Acta Numerica (2010), pp. 159–208
doi:10.1017/S0962492910000036

Computing matrix functions

Nicholas J. Higham and Awad H. Al-Mohy
School of Mathematics,
University of Manchester,
Manchester, M13 9PL, UK
E-mail: higham@maths.man.ac.uk, almohy@maths.man.ac.uk

The need to evaluate a function $f(A) \in \mathbb{C}^{n\times n}$ of a matrix $A \in \mathbb{C}^{n\times n}$ arises in a wide and growing number of applications, ranging from the numerical solution of differential equations to measures of the complexity of networks. We give a survey of numerical methods for evaluating matrix functions, along with a brief treatment of the underlying theory and a description of two recent applications. The survey is organized by classes of methods, which are broadly those based on similarity transformations, those employing approximation by polynomial or rational functions, and matrix iterations. Computation of the Fréchet derivative, which is important for condition number estimation, is also treated, along with the problem of computing $f(A)b$ without computing $f(A)$. A summary of available software completes the survey.

CONTENTS

1. Introduction

Matrix functions are as old as matrix algebra itself. In his memoir that initiated the study of matrix algebra, Cayley (1858) treated matrix square roots. The theory of matrix functions was subsequently developed by many mathematicians over the ensuing 100 years. Today, functions of matrices are widely used in science and engineering and are of growing interest, due to the succinct way they allow solutions to be expressed and recent advances in numerical algorithms for computing them. New applications are regularly being found, but the archetypal application of matrix functions is in the solution of differential equations. Early recognition of the important role of the matrix exponential in this regard can be found in the book *Elementary Matrices and Some Applications to Dynamics and Differential Equations* by aerospace engineers Frazer, Duncan and Collar (1938), which was 'the first book to treat matrices as a branch of applied mathematics' (Collar 1978).

This article provides a survey of numerical methods for computing matrix functions and is organized as follows. Section 2 describes some key elements of the theory of matrix functions. Two recent applications, to networks and roots of transition matrices, are described in Section 3. The following three sections describe methods grouped by type: those based on similarity transformations, those employing polynomial or rational approximations, and matrix iterations. Section 7 treats computation of the Fréchet derivative by five different approaches and explains how to estimate the condition number of a matrix function. The problem of computing the action of $f(A)$ on a vector is treated in Section 8, while Section 9 describes available software. An appendix gives some new results on the comparison between truncated Taylor series and Padé approximants within the scaling and squaring method for the matrix exponential.

Throughout, $\|\cdot\|$ denotes an arbitrary matrix norm unless otherwise stated. A flop is a floating point operation: $+$, $-$, $*$ or $/$. The unit roundoff is denoted by u and has the value $u = 2^{-53} \approx 1.1 \times 10^{-16}$ in IEEE double precision arithmetic. We write $\widetilde{\gamma}_k := cku/(1-cku)$ with c a small integer constant.

2. Theory

We are concerned with functions mapping $\mathbb{C}^{n\times n}$ to $\mathbb{C}^{n\times n}$ that are defined in terms of an underlying scalar function f. Thus, for example, $\det(A)$, the adjugate (or adjoint) matrix, and a matrix polynomial such as $p(X) = AX^2+BX+C$ (where all matrices are $n\times n$) are not matrix functions in the sense considered here, and elementwise evaluations such as $A \mapsto (\cos(a_{ij}))$ also are not of the required form.

The functions of a matrix in which we are interested can be defined in various ways. The multiplicity of definitions caused some confusion in the early years of the subject, until Rinehart (1955) showed all the definitions to be equivalent, modulo technical assumptions. We give two definitions, both of which are very useful in developing the theory.

2.1. Definitions

Definition 2.1. (Jordan form definition of $f(A)$) Let $A \in \mathbb{C}^{n\times n}$ have the Jordan canonical form $Z^{-1}AZ = J_A = \operatorname{diag}\bigl(J_1(\lambda_1), J_2(\lambda_2), \dots, J_p(\lambda_p)\bigr)$, where Z is non-singular,

$$J_k(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k\times m_k}, \tag{2.1}$$

and $m_1 + m_2 + \cdots + m_p = n$. Then

$$f(A) := Zf(J_A)Z^{-1} = Z \operatorname{diag}(f(J_k(\lambda_k)))Z^{-1}, \tag{2.2}$$

where

$$f(J_k(\lambda_k)) := \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \dfrac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix}. \tag{2.3}$$

When the function f is multivalued and A has a repeated eigenvalue occurring in more than one Jordan block (*i.e.*, A is derogatory), we will take the same branch for f and its derivatives in each Jordan block. This gives a *primary matrix function*. If different branches are taken for the same eigenvalue in two different Jordan blocks then a *non-primary matrix function* is obtained. We will be concerned here only with primary matrix functions, and it is these that are needed in most applications. For more on non-primary matrix functions see Higham (2008, Section 1.4).

Definition 2.2. (polynomial interpolation definition of $f(A)$) Let $\lambda_1, \dots, \lambda_s$ denote the distinct eigenvalues of $A \in \mathbb{C}^{n\times n}$ and let n_i be the index of λ_i, that is, the order of the largest Jordan block in which λ_i appears. Then $f(A) := r(A)$, where r is the unique Hermite interpolating polynomial of degree less than $\sum_{i=1}^{s} n_i$ that satisfies the interpolation conditions

$$r^{(j)}(\lambda_i) = f^{(j)}(\lambda_i), \qquad j = 0\colon n_i - 1, \quad i = 1\colon s. \tag{2.4}$$

In both these definitions the values $f^{(j)}(\lambda_i)$ appearing in (2.4) are assumed to exist, in which case f is said to be *defined on the spectrum of A*.

A proof of the equivalence of these two definitions can be found in Higham (2008, Theorem 1.12). The equivalence is easily demonstrated for the

$m_k \times m_k$ Jordan block $J_k(\lambda_k)$ in (2.1). The polynomial satisfying the interpolation conditions (2.4) is then

$$\begin{aligned} r(t) = f(\lambda_k) + (t-\lambda_k)f'(\lambda_k) + \frac{(t-\lambda_k)^2}{2!}f''(\lambda_k) + \cdots \\ + \frac{(t-\lambda_k)^{m_k-1}}{(m_k-1)!}f^{(m_k-1)}(\lambda_k), \end{aligned}$$

which is just the first m_k terms of the Taylor series of f about λ_k (assuming the Taylor series exists). Hence, from Definition 2.2,

$$\begin{aligned} f(J_k(\lambda_k)) &= r(J_k(\lambda_k)) \\ &= f(\lambda_k)I + (J_k(\lambda_k) - \lambda_k I)f'(\lambda_k) + \frac{(J_k(\lambda_k)-\lambda_k I)^2}{2!}f''(\lambda_k) + \cdots \\ &\quad + \frac{(J_k(\lambda_k)-\lambda_k I)^{m_k-1}}{(m_k-1)!}f^{(m_k-1)}(\lambda_k). \end{aligned}$$

The matrix $(J_k(\lambda_k) - \lambda_k I)^j$ is zero except for 1s on the jth superdiagonal. This expression for $f(J_k(\lambda_k))$ is therefore equal to that in (2.3).

2.2. Properties

One of the most important basic properties is that $f(A)$ is a polynomial in $A \in \mathbb{C}^{n\times n}$, which is immediate from Definition 2.2. However, the coefficients of that polynomial depend on A. This property is not surprising in view of the Cayley–Hamilton theorem, which says that any matrix satisfies its own characteristic equation: $q(A) = 0$, where $q(t) = \det(tI - A)$ is the characteristic polynomial. The theorem implies that the nth power of A, and inductively all higher powers, are expressible as a linear combination of $I, A, \dots, A^{n-1}$. Thus any power series in A can be reduced to a polynomial in A of degree at most $n-1$ (with coefficients depending on A).

Other important properties are collected in the next result, for a proof of which see Higham (2008, Theorem 1.13).

Theorem 2.3. Let $A \in \mathbb{C}^{n\times n}$ and let f be defined on the spectrum of A. Then

(a) $f(A)$ commutes with A;
(b) $f(A^T) = f(A)^T$;
(c) $f(XAX^{-1}) = Xf(A)X^{-1}$;
(d) the eigenvalues of $f(A)$ are $f(\lambda_i)$, where the λ_i are the eigenvalues of A;
(e) if $A = (A_{ij})$ is block triangular then $F = f(A)$ is block triangular with the same block structure as A, and $F_{ii} = f(A_{ii})$;
(f) if $A = \mathrm{diag}(A_{11}, A_{22}, \dots, A_{mm})$ is block diagonal then

$$f(A) = \mathrm{diag}\big(f(A_{11}), f(A_{22}), \dots, f(A_{mm})\big).$$

It is often convenient to represent a matrix function as a power series or Taylor series. The next result explains when such a series converges (Higham 2008, Theorem 4.7).

Theorem 2.4. (convergence of matrix Taylor series) Suppose f has a Taylor series expansion

$$f(z) = \sum_{k=0}^{\infty} a_k (z-\alpha)^k \qquad \left(a_k = \frac{f^{(k)}(\alpha)}{k!} \right) \tag{2.5}$$

with radius of convergence r. If $A \in \mathbb{C}^{n\times n}$ then $f(A)$ is defined and is given by

$$f(A) = \sum_{k=0}^{\infty} a_k (A-\alpha I)^k \tag{2.6}$$

if and only if each of the distinct eigenvalues $\lambda_1, \dots, \lambda_s$ of A satisfies one of the conditions

(a) $|\lambda_i - \alpha| < r$,
(b) $|\lambda_i - \alpha| = r$ and the series for $f^{(n_i-1)}(\lambda)$ (where n_i is the index of λ_i) is convergent at the point $\lambda = \lambda_i$, $i = 1\colon s$.

Four books treat the theory of matrix functions in detail and should be consulted for more information: Gantmacher (1959, Chapter 5), Horn and Johnson (1991, Chapter 6), Lancaster and Tismenetsky (1985, Chapter 9), and Higham (2008).

2.3. Particular functions

We now turn to the definitions of some specific functions. For functions having a power series with an infinite radius of convergence, the matrix function can be defined by evaluating the power series at a matrix, by Theorem 2.4. Thus the matrix exponential is given by

$$\mathrm{e}^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots \tag{2.7}$$

and the matrix cosine and sine by

$$\begin{aligned} \cos(A) &= I - \frac{A^2}{2!} + \frac{A^4}{4!} - \frac{A^6}{6!} + \cdots, \\ \sin(A) &= A - \frac{A^3}{3!} + \frac{A^5}{5!} - \frac{A^7}{7!} + \cdots. \end{aligned}$$

A natural question is to what extent scalar functional relations generalize to the matrix case. For example, are $(\mathrm{e}^A)^2 = \mathrm{e}^{2A}$, $\mathrm{e}^{\mathrm{i}A} = \cos(A) + \mathrm{i}\sin(A)$, and $\cos(2A) = 2\cos(A)^2 - I$ valid equalities for all A? The answer is yes for these particular examples. More generally, scalar identities in a single variable remain true with a matrix argument provided that all terms are defined and that the functions involved are single-valued. For multivalued functions such as the logarithm additional conditions may be needed to ensure the matrix identity is valid; an example is given below. The relevant general results can be found in Higham (2008, Section 1.3). Relations involving more than one variable do not usually generalize to matrices; for example, $e^{A+B} \neq e^A e^B$ in general.

An important function is the pth root of a matrix, where initially we assume p is a positive integer. For $A \in \mathbb{C}^{n\times n}$ we say X is a pth root of A if $X^p = A$. Note that defining pth roots implicitly via this equation gives a wider class of matrices than Definitions 2.1 and 2.2. For example, $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, but Definitions 2.1 and 2.2 provide only one square root of $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, namely itself. If A is singular with a defective zero eigenvalue then there are no primary pth roots, since the pth root function is not defined on the spectrum of A, but there may be non-primary pth roots (albeit not obtainable from Definitions 2.1 or 2.2). Conditions for the existence of pth roots of singular matrices are non-trivial (Psarrakos 2002). We will concentrate here on the non-singular case. Here there are always at least p pth roots and there are infinitely many if A is derogatory (that is, some eigenvalue appears in more than one Jordan block).

In the common case where A has no eigenvalues on $\mathbb{R}^-$, the closed negative real axis, there is a distinguished root that is real when A is real (Higham 2008, Theorem 7.2).

Theorem 2.5. (principal pth root) Let $A \in \mathbb{C}^{n\times n}$ have no eigenvalues on $\mathbb{R}^-$. There is a unique pth root X of A all of whose eigenvalues lie in the segment $\{ z : -\pi/p < \arg(z) < \pi/p \}$, and it is a primary matrix function of A. We refer to X as the *principal pth root* of A and write $X = A^{1/p}$. If A is real then $A^{1/p}$ is real.

In particular, *the principal square root* $A^{1/2}$ of a matrix A with no eigenvalues on $\mathbb{R}^-$ is the unique square root all of whose eigenvalues lie in the open right half-plane.

A logarithm of $A \in \mathbb{C}^{n\times n}$ can be defined as any matrix X such that $e^X = A$. A singular matrix has no logarithms, but for a non-singular matrix there are infinitely many. Indeed if $e^X = A$ then $e^{X+2\pi k i I} = A$ for all integers k. In practice it is usually the principal logarithm, defined in the next result, that is of interest (Higham 2008, Theorem 1.31).

Theorem 2.6. (principal logarithm) Let $A \in \mathbb{C}^{n\times n}$ have no eigenvalues on $\mathbb{R}^-$. There is a unique logarithm X of A all of whose eigenvalues lie in the strip $\{ z : -\pi < \operatorname{Im}(z) < \pi \}$. We refer to X as the *principal logarithm* of A and write $X = \log(A)$. If A is real then its principal logarithm is real.

From this point on, log always denotes the principal logarithm.

Care is needed in checking functional identities involving the logarithm. While the relation $\exp(\log(A)) = A$ is always true (for any logarithm, not just the principal one), by definition of the logarithm, the relation $\log(\exp(A)) = A$ holds if and only if $|\operatorname{Im}(\lambda_i)| < \pi$ for every eigenvalue λ_i of A (Higham 2008, Problem 1.39).

For $A \in \mathbb{C}^{n\times n}$ with no eigenvalues on $\mathbb{R}^-$, the logarithm provides a convenient way to define A^α for arbitrary real α: as $A^\alpha = \mathrm{e}^{\alpha \log(A)}$. From the relation in the previous paragraph it follows that

$$\log(A^\alpha) = \alpha \log(A), \qquad \alpha \in [-1,1]. \tag{2.8}$$

The matrix sign function, introduced by Roberts in 1971 (Roberts 1980), corresponds to the choice

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

in Definitions 2.1 and 2.2, for which $f^{(k)}(z) = 0$ for $k \geq 1$. Thus the matrix sign function is defined only for matrices $A \in \mathbb{C}^{n\times n}$ having no pure imaginary eigenvalues. If we arrange the Jordan canonical form as $A = Z \operatorname{diag}(J_1, J_2) Z^{-1}$, where the eigenvalues of $J_1 \in \mathbb{C}^{p\times p}$ lie in the open left half-plane and those of $J_2 \in \mathbb{C}^{q\times q}$ lie in the open right half-plane, then

$$\operatorname{sign}(A) = Z \begin{bmatrix} -I_p & 0 \\ 0 & I_q \end{bmatrix} Z^{-1}. \tag{2.9}$$

Another useful representation is (Higham 1994)

$$\operatorname{sign}(A) = A(A^2)^{-1/2}, \tag{2.10}$$

which generalizes the scalar formula $\operatorname{sign}(z) = z/(z^2)^{1/2}$.

3. Applications

Matrix functions are useful in a wide variety of applications. We describe just two recent ones here; more can be found in Higham (2008, Chapter 2).

