Numerical Algorithms 15 (1997) 227–242

Stable iterations for the matrix square root

Nicholas J. Higham

Department of Mathematics, University of Manchester, Manchester M13 9PL, England
E-mail: higham@ma.man.ac.uk

Received 28 April 1997
Communicated by C. Brezinski

Any matrix with no nonpositive real eigenvalues has a unique square root for which every eigenvalue lies in the open right half-plane. A link between the matrix sign function and this square root is exploited to derive both old and new iterations for the square root from iterations for the sign function. One new iteration is a quadratically convergent Schulz iteration based entirely on matrix multiplication; it converges only locally, but can be used to compute the square root of any nonsingular M-matrix. A new Padé iteration well suited to parallel implementation is also derived and its properties explained. Iterative methods for the matrix square root are notorious for suffering from numerical instability. It is shown that apparently innocuous algorithmic modifications to the Padé iteration can lead to instability, and a perturbation analysis is given to provide some explanation. Numerical experiments are included and advice is offered on the choice of iterative method for computing the matrix square root.

Keywords: matrix square root, matrix logarithm, matrix sign function, M-matrix, symmetric positive definite matrix, Padé approximation, numerical stability, Newton's method, Schulz method

AMS subject classification: 65F30

1. Introduction

Any nonsingular matrix $A \in \mathbb{C}^{n\times n}$ has a square root, that is, the equation $A = X^2$ has a solution. The number of square roots varies from two (for a nonsingular Jordan block) to infinity (any involutary matrix is a square root of the identity matrix). If A is singular, the existence of a square root depends on the Jordan structure of the zero eigenvalues [7], [21, theorem 6.4.12]. If A is real, it may or may not have a real square root; a sufficient condition for one to exist is that A have no real negative eigenvalues [14], [21, theorem 6.4.14].

Although the theory of matrix square roots is rather complicated, simplifications occur for certain classes of matrices. Consider, for example, symmetric positive (semi)definite matrices. Any such matrix has a unique symmetric positive (semi)definite square root (see, for example, [20, theorem 7.2.6]), and this square root finds use in, for example, the theory of the generalized eigenproblem [32, sec-

© J.C. Baltzer AG, Science Publishers

tion 15-10], and preconditioned iterative methods [11,23]. More generally, any matrix A having no nonpositive real eigenvalues has a unique square root for which every eigenvalue has positive real part, and it is this square root, denoted by $A^{1/2}$ and sometimes called the principal square root, that is usually of interest (see, for example, the application in boundary value problems [35]). To recap, $A^{1/2}$ is defined by

$$\left(A^{1/2}\right)^2 = A \quad \text{and} \quad \operatorname{Re}\lambda_k\left(A^{1/2}\right) > 0 \text{ for all } k, \tag{1.1}$$

where A has no nonpositive real eigenvalues and $\lambda_k(A)$ denotes an eigenvalue of A. A class of matrices having no nonpositive real eigenvalues is the much-studied class of nonsingular M-matrices. A matrix $A \in \mathbb{R}^{n\times n}$ is a nonsingular M-matrix if it can be written in the form

$$A = sI - B, \quad B \geqslant 0, \ s > \rho(B),$$

where ρ denotes the spectral radius and matrix inequalities hold componentwise. This is just one of many equivalent conditions for A to be a nonsingular M-matrix [3, theorem 2.3].

In this work we are concerned with how to compute matrix square roots. An example of an application in which matrix square roots are required is the computation of the matrix logarithm by Padé approximants. The identity

$$\log A = 2^k \log A^{1/2^k}$$

is used to reduce the logarithm argument to a matrix close to the identity, so that a Padé approximation to $\log(I + X)$ can be applied, where $X = A^{1/2^k} - I$ [9,24,25]. This method requires k successive square roots to be computed before evaluating the logarithm.

The method of reference for computing matrix square roots is to compute a Schur decomposition, compute a square root of the triangular factor, and then transform back. This method, with a recurrence for finding a square root of a triangular matrix, was suggested by Björck and Hammarling [4], and Higham [14] showed how the computations could be performed entirely in real arithmetic when computing a real square root of a real matrix. The Schur method is numerically stable. Matrix iterations $X_{k+1} = g(X_k)$, where g is a polynomial or a rational function, are attractive alternatives for computing square roots for two reasons: they are readily implemented in terms of standard "building blocks", and they are potentially well suited to parallel computation. The Newton iteration[1]

$$X_{k+1} = \tfrac{1}{2}\left(X_k + X_k^{-1}A\right), \qquad X_0 = A, \tag{1.2}$$

is well known and has good theoretical properties. If A has no nonpositive real eigenvalues then, provided all the iterates in (1.2) are defined, X_k converges quadratically to $A^{1/2}$ [13]. If A is symmetric positive definite then so is each iterate X_k. Unfortunately, this Newton iteration has such poor numerical stability that it is useless for practical computation. The instability was observed by Laasonen [30] and rigorously explained by Higham [13]. Analysis shows that unless the eigenvalues λ_i of A satisfy the very strong requirement that

---

[1] Iteration (1.2) is Newton's method for the equation $X^2 - A = 0$, with $X_0 = A$, with the equations rewritten to exploit the fact that $AX_k = X_kA$ for all k.

$$\tfrac{1}{2}\left|1-(\lambda_j/\lambda_i)^{1/2}\right| \leqslant 1, \quad i,j=1,\dots,n,$$

then small errors in the iterates can be amplified by the iteration, with the result that in floating point arithmetic the iteration fails to converge.

