A Schur–Padé Algorithm for Fractional Powers of a Matrix

Higham, Nicholas J. and Lin, Lijing

2010

MIMS EPrint: **2010.91**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

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

ISSN 1749-9097

A SCHUR–PADÉ ALGORITHM FOR FRACTIONAL POWERS OF A MATRIX*

NICHOLAS J. HIGHAM† AND LIJING LIN†

Abstract. A new algorithm is developed for computing arbitrary real powers A^p of a matrix $A \in \mathbb{C}^{n\times n}$. The algorithm starts with a Schur decomposition, takes k square roots of the triangular factor T, evaluates an $[m/m]$ Padé approximant of $(1-x)^p$ at $I - T^{1/2^k}$, and squares the result k times. The parameters k and m are chosen to minimize the cost subject to achieving double precision accuracy in the evaluation of the Padé approximant, making use of a result that bounds the error in the matrix Padé approximant by the error in the scalar Padé approximant with argument the norm of the matrix. The Padé approximant is evaluated from the continued fraction representation in bottom-up fashion, which is shown to be numerically stable. In the squaring phase the diagonal and first superdiagonal are computed from explicit formulae for $T^{p/2^j}$, yielding increased accuracy. Since the basic algorithm is designed for $p \in (-1, 1)$, a criterion for reducing an arbitrary real p to this range is developed, making use of bounds for the condition number of the A^p problem. How best to compute A^k for a negative integer k is also investigated. In numerical experiments the new algorithm is found to be superior in accuracy and stability to several alternatives, including the use of an eigendecomposition and approaches based on the formula $A^p = \exp(p\log(A))$.

Key words. matrix power, matrix root, fractional power, primary matrix function, Schur decomposition, Padé approximation, Padé approximant, matrix logarithm, matrix exponential, MATLAB

AMS subject classifications. 65F30

1. Introduction.

The need to compute fractional powers A^p of a square matrix A arises in a variety of applications, including Markov chain models in finance and healthcare [8], [28], fractional differential equations [27], discrete representations of norms corresponding to finite element discretizations of fractional Sobolev spaces [3], and the computation of geodesic-midpoints in neural networks [11]. Here, p is an arbitrary real number, not necessarily rational. Often, p is the reciprocal of a positive integer q, in which case $X = A^p = A^{1/q}$ is a qth root of A. Various methods are available for the qth root problem, based on the Schur decomposition and appropriate recurrences [13], [35], Newton or inverse Newton iterations [14], [25], Padé iterations [26], [31], or a variety of other techniques [6]; see [21, Chap. 7] and [23] for surveys. However, none of these methods is applicable for arbitrary real p.

Arbitrary matrix powers can be defined via the Cauchy integral [21, Def. 1.11]

$$A^p := \frac{1}{2\pi i}\int_\Gamma z^p (zI - A)^{-1} dz, \tag{1.1}$$

where Γ is a closed contour that encloses the spectrum $\Lambda(A)$. This definition yields many different matrices A^p, as the branch of the function z^p can be chosen independently around each eigenvalue. For practical purposes it is more useful to define A^p uniquely as follows.

---

*Version of October 19, 2010.

†School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (higham@ma.man.ac.uk, http://www.ma.man.ac.uk/~higham, Lijing.Lin@postgrad.manchester.ac.uk, http://www.maths.manchester.ac.uk/~lijing). The work of the first author was supported by a Royal Society-Wolfson Research Merit Award and by Engineering and Physical Sciences Research Council grant EP/E050441/1 (CICADA: Centre for Interdisciplinary Computational and Dynamical Analysis).

DEFINITION 1.1. *Let $A \in \mathbb{C}^{n\times n}$ have no eigenvalues on $\mathbb{R}^-$ except possibly for a semisimple zero eigenvalue, and let $p \in \mathbb{R}$. If A is nonsingular,*

$$A^p = \exp\left(p \log(A)\right), \tag{1.2}$$

where $\log(A)$ is the principal logarithm of A [21, Thm. 1.31]. Otherwise, write the Jordan canonical form of A as $A = Z\mathrm{diag}(J_1, 0)Z^{-1}$, where J_1 contains the Jordan blocks corresponding to the nonzero eigenvalues. Then

$$A^p = Z\mathrm{diag}(J_1^p, 0)Z^{-1}, \tag{1.3}$$

where J_1^p is defined by (1.2).

It follows from the theory of matrix functions that the matrix given by Definition 1.1 is independent of the particular choice of Jordan canonical form. Moreover, if A is real then A^p is real. For $p = 1/q$, with q a positive integer, A^p reduces to the principal qth root of A [21, Thm. 7.2]. For $0 < p < 1$, A^p can also be represented as the real integral [21, pp. 174, 187]

$$A^p = \frac{\sin(p\pi)}{p\pi} A \int_0^\infty (t^{1/p} I + A)^{-1}\, dt. \tag{1.4}$$

The aim of this work is to devise a reliable algorithm for computing A^p for arbitrary $p \in \mathbb{R}$. When A is diagonalizable, so that $A = XDX^{-1}$ for a diagonal $D = \mathrm{diag}(d_i)$ and nonsingular X, we can compute $A^p = XD^pX^{-1} = X\mathrm{diag}(d_i^p)X^{-1}$. Alternatively, for any A we can compute the Schur decomposition $A = QTQ^*$, with Q unitary and T upper triangular, from which $A^p = QT^pQ^*$. The matrix T^p has diagonal elements t_{ii}^p and we can obtain the superdiagonal elements from the Parlett recurrence if the t_{ii} are distinct [21, sec. 4.6], [33]. However, this approach breaks down when A is nonnormal with repeated eigenvalues.

The definition (1.2) suggests another way to compute A^p: to employ existing algorithms for the matrix exponential and the matrix logarithm. However, if we use the inverse scaling and squaring method for $X = \log(A)$ [9], [21, sec. 11.5], [29] followed by the scaling and squaring method for $\exp(pX)$ [1], [20], [22] then we are computing two Padé approximants: one of the logarithm and the other of the exponential. We expect benefits to accrue from employing a *single* Padé approximant, to $(1-x)^p$. In this work we develop an algorithm for computing A^p based on direct Padé approximation of $(1-x)^p$.

We begin, in Section 2, by investigating the conditioning of fractional powers. Padé approximation of $(1-x)^p$, and in particular how to bound the error in the approximation at a matrix argument, is the subject of Section 3. Evaluation of the matrix Padé approximant is considered in Section 4, where we investigate the numerical stability of the continued fraction representation evaluated in the bottom-up fashion. An algorithm for A^p with $p \in (-1, 1)$ that employs an initial Schur decomposition, matrix square roots, Padé approximation, and squarings, is developed in Section 5. In Section 6 we explain how to deal with general p not necessarily in the interval $(-1, 1)$ and negative integer p, while in Section 7 we extend our algorithm to handle singular matrices with a semisimple zero eigenvalue. Some alternative algorithms are considered in Section 8 and all the algorithms are compared in the numerical experiments of Section 9. Finally, some concluding remarks are given in Section 10.

2. Conditioning. We first investigate the sensitivity of A^p to perturbations in A. We denote by $L_f(A, E)$ the Fréchet derivative of f at A in the direction E,

which is a linear operator mapping E to $L_f(A,E)$ characterized by $f(A+E) = f(A) + L_f(A,E) + o(\|E\|)$. We also recall the definition and characterization of condition number

$$\kappa_f(A) := \lim_{\epsilon\to 0} \sup_{\|E\|\le\epsilon\|A\|} \frac{\|f(A+E)-f(A)\|}{\epsilon\|f(A)\|} = \frac{\|L_f(A)\|\,\|A\|}{\|f(A)\|}, \tag{2.1}$$

where

$$\|L_f(X)\| := \max_{Z\ne 0} \frac{\|L_f(X,Z)\|}{\|Z\|}. \tag{2.2}$$

For background on Fréchet derivatives and condition numbers see [21, secs. 3.1, 3.2].

Let vec denote the operator that stacks the columns of a matrix into one long vector and let $\otimes$ denote the Kronecker product. For any f, we have $\mathrm{vec}(L_f(A,E)) = K_f(A)\mathrm{vec}(E)$ for a certain matrix $K_f(A) \in \mathbb{C}^{n^2\times n^2}$ called the Kronecker representation of the Fréchet derivative and, moreover, $\|L_f(A)\|_F = \|K_f(A)\|_2$ [21, (3.20)]. It follows that, in the Frobenius norm,

$$\kappa_f(A) = \frac{\|K_f(A)\|_2\,\|A\|_F}{\|f(A)\|_F}. \tag{2.3}$$

To obtain a formula for $K_{x^p}(A)$ we first apply the chain rule [21, Thm. 3.4] to the expression $A^p = \exp(p\log(A))$, to obtain

$$L_{x^p}(A,E) = pL_{\exp}\big(p\log(A), L_{\log}(A,E)\big). \tag{2.4}$$

Then, by applying the vec operator, we find that

$$\mathrm{vec}(L_{x^p}(A,E)) = pK_{\exp}(p\log(A))\mathrm{vec}(L_{\log}(A,E)) = pK_{\exp}(p\log(A))K_{\log}(A)\mathrm{vec}(E),$$

which implies

$$K_{x^p}(A) = pK_{\exp}(p\log(A))K_{\log}(A). \tag{2.5}$$

This matrix can be computed explicitly if n is small, or its norm estimated based on a few matrix–vector products involving $K_{x^p}(A)$ and its conjugate transpose [21, sec. 3.4].

We now derive some bounds for the condition number $\kappa_{x^p}(A)$ that give insight into its size. First, note that, since $(A+\epsilon I)^p = A^p + p\epsilon A^{p-1} + O(\epsilon^2)$ for sufficiently small ϵ (by a general result on the convergence of a matrix Taylor series [21, Thm. 4.7]), we have $L_{x^p}(A,I) = pA^{p-1}$ and hence $\|L_{x^p}(A)\| \ge |p|\,\|A^{p-1}\|/\|I\|$.

Since [21, (10.15)]

$$L_{\exp}(A,E) = \int_0^1 e^{A(1-s)}Ee^{As}\,ds, \tag{2.6}$$

we have, from (2.4),

$$\begin{aligned}
\|L_{x^p}(A,E)\| &= |p|\left\|\int_0^1 e^{p\log(A)(1-s)}L_{\log}(A,E)e^{p\log(A)s}ds\right\| \\
&\le |p|\,\|L_{\log}(A,E)\|\int_0^1 e^{|p|(1-s)\|\log(A)\|}e^{|p|s\|\log(A)\|}ds \\
&\le |p|e^{|p|\|\log(A)\|}\|L_{\log}(A)\|\;\|E\|,
\end{aligned}$$

and so $\|L_{x^p}(A)\| \le |p| e^{|p|\|\log(A)\|} \|L_{\log}(A)\|$. Thus we have the upper and lower bounds

$$\frac{|p|\,\|A^{p-1}\|}{\|I\|} \le \|L_{x^p}(A)\| \le |p| e^{|p|\,\|\log(A)\|}\ \|L_{\log}(A)\|. \tag{2.7}$$

We also have the following lower bound [21, Thm. 3.14, Cor. 3.16], with $f[\lambda,\mu]$ denoting the first divided difference of $f(x) = x^p$,

$$\|L_{x^p}(A)\| \ge \max_{\lambda,\mu\in\Lambda(A)} |f[\lambda,\mu]| = \max\left(\max_{\lambda\in\Lambda(A)} |p||\lambda^{p-1}|,\ \max_{\substack{\lambda,\mu\in\Lambda(A)\\ \lambda\ne\mu}} \frac{|\lambda^p - \mu^p|}{|\lambda-\mu|} \right), \tag{2.8}$$

which is an equality for the Frobenius norm when A is normal. When A is Hermitian the lower bounds in (2.7) and (2.8) are the same for the 2-norm; we will make use of the lower bound in this case in Section 6.

