An Improved Schur–Pade Algorithm for Fractional Powers of a Matrix and their Frechet Derivatives

Higham, Nicholas J. and Lin, Lijing

2013

MIMS EPrint: **2013.1**

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

AN IMPROVED SCHUR–PADÉ ALGORITHM FOR FRACTIONAL POWERS OF A MATRIX AND THEIR FRÉCHET DERIVATIVES*

NICHOLAS J. HIGHAM† AND LIJING LIN†

Abstract. The Schur–Padé algorithm [N. J. Higham and L. Lin, *A Schur–Padé algorithm for fractional powers of a matrix*, SIAM J. Matrix Anal. Appl., 32(3):1056–1078, 2011] computes arbitrary real powers A^t of a matrix $A \in \mathbb{C}^{n\times n}$ using the building blocks of Schur decomposition, matrix square roots, and Padé approximants. We improve the algorithm by basing the underlying error analysis on the quantities $\|(I-A)^k\|^{1/k}$, for several small k, instead of $\|I-A\|$. We extend the algorithm so that it computes along with A^t one or more Fréchet derivatives, with reuse of information when more than one Fréchet derivative is required, as is the case in condition number estimation. We also derive a version of the extended algorithm that works entirely in real arithmetic when the data is real. Our numerical experiments show the new algorithms to be superior in accuracy to, and often faster than, the original Schur–Padé algorithm for computing matrix powers and more accurate than several alternative methods for computing the Fréchet derivative. They also show that reliable estimates of the condition number of A^t are obtained by combining the algorithms with a matrix norm estimator.

Key words. matrix power, fractional power, matrix root, Fréchet derivative, condition number, condition estimate, Schur decomposition, Padé approximation, Padé approximant, matrix logarithm, matrix exponential, MATLAB

AMS subject classifications. 65F30

1. Introduction.

We recently developed a Schur–Padé algorithm [25] for computing arbitrary real powers A^t of a matrix $A \in \mathbb{C}^{n\times n}$. The algorithm combines a Schur decomposition with the evaluation of a Padé approximant at a suitably transformed Schur factor. That work was motivated by the increasing appearance of fractional matrix powers in a variety of applications. In addition to the literature cited in [25], we mention relevant recent work on fractional differential equations by Benson, Meerschaert and Revielle [8], Burrage, Hale, and Kay [12], and Ilić, Turner, and Anh [31].

We recall that for $A \in \mathbb{C}^{n\times n}$ having no eigenvalues on the closed negative real axis $\mathbb{R}^-$, the principal matrix power A^t is defined as $\exp(t\log A)$, where log is the principal logarithm (the one whose eigenvalues have imaginary parts in the interval $(-\pi,\pi)$ [24, Thm. 1.31]). Without loss of generality we assume throughout that $t \in (-1,1)$. For general $t \in \mathbb{R}$ we can write $t = m + f$, where $m \in \mathbb{Z}$ and $f \in (-1,1)$, and then $A^t = A^m A^f$. How to choose between the two possible pairs (m,f) is explained in [25, Sec. 6].

This work has three aims: to improve the accuracy and efficiency of the Schur–Padé algorithm by sharpening the underlying error analysis, to extend the algorithm so that it computes the Fréchet derivative, and to develop a version of the extended algorithm that works entirely in real arithmetic when the data is real. The need for a more sophisticated error analysis is illustrated by the upper triangular matrix

$$A = \begin{bmatrix} 1 & 10^{16} & 0 \\ 0 & 1 & 10^{16} \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.1}$$

*Version of April 22, 2013. This work was supported by European Research Council Advanced Grant MATFUN (267526).

†School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (nicholas.j.higham@manchester.ac.uk, http://www.maths.man.ac.uk/~higham, lijing.lin@manchester.ac.uk, http://www.maths.manchester.ac.uk/~lijing).

For any $t \in (-1,1)$ the Schur–Padé algorithm of [25] takes 108 square roots of A then evaluates $r_m(I-A)$ for some $m \le 7$, where $r_m(x)$ denotes the $[m/m]$ Padé approximant to $(1-x)^t$. However, no square roots are necessary: since $(I-A)^k = 0$ for $k \ge 3$, the error in $r_m(I-A)$ is zero for any $m \ge 1$. By estimating the quantities $\|(I-A)^k\|^{1/k}$ for a few k, our new Algorithm 3.1 requires no square roots for this matrix, and returns a result with the same (high) accuracy as that from the original algorithm.

The Fréchet derivative of a general matrix function f on $\mathbb{C}^{n\times n}$ is a linear operator $L_f(\cdot, E)$ satisfying

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

for all $E \in \mathbb{C}^{n\times n}$. The Fréchet derivative therefore describes the sensitivity of f to small perturbations, and can be used to define a condition number

$$\operatorname{cond}(f,A) := \lim_{\epsilon\to 0} \sup_{\|E\|\le\epsilon\|A\|} \frac{\|f(A+E)-f(A)\|}{\epsilon\|f(A)\|} = \frac{\|L_f(A)\|\,\|A\|}{\|f(A)\|}, \tag{1.2}$$

where

$$\|L_f(A)\| := \max_{Z\ne 0} \frac{\|L_f(X,Z)\|}{\|Z\|} \tag{1.3}$$

[24, Sec. 3.1]. We develop an algorithm for computing L_{x^t} by Fréchet differentiating our algorithm for A^t. We then use the algorithm in conjunction with a matrix norm estimator to estimate $\operatorname{cond}(x^t, A)$. By replacing the Schur decomposition with a real Schur decomposition in the case where A and E are real, we derive an algorithm that works entirely in real arithmetic, thereby halving the required intermediate storage and halving the number of (real) arithmetic operations required.

In section 2 we extend the analysis of [25] for the error in Padé approximants of $(1-x)^t$ evaluated at a matrix argument $X \in \mathbb{C}^{n\times n}$ to use the quantities $\|X^k\|^{1/k}$ instead of $\|X\|$. In section 3 we develop a Schur–Padé algorithm based on these bounds. In section 4 we extend the algorithm to compute both A^t and the Fréchet derivative $L_{x^t}(A,E)$ and make comparisons with other approaches to computing the Fréchet derivative. In section 5 we modify the algorithms so that they use only real arithmetic when A and E are real. In section 6 we explain how to use the algorithms to estimate the condition number of A^t, based on evaluations of $L_{x^t}(A,E)$ for several E. Numerical experiments that compare the new algorithms with existing ones are given in section 7 and conclusions are given in section 8.

2. Forward error analysis. The Schur–Padé algorithm of Higham and Lin [25] computes a Schur decomposition $A = QTQ^*$ (Q unitary, T upper triangular), takes s square roots of T, evaluates the $[m/m]$ Padé approximant $r_m(x)$ of $(1-x)^t$ at $I - T^{1/2^s}$, squares the result s times, then transforms back. The choice of m and s in the algorithm is based on the forward error bound in the following theorem. The norm here is any subordinate matrix norm and $r_{km}(x)$ denotes the $[k/m]$ Padé approximant to $(1-x)^t$.

THEOREM 2.1 ([25, Thm. 3.5]). *Let $X \in \mathbb{C}^{n\times n}$ with $\|X\| < 1$. For $k \ge m$ and $-1 < t < 1$,*

$$\|(I-X)^t - r_{km}(X)\| \le |(1-\|X\|)^t - r_{km}(\|X\|)|. \tag{2.1}$$

In particular, when $-1 < t < 0$, (2.1) holds for $k \ge m-1$.

The bound in Theorem 2.1 can be sharpened by applying the technique of Al-Mohy and Higham [3] in order to base the error bound on the quantities

$$\alpha_p(X) = \max\bigl(\|X^p\|^{1/p}, \|X^{p+1}\|^{1/(p+1)}\bigr), \tag{2.2}$$

where the integer $p \geq 1$. As explained in [3], $\rho(X) \leq \alpha_p(X) \leq \|X\|$, where ρ is the spectral radius, and $\alpha_p(X) \ll \|X\|$ is possible for nonnormal X. Moreover, if $h_\ell(x) = \sum_{i=\ell}^\infty c_i x^i$ is a power series with radius of convergence ω then, from [3, Thm. 4.2(a)], for any $X \in \mathbb{C}^{n\times n}$ with $\rho(X) < \omega$ we have $\|h_\ell(X)\| \leq \sum_{i=\ell}^\infty |c_i|\alpha_p(X)^i$ for p satisfying $p(p-1) \leq \ell$. A new error bound in terms of the α_i is given in the next result.