3.1. Networks

Consider a network representing interactions between pairs of entities in a system. In recent years much work has focused on identifying computable measures that quantify characteristics of the network. Many measures are available in the literature, and they are typically expressed in terms of the network's associated undirected graph G with n nodes. The adjacency matrix $A \in \mathbb{R}^{n\times n}$ of the graph has (i,j) element equal to 1 if nodes i and j are connected and 0 otherwise. Assume $a_{ii} \equiv 0$, so that there are no loops in the graph. A walk of length m between two nodes i and j is an ordered list of nodes $i, k_1, k_2, \dots, k_{m-1}, j$ such that successive nodes in the list are connected; the nodes need not be distinct and any of them may be i or j. When $i = j$ the walk starts and ends at the same node and is called closed. The walk is a path if all the nodes in the walk are distinct. Assume that the graph is connected, so that a path exists between any two distinct nodes. It is a standard fact in graph theory that the (i, j) element of A^m is the number of different walks, if $i \neq j$, or closed walks, if $i = j$, of length m between nodes i and j. A variety of measures have been built by combining different walk lengths into a single number. Estrada and Rodríguez-Velázquez (2005*b*) define the *subgraph centrality* of node i – a measure of its 'well-connectedness' – by

$$SC_i = \left(I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots \right)_{ii} = (\mathrm{e}^A)_{ii}.$$

By combining walks of all possible lengths connecting node i to itself, and applying a weighting that decreases rapidly with the walk length, the subgraph centrality aims to capture the participation of the node in question in all subgraphs in the network. The sum of all subgraph centralities of the nodes in the graph is the *Estrada index*: $\mathrm{trace}(\mathrm{e}^A)$. Based on similar reasoning, Estrada and Hatano (2008) define the *communicability* between nodes i and j – a measure of how easy it is for 'information' to pass from node i to node j (and a generalization of the notion of shortest path between the nodes) – by

$$C_{ij} = \left(I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots \right)_{ij} = (\mathrm{e}^A)_{ij}.$$

Finally, the *betweenness* of node r is defined in Estrada, D. J. Higham and Hatano (2009) by

$$\frac{1}{(n-1)^2 - (n-1)} \sum_{\substack{i,j \\ i \neq j, i \neq r, j \neq r}} \frac{(\mathrm{e}^A - \mathrm{e}^{A - E_r})_{ij}}{(\mathrm{e}^A)_{ij}},$$

where E_r is zero except in row and column r, where it agrees with A. The betweenness measures the relative change in communicability when node r is removed from the network. Experiments in the papers cited above show that these three measures can provide useful information about practically occurring networks that is not revealed by most other measures. In this description A is symmetric, but these concepts can be extended to directed graphs, for which the adjacency matrix is non-symmetric. Of course, the matrix exponential owes its appearance to the choice of weights in the sums over walklengths. Other weights could be chosen, resulting in different matrix functions in the definitions; see Estrada and D. J. Higham (2008).

When the elements of A not only indicate the existence of a link between nodes i and j but also assign a positive weight to a link, it is natural to normalize these definitions. Crofts and D. J. Higham (2009) generalize the definition of communicability to

$$C_{ij} = \mathrm{e}^{D^{-1/2}AD^{-1/2}},$$

where $D = \operatorname{diag}(d_i)$ and $d_i = \sum_{k=1}^{n} a_{ik}$ is the generalized degree of node i. They show how this communicability measure is a useful tool in clustering patients with brain disorders.

Finally, we note that Estrada and Rodríguez-Velázquez (2005*a*) propose $\beta(A) = \operatorname{trace}(\cosh(A))/\operatorname{trace}(\mathrm{e}^A)$ as a measure of how close a graph is to being bipartite: $\beta(A) \le 1$ with $\beta(A) = 1$ if and only if the graph G is bipartite.

3.2. Roots of transition matrices

A transition matrix is a stochastic matrix: a square matrix with nonnegative entries and row sums equal to 1. In credit risk, a transition matrix records the probabilities of a firm's transition from one credit rating to another over a given time interval (Jarrow, Lando and Turnbull 1997). The shortest period over which a transition matrix can be estimated is typically one year, and annual transition matrices can be obtained from rating agencies such as Moody's Investors Service and Standard & Poor's. However, for valuation purposes, a transition matrix for a period shorter than one year is usually needed. A short-term transition matrix can be obtained by computing a root of an annual transition matrix. A six-month transition matrix, for example, is a square root of the annual transition matrix. This property has led to interest in the finance literature in the computation or approximation of roots of transition matrices (Israel, Rosenthal and Wei 2001, Kreinin and Sidelnikova 2001). Exactly the same mathematical problem arises in Markov models of chronic diseases, where the transition matrix is built from observations of the progression in patients of a disease through different severity states. Again, the observations are at an interval longer than the short time intervals required for study and the need for a matrix root arises (Charitos, de Waal and van der Gaag 2008). An early discussion of this problem, which identifies the need for roots of transition matrices in models of business and trade, is that of Waugh and Abel (1967).

These applications require a stochastic root of a given stochastic matrix A, that is, a stochastic matrix X such that $X^p = A$, where p is typically an integer, but could be rational. A number of questions arise: does such a root exist; if so, how can one be computed; and what kind of approximation should be used if a stochastic root does not exist. These are investigated in Higham and Lin (2009) and the references therein.

More generally, matrix roots A^α with a real α arise in, for example, fractional differential equations (Ilić, Turner and Simpson 2009), discrete representations of norms corresponding to finite element discretizations of fractional Sobolev spaces (Arioli and Loghin 2009), and the computation of geodesic-midpoints in neural networks (Fiori 2008).

In the next three sections we consider methods based on three general approaches: similarity transformations, polynomial or rational approximations, and matrix iterations. We do not consider methods based on Definitions 2.1 or 2.2 because neither definition leads to an efficient and numerically reliable method in general.

4. Similarity transformations

The use of similarity transformations, considered in this section, rests on the identity $f(XAX^{-1}) = Xf(A)X^{-1}$ from Theorem 2.3(c). The aim is to choose X so that f is more easily evaluated at the matrix $B = XAX^{-1}$ than at A. When A is diagonalizable, B can be taken to be diagonal, and evaluation of $f(B)$ is then trivial.

In finite precision arithmetic, this approach is reliable only if X is well-conditioned, that is, if the condition number $\kappa(X) = \|X\|\|X^{-1}\|$ is not too large. Ideally, X will be unitary, so that in the 2-norm $\kappa_2(X) = 1$. For Hermitian A, or more generally normal A, the spectral decomposition $A = QDQ^*$ with Q unitary and D diagonal is always possible, and if this decomposition can be computed then the formula $f(A) = Qf(D)Q^*$ provides an excellent way of computing $f(A)$.

For general A, if X is restricted to be unitary then the furthest that A can be reduced is to Schur form: $A = QTQ^*$, where Q is unitary and T is upper triangular. This decomposition is computed by the QR algorithm. The problem is now reduced to that of evaluating f at a triangular matrix. The following result gives an explicit formula for the evaluation.

Theorem 4.1. (function of triangular matrix) Let $T \in \mathbb{C}^{n\times n}$ be upper triangular and suppose that f is defined on the spectrum of T. Then $F = f(T)$ is upper triangular with $f_{ii} = f(t_{ii})$ and

$$f_{ij} = \sum_{(s_0,\dots,s_k)\in S_{ij}} t_{s_0,s_1} t_{s_1,s_2} \dots t_{s_{k-1},s_k} f[\lambda_{s_0},\dots,\lambda_{s_k}], \tag{4.1}$$

where $\lambda_i = t_{ii}$, S_{ij} is the set of all strictly increasing sequences of integers that start at i and end at j, and $f[\lambda_{s_0},\dots,\lambda_{s_k}]$ is the kth-order divided difference of f at $\lambda_{s_0},\dots,\lambda_{s_k}$.

Proof. See Davis (1973), Descloux (1963), or Van Loan (1975). □

While theoretically interesting, the formula (4.1) is of limited computational interest due to its exponential cost in n. However, the case $n = 2$ is worth noting. For $\lambda_1 \neq \lambda_2$ we have

$$f\left(\begin{bmatrix} \lambda_1 & t_{12} \\ 0 & \lambda_2 \end{bmatrix}\right) = \begin{bmatrix} f(\lambda_1) & t_{12}\dfrac{f(\lambda_2) - f(\lambda_1)}{\lambda_2 - \lambda_1} \\ 0 & f(\lambda_2) \end{bmatrix}. \tag{4.2}$$

When $\lambda_1 = \lambda_2 = \lambda$ we have, using a standard relation for confluent divided differences (Higham 2008, Section B.16),

$$f\left(\begin{bmatrix} \lambda & t_{12} \\ 0 & \lambda \end{bmatrix}\right) = \begin{bmatrix} f(\lambda) & t_{12}f'(\lambda) \\ 0 & f(\lambda) \end{bmatrix}. \tag{4.3}$$

(This is a special case of (7.10) below.)

A much better way to compute $f(T)$ is from a recurrence of Parlett (1976). From Theorem 2.3 we know that $F = f(T)$ is upper triangular with diagonal elements $f(t_{ii})$ and that it commutes with T. The elements in the strict upper triangle are determined by solving the equation $FT = TF$ in an appropriate order.

Algorithm 4.2. (Parlett recurrence) Given an upper triangular $T \in \mathbb{C}^{n\times n}$ with distinct diagonal elements and a function f defined on the spectrum of T, this algorithm computes $F = f(T)$ using Parlett's recurrence.

```
1  for $j = 1\colon n$
2      $f_{jj} = f(t_{jj})$
3      for $i = j-1\colon -1\colon 1$
4          $f_{ij} = t_{ij}\dfrac{f_{ii} - f_{jj}}{t_{ii} - t_{jj}} + \left(\sum_{k=i+1}^{j-1} f_{ik}t_{kj} - t_{ik}f_{kj}\right) / (t_{ii} - t_{jj})$
5      end
6  end
```

Cost: $2n^3/3$ flops.

The recurrence breaks down when $t_{ii} = t_{jj}$ for some $i \neq j$. In this case, T can be regarded as a block triangular matrix $T = (T_{ij})$, with square diagonal blocks, possibly of different sizes. Then $F = (F_{ij})$ has the same block triangular structure by Theorem 2.3(e) and by equating (i, j) blocks in $TF = FT$ we obtain

$$T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1} (F_{ik}T_{kj} - T_{ik}F_{kj}), \qquad i < j. \tag{4.4}$$

The Sylvester equation (4.4) is non-singular precisely when T_{ii} and T_{jj} have no eigenvalue in common. Assuming that this property holds for all i and j we obtain the following algorithm.

Algorithm 4.3. (block Parlett recurrence) Given an upper triangular matrix $T = (T_{ij}) \in \mathbb{C}^{n\times n}$ partitioned in block $m \times m$ form with no two diagonal blocks having an eigenvalue in common, and a function f defined on the spectrum of T, this algorithm computes $F = f(T)$ using the block form of Parlett's recurrence.

```
1  for $j = 1\colon m$
2      $F_{jj} = f(T_{jj})$
3      for $i = j-1\colon -1\colon 1$
4          Solve for $F_{ij}$ the Sylvester equation (4.4).
5      end
6  end
```

Cost: Dependent on the block sizes and f.

The problems of how to evaluate $f(T_{jj})$ and how to achieve a blocking with the desired properties are considered in the next section.

4.1. Schur–Parlett algorithm

In order to use the block Parlett recurrence we need to reorder and partition the matrix T so that no two diagonal blocks have an eigenvalue in common; here, reordering means applying a unitary similarity transformation to permute the diagonal elements whilst preserving triangularity. But in doing the reordering and defining the block structure we also need to take into account the difficulty of evaluating f at the diagonal blocks T_{ii} and the propagation of errors in the recurrence.

Consider first the evaluation of $f(T_{ii})$. For notational simplicity, let $T \in \mathbb{C}^{n\times n}$ play the role of T_{ii}. Assume that derivatives of f are available and that f has a Taylor series with an infinite radius of convergence. Then we can evaluate $f(T)$ from the Taylor series. Writing

$$T = \sigma I + M, \qquad \sigma = \operatorname{trace}(T)/n, \tag{4.5}$$

we evaluate f about the mean, σ, of the eigenvalues:

$$f(T) = \sum_{k=0}^{\infty} \frac{f^{(k)}(\sigma)}{k!} M^k. \tag{4.6}$$

If the eigenvalues of T are sufficiently close then the powers of M can be expected to decay quickly after the $(n-1)$st, and so a suitable truncation of (4.6) should yield good accuracy; indeed, in the special case where T has only one distinct eigenvalue ($t_{ii} \equiv \sigma$), M is nilpotent and $M^n = 0$.

By bounding the truncation error in the Taylor series we can construct the following algorithm that adaptively chooses the number of terms in order to achieve the desired accuracy; see Davies and Higham (2003) or Higham (2008, Section 9.1) for the details of the derivation.

Algorithm 4.4. (evaluate function of atomic block) Given a triangular matrix $T \in \mathbb{C}^{n\times n}$ whose eigenvalues $\lambda_1,\dots,\lambda_n$ are 'close,' a function f having a Taylor series with an infinite radius of convergence, and the ability to evaluate derivatives of f, this algorithm computes $F = f(T)$ using a truncated Taylor series.

```
1  $\sigma = n^{-1}\sum_{i=1}^{n}\lambda_i$, $M = T - \sigma I$, tol $= u$
2  $\mu = \|y\|_\infty$, where $y$ solves $(I - |N|)y = e$ and $N$ is the strictly
   upper triangular part of $T$. % $\mu = \|(I - |N|)^{-1}\|_\infty$
3  $F_0 = f(\sigma)I_n$
4  $P = M$
5  for $s = 1\colon\infty$
6      $F_s = F_{s-1} + f^{(s)}(\sigma)P$
7      $P = PM/(s+1)$
8      if $\|F_s - F_{s-1}\|_F \le$ tol$\|F_s\|_F$
          % Successive terms are close so check a truncation error bound.
9         Estimate or bound $\Delta = \max_{0\le r\le n-1}\omega_{s+r+1}/r!$, where
          $\omega_k = \max\{\,|f^{(k)}(t_{ii})| : i = 1\colon n\,\}$.
10        if $\mu\Delta\|P\|_F \le$ tol$\|F_s\|_F$, $F = F_s$, quit, end
11     end
12 end
```

Algorithm 4.4 costs $O(n^4)$ flops, since even if T has constant diagonal, so that M is nilpotent with $M^n = 0$, the algorithm may need to form the first $n-1$ powers of M. However, n here is the size of a block, and in most cases the blocks will be of much smaller dimension than the original matrix. Also, M is an upper triangular matrix, so forming all the powers $M^2,\dots,M^{n-1}$ costs $n^4/3$ flops – a factor 6 less than the flop count for multiplying full matrices.

Now we consider how to reorder and block the Schur factor T. We wish the Sylvester equations (4.4) to be well-conditioned, so that the equations are solved accurately and errors do not grow substantially within the recurrence. The conditioning of (4.4) is measured[1] by $\operatorname{sep}(T_{ii},T_{jj})^{-1}$, where

$$\operatorname{sep}(T_{ii},T_{jj}) = \min_{X\neq 0}\frac{\|T_{ii}X - XT_{jj}\|_F}{\|X\|_F}$$

is the separation of the diagonal blocks T_{ii} and T_{jj}. The separation is expensive to compute or estimate accurately, but we can approximate it by a lower bound that is cheap to evaluate:

$$\operatorname{sep}(T_{ii},T_{jj})^{-1} \approx \frac{1}{\min\{\,|\lambda-\mu| : \lambda\in\Lambda(T_{ii}),\ \mu\in\Lambda(T_{jj})\,\}}, \tag{4.7}$$

[1] In fact, this is a commonly used upper bound for the (interesting part of the) true condition number (Higham 2002, Section 10.3).

where $\Lambda(\cdot)$ denotes the spectrum. A second goal of the reordering is to produce diagonal blocks that have close eigenvalues, so that Algorithm 4.4 is efficient.

Denote the reordered upper triangular matrix by $\widetilde{T} = U^*TU = (\widetilde{T}_{ij})$, where U is unitary. A reasonable way to satisfy the above requirements for the diagonal blocks is to ensure that

(1) *separation between blocks*:

$$\min\{\, |\lambda - \mu| : \lambda \in \Lambda(\widetilde{T}_{ii}),\ \mu \in \Lambda(\widetilde{T}_{jj}),\ i \neq j \,\} > \delta,$$

(2) *separation within blocks*: for every block $\widetilde{T}_{ii}$ with dimension bigger than 1, for every $\lambda \in \Lambda(\widetilde{T}_{ii})$ there is a $\mu \in \Lambda(\widetilde{T}_{ii})$ with $\mu \neq \lambda$ such that $|\lambda - \mu| \le \delta$.

Here, $\delta > 0$ is a blocking parameter. These conditions produce a blocking for which the spectra of different diagonal blocks are at least distance δ apart (thus ensuring that the right-hand side of (4.7) is at most δ^{-1}) while the eigenvalues within each diagonal block are close, in the sense that every eigenvalue is at most distance δ from some other eigenvalue. How to obtain such a reordering is described in Davies and Higham (2003) and Higham (2008, Section 9.3).

The overall algorithm is as follows.

Algorithm 4.5. (Schur–Parlett algorithm) Given $A \in \mathbb{C}^{n\times n}$, a function f having a Taylor series with an infinite radius of convergence, and the ability to evaluate derivatives of f, this algorithm computes $F = f(A)$.

```
1  Compute the Schur decomposition $A = QTQ^*$.
2  If $T$ is diagonal, $F = f(T)$, goto line 10, end
3  Reorder and partition $T$ and update $Q$ to satisfy the conditions above
   with $\delta = 0.1$.
   % Now $A = QTQ^*$ is our reordered Schur decomposition,
   % with block $m \times m$ $T$.
4  for $j = 1{:}\,m$
5      Use Algorithm 4.4 to evaluate $F_{ii} = f(T_{ii})$.
6      for $i = j - 1{:}\,{-1}{:}\,1$
7          Solve the Sylvester equation (4.4) for $F_{ij}$.
8      end
9  end
10 $F = QFQ^*$
```

Cost: Roughly between $28n^3$ flops and $n^4/3$ flops, and dependent greatly on the eigenvalue distribution of A.

