a_{k-1}]$.

if $\left\|\widehat{\mathbf{Y}}_{k-1}\right\| > 0.96$ **then**
  return $p_{\text{k,ft}} = C\varepsilon_{\text{tol}}$; $e_{\text{k,ft}} = C\varepsilon_{\text{tol}}$; $f_{\text{k,ft}} = C\varepsilon_{\text{tol}}$;
else
  $M = 0$; $\widetilde{\mathbf{Q}}_{k+M-1,k-1} = \mathbf{I}_{2\times 2}$; $x_0 = \left\|\widehat{\mathbf{X}}_{k-1}\right\|$; $y_0 = \left\|\widehat{\mathbf{Y}}_{k-1}\right\|$;
  $\phi_{k-1,0} = a_{k-1}\varepsilon_{\text{tol}} - 0.5x_0y_0 - \begin{bmatrix}1 & 0\end{bmatrix}\mathbf{U}_{k-1}$;
  while $\phi_{k-1,M} \le 0$ **do**
    $M = M + 1$;
    Compute $\widetilde{\mathbf{T}}_{k+M-1}$ according to (4.17);
    $\widetilde{\mathbf{Q}}_{k+M-1,k-1} = \widetilde{\mathbf{T}}_{k+M-1}\widetilde{\mathbf{Q}}_{k+M-2,k-1}$;
    if $M = 1$ **then**
      $\widetilde{\mathbf{Q}}_{k+M-1,k} = \mathbf{I}_{2\times 2}$;
    else
      $\widetilde{\mathbf{Q}}_{k+M-1,k} = \widetilde{\mathbf{T}}_{k+M-1,k}\widetilde{\mathbf{Q}}_{k+M-1,k}$;
    end if
    $x_M = x_{M-1} + 0.5x_{M-1}y_{M-1}$; $y_M = y_{M-1}^2\,(0.75 + 0.25y_{M-1})$;
    $\phi_{k-1,M} = a_{k-1}\varepsilon_{\text{tol}} - 0.5x_My_M - \begin{bmatrix}1 & 0\end{bmatrix}\widetilde{\mathbf{Q}}_{k+M-1,k-1}\mathbf{U}_{k-1}$;
  end while
  Compute $c_{k,1}$; $c_{k,2}$; $c_{k,3}$ according to (4.14);
  return $p_{\text{k,ft}} = \frac{\phi_{k-1,M}}{Kc_{k,1}M}$; $e_{\text{k,ft}} = \frac{\phi_{k-1,M}}{Kc_{k,2}M}$; $f_{\text{k,ft}} = \frac{\phi_{k-1,M}}{Kc_{k,3}M}$; $K = \sum_{j=1}^{3}\text{sign}\,(c_{k,j})$;
end if

In addition, the Newton-Schulz method, the DB method, the IN method, and the sqrtm function [8, 11, 16] in MATLAB are also evaluated. Both DB and IN need matrix inversions. All six methods are computed using the full matrix format. The algorithms are tested with the following matrix $\mathbf{A}$, which is related to the discretization of partial differential equations,

$$\mathbf{A} = \begin{bmatrix} 1-2\lambda & \lambda & & & & \\ \lambda & 1-2\lambda & \lambda & & & \\ & \ddots & \ddots & \ddots & & \\ & & \lambda & 1-2\lambda & \lambda & \\ & & & \lambda & 1-2\lambda & \end{bmatrix}, \tag{5.1}$$

where $\lambda = -1$, and $n = 500, 1000, 1500, 2000$. The computational time and errors of the different methods are listed in Table 5.1.

As shown in Table 5.1, when the dimension reaches 2000, SIFI and the Newton-Schulz iteration are more efficient than the original algorithm (iteration (2.3)), the DB method, and the IN method, but the accuracy of these schemes is the same. The different ways of implementing iteration (2.3) have a significant impact on the efficiency. Iteration (2.3) only performs one matrix inversion, but its computation is the most time-consuming one. To explain this, we compare the matrix density of $\mathbf{A}$ and $\mathbf{A}^{-1}$ in Figure 5.1, generated using the spy function in MATLAB. As can be seen in the figure, although the matrix $\mathbf{A}$ is a very sparse banded matrix, its inverse is quite dense. The high number of calculations caused by the occurrence of a dense matrix in the matrix multiplication slows down the computational efficiency of iteration (2.3). Therefore, for a banded matrix $\mathbf{A}$, although the inverse of $\mathbf{A}$ is generally

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 2 This algorithm computes $\mathbf{A}^{1/2}$ for the given matrix $\mathbf{A}$. The (absolute/relative) error of the result will be less than the given tolerance ε_{tol}.