THEOREM 2.2. *Let $X \in \mathbb{C}^{n\times n}$ with $\rho(X) < 1$. For $k \geq m$ and $-1 < t < 1$, and for $k \geq m-1$ and $-1 < t < 0$,*

$$(I-X)^t - r_{km}(X) = \sum_{i=k+m+1}^{\infty} \psi_i X^i, \tag{2.3}$$

where every coefficient $\psi_i \equiv \psi_i(t,k,m)$ has the same sign. Moreover,

$$\|(I-X)^t - r_{km}(X)\| \leq |(1-\alpha_p(X))^t - r_{km}(\alpha_p(X))|, \tag{2.4}$$

where α_p is defined in (2.2) *and p satisfies*

$$k+m+1 \geq p(p-1). \tag{2.5}$$

Proof. The error in the $[k/m]$ Padé approximant $r_{km}(x)$ to $(1-x)^t$ can be written [25, Lem. 3.4]

$$(1-x)^t - r_{km}(x) = q_{km}(1)q_{km}(x)^{-1} \sum_{i=k+m+1}^{\infty} \frac{(-t)_i (i-(k+m))_m}{i!(i-t-m)_m} x^i, \quad |x| < 1,$$

where q_{km} is the denominator of r_{km} with $q_{km}(0) = 1$ and $(a)_i \equiv a(a+1)\dots(a+i-1)$ with $(a)_0 = 1$. We can write $q_{km}(x) = c\prod_{i=1}^m (x_i - x)$, where x_i, $i = 1\colon m$, are the zeros of $q_{km}(x)$ and $c = \prod_{i=1}^m x_i^{-1}$. Assuming $k \geq m$, from [25, Cor. 3.3], [33, Cor. 1] we have $x_i > 1$, for all i. Hence if $|x| < 1$,

$$q_{km}(x)^{-1} = \prod_{i=1}^m \left(1 - \frac{x}{x_i}\right)^{-1} = \prod_{i=1}^m \left(1 + \frac{x}{x_i} + \frac{x^2}{x_i^2} + \cdots\right) =: \sum_{i=0}^\infty d_i x^i,$$

where $d_i > 0$. (That $d_i > 0$ also follows when $t \in (-1,0)$ from a more general result of Gomilko, Greco, and Ziętak [19, Cor. 5.6].)

It follows that for any $t \in (-1,1)$, in the error expansion $(1-x)^t - r_{km}(x) = \sum_{i=k+m+1}^\infty \psi_i x^i$ each ψ_i has the same sign. Therefore (2.3) holds for $\rho(X) < 1$. Finally, by applying [3, Thm. 4.2(a)] to (2.3), we obtain

$$\begin{aligned}\|(I-X)^t - r_{km}(X)\| &\leq \sum_{i=k+m+1}^{\infty} |\psi_i|\alpha_p(X)^i \\ &= \left|\sum_{i=k+m+1}^{\infty} \psi_i \alpha_p(X)^i\right| = \bigl|(1-\alpha_p(X))^t - r_{km}(\alpha_p(X))\bigr|\end{aligned}$$

TABLE 3.1
Values of θ_m in (3.2) and η_m in (4.7).

m	1	2	3	4	5	6	7	8	9
θ_m	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
η_m	2.57e-7	3.26e-4	7.07e-3	3.28e-2	8.10e-2	1.45e-1	2.17e-1	2.89e-1	3.58e-1
m	10	11	12	13	14	15	16	32	64
θ_m	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
η_m	4.21e-1	4.78e-1	5.28e-1	5.73e-1	6.12e-1	6.47e-1	6.77e-1	8.93e-1	9.68e-1

TABLE 3.2
Values of p for which $2m+1 \ge p(p-1)$ is satisfied, for $m = 1\colon 7$, and corresponding required α_i and $d_i = \|X^i\|^{1/i}$ (in view of $\alpha_3 \le a_2 \le \alpha_1$).

m	1	2	3, 4, 5	6, 7
p	1, 2	1, 2	1, 2, 3	1, 2, 3, 4
Required α_i	α_2	α_2	α_3	α_3, α_4
Required d_i	d_2, d_3	d_2, d_3	d_3, d_4	d_3, d_4, d_5

for p satisfying (2.5).

If $-1 < t < 0$, it follows from [25, Cor. 3.3], [33, Cor. 1] that the roots x_i of q_{km} satisfy $x_i > 1$ for $k \ge m-1$, so the conclusion of the theorem holds for $k \ge m-1$. □

3. Improved Schur–Padé algorithm. Whereas the Schur–Padé algorithm of [25] is based on the norm of $I - T^{1/2^s}$, through the use of (2.1), here we will exploit (2.4), which uses the generally smaller quantities $\alpha_p(I - T^{1/2^s})$, $p \ge 1$. We require that $\alpha_p \equiv \alpha_p(I - T^{1/2^s})$ satisfies

$$|(1-\alpha_p)^t - r_m(\alpha_p)| \le u, \tag{3.1}$$

where $u = 2^{-53} \approx 1.1 \times 10^{-16}$ is the unit roundoff for IEEE double precision arithmetic, which by (2.4) ensures double precision accuracy of the Padé approximant. For given t and m, we denote by $\theta_m^{(t)}$ the largest θ such that $|(1-\theta)^t - r_m(\theta)| \le u$, and define

$$\theta_m = \min\{\, \theta_m^{(t)} : t \in [-1, 1] \,\}. \tag{3.2}$$

The values of θ_m, determined in [25], are shown in Table 3.1. Our algorithm requires that $\alpha_p(I - T^{1/2^s}) \le \theta_m$ for some p satisfying $p(p-1) \le 2m+1$.

As in [25] we compute square roots of triangular matrices by the Björck and Hammarling recurrence [11], [24, Alg. 6.3], which costs $n^3/3$ flops, and we compute r_m by evaluating its continued fraction form in bottom-up fashion (see section 4.2), which costs $2mn^3/3$ flops.

Our overall aim is to choose the parameters s and m to minimize the total cost subject to (3.1). The following reasoning is adapted from that used to derive an inverse scaling and squaring algorithm for the matrix logarithm that is also based on the α_p [4]. Although the θ_m and the cost of evaluating the Padé approximant are different than in the logarithm case, the algorithm turns out (perhaps surprisingly) to have exactly the same form.

For any putative s and m satisfying $\alpha_p(I - T^{1/2^s}) \le \theta_m$ for some p, it is worth taking one more square root (costing $n^3/3$ flops) if it allows a reduction in the Padé

degree m by more than one (resulting in a saving of at least $2n^3/3$ flops once the extra squarings are taken into account), that is, if $\alpha_p(I-T^{1/2^{s+1}}) \le \theta_{m-2}$ for some p. Since $(I-T^{1/2^{s+1}})(I+T^{1/2^{s+1}}) = I-T^{1/2^s}$, we have that $\alpha_p(I-T^{1/2^{s+1}}) \approx \frac{1}{2}\alpha_p(I-T^{1/2^s})$, for suitably large s. Since $\theta_m/2 < \theta_{m-2}$ for $m > 7$, the cost of computing T^t will be minimized if we take square roots of T repeatedly until $\alpha_p(I - T^{1/2^s}) \le \theta_7$ for some $p \in \{1,2,3,4\}$ (see Table 3.2). Some work can be saved by using the fact that $\rho(I - D) = \rho(I - T) \le \alpha_p(I - T)$, where $D = \mathrm{diag}(T)$, and so there is no need to compute $\alpha_p(I - T^{1/2^s})$ until $\rho(I - D^{1/2^s}) \le \theta_7$; let us denote the smallest such s by s_0.

After obtaining s_0 and computing $T \leftarrow T^{1/2^{s_0}}$, we first check if $m = 1$ or $m = 2$ can be used, since no more square roots need to be taken in this case. If $m = 1$ and $m = 2$ are ruled out we now determine whether more square roots are needed and which Padé degree m to use. By the analysis above, m does not exceed 7, so only α_3 and α_4 will be needed, as can be seen from Table 3.2. We next check whether $\alpha_3(I - T) \le \theta_7$. If $\alpha_3(I - T) > \theta_7$, we check whether $\alpha_4(I - T) > \theta_7$, and if it is we set $T \leftarrow T^{1/2}$ and repeat the process; if $\alpha_3(I - T) > \theta_7$ and $\alpha_4(I - T) \le \theta_7$ then it is necessary to check whether $\alpha_4(I - T) \le \theta_6$ to determine whether $m = 7$ or $m = 6$ is to be used. We now consider the case where $\alpha_3(I - T) \le \theta_7$. If $\alpha_3(I - T) > \theta_6$ and $\frac{1}{2}\alpha_3(I - T) \le \theta_5$ then the above analysis predicts that one more square root should be taken to reduce the cost and so the process is repeated with $T \leftarrow T^{1/2}$. Since it is not guaranteed that $\alpha_3(I - T^{1/2}) \le \theta_5$, we will allow at most two extra square roots to be taken to avoid unnecessary square roots. If $\frac{1}{2}\alpha_3(I - T) > \theta_5$, then no extra square root is needed and again $\alpha_4(I - T)$ needs to be checked to determine whether $m = 7$ or $m = 6$ is to be used. Consider now the case where $\alpha_3(I - T) \le \theta_6$: an extra square root is not necessary since $\theta_k < \frac{1}{2}x$ for all $x \in (\theta_{k+1}, \theta_{k+2}]$ for $k = 2, 3, 4$; we find the smallest $m \in \{3, 4, 5, 6\}$ such that $\alpha_3(I - T) \le \theta_m$ and evaluate $r_m(I - T)$.