Algorithm 4.5 is the best available method for general functions f and it usually performs in a forward stable manner, that is, the forward error is usually bounded by a modest multiple of $\operatorname{cond}(f, A)u$, where the condition number $\operatorname{cond}(f, A)$ is defined in Section 7.1. However, the algorithm can be unstable, and empirically this seems most likely for matrices having large Jordan blocks. Instability can usually be cured by increasing δ to 0.2, but Davies and Higham (2003) have shown experimentally that the algorithm can be unstable for all choices of δ.

4.2. Schur method for matrix roots

For some functions it is possible to compute $f(T)$ by a different method than the Parlett recurrence. The main cases of practical interest are pth roots. If X is a square root of the upper triangular matrix T then the diagonal of X is readily determined and the superdiagonal elements can be obtained from the equation $X^2 = T$. The corresponding Schur method, due to Björck and Hammarling (1983), can be arranged as follows.

Algorithm 4.6. (Schur method for square root) Given a non-singular $A \in \mathbb{C}^{n\times n}$ this algorithm computes $X = \sqrt{A}$ via a Schur decomposition, where $\sqrt{\cdot}$ denotes any primary square root.

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

Cost: $28\frac{1}{3}n^3$ flops.

Note that Algorithm 4.6 breaks down if $u_{ii} = -u_{jj}$ for some i and j, which can happen only if T has two equal diagonal elements that are mapped to different square roots – in other words, the algorithm is attempting to compute a non-primary square root. The algorithm can compute any primary square root by a suitable choice of the square roots of the diagonal elements. The computed square root $\widehat{X}$ satisfies

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

which is as good a residual bound as the rounded exact square root satisfies and so is essentially optimal.

For computing real square roots of real matrices it is more appropriate to employ the real Schur decomposition and thereby work entirely in real arithmetic; see Higham (1987) or Higham (2008, Section 6.2).

For pth roots, more complicated recurrences can be used to solve $X^p = T$, as shown by Smith (2003). The overall Schur algorithm has a cost of $O(pn^3)$ flops. If $p = p_1 p_2 \dots p_t$ is composite, the pth root can be computed by successively computing the p_1th, p_2th, …, p_tth roots, with a computational saving. Greco and Iannazzo (2010) show how to use the binary representation of p to compute X at a cost of $O(n^2 p + n^3 \log_2 p)$ flops, which achieves further savings when p is large and not highly composite.

4.3. Block diagonalization

If we are willing to use non-unitary transformations then we can go beyond the Schur form to block diagonal form to obtain $A = XDX^{-1}$, where D is block diagonal. Such a form can be obtained by first computing the Schur form and then eliminating off-diagonal blocks by solving Sylvester equations (Bavely and Stewart 1979, Golub and Van Loan 1996, Section 7.6.3, Lavallée, Malyshev and Sadkane 1997). In order to guarantee a well-conditioned X a bound must be imposed on the condition numbers of the individual transformations, and this bound will be a parameter in the algorithm. The attraction of block diagonal form is that computing $f(A)$ reduces to computing $f(D)$, and hence to computing $f(D_{ii})$ for each diagonal block D_{ii}, and the D_{ii} are triangular if obtained as indicated above. However, evaluating $f(D_{ii})$ is still a non-trivial calculation because Algorithms 4.2 and 4.4 may not be applicable. The Schur–Parlett method and the block diagonalization method are closely related. Both employ a Schur decomposition, both solve Sylvester equations, and both must compute $f(T_{ii})$ for triangular blocks T_{ii}. Parlett and Ng (1985, Section 5) show that the two methods are mathematically equivalent, differing only in the order in which two commuting Sylvester operators are applied.

5. Polynomial and rational approximations

A natural way to approximate $f(A)$ is to mimic what is often done for scalar functions: to approximate $f(A)$ by $r(A)$ where r is some suitable polynomial or rational approximation to f. From scalar approximation theory we may know some region of $\mathbb{C}$ in which $f(z) \approx r(z)$ is a good approximation. However, if the spectrum of A lies in this region there is no guarantee that the matrix approximation $f(A) \approx r(A)$ is equally good. Indeed if A is diagonalizable with $A = ZDZ^{-1}$, then $f(A) - r(A) = Z(f(D) - r(D))Z^{-1}$, so that $\|f(A) - r(A)\| \le \kappa(Z)\|f(D) - r(D)\|$. Hence the error in the matrix approximation is potentially as much as $\kappa(Z)$ times larger than the error in the scalar approximation. If A is normal, so that we can take $\kappa_2(Z) = 1$, then the scalar and matrix approximation problems are essentially the same. But for non-normal matrices, achieving a good approximation requires more than simply approximating well at the eigenvalues.

A general framework for approximating $f(A)$ is as follows.

Framework 5.1. (for approximating $f(A)$)

(1) Choose a suitable rational approximation r and a transformation function g and set $A \leftarrow g(A)$.
(2) Compute $X = r(A)$ by some appropriate scheme.
(3) Apply transformations to X that undo the effect of the initial transformation on A.

The purpose of step (1) is to transform A so that $f(A) \approx r(A)$ is a sufficiently good approximation. Both g and r may depend on A. In the following subsections we describe how this can be done for some specific functions f.

For a polynomial approximation at step (2) the natural choice when f has a Taylor series $f(x) = \sum_{i=0}^{\infty} a_i x^i$ is the truncated Taylor series

$$T_k(A) = \sum_{i=0}^{k} a_i A^i. \tag{5.1}$$

Among rational approximations the Padé approximants prove to be particularly useful. Recall that $r_{km}(x) = p_{km}(x)/q_{km}(x)$ is a $[k/m]$ Padé approximant of f if p_{km} and q_{km} are polynomials of degree at most k and m, respectively, $q_{km}(0) = 1$, and

$$f(x) - r_{km}(x) = O(x^{k+m+1}) \tag{5.2}$$

(Brezinski and Van Iseghem 1995, Baker and Graves-Morris 1996). Thus a Padé approximant reproduces as many terms as possible of the Taylor series about the origin. If a $[k/m]$ Padé approximant exists then it is unique.

For the evaluation at step (2) there are many possibilities and which is best depends very much on r. Consider, first, the case of polynomial r. For a matrix argument, evaluation by Horner's method is not of optimal efficiency for polynomials of degree 4 and higher. More efficient alternatives are based on explicitly forming certain matrix powers, as is done in a general method of Paterson and Stockmeyer (1973) and a variant of Van Loan (1979). For rational $r = p/q$, the possibilities are more diverse:

(1) Evaluate $p(A)$ and $q(A)$ and then solve the multiple right-hand side system $q(A)r(A) = p(A)$.
(2) Evaluate $r(A)$ from a continued fraction representation of r in either top-down or bottom-up fashion.
(3) Evaluate $r(A)$ from a partial fraction representation, ideally with linear factors but possibly admitting higher-degree factors in order to keep the coefficients real.

The choice of scheme should take into account numerical stability and will, in general, depend on f. For details of all the above schemes see Higham (2008, Sections 4.2, 4.4.3).

We now focus on some specific transcendental functions of interest.

5.1. Matrix exponential

For Hermitian A, best L_∞ approximations to e^x can be employed, for which matrix level error bounds follow immediately from error bounds at the scalar level, as noted at the start of this section. However, we concentrate in this section on general matrices.

Truncating the Taylor series is one way to obtain an approximation. Padé approximation is particularly attractive because the $[k/m]$ Padé approximants to the exponential function are known explicitly for all k and m:

$$p_{km}(x) = \sum_{j=0}^{k} \frac{(k+m-j)!\,k!}{(k+m)!\,(k-j)!}\frac{x^j}{j!}, \quad q_{km}(x) = \sum_{j=0}^{m} \frac{(k+m-j)!\,m!}{(k+m)!\,(m-j)!}\frac{(-x)^j}{j!}. \tag{5.3}$$

Of course, the truncated Taylor series is just the $[k/0]$ Padé approximant. Among Padé approximants the diagonal approximants, $r_m \equiv r_{mm}$, are preferred, because they are more efficient (Moler and Van Loan 1978) and they have advantageous stability properties (Varga 2000, Chapter 8).

The standard way in which to use these approximations is within the scaling and squaring method. This method scales the matrix according to $A \leftarrow 2^{-s}A$, with s chosen so that $\|A\|$ is of order 1, evaluates $X = r_m(A)$ for some suitable diagonal Padé approximant r_m, then undoes the effect of the scaling by repeated squaring: $X \leftarrow X^{2^s}$. The method originates with Lawson (1967). Moler and Van Loan[2] (1978) give a backward error analysis that shows that $r_m(2^{-s}A)^{2^s} = \mathrm{e}^{A+E}$ with an explicit bound on E expressed in terms of $\|A\|$. Here, E is measuring the effect of truncation errors in the Padé approximant and exact arithmetic is assumed. Based on this analysis they give a table indicating the optimal choice of s and m for a given $\|A\|$ and ϵ, where $\|E\| \le \epsilon\|A\|$ is required, and also conclude from it that Padé approximants are more efficient than Taylor series. The analysis in Moler and Van Loan (1978) led to implementations taking a fixed choice of m and s. For example, the function `expm` in version 7.1 and earlier of MATLAB used $m = 6$ and scaled so that $\|2^{-s}A\|_\infty \le 0.5$. However, a more efficient and more accurate (in floating point arithmetic) algorithm can be obtained by using higher-degree approximants, with the choice of m and s determined from sharper truncation error bounds.

[2] This classic paper was reprinted with an update in Moler and Van Loan (2003).

We need the following result of Al-Mohy and Higham (2009*b*), which is a slightly sharper version of a result of Higham (2005). Let $\nu_m = \min\{\, |t| : q_m(t) = 0 \,\}$ and

$$\Omega_m = \{\, X \in \mathbb{C}^{n\times n} : \rho(\mathrm{e}^{-X} r_m(X) - I) < 1 \text{ and } \rho(X) < \nu_m \,\},$$

where ρ denotes the spectral radius.

Lemma 5.1. For $A \in \Omega_m$ we have $r_m(2^{-s}A)^{2^s} = \mathrm{e}^{A+\Delta A}$, where $\Delta A = h_{2m+1}(2^{-s}A)$ and $h_{2m+1}(x) = \log(\mathrm{e}^{-x} r_m(x))$. Moreover,

$$\frac{\|\Delta A\|}{\|A\|} \le \frac{\sum_{k=2m+1}^{\infty} |c_k|\, \|2^{-s}A\|^k}{\|2^{-s}A\|}, \tag{5.4}$$

where $h_{2m+1}(x) = \sum_{k=2m+1}^{\infty} c_k\, x^k$.

The bound in the lemma can be evaluated for a given value of $\|A\|$ by determining the coefficients c_k symbolically and then summing a suitable number of the terms in the numerator at high precision. Using a zero-finder we can therefore determine, for a range of m, the largest value of $\|2^{-s}A\|$, denoted by θ_m, such that the bound is no larger than $u = 2^{-53} \approx 1.1 \times 10^{-16}$, the unit roundoff for IEEE double precision arithmetic; some of these constants are given in Table 5.1. By taking account of the cost of evaluating $r_m(2^{-s}A)^{2^s}$, we can determine the optimal choice of s and m for a given $\|A\|$. The analysis also needs to ensure that the effect of rounding errors in the evaluation of r_m is not significant. The ideas above were used by Higham (2005) to derive the following algorithm.

Algorithm 5.2. (scaling and squaring) This algorithm evaluates the matrix exponential $X = \mathrm{e}^A$ of $A \in \mathbb{C}^{n\times n}$ using the scaling and squaring method. It uses the constants θ_m given in Table 5.1. The algorithm is intended for IEEE double precision arithmetic.

```
1 for $m = [3\ 5\ 7\ 9]$
2     if $\|A\|_1 \le \theta_m$, evaluate $X = r_m(A)$, quit, end
3 end
4 $A \leftarrow A/2^s$ with $s \ge 0$ a minimal integer such that $\|A/2^s\|_1 \le \theta_{13}$
  ($i.e.$, $s = \lceil \log_2(\|A\|_1/\theta_{13}) \rceil$).
5 Evaluate $X = r_{13}(A)$.
6 $X \leftarrow X^{2^s}$ by repeated squaring.
```

The details of how to evaluate r_m can be found in Higham (2005), Higham (2008, Section 10.3), or Higham (2009).

Although the derivation of Algorithm 5.2 is based on minimizing the cost of the computation, it is a welcome side effect that the accuracy is usually also improved compared with the previous choices of s and m. The reason is that the most dangerous phase of the algorithm – whose effect on the overall

Table 5.1. Constants θ_m needed in Algorithm 5.2.

m	θ_m
3	1.495585217958292e-2
5	2.539398330063230e-1
7	9.504178996162932e-1
9	2.097847961257068e0
13	5.371920351148152e0

numerical stability is still not well understood – is the s squarings, and the use of high Padé degrees up to $m = 13$ together with sharper truncation error bounds tends to produce smaller values of s and hence reduce the number of squarings.

The long-standing preference of Padé approximants over Taylor series within the scaling and squaring method needs revisiting with the aid of the sharper bound of Lemma 5.1. This is done in the Appendix, where we find that Padé approximants remain preferable.

We now reconsider the choice of s. Figure 5.1 shows what happens when we force Algorithm 5.2 to choose a particular value of s. For two different matrices we plot the relative error $\|X - \widehat{X}\|_1/\|X\|_1$, where X is an accurate approximation to e^A computed at high precision in MATLAB using the Symbolic Math Toolbox. For the first matrix, `-magic(6)^2` in MATLAB, we see in the first plot of Figure 5.1 that the relative error is of order 1 with no scaling ($s = 0$) and it reaches a minimum of 2.2×10^{-13} for $s = 11$. Algorithm 5.2 chooses $s = 12$, which is nearly optimal as regards the forward error. However, for the matrix

$$A = \begin{bmatrix} -1 & -1 & -10^4 & -10^4 \\ -1 & -1 & -10^4 & -10^4 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & -1 \end{bmatrix}, \tag{5.5}$$

we see in the second plot of Figure 5.1 that the error grows mostly monotonically with s. The value $s = 12$ chosen by Algorithm 5.2 is much larger than the optimal value $s = 0$; thus Algorithm 5.2 computes a less accurate result than the optimum and at significantly greater cost. An explanation for this behaviour can be seen from the following MATLAB computation:

```
for i = 1:10, fprintf('%9.1e ', norm(A^i)^(1/i)), end
2.0e+004  2.8e+002  6.2e+001  2.8e+001  1.7e+001  1.3e+001
9.8e+000  8.2e+000  7.1e+000  6.3e+000
```

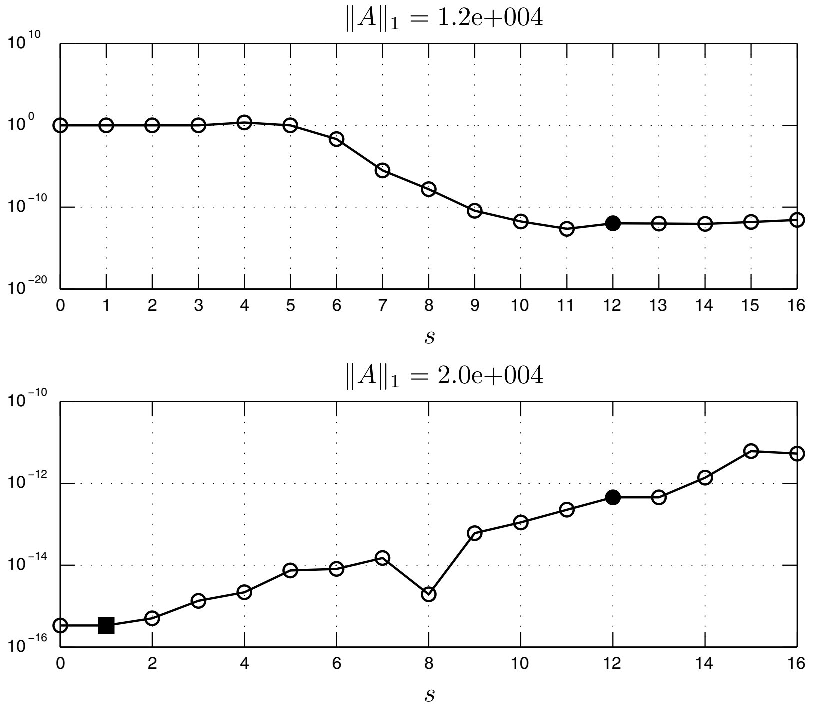

Figure 5.1. Scaling argument s versus relative error of scaling and squaring method, for A given by `-magic(6)^2` (*top*) and (5.5) (*bottom*). The symbol ● marks the value of s chosen by Algorithm 5.2; ■ marks the value given by the algorithm of Al-Mohy and Higham (2009*b*).

