SIAM J. MATRIX ANAL. APPL.
Vol. 30, No. 4, pp. 1639–1657

COMPUTING THE FRÉCHET DERIVATIVE OF THE MATRIX EXPONENTIAL, WITH AN APPLICATION TO CONDITION NUMBER ESTIMATION*

AWAD H. AL-MOHY† AND NICHOLAS J. HIGHAM†

Abstract. The matrix exponential is a much-studied matrix function having many applications. The Fréchet derivative of the matrix exponential describes the first-order sensitivity of e^A to perturbations in A and its norm determines a condition number for e^A. Among the numerous methods for computing e^A the scaling and squaring method is the most widely used. We show that the implementation of the method in [N. J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 1179–1193] can be extended to compute both e^A and the Fréchet derivative at A in the direction E, denoted by $L(A, E)$, at a cost about three times that for computing e^A alone. The algorithm is derived from the scaling and squaring method by differentiating the Padé approximants and the squaring recurrence, reusing quantities computed during the evaluation of the Padé approximant, and intertwining the recurrences in the squaring phase. To guide the choice of algorithmic parameters, an extension of the existing backward error analysis for the scaling and squaring method is developed which shows that, modulo rounding errors, the approximations obtained are $e^{A+\Delta A}$ and $L(A + \Delta A, E + \Delta E)$, with the same ΔA in both cases, and with computable bounds on $\|\Delta A\|$ and $\|\Delta E\|$. The algorithm for $L(A, E)$ is used to develop an algorithm that computes e^A together with an estimate of its condition number. In addition to results specific to the exponential, we develop some results and techniques for arbitrary functions. We show how a matrix iteration for $f(A)$ yields an iteration for the Fréchet derivative and show how to efficiently compute the Fréchet derivative of a power series. We also show that a matrix polynomial and its Fréchet derivative can be evaluated at a cost at most three times that of computing the polynomial itself and give a general framework for evaluating a matrix function and its Fréchet derivative via Padé approximation.

Key words. matrix function, Fréchet derivative, matrix polynomial, matrix iteration, matrix exponential, condition number estimation, scaling and squaring method, Padé approximation, backward error analysis

AMS subject classifications. 15A60, 65F30

DOI. 10.1137/080716426

1. Introduction.

The sensitivity of a matrix function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ to small perturbations is governed by the Fréchet derivative. The Fréchet derivative at a point $A \in \mathbb{C}^{n\times n}$ is a linear mapping

$$\begin{array}{rcl} \mathbb{C}^{n\times n} & \xrightarrow{L(A)} & \mathbb{C}^{n\times n} \\ E & \longmapsto & L(A, E) \end{array}$$

such that for all $E \in \mathbb{C}^{n\times n}$,

$$f(A + E) - f(A) - L(A, E) = o(\|E\|), \tag{1.1}$$

and it therefore describes the first-order effect on f of perturbations in A. If we need to show the dependence of L on f we will write $L_f(A, E)$.

*Received by the editors February 22, 2008; accepted for publication (in revised form) by M. H. Gutknecht August 26, 2008; published electronically January 23, 2009.
http://www.siam.org/journals/simax/30-4/71642.html

†School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (almohy@maths.manchester.ac.uk, http://www.maths.manchester.ac.uk/~almohy, higham@ma.man.ac.uk, http://www.ma.man.ac.uk/~higham). The work of the second author was supported by a Royal Society-Wolfson Research Merit Award and by Engineering and Physical Sciences Research Council grant EP/D079403.

It is desirable to be able to evaluate efficiently both $f(A)$ and the Fréchet derivative in order to obtain sensitivity information or to apply an optimization algorithm requiring derivatives. However, while the numerical computation of matrix functions is quite well developed, fewer methods are available for the Fréchet derivative, and the existing methods for $L(A,E)$ usually do not fully exploit the fact that $f(A)$ is being computed [6].

The norm of the Fréchet derivative yields a condition number [6, Theorem 3.1]:

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

where

$$\|L(A)\| := \max_{Z\neq 0} \frac{\|L(A,Z)\|}{\|Z\|} \tag{1.3}$$

and the norm is any matrix norm. When evaluating $f(A)$ we would like to be able to efficiently estimate $\mathrm{cond}(f,A)$; (1.3) shows that to do so we need to approximately maximize the norm of $L(A,Z)$ over all Z of unit norm.

The main aim of this work is to develop an efficient algorithm for simultaneously computing e^A and $L(A,E)$ and to use it to construct an algorithm for computing e^A along with an estimate of $\mathrm{cond}(\exp,A)$. The need for such algorithms is demonstrated by a recent paper in econometrics [8] in which the authors state that "One problem we did discover, that has not been accentuated in the literature, is that altering the stability properties of the coefficient matrix through a change in just one parameter can dramatically alter the theoretical and computed matrix exponential." If $A = A(t)$ depends smoothly on a vector $t \in \mathbb{C}^p$ of parameters then the change in e^A induced by small changes θh in t ($\theta \in \mathbb{C}$, $h \in \mathbb{C}^p$) is approximated by $\theta L(A, \sum_{i=1}^p h_i\, \partial A(t)/\partial t_i)$, since

$$\begin{aligned} f(A(t+\theta h)) &= f\left(A + \theta \sum_{i=1}^p \frac{\partial A(t)}{\partial t_i} h_i + O(\theta^2)\right) \\ &= f(A) + L\left(A, \theta \sum_{i=1}^p \frac{\partial A(t)}{\partial t_i} h_i + O(\theta^2)\right) + o(\theta) \\ &= f(A) + \theta L\left(A, \sum_{i=1}^p \frac{\partial A(t)}{\partial t_i} h_i\right) + o(\theta). \end{aligned}$$

Thus a single Fréchet derivative evaluation with $h = e_j$ (the jth unit vector) provides the information that the authors of [8] needed about the effect of changing a single parameter t_j.