We use the 1-norm, so $\alpha_p(X) = \max\{\|X^p\|_1^{1/p}, \|X^{p+1}\|_1^{1/(p+1)}\}$, but instead of computing $\|X^p\|_1$, we estimate it without forming X^p by using the block 1-norm estimation algorithm of Higham and Tisseur [28], so that we obtain an estimate of $\alpha_p(X)$ in $O(n^2)$ flops. The 1-norm estimator estimates $\|B\|_1$ by sampling a small number of products BY and B^*Z for $Y, Z \in \mathbb{R}^{n\times r}$, where r is a parameter that we set to 2. It requires $4r$ products on average and is rarely more than a factor of 3 away from the true 1-norm [24, p. 67].

We are now ready to state the improved algorithm. Note that we take the opportunity on lines 37, 41, and 42 to recompute quantities for which there is an explicit formula that can be evaluated accurately.

ALGORITHM 3.1 (Schur–Padé algorithm). *Given $A \in \mathbb{C}^{n\times n}$ with no eigenvalues on $\mathbb{R}^-$ and a nonzero $t \in (-1, 1)$ this algorithm computes $X = A^t$ via a Schur decomposition and Padé approximation. It uses the constants θ_m in Table* 3.1 *and the function* `normest`(A, m)*, which produces an estimate of $\|A^m\|_1$. The algorithm is intended for IEEE double precision arithmetic.*

1 Compute a (complex) Schur decomposition $A = QTQ^*$.
2 If T is diagonal, $X = QT^tQ^*$, quit, end
3 $T_0 = T$
4 Find s_0, the smallest s such that $\rho(I - D^{1/2^s}) \le \theta_7$, where $D = \mathrm{diag}(T)$.
5 for $i = 1\colon s_0$
6 $\quad T \leftarrow T^{1/2}$ using [11], [24, Alg. 6.3].
7 end
8 $s = s_0$, $q = 0$

```
9  d_2 = normest(I − T, 2)^{1/2}, d_3 = normest(I − T, 3)^{1/3}
10 α_2 = max(d_2, d_3)
11 for i = 1: 2
12       if α_2 ≤ θ_i, m = i, goto line 38, end
13 end
14 while true
15       if s > s_0, d_3 = normest(I − T, 3)^{1/3}, end
16       d_4 = normest(I − T, 4)^{1/4}, α_3 = max(d_3, d_4)
17       if α_3 ≤ θ_7
18            j_1 = min{ i: α_3 ≤ θ_i, i = 3: 7 }
19            if j_1 ≤ 6
20                 m = j_1, goto line 38
21            else
22                 if (1/2)α_3 ≤ θ_5 and q < 2
23                      q = q + 1
24                      goto line 33
25                 end
26            end
27       end
28       d_5 = normest(I − T, 5)^{1/5}, α_4 = max(d_4, d_5)
29       η = min(α_3, α_4)
30       for i = 6: 7
31            if η ≤ θ_i, m = i, goto line 38
32       end
33       T ← T^{1/2} using [11], [24, Alg. 6.3].
34       s = s + 1
35 end
36 R = I − T
37 Replace the diagonal and first superdiagonal of R by the diagonal and
   first superdiagonal of I − T_0^{1/2^s} computed via [1, Alg. 2] and [25, (5.6)],
   respectively.
38 Evaluate U = r_m(R) using the continued fraction in bottom-up fashion
   [25, Alg. 4.1].
39 for i = s: −1: 0
40     if i < s, U ← U^2, end
41     Replace diag(U) by diag(T_0)^{t/2^i}.
42     Replace first superdiagonal of U by first superdiagonal of T_0^{t/2^i}
       obtained from [25, (5.6)] with the power t/2^i.
43 end
44 X = QUQ^*
```

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

4. Computing the Fréchet derivative. We now turn to the computation of $L_{x^t}(A, E)$, the Fréchet derivative of A^t at A in the direction E. The idea we pursue is to simultaneously compute A^t and $L_{x^t}(A, E)$ in a way that reuses matrix operations from the computation of A^t in the computation of $L_{x^t}(A, E)$.

Recall that A^t is approximated by (ignoring the Schur decomposition for simplic-

ity)

$$A^t = (A^{1/2^s})^{t\cdot 2^s} = (I-X)^{t\cdot 2^s} \approx r_m(X)^{2^s},$$

where $I - X = A^{1/2^s}$ and $\rho(X) < 1$. Differentiating $A^t = ((A^{1/2})^t)^2$ and applying the chain rule [24, Thm. 3.4], we obtain

$$L_{x^t}(A,E) = A^{t/2}L_{x^t}(A^{1/2},E_1) + L_{x^t}(A^{1/2},E_1)A^{t/2}, \tag{4.1}$$

where $E_1 = L_{x^{1/2}}(A,E)$ and so $A^{1/2}E_1 + E_1A^{1/2} = E$. Using this relation we can construct the following recurrences for computing $L_{x^t}(A,E)$. First we form

$$\left.\begin{aligned} E_0 = E, \quad X_0 = A, \\ X_i = X_{i-1}^{1/2} \\ \text{Solve } X_iE_i + E_iX_i = E_{i-1} \text{ for } E_i \end{aligned}\right\} \quad i = 1\colon s, \tag{4.2}$$

after which $E_s = L_{x^{1/2^s}}(A,E)$ and $X_s = A^{1/2^s}$, and then

$$Y_s = r_m(I-X_s), \quad L_s \approx L_{x^t}(X_s,E_s), \tag{4.3}$$

$$\left.\begin{aligned} L_{i-1} &= Y_iL_i + L_iY_i \\ Y_{i-1} &= Y_i^2 \end{aligned}\right\} \quad i = s:-1:1, \tag{4.4}$$

after which $L_0 \approx L_{x^t}(A,E)$.

To approximate $L_{x^t}(X_s,E_s)$ in (4.3) we simply differentiate the Padé approximation (note that $L_{x^t}(X,E) = L_{(1-x)^t}(I-X,-E)$):

$$L_{x^t}(X_s,E_s) \approx L_{r_m}(I-X_s,-E_s). \tag{4.5}$$

We now bound the error in this approximation.

4.1. Error analysis. From Theorem 2.2, the error in $r_m(X)$ has the form

$$(I-X)^t - r_m(X) = \sum_{i=2m+1}^{\infty} \psi_i X^i =: h_{2m+1}(X). \tag{4.6}$$

Differentiating both sides of (4.6), we have

$$L_{(1-x)^t}(X,E) - L_{r_m}(X,E) = L_{h_{2m+1}}(X,E) = \sum_{i=2m+1}^{\infty} \psi_i \sum_{j=1}^{i} X^{j-1}EX^{i-j},$$

where the second equality is from [24, Prob. 3.6]. Therefore

$$\begin{aligned} \|L_{(1-x)^t}(X,E) - L_{r_m}(X,E)\| &\le \sum_{i=2m+1}^{\infty} |\psi_i| \sum_{j=1}^{i} \|X^{j-1}EX^{i-j}\| \\ &\le \left| \sum_{i=2m+1}^{\infty} i\psi_i \|X\|^{i-1}\|E\| \right| \\ &= \left| h'_{2m+1}(\|X\|) \right| \|E\|. \end{aligned}$$

Define $\eta_m^{(t)} = \max\{\, x : |h'_{2m+1}(x)| \le u \,\}$ and

$$\eta_m = \min\{\, \eta_m^{(t)} : t \in [-1,1] \,\}. \tag{4.7}$$

With $u = 2^{-53}$, we determined η_m empirically in MATLAB, using high precision computations with the Symbolic Math Toolbox for a range of $m \in [1,64]$. Table 3.1 reports the results to three significant figures. Notice that for all m we have $\eta_m < \theta_m$.

