Approximating the logarithm of a matrix to specified accuracy

Cheng, Sheung Hun and Higham,
Nicholas J and Kenney, Charles S and Laub, Alan J

2001

MIMS EPrint: **2006.142**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

SIAM J. MATRIX ANAL. APPL.
Vol. 22, No. 4, pp. 1112–1125

APPROXIMATING THE LOGARITHM OF A MATRIX TO SPECIFIED ACCURACY*

SHEUNG HUN CHENG†, NICHOLAS J. HIGHAM‡, CHARLES S. KENNEY§, AND ALAN J. LAUB¶

Abstract. The standard inverse scaling and squaring algorithm for computing the matrix logarithm begins by transforming the matrix to Schur triangular form in order to facilitate subsequent matrix square root and Padé approximation computations. A transformation-free form of this method that exploits incomplete Denman–Beavers square root iterations and aims for a specified accuracy (ignoring roundoff) is presented. The error introduced by using approximate square roots is accounted for by a novel splitting lemma for logarithms of matrix products. The number of square root stages and the degree of the final Padé approximation are chosen to minimize the computational work. This new method is attractive for high-performance computation since it uses only the basic building blocks of matrix multiplication, LU factorization and matrix inversion.

Key words. matrix logarithm, Padé approximation, inverse scaling and squaring method, matrix square root, Denman–Beavers iteration

AMS subject classification. 65F30

PII. S0895479899364015

1. Introduction.

Logarithms of matrices arise in various contexts. For example, for a physical system governed by a linear differential equation of the form

$$\frac{dy}{dt} = Xy,$$

we may be interested in determining the matrix X from observations of the state vector $y(t)$ [1], [20]. If $y(0) = y_0$ then $y(t) = e^{Xt}y_0$, where the exponential of a matrix is defined by

$$e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}.$$

By observing y at $t = 1$ for initial states consisting of the columns of the identity matrix, we obtain the matrix $A = e^X$. Under certain conditions on A and X, we can then solve for X as $X = \log A$. This raises the question of how to compute a logarithm of a matrix.

When A is near the identity matrix several methods can be used to approximate $\log A$ directly, that is, without any nontrivial transformation of A. For example,

*Received by the editors November 16, 1999; accepted for publication (in revised form) by A. Edelman August 23, 2000; published electronically March 13, 2001.

http://www.siam.org/journals/simax/22-4/36401.html

†Centre for Novel Computing, Department of Computer Science, University of Manchester, Manchester, M13 9PL, England (scheng@cs.man.ac.uk, http://www.cs.man.ac.uk/~scheng/). The work of this author was supported by Engineering and Physical Sciences Research Council grant GR/L94314.

‡Department of Mathematics, University of Manchester, Manchester, M13 9PL, England (higham@ma.man.ac.uk, http://www.ma.man.ac.uk/~higham/). The work of this author was supported by Engineering and Physical Sciences Research Council grant GR/L94314.

§ECE Department, University of California, Santa Barbara, CA 93106-9560 (kenney@seidel.ece.ucsb.edu).

¶College of Engineering, University of California, Davis, CA 95616-5294 (laub@ucdavis.edu). The work of this author was supported by NSF grant ECS-9633326.

we can truncate the Taylor series $\log(I-W) = -W - W^2/2 - W^3/3 - \cdots$, where $W = I - A$. Alternatively, we can use Padé approximations of $\log(I-W)$; see [16] and section 5 below. Unfortunately, if A is not near the identity then these methods either do not converge or converge so slowly that they are not of practical use. The standard way of dealing with this problem is to use the square root operator repeatedly to bring A near the identity:

$$\log A = 2^k \log A^{1/2^k}. \tag{1.1}$$

(Definitions of the logarithm and square root functions for matrices are given in the next section.) As k increases, $A^{1/2^k} \to I$, so for sufficiently large k we can apply a direct method to $A^{1/2^k}$. This procedure for the logarithm was introduced by Kenney and Laub [15] and is referred to as inverse scaling and squaring, since it reverses the usual scaling and squaring method of evaluating the matrix exponential: $e^X = \left(e^{X/2^k}\right)^{2^k}$ [19], [21].

Two related questions arise with the inverse scaling and squaring method. First, potentially the most expensive part of the method is the computation of the square roots. For cases where only modest accuracy is required in the logarithm it is natural to ask whether the cost of this part of the computation can be reduced by computing approximate square roots. The second question concerns the effect of errors in computing the square roots on the accuracy of the computed logarithm. In [15] the square roots are computed using the Schur method [4], [9], [12], which has essentially optimal accuracy and stability properties, but the effects of rounding errors are not analyzed.

In partial answer to these questions we develop an extension of the inverse scaling and squaring method with two key properties.

1. It aims for a specified accuracy in the computed logarithm, requiring less work when more modest accuracy is requested. When full accuracy (that of the underlying arithmetic) is requested, our method becomes a new and attractive implementation of the original inverse scaling and squaring method.
2. It can be implemented using only the basic building blocks of matrix multiplication, LU factorization, and matrix inversion. The method is therefore attractive for high-performance computation.

In view of these two properties our method may also be of interest for computing the logarithm in variable precision computing environments, such as in symbolic manipulation packages. Our bounds for the various truncation errors are developed for exact arithmetic. In floating point arithmetic, rounding errors also influence the accuracy. We do not rigorously bound the effect of rounding errors, but rather estimate it in terms of the conditioning of the problem.