We begin in section 2 by recalling a useful connection between the Fréchet derivative of a function and the same function evaluated at a certain block triangular matrix. We illustrate how this relation can be used to derive new iterations for computing $L(A,E)$ given an iteration for $f(A)$. Then in section 3 we show how to efficiently evaluate the Fréchet derivative when f has a power series expansion, by exploiting a convenient recurrence for the Fréchet derivative of a monomial. In section 4 we show that under reasonable assumptions a matrix polynomial and its Fréchet derivative can both be evaluated at a cost at most three times that of evaluating the polynomial itself. Then in section 5 we show how to evaluate the Fréchet derivative of a

rational function and give a framework for evaluating f and its Fréchet derivative via Padé approximants. In section 6 we apply this framework to the scaling and squaring algorithm for e^A [14], [17], and in particular to the implementation of Higham [5], which is the basis of MATLAB's `expm` function. We extend Higham's analysis to show that, modulo rounding errors, the approximations obtained from the new algorithm are $e^{A+\Delta A}$ and $L(A+\Delta A, E+\Delta E)$, with the *same* ΔA in both cases—a genuine backward error result. The computable bounds on $\|\Delta A\|$ and $\|\Delta E\|$ enable us to choose the algorithmic parameters in an optimal fashion. The new algorithm is shown to have significant advantages over existing ones. In section 7 we combine the new algorithm for $L(A,E)$ with an existing matrix 1-norm estimator to develop an algorithm for computing both e^A and an estimate of its condition number, and we show experimentally that the condition estimate can provide a useful guide to the accuracy of the scaling and squaring algorithm. Some concluding remarks are given in section 8.

2. Fréchet derivative via function of block triangular matrix. The following result shows that the Fréchet derivative appears as the $(1,2)$ block when f is evaluated at a certain block triangular matrix. Let $\mathcal{D}$ denote an open subset of $\mathbb{R}$ or $\mathbb{C}$.

THEOREM 2.1. *Let f be $2n-1$ times continuously differentiable on $\mathcal{D}$ and let the spectrum of X lie in $\mathcal{D}$. Then*

$$f\left(\begin{bmatrix} X & E \\ 0 & X \end{bmatrix}\right) = \begin{bmatrix} f(X) & L(X,E) \\ 0 & f(X) \end{bmatrix}. \tag{2.1}$$

Proof. See Mathias [13, Theorem 2.1] or Higham [6, section 3.1]. The result is also proved by Najfeld and Havel [15, Theorem 4.11] under the assumption that f is analytic. ∎

The significance of Theorem 2.1 is that given a smooth enough f and any method for computing $f(A)$, we can compute the Fréchet derivative by applying the method to the $2n \times 2n$ matrix in (2.1). The doubling in size of the problem is unwelcome, but if we exploit the block structure the computational cost can be reduced. Moreover, the theorem can provide a simple means to derive, and prove the convergence of, iterations for computing the Fréchet derivative.

To illustrate the use of the theorem we consider the principal square root function, $f(A) = A^{1/2}$, which for $A \in \mathbb{C}^{n\times n}$ with no eigenvalues on $\mathbb{R}^-$ (the closed negative real axis) is the unique square root X of A whose spectrum lies in the open right half-plane. The Denman–Beavers iteration

$$\begin{aligned} X_{k+1} &= \frac{1}{2}\left(X_k + Y_k^{-1}\right), \qquad X_0 = A, \\ Y_{k+1} &= \frac{1}{2}\left(Y_k + X_k^{-1}\right), \qquad Y_0 = I \end{aligned} \tag{2.2}$$

is a Newton variant that converges quadratically with [6, section 6.3]

$$\lim_{k\to\infty} X_k = A^{1/2}, \qquad \lim_{k\to\infty} Y_k = A^{-1/2}. \tag{2.3}$$

It is easy to show that if we apply the iteration to $\widetilde{A} = \left[\begin{smallmatrix} A & E \\ 0 & A \end{smallmatrix}\right]$ then iterates $\widetilde{X}_k$ and $\widetilde{Y}_k$ are produced for which

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

where

$$F_{k+1} = \frac{1}{2}\left(F_k - Y_k^{-1} G_k Y_k^{-1}\right), \qquad F_0 = E,$$
$$G_{k+1} = \frac{1}{2}\left(G_k - X_k^{-1} F_k X_k^{-1}\right), \qquad G_0 = 0. \tag{2.4}$$

By applying (2.3) and Theorem 2.1 to $\widetilde{A}$ we conclude that

$$\lim_{k\to\infty} F_k = L_{x^{1/2}}(A, E), \qquad \lim_{k\to\infty} G_k = L_{x^{-1/2}}(A, E). \tag{2.5}$$

Moreover, scaling strategies for accelerating the convergence of (2.2) [6, section 6.5] yield corresponding strategies for (2.4).

The next result shows quite generally that differentiating a fixed point iteration for a matrix function yields a fixed point iteration for the Fréchet derivative.