An alternative iteration was derived by Denman and Beavers [8] as a special case of a method for solving the algebraic Riccati equation:

$$Y_0 = A, \qquad Z_0 = I, \tag{1.3a}$$

$$\left.\begin{aligned} Y_{k+1} &= \tfrac{1}{2}\left(Y_k + Z_k^{-1}\right) \\ Z_{k+1} &= \tfrac{1}{2}\left(Z_k + Y_k^{-1}\right) \end{aligned}\right\} \quad k = 0,1,2,\dots . \tag{1.3b}$$

It can be shown that the sequences generated from (1.2) and (1.3) are related by

$$Y_k = X_k, \qquad Z_k = A^{-1}X_k, \quad k = 1,2,\dots ,$$

so that, provided A has no nonpositive real eigenvalues and all the iterates are defined,

$$Y_k \to A^{1/2}, \qquad Z_k \to A^{-1/2} \quad \text{as } k \to \infty,$$

with a quadratic rate of convergence. We will refer to (1.3) as the "DB iteration". For a general matrix, if we implement the iterations using LU factorization then the DB iteration (1.3) requires 50 percent more arithmetic operations per iteration than the Newton iteration (1.2), but for symmetric positive definite matrices the operation counts are the same if we make use of Cholesky factorizations. The DB iteration requires more storage than the Newton iteration, but this is not a major drawback since n is not usually very large in practice. The crucial property of the DB iteration is that it propagates errors in a stable fashion [13].

In the next section we show how a simple identity involving the matrix sign function allows any iteration for computing the sign function to be converted into an iteration for the matrix square root. This approach allows a simple derivation of the DB iteration and, more importantly, leads to some new numerically stable iterations for the matrix square root.

In section 3 we briefly consider the use of scaling to speed convergence. In section 4 we examine the special cases of M-matrices and symmetric positive definite matrices, comparing the iterations derived here with two others from the literature. Numerical experiments are presented in section 5, and in section 6 we give some recommendations on the choice of method for computing the matrix square root.

2. From the matrix sign function to the square root

The matrix sign function has been the subject of much research in recent years because of its usefulness as a tool in control theory and in solving nonsymmetric eigenproblems. We present a minimum of definitions and background and refer the reader to Higham [16] and Kenney and Laub [29] for further details.

The matrix sign function extends the notion of the sign (±1) of a complex number to matrices, and is defined only for $A \in \mathbb{C}^{n\times n}$ having no pure imaginary eigenvalues. For our purposes the most useful definition is

$$\operatorname{sign}(A) = A\bigl(A^2\bigr)^{-1/2}. \tag{2.1}$$

Note that, under the stated conditions on A, A^2 has no nonpositive real eigenvalues, so that $(A^2)^{-1/2}$ is defined (see (1.1)).

The following lemma relates the sign of a certain block 2×2 matrix to the matrix square root.

Lemma 1. For $A \in \mathbb{C}^{n\times n}$ having no nonpositive real eigenvalues,

$$\operatorname{sign}\left(\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix}. \tag{2.2}$$

Proof. First, note that the eigenvalues of the block 2×2 matrix on the left are plus and minus the square roots of the eigenvalues of A, hence they are not pure imaginary and the sign of this matrix is defined. Applying (2.1) we have

$$\begin{aligned}\operatorname{sign}\left(\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}\right) &= \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}\begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix}^{-1/2} = \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}\begin{bmatrix} A^{-1/2} & 0 \\ 0 & A^{-1/2} \end{bmatrix} \\ &= \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix},\end{aligned}$$

as required. □

A standard iteration for computing sign(B) is the quadratically convergent Newton iteration of Roberts [34]

$$X_{k+1} = \tfrac{1}{2}\bigl(X_k + X_k^{-1}\bigr), \qquad X_0 = B, \tag{2.3}$$

which is essentially (1.2) used to compute the square root of the identity, but with a different starting matrix. If we apply this iteration to

$$B = \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix} \tag{2.4}$$

then we know from lemma 1 that it converges to the matrix (2.2). We find that the iterates have the form

$$X_k = \begin{bmatrix} 0 & Y_k \\ Z_k & 0 \end{bmatrix}, \tag{2.5}$$

where Y_k and Z_k satisfy (1.3). Therefore we have rederived the DB iteration.

Now we derive some new iterations for the square root. The Schulz iteration for sign(B) has the form

$$X_{k+1} = \tfrac{1}{2}X_k\bigl(3I - X_k^2\bigr), \qquad X_0 = B.$$

It converges quadratically for $\|B^2 - I\| < 1$, where the norm is any consistent matrix norm [26, theorem 5.2]. Applying the iteration to (2.4) leads to the coupled iteration

$$Y_0 = A, \qquad Z_0 = I, \tag{2.6a}$$

$$\left.\begin{aligned} Y_{k+1} &= \tfrac{1}{2}Y_k(3I - Z_kY_k) \\ Z_{k+1} &= \tfrac{1}{2}(3I - Z_kY_k)Z_k \end{aligned}\right\} \quad k = 0, 1, 2, \ldots. \tag{2.6b}$$

Provided that $\|\operatorname{diag}(A - I, A - I)\| < 1$, we have quadratic convergence of Y_k to $A^{1/2}$ and Z_k to $A^{-1/2}$. This Schulz iteration for the square root has the advantage that it involves only matrix multiplication, an operation that can be implemented very efficiently on most high-performance computers. The iteration (2.6) requires less work per iteration than the DB iteration (1.3) if we can perform matrix multiplication at more than 1.5 times the rate of matrix inversion (a ratio that arises in a similar comparison between iterations for the polar decomposition [19]). It is easily seen that $Y_k = AZ_k$ for all k, but if we use this relation to reduce (2.6) to an iteration in a single variable Y_k or Z_k we reintroduce the need to compute an inverse.