Let $\widehat{\mathbf{X}}_0 = \sqrt{0.5/\|\mathbf{A}\|}\mathbf{A}$; $\widehat{\mathbf{Y}}_0 = \mathbf{I} - 0.5\mathbf{A}/\|\mathbf{A}\|$; $\widehat{\mathbf{S}}_0 = 0.5\widehat{\mathbf{X}}_0\widehat{\mathbf{Y}}_0$;
$y_0 = 0$; $x_0 = 0$; $\mathbf{U}_0 = \begin{bmatrix} x_0 & y_0 \end{bmatrix}^{\mathrm{T}}$; $\mathbf{B}_0 = \begin{bmatrix} 0 & 0 & 1 \\ 0.75 + \left\|\widehat{\mathbf{Y}}_0\right\| & 1 & 0 \end{bmatrix}$; $C = 0.01$;
$k = 0$; $a_k = 1$ (absolute error bound) or $a_k = \sqrt{\|\mathbf{A}\|}$ (relative error bound);
while $\left\|\widehat{\mathbf{S}}_k\right\| + x_k > a_k\varepsilon_{\text{tol}}$ **do**
  $k = k + 1$;
  [$p_{\text{k,ft}}$,$e_{\text{k,ft}}$,$f_{\text{k,ft}}$] = adaptive_thre $\left[\left\|\widehat{\mathbf{X}}_{k-1}\right\|, \left\|\widehat{\mathbf{Y}}_{k-1}\right\|, \mathbf{U}_{k-1}, \mathbf{B}_{k-1}, C, \varepsilon_{\text{tol}}, a_{k-1}\right]$;
  Filter out $\mathbf{P}_{k-1}$ to get $\widehat{\mathbf{P}}_{k-1}$ which satisfies $\|\Delta\mathbf{P}_{k-1}\| = \left\|\widehat{\mathbf{P}}_{k-1} - \mathbf{P}_{k-1}\right\| \le p_{\text{k,ft}}$;
  $\bar{\mathbf{Y}}_k = \widehat{\mathbf{P}}_{k-1}\left(0.75\mathbf{I} + 0.25\widehat{\mathbf{Y}}_{k-1}\right)$;
  Filter out $\bar{\mathbf{Y}}_k$ to get $\widehat{\mathbf{Y}}_k$ which satisfies $\|\mathbf{E}_k\| = \left\|\widehat{\mathbf{Y}}_k - \bar{\mathbf{Y}}_k\right\| \le e_{\text{k,ft}}$;
  $\bar{\mathbf{X}}_k = \widehat{\mathbf{X}}_{k-1} + \widehat{\mathbf{S}}_{k-1}$;
  Filter out $\bar{\mathbf{X}}_k$ to get $\widehat{\mathbf{X}}_k$which satisfies $\|\mathbf{F}_k\| = \left\|\widehat{\mathbf{X}}_k - \bar{\mathbf{X}}_k\right\| \le f_{\text{k,ft}}$;
  $\widehat{\mathbf{S}}_k = 0.5\widehat{\mathbf{X}}_k\widehat{\mathbf{Y}}_k$;
  Compute $\mathbf{T}_{k-1}$, $\mathbf{B}_{k-1}$ and $\mathbf{V}_{k-1}$ according to (4.9);
  $\mathbf{U}_k = \mathbf{T}_{k-1}\mathbf{U}_{k-1} + \mathbf{B}_{k-1}\mathbf{V}_{k-1}$;
  $x_k = \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{U}_k$;
  if $\left\|\widehat{\mathbf{Y}}_k\right\| < 1$ **then**
    $a_k = \max\left(\sqrt{\|\mathbf{A}\|},\ \left\|\widehat{\mathbf{X}}_k\right\| / \left(2 - \sqrt{1 - \left\|\widehat{\mathbf{Y}}_k\right\|}\right)\right)$;
  end if
end while
return $\mathbf{A}^{1/2} \approx \widehat{\mathbf{X}}_k$;

numerically banded due to the decay properties, the inverse of $\mathbf{A}$ may be very dense, resulting in a delay in subsequent calculations.