THEOREM 2.2. *Let f and g be $n-1$ times continuously differentiable on $\mathcal{D}$. Suppose that for any matrix $X \in \mathbb{C}^{n\times n}$ whose spectrum lies in $\mathcal{D}$, g has the fixed point $f(X)$, that is, $f(X) = g(f(X))$. Then for any such X, L_g at $f(X)$ has the fixed point $L_f(X, E)$ for all E.*

Proof. Applying the chain rule to $f(X) \equiv g(f(X))$ gives the relation $L_f(X, E) = L_g(f(X), L_f(X, E))$, which is the result. ∎

The iteration (2.4) for computing the Fréchet derivative of the square root function is new, and other new iterations for the Fréchet derivative of the matrix square root and related functions can be derived, and their convergence proved, in the same way, or directly by using Theorem 2.2. In the case of the Newton iteration for the matrix sign function this approach yields an iteration for the Fréchet derivative proposed by Kenney and Laub [10, Theorem 3.3] (see also [6, Theorem 5.7]) and derived using Theorem 2.1 by Mathias [13].

In the rest of this paper we consider the situation in which the underlying method for computing $f(A)$ is based on direct approximation rather than iteration, and we develop techniques that are more sophisticated than a direct application of Theorem 2.1.

3. Fréchet derivative via power series. When f has a power series expansion the Fréchet derivative can be expressed as a related series expansion.

THEOREM 3.1. *Suppose f has the power series expansion $f(x) = \sum_{k=0}^{\infty} a_k x^k$ with radius of convergence r. Then for $A, E \in \mathbb{C}^{n\times n}$ with $\|A\| < r$, the Fréchet derivative*

$$L_f(A, E) = \sum_{k=1}^{\infty} a_k \sum_{j=1}^{k} A^{j-1} E A^{k-j}. \tag{3.1}$$

Proof. See [6, Problem 3.6]. ∎

The next theorem gives a recurrence that can be used to evaluate (3.1).

THEOREM 3.2. *Under the assumptions of Theorem* 3.1*,*

$$L_f(A, E) = \sum_{k=1}^{\infty} a_k M_k, \tag{3.2}$$

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

$$M_k = M_{\ell_1} A^{\ell_2} + A^{\ell_1} M_{\ell_2}, \qquad M_1 = E, \tag{3.3}$$

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

$$M_k = M_{k-1}A + A^{k-1}M_1, \qquad M_1 = E. \tag{3.4}$$

In addition,

$$\|f(A)\| \le \widetilde{f}(\|A\|), \qquad \|L_f(A)\| \le \widetilde{f}'(\|A\|), \tag{3.5}$$

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

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

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

One way to develop the recurrence (3.3) is by applying Theorem 2.1 to the monomial $x^k = x^{\ell_1+\ell_2}$. A more direct approach is to use the product rule for Fréchet derivatives [6, Theorem 3.3] to obtain

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

Taking $\ell_1 = k-1$ and $\ell_2 = 1$ gives (3.4). It is straightforward to see that $\|f(A)\| \le \widetilde{f}(\|A\|)$. Taking norms in (3.1) gives

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

and maximizing over all nonzero E gives $\|L_f(A)\| \le \widetilde{f}'(\|A\|)$. ∎

The recurrence (3.3) will prove very useful in the rest of the paper.

4. Cost analysis for polynomials. Practical methods for approximating $f(A)$ may truncate a Taylor series to a polynomial or use a rational approximation. Both cases lead to the need to evaluate both a polynomial and its Fréchet derivative at the same argument. The question arises "what is the extra cost of computing the Fréchet derivative?" Theorem 3.2 does not necessarily answer this question because it only describes one family of recurrences for evaluating the Fréchet derivative. Moreover, the most efficient polynomial evaluation schemes are based on algebraic rearrangements that avoid explicitly forming all the matrix powers. Does an efficient evaluation scheme for a polynomial p also yield an efficient evaluation scheme for L_p?

Consider schemes for evaluating $p_m(X)$, where p_m is a polynomial of degree m and $X \in \mathbb{C}^{n\times n}$, that consist of s steps of the form

$$q_1^{(k)}(X) = q_2^{(k-1)}(X)\, q_3^{(k-1)}(X) + q_4^{(k-1)}(X), \qquad k = 1\colon s, \tag{4.1a}$$

$$\deg q_i^{(k)} < m, \quad i = 1\colon 4,\ k < s, \qquad \deg q_i^{(k)} \ge 1, \quad i = 2\colon 3, \tag{4.1b}$$

where the q_i are polynomials, $q_i^{(k)}$, $i = 2\colon 4$, is a linear combination of $q_1^{(1)}, \dots, q_1^{(k-1)}$, and $p_m(X) = q_1^{(s)}(X)$. This class contains all schemes of practical interest, which include Horner's method, evaluation by explicit powers, and the Paterson and Stockmeyer method [16] (all of which are described in [6, section 4.2]), as well as more ad hoc schemes such as those described below. We measure the cost of the scheme by the

number of matrix multiplications it requires. The next result shows that the overhead of evaluating the Fréchet derivative is at most twice the original cost.