Kenney and Laub [26] derive a family of iterations for computing sign(B) based on Padé approximations to the function $f(x) = (1 - x)^{-1/2}$. They have the form

$$X_{k+1} = X_k p_r\bigl(X_k^2\bigr)q_s\bigl(X_k^2\bigr)^{-1}, \qquad X_0 = B,$$

where p_r and q_s are polynomials of degree r and s respectively. Pandey et al. [31] derive an explicit partial fraction form for the iteration with $r = s - 1$ (see also Kenney and Laub [28] for an alternative derivation). The iteration is

$$X_{k+1} = \frac{1}{p}X_k \sum_{i=1}^{p} \frac{1}{\xi_i}\bigl(X_k^2 + \alpha_i^2 I\bigr)^{-1}, \qquad X_0 = B, \tag{2.7}$$

where

$$\xi_i = \frac{1}{2}\left(1 + \cos\frac{(2i - 1)\pi}{2p}\right), \qquad \alpha_i^2 = \frac{1}{\xi_i} - 1, \quad i = 1, \ldots, p.$$

On applying the iteration to B in (2.4) we find that the iterates have the form (2.5) and we obtain the following Padé iteration for the square root:

$$Y_0 = A, \qquad Z_0 = I, \tag{2.8a}$$

$$\left.\begin{aligned} Y_{k+1} &= \frac{1}{p} Y_k \sum_{i=1}^{p} \frac{1}{\xi_i} \left(Z_k Y_k + \alpha_i^2 I\right)^{-1} \\ Z_{k+1} &= \frac{1}{p} Z_k \sum_{i=1}^{p} \frac{1}{\xi_i} \left(Y_k Z_k + \alpha_i^2 I\right)^{-1} \end{aligned}\right\} \quad k = 0, 1, 2, \ldots. \tag{2.8b}$$

Several properties of this iteration are worth noting. The first and third properties can be deduced from the theory for (2.7) [26,28,31].

(1) In iteration (2.8), $Y_k \to A^{1/2}$ and $Z_k \to A^{-1/2}$ for any A having no nonpositive real eigenvalues, with order of convergence $2p$.

(2) One step of iteration (2.8) with $p = 1$ gives

$$Y_{k+1} = 2Y_k(Z_k Y_k + I)^{-1}, \qquad Z_{k+1} = 2Z_k(Y_k Z_k + I)^{-1}. \tag{2.9}$$

This is closely related to the DB iteration (1.3), in that Y_k from (2.9) equals the inverse of Y_k from (1.3), with $Y_0 = A^{-1}$ in (1.3a), and likewise for the Z_ks. The DB iteration (1.3) is generally to be preferred to (2.9) since it requires much less work per iteration.

(3) If p is a power of 2, one step of iteration (2.8) yields the matrices from $\log_2 p + 1$ steps of the same iteration with $p = 1$ (that is, (2.9)).

The Padé iteration (2.8) is attractive for several reasons. First, it is very well suited for parallel computation, since the $2p$ matrix inverses required on each iteration can be done in parallel. See [31] for results on parallel implementation of (2.7) and [18] for results on parallel implementation of a variant of (2.7) that computes the unitary polar factor of a matrix. The second important feature of the Padé iteration is that it is numerically stable, inheriting its stability from the sign iteration from which it is derived. However, this stability is easily lost if we try to manipulate the iteration, as we now explain.

It is easy to see that the Y_k and Z_k generated by (2.8) are both rational functions of A for all k, and hence they commute. We can therefore rewrite (2.8b) as

$$\left.\begin{aligned} Y_{k+1} &= \frac{1}{p} Y_k \sum_{i=1}^{p} \frac{1}{\xi_i} \left(Y_k Z_k + \alpha_i^2 I\right)^{-1} \\ Z_{k+1} &= \frac{1}{p} Z_k \sum_{i=1}^{p} \frac{1}{\xi_i} \left(Y_k Z_k + \alpha_i^2 I\right)^{-1} \end{aligned}\right\} \quad k = 0, 1, 2, \ldots, \tag{2.10}$$

in which there are p matrix inversions per iteration instead of $2p$. We can go further and use the relation $Y_k = AZ_k$ to obtain the single iteration

$$Y_{k+1} = \frac{1}{p} Y_k \sum_{i=1}^{p} \frac{1}{\xi_i} \left(Y_k A^{-1} Y_k + \alpha_i^2 I\right)^{-1}, \qquad Y_0 = A, \tag{2.11}$$

though this is hardly an improvement since it requires the same number of arithmetic operations per iteration as (2.10).

Iterations (2.10) and (2.11) are both, like the Newton iteration (1.2), of no practical use because of numerical instability. In the appendix we give a short perturbation analysis to explain why (2.10) has such different numerical stability properties to (2.8), and we give a numerical illustration in section 5.

3. Scaling

Although the methods described in the previous section have at least quadratic rates of convergence, rapid convergence is not guaranteed in practice because the error can take many iterations to become small enough for quadratic convergence to be observed. Scaling to reduce the number of iterations is a much-studied and well understood topic for iterations for the matrix sign function and the polar decomposition; see [27] and the references therein. Since all the iterations in the last section are derived or directly related to the Newton iteration (2.3) for the sign function, scalings for this Newton iteration translate directly to scalings for the square root iterations.