3. Padé approximation and error bounds. A $[k/m]$ Padé approximant of $(1-x)^p$ is a rational function $r_{km}(x) = p_{km}(x)/q_{km}(x)$ with $q_{km}(0) = 1$ such that

$$(1-x)^p - r_{km}(x) = O(x^{k+m+1}),$$

where p_{km} and q_{km} are polynomials of degree at most k and m, respectively. If a $[k/m]$ Padé approximant exists then it is unique [4, Thm. 1.1], [5, Thm. 1.4.3], [21, Prob. 4.2]. The aims of this section are to show the existence of Padé approximants of $(1-x)^p$ and to investigate the error in the Padé approximant at a matrix argument $X \in \mathbb{C}^{n\times n}$ with $\|X\| < 1$. Throughout this section the norm is assumed to be a subordinate matrix norm.

The scalar hypergeometric function is

$${}_2F_1(\alpha,\beta,\gamma,x) \equiv 1 + \frac{\alpha\beta}{\gamma}x + \frac{\alpha(\alpha+1)\beta(\beta+1)}{2!\gamma(\gamma+1)}x^2 + \cdots = \sum_{i=0}^{\infty} \frac{(\alpha)_i(\beta)_i}{i!(\gamma)_i}x^i, \tag{3.1}$$

where $\alpha,\beta,\gamma,x \in \mathbb{R}$, γ is not a nonpositive integer, $(a)_0 = 1$, and $(a)_i \equiv a(a+1)\dots(a+i-1)$ for $i \ge 1$. Replacing x in (3.1) with $X \in \mathbb{C}^{n\times n}$ we obtain the matrix hypergeometric function

$${}_2F_1(\alpha,\beta,\gamma,X) \equiv \sum_{i=0}^{\infty} \frac{(\alpha)_i(\beta)_i}{i!(\gamma)_i}X^i. \tag{3.2}$$

Since (3.1) converges if $|x| < 1$ [2, Thm. 2.1.1], the matrix series (3.2) converges if $\rho(X) < 1$ [21, Thm. 4.7], where ρ is the spectral radius. We are interested in the special case where $\alpha = -p,\ \beta = 1,\ \gamma = 1$, and $|x| < 1$:

$${}_2F_1(-p,1,1,x) = 1 - px + \frac{p(p-1)}{2}x^2 + \cdots = (1-x)^p.$$

The following lemma shows the existence of the Padé approximants of $(1-x)^p$ for all $p \in \mathbb{R}$.

LEMMA 3.1. *For $p \in \mathbb{R}$, the $[k/m]$ Padé approximant of $(1-x)^p$ exists for all nonnegative integers k and m.*

Proof. It is shown in [4, p. 65], [5, sec. 2.3] that for any $\alpha,\gamma \in \mathbb{R}$ the $[k/m]$ Padé approximant of the general hypergeometric function ${}_2F_1(\alpha,1,\gamma,x)$ exists for

$k-m+1 \geq 0$ and that the denominator $q_{km}(x)$ is given explicitly by

$$q_{km}(x) = \sum_{i=0}^{m} \frac{(-m)_i(-(\alpha+k))_i}{i!(1-(\gamma+k+m))_i} x^i \tag{3.3}$$

$$= {}_2F_1(-m, -(\alpha+k), 1-(\gamma+k+m), x). \tag{3.4}$$

Thus $[k/m]$ Padé approximants to $(1-x)^p$ exist for all $p \in \mathbb{R}$ for $k \geq m$. From $(1-x)^p = 1/(1-x)^{-p}$, and the duality property that the $[k/m]$ Padé approximant of the reciprocal of a function is the reciprocal of the $[m/k]$ Padé approximant of the function [5, Thm. 1.5.1], it follows that $(1-x)^p$ has a $[k/m]$ Padé approximant for $k \leq m$. ∎

We now state some properties of $q_{km}(x)$. The following result of Kenney and Laub bounds the condition number number of the matrix $q_{km}(X)$.

LEMMA 3.2. *Let $q_{km}(x)$ be the denominator polynomial of the $[k/m]$ Padé approximant of ${}_2F_1(\alpha, 1, \gamma, x)$ where $0 < \alpha < \gamma$ and $k-m+1 \geq 0$. The zeros of $q_{km}(x)$ are all simple and lie in the interval $(1, \infty)$. Furthermore, for $X \in \mathbb{C}^{n\times n}$ with $\|X\| < 1$,*

$$\|q_{km}(X)\| \leq q_{km}(-\|X\|), \quad \|q_{km}(X)^{-1}\| \leq q_{km}(\|X\|)^{-1} \tag{3.5}$$

and hence

$$\kappa(q_{km}(X)) \leq \frac{q_{km}(-\|X\|)}{q_{km}(\|X\|)}. \tag{3.6}$$

Proof. See [30, Cor. 1 and Lem. 3], where $X \in \mathbb{R}^{n\times n}$ is assumed; the proofs there are nevertheless valid for complex X. ∎

COROLLARY 3.3. *Let $q_{km}(x)$ be the denominator polynomial of the $[k/m]$ Padé approximant of $(1-x)^p$ with $-1 < p < 1$ and $k-m \geq 0$. Then the zeros of $q_{km}(x)$ are all simple and lie in the interval $(1, \infty)$ and for $X \in \mathbb{C}^{n\times n}$ with $\|X\| < 1$, the matrix $q_{km}(X)$ satisfies (3.5) and (3.6). In particular, when $-1 < p < 0$ these conclusions hold for $k-m+1 \geq 0$.*

Proof. It is straightforward to show that $(1-x)^p = 1 - px \cdot {}_2F_1(1-p, 1, 2, x)$ and, moreover, that if $k \geq m$ then the $[k/m]$ Padé approximant of $(1-x)^p$ is $p_{km}/\widetilde{q}_{k-1,m} = 1 - px\widetilde{r}_{k-1,m}$, where $\widetilde{r}_{k-1,m} = \widetilde{p}_{k-1,m}/\widetilde{q}_{k-1,m}$ is the $[k-1/m]$ Padé approximant of ${}_2F_1(1-p, 1, 2, x)$.

Since $-1 < p < 1$ we have $0 < 1-p < 2$, and since also $(k-1) - m + 1 \geq 0$ the properties of $\widetilde{q}_{k-1,m}(x)$ in Lemma 3.2 all hold. If $-1 < p < 0$, it follows from Lemma 3.2 with $\alpha = -p$ and $\gamma = 1$ that the conclusions hold for $k-m+1 \geq 0$. ∎

Denote by $E\big({}_2F_1(\alpha, 1, \gamma, \cdot), k, m, x\big)$ the error in the $[k/m]$ Padé approximant to ${}_2F_1(\alpha, 1, \gamma, x)$, that is,

$$E\big({}_2F_1(\alpha, 1, \gamma, \cdot), k, m, x\big) = {}_2F_1(\alpha, 1, \gamma, x) - r_{km}(x). \tag{3.7}$$

The following lemma provides a series expansion for this error.

LEMMA 3.4. *For $|x| < 1$, $k-m+1 \geq 0$, and α not a negative integer, the error (3.7) can be written*

$$E\big({}_2F_1(\alpha, 1, \gamma, \cdot), k, m, x\big) = \frac{q_{km}(1)}{q_{km}(x)} \sum_{i=k+m+1}^{\infty} \frac{(\alpha)_i(i-(k+m))_m}{(\gamma)_i(i+\alpha-m)_m} x^i. \tag{3.8}$$

Proof. See Kenney and Laub [30, Thm. 5]. The statement of Theorem 5 in [30] requires $0 < \alpha < \gamma$, but in fact only the condition that α is not a negative integer (and hence $(i+\alpha-m)_m$ is nonzero) is needed in the proof. □

We are now in a position to bound the error in Padé approximation of the matrix function $(I-X)^p = {}_2F_1(-p,1,1,X)$. The following result, which for $-1 < p < 0$ is a special case of [30, Cor. 4], shows that the error is bounded by the error of the same approximation at the scalar argument $\|X\|$.

THEOREM 3.5. *For $k-m \geq 0$, $-1 < p < 1$, and $\|X\| < 1$,*

$$\|E\big((I-X)^p, k, m, X\big)\| \leq |E\big((1-\|X\|)^p, k, m, \|X\|\big)|. \tag{3.9}$$

In particular, when $-1 < p < 0$, (3.9) holds for $k-m+1 \geq 0$.

Proof. For any matrix X with $\|X\| < 1$, $(I-X)^p = {}_2F_1(-p,1,1,X)$ is defined and, by (3.8),

$$E((I-X)^p, k, m, X) = q_{km}(1)q_{km}(X)^{-1} \sum_{i=k+m+1}^{\infty} \frac{(-p)_i (i-(k+m))_m}{i!(i-p-m)_m} X^i, \tag{3.10}$$

where $q_{km}(x)$ is the denominator of the $[k/m]$ Padé approximant to $(1-x)^p$. We claim that every coefficient in the sum has the same sign, that is, the signs are independent of i for $i \geq k+m+1$. Indeed, $(-p)_i < 0$ for $0 < p < 1$ and $(-p)_i > 0$ for $-1 < p < 0$, and clearly $(i-(k+m))_m > 0$ and $(i-p-m)_m > 0$. Therefore, by Corollary 3.3 and the second inequality in (3.5), we have

$$\begin{aligned}
\|E((I-X)^p, k, m, X)\| &\leq \frac{|q_{km}(1)|}{q_{km}(\|X\|)} \sum_{i=k+m+1}^{\infty} \frac{|(-p)_i|(i-(k+m))_m}{i!(i-p-m)_m} \|X\|^i \\
&= \frac{|q_{km}(1)|}{q_{km}(\|X\|)} \left| \sum_{i=k+m+1}^{\infty} \frac{(-p)_i (i-(k+m))_m}{i!(i-p-m)_m} \|X\|^i \right| \\
&= \big|E\big((1-\|X\|)^p, k, m, \|X\|\big)\big|.
\end{aligned}$$

If $-1 < p < 0$, the result holds for $k-m+1 \geq 0$, since Corollary 3.3 shows that the required bound $\|q_{km}(X)^{-1}\| \leq q_{km}(\|X\|)^{-1}$ still holds in this case. □

In practice, we would like to select k and m to minimize the error for a given order of approximation. The following result of Kenny and Laub [30, Thm. 6] is useful in this respect.

THEOREM 3.6. *Let $k-m+1 \geq 0$ and $0 < \alpha < \gamma$, and let the subordinate matrix norm $\|\cdot\|$ satisfy $\|\widetilde{M}\| \leq \|M\|$ whenever $0 \leq \widetilde{M} \leq M$, where the latter inequalities are interpreted componentwise. Then, if $X \in \mathbb{R}^{n\times n}$ has nonnegative entries,*

$$\|E\big({}_2F_1(\alpha,1,\gamma,\cdot), k, m, X\big)\| \leq \|E\big({}_2F_1(\alpha,1,\gamma,\cdot), k+1, m-1, X\big)\|. \tag{3.11}$$

Applying Theorem 3.6 with $\alpha = -p \in (0,1)$ and $\gamma = 1$, we obtain the corresponding result for $(I-X)^p$, where $-1 < p < 0$. For $0 < p < 1$, the inequality (3.11) holds for k, m satisfying $k-m \geq 0$; this can be proved in the same way as Theorem 3.6, using Corollary 3.3. We conclude that when X has nonnegative entries, the error is reduced as k and m approach the main diagonal ($k = m$) and first superdiagonal ($k+1 = m$) of the Padé table. In the rest of the paper we will concentrate on the use of the diagonal Padé approximants $r_m \equiv r_{mm}$.