THEOREM 4.1. *Let p be a polynomial and let π_p denote the cost of evaluating $p(X)$ by any scheme of the form* (4.1). *Let σ_p denote the extra cost required to compute $L_p(X,E)$ by using the scheme obtained by differentiating the scheme for $p(X)$. Then $\sigma_p \le 2\pi_p$.*

Proof. The proof is by induction on the degree m of the polynomial. For $m = 1$, $p_1(x) = b_0 + b_1 x$ and the only possible scheme is the obvious evaluation $p_1(X) = b_0 I + b_1 X$ with $\pi_1 = 0$. The corresponding Fréchet derivative scheme is $L_{p_1}(X,E) = b_1 E$ and $\sigma_1 = 0$, so the result is trivially true for $m = 1$. Suppose the result is true for all polynomials of degree at most $m-1$ and consider a polynomial p_m of degree m. By (4.1) the last stage of the scheme can be written $p_m(X) = q_2^{(s-1)}(X) q_3^{(s-1)}(X) + q_4^{(s-1)}(X)$, where the polynomials $q_i \equiv q_i^{(s-1)}$, $i = 2\colon 4$ are all of degree less than m. Note that $\pi_{p_m} = \pi_{q_2} + \pi_{q_3} + \pi_{q_4} + 1$ and by the inductive hypothesis, $\sigma_{q_i} \le 2\pi_{q_i}$, $i = 2\colon 4$. Now $L_{p_m}(X,E) = L_{q_2}(X,E) q_3(X) + q_2(X) L_{q_3}(X,E) + L_{q_4}(X,E)$ by the product rule and so

$$\sigma_{p_m} \le \sigma_{q_2} + \sigma_{q_3} + \sigma_{q_4} + 2 \le 2(\pi_{q_2} + \pi_{q_3} + \pi_{q_4} + 1) = 2\pi_{p_m},$$

as required. This proof tacitly assumes that there are no dependencies between the $q_i^{(k)}$ that reduce the cost of evaluating p, for example, $q_2^{(s-1)} = q_3^{(s-1)}$. However, any dependencies equally benefit the L_p evaluation and the result remains valid. ∎

To illustrate the theorem, consider the polynomial $p(X) = I + X + X^2 + X^3 + X^4 + X^5$. Rewriting it as

$$p(X) = I + X\left(I + X^2 + X^4\right) + X^2 + X^4,$$

we see that $p(X)$ can be evaluated in just three multiplications via $X_2 = X^2$, $X_4 = X_2^2$, and $p(X) = I + X(I + X_2 + X_4) + X_2 + X_4$. Differentiating gives

$$\begin{aligned} L_p(X,E) &= L_{x(1+x^2+x^4)}(X,E) + M_2 + M_4 \\ &= E(I + X_2 + X_4) + X(M_2 + M_4) + M_2 + M_4, \end{aligned}$$

where $M_2 = XE + EX$ and $M_4 = M_2 X_2 + X_2 M_2$ by (3.3). Hence the Fréchet derivative can be evaluated with six additional multiplications, and the total cost is nine multiplications.

5. Computational framework. For a number of important functions f, such as the exponential, the logarithm, and the sine and cosine, successful algorithms for $f(A)$ have been built on the use of Padé approximants: a Padé approximant r_m of f of suitable degree m is evaluated at a transformed version of A and the transformation is then undone. Here, $r_m(x) = p_m(x)/q_m(x)$ with p_m and q_m polynomials of degree m such that $f(x) - r_m(x) = O(x^{2m+1})$ [2]. It is natural to make use of this Padé approximant by approximating L_f by the Fréchet derivative L_{r_m} of r_m. The next result shows how to evaluate L_{r_m}.

LEMMA 5.1. *The Fréchet derivative L_{r_m} of the rational function $r_m(x) = p_m(x)/q_m(x)$ satisfies*

$$q_m(A) L_{r_m}(A,E) = L_{p_m}(A,E) - L_{q_m}(A,E) r_m(A). \tag{5.1}$$

Proof. Applying the Fréchet derivative product rule to $q_m r_m = p_m$ gives

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

which rearranges to the result. ∎

We can now state a general framework for simultaneously approximating $f(A)$ and $L_f(A,E)$ in a way that reuses matrix multiplications from the approximation of f in the approximation of L_f.

1. Choose a suitable Padé degree m and transformation function g and set $A \leftarrow g(A)$.
2. Devise efficient schemes for evaluating $p_m(A)$ and $q_m(A)$.
3. Fréchet differentiate the schemes in the previous step to obtain schemes for evaluating $L_{p_m}(A,E)$ and $L_{q_m}(A,E)$. Use the recurrences (3.3) and (3.4) as necessary.
4. Solve $q_m(A)r_m(A) = p_m(A)$ for $r_m(A)$.
5. Solve $q_m(A)L_{r_m}(A,E) = L_{p_m}(A,E) - L_{q_m}(A,E)r_m(A)$ for $L_{r_m}(A,E)$.
6. Apply the appropriate transformations to $r_m(A)$ and $L_{r_m}(A,E)$ that undo the effect of the initial transformation on A.

In view of Theorem 4.1, the cost of this procedure is at most $(3\pi_m + 1)M + 2D$, where $\pi_m M$ is the cost of evaluating both $p_m(A)$ and $q_m(A)$, and M and D denote a matrix multiplication and the solution of a matrix equation, respectively.

If we are adding the capability to approximate the Fréchet derivative to an existing Padé-based method for $f(A)$ then our attention will focus on step 1, where we must reconsider the choice of m and transformation to ensure that both f and L_f are approximated to sufficient accuracy.

In the next section we apply this framework to the matrix exponential.