The scaled Newton iteration for sign(B) has the form

$$X_{k+1} = \tfrac{1}{2}\bigl(\gamma_k X_k + \gamma_k^{-1} X_k^{-1}\bigr), \qquad X_0 = B, \tag{3.1}$$

which corresponds to replacing X_k by $\gamma_k X_k$, for some scaling parameter γ_k, at the start of each iteration. We consider just one of the various scaling schemes: the determinantal scaling of Byers [6], which takes

$$\gamma_k = \bigl|\det(X_k)^{-1/n}\bigr|.$$

This scaling parameter is easily formed during the computation of X_k^{-1}.

To translate the scaling to the DB iteration (1.3) for the square root, we use the connection between these two iterations established in section 2. From (2.5) we have

$$\det(X_k) = (-1)^n \det(Y_k)\det(Z_k),$$

so the scaled version of the DB iteration is

$$Y_0 = A, \qquad Z_0 = I, \tag{3.2a}$$

$$\left.\begin{aligned} \gamma_k &= \bigl|\bigl(\det(Y_k)\det(Z_k)\bigr)^{-1/(2n)}\bigr| \\ Y_{k+1} &= \tfrac{1}{2}\bigl(\gamma_k Y_k + \gamma_k^{-1} Z_k^{-1}\bigr) \\ Z_{k+1} &= \tfrac{1}{2}\bigl(\gamma_k Z_k + \gamma_k^{-1} Y_k^{-1}\bigr) \end{aligned}\right\} \quad k = 0, 1, 2, \ldots. \tag{3.2b}$$

Again, γ_k is easily computed during the inversion of Y_k and Z_k. We mention in passing that a different scaling strategy for (1.3) is proposed, without explanation, in [22]; it requires estimates of the spectral radii of A and A^{-1}.

It is not hard to see that Byers' scaling strategy is applicable also to the Padé iteration, yielding the scaled iteration

$$Y_0 = A, \qquad Z_0 = I, \tag{3.3a}$$

$$\left.\begin{aligned} \gamma_k &= \left|\big(\det(Y_k)\det(Z_k)\big)^{-1/(2n)}\right| \\ Y_{k+1} &= \frac{1}{p}\gamma_k Y_k \sum_{i=1}^{p} \frac{1}{\xi_i}\big(\gamma_k^2 Z_k Y_k + \alpha_i^2 I\big)^{-1} \\ Z_{k+1} &= \frac{1}{p}\gamma_k Z_k \sum_{i=1}^{p} \frac{1}{\xi_i}\big(\gamma_k^2 Y_k Z_k + \alpha_i^2 I\big)^{-1} \end{aligned}\right\} \quad k = 0, 1, 2, \ldots . \tag{3.3b}$$

Here, the computation of the scaling factor is a significant added expense for small values of p.

4. M-matrices and symmetric positive definite matrices

In this section we consider the important classes of M-matrices and symmetric positive definite matrices and compare our new iterations with some existing ones.

Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular M-matrix, so that

$$A = sI - B, \quad B \geqslant 0, \ s > \rho(B).$$

This representation is not unique, but for computational purposes we can take $s = \max_i a_{ii}$. Then we can write

$$A = s(I - C), \quad C = s^{-1}B \geqslant 0, \ \rho(C) < 1, \tag{4.1}$$

and our task reduces to finding $(I - C)^{1/2}$. Since $\rho(C) < 1$, there is a consistent norm such that $\|C\| < 1$, by a standard result [20, lemma 5.6.10]. Hence the Schulz iteration (2.6) is guaranteed to converge when applied to $I - C$, and so we can efficiently compute the square root of an M-matrix using only matrix multiplication.

Another iteration has been used in the literature to prove theoretical results about M-matrices and quadratic matrix equations [2,5]. The idea is to use (4.1) and to iterate

$$P_{k+1} = \tfrac{1}{2}\big(C + P_k^2\big), \qquad P_0 = 0. \tag{4.2}$$

Let

$$(I - C)^{1/2} = I - P. \tag{4.3}$$

A simple induction shows that [5, lemma 2.1]

$$0 \leqslant P_k \leqslant P_{k+1} \leqslant P, \quad k \geqslant 0. \tag{4.4}$$

We also find that $\rho(P) \leqslant \rho(C) < 1$. Since the sequence P_k is nondecreasing and bounded above it has a limit, Q say, which must satisfy $(I - Q)^2 = I - C$. But (4.4) implies $\rho(Q) \leqslant \rho(P) < 1$, which means that $I - Q$ has eigenvalues with positive real part and hence equals $(I - C)^{1/2} = I - P$. Hence $P_k \to P$ as $k \to \infty$.

From (4.2) and the square of (4.3) we have, since P and P_k are both polynomials in C and hence commute,

$$P - P_{k+1} = \tfrac{1}{2}(P + P_k)(P - P_k).$$

Using a consistent norm for which $\|P\| < 1$ we obtain

$$\|P - P_{k+1}\| \leqslant \theta_k \|P - P_k\|, \quad \theta_k = (\|P\| + \|P_k\|)/2 \tag{4.5}$$

and $\theta_k \to \|P\| < 1$ as $k \to \infty$. Hence iteration (4.2) converges linearly to P in (4.3). The convergence is too slow for the iteration to be of practical use unless $\|C\|$ is less than, say, $1/2$, hence the Schulz iteration (2.6) is preferable in general.

The iteration (4.2) has also been used for symmetric positive definite matrices A by Albrecht [1] and (in a more general form) Pulay [33]. Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and let λ_i denote the ith largest eigenvalue of A, so that the 2-norm condition number $\kappa_2(A) = \lambda_1/\lambda_n$. Then we can write

$$A = s(I - C), \quad s = (\lambda_1 + \lambda_n)/2, \qquad \rho(C) = \|C\|_2 = \frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} < 1,$$

where the given value of s is easily shown to minimize $\|C\|_2$. The error bound (4.5) is still valid and $\|C\|_2 < 1$, so the iteration (4.2) can be used to compute $(I - C)^{1/2}$. However, unless A is extremely well conditioned ($\kappa_2(A) < 3$, say) convergence will be exceedingly slow.