4. Evaluating Padé approximants of $(I - X)^p$. The Padé approximant $r_m(x)$ to $(1-x)^p$ has the continued fraction expansion [4, p. 66], [5, p. 174]

$$r_m(x) = 1 + \cfrac{c_1 x}{1 + \cfrac{c_2 x}{1 + \cfrac{c_3 x}{\cdots \cfrac{}{1 + \cfrac{c_{2m-1}x}{1 + c_{2m}x}}}}}, \tag{4.1}$$

where

$$c_1 = -p, \quad c_{2j} = \frac{-j+p}{2(2j-1)}, \quad c_{2j+1} = \frac{-j-p}{2(2j+1)}, \quad j = 1, 2, \ldots.$$

This expansion provides a convenient means to evaluate $r_m(X)$ for $X \in \mathbb{C}^{n\times n}$. However, just as for the logarithm [18], there are several possible methods for evaluation at a matrix argument:

1. Top-down evaluation of (4.1).
2. Bottom-up evaluation of (4.1).
3. Evaluation of the numerator and denominator in the representation $r_m(x) = p_m(x)/q_m(x)$ by Horner's method or the Paterson and Stockmeyer method [21, sec. 4.2], [34].
4. Evaluation of $r_m(x) = p_m(x)/q_m(x)$ using the representations of p_m and q_m as products of linear factors (the zeros of p_m and q_m are all real).
5. Evaluation of the partial fraction representation $r_m(x) = \alpha_0 + \sum_{j=1}^m \alpha_j/(\beta_j - x)$.

A detailed comparison of these possibilities with respect to numerical stability and computational cost is given by Lin [32]. The method that is found to be the best in the context of the algorithm to be developed in the next section is bottom-up evaluation of (4.1), which is summarized as follows.

ALGORITHM 4.1 (continued fraction, bottom-up). *This algorithm evaluates the continued fraction* (4.1) *in bottom-up fashion at the matrix* $X \in \mathbb{C}^{n\times n}$.

```
Y_{2m} = c_{2m} X
for j = 2m-1: -1: 1
    Solve (I + Y_{j+1}) Y_j = c_j X for Y_j
end
r_m = I + Y_1
```

We now investigate the numerical stability of this recurrence. Let $\|\cdot\|$ denote any p-norm, assume that $\|Y_j\| < 1$ for all j, and let $\widehat{Y}_j \equiv Y_j + \Delta Y_j$ denote the computed Y_j. The errors in obtaining Y_j from $(I+Y_{j+1})Y_j = c_j X$ result from forming the right-hand side and solving the system. We assume that the solver is stable, so that [19, sec. 9]

$$(I + \widehat{Y}_{j+1})\widehat{Y}_j = c_j X + F_j + R_j,$$

where $\|F_j\| \le u|c_j|\,\|X\|$ and $\|R_j\| \le \alpha_n u(1 + \|\widehat{Y}_{j+1}\|)\|\widehat{Y}_j\|$, for some constant α_n, where u is the unit roundoff. Then $(I + Y_{j+1})\Delta Y_j = F_j + R_j - \Delta Y_{j+1}Y_j + O(u^2)$, which implies

$$\|\Delta Y_j\| \le \frac{1}{1 - \|Y_{j+1}\|}\bigl(u|c_j|\,\|X\| + \alpha_n u(1 + \|Y_{j+1}\|)\,\|Y_j\| + \|Y_j\|\,\|\Delta Y_{j+1}\|\bigr) + O(u^2), \quad j = 2m-1:-1:1, \qquad \|\Delta Y_{2m}\| \le u|c_{2m}|\,\|X\|. \tag{4.2}$$

TABLE 4.1
Constants d in the bounds $\|\Delta Y_1\|/\|Y_1\| \le du + O(u^2)$ for different $\|X\|$ and p.

	p				
$\|X\|$	0.1	0.3	0.5	0.7	0.9
0.99	3.46e2	3.21e2	2.90e2	2.56e2	2.19e2
0.95	6.53e1	6.08e1	5.55e1	4.97e1	4.33e1
0.90	3.12e1	2.92e1	2.68e1	2.43e1	2.15e1
0.75	1.14e1	1.07e1	1.00e1	9.24e0	8.42e0
0.50	5.01e0	4.80e0	4.59e0	4.36e0	4.12e0
0.25	2.98e0	2.91e0	2.85e0	2.77e0	2.70e0
0.10	2.32e0	2.30e0	2.28e0	2.26e0	2.23e0

TABLE 4.2
Minimal values of m for which (4.4) holds.

	p				
$\|X\|$	0.1	0.3	0.5	0.7	0.9
0.99	88	100	100	84	79
0.95	38	39	39	39	36
0.90	27	27	27	27	26
0.75	16	16	16	16	15
0.50	9	10	10	10	10
0.25	6	6	7	7	6
0.10	5	5	5	5	5

We can bound $\|Y_j\|$ from the recurrence

$$\|Y_j\| \le \frac{|c_j|\|X\|}{1-\|Y_{j+1}\|}, \qquad j = 2m-1:-1:1, \qquad \|Y_{2m}\| = |c_{2m}|\|X\|. \tag{4.3}$$

Together, the recurrences (4.2) and (4.3) allow us to compute, to first order, a bound on $\|\Delta Y_1\|$ for any given $\|X\|$. An upper bound for the relative error can then be obtained by using $\|Y_1\| \ge |c_1|\|X\|/(1+\|Y_2\|)$ together with the upper bound for $\|Y_2\|$ from (4.3).

Table 4.1 shows the values of the bound for $\|\Delta Y_1\|/\|Y_1\|$ for a range of $p \in (0,1)$ and $\|X\| \in (0,1)$, with $\alpha_n \equiv 1$ (the bound scales roughly linearly with α_n). Here, the values of m, shown in Table 4.2, are chosen as the smaller of 100 and the minimal value for which

$$\|r_m(X) - (I-X)^p\| \le |(1-\|X\|)^p - r_m(\|X\|)| \le u, \tag{4.4}$$

with $u = 2^{-53} \approx 1.1 \times 10^{-16}$, where the first inequality always holds by Theorem 3.5. The assumption $\|Y_j\| < 1$ was found to be satisfied in every case. The results show that as long as we keep $\|X\|$ below 0.9, say, the numerical stability of Algorithm 4.1 will be excellent. In fact, in Algorithm 5.1 we will limit $\|X\|$ to about 0.3, for other reasons.

5. Schur–Padé algorithm for A^p. Now we develop an algorithm for computing A^p for a real $p \in (-1,1)$, where A has no nonpositive real eigenvalues. We can restrict p to $(-1,1)$ without loss of generality, since in general we can compute $A^p = A^{p_1}A^{p_2}$ with $p_1 \in (-1,1)$ and p_2 an integer. How best to choose p_1 and p_2 is considered in Section 6.

Our algorithm exploits the relation $A^p = (A^{1/2^k})^{p\cdot 2^k}$. We take square roots of A repeatedly until $A^{1/2^k}$ is close to the identity matrix. Then, with $X = I -$

TABLE 5.1
$\theta_m^{(p)}$, *for* $p = 1/2$ *and selected* m.

m	1	2	3	4	5	6	7	8	9
$\theta_m^{(1/2)}$	1.53e-5	2.25e-3	1.92e-2	6.08e-2	1.25e-1	2.03e-1	2.84e-1	3.63e-1	4.35e-1
m	10	11	12	13	14	15	16	32	64
$\theta_m^{(1/2)}$	4.99e-1	5.55e-1	6.05e-1	6.47e-1	6.84e-1	7.17e-1	7.44e-1	9.27e-1	9.81e-1

TABLE 5.2
Minimum values of $\theta_m^{(p)}$, *for* $p \in [-1, 1]$.

m	1	2	3	4	5	6	7	8	9
$\min_p \theta_m^{(p)}$	1.51e-5	2.24e-3	1.88e-2	6.04e-2	1.24e-1	2.00e-1	2.79e-1	3.55e-1	4.25e-1
m	10	11	12	13	14	15	16	32	64
$\min_p \theta_m^{(p)}$	4.87e-1	5.42e-1	5.90e-1	6.32e-1	6.69e-1	7.00e-1	7.28e-1	9.15e-1	9.76e-1

$A^{1/2^k}$, we can use the approximation $(A^{1/2^k})^p \approx r_m(X)$, where r_m is the $[m/m]$ Padé approximant to $(1-x)^p$. We recover an approximation to the pth power of the original matrix from $A^p \approx r_m(X)^{2^k}$. This approach is analogous to the inverse scaling and squaring method for the matrix logarithm [9], [21, sec. 11.5], [29]. In order to facilitate the computation of the square roots we compute an initial Schur decomposition $A = QTQ^*$, so that the problem is reduced to that for a triangular matrix.

For any $p \in [-1, 1]$ and m we denote by $\theta_m^{(p)}$ the largest value of $\|X\|$ such that the second inequality holds in (4.4). With $u = 2^{-53}$, we determined $\theta_m^{(p)}$ empirically in MATLAB, using high precision computations with the Symbolic Math Toolbox. For $p = 1/2$ and a range of $m \in [1, 64]$. Table 5.1 reports the results to three significant figures. To see how the values of $\theta_m^{(p)}$ vary with p for a specific m, we show in Figure 5.1 the values of $\theta_m^{(p)}$ corresponding to 324 different values of p between -0.999 and 0.999, for a range of m. Table 5.2 reports the corresponding minimum values of $\theta_m^{(p)}$ over $p \in [-1, 1]$. For each m, $\theta_m^{(p)}$ tends to 1 as p tends to -1, 0 or 1. Our results show, however, that the relative variation of $\theta_m^{(p)}$ with p is slight, except when p is within distance about 10^{-4} of -1, 0, or 1. We therefore base our algorithm on the values

$$\theta_m = \min_{p \in [-1,1]} \theta_m^{(p)}, \tag{5.1}$$

and do not optimize the algorithm parameters separately for each particular p.