6. Scaling and squaring algorithm for the exponential and its Fréchet derivative. The scaling and squaring method for computing the exponential of $A \in \mathbb{C}^{n\times n}$ is based on the relation

$$e^A = \left(e^{2^{-s}A}\right)^{2^s}. \tag{6.1}$$

For suitably chosen nonnegative integers s and m, this method approximates $e^{2^{-s}A}$ by $r_m(2^{-s}A)$, where r_m is the $[m/m]$ Padé approximant to e^x, and it takes $e^A \approx (r_m(2^{-s}A))^{2^s}$. A choice of the parameters s and m with a certain optimality property is given in the following algorithm from [5], [6, Algorithm 10.20], which forms the basis of MATLAB's `expm` function.

ALGORITHM 6.1 (scaling and squaring algorithm for exponential). *This algorithm evaluates the matrix exponential $X = e^A$ of $A \in \mathbb{C}^{n\times n}$ using the scaling and squaring method. It uses the parameters θ_m given in Table* 6.1. *The algorithm is intended for IEEE double precision arithmetic.*

```
1  for m = [3 5 7 9]
2      if ||A||_1 ≤ θ_m, evaluate X = r_m(A) using (6.11) and (6.14), quit, end
3  end
4  A ← 2^{-s}A with s = ⌈log_2(||A||_1/θ_13)⌉
5  Evaluate r_13(A) using (6.14) and the preceding equations.
6  X = r_13(A)^{2^s} by repeated squaring.
```

Cost: $(\pi_m + s)M + D$, where m is the degree of Padé approximant used and π_m (tabulated in [5, Table 2.2]) is the cost of evaluating p_m and q_m.

Our aim is now to adapt this algorithm to compute $L_{\exp}(A,E)$ along with e^A.

TABLE 6.1
Maximal values ℓ_m of $\|2^{-s}A\|$ such that the backward error bound (6.10) *does not exceed $u = 2^{-53}$, along with maximal values θ_m such that a bound for $\|\Delta A\|/\|A\|$ does not exceed u.*

m	1	2	3	4	5	6	7	8	9	10
θ_m	3.65e-8	5.32e-4	1.50e-2	8.54e-2	2.54e-1	5.41e-1	9.50e-1	1.47e0	2.10e0	2.81e0
ℓ_m	2.11e-8	3.56e-4	1.08e-2	6.49e-2	2.00e-1	4.37e-1	7.83e-1	1.23e0	1.78e0	2.42e0

m	11	12	13	14	15	16	17	18	19	20
θ_m	3.60e0	4.46e0	5.37e0	6.33e0	7.34e0	8.37e0	9.44e0	1.05e1	1.17e1	1.28e1
ℓ_m	3.13e0	3.90e0	4.74e0	5.63e0	6.56e0	7.52e0	8.53e0	9.56e0	1.06e1	1.17e1

A recurrence for the Fréchet derivative of the exponential can be obtained by differentiating (6.1). Note first that differentiating the identity $e^A = (e^{A/2})^2$ using the chain rule [6, Theorem 3.4] along with $L_{x^2}(A,E) = AE + EA$ gives the relation

$$\begin{aligned} L_{\exp}(A,E) &= L_{x^2}\big(e^{A/2}, L_{\exp}(A/2,E/2)\big) \\ &= e^{A/2}L_{\exp}(A/2,E/2) + L_{\exp}(A/2,E/2)e^{A/2}. \end{aligned} \tag{6.2}$$

Repeated use of this relation leads to the recurrence

$$\begin{aligned} \widetilde{L}_s &= L_{\exp}(2^{-s}A, 2^{-s}E), \\ \widetilde{L}_{i-1} &= e^{2^{-i}A}\widetilde{L}_i + \widetilde{L}_i e^{2^{-i}A}, \qquad i = s\colon -1\colon 1 \end{aligned} \tag{6.3}$$

for $\widetilde{L}_0 = L_{\exp}(A,E)$. Our numerical method replaces $L_{\exp}$ by L_{r_m} and $e^{2^{-i}A}$ by $r_m(2^{-s}A)^{2^{s-i}}$, producing approximations L_i to $\widetilde{L}_i$:

$$\begin{aligned} X_s &= r_m(2^{-s}A), \\ L_s &= L_{r_m}(2^{-s}A, 2^{-s}E), \\ \left.\begin{aligned} L_{i-1} &= X_i L_i + L_i X_i \\ X_{i-1} &= X_i^2 \end{aligned}\right\} & \quad i = s\colon -1\colon 1. \end{aligned} \tag{6.4}$$

The key question is what can be said about the accuracy or stability of L_0 relative to that of the approximation $X_0 = (r_m(2^{-s}A))^{2^s}$ to e^A. To answer this question we recall the key part of the error analysis from [5] (see also [6, section 10.3]), which is summarized in the following result. We denote by log the principal logarithm of $A \in \mathbb{C}^{n\times n}$, which is defined for A with no eigenvalues on $\mathbb{R}^-$ and is the unique logarithm whose eigenvalues all have imaginary parts in $(-\pi,\pi)$.