For symmetric positive definite matrices, the best alternative iteration to the iterations of section 2 is the following method of Higham [12].

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

1. $A = R^T R$ (Cholesky factorization).
2. Compute $U = X_\infty$ from $X_{k+1} = \frac{1}{2}(X_k + X_k^{-T})$, $k = 1, 2, \ldots$, with $X_0 = R$.
3. $X = U^T R$.

The iteration in step 2 is a Newton iteration for computing U in the polar decomposition $R = UH$ (U orthogonal, H symmetric positive definite); thus $A = R^T R = HU^T UH = H^2$. This Newton iteration requires one nonsymmetric matrix inversion per iteration, which is equivalent to the two symmetric positive definite inversions per iteration required by the DB iteration (1.3). The iteration is stable and is readily accelerated by the incorporation of scaling parameters [12,27]. Algorithm 2 can be extended to deal with positive *semi*definite matrices by using a Cholesky factorization with pivoting [19].

5. Numerical experiments

To give further insight into the methods, we present the results of some numerical experiments. All computations were performed in MATLAB, which has unit roundoff $u = 2^{-53} \approx 1.1 \times 10^{-16}$. We use matrices from the Test Matrix Toolbox [15], [17, appendix E].

For each method we monitored convergence using the relative residual

$$\mathrm{res}(X_k) = \frac{\|A - X_k^2\|_2}{\|A\|_2},$$

where X_k denotes the iterate converging (in theory) to $A^{1/2}$. Note that res can be regarded as the backward error of X_k. This is not a quantity that one would use in practice, because it requires the computation of X_k^2 (and expensive 2-norms), but it is useful for judging the behaviour of the methods. We report the point at which res approximately reached a minimum and stopped decreasing significantly.

For nonsymmetric matrices it is not reasonable to expect res to reach the unit roundoff level. To see why, let $X = A^{1/2}$ and consider $\widetilde{X} = X + E$, where $\|E\|_2 \leqslant \varepsilon\|X\|_2$. If $\varepsilon = u$, we can consider $\widetilde{X}$ to be a rounded approximation to X. We have

$$\mathrm{res}(\widetilde{X}) = \frac{\|-XE - EX - E^2\|_2}{\|A\|_2} \leqslant \frac{(2\varepsilon + \varepsilon^2)\|X\|_2^2}{\|A\|_2}.$$

For a symmetric matrix A, $\|X\|_2^2 = \|A\|_2$, but for nonsymmetric matrices $\|X\|_2^2/\|A\|_2$ can be arbitrarily large. The quantity

$$\alpha(X) = \frac{\|X\|_2^2}{\|A\|_2} \geqslant 1$$

can be loosely regarded as a condition number for the matrix square root; see [14].

We also report

$$\mathrm{err}(X_k) = \frac{\|X - X_k\|_2}{\|X\|_2},$$

where X is the approximation to $A^{1/2}$ computed using the (real) Schur method.

When scaling was in operation it was used only while the relative ∞-norm change in successive iterates exceeded 10^{-2}, because after this point the quadratic convergence leads to rapid convergence. For algorithm 2 we used the stopping criterion from [12], which is based on the relative change in the iterates of the polar decomposition iteration, and scaling was always used.

The first example concerns the block tridiagonal symmetric positive definite M-matrix resulting from discretizing Poisson's equation with the 5-point operator [10, section 4.5.4]. The Schulz iteration (2.6), the unscaled Padé iteration (2.8), iteration (4.2) and the DB iteration (1.3) were applied to $I - C$ using the splitting (4.1) with $s = \max_i a_{ii}$. The results are given in table 1. All the iterations perform well on this

Table 1
Results for Poisson matrix. Dimension $n = 64$, $\rho(C) = 0.94$ in (4.1).

Method	Iters.	res	err
(4.2)	109	7.1e−15	1.8e−14
Schulz (2.6)	9	2.4e−16	5.0e−15
Padé (2.8), $p = 1$	6	4.3e−16	4.9e−15
Padé (2.8), $p = 2$	3	7.9e−16	4.9e−15
Padé (2.8), $p = 3$	3	4.3e−16	4.9e−15
Padé (2.8), $p = 4$	2	7.7e−16	5.1e−15
Algorithm 2	6	5.0e−16	5.0e−15
DB (1.3)	6	1.1e−15	5.0e−15

Table 2
Results for Frank matrix A with $n = 12$; $\alpha(A^{1/2}) = 8.7 \times 10^7$.

Method	Iters.	res	err
Padé unscaled (2.8) $p = 1$	7	2.7e−9	2.1e−9
$p = 2$	4	1.5e−9	2.1e−9
$p = 3$	3	5.0e−10	2.1e−9
$p = 4$	3	3.3e−10	2.1e−9
Padé scaled (3.3) $p = 1$	5	9.1e−9	2.1e−9
$p = 2$	3	2.9e−9	2.1e−9
$p = 3$	4	2.7e−10	2.1e−9
$p = 4$	3	3.4e−10	2.1e−9
“Fast” Padé (2.10) $p = 1$	6	7.9e−3	8.5e−7
$p = 2$	3	6.1e−6	1.5e−8
$p = 3$	3	5.6e−8	2.1e−9
$p = 4$	3	9.6e−8	2.1e−9
DB unscaled (1.3)	7	4.0e−9	2.1e−9
DB scaled (3.2)	5	4.0e−8	2.1e−9

problem, with the exception of the linearly convergent iteration (4.2), which is very slow to converge.