In designing the algorithm we minimize the cost subject to achieving the desired accuracy, adapting a strategy used within the inverse scaling and squaring algorithm for the matrix logarithm in [9], [21, sec. 11.5]. Computing a square root of a triangular matrix T by the Schur method of Björck and Hammarling [7], [21, Alg. 6.3] costs $n^3/3$ flops, while evaluating $r_m(T)$ by Algorithm 4.1 costs $(2m-1)n^3/3$ flops. Bearing in mind the squaring phase, it is therefore worthwhile to compute an extra square root if it allows a reduction in the Padé degree m by more than 1. Considering that

$$\|I - T^{1/2}\| = \|(I + T^{1/2})^{-1}(I - T)\| \approx \tfrac{1}{2}\|I - T\| \tag{5.2}$$

once $T \approx I$ and that, from Table 5.2, $\theta_m/2 < \theta_{m-2}$ for $m > 7$, the cost of computing T^p when $\|I - T\| > \theta_7$ will be minimized if we take square roots of T repeatedly until

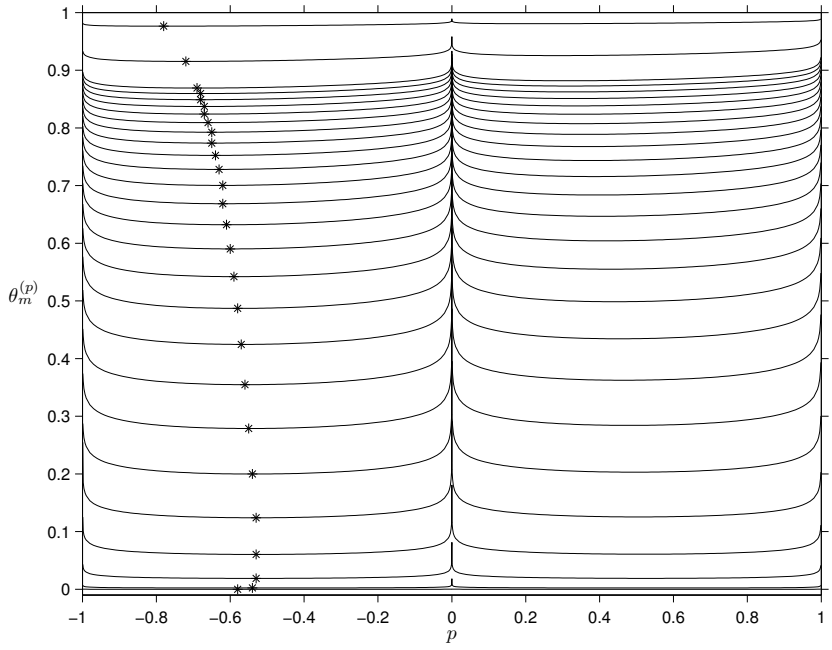

FIG. 5.1. $\theta_m^{(p)}$ *against* p*, for* $m = 1\colon 25, 32, 64$*;* $m = 1$ *is the lowest curve and* $m = 64$ *the highest curve.* θ_m *in* (5.1) *is marked as "*$*$*". The curves are not symmetric about* $p = 0$*.*

$\|I - T^{1/2^k}\| \le \theta_7$. Then it is worth taking one more square root if it reduces the required m by more than 1.

An important final ingredient of our algorithm is a special implementation of the squaring phase, obtained by adapting the approach suggested by Al-Mohy and Higham [1] for the matrix exponential. The squaring phase forms $r_m(I - T^{1/2^k})^{2^j} \approx T^{p/2^{k-j}}$, $j = 1\colon k$. But we can evaluate the diagonal and first superdiagonal elements of $T^{p/2^{k-j}}$ *exactly* from explicit formulae, and injecting these values into the recurrence should reduce the propagation of errors. The diagonal entries are computed in the obvious way. We now derive an appropriate formula for the first superdiagonal.

The (1,2) element of $F = \left[\begin{smallmatrix} \lambda_1 & t_{12} \\ 0 & \lambda_2 \end{smallmatrix}\right]^p$ is given by $f_{12} = t_{12}(\lambda_2^p - \lambda_1^p)/(\lambda_2 - \lambda_1)$ if $\lambda_1 \ne \lambda_2$, or $p\lambda_1^{p-1}t_{12}$ otherwise [21, sec. 4.6]. We need a way of evaluating the divided difference $(\lambda_2^p - \lambda_1^p)/(\lambda_2 - \lambda_1)$ accurately even when λ_1 and λ_2 are very close; this formula itself suffers from cancellation. We have

$$\begin{aligned}
\frac{\lambda_2^p - \lambda_1^p}{\lambda_2 - \lambda_1} &= \frac{\exp(p\log\lambda_2) - \exp(p\log\lambda_1)}{\lambda_2 - \lambda_1} \\
&= \exp\left(\tfrac{p}{2}\left(\log\lambda_2 + \log\lambda_1\right)\right) \frac{\exp\left(\tfrac{p}{2}\left(\log\lambda_2 - \log\lambda_1\right)\right) - \exp\left(\tfrac{p}{2}\left(\log\lambda_1 - \log\lambda_2\right)\right)}{\lambda_2 - \lambda_1} \\
&= \exp\left(\tfrac{p}{2}\left(\log\lambda_2 + \log\lambda_1\right)\right) \frac{2\sinh\left(\tfrac{p}{2}\left(\log\lambda_2 - \log\lambda_1\right)\right)}{\lambda_2 - \lambda_1}.
\end{aligned}$$

The remaining problem is to evaluate $w = \log\lambda_2 - \log\lambda_1$ accurately. To avoid cancellation we can rewrite [21, sec. 11.6.2]

$$w = \log\left(\frac{\lambda_2}{\lambda_1}\right) + 2\pi i\mathcal{U}(\log\lambda_2 - \log\lambda_1) = \log\left(\frac{1+z}{1-z}\right) + 2\pi i\mathcal{U}(\log\lambda_2 - \log\lambda_1),$$

where $z = (\lambda_2 - \lambda_1)/(\lambda_2 + \lambda_1)$ and $\mathcal{U}(z)$ is the unwinding number of $z \in \mathbb{C}$ defined by

$$\mathcal{U}(z) := \frac{z - \log(e^z)}{2\pi i} = \left\lceil \frac{\operatorname{Im} z - \pi}{2\pi} \right\rceil \in \mathbb{Z}. \tag{5.3}$$

Then, using the hyperbolic arc tangent $\operatorname{atanh}(z)$, defined by

$$\operatorname{atanh}(z) := \frac{1}{2} \log\left(\frac{1+z}{1-z}\right), \tag{5.4}$$

w can be expressed as

$$w = 2\operatorname{atanh}(z) + 2\pi i \mathcal{U}(\log\lambda_2 - \log\lambda_1).$$

Hence

$$f_{12} = t_{12} \exp\left(\tfrac{p}{2}\left(\log\lambda_2 + \log\lambda_1\right)\right) \frac{2\sinh\left(p\left(\operatorname{atanh}(z) + \pi i \mathcal{U}(\log\lambda_2 - \log\lambda_1)\right)\right)}{\lambda_2 - \lambda_1}. \tag{5.5}$$

Overall, we have the formula

$$f_{12} = \begin{cases} t_{12} p \lambda_1^{p-1}, & \lambda_1 = \lambda_2, \\ t_{12} \dfrac{\lambda_2^p - \lambda_1^p}{\lambda_2 - \lambda_1}, & |\lambda_1| < |\lambda_2|/2 \text{ or } |\lambda_2| < |\lambda_1|/2, \\ (5.5), & \text{otherwise}, \end{cases} \tag{5.6}$$

where we evaluate the usual divided difference if λ_1 and λ_2 are sufficiently far apart. We are assuming that accurate implementations of the scalar sinh and atanh functions are available. The definition (5.4) is that used in MATLAB; there is an alternative to (5.4) which necessitates modifications to (5.5) described in [21, sec. 11.6.2].

Now we state the overall algorithm.

ALGORITHM 5.1 (Schur–Padé algorithm). *Given $A \in \mathbb{C}^{n\times n}$ with no eigenvalues on $\mathbb{R}^-$ and a nonzero $p \in (-1, 1)$ this algorithm computes $X = A^p$ via a Schur decomposition and Padé approximation. It uses the constants $\theta_m := \min_p \theta_m^{(p)}$ in Table* 5.2. *The algorithm is intended for IEEE double precision arithmetic.*

```
1  Compute a (complex) Schur decomposition A = QTQ*.
2  If T is diagonal, X = QT^pQ*, quit, end
3  T_0 = T
4  k = 0, q = 0
5  while true
6        τ = ||T − I||_1
7        if τ ≤ θ_7
8          q = q + 1
9          j_1 = min{ i: τ ≤ θ_i, i = 3:7 }
10         j_2 = min{ i: τ/2 ≤ θ_i, i = 3:7 }
11         if j_1 − j_2 ≤ 1 or q = 2, m = j_1, goto line 16, end
12       end
13       T ← T^{1/2} using the Schur method [21, Alg. 6.3].
14       k = k + 1
15 end
16 Evaluate U = r_m(I − T) using Algorithm 4.1.
```

17 for $i = k\colon -1\colon 0$
18 if $i < k$, $U \leftarrow U^2$, end
19 Replace $\mathrm{diag}(U)$ by $\mathrm{diag}(T_0)^{p/2^i}$.
20 Replace first superdiagonal of U by first superdiagonal of $T_0^{p/2^i}$ obtained from (5.6) with $p \leftarrow p/2^i$.
21 end
22 $X = QUQ^*$

Cost: $25n^3$ flops for the Schur decomposition plus $(2k + 2m - 1)n^3/3$ flops for U and $3n^3$ to get X: about $(28 + (2k + 2m - 1)/3)n^3$ flops in total.

Note that line 2 simply computes T^p in the obvious way when T is diagonal, that is, when A is normal; there is no need for Padé approximation in this case.

If A is real, we could take the real Schur decomposition at line 1, and compute the square roots of the now quasitriangular T at line 13 using the real Schur method [17], [21, Alg. 6.7]. This would guarantee a real computed $\widehat{X}$ and could be faster due to the avoidance of complex arithmetic.

6. General $p \in \mathbb{R}$. In developing the Schur–Padé algorithm we assumed $p \in (-1, 1)$. For a general noninteger $p \in \mathbb{R}$ there are two ways to reduce the power to the interval $(-1, 1)$. We can write

$$p = \lfloor p \rfloor + p_1, \qquad p_1 > 0, \tag{6.1a}$$

$$p = \lceil p \rceil + p_2, \qquad p_2 < 0, \tag{6.1b}$$

where $p_1 - p_2 = 1$. To choose between these two possibilities we will concentrate on the computation of A^{p_1} and A^{p_2} and ask which of these computations is the better conditioned. To make the analysis tractable we assume that A is Hermitian positive definite with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n > 0$ and we use the lower bound (2.8), which is now an equality for the Frobenius norm. Using the mean value theorem, we obtain, for $p \in (-1, 1)$ and $f(x) = x^p$,

$$\begin{aligned} \|L_{x^p}(A)\|_F &= \max_{i \leq j} |f[\lambda_i, \lambda_j]| = \max_{i \leq j} |f'(\xi_{ij})|, \qquad \xi_{ij} \in [\lambda_i, \lambda_j] \\ &= |f'(\lambda_n)| = |p|\lambda_n^{p-1}. \end{aligned}$$

Hence, by (2.1) for the Frobenius norm,

$$\kappa_{x^p}(A) = \frac{|p|\lambda_n^{p-1}\|A\|_F}{\|A^p\|_F} \approx \frac{|p|\lambda_n^{p-1}\|A\|_2}{\|A^p\|_2} = \begin{cases} |p|\kappa_2(A)^{1-p}, & p \geq 0, \\ |p|\kappa_2(A), & p \leq 0, \end{cases}$$

where $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \lambda_1/\lambda_n$. Since $p_1 > 0$ and $p_2 < 0$, in order to minimize the lower bound we should choose p_1 if $p_1\kappa_2(A)^{1-p_1} \leq -p_2\kappa_2(A) = (1 - p_1)\kappa_2(A)$, that is, if $\kappa_2(A) \geq \exp(p_1^{-1}\log\bigl(p_1/(1-p_1)\bigr)$. Thus, for example, if $p_1 \leq 0.5$ then p_1 is always chosen, while if $p_1 = 0.75$ or $p_1 = 0.99$ then p_1 is chosen for $\kappa_2(A) \geq 4.3$ and $\kappa_2(A) \geq 103.7$, respectively.

Now we consider how to handle integer p. When p is positive, A^p should be computed by binary powering [21, Alg. 4.1]. When p is negative there are several possibilities, of which we state three. We write GEPP for Gaussian elimination with partial pivoting.