The computational efficiency of SIFI is about the same as the Newton-Schulz iteration, but the computational time of SIFI is always lower. Theoretically, the convergence properties of these two algorithms are the same, and the number of matrix multiplications required for each iteration is also 3, so their computational efficiency should be equivalent. We think that the reason for this phenomenon is that, as the iteration progresses, the sequence $\{\mathbf{Y}_k\}$ generated by SIFI converges to $\mathbf{O}$. It tends to be much sparser than the sequence $\{\mathbf{Y}_k\}$ generated by the Newton-Schulz iteration, which converges to $\mathbf{A}^{-1/2}$. As the matrix dimension grows, the efficiency gap widens. If the dimension n reaches 5000, then the gap in the computation time becomes larger than 5 s.

Among the six methods, the sqrtm function has the best computational efficiency, but its accuracy is also the worst. This is likely because sqrtm is based on LAPACK, which is highly optimized. Obviously, SIFI is a better choice when high-precision computations are needed. It must be pointed out that in this experiment, the maximum dimension is 2000, and full matrices instead of sparse matrices are used. In the following experiments, we will further study computational results involving sparse matrices.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.1
Comparisons of different algorithms.

Method	$n = 500$		$n = 1000$		$n = 1500$		$n = 2000$	
	err	time(s)	err	time(s)	err	time(s)	err	time(s)
SIFI	1.42E-15	0.149	1.42E-15	0.848	1.42E-15	2.172	1.42E-15	4.331
Iteration (2.3) (a)	9.84E-16	0.264	9.84E-16	2.918	1.29e-15	7.593	9.96E-16	13.697
Iteration (2.3) (b)	9.70E-16	0.124	8.73E-16	0.782	9.95E-15	2.195	8.89E-16	4.526
Newton-Schulz	1.37E-15	0.150	1.37E-15	0.896	1.37E-15	2.189	1.37E-15	4.358
DB	1.39E-15	0.136	1.39E-15	1.731	1.39E-15	3.209	1.39E-15	5.855
IN	5.52E-16	0.159	5.52E-16	1.983	5.52E-16	4.974	5.52E-16	9.225
sqrtm	4.41E-14	0.008	8.31E-14	0.037	1.13E-13	0.100	1.45E-13	0.207

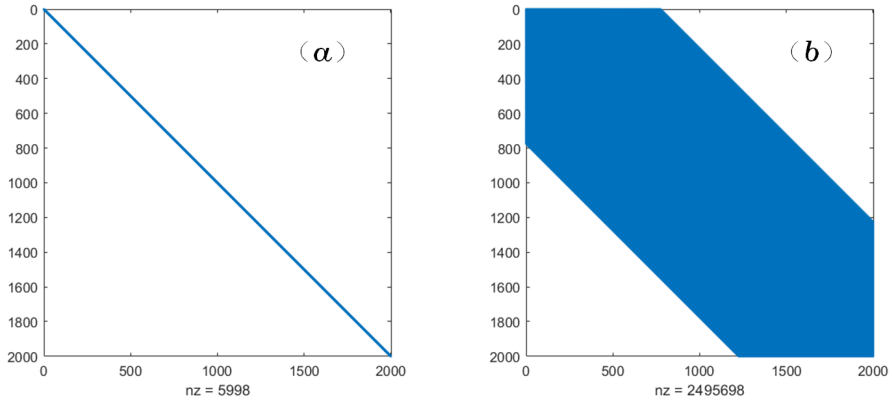


FIG. 5.1. *Comparison of the sparsity of matrix* $\mathbf{A}$ *and* $\mathbf{A}^{-1}$, *(a)* $\mathbf{A}$, *(b)* $\mathbf{A}^{-1}$.