Our new method is based on a splitting lemma for the logarithm (Lemma 2.1 below), which says that if $A = BC$ and B and C commute then, under certain conditions,

$$\log A = \log B + \log C.$$

In the special case $B = C = A^{1/2}$ we recover the basis of (1.1): $\log A = 2\log A^{1/2}$. We apply the splitting lemma to the Denman–Beavers (DB) iteration for the matrix square root [5]:

$$\begin{aligned} Y_{k+1} &= \left(Y_k + Z_k^{-1}\right)/2, \qquad Y_0 = A, \\ Z_{k+1} &= \left(Z_k + Y_k^{-1}\right)/2, \qquad Z_0 = I. \end{aligned}$$

The DB iteration converges quadratically with $Y_k \to A^{1/2}$ and $Z_k \to A^{-1/2}$. The splitting lemma can be used to show that

$$\log A = 2\log Y_k - \log Y_k Z_k.$$

The matrix product $Y_k Z_k$ converges to the identity and so its logarithm converges to zero. Our approach is to iterate until $\log Y_k Z_k$ is sufficiently small, then apply the process recursively on Y_k, monitoring the error build-up as we proceed. We thus apply an incomplete square root cascade that brings A close enough to the identity so that the logarithm can be approximated directly.

To increase the efficiency of our method we develop in section 3 a product form of the DB iteration that trades one of the matrix inversions for a matrix multiplication and automatically generates the products $Y_k Z_k$. We also incorporate scaling to reduce the overall number of iterations. The product form iteration turns out to be closely related to the standard Newton iteration for the matrix sign function, as explained in section 4. In section 5 we develop the implementation details for the incomplete square root cascade. Our method uses a Padé approximation, explained in section 6, whose order is chosen in section 7 together with the number of square root stages in order to minimize the computational work. Numerical experiments are described in section 8 and conclusions are given in section 9.

2. Splitting lemma. We begin by defining the matrix logarithm and square root functions. Let A be a real or complex matrix of order n with no eigenvalues on $\mathbb{R}^-$ (the closed negative real axis). Then there exists a unique matrix X such that [15]

1. $e^X = A$;
2. the eigenvalues of X lie in the strip $\{\, z : -\pi < \mathrm{Im}(z) < \pi \,\}$;

We refer to X as the (principal) logarithm of A and write $X = \log A$. Similarly, there is a unique matrix S such that [9], [15]

1. $S^2 = A$;
2. the eigenvalues of S lie in the open halfplane: $0 < \mathrm{Re}(z)$.

We refer to S as the (principal) square root of A and write $S = A^{1/2}$.

If A is real then its principal logarithm and principal square root are also real.

For our first result, we need to define the open halfplane associated with $z = \rho e^{i\theta}$, which is the set of complex numbers $w = \zeta e^{i\phi}$ such that $-\pi/2 < \phi - \theta < \pi/2$.

LEMMA 2.1 (splitting lemma). *Suppose that $A = BC$ has no eigenvalues on $\mathbb{R}^-$ and*

1. *$BC = CB$;*
2. *every eigenvalue of B lies in the open halfplane of the corresponding eigenvalue of $A^{1/2}$ (or, equivalently, the same condition holds for C).*

Then $\log A = \log B + \log C$.

Proof. First we show that the logarithms of B and C are well defined. Since $A = BC = CB$ it follows that A commutes with B and C. Thus there is a correspondence between the eigenvalues a, b, and c of A, B, and C: $a = bc$. Express these eigenvalues in polar form as

$$a = \alpha e^{i\theta}, \quad b = \beta e^{i\phi}, \quad c = \gamma e^{i\psi}.$$

Since A has no eigenvalues on $\mathbb{R}^-$,

$$-\pi < \theta < \pi. \tag{2.1}$$

The eigenvalues of B lie in the open halfplanes of the corresponding eigenvalues of $A^{1/2}$, that is,

$$-\frac{\pi}{2} < \phi - \frac{\theta}{2} < \frac{\pi}{2}. \tag{2.2}$$

The relation $a = bc$ gives $\theta = \phi + \psi$, from which we have $\psi - \theta/2 = \theta/2 - \phi$. It follows from (2.2) that the eigenvalues of C lie in the open halfplanes of the corresponding eigenvalues of $A^{1/2}$. Thus, in view of (2.1), B and C have no eigenvalues on $\mathbb{R}^-$ and their logarithms are well defined.

Next, we show that $e^{\log B + \log C} = A$. The matrices $\log B$ and $\log C$ commute since B and C do. Using the well-known result that the exponential of the sum of commuting matrices is the product of the exponentials [14, Thm. 6.2.38], we have

$$e^{\log B + \log C} = e^{\log B} e^{\log C} = BC = A.$$

It remains to show that the eigenvalues of $\log B + \log C$ have imaginary parts in $(-\pi, \pi)$. This follows since, in view of the commutativity of B and C, the eigenvalues of $\log B + \log C$ are $\log b + \log c = \log a$. □

Note that for $A = BC$ the commutativity condition $BC = CB$ is not enough to guarantee that $\log A = \log B + \log C$, as the following scalar example shows. Let $a = e^{-2\epsilon i}$ and $b = c = e^{(\pi - \epsilon)i}$ for ϵ small and positive. Then $a = bc$ but

$$\log a = -2\epsilon i \neq (\pi - \epsilon)i + (\pi - \epsilon)i = \log b + \log c.$$

The reason for this behavior is that b and c are equal to a nonprincipal square root of a, and hence are not in the halfplane of $a^{1/2}$.

3. DB square root iteration. The DB iteration [5] for the square root of a matrix A with no eigenvalues on $\mathbb{R}^-$ is

$$\begin{aligned} Y_{k+1} &= \left(Y_k + Z_k^{-1}\right)/2, \qquad Y_0 = A, \\ Z_{k+1} &= \left(Z_k + Y_k^{-1}\right)/2, \qquad Z_0 = I. \end{aligned} \tag{3.1}$$

The iteration has the properties [8] (and see Theorem 4.1, below)

$$\lim_{k\to\infty} Y_k = A^{1/2}, \qquad \lim_{k\to\infty} Z_k = A^{-1/2}$$

and, for all k,

$$\begin{aligned} Y_k &= AZ_k, \\ Y_k Z_k &= Z_k Y_k, \\ Y_{k+1} &= (Y_k + AY_k^{-1})/2. \end{aligned} \tag{3.2}$$

The next lemma is the basis for our use of the DB iteration for computing the logarithm.