In the second example A is the Frank matrix, an upper Hessenberg matrix with real, positive eigenvalues occurring in reciprocal pairs, the $\lfloor n/2 \rfloor$ smallest of which are ill conditioned. We take $n = 12$, for which the eigenvalues range from 0.29 to 32.2. We tried the original Padé iteration (2.8), both scaled and unscaled, and the modified (“fast”) version (2.10) that we expect from the analysis in the appendix to be unstable. The results, including those for the DB iteration, are given in table 2. The smallest value of res produced by any of the iterations is of order $\alpha(A^{1/2})u$, as would be expected. The modified Padé iteration gives much larger values of res. Furthermore, unlike for the original Padé iteration, for which res(X_k) is approximately constant once the iteration has “converged”, for the modified version res ultimately grows rapidly and becomes many orders of magnitude greater than 1. Scaling brings a slight reduction in the number of iterations, while not affecting the stability.

Table 3
Results for randsvd matrix A with $n = 16$; $\kappa_2(A) = 10^6$.

Method	Iters.	res	err
Padé unscaled (2.8), general $p = 1$	14	5.4e−15	2.5e−14
$p = 2$	8	5.1e−15	2.3e−14
$p = 3$	6	9.5e−15	2.3e−14
$p = 4$	5	4.7e−15	2.3e−14
Padé scaled (3.3), general $p = 1$	7	2.6e−15	2.8e−14
$p = 2$	4	9.6e−14	1.0e−13
$p = 3$	4	5.0e−14	5.4e−14
$p = 4$	3	8.1e−14	8.6e−14
Padé unscaled (2.8), Cholesky $p = 1$	10	4.0e−7	2.3e−4
$p = 2$	6	1.4e−9	5.5e−7
$p = 3$	5	4.2e−10	4.3e−10
$p = 4$	4	1.1e−9	5.5e−7
Padé scaled (3.3), Cholesky $p = 1$	5	1.0e−5	7.4e−6
$p = 2$	4	3.9e−9	3.8e−9
$p = 3$	3	9.4e−10	9.0e−10
$p = 4$	3	4.3e−9	4.2e−9
DB unscaled (1.3), general	13	3.0e−12	1.5e−10
DB scaled (3.2), general	7	2.1e−14	2.4e−14
DB unscaled (1.3), Cholesky	13	2.0e−12	1.5e−10
DB scaled (3.2), Cholesky	7	2.1e−14	2.5e−14
Algorithm 2	8	6.9e−16	2.2e−14

Our third example concerns a random symmetric positive definite 16×16 matrix A with $\kappa_2(A) = 10^6$ generated using the routine `randsvd` from the Test Matrix Toolbox. We computed $A^{1/2}$ using the DB iteration and the Padé iteration in two ways: first without exploiting the definiteness and then using Cholesky factorization for the inverses, trying the scaled and unscaled iterations in both cases. We also applied algorithm 2. The results are given in table 3. There are two striking features of the results. First, scaling greatly speeds convergence for the DB iteration and the Padé iteration with $p = 1, 2$. Second, exploiting the definiteness leads to a marked loss of stability and accuracy for the Padé iteration, though not for the DB iteration. The reason for the poor behaviour of the Padé iteration when definiteness is exploited is not clear; it appears that by enforcing symmetry we lose the satisfaction of the identities such as $Y_k = AZ_k$ that underlie the iteration.

6. Conclusions

Perhaps surprisingly, the theory and practice of iterative methods for computing the matrix square root is less well developed than for the matrix sign function and the polar decomposition. The stable iterations considered here involve coupled iterations whose derivations rest on the connection between the sign function and the square root established in lemma 1. Whether more direct derivations exist is unclear.

A theme of this paper is the tradeoff between speed and stability. The single variable Newton iteration (1.2) is unstable, and the Padé iteration (2.8) becomes unstable when we attempt to reduce the cost of its implementation. Iterations for the matrix square root appear to be particularly delicate with respect to numerical stability.

We conclude with some recommendations on how to compute the matrix square root $A^{1/2}$.

1. If possible, use the Schur method [4,14]. It is numerically stable and allows ready access to square roots other than the one with eigenvalues in the right-half plane that we denote by $A^{1/2}$.
2. If A is symmetric positive definite, the best alternative to the Schur method is algorithm 2.
3. If an iteration using only matrix multiplication is required and $\|A - I\| < 1$ for some consistent norm, use the Schulz iteration (2.6). The convergence condition is satisfied for M-matrices and for symmetric positive definite matrices if the scalings of section 4 are used.
4. If an iteration suitable for parallel implementation is required, use the Padé iteration (2), with a value of p appropriate to the desired degree of parallelism (cf. [18,31]). Do not exploit the definiteness when calculating the matrix inverses. Scaling should be considered when p is small: it is effective but relatively expensive.
5. Iteration (1.3) is recommended in general. It has the advantages that scaling is inexpensive and that exploiting definiteness does not affect the numerical stability.

Appendix

We analyse the numerical stability of the Padé iteration (2.8) and the rewritten version (2.10), for $p = 1$. To reveal the difference between these two iterations it is sufficient to suppose that A is diagonal

$$A = \Lambda = \mathrm{diag}(\lambda_i)$$

(as long as A is diagonalizable, we can in any case diagonalize the iteration and obtain essentially the same equations). We consider "almost converged" iterates

$$\widetilde{Y}_k = \Lambda^{1/2} + E_k, \qquad \widetilde{Z}_k = \Lambda^{-1/2} + F_k$$

and suppose that

$$\widetilde{Y}_{k+1} = \Lambda^{1/2} + E_{k+1}, \qquad \widetilde{Z}_{k+1} = \Lambda^{-1/2} + F_{k+1}$$

are computed exactly from $\widetilde{Y}_k$ and $\widetilde{Z}_k$. The aim of the analysis is to determine how the iterations propagate the errors $\{E_k, F_k\} \rightarrow \{E_{k+1}, F_{k+1}\}$. For numerical stability we require that the errors do not grow. (Note that the Padé iterations do not give us a free choice of starting matrix, so it is not necessarily the case that *arbitrary* errors in the iterates are damped out.)