5.2. Changes of the sparse matrix bandwidth in the iteration. The second experiment focuses on the change of the bandwidths of the matrices involved in the proposed method by using the same matrix defined in (5.1) with $\lambda = -1$ and $n = 10000$. The SIFI and SIFI_F algorithms are applied to the computation of the square root of this matrix. The SIFI_F method used here is the relative-error version of Algorithm 2 with a relative tolerance error $\varepsilon_{\text{tol}} = 10^{-13}$.

Both the SIFI and the SIFI_F require 7 iterations with the given tolerance. In Table 5.2, the real bandwidths $\lambda(\mathbf{X}_k)$ and $\lambda(\widehat{\mathbf{X}}_k)$ during the 7 iterations, corresponding to the matrices involved in SIFI and SIFI_F, respectively, are listed. The upper bound for the real bandwidth of $\mathbf{X}_k$ evaluated in terms of Theorem 3.5, $\frac{3^k+1}{2}\lambda(\mathbf{A})$, is also given in the last line of Table 5.2. It can be seen from Table 5.2 that $\lambda(\mathbf{X}_k)$ and $\frac{3^k+1}{2}\lambda(\mathbf{A})$ agree in the first five iteration steps, and $\lambda(\mathbf{X}_k)$ is smaller than $\frac{3^k+1}{2}\lambda(\mathbf{A})$ in the last two iteration steps. This is because many elements in $\mathbf{X}_6$ and $\mathbf{X}_7$ are so close to zero that the algorithm treats them as zeros. Additionally, $\lambda(\widehat{\mathbf{X}}_k)$ is the same as $\lambda(\mathbf{X}_k)$ in the first two steps and is far smaller than $\lambda(\mathbf{X}_k)$ in the last five steps, which means that the filtering method significantly improves the sparsity of matrices during the iterative process. This numerical observation is consistent with the prediction of the theorems in Section 3.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.2
Comparisons of $\lambda(\widehat{\mathbf{X}}_k)$, $\lambda(\mathbf{X}_k)$, *and* $\frac{3^k+1}{2}\lambda(\mathbf{A})$.

k	1	2	3	4	5	6	7
$\lambda(\widehat{\mathbf{X}}_k)$	2	4	8	10	10	10	8
$\lambda(\mathbf{X}_k)$	2	4	10	28	82	206	232
$\frac{3^k+1}{2}\lambda(\mathbf{A})$	2	4	10	28	82	244	730

TABLE 5.3
Comparisons of SIFI_F, SIFI, and sqrtm.

SIFI_F		SIFI		sqrtm	
err	time (s)	err	time (s)	err	time (s)
7.62E-15	0.12	1.86E-15	75.44	9.94E-13	14.93

The computational times and the errors corresponding to the SIFI_F method, the SIFI method, and the sqrtm function are listed in Table 5.3. It can be seen that SIFI_F performs much better than SIFI and the sqrtm function in both computational efficiency and accuracy, which reflects the potential of SIFI_F in the computation of matrix square roots with sparse approximations.

5.3. Performance of SIFI_F. In this experiment, 45 sparse adjacency matrices[1] are used to test the performance of SIFI_F. The dimensions of these matrices range from 10000 to 343791. As it cannot be guaranteed that the adjacency matrix has a principal square root, we actually use the matrix $\mathbf{A}_i = \mathbf{I} - 0.5\mathbf{B}_i/\rho(\mathbf{B}_i)$, where $\mathbf{B}_i$ is the original adjacency matrix. The SIFI_F method with relative error tolerance $\varepsilon_{\text{tol}} = 10^{-14}$ is used for the computation of these matrices. To test the computational efficiency of SIFI_F, the sqrtm function is also evaluated here. As these matrices are too large, the DB method, the IN method, the iterative scheme (2.3), and SIFI are not suitable. The sqrtm function is only suitable for the first 14 matrices of the 45 matrices and fails for the last 31 matrices due to insufficient memory.

The computational time and error comparisons between SIFI_F and sqrtm are displayed in Figure 5.2(a) and 5.2(b), respectively. It can be observed that SIFI_F performs better than sqrtm in terms of both computational accuracy and efficiency.

In Table 5.4, the dimension and sparsity of the 31 matrices are listed in columns 2 and 3. The relative errors of the principal matrix square roots obtained by SIFI_F and the time spent in the computation are listed in columns 4 and 5. The sparsity of $\widehat{\mathbf{X}}_k$ is listed in the last column.