LEMMA 3.1. *The DB iterates satisfy the splitting relations*

$$\begin{aligned} \log A &= \log Y_k - \log Z_k \\ &= 2\log Y_k - \log Y_k Z_k \\ &= -2\log Z_k + \log Y_k Z_k. \end{aligned}$$

Proof. Since $A = Y_k Z_k^{-1}$, Y_k and Z_k commute and $\log Z_k^{-1} = -\log Z_k$, the first equality follows from Lemma 2.1 if we can show that the eigenvalues of Y_k are in the halfplane of the corresponding eigenvalues of $A^{1/2}$. By (3.2), the individual eigenvalues of Y_k follow the scalar iteration

$$y_{k+1} = (y_k + a y_k^{-1})/2, \qquad y_0 = a,$$

where a is an eigenvalue of A. This is just the scalar Newton iteration for the square root of a and it has the property that the iterates y_k remain in the halfplane of $a^{1/2}$ (see, e.g., [8]). Similar arguments show that $\log Y_k Z_k = \log Y_k + \log Z_k$, which yields the remaining two equalities. ∎

To see how to use Lemma 3.1, note that since $Y_k \to A^{1/2}$ and $Z_k \to A^{-1/2}$, $Y_k Z_k \to I$ and $\log Y_k Z_k \to 0$. Suppose we terminate the DB iteration after k iterations; we can write

$$\log A = 2 \log Y_k - E_1,$$

where we wish $E_1 = \log Y_k Z_k$ to be suitably small. Define $Y^{(1)} = Y_k$, $Z^{(1)} = Z_k$. We now apply the DB iteration to $Y^{(1)}$, again for a finite number of iterations. Continuing this process leads after s steps to

$$\log A = 2^s \log Y^{(s)} - E_1 - 2E_2 - \cdots - 2^{s-1} E_s, \quad E_i = \log Y^{(i)} Z^{(i)}, \tag{3.3}$$

where $Y^{(i)}$ and $Z^{(i)}$ are the final iterates from the DB iteration applied to $Y^{(i-1)}$. Our aim is that $\log Y^{(s)}$ be easy to compute and the E_i terms be small enough to be ignored. Note that we could apply the DB iteration to the $Z^{(i)}$ instead of the $Y^{(i)}$; all the following analysis is easily adapted for this choice.

We need to bound the error terms $E_i = \log Y^{(i)} Z^{(i)}$ without computing a matrix logarithm. One way to do this is as follows. Using the Taylor expansion of $\log(1+x)$ it is easy to show that if $\|I - YZ\| < 1$ then

$$\|\log YZ\| \le |\log(1 - \|I - YZ\|)|. \tag{3.4}$$

Here, and throughout, the norm is any subordinate matrix norm. The terms $Y_k Z_k$ are not formed during the DB iteration. However, a little manipulation shows that $Y_{k+1} Z_{k+1} - I = (Y_{k+1} - Y_k)(Z_{k+1} - Z_k)$ and hence

$$\|Y_{k+1} Z_{k+1} - I\| \le \|Y_{k+1} - Y_k\| \|Z_{k+1} - Z_k\|. \tag{3.5}$$

Thus, if this upper bound does not exceed 1, we have a bound for $\|E_i\|$ that can be computed at no extra cost and can be used to decide when the E_i terms can be neglected. However, both the bounds (3.4) and (3.5), and hence the bound for $\|E_i\|$, can be weak. Fortunately, there is a better approach: we can reformulate the DB iteration in terms of Y_k (or Z_k) and the required product $M_k = Y_k Z_k$, as the next lemma shows.

LEMMA 3.2 (product form of DB iteration). *Let Y_k and Z_k be the DB iterates for A and define $M_k = Y_k Z_k$. Then*

$$\begin{aligned} M_{k+1} &= \frac{1}{2}\left(I + \frac{M_k + M_k^{-1}}{2}\right), && M_0 = A, \\ Y_{k+1} &= Y_k (I + M_k^{-1})/2, && Y_0 = A, \\ Z_{k+1} &= Z_k (I + M_k^{-1})/2, && Z_0 = I. \end{aligned} \tag{3.6}$$

In a high-performance computing environment, iterating with M_k and Y_k from (3.6), at the cost of one inversion and one multiplication per iteration, is preferable to iterating with Y_k and Z_k from (3.1), at the cost of two inversions per iteration, since matrix multiplication is faster than matrix inversion.

In practice, it is vital to scale matrix iterations to produce reasonably fast overall convergence. Higham [11] derives a scaling for the DB iteration based on $\theta = \det(Y_k)\det(Z_k)$: it requires Y_k and Z_k to be multiplied by $|\theta^{-1/(2n)}|$ at the start of the $(k+1)$st iteration, where A is of order n. For the product form of the iteration, since $\det(Y_k)\det(Z_k) = \det(M_k)$ and we invert and hence factorize M_k, θ is available at no extra cost.