ALGORITHM 6.1. *This algorithm computes $X = A^p$ for $p = -k \in \mathbb{Z}^-$.*
1 $Y = A^k$ by binary powering
2 $X = Y^{-1}$ via GEPP

ALGORITHM 6.2. *This algorithm computes $X = A^p$ for $p = -k \in \mathbb{Z}^-$.*

1 $Y = A^{-1}$ via GEPP
2 $X = Y^k$ by binary powering

ALGORITHM 6.3. *This algorithm computes $X = A^p$ for $p = -k \in \mathbb{Z}^-$.*

1 Compute a factorization $PA = LU$ by GEPP.
2 $X_0 = I$
3 for $i = 0\colon k-1$
4 Solve $LX_{i+1/2} = PX_i$
5 Solve $UX_{i+1} = X_{i+1/2}$
6 end
7 $X = X_k$

Algorithms 6.1 and 6.2 have the same cost. Algorithm 6.3 is more expensive as it does not take advantage of binary powering. However, our main interest is in accuracy. Algorithm 6.1 inverts A^k, which is potentially a much more ill conditioned matrix than A. Intuitively, Algorithm 6.2 should therefore be preferred. Algorithm 6.3 does not explicitly invert a matrix but relies on triangular solves, and triangular systems are typically solved to higher accuracy than we might expect from conditioning considerations [19, chap. 8]. Rounding error analysis for these three algorithms yields forward error bounds whose respective sizes are difficult to compare [32]. Therefore we will use numerical experiments to guide our choice (see Experiment 7 in Section 9).

7. Singular matrices. Since our aim is to develop an algorithm of the widest possible applicability, we would like to extend Algorithm 5.1 so that it handles singular matrices with a semisimple zero eigenvalue. If A is singular then the Schur factor T will be singular. We reorder T (using unitary similarities) so that it has the form

$$T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \tag{7.1}$$

where T_{11} is nonsingular and T_{22} has zero diagonal. The zero eigenvalue is semisimple if and only if $T_{22} = 0$, by rank considerations. If $T_{22} = 0$ then $U = T^p$ is given by

$$U = \begin{bmatrix} U_{11} & T_{11}^{-1}U_{11}T_{12} \\ 0 & 0 \end{bmatrix}, \qquad U_{11} = T_{11}^p. \tag{7.2}$$

The diagonal blocks in this expression follow from the fact that any primary matrix function of a block triangular matrix is block triangular [21, Thm. 1.13], while the (1,2) block is obtained from the equation $TU = UT$. The conclusion is that we should obtain U_{11} from Algorithm 5.1 and compute U_{12} separately from the given formula.

In floating point arithmetic we are unlikely to obtain exact zeros on the diagonal of T. Consider, for example, the MATLAB matrix `A = gallery(5)`, which has integer entries and a Jordan form with one 5×5 Jordan block corresponding to the eigenvalue 0. The computed triangular Schur factor T has positive diagonal entries all of order 10^{-2}. The computed square root (for example) from Algorithm 5.1 has norm of order 10^{10}. Without further computations involving "difficult rank decisions" [12, sec. 7.6.5], which would effectively be the first stages of computing the Jordan form, it is not possible to determine whether it makes sense to compute A^p with $p \notin \mathbb{Z}$ when A is singular. We will therefore not pursue the development of a practical algorithm for the singular case.

```
function X = powerm(A,p,str)
%POWERM    Arbitrary power of matrix.
%   POWERM(A,p) computes the p'th power of A for a nonsingular,
%   diagonalizable matrix A and an arbitrary real number p.
%   POWERM(A,p,'nobalance') performs the computation with balancing
%   disabled in the underlying eigendecomposition.

if nargin == 3 && strcmp(str,'nobalance')
   [V,D] = eig(A,'nobalance');
else
   [V,D] = eig(A);
end
X = V*diag(diag(D).^p)/V;
```

FIG. 8.1. *MATLAB function* `powerm`.

8. Alternative algorithms. A number of alternatives to and variations of Algorithm 5.1 can be formulated. They are based on initial reduction to Schur form, the exp-log formula (1.2), and the Schur–Parlett algorithm of Davies and Higham [10], [21, Alg. 9.6]. The Schur–Parlett algorithm is designed for computing $f(A)$ for any f for which functions of arbitrary triangular matrices can be reliably computed. It employs a reordered and partitioned Schur triangular factor, computes $f(T_{ii})$ for the diagonal blocks T_{ii} by the given method and obtains the off-diagonal blocks by the block Parlett recurrence.

We summarize the main possibilities.

1. `Schur-Pade`: Algorithm 5.1.
2. `SP-Pade`: the Schur–Parlett method using Algorithm 5.1 on the diagonal blocks T_{ii}.
3. `SP-ss-iss`: the Schur–Parlett method with evaluation of $\exp(p\log(T_{ii}))$ by the inverse scaling and squaring method for the logarithm [21, sec. 11.5] and the scaling and squaring method for the exponential [1].
4. `tri-ss-iss`: reduction to Schur form T with evaluation of $\exp(p\log(T))$ by the inverse scaling and squaring method for the logarithm applied to the whole matrix T and the scaling and squaring method for the exponential.
5. `powerm`: the algorithm discussed in Section 1 based on an eigendecomposition, which is implemented in the MATLAB function of Figure 8.1.

Note that a variant of `tri-ss-iss` that works directly on A instead of reducing to Schur form is not competitive in cost with `tri-ss-iss`, since computing square roots of full matrices is relatively expensive [21, Chap. 6].

We make some brief comments on the relative merits of these methods.

For the methods that employ a Schur decomposition the cost will be dominated by the cost of computing the Schur decomposition unless $\|A\|$ is large. If the matrix is already triangular then `Schur-Pade` and `tri-ss-iss` have similar cost, and in particular require approximately the same number of square roots.

`SP-Pade` differs from `Schur-Pade` in that it applies Padé approximation to each diagonal block of T (possibly with a different degree for each block) rather than to T as a whole. It is possible for the partitioning to be the trivial one, $T \equiv T_{11}$, in which case `SP-Pade` and `Schur-Pade` are identical.

An advantage in cost of `SP-Pade` and `SP-ss-iss` over `Schur-Pade` is that large elements of T do not affect the number of square roots computed, and hence the cost,

as long as they lie in the superdiagonal blocks T_{ij} of the Schur–Parlett partitioning of T.

In the next section we compare these methods numerically.

9. Numerical experiments. Our numerical experiments were carried out in MATLAB R2010a, for which the unit roundoff $u = 2^{-53} \approx 1.1 \times 10^{-16}$. Our implementations of `SP-Pade` and `SP-ss-iss` are obtained by modifying the MATLAB function `funm`. For all methods except `powerm` we evaluate powers of 2×2 triangular matrices directly, using the formula (5.6).

Relative errors are measured in the Frobenius norm. For the "exact" solution we take the matrix computed using `powerm` at 100 digit precision with the VPA arithmetic of the Symbolic Math Toolbox; thus we can compute relative errors only when A is diagonalizable.

When $q = 1/p$ is an integer, another measure of the quality of a computed solution X is its relative residual,

$$\rho(X) = \frac{\|A - X^q\|}{\|X\|\,\eta(X)},$$

where $\eta(X) = \left\|\sum_{i=0}^{q-1} \left(X^{q-1-i}\right)^T \otimes X^i\right\|$ if $p > 0$ and $\eta(X) = \left\|\sum_{i=1}^{-q} \left(X^{-i}\right)^T \otimes X^{i+q-1}\right\|$ if $p < 0$, with $\otimes$ denoting the Kronecker product. This is a more practically useful definition of relative residual than $\|A - X^q\|/\|X^q\|$, as explained in [14], [21, Prob. 7.16].

Experiment 1. We computed the pth power of the matrix

$$A(\epsilon) = \begin{bmatrix} 1 & 1 \\ 0 & 1+\epsilon \end{bmatrix}, \tag{9.1}$$

for $p \in \{0.1, 0.5, 0.9\}$ and $\epsilon = 10^{-t}$ with 65 equally spaced values of $t \in [0, 16]$. The condition number $\kappa_{x^p}(A(\epsilon))$ is of order 1 for all these ϵ and p. The relative errors for `powerm` are shown in Figure 9.1. Clearly, the errors deteriorate as t increases and $A(\epsilon)$ approaches a defective matrix; the reason for the "bifurcation" in the error curves is not clear. The other methods defined in Section 8 all produce results with relative error less than $4u$ in all cases.

Experiment 2. In this experiment we formed 50 random 50×50 matrices with elements from the normal (0,1) distribution; any matrix with an eigenvalue on $\mathbb{R}^-$ was discarded and another random matrix generated. Then we reduced A to Hessenberg form using the MATLAB function `hess` and computed $A^{1/3}$ by all five methods as well as by `powerm_nb`, the latter denoting `powerm` with the `'nobalance'` argument, which inhibits the use of balancing in the eigendecomposition. The results, with 2-norms used in the residuals, are shown in Figure 9.2. The improved performance of `powerm_nb` over `powerm` shows that it is the balancing that is affecting the numerical stability of `powerm` in this example. This is not surprising, because Watkins [36] has pointed out that for upper Hessenberg matrices balancing can seriously degrade accuracy in the eigendecomposition and should not be automatically used.

We note that using `powerm_nb` in place of `powerm` makes no difference to the results in Experiment 1, as balancing has no effect in that example.

Experiment 3. In this experiment we use a selection of 10×10 nonsingular matrices taken from the MATLAB `gallery` function and from the Matrix Computation Toolbox [15]. Any matrix found to have an eigenvalue on $\mathbb{R}^-$ was squared. We computed A^p for $p \in \{1/52,\ 1/12,\ 1/3,\ 1/2\}$, these values being ones likely to occur in applications where roots of transition matrices are required [21, sec. 2.3], [24], as

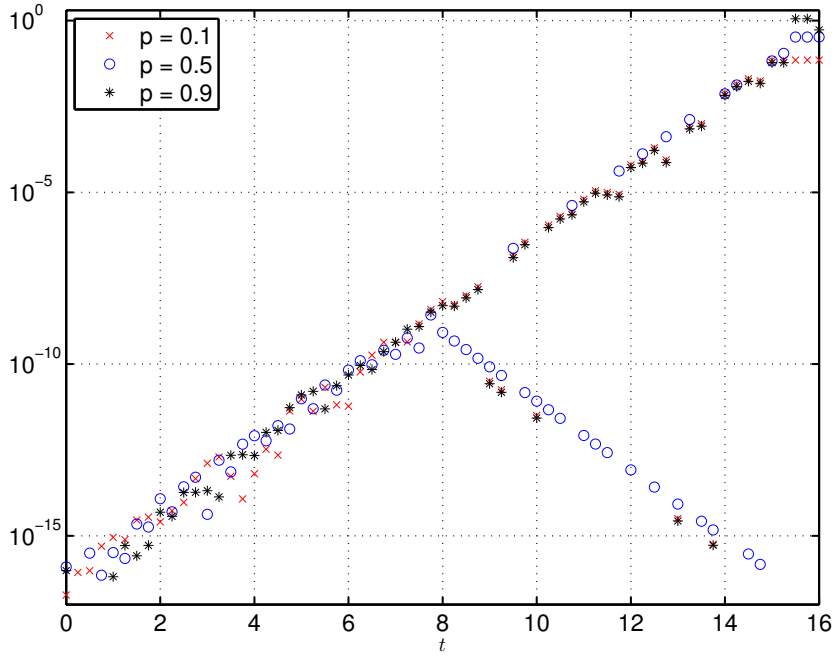

FIG. 9.1. *Experiment* 1*: relative errors for* `powerm` *on matrix* (9.1) *with* $\epsilon = 10^{-t}$.

well as the negatives of these values. This gives 376 problems in total. We omit `tri-ss-iss` from this test, as it is generally outperformed by `SP-ss-iss` (as can be seen in Experiment 2). Figures 9.3 and 9.5 show the relative errors and relative residuals. The solid line in Figure 9.3 is $\kappa_{x^p}(A)u$, where κ_{x^p} is computed via (2.3) and (2.5) using codes from the Matrix Function Toolbox [16] that compute $K_{\exp}$ and $K_{\log}$; the problems are sorted by decreasing condition number. Figures 9.4 and 9.6 show performance profiles. A performance profile shows the proportion π of problems where the performance ratio of a method is at least α, where the performance ratio for a method on a problem is the error or residual of that method divided by the smallest error or residual over all the methods. The errors and residuals lead to the same conclusions. First, `powerm` often produces very good results but is sometimes very unstable. Second, `Schur-Pade` `SP-Pade` and `SP-ss-iss` perform similarly, with `Schur-Pade` having a slight edge overall.