THEOREM 6.2. *Suppose that*

$$\|e^{-A}r_m(A) - I\| < 1, \qquad \|A\| < \min\{\,|t| : q_m(t) = 0\,\} \tag{6.5}$$

for some consistent matrix norm, so that $g_m(A) = \log(e^{-A}r_m(A))$ is guaranteed to be defined. Then $r_m(A) = e^{A+g_m(A)}$ and $\|g_m(A)\| \le -\log(1 - \|e^{-A}r_m(A) - I\|)$. In particular, if (6.5) *is satisfied with $A \leftarrow 2^{-s}A$ then*

$$r_m(2^{-s}A) = e^{2^{-s}A + g_m(2^{-s}A)}, \tag{6.6}$$

so that $r_m(2^{-s}A)^{2^s} = e^{A + 2^s g_m(2^{-s}A)}$, where

$$\frac{\|2^s g_m(2^{-s}A)\|}{\|A\|} \le \frac{-\log(1 - \|e^{-2^{-s}A}r_m(2^{-s}A) - I\|)}{\|2^{-s}A\|}. \tag{6.7}$$

∎

Differentiating (6.6) gives, using the chain rule,

$$\begin{aligned} L_s &= L_{r_m}(2^{-s}A, 2^{-s}E) \\ &= L_{\exp}\bigl(2^{-s}A + g_m(2^{-s}A), 2^{-s}E + L_{g_m}(2^{-s}A, 2^{-s}E)\bigr). \end{aligned} \tag{6.8}$$

From (6.4), (6.6), and (6.8),

$$\begin{aligned} L_{s-1} &= r_m(2^{-s}A)L_s + L_s r_m(2^{-s}A) \\ &= e^{2^{-s}A+g_m(2^{-s}A)}\, L_{\exp}\bigl(2^{-s}A + g_m(2^{-s}A), 2^{-s}E + L_{g_m}(2^{-s}A, 2^{-s}E)\bigr) \\ &\quad + L_{\exp}\bigl(2^{-s}A + g_m(2^{-s}A), 2^{-s}E + L_{g_m}(2^{-s}A, 2^{-s}E)\bigr)\, e^{2^{-s}A+g_m(2^{-s}A)} \\ &= L_{\exp}\bigl(2^{-(s-1)}A + 2g_m(2^{-s}A), 2^{-(s-1)}E + L_{g_m}(2^{-s}A, 2^{-(s-1)}E)\bigr), \end{aligned}$$

where we have used (6.2) and the fact that L is linear in its second argument. Continuing this argument inductively, and using

$$X_i = X_s^{2^{s-i}} = \bigl(e^{2^{-s}A+g_m(2^{-s}A)}\bigr)^{2^{s-i}} = e^{2^{-i}A+2^{s-i}g_m(2^{-s}A)},$$

we obtain the following result.

THEOREM 6.3. *If* (6.5) *is satisfied with* $A \leftarrow 2^{-s}A$ *then* L_0 *from* (6.4) *satisfies*

$$L_0 = L_{\exp}\bigl(A + 2^s g_m(2^{-s}A), E + L_{g_m}(2^{-s}A, E)\bigr). \qquad \square \tag{6.9}$$

Theorem 6.3 is a backward error result: it says that L_0 is the exact Fréchet derivative for the exponential of a perturbed matrix in a perturbed direction. We emphasize that the backward error is with respect to the effect of truncation errors in the Padé approximation, not to rounding errors, which for the moment are ignored.

Theorems 6.2 and 6.3 show that $X_0 = e^{A+\Delta A}$ and $L_0 = L_{\exp}(A + \Delta A, E + \Delta E)$ with the *same* $\Delta A = 2^s g_m(2^{-s}A)$. We already know from the analysis in [5] how to choose s and m to keep ΔA acceptably small. It remains to investigate the norm of $\Delta E = L_{g_m}(2^{-s}A, E)$.

Let $\widetilde{g}_m(x) = \sum_{k=2m+1}^{\infty} c_k x^k$ be the power series resulting from replacing the coefficients of the power series expansion of the function $g_m(x) = \log(e^{-x} r_m(x))$ by their absolute values. Using the second bound in (3.5) we have

$$\frac{\|\Delta E\|}{\|E\|} = \frac{\|L_{g_m}(2^{-s}A, E)\|}{\|E\|} \le \|L_{g_m}(2^{-s}A)\| \le \widetilde{g}_m{}'(\theta), \tag{6.10}$$

where $\theta = \|2^{-s}A\|$. Define $\ell_m = \max\{\, \theta : \widetilde{g}_m{}'(\theta) \le u \,\}$, where $u = 2^{-53} \approx 1.1 \times 10^{-16}$ is the unit roundoff for IEEE double precision arithmetic. Using MATLAB's Symbolic Math Toolbox we evaluated ℓ_m, $m = 1\colon 20$, by summing the first 150 terms of the series symbolically in 250 decimal digit arithmetic. Table 6.1 shows these values along with analogous values θ_m calculated in [5], which are the maximal values of θ such that a bound on $\|\Delta A\|/\|A\|$ obtained from (6.7) does not exceed u. In every case $\ell_m < \theta_m$, which is not surprising given that we are approximating L_{r_m} by an approximation chosen for computational convenience rather than its approximation properties, but

the ratio θ_m/ℓ_m is close to 1. For each m, if $\theta \le \ell_m$ then we are assured that

$$X_0 = e^{A+\Delta A}, \quad L_0 = L_{\exp}(A+\Delta A, E+\Delta E), \quad \|\Delta A\| \le u\|A\|, \quad \|\Delta E\| \le u\|E\|;$$

in other words, perfect backward stability is guaranteed for such θ.