Matrix iterations such as the DB iteration can suffer from numerical instability. Although an iteration may be globally convergent for the specified starting matrices, rounding errors can introduce perturbations that grow unboundedly, this phenomenon usually being associated with loss of commutativity of the iterates. We define an iteration $X_{k+1} = f(X_k)$ to be stable in a neighborhood of a solution $X = f(X)$ if the error matrices $E_k = X - X_k$ satisfy

$$E_{k+1} = L(E_k) + O(\|E_k\|^2),$$

where L is a linear operator that has bounded powers, that is, there exists a constant c such that for all $p > 0$ and arbitrary E of unit norm, $\|L^p(E)\| \le c$. The DB iteration is stable [8], [11]; the iteration (3.2), which is a standard Newton iteration for $A^{1/2}$, is unstable unless the eigenvalues λ_i of A satisfy [8] $\max_{i,j}\left|1 - (\lambda_i/\lambda_j)^{1/2}\right| \le 2$.

It is easy to show that the product form of the DB iteration is stable. Define the error terms $G_k = Y_k - A^{1/2}$, $H_k = Z_k - A^{-1/2}$, and $J_k = M_k - I$. Simple manipulations show that, to first order in G_k, H_k, and J_k,

$$\begin{bmatrix} G_{k+1} \\ H_{k+1} \\ J_{k+1} \end{bmatrix} = \begin{bmatrix} I & 0 & -A^{1/2}/2 \\ 0 & I & -A^{-1/2}/2 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} G_k \\ H_k \\ J_k \end{bmatrix} \equiv C \begin{bmatrix} G_k \\ H_k \\ J_k \end{bmatrix}.$$

The coefficient matrix C is idempotent ($C^2 = C$) and hence has bounded powers. Thus the iteration is stable.

Before explaining the use of the modified DB iteration, we develop more insight into its properties by relating it to a well-known iteration for the matrix sign function.

4. Relation to matrix sign function iteration. For a matrix N with no eigenvalues on the imaginary axis the sign function is defined by [10], [18]

$$\operatorname{sign} N = N\left(N^2\right)^{-1/2}.$$

The standard approach to compute $\operatorname{sign} N$ is to use the Newton iteration

$$N_{k+1} = \left(N_k + N_k^{-1}\right)/2, \qquad N_0 = N.$$

This iteration converges quadratically to $S = \operatorname{sign} N$, with error evolving in the Cayley metric according to [17]

$$(N_{k+1} - S)(N_{k+1} + S)^{-1} = \left((N_k - S)(N_k + S)^{-1}\right)^2.$$

The following theorem shows that the DB iterates are scaled versions of the Newton iterates for $\operatorname{sign} A^{1/2}$.

THEOREM 4.1. *Let A have no eigenvalues on $\mathbb{R}^-$. Let N_k be the Newton iterates for* $\operatorname{sign} A^{1/2}$ $(= I)$ *and Y_k, Z_k, and $M_k = Y_k Z_k$ be the DB iterates for A in* (3.1) *and* (3.6). *Then*

$$Y_k = A^{1/2} N_k, \qquad Z_k = A^{-1/2} N_k, \qquad M_k = N_k^2.$$

Proof. A straightforward induction, making use of the fact that N_k commutes with $A^{1/2}$. ∎

Theorem 4.1 implies that the DB iterates Y_k, Z_k, and M_k converge quadratically to $A^{1/2}$, $A^{-1/2}$, and I, respectively, with errors evolving in the Cayley metric according to

$$\begin{aligned}
(Y_{k+1} - A^{1/2})(Y_{k+1} + A^{1/2})^{-1} &= ((Y_k - A^{1/2})(Y_k + A^{1/2})^{-1})^2, \\
(Z_{k+1} - A^{-1/2})(Z_{k+1} + A^{-1/2})^{-1} &= ((Z_k - A^{-1/2})(Z_k + A^{-1/2})^{-1})^2, \\
(N_{k+1} - I)(N_{k+1} + I)^{-1} &= ((N_k - I)(N_k + I)^{-1})^2,
\end{aligned}$$

where $N_k = M_k^{1/2}$.

From [18] we know that k steps of the Newton iteration for the sign function generate the kth diagonal Padé approximation to the sign function, which is given by $r_k = p_k/q_k$, where p_k and q_k are the even and odd parts, respectively, of the polynomial $(1+x)^{2^k}$. Using Theorem 4.1 we can therefore obtain explicit rational expressions for the DB iterates. For example, $Y_k = \widetilde{p}_k(A)\widetilde{q}_k^{-1}(A)$, where

$$\begin{aligned}
\widetilde{p}_k(A) &= \binom{2^k}{0} + \binom{2^k}{2} A + \binom{2^k}{4} A^2 + \cdots + \binom{2^k}{2^k} A^{2^{k-1}}, \\
\widetilde{q}_k(A) &= \binom{2^k}{1} I + \binom{2^k}{3} A + \binom{2^k}{5} A^2 + \cdots + \binom{2^k}{2^k - 1} A^{2^{k-1}-1}.
\end{aligned}$$

5. Incomplete square root cascade. We return now to the use of the product form of the DB iteration to compute the logarithm. The following algorithm describes how we use the DB iteration, but omits convergence tests.

ALGORITHM 5.1. *This algorithm runs an incomplete square root cascade on the matrix A of order n, using the product form of the DB iteration with scaling. The DB iteration is invoked s times, with k_i iterations on the ith invocation.*

for $i = 1\colon s$
  if $i = 1$
    $M_0 = A$, $Y_0 = A$
  else
    $M_0 = Y^{(i-1)}$, $Y_0 = Y^{(i-1)}$
  end
  for $k = 0\colon k_i - 1$
    $\gamma_k = |(\det(M_k))^{-1/(2n)}|$
    $M_{k+1} = \frac{1}{2}\Big(I + \dfrac{\gamma_k^2 M_k + \gamma_k^{-2} M_k^{-1}}{2}\Big)$
    $Y_{k+1} = \frac{1}{2}\gamma_k Y_k (I + \gamma_k^{-2} M_k^{-1})$
  end
  $M^{(i)} = M_{k_i}$, $Y^{(i)} = Y_{k_i}$
end