From Table 5.4 we find that SIFI_F can compute the square roots of large-scale sparse matrices efficiently and accurately. Due to the use of adaptive filtering, the obtained principal square root matrix is also sparse, indicating that SIFI_F is highly suitable for the computation when the ε-bandwidth of $\mathbf{A}^{1/2}$ is much smaller than the real bandwidth.

6. Conclusion. In this study, a new stable iterative method that fully avoids matrix inversions (denoted by SIFI) is proposed, based on the Newton iteration. The corresponding proof of the numerical stability of the method is given. The selection of an appropriate initial matrix for the iteration is also suggested, which broadens the applicable matrix range of the original Newton iteration. The sparsity of the principal matrix square root is discussed by using the real bandwidth and the ε-bandwidth. The analysis shows that there exists a sparser

[1] The matrices are obtained from https://networkrepository.com.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

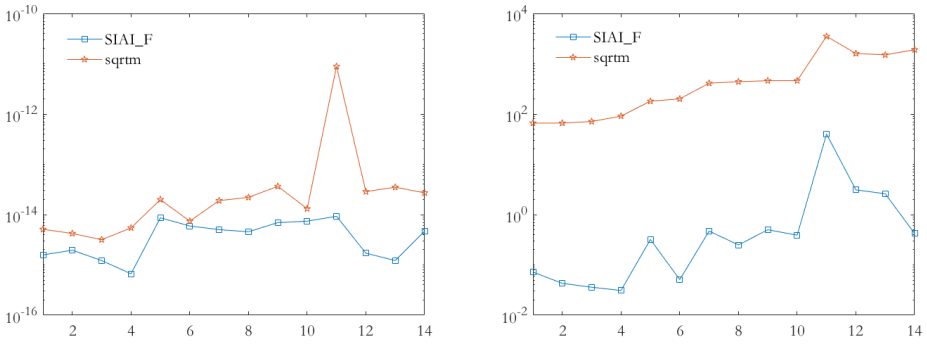
TABLE 5.4
Experiment results of $\mathbf{A}_{15} \sim \mathbf{A}_{45}$.

No.	Dimension	Sparsity ($\mathbf{A}$)	err	Time (s)	Sparsity ($\widehat{\mathbf{X}}_k$)
$\mathbf{A}_{15}$	38000	1.22E-04	8.02e-15	3.20E+00	2.30E-03
$\mathbf{A}_{16}$	38120	8.35E-04	1.06e-14	7.90E+01	4.50E-03
$\mathbf{A}_{17}$	38744	1.20E-03	7.43e-15	1.40E+01	3.70E-03
$\mathbf{A}_{18}$	43471	1.09E-04	6.39e-15	2.70E+00	1.30E-03
$\mathbf{A}_{19}$	43644	1.40E-04	2.89e-15	3.00E+00	1.80E-03
$\mathbf{A}_{20}$	43747	1.09E-04	6.39e-15	2.60E+00	1.30E-03
$\mathbf{A}_{21}$	56468	9.68E-05	6.11e-15	1.30E+02	8.50E-03
$\mathbf{A}_{22}$	57735	9.98E-05	2.18e-15	5.50E-01	9.95E-05
$\mathbf{A}_{23}$	68121	5.14E-04	6.17e-15	1.50E+00	1.89E-04
$\mathbf{A}_{24}$	83995	7.19E-05	1.43e-15	1.50E+00	3.53E-04
$\mathbf{A}_{25}$	91813	2.58E-05	7.92e-15	5.60E-01	7.87E-05
$\mathbf{A}_{26}$	116670	2.60E-05	5.07e-15	1.50E+00	1.69E-04
$\mathbf{A}_{27}$	122494	2.58E-05	5.33e-15	2.40E+00	2.66E-04
$\mathbf{A}_{28}$	122750	2.30E-05	7.56e-15	1.70E+00	1.49E-04
$\mathbf{A}_{29}$	126483	2.40E-05	4.72e-15	1.80E+00	1.73E-04
$\mathbf{A}_{30}$	127998	2.37E-05	6.52e-15	1.80E+00	1.72E-04
$\mathbf{A}_{31}$	131488	2.30E-05	7.19e-15	2.70E+00	2.67E-04
$\mathbf{A}_{32}$	131490	1.98E-05	3.72e-15	2.10E+00	1.59E-04
$\mathbf{A}_{33}$	135276	2.25E-05	6.83e-15	2.00E+00	1.61E-04
$\mathbf{A}_{34}$	136239	2.23E-05	4.60e-15	1.80E+00	1.48E-04
$\mathbf{A}_{35}$	140999	2.15E-05	5.16e-15	2.10E+00	1.51E-04
$\mathbf{A}_{36}$	147772	2.06E-05	5.47e-15	2.30E+00	1.52E-04
$\mathbf{A}_{37}$	152773	1.99E-05	6.24e-15	2.40E+00	1.44E-04
$\mathbf{A}_{38}$	153563	1.98E-05	5.60e-15	2.30E+00	1.45E-04
$\mathbf{A}_{39}$	158058	1.93E-05	5.77e-15	2.40E+00	1.42E-04
$\mathbf{A}_{40}$	161070	7.82E-05	7.46e-15	1.20E+01	2.03E-04
$\mathbf{A}_{41}$	162650	1.45E-05	7.94e-15	1.50E+00	7.52E-05
$\mathbf{A}_{42}$	172175	1.77E-05	5.69e-15	2.60E+00	1.32E-04
$\mathbf{A}_{43}$	203954	2.33E-05	2.40e-15	2.50E+00	8.55E-05
$\mathbf{A}_{44}$	245874	5.51E-06	6.49e-15	6.00E-01	1.07E-05
$\mathbf{A}_{45}$	343791	7.29E-06	9.04e-15	1.40E+00	1.58E-05