In order to develop an algorithm we now need to look at the cost of evaluating $r_m = p_m/q_m$ and L_{r_m}, where r_m is the $[m/m]$ Padé approximant to e^x. Higham [5] shows how to efficiently evaluate $p_m(A)$ and $q_m(A)$ by using one type of scheme for $m \le 11$ and another for $m \ge 12$; the number of matrix multiplications, π_m, required to compute $p_m(A)$ and $q_m(A)$ is given in [5, Table 2.2]. As Theorem 4.1 suggests, the Fréchet derivatives L_{p_m} and L_{q_m} can be calculated at an extra cost of $2\pi_m$ multiplications by differentiating the schemes for p_m and q_m. We now give the details.

We consider the odd degree Padé approximants to the exponential function. Analogous techniques apply to the even degree approximants (which, as in Algorithm 6.1, it will turn out we do not need). For $m = 3, 5, 7, 9$, we decompose $p_m = \sum_{i=0}^{m} b_i x^i$ into its odd and even parts:

$$p_m(x) = x \sum_{k=0}^{(m-1)/2} b_{2k+1} x^{2k} + \sum_{k=0}^{(m-1)/2} b_{2k} x^{2k} =: u_m(x) + v_m(x). \tag{6.11}$$

It follows that $q_m(x) = -u_m(x) + v_m(x)$ since $q_m(x) = p_m(-x)$, and hence

$$L_{p_m} = L_{u_m} + L_{v_m}, \qquad L_{q_m} = -L_{u_m} + L_{v_m}.$$

We obtain $L_{u_m}(A,E)$ and $L_{v_m}(A,E)$ by differentiating u_m and v_m, respectively:

$$L_{u_m}(A,E) = A \sum_{k=1}^{(m-1)/2} b_{2k+1} M_{2k} + E \sum_{k=0}^{(m-1)/2} b_{2k+1} A^{2k} \tag{6.12}$$

$$L_{v_m}(A,E) = \sum_{k=1}^{(m-1)/2} b_{2k} M_{2k}. \tag{6.13}$$

The $M_k = L_{x^k}(A,E)$ are computed using (3.3).

For $m = 13$ it is more efficient to use the odd-even splitting $p_{13} = u_{13} + v_{13}$, where

$$\begin{aligned} u_{13}(x) &= xw(x), \quad w(x) = x^6 w_1(x) + w_2(x), \quad v_{13}(x) = x^6 z_1(x) + z_2(x), \\ w_1(x) &= b_{13}x^6 + b_{11}x^4 + b_9 x^2, \quad w_2(x) = b_7 x^6 + b_5 x^4 + b_3 x^2 + b_1, \\ z_1(x) &= b_{12}x^6 + b_{10}x^4 + b_8 x^2, \quad z_2(x) = b_6 x^6 + b_4 x^4 + b_2 x^2 + b_0. \end{aligned}$$

Differentiating these polynomials yields

$$\begin{aligned} L_{u_{13}}(A,E) &= AL_w(A,E) + Ew(A), \\ L_{v_{13}}(A,E) &= A^6 L_{z_1}(A,E) + M_6 z_1(A) + L_{z_2}(A,E), \end{aligned}$$

where

$$\begin{aligned} L_w(A,E) &= A^6 L_{w_1}(A,E) + M_6 w_1(A) + L_{w_2}(A,E), \\ L_{w_1}(A,E) &= b_{13}M_6 + b_{11}M_4 + b_9 M_2, \\ L_{w_2}(A,E) &= b_7 M_6 + b_5 M_4 + b_3 M_2, \\ L_{z_1}(A,E) &= b_{12}M_6 + b_{10}M_4 + b_8 M_2, \\ L_{z_2}(A,E) &= b_6 M_6 + b_4 M_4 + b_2 M_2. \end{aligned}$$

TABLE 6.2
Number of matrix multiplications, ω_m, required to evaluate $r_m(A)$ and $L_{r_m}(A,E)$, and measure of overall cost C_m in (6.17).

m	1	2	3	4	5	6	7	8	9	10
ω_m	1	4	7	10	10	13	13	16	16	19
C_m	25.5	12.5	8.5	6.9	5.3	5.2	4.4	4.7	4.2	4.7

m	11	12	13	14	15	16	17	18	19	20
ω_m	19	19	19	22	22	22	22	25	25	25
C_m	4.4	4.0	3.8	4.5	4.3	4.1	3.9	4.7	4.6	4.5

Then $L_{p_{13}} = L_{u_{13}} + L_{v_{13}}$ and $L_{q_{13}} = -L_{u_{13}} + L_{v_{13}}$. We finally solve for $r_m(A)$ and $L_{r_m}(A,E)$ the equations

$$(-u_m + v_m)(A)r_m(A) = (u_m + v_m)(A), \tag{6.14}$$

$$(-u_m + v_m)(A)L_{r_m}(A,E) = (L_{u_m} + L_{v_m})(A,E) + (L_{u_m} - L_{v_m})(A,E)r_m(A). \tag{6.15}$$