While $\|A\|_2$ is of order 10^4, the successive powers of A are smaller than might be expected, or, in other words, the inequalities $\|A^i\|_2 \le \|A\|_2^i$ are very weak. Since the derivation of Algorithm 5.2 is based on bounding a power series by making use of such inequalities, it is not surprising that the algorithm can make a conservative choice of s. To do better, it is necessary to take account of the behaviour of the powers of A.

Notice that the matrix A in (5.5) is block 2×2 block upper triangular with both diagonal blocks having norm of order 1, while the off-diagonal block has norm of order 10^4. In this situation, Algorithm 5.2 is forced to take a large s in order to bring the overall matrix norm down to $\theta_{13} \approx 5.3$, even though this is not necessary as regards computing the exponentials of of the diagonal blocks. That it might not be necessary to let the off-diagonal blocks determine s can be seen from the formula (see, *e.g.*, Higham (2008, Problem 10.12), Van Loan (1978))

$$\exp\left(\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22}\end{bmatrix}\right) = \begin{bmatrix} \mathrm{e}^{A_{11}} & \displaystyle\int_0^1 \mathrm{e}^{A_{11}(1-s)} A_{12}\, \mathrm{e}^{A_{22}s}\, \mathrm{d}s \\ 0 & \mathrm{e}^{A_{22}} \end{bmatrix}. \tag{5.6}$$

Since A_{12} enters only *linearly* in the expression for e^A it is reasonable to argue that its norm should not influence s.

This phenomenon, whereby a much larger s is chosen than necessary in order to approximate e^A to the desired accuracy, was identified by Kenney and Laub (1998), and later by Dieci and Papini (2000), and is referred to as overscaling.

Al-Mohy and Higham (2009*b*) derive a new scaling and squaring algorithm that exploits the quantities $d_k = \|A^k\|^{1/k}$ for a few values of k. The key idea is to bound the truncation error for the Padé approximant using these values. Specifically, it can be shown that with h_{2m+1} defined as in Lemma 5.1 we have

$$\|h_{2m+1}(A)\| \le \widetilde{h}_{2m+1}\bigl(\max(d_p, d_{p+1})\bigr) \quad \text{if } 2m+1 \ge p(p-1),$$

where $\widetilde{h}_{2m+1}(x) = \sum_{k=2m+1}^{\infty} |c_k| x^k$. The algorithm re-uses the θ_m values in Table 5.1, but the scaling is now chosen so that $\max(d_k, d_{k+1})$, rather than $\|A\|$, is bounded by θ_m (for some appropriate k and m). In addition to computing d_k for values of k for which A^k must in any case be formed, the algorithm also estimates d_k for a few additional k using the matrix norm estimator of Higham and Tisseur (2000). The algorithm of Al-Mohy and Higham (2009*b*) also incorporates an improvement for the triangular case; the need for exponentials of triangular matrices arises, for example, in the solution of radioactive decay equations (Morai and Pacheco 2003, Yuan and Kernan 2007). In the squaring phase the diagonal and first superdiagonal of $X_i = (r_m(T))^{2^i}$ (where T is the scaled triangular matrix) are computed from exact formulae for the corresponding elements of $\mathrm{e}^{2^i T}$, which both gives a more accurate diagonal and first superdiagonal and reduces the propagation of errors in the squaring recurrence. The new algorithm generally provides accuracy at least as good as Algorithm 5.2 at no higher cost, and for matrices that are triangular or cause overscaling it usually yields significant improvements in accuracy, cost, or both. Figure 5.1 shows that for the matrix (5.5) the new algorithm chooses an almost optimal value of s, yielding a much more accurate solution than Algorithm 5.2.

5.2. Matrix logarithm

For the matrix logarithm we have the expansion

$$\log(I+X) = X - \frac{X^2}{2} + \frac{X^3}{3} - \frac{X^4}{4} + \cdots, \qquad \rho(X) < 1. \tag{5.7}$$

In order to use Framework 5.1 we need to choose the initial transformation g to bring A close to the identity. This is achieved by repeatedly taking square roots until $A^{1/2^k}$ is sufficiently close to I. By (2.8), the square roots are undone by scalar multiplication, so the overall approximation is $2^k r(A^{1/2^k} - I)$

for some suitable rational approximation $r(x)$ to $\log(1+x)$. This approximation was originally used by Briggs for computing logarithms of scalars (Goldstine 1977, Phillips 2000). For matrices, the approximation leads to the inverse scaling and squaring method, which was proposed by Kenney and Laub (1989*a*).

As for the exponential, the preferred approximations are diagonal Padé approximants. The diagonal Padé approximants r_m to $\log(1+x)$ are known explicitly in two different forms. They have the continued fraction expansion

$$r_m(x) = \cfrac{c_1 x}{1 + \cfrac{c_2 x}{1 + \cfrac{c_3 x}{1 + \cdots + \cfrac{c_{2m-2} x}{1 + \cfrac{c_{2m-1} x}{1 + c_{2m} x}}}}}, \tag{5.8a}$$

$$c_1 = 1, \quad c_{2j} = \frac{j}{2(2j-1)}, \quad c_{2j+1} = \frac{j}{2(2j+1)}, \quad j = 1, 2, \ldots, \tag{5.8b}$$

which is the truncation of an infinite continued fraction for $\log(1+x)$. They can also be represented in linear partial fraction form

$$r_m(x) = \sum_{j=1}^{m} \frac{\alpha_j^{(m)} x}{1 + \beta_j^{(m)} x}, \tag{5.9}$$

where the $\alpha_j^{(m)}$ are the weights and the $\beta_j^{(m)}$ are the nodes of the m-point Gauss–Legendre quadrature rule on $[0,1]$, all of which are real.

Analogously to the scaling and squaring method for the exponential, early algorithms used a fixed Padé degree and a fixed condition $\|A^{1/2^k} - I\| \le \theta$ for determining how many square roots to take. Kenney and Laub (1989*a*) take $m = 8$ and $\theta = 0.25$, while Dieci, Morini and Papini (1996) take $m = 9$ and $\theta = 0.35$. The use of a degree m dependent on $\|A\|$ originates with Cheng, Higham, Kenney and Laub (2001), who exploit the following result of Kenney and Laub (1989*b*).

Theorem 5.3. For $\|X\| < 1$ and any subordinate matrix norm,

$$\|r_m(X) - \log(I+X)\| \le \bigl|r_m(-\|X\|) - \log(1-\|X\|)\bigr|. \tag{5.10}$$

This result says that the error in the matrix Padé approximation is bounded by the error in the scalar approximation at minus the norm of the matrix. After each square root is taken, this bound can be used to check whether the approximation error is guaranteed to be sufficiently small for a particular m. In designing an algorithm it is necessary to consider whether, when (5.10) is satisfied for some allowable m, it is worth taking another square root in the hope that the extra cost will be outweighed by being able to take a smaller m. Guidance in answering this question comes from the approximation, valid for large enough k,

$$\|I - A^{1/2^{k+1}}\| \approx \frac{1}{2}\|I - A^{1/2^k}\|, \tag{5.11}$$

which follows from $\bigl(I - A^{1/2^{k+1}}\bigr)\bigl(I + A^{1/2^{k+1}}\bigr) = I - A^{1/2^k}$.

Unlike for the exponential, the details of an inverse scaling and squaring algorithm depend on whether the matrix A is full or triangular, because the appropriate method for computing the square roots (and hence the cost of the square root stage) depends on the structure. If A is triangular, or an initial Schur factorization is computed to reduce to the triangular case, then the Schur method (Algorithm 4.6) can be used, otherwise a variant of the Newton iteration is appropriate; see Section 6.1. For evaluating the Padé approximant the best compromise between speed and accuracy turns out to be the partial fraction form (Higham 2001).

Cheng *et al.* (2001) give an inverse scaling and squaring algorithm based on the above ideas that accepts as input a parameter specifying the desired accuracy and computes square roots using the product form of the Denman–Beavers iteration (6.9), with the number of iterations used to compute each square root carefully chosen to minimize the overall cost. Higham (2008, Section 11.5) gives algorithms for both the full and triangular cases in which the parameters used in the algorithm's logic are precomputed, analogously as for Algorithm 5.2, rather than computed at run-time as in Cheng *et al.* (2001).

5.3. Trigonometric functions

We can apply Framework 5.1 to trigonometric functions by scaling $A \leftarrow 2^{-s}A$ in step (1) and using the appropriate double angle formulas in step (3). This idea was first proposed by Serbin and Blalock (1980) for the matrix cosine, using $\cos(2A) = 2\cos(A)^2 - I$.

It is not known whether Padé approximants r_{km} to the matrix cosine and sine exist for all degrees k and m, though they can be determined symbolically for the range of degrees of practical interest. Higham and Smith (2003) develop an algorithm that chooses s so that $\|2^{-s}A\|_\infty \le 1$, approximates $\cos(A) \approx r_8(2^{-s}A)$, and then uses the double angle recurrence. Hargreaves and Higham (2005) extend this approach by choosing the degree of the Padé approximant and the amount of scaling based on truncation error bounds expressed in terms of $\|A^2\|^{1/2}$ instead of $\|A\|$, thus using a more rudimentary version of the approach used for the exponential by Al-Mohy and Higham (2009*b*). They also derive an algorithm for computing both $\cos(A)$ and $\sin(A)$, by adapting the ideas developed for the cosine and intertwining the cosine and sine double angle recurrences.

So far there is little work on algorithms for other trigonometric functions. Some ideas concerning the tangent and inverse tangent for Hermitian matrices are given by Cheng, Higham, Kenney and Laub (2000).

6. Matrix iterations

Matrix roots, the matrix sign function, and the unitary polar factor (which we will not consider in this paper) are all amenable to computation by matrix iterations of the form

$$X_{k+1} = g(X_k), \tag{6.1}$$

where g is some nonlinear, usually rational, function. The starting matrix X_0 is almost always A in practice, and indeed in some cases A does not appear in the iteration itself, that is, g is independent of A. Such iterations are attractive because they are easy to implement, requiring just the basic building blocks of matrix multiplication and the solution of multiple right-hand side linear systems. It might appear that convergence analysis is straightforward, reducing to the scalar case that is usually well understood, but this is not necessarily so. Moreover, the numerical stability of matrix iterations in finite precision arithmetic is a subtle issue, with small changes in the form of the iteration sometimes greatly changing the stability.

6.1. Matrix sign function and square root

Two fundamental iterations are the Newton iteration for the matrix sign function,

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \qquad X_0 = A, \tag{6.2}$$

and the Newton iteration for the matrix square root,

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A), \qquad X_0 = A. \tag{6.3}$$

The term 'Newton iteration' needs explaining. Iteration (6.2) is precisely Newton's method applied to the equation $X^2 = I$, assuming that the iterates are uniquely defined (Higham 2008, Problem 5.8). Likewise, iteration (6.3) is Newton's method applied to $X^2 = A$ provided the iterates are uniquely defined, and the proof (Higham 2008, Lemma 6.8) relies on the fact that $X_kA = AX_k$ for all k. Note, however, that if we select an arbitrary X_0 in (6.3) then the iteration is in general no longer equivalent to Newton's method, because the iterates X_k will not now commute with A.

For $A \in \mathbb{C}^{n\times n}$ having no pure imaginary eigenvalues, the iterates (6.2) converge quadratically to $\operatorname{sign}(A)$. For $A \in \mathbb{C}^{n\times n}$ with no eigenvalues on $\mathbb{R}^-$, the X_k from (6.3) converge quadratically to the principal square root, $A^{1/2}$. These results can be proved at the matrix level without using a

Table 6.1. Iteration functions $f_{\ell m}$ from the Padé family (6.4).

	$m = 0$	$m = 1$	$m = 2$
$\ell = 0$	x	$\dfrac{2x}{1+x^2}$	$\dfrac{8x}{3+6x^2-x^4}$
$\ell = 1$	$\dfrac{x}{2}(3-x^2)$	$\dfrac{x(3+x^2)}{1+3x^2}$	$\dfrac{4x(1+x^2)}{1+6x^2+x^4}$
$\ell = 2$	$\dfrac{x}{8}(15-10x^2+3x^4)$	$\dfrac{x}{4}\dfrac{(15+10x^2-x^4)}{1+5x^2}$	$\dfrac{x(5+10x^2+x^4)}{1+10x^2+5x^4}$

transformation to Jordan form to introduce the scalar case. For the sign iteration the convergence reduces to the fact that $G^k \to 0$ as $k \to \infty$ if the spectral radius $\rho(G) < 1$. The convergence of the square root iteration can be shown to be equivalent to that of the sign iteration for $X_0 = A^{1/2}$. See Higham (2008, Theorems 5.6, 6.9) for details.

The Newton iteration for sign(A), originally proposed by Roberts (1980), is the inverse of a member of an infinite Padé family of iterations that have some remarkable properties. The (ℓ, m) iteration is

$$X_{k+1} = X_k\, p_{\ell m}(I - X_k^2)\, q_{\ell m}(I - X_k^2)^{-1} =: f_{\ell m}(X_k), \qquad X_0 = A, \quad (6.4)$$

where $r_{\ell m}(\xi) = p_{\ell m}(\xi)/q_{\ell m}(\xi)$ is the $[\ell/m]$ Padé approximant to $h(\xi) = (1-\xi)^{-1/2}$. The appearance of h arises from the relation $\operatorname{sign}(z) = z/(z^2)^{1/2} = z/(1-(1-z^2))^{1/2} = zh(1-z^2)$.

Table 6.1 shows the first nine iteration functions $f_{\ell m}$ from this family. For $\ell = m$ and $\ell = m-1$, the polynomials $xp_{\ell m}(1-x^2)$ and $q_{\ell m}(1-x^2)$ are, respectively, the odd and even parts of $(1+x)^{\ell+m+1}$ (Kenney and Laub 1991*b*), which provides an easy way to generate the iteration functions. Note that f_{01} gives the iteration

$$X_{k+1} = 2X_k(I + X_k^2)^{-1}, \qquad X_0 = A, \quad (6.5)$$

which generates matrices that are the inverses of the those from (6.2), while f_{10} gives the Newton–Schulz iteration

$$X_{k+1} = \frac{1}{2}X_k(3I - X_k^2), \qquad X_0 = A. \quad (6.6)$$

This latter iteration can be derived from the Newton iteration (6.2) by approximating the inverse therein by one step of the Newton–Schulz iteration for the matrix inverse with X_k as starting value.

The following result of Kenney and Laub (1991*b*) describes the convergence properties.

Theorem 6.1. (convergence of Padé iterations) Let $A \in \mathbb{C}^{n\times n}$ have no pure imaginary eigenvalues and let $S = \operatorname{sign}(A)$. Consider the iteration (6.4) with $\ell + m > 0$ and any subordinate matrix norm.

(a) For $\ell \geq m-1$, if $\|I - A^2\| < 1$ then $X_k \to S$ as $k \to \infty$ and $\|I - X_k^2\| < \|I - A^2\|^{(\ell+m+1)^k}$.

(b) For $\ell = m-1$ and $\ell = m$,

$$(S - X_k)(S + X_k)^{-1} = \left[(S - A)(S + A)^{-1}\right]^{(\ell+m+1)^k}$$

and hence $X_k \to S$ as $k \to \infty$.

Theorem 6.1 shows that the iterations with $\ell = m-1$ and $\ell = m$ are globally convergent (that is, convergent for any A), while those with $\ell \geq m+1$ have local convergence, the convergence rate being $\ell + m + 1$ in every case.

Other interesting properties of the Padé family can be found in Higham (2008, Theorem 5.9).

The Padé iterations for the sign function have analogues for the square root that can be derived using the relation (Higham 1997)

$$\operatorname{sign}\left(\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix}. \tag{6.7}$$

By applying any sign iteration to the matrix $\left[\begin{smallmatrix} 0 & A \\ I & 0 \end{smallmatrix}\right]$, using (6.7), and then reading off the (1,2) and (2,1) blocks, a coupled iteration for the square root is obtained. For example, if we start with the Newton iteration (6.2), we obtain

$$\begin{aligned} X_{k+1} &= \frac{1}{2}\left(X_k + Y_k^{-1}\right), & X_0 &= A, \\ Y_{k+1} &= \frac{1}{2}\left(Y_k + X_k^{-1}\right), & Y_0 &= I. \end{aligned} \tag{6.8}$$

This iteration was originally derived by Denman and Beavers (1976). It is easy to show that $Y_k \equiv A^{-1}X_k$, and so X_k satisfies (6.3). A general result of Higham, Mackey, Mackey and Tisseur (2005, Theorem 4.5) shows that for essentially any sign iteration this approach produces a coupled iteration with matrices X_k converging to $A^{1/2}$ and Y_k converging to $A^{-1/2}$, both with the same order of convergence as the original sign iteration. Thus the convergence is quadratic for (6.8). A variant of (6.8) that trades a matrix inverse for a multiplication is obtained by setting $M_k = X_k Y_k$ in (6.8):

$$\begin{aligned} X_{k+1} &= \frac{1}{2}X_k\left(I + M_k^{-1}\right), & X_0 &= A, \\ M_{k+1} &= \frac{1}{2}\left(I + \frac{M_k + M_k^{-1}}{2}\right), & M_0 &= A. \end{aligned} \tag{6.9}$$

At first sight, it may be unclear why (6.8) or (6.9) might be preferred to the Newton iteration (6.3), since they require $4n^3$ flops per iteration versus $8n^3/3$ flops for (6.3). The answer is that the stability properties of the iterations are very different.