With the notation of Algorithm 5.1, we can rewrite (3.3) as

$$\log A = 2^s \log Y^{(s)} - \log M^{(1)} - 2\log M^{(2)} - \cdots - 2^{s-1}\log M^{(s)}. \tag{5.1}$$

Rather than simply discard the terms $\log M^{(i)} = \log M_k$, we can approximate them using

$$\log M_k \approx M_k - I. \tag{5.2}$$

The error in this approximation satisfies

$$\|\log M_k - (M_k - I)\| \approx \|(M_k - I)^2\|/2. \tag{5.3}$$

For comparison, the error resulting from continuing for one more iteration and then discarding $\log M_{k+1}$ is

$$\|\log M_{k+1}\| \approx \|M_{k+1} - I\|. \tag{5.4}$$

It can be shown that

$$M_{k+1} - I = \frac{1}{4}(M_k - I)^2 M_k^{-1},$$

and hence the error term in (5.4) is approximately half that in (5.3) close to convergence (recall that $M_k \to I$). The product form of the DB iteration thus has an advantage over the original iteration; because it generates M_k explicitly it allows us to use the approximation (5.2) and thus to obtain similar accuracy in the logarithm with one less iteration.

Define the approximation $L^{(s)}$ to $\log A$ by

$$L^{(s)} = 2^s \log Y^{(s)} - \big(M^{(1)} - I\big) - 2\big(M^{(2)} - I\big) - \cdots - 2^{s-1}\big(M^{(s)} - I\big). \tag{5.5}$$

Then, subtracting (5.5) from (5.1) gives

$$\log A = L^{(s)} - \widetilde{E}_1 - 2\widetilde{E}_2 - \cdots - 2^{s-1}\widetilde{E}_s, \tag{5.6}$$

where

$$\widetilde{E}_i = \log M^{(i)} - \big(M^{(i)} - I\big).$$

THEOREM 5.2. *Let $\delta > 0$. In the ith product DB square root stage of Algorithm* 5.1 *let k_i be large enough so that*

$$\Big|\, \|W^{(i)}\| + \log(1 - \|W^{(i)}\|) \,\Big| \le \delta/4^{i-1}, \tag{5.7}$$

where $W^{(i)} = I - M^{(i)}$. Then

$$\|\log A - L^{(s)}\| \le 2\delta\left(1 - \frac{1}{2^s}\right). \tag{5.8}$$

Proof. Using the bound

$$\|\widetilde{E}_i\| = \|W^{(i)} + \log(I - W^{(i)})\| \le |\|W^{(i)}\| + \log(1 - \|W^{(i)}\|)| \le \delta/4^{i-1}$$

in (5.6) and summing a geometric series yields the result. ∎

To obtain a logarithm approximation, the final step is to approximate $\log Y^{(s)}$, by $\widetilde{L}$, say. Then our approximation to $\log A$ is

$$\widetilde{X} = 2^s \widetilde{L} - \sum_{k=1}^{s} 2^{k-1}\big(M^{(k)} - I\big). \tag{5.9}$$

Assuming we choose the k_i as in Theorem 5.2, then (5.8) leads to

$$\|\widetilde{X} - \log A\| \le 2^s \|\widetilde{L} - \log Y^{(s)}\| + 2\delta\left(1 - \frac{1}{2^s}\right). \tag{5.10}$$

It is natural to require that the error due to our approximation of $\log Y^{(s)}$ satisfy the same bound as the error introduced by the incomplete square roots; thus we require

$$\|\widetilde{L} - \log Y^{(s)}\| \le 2^{1-s}\delta\left(1 - \frac{1}{2^s}\right). \tag{5.11}$$

Then we have the overall error bound

$$\|\widetilde{X} - \log A\| \le 4\delta\left(1 - \frac{1}{2^s}\right) < 4\delta. \tag{5.12}$$

Two questions arise: How shall we select s and how can we find $\widetilde{L}$ such that (5.11) is satisfied? These questions are treated in the next two sections. We close this section by noting that (5.10) shows that the error in approximating $\log Y^{(s)}$ is magnified by a factor 2^s. This is a fundamental limitation of the inverse scaling and squaring approach that is also identified in [7].

6. Padé approximants. If A is near the identity matrix then rational approximation of $\log A$ is practical. Diagonal Padé approximants preserve some important properties of the logarithm and offer rapid convergence as the degree of the approximant increases [16]. For a given scalar function

$$f(x) = \sum_{n=0}^{\infty} a_n x^n,$$

we say that the rational function $r_{km} = p_{km}/q_{km}$ is a $[k/m]$ Padé approximant of f if p_{km} is a polynomial in x of degree at most k, q_{km} is a polynomial in x of degree at most m, and $f(x) - r_{km}(x) = O\big(x^{k+m+1}\big)$. In addition, we usually require that p_{km} and q_{km} are relatively prime (have no common zeros) and that q_{km} has been normalized so that $q_{km}(0) = 1$. These conditions ensure that if a $[k/m]$ approximant exists then it is unique; see [2] and [3]. Following Kenney and Laub [16] we restrict our attention to the diagonal ($k = m$) Padé approximants of $f(x) = \log(1 - x)$, the first three of which are (here, for convenience we have not normalized q_{mm})

$$r_{11}(x) = \frac{-2x}{2 - x}, \quad r_{22}(x) = \frac{-6x + 3x^2}{6 - 6x + x^2}, \quad r_{33}(x) = \frac{-60x + 60x^2 - 11x^3}{60 - 90x + 36x^2 - 3x^3}.$$

Kenney and Laub [16] show that the error in the Padé approximant evaluated at a matrix argument X is bounded by the error in the scalar approximation with $x = \|X\|$, provided that $\|X\| < 1$:

$$\|r_{mm}(X) - \log(I - X)\| \le |r_{mm}(\|X\|) - \log(1 - \|X\|)|. \tag{6.1}$$

This bound can be evaluated at negligible cost given r_{mm}.