Experiment 4. This experiment is identical to the previous one except that we use the upper triangular QR factor R of each matrix and replace every negative diagonal element of R by its absolute value. The errors and residuals and their performance profiles are shown in Figures 9.7–9.10. For this class of matrices `Schur-Pade` is clearly greatly superior to the other methods. The performance profiles are qualitatively similar if we use the Schur factor instead of the QR factor.

Experiment 5. In this experiment we compute the three bounds in (2.7), (2.8) as well as the true norm of the Fréchet derivative $\|L_{x^p}(A)\|$ for the same matrices and values of p as in Experiment 3, using the Frobenius norm. The computed upper bound, which sometimes overflowed, was set to the minimum of 10^{30} and itself. The results are plotted in Figure 9.11. The results show that the lower bounds are sharper than the upper bounds and that they are often correct to within a couple of orders of

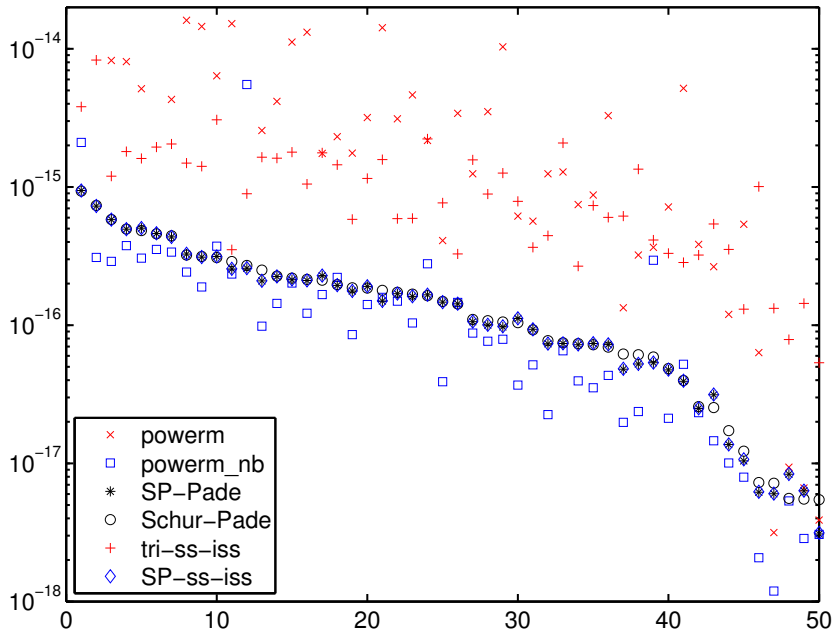

FIG. 9.2. *Experiment 2: relative residuals for 50 random Hessenberg matrices.*

magnitude, being less reliable for the very ill conditioned problems.

Experiment 6. In this experiment, we test our proposed choice of the fractional part of p when $p \notin [-1, 1]$. For $\kappa_2(A)$ we use the lower bound $\max_i |t_{ii}| / \min_i |t_{ii}|$ in the prescription of Section 6, where T is the triangular Schur factor. We use the same matrices as in Experiment 3 and compute A^p for $p = 3.9, 3.7, 3.3, 3.1$. The performance profiles of the relative errors are shown in Figure 9.12. Our strategy chose p_1 in 169 of the 197 cases in this experiment. Indeed, p_1 is almost as good a choice as the "optimal" choice, as can be seen in two ways. First, the performance profile curve for p_1 is almost indistinguishable from that for the "optimal" choice and so is omitted from the figure. Second, the maximum and minimum values of the relative error for p_1 divided by that for p_2 were 3.2 and 1.3×10^{-16}, respectively.

Experiment 7. In this final experiment we compare Algorithms 6.1, 6.2, and 6.3, all of which compute A^p where $p = -k$ is a negative integer. We test the algorithms on the same set of matrices as in Experiment 3 for $p = -3, -5, -7, -9$. The results are shown in Figures 9.13 and 9.14. Algorithms 6.2 and 6.3 clearly produce much more accurate results than Algorithm 6.1, as we expected. There is little to choose between Algorithms 6.2 and 6.3; we favour the former in view of its lower computational cost.

10. Concluding remarks. We have derived a new algorithm (Algorithm 5.1) for computing arbitrary powers A^p of a matrix, based on diagonal Padé approximants of $(1 - x)^p$ and the Schur decomposition. The algorithm performs in a generally numerically stable fashion in our tests, with relative error usually less than the product of the condition number of the problem and the unit roundoff. Our experiments demonstrate the superiority of this approach over alternatives based on separate approximation of the exponential and logarithm in the formula $A^p = \exp(p \log(A))$ using the best available methods. The use of Algorithm 5.1 within the Schur–Parlett

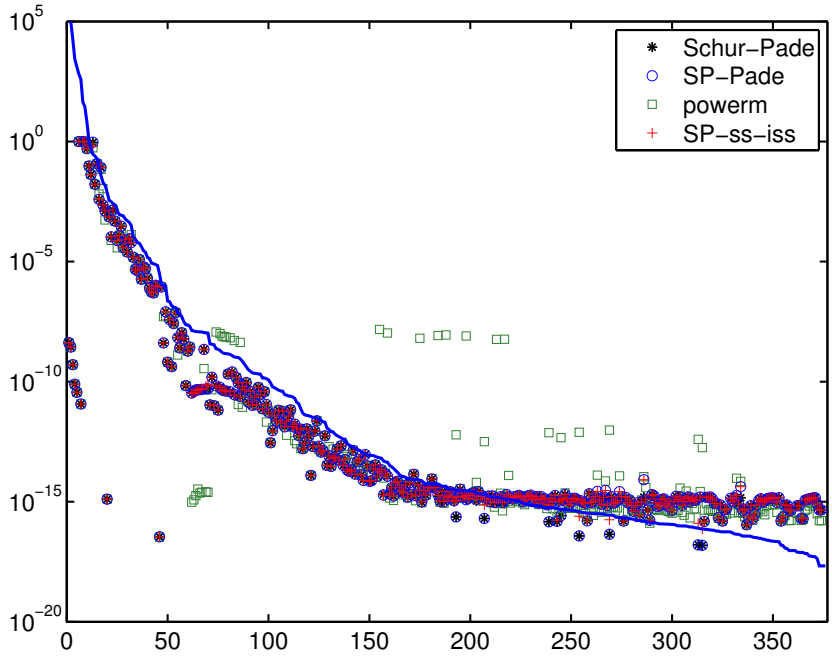

FIG. 9.3. *Experiment 3: relative errors for a selection of* 10×10 *matrices and several* p.

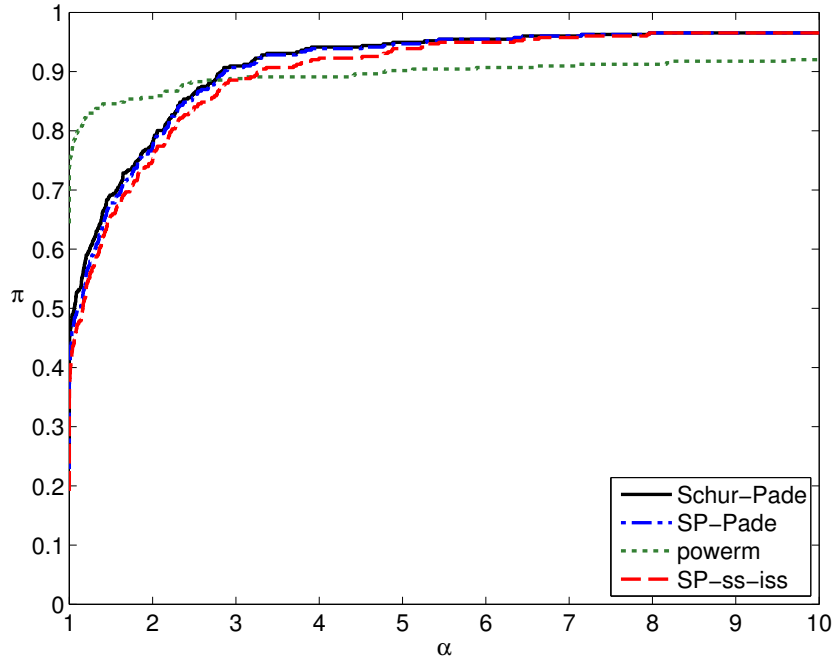

FIG. 9.4. *Experiment 3: performance profile of relative errors.*

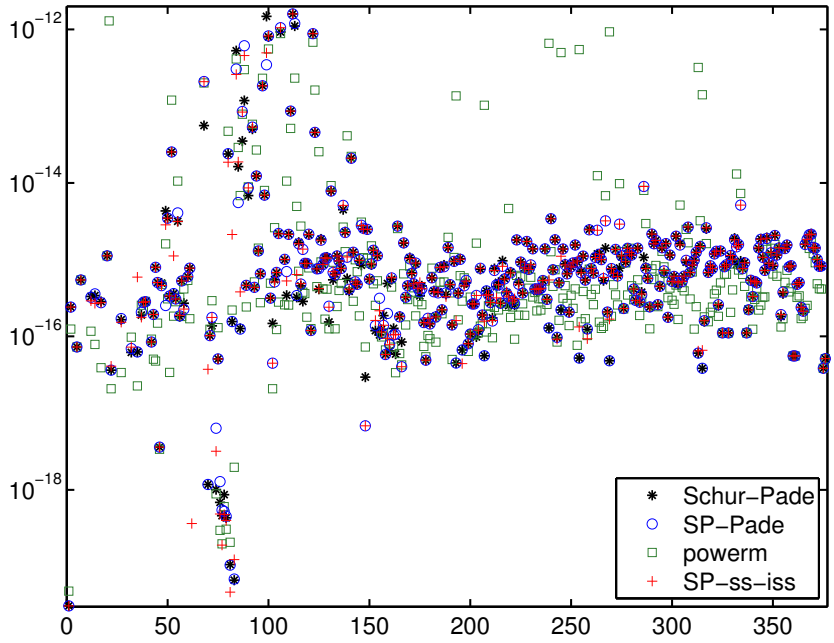

FIG. 9.5. *Experiment 3: relative residuals for a selection of* 10×10 *matrices and several* p.