In general, stability can be defined as follows for a matrix iteration (6.1).

Definition 6.2. (stability) Consider an iteration $X_{k+1} = g(X_k)$ with a fixed point X. Assume that g is Fréchet-differentiable[3] at X. The iteration is *stable in a neighbourhood of* X if the Fréchet derivative $L_g(X)$ has bounded powers, that is, there exists a constant c such that $\|L_g^i(X)\| \le c$ for all $i > 0$.

Note that stability concerns behaviour close to convergence and so is an asymptotic property. The motivation for the definition is that L_g determines how a perturbation E_k in X_k is propagated through the iteration, and this perturbation will have a bounded effect near X if the iteration is stable. Although it is an asymptotic property, this notion proves to be a good predictor of the overall numerical stability of a matrix iteration. It is sometimes the case that L_g is idempotent, that is, $L_g^2(X,E) = L_g(X, L_g(X,E)) = L_g(X,E)$, in which case stability is immediate. This notion of stability was introduced by Cheng *et al.* (2001) and developed further by Higham (2008, Section 4.9.4).

For the Newton iteration (6.3) we have $L_g(X,E) = (E - X^{-1}EX^{-1}A)/2$, and, at the fixed point, $L_g(A^{1/2},E) = (E - A^{-1/2}EA^{1/2})/2$. The eigenvalues of $L_g(A^{1/2})$ can be shown to be

$$\mu_{ij} = \frac{1}{2}(1 - \lambda_i^{1/2}\lambda_j^{-1/2}), \qquad i,j = 1\colon n,$$

where the λ_i are the eigenvalues of A, and the μ_{ij} will be within the unit circle (which is necessary for stability) only for very well-behaved matrices A. Thus in general the iteration is unstable, and this is easily demonstrated experimentally. The instability was first identified by Laasonen (1958). Higham (1986*b*) explained the instability and derived the stability condition $|\mu_{ij}| < 1$.

It is perhaps surprising that the process of rewriting the Newton iteration in the coupled form given by the Denman–Beavers iteration (6.8) stabilizes it. The iteration function is now

$$G(X,Y) = \frac{1}{2}\begin{bmatrix} X + Y^{-1} \\ Y + X^{-1} \end{bmatrix}.$$

[3] See Section 7 for the definition of Fréchet derivative.

At the limit $X = A^{1/2}$, $Y = A^{-1/2}$ we have

$$L_g(A^{1/2}, A^{-1/2}; E, F) = \frac{1}{2}\begin{bmatrix} E - A^{1/2}FA^{1/2} \\ F - A^{-1/2}EA^{-1/2} \end{bmatrix}, \tag{6.10}$$

and it is easy to see that $L_g(A^{1/2}, A^{-1/2})$ is idempotent. Hence the Denman–Beavers iteration is stable. The Denman–Beavers iteration is just one of several ways of rewriting the Newton iteration as a coupled iteration, all of which are stable; see Higham (2008, Chapter 6) for more details.

The iterations described above are all of limited use in their basic forms. If we take $A = \theta \in \mathbb{R}$ for some $\theta \gg 1$, then the sign iteration (6.2) approximately effects $x \leftarrow x/2$ in the early stages, so many steps are needed before the asymptotic quadratic convergence comes into effect. In practice the iterations are used in conjunction with scaling. Usually, scaling consists of replacing X_k by $\mu_k X_k$ for some $\mu_k > 0$ in the formula for X_{k+1} (with a corresponding change for the other iterate in a coupled iteration), though more general scalings with more than one parameter can also be considered. Standard scalings for the Newton sign iteration are:

$$\text{determinantal scaling} \quad \mu_k = |\det(X_k)|^{-1/n}, \tag{6.11}$$

$$\text{spectral scaling} \quad \mu_k = \sqrt{\rho(X_k^{-1})/\rho(X_k)}, \tag{6.12}$$

$$\text{norm scaling} \quad \mu_k = \sqrt{\|X_k^{-1}\|/\|X_k\|}. \tag{6.13}$$

Each of these is in some way trying to bring X_k closer to $\operatorname{sign}(X_k) = \operatorname{sign}(A)$. Experiments show that there is no clear best scaling, but spectral scaling does have a finite termination property when A has only real eigenvalues (Barraud 1979, Section 4, Higham 2008, Theorem 5.11). Corresponding scalings for the square root iterations can be derived by using the connections with the sign function in Higham (2008, Theorem 6.9) and (6.7).

A number of linearly convergent iterations have been investigated in the literature, mostly for the square root and usually with structured matrices in mind. We mention three of them. First, with $A \equiv I - C$, is the binomial iteration,

$$X_{k+1} = \frac{1}{2}(C + X_k^2), \qquad X_0 = 0, \tag{6.14}$$

so called because it is essentially a convenient way of evaluating the binomial expansion $(I - C)^{1/2} = \sum_{j=0}^{\infty} \binom{\frac{1}{2}}{j}(-C)^j \equiv I - P$. The iterates X_k converge to $I - P$ if the eigenvalues of C lie in the main cardioid of the Mandelbrot set stretched by a factor 4 (Higham 2008, Theorem 6.14). If C has non-negative elements ($C \geq 0$) and spectral radius less than 1 then the convergence is monotonic from below in the elementwise ordering. An important class of matrices satisfying the latter condition after scaling is the non-singular

M-matrices, which are the non-singular $A \in \mathbb{R}^{n\times n}$ such that $A = sI - B$, where $B \geq 0$ and $s > \rho(B)$. If we write $A = s(I - C)$ with $C = s^{-1}B \geq 0$ then $\rho(C) < 1$ and so the binomial iteration converges monotonically when applied to $I - C$.

Next, the Pulay iteration (Pulay 1966) writes $A^{1/2} = D^{1/2} + B$ with D diagonal and positive definite ($D = \mathrm{diag}(A)$ being the natural choice if it is positive definite) and computes B as the limit of the B_k from the iteration

$$D^{1/2}B_{k+1} + B_{k+1}D^{1/2} = A - D - B_k^2, \qquad B_0 = 0. \tag{6.15}$$

Finally, the Visser iteration (Visser 1937, Elsner 1970) has the form

$$X_{k+1} = X_k + \alpha(A - X_k^2), \qquad X_0 = (2\alpha)^{-1}I. \tag{6.16}$$

Both iterations are forms of modified Newton iterations with the Fréchet derivative approximated by a constant. A sufficient condition for the convergence of the Pulay iteration is that $A^{1/2} - D^{1/2}$ is small compared with $D^{1/2}$ (Higham 2008, Theorem 6.15). The Visser iteration is related to the binomial iteration; convergence to $A^{1/2}$ is guaranteed if the eigenvalues of $I - 4\alpha^2 A$ lie in the cardioid referred to above.

For Hermitian positive definite matrices a very good way to compute the principal square root without computing the spectral decomposition is as follows (Higham 1986*a*).

Algorithm 6.3. Given a Hermitian positive definite matrix $A \in \mathbb{C}^{n\times n}$ this algorithm computes $H = A^{1/2}$.

```
1  $A = R^*R$ (Cholesky factorization).
2  Compute the Hermitian polar factor $H$ of $R$ by applying
   (Higham 2008, Alg. 8.20) to $R$ (exploiting the triangularity of $R$).
```

Cost: Up to about $15\frac{2}{3}n^3$ flops.

The algorithm used in step 2 of Algorithm 6.3 employs a (scaled) Newton iteration for the unitary polar factor that is closely related to the Newton iteration (6.2) for the matrix sign function.

6.2. Matrix pth root

The Newton iteration for the principal pth root of A analogous to (6.3) is

$$X_{k+1} = \frac{1}{p}\big[(p-1)X_k + X_k^{1-p}A\big], \qquad X_0 = I. \tag{6.17}$$

As in the $p = 2$ case, this iteration is unstable except for A with spectrum clustered around 1. The convergence properties for $p > 2$, however, are much more complicated than for $p = 2$; for convergence, the eigenvalues of A must lie in regions of the complex plane having a fractal structure. Indeed, these regions are convergence regions for the map $x_{k+1} = [(p-1)x_k + x_k^{1-p}a]/p$, which is a classic example for the analysis of rational iterations via the theory of Julia sets (Peitgen, Jürgens and Saupe 1992, Schroeder 1991).

Iannazzo (2006) identifies a set $\{ z \in \mathbb{C} : \operatorname{Re} z > 0 \text{ and } |z| \le 1 \} \cup \mathbb{R}^+$ such that if the eigenvalues of A lie in the set then the iteration (6.17) converges quadratically to $A^{1/p}$. He suggests an algorithm that uses an initial square root followed by a scaling such that a coupled, stable form of iteration (6.17) converges for the preprocessed matrix. Instead of applying Newton's method to $X^p = A$, which gives (6.17), we can apply it to $X^{-p} = A$, which leads to the inverse Newton iteration

$$X_{k+1} = \frac{1}{p}\big[(p+1)X_k - X_k^{p+1}A\big], \qquad X_0 = c^{-1}I. \tag{6.18}$$

Here, $c > 0$ is a parameter, and Guo and Higham (2006) show that the iteration converges quadratically to $A^{-1/p}$ if all the eigenvalues of A are in the set

$$\operatorname{conv}\big\{ \{ z : |z - c^p| \le c^p \}, (p+1)c^p \big\} \setminus \{ 0, (p+1)c^p \}, \tag{6.19}$$

where conv is the convex hull. One useful practical conclusion that can be drawn is that if A is stochastic and strictly row-diagonally dominant then iteration (6.18) with $c = 1$ converges; this is relevant for the application described in Section 3.2. It is necessary to rewrite (6.18) in a coupled form for stability:

$$\begin{aligned} X_{k+1} &= X_k\left(\frac{(p+1)I - M_k}{p}\right), & X_0 &= \frac{1}{c}I, \\ M_{k+1} &= \left(\frac{(p+1)I - M_k}{p}\right)^p M_k, & M_0 &= \frac{1}{c^p}A; \end{aligned} \tag{6.20}$$

we have $X_k \to A^{-1/p}$ and $M_k \to I$. Guo and Higham combine this iteration with an initial Schur reduction to triangular form followed by the computation of a sufficient number of square roots, computed using Algorithm 4.6, so that fast convergence is expected for (6.20), if $A^{-1/p}$ is required, or a variant of (6.20) given by Guo and Higham, if $A^{1/p}$ is required.

Guo (2009, Theorem 5) obtains a more useful convergence result for (6.17) than Iannazzo's by proving convergence when the eigenvalues of A lie in the set $\{ z \in \mathbb{C} : |z - 1| \le 1 \}$, and he gives an analogue of Guo and Higham's algorithm based on the coupled version of (6.17). Guo (2009, Theorem 13) also shows that if $A = I - B$ with $\rho(B) < 1$ then X_k from (6.17) satisfies $X_k = \sum_{i=0}^{\infty} c_i^{(k)} B^i$, where $c_i^{(k)} = b_i$, $i = 0\colon 2^k - 1$ and $(1-x)^{1/p} = \sum_{i=0}^{\infty} b_i x^i$. Thus k steps of the Newton iteration reproduce 2^k terms of the binomial series. He also obtains an analogous result for (6.18).

A Padé family of iterations for the pth root is investigated by Laszkiewicz and Ziętak (2009). A variety of other methods exist for computing pth roots; see Bini, Higham and Meini (2005).

7. Fréchet derivative

The Fréchet derivative of a matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ at $A \in \mathbb{C}^{n\times n}$ is a linear mapping

$$\begin{tikzcd} \mathbb{C}^{n\times n} \arrow[r, "L_f(A)"] & \mathbb{C}^{n\times n} \\ E \arrow[r, mapsto] & L_f(A,E) \end{tikzcd}$$

such that

$$f(A+E) - f(A) - L_f(A,E) = o(\|E\|) \qquad (7.1)$$

for all $E \in \mathbb{C}^{n\times n}$. Thus it describes the first-order effect on f of perturbations in A. We note that a sufficient condition for the Fréchet derivative to be defined is that f is $2n-1$ times continuously differentiable on an open subset of $\mathbb{R}$ or $\mathbb{C}$ containing the spectrum of A (Higham 2008, Theorem 3.8).

7.1. Condition number

The norm of the Fréchet derivative,

$$\|L_f(A)\| := \max_{\|Z\|=1} \|L_f(A,Z)\|, \qquad (7.2)$$

appears in an explicit expression for the condition number of the matrix function f at A (Higham 2008, Theorem 3.1):

$$\operatorname{cond}(f,A) := \lim_{\epsilon\to 0} \sup_{\|E\|\le\epsilon\|A\|} \frac{\|f(A+E)-f(A)\|}{\epsilon\|f(A)\|} = \frac{\|L_f(A)\|\,\|A\|}{\|f(A)\|}. \qquad (7.3)$$

Ideally, along with $f(A)$ we would like to produce an estimate of $\operatorname{cond}(f,A)$. This can be done by converting the problem to one of matrix norm estimation. Since L_f is a linear operator,

$$\operatorname{vec}(L_f(A,E)) = K(A)\operatorname{vec}(E) \qquad (7.4)$$

for some $K(A) \in \mathbb{C}^{n^2\times n^2}$ that is independent of E. We refer to $K(A)$ as the Kronecker form of the Fréchet derivative. To estimate $\|L_f(A)\|_F$ we can apply the power method to $K(A)$, since $\|L_f(A)\|_F = \|K(A)\|_2$. The resulting algorithm can be written entirely in terms of $L_f(A)$ and $L_f^\star(A)$, the adjoint of $L_f(A)$ defined with respect to the inner product $\langle X,Y\rangle = \operatorname{trace}(Y^*X)$. When $A \in \mathbb{R}^{n\times n}$ and $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, the adjoint is given by $L_f^\star(A) = L_f(A^T)$. In the complex case, $L_f^\star(A) = L_{\overline{f}}(A^*)$, where $\overline{f}(z) := \overline{f(\overline{z})}$, so that if f has a power series representation then $\overline{f}$ is obtained by conjugating the coefficients.

Algorithm 7.1. (power method on Fréchet derivative) Given $A \in \mathbb{C}^{n\times n}$ and the Fréchet derivative L_f of a function f, this algorithm uses the power method to produce an estimate $\gamma \le \|L_f(A)\|_F$.

```
1 Choose a non-zero starting matrix $Z_0 \in \mathbb{C}^{n\times n}$
2 for $k = 0\colon \infty$
3     $W_{k+1} = L_f(A, Z_k)$
4     $Z_{k+1} = L_f^\star(A, W_{k+1})$
5     $\gamma_{k+1} = \|Z_{k+1}\|_F/\|W_{k+1}\|_F$
6     if converged, $\gamma = \gamma_{k+1}$, quit, end
7 end
```

A random Z_0 is a reasonable choice. However, our preference is to apply instead of the power method the block 1-norm estimator of Higham and Tisseur (2000), available as `normest1` in MATLAB. This produces a quantity γ with $\gamma \le \|K(A)\|_1$, where $\|K(A)\|_1 \in \bigl[n^{-1}\|L_f(A)\|_1, n\|L_f(A)\|_1\bigr]$.

Both the above approaches require the ability to evaluate $L_f(A,E)$ and $L_f^\star(A,E)$. In the rest of this section we discuss several general approaches.

7.2. Power series

When f has a power series expansion the Fréchet derivative can be expressed as a related series expansion (Higham 2008, Problem 3.6)

Theorem 7.2. Let f have the power series expansion $f(x) = \sum_{k=0}^\infty a_k x^k$ with radius of convergence r. Then, for $A, E \in \mathbb{C}^{n\times n}$ with $\|A\| < r$,

$$L_f(A,E) = \sum_{k=1}^\infty a_k \sum_{j=1}^k A^{j-1} E A^{k-j}. \tag{7.5}$$

The next theorem, from Al-Mohy and Higham (2009*a*), gives a recurrence that can be used to evaluate (7.5), as well as a useful bound on $\|L_f(A)\|$.

Theorem 7.3. Under the assumptions of Theorem 7.2,

$$L_f(A,E) = \sum_{k=1}^\infty a_k M_k, \tag{7.6}$$

where $M_k = L_{x^k}(A,E)$ satisfies the recurrence

$$M_k = M_{\ell_1} A^{\ell_2} + A^{\ell_1} M_{\ell_2}, \qquad M_1 = E, \tag{7.7}$$

with $k = \ell_1 + \ell_2$ and ℓ_1 and ℓ_2 positive integers. In particular,

$$M_k = M_{k-1} A + A^{k-1} M_1, \qquad M_1 = E. \tag{7.8}$$

In addition,

$$\|f(A)\| \le \widetilde{f}(\|A\|), \qquad \|L_f(A)\| \le \widetilde{f}'(\|A\|), \tag{7.9}$$

where $\widetilde{f}(x) = \sum_{k=0}^\infty |a_k| x^k$.