7. Inverse scaling and squaring with specified accuracy. The availability of the error bound (6.1) for the Padé approximation makes possible a strategy for choosing s (the number of incomplete DB square root stages) and the order m of the final Padé approximation in order to achieve the desired accuracy with minimal work. For Padé approximation to be applicable s must be large enough so that $\|I - Y^{(s)}\| < 1$. Once this point is reached, we can compare the work required to produce an acceptable Padé approximation at the current square root stage with the work required to carry out another square root stage and then evaluate a Padé approximation.

In view of (5.11) and (6.1), a suitable order m_k of the Padé approximation at the kth square root stage is the smallest m for which

$$|r_{mm}(\|X\|) - \log(1 - \|X\|)| \le 2^{1-k}\delta\left(1 - \frac{1}{2^k}\right), \qquad X = I - Y^{(k)}, \tag{7.1}$$

where r_{mm} is the Padé approximant of order m as described in section 6. With this choice of m, and with the number of DB iterations k_i chosen as in Theorem 5.2, we have the bound (5.12), that is,

$$\|\widetilde{X} - \log A\| < 4\delta, \tag{7.2}$$

where $\widetilde{X}$ is given by (5.9) with $\widetilde{L} = r_{mm}(I - Y^{(k)})$. Note that this bound does not incorporate the effects of rounding errors. We comment below on the effects of roundoff.

Having determined m_k, we can consider whether to iterate further or not, by examining the cost of evaluating the Padé approximation. Several methods of evaluation are described and compared with respect to cost, storage, and accuracy in [13]. The best overall method is based on the partial fraction expansion

$$r_{mm}(x) = \sum_{j=1}^{m} \frac{\alpha_j^{(m)} x}{1 + \beta_j^{(m)} x}, \tag{7.3}$$

where the $\alpha_j^{(m)}$ are the weights and the $\beta_j^{(m)}$ the nodes of the m-point Gauss–Legendre quadrature rule on $[0, 1]$. Evaluating r_{mm} at the matrix argument X with $m = m_k$ requires the solution of m_k linear systems each having n right-hand sides, which we will regard as equivalent to m_k matrix inversions.

To estimate the cost of proceeding for a further square root stage we need to know the number of iterations in that stage and the degree m_{k+1} of the Padé approximation at the end of the stage. Since $A^{1/2^k} \to I$ as k increases, the square roots become easier to compute with increasing k, but this is compensated for by the more stringent accuracy demanded by the condition (5.7). In practice, the number of square root iterations frequently stays the same or decreases by 1 from one stage to the next. For our calculations we assume that the number of square root iterations on the $(k+1)$st stage is the same as that on the kth stage, which we denote by it_k. The estimated cost of the next square root stage is therefore it_k matrix multiplications and it_k matrix inversions.

To estimate m_{k+1} we note that

$$\left(I - A^{1/2^{k+1}}\right)\left(I + A^{1/2^{k+1}}\right) = I - A^{1/2^k}. \tag{7.4}$$

Since $A^{1/2^k} \to I$ we have

$$\|I - A^{1/2^{k+1}}\| \approx \frac{1}{2}\|I - A^{1/2^k}\|. \tag{7.5}$$

We therefore use the approximation $\|I-Y^{(k+1)}\| \approx \|I-Y^{(k)}\|/2$ in (7.1) to determine m_{k+1}.

Denoting by α the ratio "cost of matrix inversion divided by cost of matrix multiplication," we terminate the square root iterations if

$$m_k \le m_{k+1} + (1+\alpha)\mathrm{it}_k. \tag{7.6}$$

For our tests we have taken $\alpha = 1$ (as suggested by the operation counts), but on many computers a value of α bigger than 1 and possibly depending on n would be more appropriate.

It is worth stressing that if any of the assumptions underlying our choice of s and m are not satisfied then the efficiency of the computation may be less than optimal but the error bound (7.2) still holds.

We summarize our overall algorithm as follows.

ALGORITHM 7.1. *Given a matrix A with no eigenvalues on $\mathbb{R}^-$, and a tolerance $\delta > 0$, this algorithm approximates $X = \log A$ to within absolute accuracy 4δ (ignoring roundoff).*

1. Run Algorithm 5.1 with the k_i chosen as in Theorem 5.2, choosing s, the number of DB iteration stages, as the first k for which $\|I-Y^{(k)}\| \le 0.99$, and (7.6) is satisfied with $m_k \le 16$.
2. Use (7.3) to evaluate X, the $[m_s/m_s]$ Padé approximation $r_{m_s,m_s}(B)$ to $\log(I-B)$, where $B = I - Y^{(s)}$.
3. $X = 2^s X - \sum_{k=1}^{s} 2^{k-1}\big(M^{(k)} - I\big)$.

Now we return to the effects of roundoff. We do not attempt here a full rounding error analysis of Algorithm 7.1, as experience shows that it is difficult to obtain useful error bounds for iterations for the matrix square root and sign function. However, several observations can be made. First, it is shown in [13] that with the parameters 0.99 and 16 in step 1 of Algorithm 7.1 the Padé approximation is evaluated to high accuracy, because the matrices that are inverted are very well conditioned. Second, for tolerances δ sufficiently larger than u the rounding errors can be subsumed in the truncation errors. Finally, even if the computed $\widehat{X}$ has perfect backward stability, that is,

$$\widehat{X} = \log(A + \Delta A), \qquad \|\Delta A\| \le u\|A\|, \tag{7.7}$$

where u is the unit roundoff, then the best forward error bound is [6], [15]

$$\frac{\|\widehat{X} - X\|}{\|X\|} \le \|G'(A)\|\frac{\|A\|}{\|X\|}u + O(u^2),$$

where $G'(A)$ is the Fréchet derivative of $G(A) = \log A = X$ at A. The term

$$\mathrm{cond}_G(A) = \|G'(A)\|\frac{\|A\|}{\|X\|}$$

is a condition number for the logarithm function and it is notably absent in (7.2). Since no numerical algorithm can be expected to do better than achieve (7.7), we must accept that the computed $\widehat{X}$ will at best satisfy the modified version of (7.2)

$$\|\widehat{X} - \log A\|_1 \le \mathrm{cond}_G(A)\|X\|u + 4\delta. \tag{7.8}$$

Methods for estimating $\mathrm{cond}_G(A)$ are developed in [15].