Our error bound for the Fréchet derivative of r_m is based on $\|X\|$, whereas the error bound for r_m itself is based on $\alpha_p(X)$. The same situation holds in the work of Al-Mohy, Higham, and Relton [5] for the matrix logarithm, and the arguments used there apply here, too, to show that an α_p-based bound for $\|L_{(1-x)^t}(X,E) - L_{r_m}(X,E)\|$ is not possible. Despite the fact that $\eta_m < \theta_m$, we will base our algorithm for computing A^t and $L_{x^t}(A,E)$ on the condition $\alpha_p(I - A^{1/2^s}) \le \theta_m$ (analogously to [5]) and will test experimentally whether this produces accurate Fréchet derivatives.

4.2. Evaluating the Fréchet derivative of r_m. In [25], $r_m(X)$ is computed by evaluating the continued fraction [6, p. 66], [7, p. 174]

$$r_m(x) = 1 + \cfrac{c_1 x}{1 + \cfrac{c_2 x}{1 + \cfrac{c_3 x}{\cdots \cfrac{}{1 + \cfrac{c_{2m-1}x}{1 + c_{2m}x}}}}}, \tag{4.8}$$

where

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

in bottom-up fashion. Denote

$$\begin{aligned} &y_{2m}(x) = c_{2m}x, \\ &y_j(x) = \frac{c_j x}{1 + y_{j+1}(x)}, \quad j = 2m-1 : -1 : 1. \end{aligned} \tag{4.9}$$

Then we have $r_m(x) = 1 + y_1(x)$. From (4.9), $(1+y_{j+1}(x))y_j(x) = c_j x$. Differentiating the matrix analogue, we have

$$L_{y_{j+1}}(X,E)y_j(X) + (I + y_{j+1}(X))L_{y_j}(X,E) = c_j E,$$

which, together with $L_{y_m}(X,E) = c_{2m}E$, provide a recurrence for computing $L_{r_m}(X,E) = L_{y_1}(X,E)$. We obtain the following algorithm for evaluating both r_m and L_{r_m}.

ALGORITHM 4.1 (continued fraction, bottom-up). *This algorithm evaluates $r_m(X)$ and $L_{r_m}(X,E)$ for $X, E \in \mathbb{C}^{n\times n}$.*

```
1  Y_2m = c_2m X, Z_2m = c_2m E
2  for j = 2m − 1: −1: 1
3      Solve (I + Y_{j+1})Y_j = c_j X for Y_j.
4      Solve (I + Y_{j+1})Z_j = c_j E − Z_{j+1}Y_j for Z_j.
5  end
6  r_m = I + Y_1
7  L_{r_m} = Z_1
```

Cost: In the case where X is a triangular matrix and E is full, the total cost is $(2m-1)(n^3/3+2n^3)$ flops.

We are now ready to state the overall algorithm for computing both A^t and $L_{x^t}(A,E)$. In lines 3–8 we employ an explicit formula obtained from the Daleckiĭ and Kreĭn theorem for the Fréchet derivative that applies in the case of normal A [24, Thm. 3.11].

ALGORITHM 4.2 (Schur–Padé algorithm for matrix power and Fréchet derivative). *Given $A \in \mathbb{C}^{n\times n}$ with no eigenvalues on $\mathbb{R}^-$ and a nonzero $t \in (-1,1)$ this algorithm computes $X = A^t$ and its Fréchet derivative $L_{x^t}(A,E)$ via a Schur decomposition and Padé approximation. It uses the constants θ_m in Table* 3.1. *The algorithm is intended for IEEE double precision arithmetic.*

1 Compute a (complex) Schur decomposition $A = QTQ^*$.
2 $E \leftarrow Q^*EQ$
3 If T is diagonal
4 $\quad X = QT^tQ^*$
5 $\quad$ Form K, where k_{ij} is the divided difference $f[t_{ii}, t_{jj}]$ for $f(x) = x^t$ computed from [25, (5.6)].
6 $\quad L = Q(K \circ E)Q^*$, where $\circ$ is the Hadamard product
7 $\quad$ quit
8 end
9 $T_0 = T$
10 Find s_0, the smallest s such that $\rho(I - D^{1/2^s}) \le \theta_7$, where $D = \mathrm{diag}(T)$.
11 $E_0 = E$
12 for $i = 1\colon s_0$
13 $\quad T \leftarrow T^{1/2}$ using [11], [24, Alg. 6.3].
14 $\quad$ Solve $TE_i + E_iT = E_{i-1}$ for E_i by substitution.
15 end
16 ... Execute lines 8–44 in Algorithm 3.1 but with the following changes:
Immediately after line 33 execute
"Solve $TE_{s+1} + E_{s+1}T = E_s$ for E_{s+1} by substitution."
Replace line 38 with
"Evaluate $U = r_m(R)$ and $V = L_{r_m}(R, -E_s)$ using Algorithm 4.1."
Replace line 40 with
"if $i < s$, $V \leftarrow UV + VU$, $U \leftarrow U^2$, end "
Replace line 44 with
"$X = QUQ^*$, $L = QVQ^*$"

Cost: The cost is the $(28 + (2s + 2m - 1)/3)n^3$ flops cost of Algorithm 3.1 plus the extra cost for computing $L_{x^t}(A,E)$ of s solves of triangular Sylvester equations, $2m-1$ full-triangular matrix multiplications and $2m-1$ solves of multiple right-hand side triangular systems (in Algorithm 4.1), $2s$ full-triangular matrix multiplications, and 2 full matrix multiplications, namely an extra cost of $(4s + 4m + 2)n^3$ flops.

In some situations, such as in condition estimation (see section 6), several Fréchet derivatives $L_{x^t}(A,E)$ are needed for a fixed A and different E. The parameters s and m depend only on A so we need only compute them once; moreover, we can save and reuse the Schur decomposition, the square roots $T^{1/2^i}$, and the powers $U^{2^i} \approx T^{2^{i-s}t}$.

4.3. Alternative algorithms. We describe several alternative ways to compute the Fréchet derivative of A^t.

By applying the chain rule to the expression $A^t = \exp(t \log A)$ we obtain [25,

Sec. 2]

$$L_{x^t}(A,E) = tL_{\exp}\big(t\log A, L_{\log}(A,E)\big).$$

The first method evaluates this formula using the inverse scaling and squaring algorithm of Al-Mohy, Higham, and Relton [5] to evaluate $L_{\log}$ and the scaling and squaring algorithm of Al-Mohy and Higham [2] to evaluate $L_{\exp}$. Both these algorithms are based on Padé approximation. The total cost, assuming a Schur decomposition is initially computed and used for both the $L_{\log}$ and $L_{\exp}$ computations and that the maximal Padé degree is chosen in each algorithm, is about $(68\frac{2}{3} + \frac{7}{3}(s_1+s_2))n^3$ flops, where s_1 and s_2 are the scaling parameters for the $L_{\log}$ and $L_{\exp}$ computations, respectively. This is to be compared with $(62\frac{1}{3} + 4\frac{2}{3}s)n^3$ flops for Algorithm 4.2, and since s will be of similar size to s_1 these two approaches will be of broadly similar cost.

A second method is based on the property, for arbitrary f [24, (3.16)],

$$f\left(\begin{bmatrix} A & E \\ 0 & A \end{bmatrix}\right) = \begin{bmatrix} f(A) & L_f(A,E) \\ 0 & f(A) \end{bmatrix}, \tag{4.10}$$

which shows that by applying Algorithm 3.1 to the $2n \times 2n$ matrix $\left[\begin{smallmatrix} A & E \\ 0 & A \end{smallmatrix}\right]$ we obtain A^t and $L_{x^t}(A,E)$ simultaneously. This method has two drawbacks. First, it has eight times the cost and four times the storage requirement of Algorithm 3.1 (both of which can be reduced by exploiting the block triangular structure). Second, since $L_f(A,E)$ is linear in E the norm of E should not affect an algorithm for computing $L_f(A,E)$, but Algorithm 3.1 applied to $\left[\begin{smallmatrix} A & E \\ 0 & A \end{smallmatrix}\right]$ will be affected by $\|E\|$ and the best way to scale E is not clear.