matrix approximation with similar accuracy for each matrix involved in every iteration step of SIFI. The filtering technique with an adaptive filtering threshold based on the error analysis is proposed to obtain such a sparser matrix approximation. Combining the filtering technique with SIFI yields the method SIFI_F. Numerical examples show that the SIFI method is more efficient than the Newton iteration, the DB method, and the IN method. The SIFI_F method can efficiently and accurately compute the principal square root of a matrix when it is nearly sparse.

According to the analysis in Section 2, the efficiency of the proposed algorithm depends on the ε-bandwidth of $\mathbf{A}^{1/2}$. If the ε-bandwidth of $\mathbf{A}^{1/2}$ is not significantly smaller than the dimension of the input matrix, then the filtering technique provides only a very limited improvement in the computational efficiency. However, we believe our approach is a valuable attempt for the computation of the square roots of sparse matrices. In the future, we will try to extend the algorithm to the computation of the matrix p-th root.

ETNA
Kent State University and
Johann Radon Institute (RICAM)



(a) Computational error (b) Computational time

FIG. 5.2. *Comparison of SIFI_F with sqrtm with respect to computational accuracy. The horizontal axes of these two figures represent the matrix numbers* $\mathbf{A}_1 \sim \mathbf{A}_{14}$*, the vertical axis of (a) is the computational error and that of (b) is the computational time.*