8. Numerical experiments. We have implemented Algorithm 7.1 in MATLAB, for which the unit roundoff $u = 2^{-53} \approx 1.1 \times 10^{-16}$. The various tests in Algorithm 7.1 use the 1-norm. We describe results for three matrices $A \in \mathbb{R}^{16\times 16}$.

Matrix 1: $\kappa_2(A) = 10^8$, $\mathrm{cond}_G(A) \approx 10^8$. A is a random symmetric positive definite matrix with eigenvalues exponentially distributed between 10^{-8} and 1, formed using MATLAB's `gallery('randsvd',...)`.

Matrix 2: $\kappa_2(A) = 1.2 \times 10^6$, $\mathrm{cond}_G(A) \approx 5 \times 10^9$. $A = QTQ^T$, where Q is a random orthogonal matrix and T, obtained using `gallery('rschur',...)`, is in real Schur form with eigenvalues $\alpha_j + i\beta_j$, $\alpha_j = -j^2/10$, $\beta_j = -j$, $j = 1\colon n/2$ and $(2j, 2j+1)$ elements μ (thus μ controls the nonnormality of the matrix). We took $\mu = 25$.

Matrix 3: $\kappa_2(A) = 13$, $\mathrm{cond}_G(A) \approx 1$. $A = QTQ^T$ is the same as matrix 2, but with $\mu = 0$.

For the tests we needed the exact logarithm, which we approximated by X_* computed using our own implementation of the inverse scaling and squaring method. Our code computes a Schur decomposition, computes square roots by the Schur method [4], [9], [12], and uses the [8/8] Padé approximation once $\|A^{1/2^k} - I\|_1 \le 0.25$. (Then the Padé approximation has error safely less than u [16, sect. 3].)

For each matrix we applied Algorithm 7.1 with tolerance $\delta = \epsilon\|X_*\|_F/4$, with ϵ ranging from 10^{-16} to 10^{-1}. The results are shown in Figure 8.1. In each plot ϵ is on the x-axis. The plots in the first row show the total number $\sum_{i=1}^{s} k_i$ of inner DB iterations (using the notation of Algorithm 5.1), and those in the second row show the total number of matrix multiplications for the complete logarithm computation (counting a matrix inversion as a multiplication). In the third row is plotted an approximation $\|\widehat{X} - X_*\|_F/\|X_*\|_F$ to the relative error.

The number of square roots computed by the inverse scaling and squaring method for Matrices 1–3 was 7, 20, and 5, respectively. The corresponding operation counts are about $32n^3$–$40n^3$ flops. The number of incomplete square root stages used by Algorithm 7.1 for Matrices 1–3 was in the ranges 5–7, 18–20, and 4–5, respectively. From the second row of Figure 8.1 we can see that the operation count for Algorithm 7.1 varies between about $20n^3$ and $150n^3$ flops, and only for very relaxed tolerances does Algorithm 7.1 better the flop count of the inverse scaling and squaring method. However, these operation counts do not reflect the fact that Algorithm 7.1 is built from high-level computational kernels that can be implemented very efficiently.

The results reported are for the Y form of the DB iteration, as specified in Algorithm 5.1. The corresponding Z form performs similarly.

We make the following comments on the results.

1. The number of inner iterations and the number of matrix multiplications both vary with the tolerance δ by factors up to 3.2, confirming that incomplete square root iterations with careful choice of the degree of Padé approximation can produce substantial savings in work.
2. Ideally, the relative error would be approximately equal to ϵ. For Matrix 1 this is the case down to $\epsilon = 10^{-8}$, at which point the relative error levels off due to ill-conditioning: the cond_G term in (7.8) starts to dominate. For Matrix 2 the relative errors are approximately constant at about 10^{-6}. Given that $\mathrm{cond}_G(A) \approx 5 \times 10^9$ this is the level of relative error we would expect for the smallest δ. Why the relative error increases only slightly with increasing δ is unclear, but may be related to the large number of (incomplete) square roots required and the consequent rapid decrease in the convergence tolerance. For