The other approaches that we consider are applicable only when $q = 1/t$ is an integer. The problem is then to compute the Fréchet derivative for the matrix qth root. This can be done by exploring the relation [24, Thm. 3.5]

$$L_{x^q}(A^{1/q}, L_{x^{1/q}}(A,E)) = E. \tag{4.11}$$

Since $L_{x^q}(X,E) = \sum_{j=1}^{q} X^{j-1}EX^{q-j}$, $L_{x^{1/q}}(A,E)$ can be obtained by solving the generalized Sylvester equation $\sum_{j=1}^{q}(A^{1/q})^{j-1}Y(A^{1/q})^{q-j} = E$ for Y. An explicit formula expressing the solution of the more general equation $\sum_{i=1}^{m} A^{m-i}XB^i = Y$ as an infinite integral is given by Bhatia and Uchiyama [9]. However, currently no efficient algorithm is known for solving this equation. Another way to compute $L_{x^{1/q}}(A,E)$ is proposed by Cardoso [13]. The idea is first to write the Fréchet derivative $L_{x^q}(A,E)$ in terms of the solution of a set of q recursive Sylvester equations and then reverse the procedure (in view of (4.11)) to get $L_{x^{1/q}}$. The matrix $A^{1/q}$ must be computed by some other method before applying this procedure. To save computation in solving the Sylvester equations, an initial Schur decomposition is used, which makes the coefficients of the Sylvester equations triangular. This method costs $(4q + 31\frac{1}{3})n^3$ flops plus the cost of computing $A^{1/q}$ and always requires complex arithmetic, even when A and E are real. Recall that the extra cost in Algorithm 4.2 in addition to that for computing A^t is $(4s + 4m + 2)n^3$ flops. The values of of q, s, and m depend on the problem but $m \le 7$, so Cardoso's algorithm will be competitive in cost only if $q \le s$.

Another method for the qth root case is proposed by Cardoso [14], who applies the repeated trapezium rule to an integral representation of the Fréchet derivative. This method is competitive in cost only when low accuracy is required (relative errors $\ge u^{1/2}$, say), and its cost increases rough linearly with q, so we will not consider it further.

5. Algorithm for real data. In the case where A and E are real both A^t and $L_{x^t}(A,E)$ are real, so an algorithm that avoids complex arithmetic is desired to increase the efficiency of the computation and guarantee a real result in floating point arithmetic. We summarize the changes that can be made to Algorithms 4.1 and 4.2 so that they work entirely in real arithmetic for real inputs.

1. Use the real Schur decomposition instead of the Schur decomposition.

2. Compute real square roots of real quasi-triangular matrices using the recurrence of Higham [23], [24, Alg. 6.7].

3. Instead of updating the diagonal and the first superdiagonal elements before the squaring stage by using an explicit formula for the tth power of a 2×2 triangular matrix, update the (full) 2×2 diagonal blocks. Assume the 2×2 diagonal blocks are of the form

$$B = \begin{bmatrix} a & b \\ c & a \end{bmatrix}$$

with $bc < 0$, which is the case when the real Schur decomposition is computed by MATLAB. Then B has eigenvalues $\lambda_\pm = a \pm i\beta$, where $\beta = (-bc)^{1/2}$. Let $\theta = \arg(\lambda_+) \in (0,\pi)$ and $r = |\lambda_+|$. It can be shown that

$$B^t = \frac{r^t}{\beta}\begin{bmatrix} \beta\cos(t\theta) & b\sin(t\theta) \\ c\sin(t\theta) & \beta\cos(t\theta) \end{bmatrix},$$

which can be evaluated to high relative accuracy as long as we are able to compute θ, cos, and sin accurately. The explicit formula for 2×2 triangular matrices can still be used to update the first superdiagonal elements when two or more successive 1×1 diagonal blocks are found.

With these changes we will gain a halving of the storage required for intermediate matrices and an approximate halving of the operation count measured in real arithmetic operations. Fortran experiments reported in [5] with the inverse scaling and squaring algorithm for the matrix logarithm show that use of the real instead of complex Schur form halves the run time, and the same will be true here since exactly the same computational kernels are used.

6. Condition number estimation. From (1.2) and (1.3) it is clear that the essential task in computing or estimating the condition number is to compute or estimate the norm $\|L_{x^t}(A)\|$ of the Fréchet derivative.

Denoting by vec the operator that stacks the columns of a matrix into one long vector, for a general f we have $\mathrm{vec}(L_f(A,Z)) = K_f(A)z$, where $z = \mathrm{vec}(Z)$, for a certain matrix $K_f(A) \in \mathbb{C}^{n^2\times n^2}$ called the Kronecker representation of the Fréchet derivative. Moreover [24, Lem. 3.18],

$$\frac{\|L_f(A)\|_1}{n} \le \|K_f(A)\|_1 \le n\|L_f(A)\|_1. \tag{6.1}$$

We will therefore apply the block matrix 1-norm estimation algorithm of [28] to $K_{x^t}(A)$, which requires the computation of $K_{x^t}(A)y$ and $K_{x^t}(A)^*z$ for given vectors y and z. In order to avoid forming the $n^2\times n^2$ matrix $K_{x^t}(A)$ we compute $K_{x^t}(A)y$ as $\mathrm{vec}(L_{x^t}(A,Y))$, where $\mathrm{vec}(Y) = y$. How to compute $K_{x^t}(A)^*z$ is less clear. The following results provide an answer for a general function f.

Our analysis makes use of the adjoint $L_f^\star$ of the Fréchet derivative L_f, which is defined by the condition

$$\langle L_f(A,G), H\rangle = \langle G, L_f^\star(A,H)\rangle \tag{6.2}$$

for all $G, H \in \mathbb{C}^{n\times n}$, where $\langle X, Y\rangle = \mathrm{trace}(Y^*X) = \mathrm{vec}(Y)^*\mathrm{vec}(X)$.

LEMMA 6.1. $K_f(A)^*\mathrm{vec}(H) = \mathrm{vec}(L_f^\star(A, H))$.

Proof. We have $\langle L_f(A, G), H\rangle = \mathrm{vec}(H)^*\mathrm{vec}(L_f(A, G)) = \mathrm{vec}(H)^*K_f(A)\mathrm{vec}(G)$ and $\langle G, L_f^\star(A, H)\rangle = \mathrm{vec}(L_f^\star(A, H))^*\mathrm{vec}(G)$. By the definition (6.2) of adjoint these expressions are equal for all G and so $\mathrm{vec}(H)^*K_f(A) = \mathrm{vec}(L_f^\star(A, H))^*$, which yields the result. ∎

LEMMA 6.2. *Let f be $2n-1$ times continuously differentiable on an open subset $\mathcal{D}$ of $\mathbb{R}$ or $\mathbb{C}$ such that each connected component of $\mathcal{D}$ is closed under conjugation, and suppose that $\overline{f}(A)^* = \overline{f}(A^*)$ for all $A \in \mathbb{C}^{n\times n}$ with spectrum in $\mathcal{D}$, where $\overline{f}(z) := \overline{f(\overline{z})}$. Then*

$$L_f^\star(A, E) = L_{\overline{f}}(A^*, E) = L_{\overline{f}}(A, E^*)^*. \tag{6.3}$$

Proof. Suppose, first, that f has the form $f(x) = \alpha x^k$, so that $L_f(A, G) = \alpha\sum_{i=1}^k A^{i-1}GA^{k-i}$. Then

$$\begin{aligned}
\langle L_f(A, G), H\rangle &= \mathrm{trace}\Big(H^*\alpha\sum_{i=1}^k A^{i-1}GA^{k-i}\Big)\\
&= \mathrm{trace}\Big(\alpha\sum_{i=1}^k A^{k-i}H^*A^{i-1}G\Big)\\
&= \Big\langle G, \overline{\alpha}\sum_{i=1}^k (A^*)^{i-1}H(A^*)^{k-i}\Big\rangle\\
&= \langle G, L_{\overline{f}}(A^*, H)\rangle,
\end{aligned}$$

and so $L_f^\star(A, H) = L_{\overline{f}}(A^*, H)$, which is the first equality in (6.3). By the linearity of L_f it follows that this equality holds for any polynomial. Finally, the equality holds for all f satisfying the conditions of the theorem because the Fréchet derivative of f is the same as that of the polynomial that interpolates f and its derivatives at the zeros of the characteristic polynomial of $\mathrm{diag}(A, A)$ [24, Thm. 3.7], [29, Thm. 6.6.14].

Let $g = \overline{f}$. By the definition of Fréchet derivative, $L_g(A, E) = g(A+E) - g(A) + o(\|E\|)$. Taking the conjugate transpose gives $L_g(A, E)^* = g(A + E)^* - g(A)^* + o(\|E\|) = g(A^* + E^*) - g(A^*) + o(\|E\|) = L_g(A^*, E^*) + o(\|E\|)$, and by the linearity of the Fréchet derivative it follows that $L_g(A, E)^* = L_g(A^*, E^*)$, which is equivalent to the second equality in (6.3). ∎

If f is $n-1$ times continuously differentiable on $\mathcal{D}$ (so that $f(A)$ is a continuous matrix function on the set of matrices with spectrum in $\mathcal{D}$ [24, Thm. 1.19]), then $f(A^*) = f(A)^*$ for all A with spectrum in $\mathcal{D}$ is equivalent to $\overline{f} \equiv f$ [26, Proof of Thm. 3.2]. Some other equivalent conditions for $f(A^*) = f(A)^*$ are given in [24, Thm. 1.18], [26, Thm. 3.2].