Proof. Since the power series can be differentiated term-by-term within its radius of convergence, we have

$$L_f(A,E) = \sum_{k=1}^{\infty} a_k M_k, \qquad M_k = L_{x^k}(A,E).$$

The product rule for Fréchet derivatives (Higham 2008, Theorem 3.3) yields

$$M_k = L_{x^k}(A,E) = L_{x^{\ell_1}}(A,E)A^{\ell_2} + A^{\ell_1}L_{x^{\ell_2}}(A,E) = M_{\ell_1}A^{\ell_2} + A^{\ell_1}M_{\ell_2}.$$

Taking $\ell_1 = k-1$ and $\ell_2 = 1$ gives (7.8). It is straightforward to see that $\|f(A)\| \le \widetilde{f}(\|A\|)$. Taking norms in (7.5) gives

$$\|L_f(A,E)\| \le \|E\| \sum_{k=1}^{\infty} k|a_k|\|A\|^{k-1} = \|E\|\widetilde{f}'(\|A\|),$$

and maximizing over all non-zero E gives $\|L_f(A)\| \le \widetilde{f}'(\|A\|)$. □

7.3. Block triangular matrix formula

If f is $2n-1$ times continuously differentiable on an open subset of $\mathbb{R}$ or $\mathbb{C}$ containing the spectrum of $A \in \mathbb{C}^{n\times n}$ then (Higham 2008, Section 3.2)

$$f\left(\begin{bmatrix} A & E \\ 0 & A \end{bmatrix}\right) = \begin{bmatrix} f(A) & L_f(A,E) \\ 0 & f(A) \end{bmatrix}. \tag{7.10}$$

Thus $L_f(A,E)$ can be obtained by evaluating f at the $2n \times 2n$ matrix $\left[\begin{smallmatrix} A & E \\ 0 & A \end{smallmatrix}\right]$ and reading off the (1,2) block. This approach is pragmatic, but for an $O(n^3)$ method its cost is up to 8 times the cost of evaluating $f(A)$ alone, this multiplier being mitigated by the block triangular, block Toeplitz structure of the argument. A drawback noted by Al-Mohy and Higham (2009*a*) is that since $L_f(A,\alpha E) = \alpha L_f(A,E)$ the norm of E can be chosen at will, but the choice may affect the accuracy of the algorithm used to evaluate (7.10), and it is difficult to know what is the optimal choice.

We illustrate the use of (7.10) for the matrix square root by applying the Denman–Beavers iteration (6.8) to $\widetilde{A} = \left[\begin{smallmatrix} A & E \\ 0 & A \end{smallmatrix}\right]$. Iterates $\widetilde{X}_k$ and $\widetilde{Y}_k$ are produced for which

$$\widetilde{X}_k = \begin{bmatrix} X_k & F_k \\ 0 & X_k \end{bmatrix}, \qquad \widetilde{Y}_k = \begin{bmatrix} Y_k & G_k \\ 0 & Y_k \end{bmatrix},$$

where X_k and Y_k satisfy (6.8) and

$$\begin{aligned} F_{k+1} &= \frac{1}{2}\big(F_k - Y_k^{-1}G_kY_k^{-1}\big), \qquad & F_0 &= E, \\ G_{k+1} &= \frac{1}{2}\big(G_k - X_k^{-1}F_kX_k^{-1}\big), \qquad & G_0 &= 0. \end{aligned} \tag{7.11}$$

From (7.10) we conclude that

$$\lim_{k\to\infty} F_k = L_{x^{1/2}}(A,E), \qquad \lim_{k\to\infty} G_k = L_{x^{-1/2}}(A,E). \tag{7.12}$$

The iteration (7.11) is due to Al-Mohy and Higham (2009*a*), who derive it in this way.

7.4. Differentiating an algorithm

If we have an algorithm for computing $f(A)$ then we might expect that differentiating it will provide an algorithm for computing the Fréchet derivative. We describe two situations where this idea proves useful.

Framework 5.1 uses a rational approximation $f(A) \approx r(A)$. Obviously, we can approximate $L_f(A,E)$ by $L_r(A,E)$, where the accuracy of this approximation remains to be investigated. By Fréchet-differentiating Framework 5.1 we obtain an algorithm for simultaneously computing $f(A)$ and $L_f(A,E)$.

Framework 7.1. Framework for approximating $f(A)$ and $L_f(A,E)$.

(1) Choose a suitable rational approximation r and a transformation function g and set $A \leftarrow g(A)$.
(2) Transform $E \leftarrow L_g(A,E)$ (since $L_{f\circ g} = L_f(g(A), L_g(A,E))$ by the chain rule for Fréchet derivatives (Higham 2008, Theorem 3.4)).
(3) Compute $r(A)$ and $L_r(A,E)$ by some appropriate scheme.
(4) Apply transformations to $r(A)$ and $L_r(A,E)$ that undo the effect of the initial transformation on A.

The natural way to obtain $L_r(A,E)$ at step (3) is by differentiating the scheme for r, which can be done with the aid of the following lemma from Al-Mohy and Higham (2009*a*) if the numerator and denominator polynomials are explicitly computed.

Lemma 7.4. The Fréchet derivative L_{r_m} of the rational function $r_m(x) = p_m(x)/q_m(x)$ satisfies

$$q_m(A)L_{r_m}(A,E) = L_{p_m}(A,E) - L_{q_m}(A,E)r_m(A). \tag{7.13}$$

Proof. Applying the Fréchet derivative product rule (Higham 2008, Theorem 3.3) to $q_m r_m = p_m$ gives

$$L_{p_m}(A,E) = L_{q_m r_m}(A,E) = L_{q_m}(A,E)r_m(A) + q_m(A)L_{r_m}(A,E),$$

which rearranges to the result. □

It can be shown (Al-Mohy and Higham 2009*a*, Theorem 4.1) that, for polynomials p and a class of schemes for evaluating $p(A)$ that contains all schemes of practical interest, the cost of evaluating $p(A)$ and $L_p(A,E)$ together is at most three times the cost of evaluating $p(A)$ alone.

Framework 7.1 has been used in conjunction with Algorithm 5.2 by Al-Mohy and Higham (2009*a*) to develop a scaling and squaring algorithm that computes e^A and $L_{\exp}(A, E)$ at about three times the cost of computing e^A alone. It improves on an earlier 'Kronecker–Sylvester scaling and squaring algorithm' of Kenney and Laub (1998) that is significantly more expensive and uses complex arithmetic even when A is real.

The second use of differentiation is to differentiate a matrix iteration. For example, we can differentiate the Newton iteration (6.2) for the matrix sign function to obtain

$$Y_{k+1} = \frac{1}{2}(Y_k - X_k^{-1} Y_k X_k^{-1}), \qquad Y_0 = E, \tag{7.14}$$

where X_k is defined by (6.2). The following result of Al-Mohy and Higham (2009*a*) shows that under reasonable assumptions this procedure will always produce an iteration having the required derivative as a fixed point.

Theorem 7.5. Let f and g be $2n-1$ times continuously differentiable on an open subset $\mathcal{D}$ of $\mathbb{R}$ or $\mathbb{C}$. Suppose that for any matrix $X \in \mathbb{C}^{n\times n}$ whose spectrum lies in $\mathcal{D}$, g has the fixed point $f(X)$, that is, $f(X) = g(f(X))$. Then, for any such X, L_g at $f(X)$ has the fixed point $L_f(X, E)$ for all E.

Theorem 7.5 does not readily lead to a convergence result. If we derive (7.14) by instead applying the Newton iteration to $\left[\begin{smallmatrix} A & E \\ 0 & A \end{smallmatrix}\right]$ (as was done by Mathias (1996)) then from (7.10) it is easy to see that $\lim_{k\to\infty} Y_k = L_{\text{sign}}(A, E)$. Iteration (7.14) is due to Kenney and Laub (1991*a*).

7.5. Finite differences

A natural approach is to approximate the Fréchet derivative by the finite difference

$$L_f(A, E) \approx \frac{f(A + hE) - f(A)}{h}, \tag{7.15}$$

for a suitably chosen h. Two types of errors affect this approximation: truncation errors caused by taking a finite h, and rounding errors in floating point arithmetic caused by subtracting two nearly equal matrices that are both contaminated by error. A standard argument based on balancing bounds for the two types of error leads to the choice (Higham 2008, Section 3.4)

$$h = \left(\frac{u\|f(A)\|}{\|E\|^2} \right)^{1/2}, \tag{7.16}$$

for which the overall error has a bound of order $u^{1/2}\|f(A)\|^{1/2}\|E\|$. The conclusion is that subtractive cancellation in floating point arithmetic limits the smallest relative error that can be obtained to order $u^{1/2}$.

7.6. Complex step approximation

Assume that $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ and $A, E \in \mathbb{R}^{n\times n}$. Replacing E by $\mathrm{i}hE$ in (7.1), where $\mathrm{i} = \sqrt{-1}$, and using the linearity of L_f, we obtain

$$f(A + \mathrm{i}hE) - f(A) - \mathrm{i}hL_f(A, E) = o(h).$$

Thus

$$f(A) \approx \operatorname{Re} f(A + \mathrm{i}hE), \tag{7.17}$$

$$L_f(A, E) \approx \operatorname{Im} \frac{f(A + \mathrm{i}hE)}{h}, \tag{7.18}$$

so that with one function evaluation at a complex argument we can approximate both f and $L_f(A, E)$. The approximation (7.18) is known as the complex step approximation; it has been known for some time in the scalar case (Squire and Trapp 1998, Giles, Duta, Müller and Pierce 2003, Martins, Sturdza and Alonso 2003, Shampine 2007), and was proposed for matrices by Al-Mohy and Higham (2010). In the latter paper it is shown that for analytic f the error in the approximations (7.17) and (7.18) is $O(h^2)$ and that the same is true for the matrix sign function.

An important advantage of the complex step approximation over the finite difference approximation (7.15) is that h is not restricted by floating point arithmetic considerations. Indeed practical experience reported in the papers cited above has demonstrated the ability of the approximation to produce accurate approximations in the scalar case even with h as small as 10^{-100}, which is the value used in software at the National Physical Laboratory according to Cox and Harris (2004). Al-Mohy and Higham (2010) show experimentally that when used in conjunction with condition estimation (see Section 7.1) the complex step approximation leads to significantly more reliable condition estimates than the finite difference approximation.

One caveat is that the underlying method for evaluating f must not employ complex arithmetic, which rules out methods based on the (complex) Schur form. The reason is that since the Fréchet derivative is assumed real, if the evaluation introduces a non-trivial imaginary part at any point then that term must be subject to massive subtractive cancellation in order for a final imaginary part of $O(h)$ to be produced. Looked at another way, the complex step approximation is essentially carrying out a form of automatic differentiation with h acting as a symbolic variable, and the introduction of pure imaginary numbers within the evaluation disturbs this process.

8. The $f(A)b$ problem

In many applications, especially those originating from partial differential equations and those with a large, sparse matrix A, it is the action of $f(A)$ on a vector, $f(A)b$, that is required and not $f(A)$. Of course, this is analogous to the requirement to solve a linear $Ax = b$ without computing A^{-1} (though the inverse function is distinguished from the other functions considered in this paper in that we rarely need to compute it explicitly). We discuss two general approaches.

8.1. Quadrature and rational approximation

If we have an integral representation $f(A) = \int r(A,t)\,\mathrm{d}t$, where r is a rational function of A, then we can apply quadrature and approximate $f(A)b$ by $\sum_i r(A,t_i)b$. Depending on how r is represented, the evaluation of this approximation reduces to solving one or more linear systems with coefficient matrices that are polynomials in A. This, of course, is equivalent to approximating $f(A)$ by the rational function $\sum_i r(A,t_i)$. Two notable examples of integral representations are (see, *e.g.*, Higham (2008))

$$\log(A) = \int_0^1 (A-I)\bigl[t(A-I)+I\bigr]^{-1}\,\mathrm{d}t, \tag{8.1}$$

$$\operatorname{sign}(A) = \frac{2}{\pi}A\int_0^\infty (t^2I+A^2)^{-1}\,\mathrm{d}t, \tag{8.2}$$

for which appropriate Gaussian quadrature rules lead to Padé approximants (for the details for (8.1), see Dieci *et al.* (1996, Theorem 4.3)).

More generally, for analytic functions f we can employ the Cauchy integral formula

$$f(A) = \frac{1}{2\pi \mathrm{i}}\int_\Gamma f(z)\,(zI-A)^{-1}\,\mathrm{d}z, \tag{8.3}$$

where Γ is a closed contour that lies in the region of analyticity of f and winds once around the spectrum in the anticlockwise direction. This formula is equivalent to the definitions given in Section 2.1 (Horn and Johnson 1991, Theorem 6.2.28). From (8.3) we have

$$f(A)b = \frac{1}{2\pi \mathrm{i}}\int_\Gamma f(z)\,(zI-A)^{-1}b\,\mathrm{d}z, \tag{8.4}$$

and so quadrature will reduce to solving linear systems with $zI-A$. Any method based on (8.4) will need to be specialized to particular classes of f and matrices A, since the selection of the contour Γ will be crucial to the efficiency and reliability of the method. Davies and Higham (2005) show that simply taking Γ to be a circle enclosing the spectrum is not generally a good choice. Hale, Higham and Trefethen (2008) develop methods for functions such as the square root and the logarithm with singularities in $(-\infty,0]$ and for A with eigenvalues on or near the positive real axis. Their key idea is to use conformal mappings to transform the integral into one for which the repeated trapezium rule converges very quickly. We mention just one simple idea used therein, and will not illustrate the conformal mappings that are the most important part of the technique. For the square root, we can rewrite the problem as $A \cdot A^{-1} f(A)$ and change variables to $w = z^{1/2}$, so that (8.3) becomes

$$A^{1/2} = \frac{A}{2\pi \mathrm{i}} \int_{\Gamma_z} z^{-1/2} \, (zI - A)^{-1} \, \mathrm{d}z = \frac{A}{\pi \mathrm{i}} \int_{\Gamma_w} (w^2 I - A)^{-1} \, \mathrm{d}w. \tag{8.5}$$

The transformed integrand is analytic at the origin and hence easier to handle. A natural question is what rational approximation these methods produce. For f the square root and A with positive real eigenvalues, method 3 in Hale *et al.* (2008) produces a certain best rational approximation discovered by Zolotarev in 1877. The methods in Hale *et al.* (2008) are not restricted to Hermitian matrices but they do require estimates of the spectrum of A.

8.2. Krylov methods

The most studied methods for the $f(A)b$ problem are those based on Krylov subspaces. By computing a sequence of matrix–vector products with A they aim to reduce the problem to one of the same form but with a much smaller matrix. This is done by projecting the problem onto a Krylov subspace

$$\mathcal{K}_k(A, b) = \mathrm{span}\{b, Ab, \ldots, A^{k-1} b\}.$$

If the Arnoldi process (Saad 2003, Section 6.3) with matrix A and starting vector $q_1 = b/\|b\|_2$ completes k steps then we have

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T, \tag{8.6}$$

where $Q_k = [q_1, \ldots, q_k]$ has orthonormal columns and $H_k = (h_{ij})$ is $k \times k$ upper Hessenberg. The columns of Q_k form an orthonormal basis for the Krylov subspace $\mathcal{K}_k(A, q_1)$. We can then approximate $f(A)b$ by

$$\begin{aligned} f_k &:= \|b\|_2 Q_k f(H_k) e_1, \\ &= Q_k f(H_k) Q_k^* b. \end{aligned} \tag{8.7}$$

The evaluation of f is carried out on the $k \times k$ matrix H_k, where $k \ll n$ in practice, and can be done by any available method. Effectively, we are evaluating f on the smaller Krylov subspace $\mathcal{K}_k(A, q_1)$ and then expanding the result back onto the original space $\mathbb{C}^n$. This procedure can be viewed as a form of model order reduction (Antoulas 2005, Frommer and Simoncini 2008*a*).

Few convergence results or error bounds are available for (8.7). However, two results of Saad (1992) provide fundamental insight into this approximation.

Lemma 8.1. Let $A \in \mathbb{C}^{n \times n}$ and Q_k, H_k be the result of k steps of the Arnoldi process on A with starting vector q_1. Then, for any polynomial p_j

of degree $j \le k-1$ we have

$$p_j(A)q_1 = Q_k p_j(H_k)e_1.$$

Theorem 8.2. Let Q_k, H_k be the result of k steps of the Arnoldi process on $A \in \mathbb{C}^{n\times n}$ with starting vector $q_1 = b/\|b\|_2$. Then

$$\|b\|_2 Q_k f(H_k)e_1 = \widetilde{p}_{k-1}(A)b,$$

where $\widetilde{p}_{k-1}$ is the unique polynomial of degree at most $k-1$ that interpolates f on the spectrum of H_k (that is, in the sense of (2.4)).