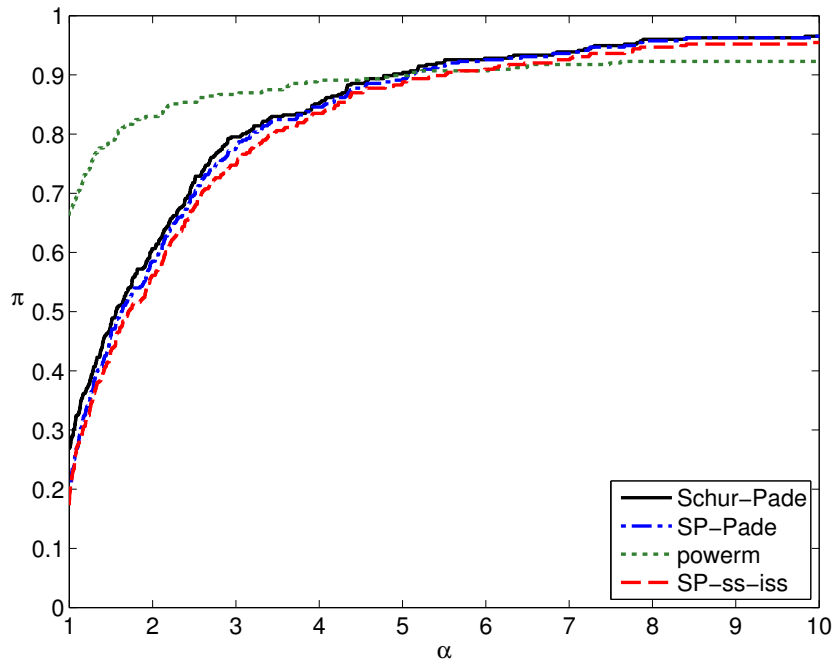

FIG. 9.6. *Experiment 3: performance profile of relative residuals.*

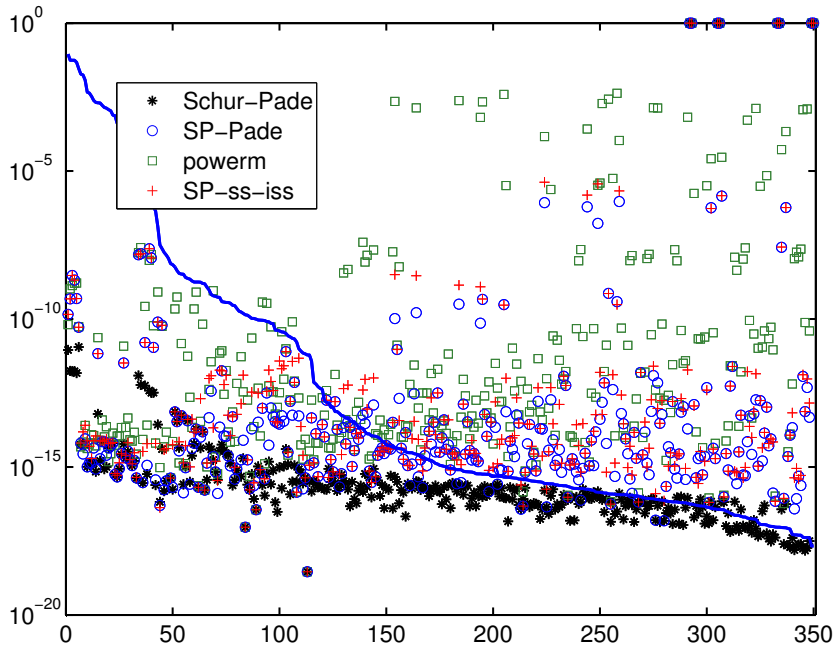

FIG. 9.7. *Experiment 4: relative errors for a selection of* 10×10 *triangular matrices and several* p.

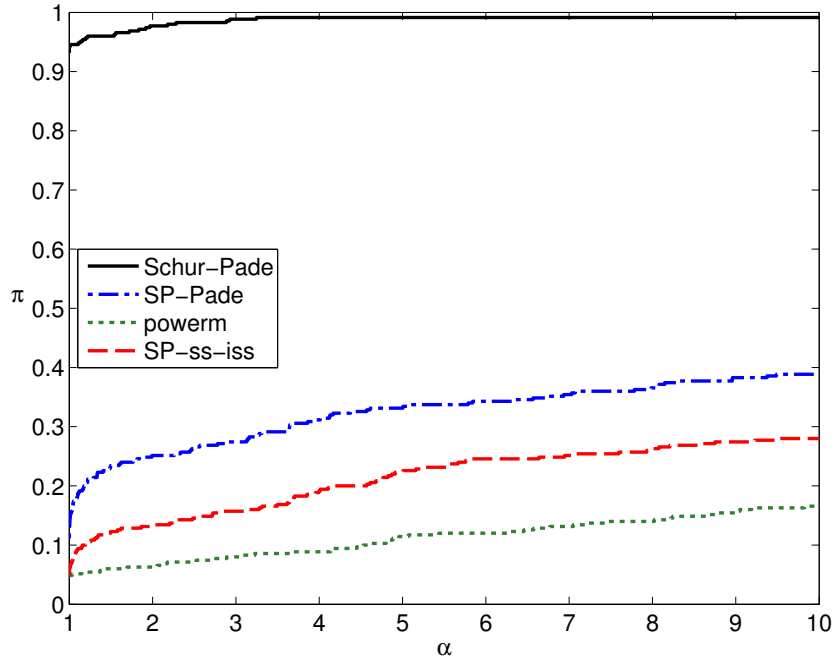

FIG. 9.8. *Experiment 4: performance profile of relative errors.*

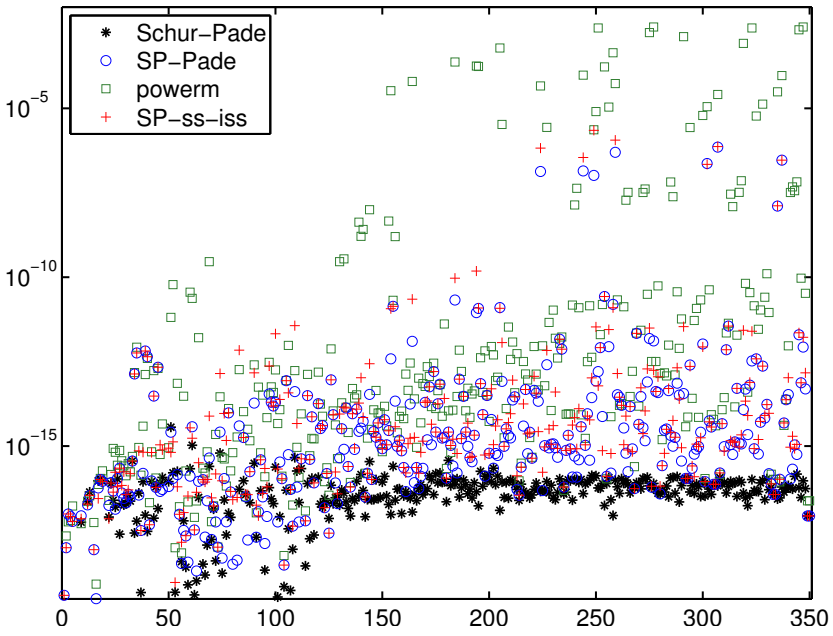

FIG. 9.9. *Experiment 4: relative residuals for a selection of* 10×10 *triangular matrices and several* p.

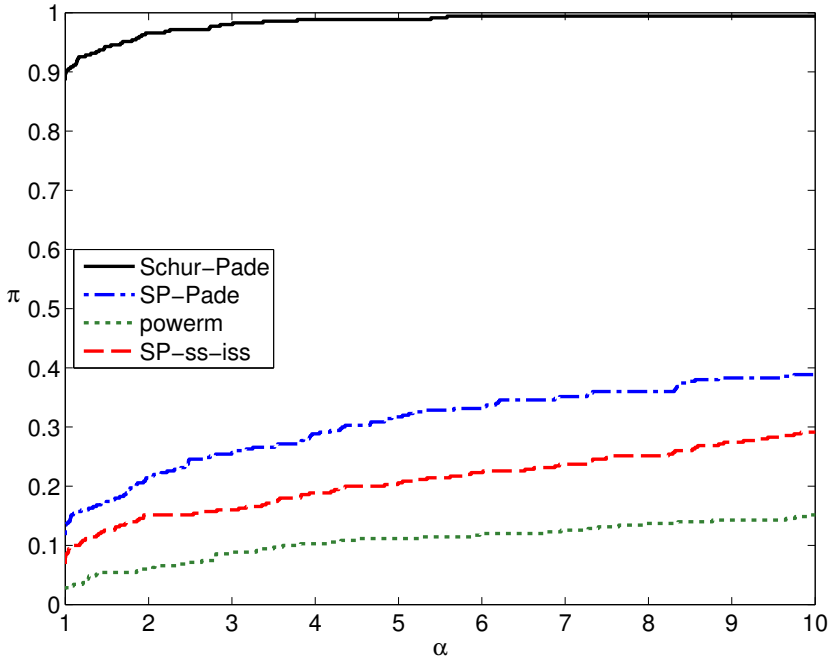

FIG. 9.10. *Experiment 4: performance profile of relative residuals.*

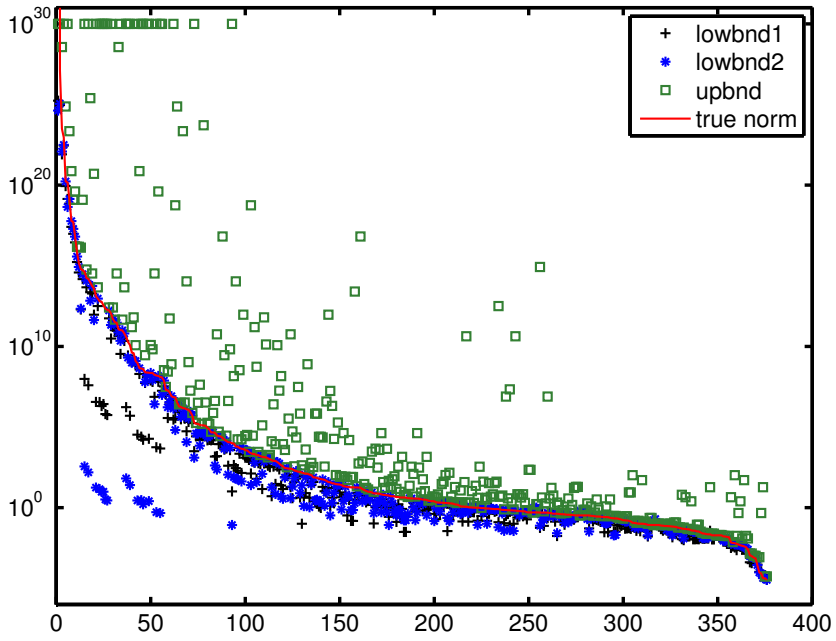

FIG. 9.11. *Experiment 5: the lower bounds* `lowbnd1` *in* (2.7) *and* `lowbnd2` *in* (2.8)*, the upper bound* `upbnd` *in* (2.8)*, and the true norm* $\|L_{x^p}(A)\|_F$*, for the matrices in Experiment* 3.