Combining Lemmas 6.1 and 6.2 gives $K_f(A)^*\mathrm{vec}(H) = \mathrm{vec}(L_{\overline{f}}(A, H^*)^*)$. For our function $f(x) = x^t$ we have $\overline{f} \equiv f$ and so to implement the condition estimation we just need to evaluate $L_f(A)$,

7. Numerical experiments. Our numerical experiments were carried out in MATLAB R2012b in IEEE double precision arithmetic. We use the same set of 45 test matrices that was used in [25] to test the original Schur–Padé algorithm. This

set includes a selection of 10×10 nonsingular matrices taken from the MATLAB `gallery` function and from the Matrix Computation Toolbox [22]. Any matrix found to have an eigenvalue on $\mathbb{R}^-$ was squared; if it still had an eigenvalue on $\mathbb{R}^-$ it was discarded. We report experiments using the matrices in the test set and their complex Schur factors T. Our previous experience [2], [5], [25] is that for methods that begin with a reduction to Schur form, differences in accuracy between different methods are greater when the original matrix is triangular, as errors in the transformations to and from Schur form are avoided.

We test each matrix with each of the 14 t-values in the vector constructed by

$$v = [\,1/52 \quad 1/12 \quad 1/3 \quad 1/2\,], \quad v = [\,v \quad 1 - v(1\colon 3)\,], \quad v = [\,v \quad -v\,].$$

For the computation of A^t alone we tested:

1. `SPade`: Algorithm 3.1.
2. `SPade-real`: the real version of Algorithm 3.1, as described in section 5.
3. `SPade-old`: the original Schur–Padé algorithm [25, Alg 5.1], which is based on the error analysis in Theorem 2.1 expressed in terms of $\|A\|$.

Relative errors are measured in the 1-norm. To compute the "exact" A^t we run `powerm` [25, Fig. 8.1] (which uses the eigendecomposition of A) in 300 digit precision with the VPA arithmetic of the Symbolic Math Toolbox, but we subject A to a random perturbation of relative norm 10^{-150} in order to ensure it is diagonalizable, following the approximate diagonalization approach of Davis [15]. All errors are postprocessed using the transformation in [17] to lessen the influence of tiny relative errors on the performance profiles.

For the Fréchet derivative $L_{x^t}(A, E)$, we tested six algorithms:

1. `SPade-Fre`: Algorithm 4.2.
2. `SPade-Fre-real`: the real version of Algorithm 4.2, as described in section 5.
3. `SPade-Fre-mod`: Algorithm 4.2 modified so as to call `SPade-old` instead of `SPade`.
4. `explog-Fre`: reduction to Schur form $T = Q^*AQ$ followed by evaluation of $L_{x^t}(A, E) = tQL_{\exp}(t \log T, L_{\log}(T, Q^*EQ))Q^*$ by the inverse scaling and squaring method for the Fréchet derivative of the logarithm [5] and the scaling and squaring method for the Fréchet derivative of the exponential [2].
5. `SPade-2by2`: `SPade` applied to the block 2×2 matrix in (4.10).
6. `rootm-Fre` (applied when $t = 1/q$ for some integer q): reduction to Schur form T with $T^{1/q}$ computed by `SPade` and $L_{x^{1/q}}(T, Q^*EQ)$ computed by [13, Alg. 3.5].

To obtain the "exact" Fréchet derivative we apply the same approach as above to (4.10).

We note that for an efficient implementation, which is not our concern here, it is important to implement carefully the computation of square roots of (quasi-) triangular matrices and the solution of (quasi-) triangular Sylvester equations. Efficient blocked and recursive ways to carry out these operations are described by Deadman, Higham, and Ralha [16] and Jonsson and Kågström [32], respectively.

Experiment 1. In this experiment, we compute A^t by `SPade` and `SPade-old` for the Schur factors of the matrices in the test set and for all values of t in the vector v. Figure 7.1 shows the relative errors, with the problems sorted by decreasing condition number. The solid line is $\mathrm{cond}(x^t, A)u$. Figure 7.2 shows the corresponding performance profile [18], [21, Sec. 22.4]. Figure 7.3 shows the ratios of the costs of the algorithms, where the cost is measured by the n^3 terms in the operation counts.

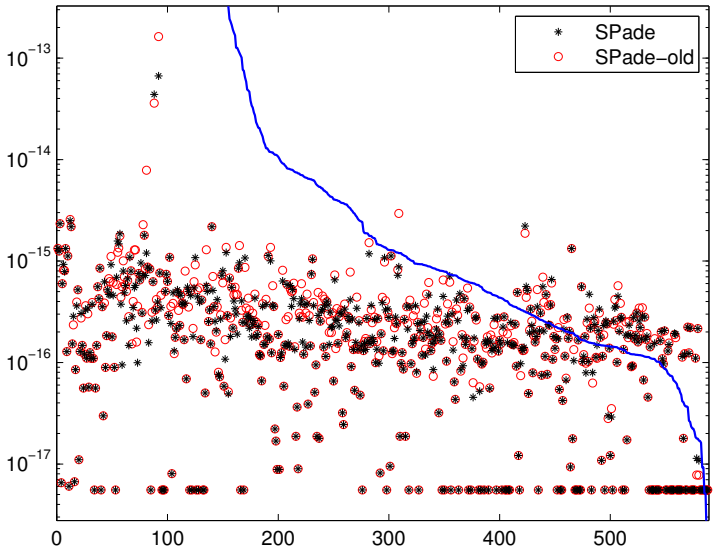

FIG. 7.1. *Experiment* 1*: relative errors in* A^t *for a selection of* 10×10 *triangular matrices and several* t*.*

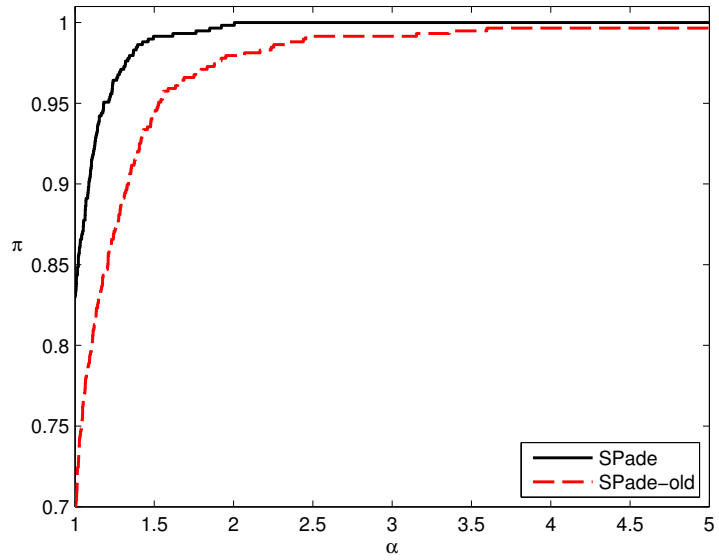

FIG. 7.2. *Experiment* 1*: performance profile for the data in Figure* 7.1.

The results show that `SPade` and `SPade-old` both perform in a stable manner but that `SPade` outperforms `SPade-old` and has a cost that is no larger and up to around 40 percent less than that of `SPade-old`. In some of the test problems `SPade` requires only half as many matrix square roots as `SPade-old`.

When the experiment is repeated with the full (un-Schur-reduced) matrices, the same trends are seen but they are less pronounced.