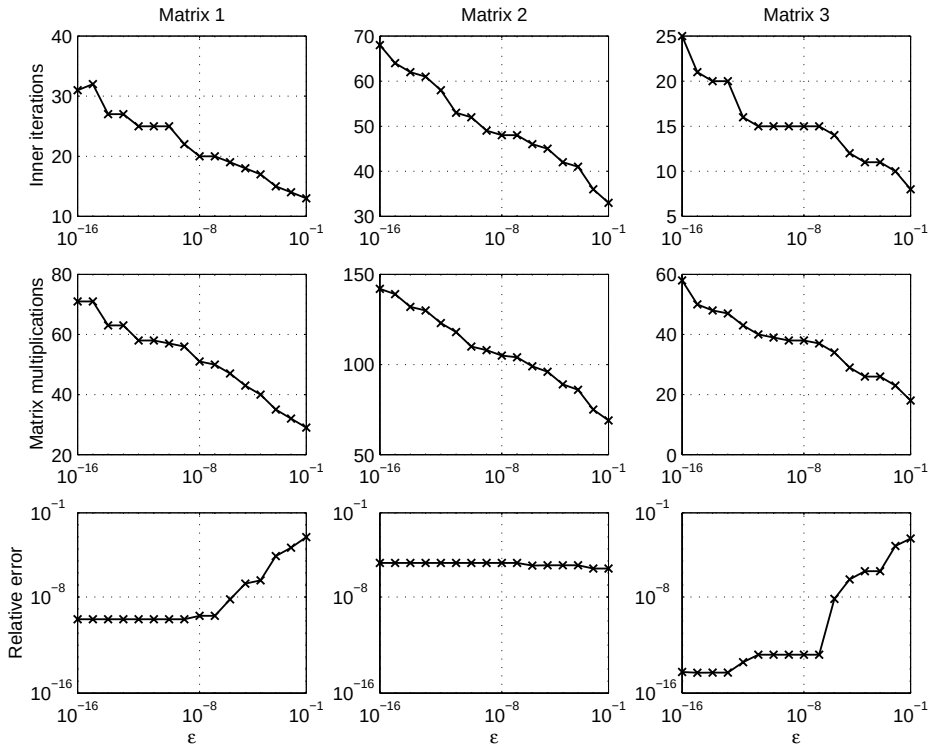

FIG. 8.1. *Results for Matrices* 1–3.

Matrix 3 the relative errors are somewhat less than ϵ, again for reasons that are not clear.

3. We also implemented the inverse scaling and squaring method using the DB iteration (3.1) with scaling, with the standard convergence test of the form $\|Y_{k+1} - Y_k\|/\|Y_{k+1}\| \le \theta$ and using the [8/8] Padé approximation. With $\theta = nu$ the number of inner iterations was 85, 506 (due to convergence problems, even with this relaxed tolerance), and 35 for Matrices 1–3, compared with 31, 68, and 25 for Algorithm 7.1 with $\delta = 10^{-16}\|X\|_F/4$; the accuracy of the computed logarithms was similar in both cases. The improved efficiency of Algorithm 7.1 is due to the better convergence test (based on $\|M_k - I\|$) and the use of the free approximation (5.2).

9. Conclusion. This work makes three main contributions. First, we have obtained a splitting result, Lemma 2.1, which gives conditions under which the logarithm of a matrix product is the sum of the logarithms. Second, we have derived a product form (3.6) of the DB iteration for the matrix square root; it trades a matrix inversion for a matrix multiplication and, unlike the original iteration, has a natural stopping test (based on $\|M_k - I\|$). We used the lemma and the iteration to derive a new version of the inverse scaling and squaring method for computing the matrix logarithm. The key features of our method are that it adapts itself to a specified accuracy (modulo the effects of roundoff) by carrying out incomplete square root computations and choosing a suitable Padé approximation, and that the computational kernels are matrix multiplication, LU factorization, and matrix inversion, making the method attractive for high-performance computation.

Acknowledgments. We thank the referees for their helpful comments.

REFERENCES

[1] R. C. Allen and S. A. Pruess, *An analysis of an inverse problem in ordinary differential equations*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 176–185.
[2] G. A. Baker, Jr., *Essentials of Padé Approximants*, Academic Press, New York, 1975.
[3] G. A. Baker, Jr. and P. Graves-Morris, *Padé Approximants*, Encyclopedia Math. Appl., 2nd ed., Cambridge University Press, Cambridge, England, 1996.
[4] A. Björck and S. Hammarling, *A Schur method for the square root of a matrix*, Linear Algebra Appl., 52/53 (1983), pp. 127–140.
[5] E. D. Denman and A. N. Beavers, Jr., *The matrix sign function and computations in systems*, Appl. Math. Comput., 2 (1976), pp. 63–94.
[6] L. Dieci, B. Morini, and A. Papini, *Computational techniques for real logarithms of matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 570–593.
[7] L. Dieci and A. Papini, *Conditioning and Padé approximation of the logarithm of a matrix*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 913–930.
[8] N. J. Higham, *Newton's method for the matrix square root*, Math. Comp., 46 (1986), pp. 537–549.
[9] N. J. Higham, *Computing real square roots of a real matrix*, Linear Algebra Appl., 88/89 (1987), pp. 405–430.
[10] N. J. Higham, *The matrix sign decomposition and its relation to the polar decomposition*, Linear Algebra Appl., 212/213 (1994), pp. 3–20.
[11] N. J. Higham, *Stable iterations for the matrix square root*, Numer. Algorithms, 15 (1997), pp. 227–242.
[12] N. J. Higham, *A New* `sqrtm` *for MATLAB*, Numerical Analysis Report 336, Manchester Centre for Computational Mathematics, Manchester, England, January 1999.
[13] N. J. Higham, *Evaluating Padé approximants of the matrix logarithm*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1126–1135.
[14] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, England, 1991.
[15] C. Kenney and A. J. Laub, *Condition estimates for matrix functions*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 191–209.
[16] C. Kenney and A. J. Laub, *Padé error estimates for the logarithm of a matrix*, Internat. J. Control, 50 (1989), pp. 707–730.
[17] C. Kenney and A. J. Laub, *Rational iterative methods for the matrix sign function*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 273–291.
[18] C. S. Kenney and A. J. Laub, *The matrix sign function*, IEEE Trans. Automat. Control, 40 (1995), pp. 1330–1348.
[19] C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Rev., 20 (1978), pp. 801–836.
[20] B. Singer and S. Spilerman, *The representation of social processes by Markov models*, Amer. J. Sociology, 82 (1976), pp. 1–54.
[21] R. C. Ward, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM J. Numer. Anal., 14 (1977), pp. 600–610.