REFERENCES

[1] A. H. AL-MOHY AND N. J. HIGHAM, *Improved inverse scaling and squaring algorithms for the matrix logarithm*, SIAM J. Sci. Comput., 34 (2012), pp. C153–C169.
[2] S. AMAT, J. A. EZQUERRO, AND M. A. HERNÁNDEZ-VERÓN, *Iterative methods for computing the matrix square root*, SeMA J., 70 (2015), pp. 11–21.
[3] ———, *On a new family of high-order iterative methods for the matrix pth root*, Numer. Linear Algebra Appl., 22 (2015), pp. 585–595.
[4] R. ANIL, V. GUPTA, T. KOREN, K. REGAN, AND Y. SINGER, *Scalable second order optimization for deep learning*, Preprint on arXiv, 2021, https://arxiv.org/abs/2002.09018
[5] M. BENZI AND G. H. GOLUB, *Bounds for the entries of matrix functions with applications to preconditioning*, BIT, 39 (1999), pp. 417–438.
[6] M. BENZI AND N. RAZOUK, *Decay bounds and* $O(n)$ *algorithms for approximating functions of sparse matrices*, Electron. Trans. Numer. Anal., 28 (2007/08), pp. 16–39. https://etna.ricam.oeaw.ac.at/vol.28.2007-2008/pp16-39.dir/pp16-39.pdf
[7] D. A. BINI, N. J. HIGHAM, AND B. MEINI, *Algorithms for the matrix pth root*, Numer. Algorithms, 39 (2005), pp. 349–378.
[8] A. BJÖRCK AND S. HAMMARLING, *A Schur method for the square root of a matrix*, Linear Algebra Appl., 52/53 (1983), pp. 127–140.
[9] G. W. CROSS AND P. LANCASTER, *Square roots of complex matrices*, Linear and Multilinear Algebra, 1 (1973/74), pp. 289–293.
[10] P. I. DAVIES, N. J. HIGHAM, AND F. TISSEUR, *Analysis of the Cholesky method with iterative refinement for solving the symmetric definite generalized eigenproblem*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 472–493.
[11] E. DEADMAN, N. J. HIGHAM, AND R. RALHA, *Blocked Schur algorithms for computing the matrix square root*, in Applied Parallel and Scientific Computing (PARA 2012), P. Manninen and P. Öster, eds., Lecture Notes in Computer Science, Vol. 7782, Springer, Berlin, 2012, pp. 171–182.
[12] S. DEMKO, W. F. MOSS, AND P. W. SMITH, *Decay rates for inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499.
[13] E. D. DENMAN AND A. N. BEAVERS, JR., *The matrix sign function and computations in systems*, Appl. Math. Comput., 2 (1976), pp. 63–94.
[14] S. FENG, *Kernel pooling feature representation of pre-trained convolutional neural networks for leaf recognition*, Multimed. Tools Appl., 81 (2022), pp. 4255–4282.
[15] N. J. HIGHAM, *Newton's method for the matrix square root*, Math. Comp., 46 (1986), pp. 537–549.
[16] ———, *Computing real square roots of a real matrix*, Linear Algebra Appl., 88/89 (1987), pp. 405–430.
[17] ———, *Stable iterations for the matrix square root*, Numer. Algorithms, 15 (1997), pp. 227–242.
[18] ———, *Functions of Matrices*, SIAM, Philadelphia, 2008.
[19] W. HOSKINS AND D. WALTON, *A faster method of computing the square root of a matrix*, IEEE Trans. Automat. Control, 23 (1978), pp. 494–495.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[20] B. IANNAZZO, *A note on computing the matrix square root*, Calcolo, 40 (2003), pp. 273–283.
[21] J. D. LAWSON AND Y. LIM, *The geometric mean, matrices, metrics, and more*, Amer. Math. Monthly, 108 (2001), pp. 797–812.
[22] P. LI, J. XIE, Q. WANG, AND Z. GAO, *Towards faster training of global covariance pooling networks by iterative matrix square root normalization*, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Conference Proceedings, Los Alamitos, 2018, pp. 947–955.
[23] B. MEINI, *The matrix square root from a new functional perspective: theoretical results and computational issues*, SIAM J. Matrix Anal. Appl., 26 (2004/05), pp. 362–376.
[24] M. MOMENZADEH AND T. LOTFI, *New iterative methods for finding matrix sign function: derivation and application*, Comput. Appl. Math., 38 (2019), Paper No. 53, 15 pages.
[25] Y. SONG, N. SEBE, AND W. WANG, *Why approximate matrix square root outperforms accurate SVD in global covariance pooling?*, in 2021 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Conference Proceedings, Los Alamitos, 2021, pp. 1095–1103.
[26] H. WANG AND X. LIU, *The polar-like decomposition and its applications*, Filomat, 33 (2019), pp. 3977–3983.
[27] F. WU, Q. GAO, AND W.-X. ZHONG, *Fast precise integration method for hyperbolic heat conduction problems*, Appl. Math. Mech. (English Ed.), 34 (2013), pp. 791–800.
[28] F. WU, K. ZHANG, L. ZHU, AND J. HU, *High-performance computation of the exponential of a large sparse matrix*, SIAM J. Matrix Anal. Appl., 42 (2021), pp. 1636–1655.