We will carry out a first order analysis, dropping second order terms without comment. We have

$$\begin{aligned}\big(\widetilde{Y}_k\widetilde{Z}_k + I\big)^{-1} &= \big(2\big(I + \tfrac{1}{2}\big(\Lambda^{1/2}F_k + E_k\Lambda^{-1/2}\big)\big)\big)^{-1}\\ &= \tfrac{1}{2}\big(I - \tfrac{1}{2}\Lambda^{1/2}F_k - \tfrac{1}{2}E_k\Lambda^{-1/2}\big).\end{aligned}$$

The rewritten Padé iteration (2.10) with $p = 1$ is

$$Y_{k+1} = 2Y_k(Y_kZ_k + I)^{-1}, \qquad Z_{k+1} = 2Z_k(Y_kZ_k + I)^{-1}. \tag{A.1}$$

We have

$$\begin{aligned}\widetilde{Y}_{k+1} &= 2\big(\Lambda^{1/2} + E_k\big)\cdot\tfrac{1}{2}\big(I - \tfrac{1}{2}\Lambda^{1/2}F_k - \tfrac{1}{2}E_k\Lambda^{-1/2}\big)\\ &= \Lambda^{1/2} - \tfrac{1}{2}\Lambda F_k - \tfrac{1}{2}\Lambda^{1/2}E_k\Lambda^{-1/2} + E_k,\end{aligned}$$

$$\begin{aligned}\widetilde{Z}_{k+1} &= 2\big(\Lambda^{-1/2} + F_k\big)\cdot\tfrac{1}{2}\big(I - \tfrac{1}{2}\Lambda^{1/2}F_k - \tfrac{1}{2}E_k\Lambda^{-1/2}\big)\\ &= \Lambda^{-1/2} - \tfrac{1}{2}F_k - \tfrac{1}{2}\Lambda^{-1/2}E_k\Lambda^{-1/2} + F_k\\ &= \Lambda^{-1/2} - \tfrac{1}{2}\Lambda^{-1/2}E_k\Lambda^{-1/2} + \tfrac{1}{2}F_k.\end{aligned}$$

Hence

$$\begin{aligned}E_{k+1} &= E_k - \tfrac{1}{2}\Lambda^{1/2}E_k\Lambda^{-1/2} - \tfrac{1}{2}\Lambda F_k,\\ F_{k+1} &= -\tfrac{1}{2}\Lambda^{-1/2}E_k\Lambda^{-1/2} + \tfrac{1}{2}F_k.\end{aligned}$$

Writing $E_k = (e_{ij}^{(k)})$ and $F_k = (f_{ij}^{(k)})$, we have

$$\begin{bmatrix} e_{ij}^{(k+1)}\\ f_{ij}^{(k+1)}\end{bmatrix} = \begin{bmatrix} 1 - \frac{1}{2}\left(\frac{\lambda_i}{\lambda_j}\right)^{1/2} & -\frac{\lambda_i}{2}\\ -\frac{1}{2}\left(\frac{1}{\lambda_i\lambda_j}\right)^{1/2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix} e_{ij}^{(k)}\\ f_{ij}^{(k)}\end{bmatrix} =: M_{ij}^{(1)}\begin{bmatrix} e_{ij}^{(k)}\\ f_{ij}^{(k)}\end{bmatrix}.$$

Using

$$\begin{aligned}\big(\widetilde{Z}_k\widetilde{Y}_k + I\big)^{-1} &= \big(2\big(I + \tfrac{1}{2}\big(\Lambda^{-1/2}E_k + F_k\Lambda^{1/2}\big)\big)\big)^{-1}\\ &= \tfrac{1}{2}\big(I - \tfrac{1}{2}\Lambda^{-1/2}E_k - \tfrac{1}{2}F_k\Lambda^{1/2}\big)\end{aligned}$$

we find that the corresponding equations for the original Padé iteration (2.9) are

$$\begin{bmatrix} e_{ij}^{(k+1)}\\ f_{ij}^{(k+1)}\end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2}(\lambda_i\lambda_j)^{1/2}\\ -\frac{1}{2}\left(\frac{1}{\lambda_i\lambda_j}\right)^{1/2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix} e_{ij}^{(k)}\\ f_{ij}^{(k)}\end{bmatrix} =: M_{ij}^{(2)}\begin{bmatrix} e_{ij}^{(k)}\\ f_{ij}^{(k)}\end{bmatrix}.$$

(Not surprisingly, in view of the connection between (2.9) and the DB iteration (1.3), exactly the same equations hold for the DB iteration [13].)

The eigenvalues of $M_{ij}^{(2)}$ are 0 and 1. The powers of $M_{ij}^{(2)}$ are therefore bounded and so in iteration (2.9) the effect of the errors E_k and F_k on later iterates is bounded, to first order. However, $\rho(M_{ij}^{(1)}) > 1$ unless the eigenvalues of A cluster around 1. For example, if $\lambda_i = 16$ and $\lambda_j = 1$, then the eigenvalues of $M_{ij}^{(1)}$ are 1 and -1.5. Therefore, iteration (A.1) will, in general, amplify E_k and F_k, with the induced errors growing unboundedly as the iteration proceeds.