The lemma says that the approximation (8.7) is exact if f is a sufficiently low-degree polynomial. The theorem shows that the approximation (8.7) is an exact approximation not for f but for a Hermite interpolating polynomial based on the spectrum of H_k.

The use of Krylov methods for the $f(A)b$ problem is an active area of research. We will not try to give a summary, but instead point out a few very recent contributions, namely Afanasjew, Eiermann, Ernst and Güttel (2008), Grimm and Hochbruck (2008), Frommer and Simoncini (2008*b*), and Popolizio and Simoncini (2008).

9. The software scene

In this final section we give an outline of available software for computing matrix functions.

9.1. MATLAB

MATLAB has a number of functions for computing $f(A)$. Function `funm` implements the Schur–Parlett algorithm (Algorithm 4.5), and so is applicable to general functions having a Taylor series with an infinite radius of convergence. When invoked for the exponential it evaluates the exponential of any 2×2 diagonal blocks T_{ii} using an explicit formula (Higham 2008, Section 10.4.3) that, unlike the general formula (4.2), avoids cancellation in floating point arithmetic. Function `sqrtm` implements the Schur method for the matrix square root, Algorithm 4.6. Function `expm` implements Algorithm 5.2, the scaling and squaring algorithm. Function `logm` implements a specialized version of the Schur–Parlett algorithm in which $\log(T_{ii})$ is evaluated by an explicit formula (Higham 2008, Section 11.6.2) if T_{ii} is 2×2 or by the inverse scaling and squaring algorithm if T_{ii} has larger dimension.

The Symbolic Math Toolbox has two relevant functions, which are contained in the MuPAD engine (in MATLAB R2008b the default engine was changed from Maple to MuPAD). The function `numeric::expMatrix` (The MathWorks 2009*a*) can use hardware floating point arithmetic or variable precision software floating point arithmetic to compute e^A or $\mathrm{e}^A b$. By default a Taylor series is used, apparently without scaling and squaring. Other options are diagonalization for diagonalizable matrices, interpolation (the form of interpolating polynomial is not specified), and a Krylov method for $e^{A}b$ only. The function `numeric::fMatrix` (The MathWorks 2009*b*) computes $f(A)$ for a general function f but requires that A is diagonalizable.

9.2. Octave

GNU Octave (Octave 2009) is a free 'MATLAB-like' system. It includes a function `expm` that implements Ward's (1977) version of the scaling and squaring method (which uses fixed Padé degree $m = 8$, with scaling so that $\|2^{-s}A\|_1 \le 1$), as well as a function `thfm` for evaluating trigonometric and hyperbolic functions. The latter function expresses a variety of trigonometric and hyperbolic functions in terms of the exponential, and inverse trigonometric and inverse hyperbolic functions in terms of the logarithm and square root. For example, it evaluates $\cos(A) = (e^{iA} + e^{-iA})/2$ (or as $\operatorname{Re} e^{iA}$ when A is real) and $\arctan(A) = -(i/2)\log((I + iA)(I - iA)^{-1})$. Some of these formulas are of uncertain numerical reliability and need careful numerical stability analysis before they can be recommended for use; *cf.* the analysis for the scalar case in Bradford, Corless, Davenport, Jeffrey and Watt (2002) and Kahan (1987).

9.3. Other software

Algorithm 5.2 is used by the MatrixExp function of Mathematica for matrices of machine numbers and by the NAG Library routine `F01ECF` (from Mark 22).

The Matrix Function Toolbox (Higham) contains over 40 MATLAB functions implementing many of the algorithms described in Higham (2008).

Sidje (1998) provides a package called Expokit containing MATLAB and Fortran codes for computing e^{A} and $e^{A}b$.

Koikari (2009) gives Fortran 95 code for computing the ψ functions by scaling and squaring or by a block Schur–Parlett algorithm, and the EXPINT package of Berland, Skaflestad and Wright (2007) also contains a function based on scaling and squaring for evaluating the ψ functions. These functions are defined by $\psi_k(z) = \sum_{j=0}^{\infty} z^j/(j+k)!$, $k = 0, 1, 2, \ldots$ and play an important role within exponential integrators (Hochbruck and Ostermann 2010).

Acknowledgements

We thank Lijing Lin for her helpful comments on a draft manuscript. This work was supported in part by EPSRC grant EP/E050441/1 (CICADA: Centre for Interdisciplinary Computational and Dynamical Analysis).

Appendix: Cost of Padé versus Taylor approximants within the scaling and squaring method

In this appendix we compare the efficiency of diagonal Padé approximants and Taylor approximants within the scaling and squaring method for the matrix exponential, based on the use of refined backward error bounds in both cases.

For $A \in \mathbb{C}^{n\times n}$ we use the Paterson–Stockmeyer scheme (see Paterson and Stockmeyer (1973), Higham (2008, Section 4.2)) in order to evaluate $T_m(A) = \sum_{k=0}^{m} A^k/k!$ as

$$T_m(A) = \sum_{k=0}^{\ell} g_k(A)(A^\tau)^k, \quad \ell = \lfloor m/\tau \rfloor, \tag{A.1}$$

where $1 \le \tau \le m$ is an integer and

$$g_k(A) = \begin{cases} \sum_{i=1}^{\tau} A^{\tau-i}/(\tau k + \tau - i)!, & k = 0 : \ell - 1, \\ \sum_{i=\ell\tau}^{m} A^{i-\ell\tau}/i!, & k = \ell. \end{cases}$$

Horner's rule is used on (A.1). This scheme evaluates $T_m(A)$ with a number of matrix multiplications equal to

$$\widetilde{\pi}_m = \ell + \tau - 1 - \phi(m,\tau), \qquad \phi(m,\tau) = \begin{cases} 1, & \text{if } \tau \mid m, \\ 0, & \text{otherwise.} \end{cases} \tag{A.2}$$

The choice $\tau = \sqrt{m}$ approximately minimizes this quantity (Higham 2008, Section 4.2), so we take τ either $\lfloor \sqrt{m} \rfloor$ or $\lceil \sqrt{m}\, \rceil$ since both yield the same operation count (Hargreaves 2005, Theorem 1.7.4).

Lemma 5.1 is applicable with trivial modifications to any rational approximation to e^x, not just diagonal Padé approximants, so we can replace r_m therein by T_m. Thus, with $h_m(x) = \log(\mathrm{e}^{-x}T_m(x)) = \sum_{k=m+1}^{\infty} c_k x^k$ in Lemma 5.1, we calculate the parameters

$$\widetilde{\theta}_m = \max\{\, \theta : \widetilde{h}_m(\theta)/\theta \le u = 2^{-53} \,\}, \tag{A.3}$$

where $\widetilde{h}_m(x) = \sum_{k=m+1}^{\infty} |c_k| x^k$, using the techniques described just after Lemma 5.1. Then we know that $T_m(2^{-s}A)^{2^s}$ has backward error at most $u = 2^{-53}$ for $\|2^{-s}A\| \le \widetilde{\theta}_m$. We select s as the smallest non-negative integer such that $2^{-s}\|A\| \le \widetilde{\theta}_m$, which is given by $s = \max(\lceil \log_2(\|A\|/\widetilde{\theta}_m)\rceil, 0)$. Then the number of matrix multiplications required to evaluate $T_m(2^{-s}A)^{2^s}$ is

$$\underbrace{\lfloor m/\lceil \sqrt{m}\,\rceil \rfloor + \lceil \sqrt{m}\,\rceil - 1 - \phi(m, \lceil \sqrt{m}\,\rceil)}_{\widetilde{\pi}_m} + \max(\lceil \log_2(\|A\|/\widetilde{\theta}_m)\rceil, 0). \tag{A.4}$$

When $s > 0$ we are interested in the m that minimizes the cost. To obtain

Table A.1. The number of matrix products $\widetilde{\pi}_m$ in (A.2) needed for the Paterson–Stockmeyer scheme, $\widetilde{\theta}_m$ defined by (A.3), and C_m from (A.5).

m	$\widetilde{\theta}_m$	$\widetilde{\pi}_m$	C_m	m	$\widetilde{\theta}_m$	$\widetilde{\pi}_m$	C_m	m	$\widetilde{\theta}_m$	$\widetilde{\pi}_m$	C_m
1	2.29e-16	0	53.00	11	2.14e-1	5	8.22	21	1.62	8	8.30
2	2.58e-8	1	27.21	12	3.00e-1	5	7.74	22	1.82	8	8.14
3	1.39e-5	2	19.14	13	4.00e-1	6	8.32	23	2.01	8	7.99
4	3.40e-4	2	14.52	14	5.14e-1	6	7.96	24	2.22	8	7.85
5	2.40e-3	3	12.70	15	6.41e-1	6	7.64	25	2.43	8	7.72
6	9.07e-3	3	10.79	16	7.81e-1	6	7.36	26	2.64	9	8.60
7	2.38e-2	4	10.39	17	9.31e-1	7	8.10	27	2.86	9	8.48
8	5.00e-2	4	9.32	18	1.09	7	7.87	28	3.08	9	8.38
9	8.96e-2	4	8.48	19	1.26	7	7.67	29	3.31	9	8.27
10	1.44e-1	5	8.79	20	1.44	7	7.48	30	3.54	9	8.18

a suitable measure of the cost we ignore the constant terms in (A.4) (since they are common to each m) and consider

$$C_m = \lfloor m/\lceil \sqrt{m}\,\rceil \rfloor + \lceil \sqrt{m}\,\rceil - \phi(m, \lceil \sqrt{m}\,\rceil) - \log_2(\widetilde{\theta}_m). \tag{A.5}$$

We tabulate $\widetilde{\theta}_m$, $\widetilde{\pi}_m$, and C_m, for $m = 1\colon 30$, in Table A.1, and find that $m = 16$ is the global minimizer of C_m, which suggests using $T_{16}(2^{-s}A)^{2^s}$ to approximate e^A when $\|A\| \geq \widetilde{\theta}_{16} \approx 0.78$. Corresponding analysis was done for Padé approximants by Higham (2005) and we use the number of matrix multiplications π_m from Higham (2005, Table 2.2), as well as the θ_i in Table 5.1.

Now we compare the cost of Taylor and Padé approximants. Assume first that $\|A\| \geq \theta_{13} \approx 5.4$. Computing $T_{16}(2^{-s}A)$ requires six matrix multiplications, and so the overall cost from (A.4) of approximating e^A is $c_T := 6 + s$, while Algorithm 5.2, which chooses a non-negative integer t so that $\frac{1}{2}\theta_{13} < \|2^{-t}A\| \leq \theta_{13}$, computes $r_{13}(2^{-t}A)^{2^t}$ with cost $c_P := 6+4/3+t$, where the term 4/3 accounts for the solution of the multiple right-hand side linear system for the Padé approximant. Since $\frac{1}{2}\theta_{13} < 4\widetilde{\theta}_{16}$, there are two cases to consider. First, when $\|2^{-t}A\| \in (4\widetilde{\theta}_{16}, \theta_{13}]$ we have $2^{-t-3}\|A\| \leq \frac{1}{8}\theta_{13} < \widetilde{\theta}_{16}$ and hence $s = t + 3$. Therefore, $1 < c_T/c_P = (9 + t)/(7\frac{1}{3} + t) \leq 27/22$. Secondly, when $\|2^{-t}A\| \in (\frac{1}{2}\theta_{13}, 4\widetilde{\theta}_{16}]$ we have $2^{-t-2}\|A\| \leq \widetilde{\theta}_{16}$ and hence $s = t + 2$. Therefore, $1 < c_T/c_P = (8 + t)/(7\frac{1}{3} + t) \leq 12/11$.

A remaining question is whether when $\|A\| < \theta_{13}$ a Taylor series can be more efficient than a Padé approximant. The answer can be seen from Figure A.1, where '$\circ$' indicates the points $(\widetilde{\theta}_m, \widetilde{\pi}_m)$, $m = 4, 6, 9, 12, 16, 20, 25, 30$, and '$\square$' indicates the points $(\theta_m, \pi_m + 4/3)$, $m = 3, 5, 7, 9, 13$. Notice that the dotted curve, which represents the cost of Taylor series, lies below the

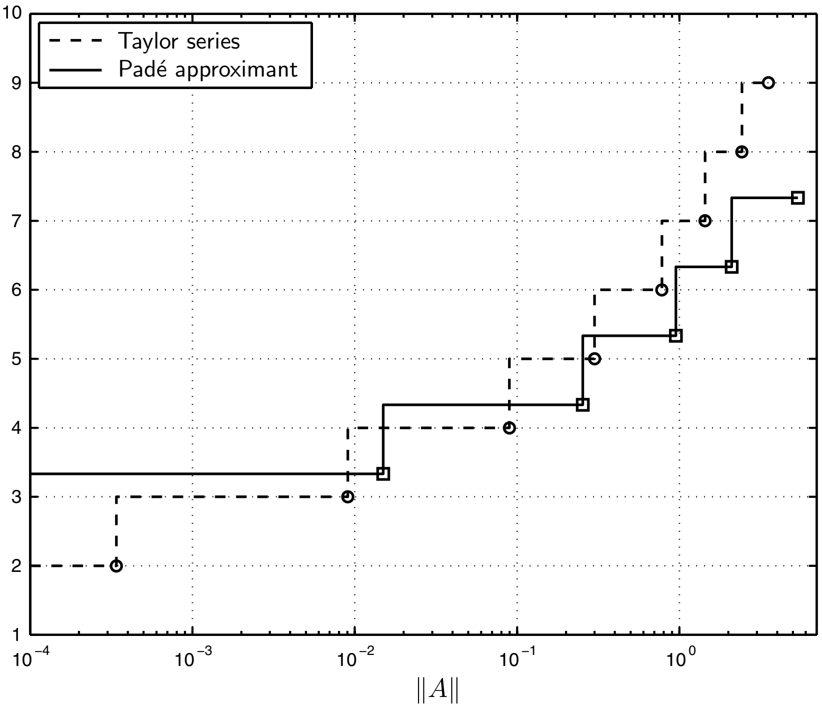

Figure A.1. $\|A\|$ versus cost in equivalent matrix multiplications of evaluating Taylor and Padé approximants to e^A in double precision.

solid curve in three intervals: $[0, \widetilde{\theta}_6]$, $(\theta_3, \widetilde{\theta}_9]$, and $(\theta_5, \widetilde{\theta}_{12}]$. Therefore, it is more efficient to use $T_m(A)$ rather Algorithm 5.2 if $\|A\|$ lies in any of these intervals.

We conclude that any algorithm based on Taylor series will cost up to 23% more than the Padé approximant-based Algorithm 5.2 and cannot have a lower cost for $\|A\| > \widetilde{\theta}_{12}$. Moreover the Taylor series requires a larger amount of scaling (since we are scaling to reduce $\|A\|$ below 0.78 instead of 5.4), which is undesirable from the point of view of possible numerical instability in the squaring phase.

We repeated the analysis for single precision: $u = 2^{-24} \approx 6.0 \times 10^{-8}$. For Padé approximants the optimal degree is now $m = 7$ with $\theta_7 \approx 3.9$ (Higham 2005), while for Taylor series it is $m = 9$ with $\widetilde{\theta}_9 \approx 0.78$. The conclusion is similar to that for double precision arithmetic: Padé approximation is more efficient than the Taylor series for $\|A\| > \widetilde{\theta}_9$ (up to 31% more efficient), and only for certain intervals of smaller $\|A\|$ is the Taylor series the more efficient.

In summary, Padé approximants are preferable to truncated Taylor series within the scaling and squaring method in both single and double precision, due to their greater efficiency and the lesser amount of scaling that they require.

REFERENCES

M. Afanasjew, M. Eiermann, O. G. Ernst and S. Güttel (2008), 'Implementation of a restarted Krylov subspace method for the evaluation of matrix functions', *Linear Algebra Appl.* **429**, 2293–2314.

A. H. Al-Mohy and N. J. Higham (2009*a*), 'Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation', *SIAM J. Matrix Anal. Appl.* **30**, 1639–1657.

A. H. Al-Mohy and N. J. Higham (2009*b*), 'A new scaling and squaring algorithm for the matrix exponential', *SIAM J. Matrix Anal. Appl.* **31**, 970–989.

A. H. Al-Mohy and N. J. Higham (2010), 'The complex step approximation to the Fréchet derivative of a matrix function', *Numer. Algorithms* **53**, 133–148.

A. C. Antoulas (2005), *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, PA, USA.

M. Arioli and D. Loghin (2009), 'Discrete interpolation norms with applications', *SIAM J. Numer. Anal.* **47**, 2924–2951.

G. A. Baker, Jr. and P. Graves-Morris (1996), *Padé Approximants*, Vol. 59 of *Encyclopedia of Mathematics and its Applications*, second edn, Cambridge University Press, Cambridge, UK.

A. Y. Barraud (1979), 'Investigations autour de la fonction signe d'une matrice application a l'équation de Riccati', *RAIRO Automatique/Systems Analysis and Control* **13**, 335–368.