Experiment 2. In this experiment, we compute $L_{x^t}(A, E)$ for the Schur factors of

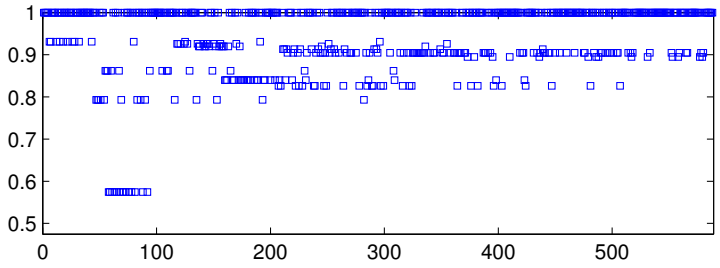

FIG. 7.3. *Experiment* 1*: ratios "cost of* `SPade`*/cost of* `SPade-old`*".*

the matrices in the test set using a different E, generated as `randn(n)`, for each pair of A and t. Note that `rootm-Fre` is only applicable for the problems where $t = 1/q$ for some integer q (the qth root problem). Figures 7.4 and 7.5 show the results from the qth root problems and Figures 7.6 and 7.7 show the results from the rest of the problems: those where $t \in [\pm 51/52 \ \pm 11/12 \ \pm 2/3]$. In each case we show the relative errors and the corresponding performance profile. The solid line in Figures 7.4 and 7.6 is $\mathrm{cond}_L(A,E)u$, where $\mathrm{cond}_L(A,E)$ is the condition number of the Fréchet derivative, defined as

$$\mathrm{cond}_L(A,E) = \lim_{\epsilon \to 0} \sup_{\substack{\|\Delta A\| \le \epsilon \|A\| \\ \|\Delta E\| \le \epsilon \|E\|}} \frac{\|L_{x^t}(A+\Delta A, E+\Delta E) - L_{x^t}(A,E)\|}{\epsilon \|L_{x^t}(A,E)\|}.$$

We estimated $\mathrm{cond}_L(A,E)$ using an algorithm of Higham and Relton [27].

Some observations can be made. First, all the methods behave stably, but the errors for `rootm-Fre` and `explog-Fre` are significantly larger than those for the other algorithms. `SPade-Fre`, `SPade-Fre-mod`, and (the very expensive) `SPade-2by2` perform very similarly. However, for the non-qth root problems, `SPade-Fre` is the clear winner.

Again, similar results, but with less pronounced differences, are obtained when working with the full test matrices.

Experiment 3. In this experiment, we test the real-arithmetic algorithms `SPade-real` and `SPade-Fre-real` on the (un-Schur-reduced) real matrices from the test set and we show just performance profiles. Figure 7.8 compares `SPade` with `SPade-real` and `SPade-old` for computing A^t. Figure 7.9 compares `SPade-Fre-real`, `SPade-Fre`, `SPade-2by2`, and `explog-Fre` for all values of t in the vector v (`rootm-Fre` is omitted due to its poor performance in the previous experiment). For these real problems, `SPade-real` and `SPade-Fre-real` show a significant improvement in accuracy over the algorithms working in complex arithmetic.

Finally, we mention that we used `SPade-Fre` with the block 1-norm estimator to estimate the condition numbers of matrices in the test set and found the estimate of $\|K_{x^t}(A)\|_1$ always to be within a factor 2 of the true quantity.

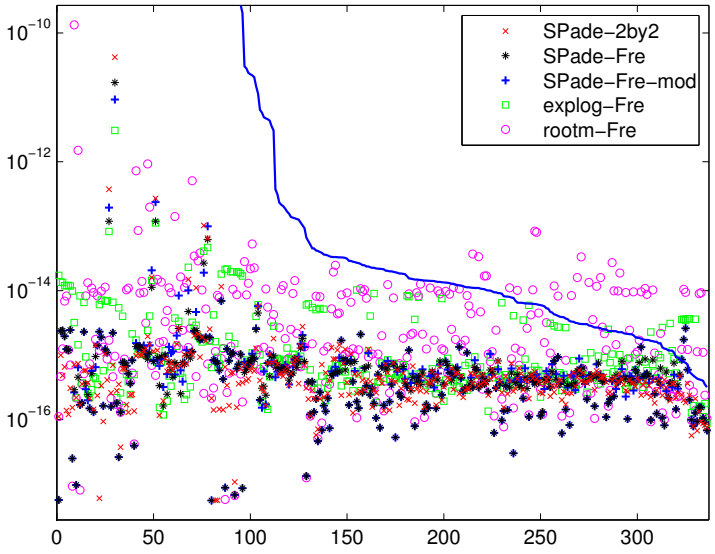

FIG. 7.4. *Experiment 2: relative errors in* $L_{x^t}(A,E)$ *for the problems with* $t = 1/q$ *for an integer* q.

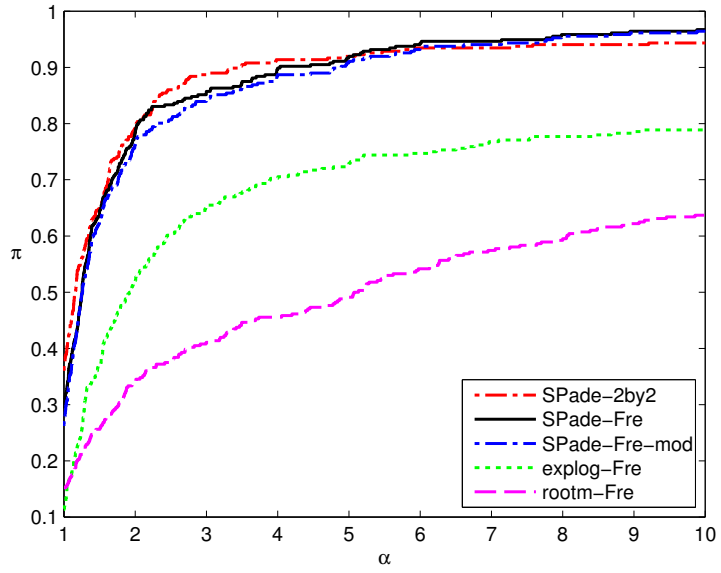

FIG. 7.5. *Experiment 2: performance profile for the data in Figure* 7.4.

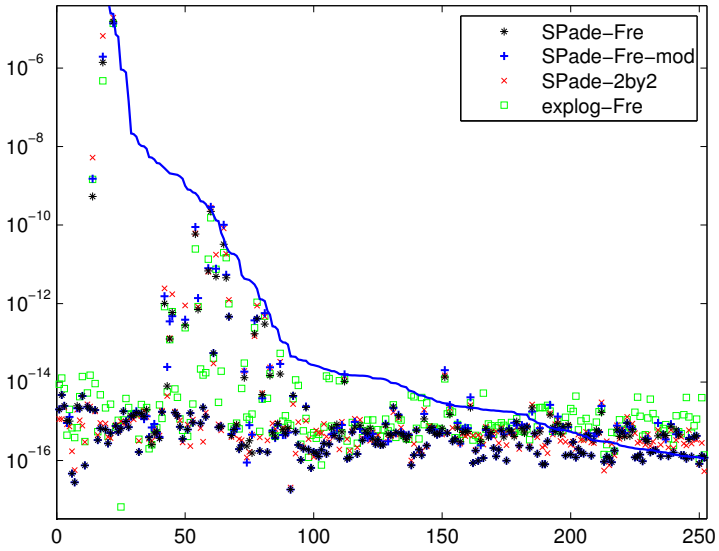

FIG. 7.6. *Experiment* 2*: relative errors in* $L_{x^t}(A,E)$ *for the problems where* $t \neq 1/q$ *for an integer* q*.*

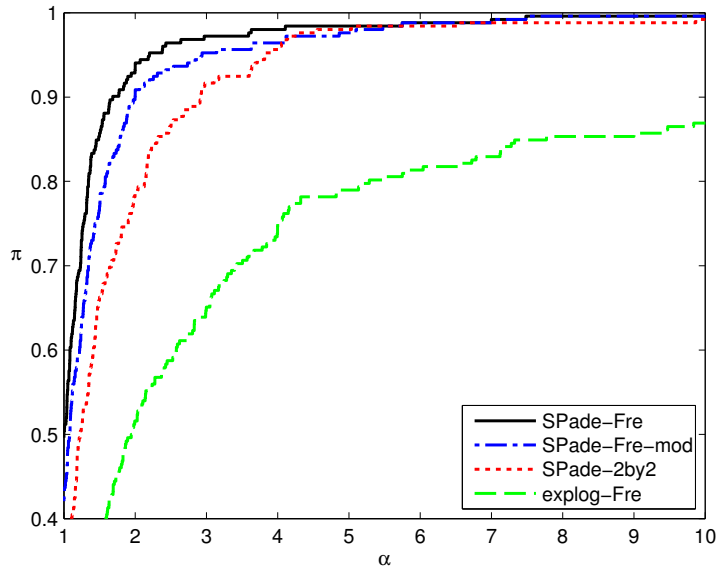

FIG. 7.7. *Experiment* 2*: performance profile for the data in Figure* 7.6.