The conclusion of this analysis is that the Padé iteration (2.9) is numerically stable, but the rewritten version (A.1) is not.

References

[1] J. Albrecht, Bemerkungen zu Iterationsverfahren zur Berechnung von $A^{1/2}$ und A^{-1}, Z. Angew. Math. Mech. 57 (1977) T262–T263.

[2] G. Alefeld and N. Schneider, On square roots of M-matrices, Linear Algebra Appl. 42 (1982) 119–132.

[3] A. Berman and R.J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences* (Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994; first published 1979 by Academic Press).

[4] Å. Björck and S. Hammarling, A Schur method for the square root of a matrix, Linear Algebra Appl. 52/53 (1983) 127–140.

[5] G.J. Butler, C.R. Johnson and H. Wolkowicz, Nonnegative solutions of a quadratic matrix equation arising from comparison theorems in ordinary differential equations, SIAM J. Alg. Disc. Meth. 6(1) (1985) 47–53.

[6] R. Byers, Solving the algebraic Riccati equation with the matrix sign function, Linear Algebra Appl. 85 (1987) 267–279.

[7] G.W. Cross and P. Lancaster, Square roots of complex matrices, Linear and Multilinear Algebra 1 (1974) 289–293.

[8] E.D. Denman and A.N. Beavers, Jr., The matrix sign function and computations in systems, Appl. Math. Comput. 2 (1976) 63–94.

[9] L. Dieci, B. Morini and A. Papini, Computational techniques for real logarithms of matrices, SIAM J. Matrix Anal. Appl. 17(3) (1996) 570–593.

[10] G.H. Golub and C.F. Van Loan, *Matrix Computations* (Johns Hopkins Univ. Press, 3rd ed., Baltimore, MD, USA, 1996).

[11] L.A. Hageman and D.M. Young, *Applied Iterative Methods* (Academic Press, New York, 1981).

[12] N.J. Higham, Computing the polar decomposition – with applications, SIAM J. Sci. Statist. Comput. 7(4) (1986) 1160–1174.

[13] N.J. Higham, Newton's method for the matrix square root, Math. Comp. 46(174) (1986) 537–549.

[14] N.J. Higham, Computing real square roots of a real matrix, Linear Algebra Appl. 88/89 (1987) 405–430.

[15] N.J. Higham, The Test Matrix Toolbox for Matlab (version 3.0), Numerical Analysis Report No. 276, Manchester Centre for Computational Mathematics, Manchester, England, September (1995) 70 pp.

[16] N.J. Higham, The matrix sign decomposition and its relation to the polar decomposition, Linear Algebra Appl. 212/213 (1994) 3–20.

[17] N.J. Higham, *Accuracy and Stability of Numerical Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996).

[18] N.J. Higham and P. Papadimitriou, A parallel algorithm for computing the polar decomposition, Parallel Comput. 20(8) (1994) 1161–1173.

[19] N.J. Higham and R.S. Schreiber, Fast polar decomposition of an arbitrary matrix, SIAM J. Sci. Statist. Comput. 11(4) (1990) 648–655.

[20] R.A. Horn and C.R. Johnson, *Matrix Analysis* (Cambridge University Press, 1985).

[21] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis* (Cambridge University Press, 1991).

[22] W.D. Hoskins and D.J. Walton, A faster method of computing the square root of a matrix, IEEE Trans. Automat. Control. AC-23(3) (1978) 494–495.

[23] T.J.R. Hughes, I. Levit and J. Winget, Element-by-element implicit algorithms for heat conduction, J. Eng. Mech. 109(2) (1983) 576–585.

[24] C. Kenney and A.J. Laub, Condition estimates for matrix functions, SIAM J. Matrix Anal. Appl. 10(2) (1989) 191–209.

[25] C. Kenney and A.J. Laub, Padé error estimates for the logarithm of a matrix, Internat J. Control 50(3) (1989) 707–730.

[26] C. Kenney and A.J. Laub, Rational iterative methods for the matrix sign function, SIAM J. Matrix Anal. Appl. 12(2) (1991) 273–291.

[27] C. Kenney and A.J. Laub, On scaling Newton's method for polar decomposition and the matrix sign function, SIAM J. Matrix Anal. Appl. 13(3) (1992) 688–706.

[28] C.S. Kenney and A.J. Laub, A hyperbolic tangent identity and the geometry of Padé sign function iterations, Numerical Algorithms 7 (1994) 111–128.

[29] C.S. Kenney and A.J. Laub, The matrix sign function, IEEE Trans. Automat. Control 40(8) (1995) 1330–1348.

[30] P. Laasonen, On the iterative solution of the matrix equation $AX^2 - I = 0$, M.T.A.C. 12 (1958) 109–116.

[31] P. Pandey, C. Kenney and A.J. Laub, A parallel algorithm for the matrix sign function, Internat. J. High Speed Comput. 2(2) (1990) 181–191.

[32] B.N. Parlett, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs, NJ, 1980).

[33] P. Pulay, An iterative method for the determination of the square root of a positive definite matrix, Z. Angew. Math. Mech. 46 (1966) 151.

[34] J.D. Roberts, Linear model reduction and solution of the algebraic Riccati equation by use of the sign function, Internat. J. Control 32(4) (1980) 677–687. First issued as report CUED/B-Control/TR13, Department of Engineering, University of Cambridge (1971).

[35] B.A. Schmitt, An algebraic approximation for the matrix exponential in singularly perturbed boundary value problems, SIAM J. Numer. Anal. 27(1) (1990) 51–66.