C. A. Bavely and G. W. Stewart (1979), 'An algorithm for computing reducing subspaces by block diagonalization', *SIAM J. Numer. Anal.* **16**, 359–367.

H. Berland, B. Skaflestad and W. Wright (2007), 'EXPINT: A MATLAB package for exponential integrators', *ACM Trans. Math. Software* **33**, #4.

D. A. Bini, N. J. Higham and B. Meini (2005), 'Algorithms for the matrix pth root', *Numer. Algorithms* **39**, 349–378.

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

R. J. Bradford, R. M. Corless, J. H. Davenport, D. J. Jeffrey and S. M. Watt (2002), 'Reasoning about the elementary functions of complex analysis', *Annals of Mathematics and Artificial Intelligence* **36**, 303–318.

C. Brezinski and J. Van Iseghem (1995), A taste of Padé approximation. In *Acta Numerica*, Vol. 4, Cambridge University Press, pp. 53–103.

A. Cayley (1858), 'A memoir on the theory of matrices', *Philos. Trans. Roy. Soc. London* **148**, 17–37.

T. Charitos, P. R. de Waal and L. C. van der Gaag (2008), 'Computing short-interval transition matrices of a discrete-time Markov chain from partially observed data', *Statistics in Medicine* **27**, 905–921.

S. H. Cheng, N. J. Higham, C. S. Kenney and A. J. Laub (2000), Return to the middle ages: A half-angle iteration for the logarithm of a unitary matrix. In *Proc. 14th International Symposium of Mathematical Theory of Networks and Systems, Perpignan, France.* CD ROM.

S. H. Cheng, N. J. Higham, C. S. Kenney and A. J. Laub (2001), 'Approximating the logarithm of a matrix to specified accuracy', *SIAM J. Matrix Anal. Appl.* **22**, 1112–1125.

A. R. Collar (1978), 'The first fifty years of aeroelasticity', *Aerospace* (*Royal Aeronautical Society Journal*) **5**, 12–20.

M. G. Cox and P. M. Harris (2004), Numerical analysis for algorithm design in metrology, Software Support for Metrology Best Practice Guide No. 11, National Physical Laboratory, Teddington, UK.

J. J. Crofts and D. J. Higham (2009), 'A weighted communicability measure applied to complex brain networks', *J. Roy. Soc. Interface* **6**, 411–414.

P. I. Davies and N. J. Higham (2003), 'A Schur–Parlett algorithm for computing matrix functions', *SIAM J. Matrix Anal. Appl.* **25**, 464–485.

P. I. Davies and N. J. Higham (2005), Computing $f(A)b$ for matrix functions f. In *QCD and Numerical Analysis III* (A. Boriçi, A. Frommer, B. Joó, A. Kennedy and B. Pendleton, eds), Vol. 47 of *Lecture Notes in Computational Science and Engineering*, Springer, Berlin, pp. 15–24.

C. Davis (1973), 'Explicit functional calculus', *Linear Algebra Appl.* **6**, 193–199.

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

J. Descloux (1963), 'Bounds for the spectral norm of functions of matrices', *Numer. Math.* **15**, 185–190.

L. Dieci and A. Papini (2000), 'Padé approximation for the exponential of a block triangular matrix', *Linear Algebra Appl.* **308**, 183–202.

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

L. Elsner (1970), 'Iterative Verfahren zur Lösung der Matrizengleichung $X^2 - A = 0$', *Buletinul Institutului Politehnic din Iasi* **xvi**, 15–24.

E. Estrada and N. Hatano (2008), 'Communicability in complex networks', *Phys. Review E* **77**, 036111.

E. Estrada and D. J. Higham (2008), Network properties revealed through matrix functions. Mathematics Research Report 17, University of Strathclyde, Scotland, UK. To appear in *SIAM Rev.*

E. Estrada and J. A. Rodríguez-Velázquez (2005*a*), 'Spectral measures of bipartivity in complex networks', *Phys. Review E* **72**, 046105.

E. Estrada and J. A. Rodríguez-Velázquez (2005*b*), 'Subgraph centrality in complex networks', *Phys. Review E* **71**, 056103.

E. Estrada, D. J. Higham and N. Hatano (2009), 'Communicability betweenness in complex networks', *Physica A* **388**, 764–774.

S. Fiori (2008), 'Leap-frog-type learning algorithms over the Lie group of unitary matrices', *Neurocomputing* **71**, 2224–2244.

R. A. Frazer, W. J. Duncan and A. R. Collar (1938), *Elementary Matrices and Some Applications to Dynamics and Differential Equations*, Cambridge University Press, Cambridge, UK. 1963 printing.

A. Frommer and V. Simoncini (2008*a*), Matrix functions. In *Model Order Reduction: Theory, Research Aspects and Applications* (W. H. A. Schilders, H. A. van der Vorst and J. Rommes, eds), Springer, Berlin, pp. 275–303.

A. Frommer and V. Simoncini (2008*b*), 'Stopping criteria for rational matrix functions of Hermitian and symmetric matrices', *SIAM J. Sci. Comput.* **30**, 1387–1412.

F. R. Gantmacher (1959), *The Theory of Matrices*, Vol. one, Chelsea, New York.

M. B. Giles, M. C. Duta, J.-D. Müller and N. A. Pierce (2003), ‘Algorithm developments for discrete adjoint methods’, *AIAA Journal* **4**, 198–205.

H. H. Goldstine (1977), *A History of Numerical Analysis from the 16th through the 19th Century*, Springer, New York.

G. H. Golub and C. F. Van Loan (1996), *Matrix Computations*, third edn, Johns Hopkins University Press, Baltimore, MD, USA.

F. Greco and B. Iannazzo (2010), ‘A binary powering algorithm for computing primary matrix roots’, *Numer. Algorithms.* To appear.

V. Grimm and M. Hochbruck (2008), ‘Rational approximation to trigonometric operator’, *BIT* **48**, 215–229.

C.-H. Guo (2009), ‘On Newton’s method and Halley’s method for the principal pth root of a matrix’, *Linear Algebra Appl.* **432**, 1905–1922.

C.-H. Guo and N. J. Higham (2006), ‘A Schur–Newton method for the matrix pth root and its inverse’, *SIAM J. Matrix Anal. Appl.* **28**, 788–804.

N. Hale, N. J. Higham and L. N. Trefethen (2008), ‘Computing A^α, $\log(A)$ and related matrix functions by contour integrals’, *SIAM J. Numer. Anal.* **46**, 2505–2523.

G. Hargreaves (2005), Topics in matrix computations: Stability and efficiency of algorithms. PhD thesis, University of Manchester, Manchester, England.

G. I. Hargreaves and N. J. Higham (2005), ‘Efficient algorithms for the matrix cosine and sine’, *Numer. Algorithms* **40**, 383–400.

N. J. Higham (1986*a*), ‘Computing the polar decomposition: With applications’, *SIAM J. Sci. Statist. Comput.* **7**, 1160–1174.

N. J. Higham (1986*b*), ‘Newton’s method for the matrix square root’, *Math. Comp.* **46**, 537–549.

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

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

N. J. Higham (1997), ‘Stable iterations for the matrix square root’, *Numer. Algorithms* **15**, 227–242.

N. J. Higham (2001), ‘Evaluating Padé approximants of the matrix logarithm’, *SIAM J. Matrix Anal. Appl.* **22**, 1126–1135.

N. J. Higham (2002), *Accuracy and Stability of Numerical Algorithms*, second edn, SIAM, Philadelphia, PA, USA.

N. J. Higham (2005), ‘The scaling and squaring method for the matrix exponential revisited’, *SIAM J. Matrix Anal. Appl.* **26**, 1179–1193.

N. J. Higham (2008), *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, PA, USA.

N. J. Higham (2009), ‘The scaling and squaring method for the matrix exponential revisited’, *SIAM Rev.* **51**, 747–764.

N. J. Higham, ‘The Matrix Function Toolbox’. http://www.ma.man.ac.uk/~higham/mftoolbox.

N. J. Higham and L. Lin (2009), On pth roots of stochastic matrices. MIMS EPrint 2009.21, Manchester Institute for Mathematical Sciences, The University of Manchester, UK.

N. J. Higham and M. I. Smith (2003), ‘Computing the matrix cosine’, *Numer. Algorithms* **34**, 13–26.

N. J. Higham and F. Tisseur (2000), ‘A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra’, *SIAM J. Matrix Anal. Appl.* **21**, 1185–1201.

N. J. Higham, D. S. Mackey, N. Mackey and F. Tisseur (2005), ‘Functions preserving matrix groups and iterations for the matrix square root’, *SIAM J. Matrix Anal. Appl.* **26**, 849–877.

M. Hochbruck and A. Ostermann (2010), Exponential integrators. In *Acta Numerica*, Vol. 19, Cambridge University Press, pp. 209–286.

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

B. Iannazzo (2006), ‘On the Newton method for the matrix Pth root’, *SIAM J. Matrix Anal. Appl.* **28**, 503–523.

M. Ilić, I. W. Turner and D. P. Simpson (2009), ‘A restarted Lanczos approximation to functions of a symmetric matrix’, *IMA J. Numer. Anal.* Advance Access published on June 17, 2009. doi:10.1093/imanum/drp003.

R. B. Israel, J. S. Rosenthal and J. Z. Wei (2001), ‘Finding generators for Markov chains via empirical transition matrices, with applications to credit ratings’, *Mathematical Finance* **11**, 245–265.

R. A. Jarrow, D. Lando and S. M. Turnbull (1997), ‘A Markov model for the term structure of credit risk spreads’, *Rev. Financial Stud.* **10**, 481–523.

W. Kahan (1987), Branch cuts for complex elementary functions, or Much Ado About Nothing’s sign bit. In *The State of the Art in Numerical Analysis* (A. Iserles and M. J. D. Powell, eds), Oxford University Press, pp. 165–211.

C. S. Kenney and A. J. Laub (1989*a*), ‘Condition estimates for matrix functions’, *SIAM J. Matrix Anal. Appl.* **10**, 191–209.

C. S. Kenney and A. J. Laub (1989*b*), ‘Padé error estimates for the logarithm of a matrix’, *Internat. J. Control* **50**, 707–730.

C. S. Kenney and A. J. Laub (1991*a*), ‘Polar decomposition and matrix sign function condition estimates’, *SIAM J. Sci. Statist. Comput.* **12**, 488–504.

C. S. Kenney and A. J. Laub (1991*b*), ‘Rational iterative methods for the matrix sign function’, *SIAM J. Matrix Anal. Appl.* **12**, 273–291.

C. S. Kenney and A. J. Laub (1998), ‘A Schur–Fréchet algorithm for computing the logarithm and exponential of a matrix’, *SIAM J. Matrix Anal. Appl.* **19**, 640–663.

S. Koikari (2009), ‘Algorithm 894: On a block Schur–Parlett algorithm for φ-functions based on the sep-inverse estimate’, *ACM Trans. Math. Software* **36**, #12.

A. Kreinin and M. Sidelnikova (2001), ‘Regularization algorithms for transition matrices’, *Algo Research Quarterly* **4**, 23–40.

P. Laasonen (1958), ‘On the iterative solution of the matrix equation $AX^2 - I = 0$’, *Math. Tables Aids Comp.* **12**, 109–116.

P. Lancaster and M. Tismenetsky (1985), *The Theory of Matrices*, second edn, Academic Press, London.

B. Laszkiewicz and K. Ziętak (2009), 'A Padé family of iterations for the matrix sector function and the matrix pth root', *Numer. Linear Algebra Appl.* **16**, 951–970.

P.-F. Lavallée, A. Malyshev and M. Sadkane (1997), Spectral portrait of matrices by block diagonalization. In *Numerical Analysis and its Applications* (L. Vulkov, J. Waśniewski and P. Yalamov, eds), Vol. 1196 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 266–273.

J. D. Lawson (1967), 'Generalized Runge-Kutta processes for stable systems with large Lipschitz constants', *SIAM J. Numer. Anal.* **4**, 372–380.

J. R. R. A. Martins, P. Sturdza and J. J. Alonso (2003), 'The complex-step derivative approximation', *ACM Trans. Math. Software* **29**, 245–262.

R. Mathias (1996), 'A chain rule for matrix functions and applications', *SIAM J. Matrix Anal. Appl.* **17**, 610–620.

The MathWorks (2009*a*), 'numeric::expMatrix: The exponential of a matrix'. http://www.mathworks.com/access/helpdesk/help/toolbox/mupad/numeric/expMatr.html, retrieved on September 29, 2009.

The MathWorks (2009*b*), 'numeric::fMatrix: Functional calculus for numerical square matrices'. http://www.mathworks.com/access/helpdesk/help/toolbox/mupad/numeric/fMatrix.html, retrieved on September 29, 2009.

C. B. Moler and C. F. Van Loan (1978), 'Nineteen dubious ways to compute the exponential of a matrix', *SIAM Rev.* **20**, 801–836.

C. B. Moler and C. F. Van Loan (2003), 'Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later', *SIAM Rev.* **45**, 3–49.

L. Morai and A. F. Pacheco (2003), 'Algebraic approach to the radioactive decay equations', *Amer. J. Phys.* **71**, 684–686.

Octave Version 3.2.2 (2009). http://www.octave.org.

B. N. Parlett (1976), 'A recurrence among the elements of functions of triangular matrices', *Linear Algebra Appl.* **14**, 117–121.

B. N. Parlett and K. C. Ng (1985), Development of an accurate algorithm for $\exp(Bt)$. Technical Report PAM-294, Center for Pure and Applied Mathematics, University of California, Berkeley. Fortran program listings are given in an appendix with the same report number printed separately.

M. S. Paterson and L. J. Stockmeyer (1973), 'On the number of nonscalar multiplications necessary to evaluate polynomials', *SIAM J. Comput.* **2**, 60–66.

H.-O. Peitgen, H. Jürgens and D. Saupe (1992), *Fractals for the Classroom, Part Two: Complex Systems and Mandelbrot Set*, Springer, New York.

G. M. Phillips (2000), *Two Millennia of Mathematics: From Archimedes to Gauss*, Springer, New York.

M. Popolizio and V. Simoncini (2008), 'Acceleration techniques for approximating the matrix exponential operator', *SIAM J. Matrix Anal. Appl.* **30**, 657–683.

P. J. Psarrakos (2002), 'On the mth roots of a complex matrix', *Electron. J. Linear Algebra* **9**, 32–41.

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

R. F. Rinehart (1955), 'The equivalence of definitions of a matric function', *Amer. Math. Monthly* **62**, 395–414.

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

Y. Saad (1992), 'Analysis of some Krylov subspace approximations to the matrix exponential operator', *SIAM J. Numer. Anal.* **29**, 209–228.

Y. Saad (2003), *Iterative Methods for Sparse Linear Systems*, second edn, SIAM, Philadelphia, PA, USA.

M. Schroeder (1991), *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*, W. H. Freeman, New York.

S. M. Serbin and S. A. Blalock (1980), 'An algorithm for computing the matrix cosine', *SIAM J. Sci. Statist. Comput.* **1**, 198–204.

L. F. Shampine (2007), 'Accurate numerical derivatives in MATLAB', *ACM Trans. Math. Software.* #26, 17 pp.

R. B. Sidje (1998), 'Expokit: A software package for computing matrix exponentials', *ACM Trans. Math. Software* **24**, 130–156.

R. B. Sidje, 'Expokit'. http://www.maths.uq.edu.au/expokit, retrieved October 8, 2009.

M. I. Smith (2003), 'A Schur algorithm for computing matrix pth roots', *SIAM J. Matrix Anal. Appl.* **24**, 971–989.

W. Squire and G. Trapp (1998), 'Using complex variables to estimate derivatives of real functions', *SIAM Rev.* **40**, 110–112.

C. F. Van Loan (1975), A study of the matrix exponential. Numerical Analysis Report No. 10, University of Manchester, Manchester, UK. Reissued as MIMS EPrint 2006.397, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, November 2006.

C. F. Van Loan (1978), 'Computing integrals involving the matrix exponential', *IEEE Trans. Automat. Control* **AC-23**, 395–404.

C. F. Van Loan (1979), 'A note on the evaluation of matrix polynomials', *IEEE Trans. Automat. Control* **AC-24**, 320–321.

R. S. Varga (2000), *Matrix Iterative Analysis*, second edn, Springer, Berlin.

C. Visser (1937), 'Note on linear operators', *Proc. Kon. Akad. Wet. Amsterdam* **40**, 270–272.

R. C. Ward (1977), 'Numerical computation of the matrix exponential with accuracy estimate', *SIAM J. Numer. Anal.* **14**, 600–610.

F. V. Waugh and M. E. Abel (1967), 'On fractional powers of a matrix', *J. Amer. Statist. Assoc.* **62**, 1018–1021.

D. Yuan and W. Kernan (2007), 'Explicit solutions for exit-only radioactive decay chains', *J. Appl. Phys.* **101**, 094907 1–12.