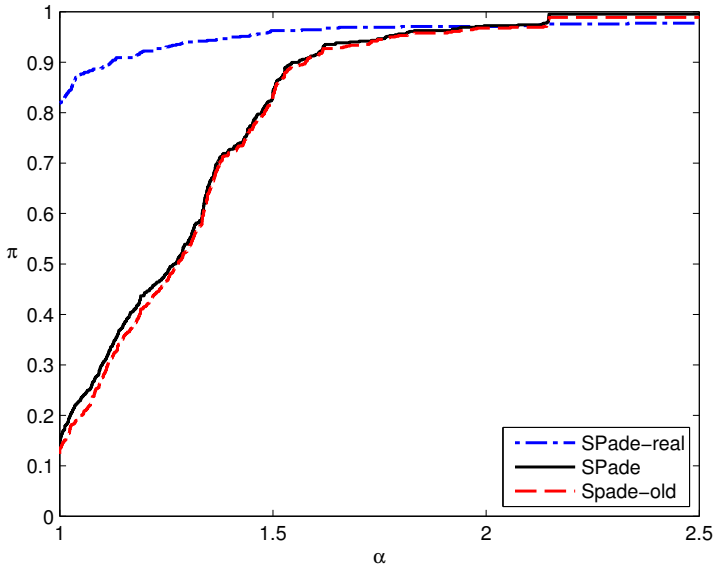

FIG. 7.8. *Experiment 3: performance profile for the relative errors in A^t for the real test problems.*

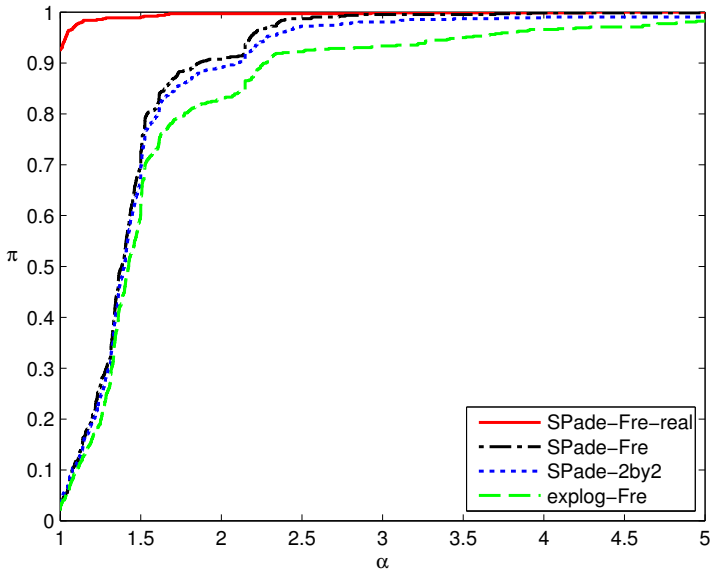

FIG. 7.9. *Experiment 3: performance profile for the relative errors in $L_{x^t}(A,E)$ for the real test problems.*

8. Conclusions. This work provides three main contributions.

1. *The improved Schur–Padé algorithm with sharper underlying error analysis.* Our experiments show that the improved algorithm is more accurate and often more efficient (by up to 40 percent for our test problems) than the original algorithm of [25].

2. *An extension of the improved Schur–Padé algorithm that also computes the Fréchet derivative.* We have shown this algorithm to be superior in accuracy to and at least as efficient as alternative techniques. The fact that the Fréchet derivative computation in Algorithm 4.2 is based on an error bound that is strictly valid only for the A^t computation itself (see section 4.1) does not affect the accuracy of the computed Fréchet derivatives in our experiments (as was found analogously for the matrix logarithm in [5]).

3. *The real arithmetic versions of the improved and extended algorithms for real data.* These bring a significant improvement in accuracy over the complex versions, run at twice the speed, and need only half the intermediate storage.

Finally, it is worth emphasizing that when $1/t = q \in \mathbb{Z}$, our algorithms are very competitive with algorithms specialized to the matrix qth root problem, as is shown here, in further tests we have conducted that are not reported here, and in [25], [30]. Moreover, our algorithms have an operation count independent of q, unlike most algorithms for the qth root problem [10], [20], [24, Chap. 7], [30], and this is significant for applications requiring a large q, such as the optic applications in [34] in which q can be as large as 10^5.

REFERENCES

[1] Awad H. Al-Mohy. A more accurate Briggs method for the logarithm. *Numer. Algorithms*, 59(3):393–402, 2012.

[2] Awad H. Al-Mohy and Nicholas J. Higham. Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM J. Matrix Anal. Appl.*, 30(4):1639–1657, 2009.

[3] 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.

[4] Awad H. Al-Mohy and Nicholas J. Higham. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM J. Sci. Comput.*, 34(4):C152–C169, 2012.

[5] Awad H. Al-Mohy, Nicholas J. Higham, and Samuel D. Relton. Computing the Fréchet derivative of the matrix logarithm and estimating the condition number. MIMS EPrint 2012.72, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, July 2012. Revised December 2012.

[6] George A. Baker, Jr. *Essentials of Padé Approximants.* Academic Press, New York, 1975.

[7] George A. Baker, Jr. and Peter Graves-Morris. *Padé Approximants*, volume 59 of *Encyclopedia of Mathematics and Its Applications.* Cambridge University Press, second edition, 1996.

[8] David A. Benson, Mark M. Meerschaert, and Jordan Revielle. Fractional calculus in hydrologic modeling: A numerical perspective. *Advances in Water Resources*, 51:479–497, 2013.

[9] Rajendra Bhatia and Mitsuru Uchiyama. The operator equation $\sum_{i=0}^{n} A^{n-i}XB^i = Y$. *Expositiones Mathematicae*, 27(3):251–255, 2009.

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

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

[12] K. Burrage, N. Hale, and D. Kay. An efficient implicit FEM scheme for fractional-in-space reaction-diffusion equations. *SIAM J. Sci. Comput.*, 34(4):A2145–A2172, 2012.

[13] João R. Cardoso. Evaluating the Fréchet derivative of the matrix pth root. *Electron. Trans. Numer. Anal.*, 38:202–217, 2011.

[14] João R. Cardoso. Computation of the matrix pth root and its Fréchet derivative by integrals. *Electron. Trans. Numer. Anal.*, 39:414–436, 2012.

[15] E. B. Davies. Approximate diagonalization. *SIAM J. Matrix Anal. Appl.*, 29(4):1051–1064, 2007.

[16] Edvin Deadman, Nicholas J. Higham, and Rui Ralha. Blocked Schur algorithms for computing the matrix square root. In P. Manninen and P. Öster, editors, *Applied Parallel and Scientific Computing: 11th International Conference, PARA 2012, Helsinki, Finland*, volume 7782 of *Lecture Notes in Computer Science*, pages 171–182. Springer-Verlag, Berlin, 2013.

[17] Nicholas J. Dingle and Nicholas J. Higham. Reducing the influence of tiny normwise relative errors on performance profiles. MIMS EPrint 2011.90, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, November 2011. Revised January 2013. To appear in ACM Trans. Math. Software.

[18] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Math. Programming*, 91:201–213, 2002.

[19] Oleksandr Gomilko, Federico Greco, and Krystyna Ziętak. A Padé family of iterations for the matrix sign function and related problems. *Numer. Linear Algebra Appl.*, 19(3):585–605, 2012.

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

[21] Desmond J. Higham and Nicholas J. Higham. *MATLAB Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2005.

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

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

[24] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[25] Nicholas J. Higham and Lijing Lin. A Schur–Padé algorithm for fractional powers of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(3):1056–1078, 2011.

[26] Nicholas J. Higham, D. Steven Mackey, Niloufer Mackey, and Françoise Tisseur. Functions preserving matrix groups and iterations for the matrix square root. *SIAM J. Matrix Anal. Appl.*, 26(3):849–877, 2005.

[27] Nicholas J. Higham and Samuel D. Relton. The condition number of the Fréchet derivative of a matrix function. MIMS EPrint, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2013. In preparation.

[28] Nicholas J. Higham and Françoise Tisseur. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM J. Matrix Anal. Appl.*, 21(4):1185–1201, 2000.

[29] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991.

[30] Bruno Iannazzo and Carlo Manasse. A Schur logarithmic algorithm for fractional powers of matrices. Technical report, May 2012. Manuscript.

[31] M. Ilić, I. W. Turner, and V. Anh. A numerical solution using an adaptively preconditioned Lanczos method for a class of linear systems related with the fractional Poisson equation. *J. App. Math. Stoch. Anal.*, 2008. Article ID 104525, 26 pages.

[32] Isak Jonsson and Bo Kågström. Recursive blocked algorithms for solving triangular systems—Part I: One-sided and coupled Sylvester-type matrix equations. *ACM Trans. Math. Software*, 28(4):392–415, 2002.

[33] Charles S. Kenney and Alan J. Laub. Padé error estimates for the logarithm of a matrix. *Internat. J. Control*, 50(3):707–730, 1989.

[34] H. D. Noble and R. A. Chipman. Mueller matrix roots algorithm and computational considerations. *Opt. Express*, 20(1):17–31, 2012.