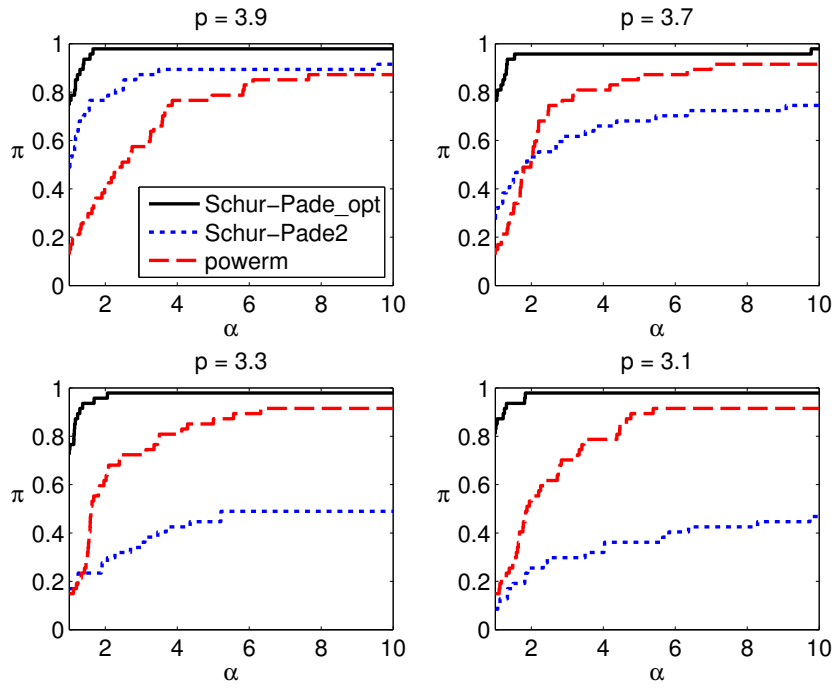

FIG. 9.12. *Experiment 6: performance profile of relative errors. The legend for first plot applies to all four plots.* `Schur-Pade2` *uses* p_2 *in* (6.1b) *and* `Schur-Pade_opt` *uses the choice defined in Section* 6.

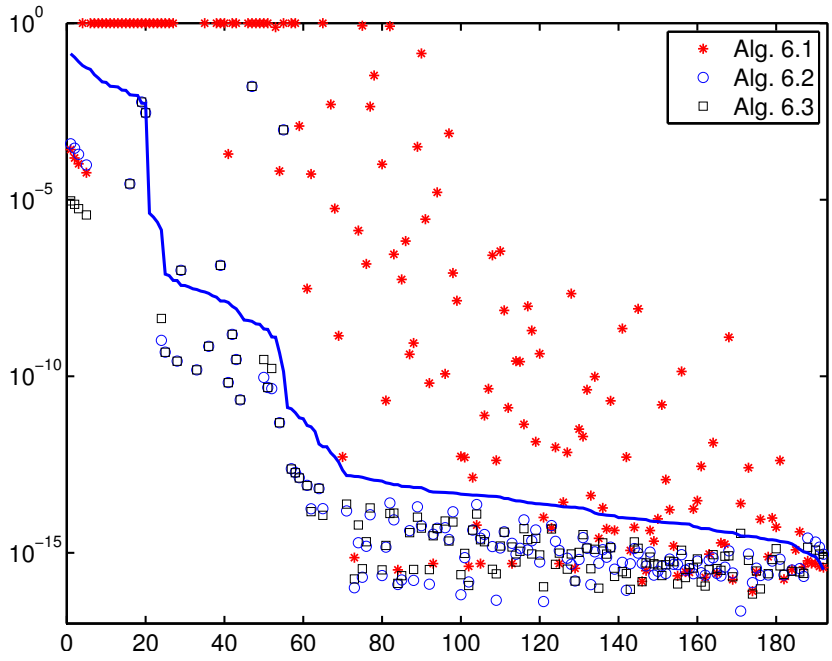

FIG. 9.13. *Experiment 7: relative errors for Algorithms* 6.1, 6.2, *and* 6.3 *for a selection of* 10×10 *matrices and several negative integers* p.

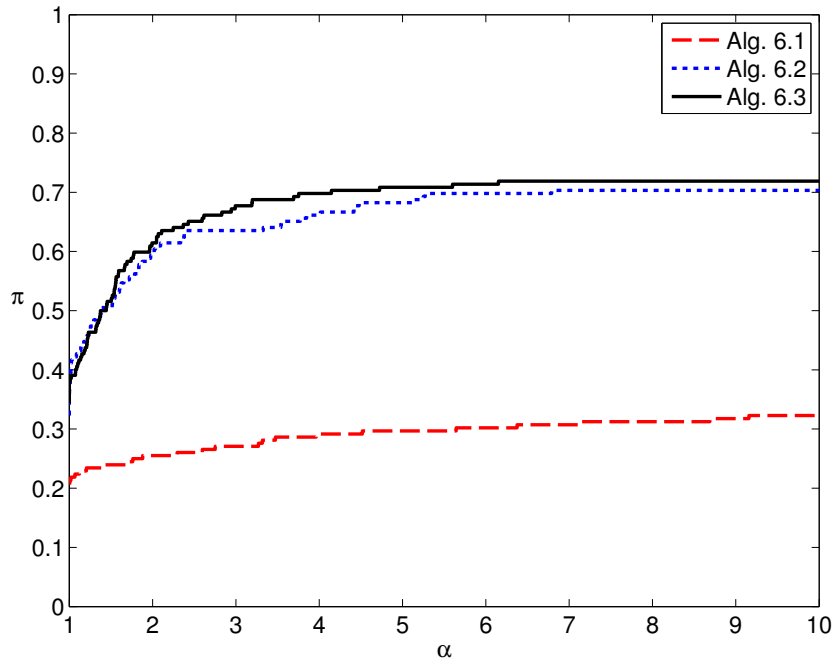

FIG. 9.14. *Experiment 7: performance profile of relative errors .*

algorithm (to compute T_{ii}^p for the diagonal blocks T_{ii} of the blocked and re-ordered triangular Schur factor) merits consideration as it is generally faster than applying it to the whole T, but Algorithm 5.1 is significantly more accurate in our tests with triangular matrices (Experiment 4).

MATLAB has a built-in function `mpower` for which the function call `mpower(A,p)` is equivalent to the syntax `A^p`. In our tests with MATLAB R2010a, `mpower` performs identically to our `powerm` function for noninteger p, and in particular performs badly on matrices that are defective or nearly defective. For negative integer p, `mpower` performs identically to Algorithm 6.1 in our tests.

REFERENCES

[1] Awad H. Al-Mohy and Nicholas J. Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.*, 31(3):970–989, 2009.

[2] George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*. Cambridge University Press, Cambridge, UK, 1999. xvi+664 pp. ISBN 0-521-78988-5.

[3] Mario Arioli and Daniel Loghin. Discrete interpolation norms with applications. *SIAM J. Numer. Anal.*, 47(4):2924–2951, 2009.

[4] George A. Baker, Jr. *Essentials of Padé Approximants*. Academic Press, New York, 1975. xi+306 pp.

[5] George A. Baker, Jr. and Peter Graves-Morris. *Padé Approximants*, volume 59 of *Encyclopedia of Mathematics and Its Applications*. Second edition, Cambridge University Press, Cambridge, UK, 1996. xiv+746 pp.

[6] Dario A. Bini, Nicholas J. Higham, and Beatrice Meini. Algorithms for the matrix pth root. *Numer. Algorithms*, 39(4):349–378, 2005.

[7] Åke Björck and Sven Hammarling. A Schur method for the square root of a matrix. *Linear Algebra Appl.*, 52/53:127–140, 1983.

[8] Theodore Charitos, Peter R. de Waal, and Linda C. van der Gaag. Computing short-interval transition matrices of a discrete-time Markov chain from partially observed data. *Statistics in Medicine*, 27:905–921, 2008.

[9] Sheung Hun Cheng, Nicholas J. Higham, Charles S. Kenney, and Alan J. Laub. Approximating the logarithm of a matrix to specified accuracy. *SIAM J. Matrix Anal. Appl.*, 22(4):1112–1125, 2001.

[10] Philip I. Davies and Nicholas J. Higham. A Schur–Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.*, 25(2):464–485, 2003.

[11] Simone Fiori. Leap-frog-type learning algorithms over the Lie group of unitary matrices. *Neurocomputing*, 71:2224–2244, 2008.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Third edition, Johns Hopkins University Press, Baltimore, MD, USA, 1996. xxvii+694 pp. ISBN 0-8018-5413-X (hardback), 0-8018-5414-8 (paperback).

[13] Federico Greco and Bruno Iannazzo. A binary powering algorithm for computing primary matrix roots. *Numer. Algorithms*, 55(1):59–78, 2010.

[14] Chun-Hua Guo and Nicholas J. Higham. A Schur–Newton method for the matrix pth root and its inverse. *SIAM J. Matrix Anal. Appl.*, 28(3):788–804, 2006.

[15] Nicholas J. Higham. The Matrix Computation Toolbox. `http://www.ma.man.ac.uk/~higham/mctoolbox`.

[16] Nicholas J. Higham. The Matrix Function Toolbox. `http://www.ma.man.ac.uk/~higham/mftoolbox`.

[17] Nicholas J. Higham. Computing real square roots of a real matrix. *Linear Algebra Appl.*, 88/89: 405–430, 1987.

[18] Nicholas J. Higham. Evaluating Padé approximants of the matrix logarithm. *SIAM J. Matrix Anal. Appl.*, 22(4):1126–1135, 2001.

[19] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. xxx+680 pp. ISBN 0-89871-521-0.

[20] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM J. Matrix Anal. Appl.*, 26(4):1179–1193, 2005.

[21] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-

46-7.
[22] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Rev.*, 51(4):747–764, 2009.
[23] Nicholas J. Higham and Awad H. Al-Mohy. Computing matrix functions. *Acta Numerica*, 19: 159–208, 2010.
[24] Nicholas J. Higham and Lijing Lin. On pth roots of stochastic matrices. MIMS EPrint 2009.21, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, March 2009. 19 pp. Revised January 2010. Linear Algebra Appl., DOI: 10.1016/j.laa.2010.04.007.
[25] Bruno Iannazzo. On the Newton method for the matrix Pth root. *SIAM J. Matrix Anal. Appl.*, 28(2):503–523, 2006.
[26] Bruno Iannazzo. A family of rational iterations and its application to the computation of the matrix pth root. *SIAM J. Matrix Anal. Appl.*, 30(4):1445–1462, 2009.
[27] M. Ilić, I. W. Turner, and D. P. Simpson. A restarted Lanczos approximation to functions of a symmetric matrix. *IMA J. Numer. Anal.*, 30:1044–1061, 2010.
[28] Robert B. Israel, Jeffrey S. Rosenthal, and Jason Z. Wei. Finding generators for Markov chains via empirical transition matrices, with applications to credit ratings. *Math. Finance*, 11 (2):245–265, 2001.
[29] Charles S. Kenney and Alan J. Laub. Condition estimates for matrix functions. *SIAM J. Matrix Anal. Appl.*, 10(2):191–209, 1989.
[30] Charles S. Kenney and Alan J. Laub. Padé error estimates for the logarithm of a matrix. *Internat. J. Control*, 50(3):707–730, 1989.
[31] Beata Laszkiewicz and Krystyna Ziętak. A Padé family of iterations for the matrix sector function and the matrix pth root. *Numer. Linear Algebra Appl.*, 16:951–970, 2009.
[32] Lijing Lin. *Roots of Stochastic Matrices and Fractional Matrix Powers*. PhD thesis, The University of Manchester, Manchester, UK, 2010. In preparation.
[33] Beresford N. Parlett. A recurrence among the elements of functions of triangular matrices. *Linear Algebra Appl.*, 14:117–121, 1976.
[34] Michael S. Paterson and Larry J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.*, 2(1):60–66, 1973.
[35] Matthew I. Smith. A Schur algorithm for computing matrix pth roots. *SIAM J. Matrix Anal. Appl.*, 24(4):971–989, 2003.
[36] David S. Watkins. A case where balancing is harmful. *Electron. Trans. Numer. Anal.*, 23:1–4, 2006.