We are now in a position to choose the degree m and the scaling parameter s. Table 6.2 reports the total number of matrix multiplications, $\omega_m = 3\pi_m + 1$, necessary to evaluate r_m and L_{r_m} for a range of m, based on [5, Table 2.2] and the observations above. In evaluating the overall cost we need to take into account the squaring phase. If $\|A\| > \ell_m$ then in order to use the $[m/m]$ Padé approximant we must scale A by 2^{-s} so that $\|2^{-s}A\| \le \ell_m$, that is, we need $s = \lceil \log_2(\|A\|/\ell_m) \rceil$. From the recurrence (6.4), we see that $3s$ matrix multiplications are added to the cost of evaluating r_m and L_{r_m}. Thus the overall cost in matrix multiplications is

$$\omega_m + 3s = 3\pi_m + 1 + 3\max(\lceil \log_2 \|A\| - \log_2 \ell_m \rceil, 0). \tag{6.16}$$

To minimize the cost we therefore choose m to minimize the quantity

$$C_m = \pi_m - \log_2 \ell_m, \tag{6.17}$$

where we have dropped the constant terms and factors in (6.16). Table 6.2 reports the C_m values. The table shows that $m = 13$ is the optimal choice, just as it is for the scaling and squaring method for the exponential itself [5]. The ω_m values also show that only $m = 1, 2, 3, 5, 7, 9$ need be considered if $\|A\| < \ell_{13}$. As in [5] we rule out $m = 1$ and $m = 2$ on the grounds of possible loss of significant figures in floating point arithmetic.

It remains to check that the evaluation of L_{p_m}, L_{q_m}, and L_{r_m} is done accurately in floating point arithmetic. The latter matrix is evaluated from (6.15), which involves solving a matrix equation with coefficient matrix $q_m(A)$, just as in the evaluation of r_m, and the analysis from [5] guarantees that $q_m(A)$ is well conditioned for the scaled A. It can be shown that for our schemes for evaluating L_{p_m} we have

$$\|L_{p_m}(A,E) - fl(L_{p_m}(A,E))\|_1 \le \widetilde{\gamma}_{n^2} p'_m(\|A\|_1)\|E\|_1 \approx \widetilde{\gamma}_{n^2} e^{\|A\|_1/2}\|E\|_1,$$

where we have used the facts that p_m has positive coefficients and $p_m(x) \approx e^{x/2}$. Here, $\widetilde{\gamma}_k = cku/(1-cku)$, where c denotes a small integer constant. At least in an absolute sense, this bound is acceptable for $\|A\| \le \ell_{13}$. An entirely analogous bound can be obtained for L_{q_m}, since $q_m(x) = p_m(-x)$.

We now state the complete algorithm.

ALGORITHM 6.4 (scaling and squaring algorithm for exponential and Fréchet derivative). *Given $A, E \in \mathbb{C}^{n\times n}$ this algorithm computes $R = e^A$ and $L = L_{\exp}(A, E)$ by a scaling and squaring algorithm. It uses the parameters ℓ_m listed in Table* 6.1. *The algorithm is intended for IEEE double precision arithmetic.*

```
1  for m = [3 5 7 9]
2      if ‖A‖_1 ≤ ℓ_m
3         Evaluate U = u_m(A) and V = v_m(A), using (6.11).
4         Evaluate L_u = L_{u_m}(A, E) and L_v = L_{v_m}(A, E), using (6.12) and (6.13).
5         s = 0; goto line 26
6      end
7  end
8  s = ⌈log_2(‖A‖_1/ℓ_13)⌉, the minimal integer such that ‖2^{-s}A‖_1 ≤ ℓ_13.
9  A ← 2^{-s}A and E ← 2^{-s}E
10 A_2 = A^2, A_4 = A_2^2, A_6 = A_2A_4
11 M_2 = AE + EA, M_4 = A_2M_2 + M_2A_2, M_6 = A_4M_2 + M_4A_2
12 W_1 = b_13A_6 + b_11A_4 + b_9A_2
13 W_2 = b_7A_6 + b_5A_4 + b_3A_2 + b_1I
14 Z_1 = b_12A_6 + b_10A_4 + b_8A_2
15 Z_2 = b_6A_6 + b_4A_4 + b_2A_2 + b_0I
16 W = A_6W_1 + W_2
17 U = AW
18 V = A_6Z_1 + Z_2
19 L_{w_1} = b_13M_6 + b_11M_4 + b_9M_2
20 L_{w_2} = b_7M_6 + b_5M_4 + b_3M_2
21 L_{z_1} = b_12M_6 + b_10M_4 + b_8M_2
22 L_{z_2} = b_6M_6 + b_4M_4 + b_2M_2
23 L_w = A_6L_{w_1} + M_6W_1 + L_{w_2}
24 L_u = AL_w + EW
25 L_v = A_6L_{z_1} + M_6Z_1 + L_{z_2}
26 Solve (−U + V)R = U + V for R.
27 Solve (−U + V)L = L_u + L_v + (L_u − L_v)R for L.
28 for k = 1: s
29     L ← RL + LR
30     R ← R^2
31 end
```

Cost: $(\omega_m + 3s)M + 2D$, where m is the degree of Padé approximant used and ω_m is given in Table 6.2. The linear systems at lines 26 and 27 have the same coefficient matrix, so an LU factorization can computed once and reused.

Since $L_{\exp}(A, \alpha E) = \alpha L_{\exp}(A, E)$, an algorithm for computing $L_{\exp}(A, E)$ should not be influenced in any significant way by $\|E\|$, and this is the case for Algorithm 6.4. Najfeld and Havel [15] propose computing $L_{\exp}(A, E)$ using their version of the scaling and squaring method for the exponential in conjunction with (2.1). With this approach E affects the amount of scaling, and overscaling results when $\|E\| \gg \|A\|$, while how to scale E to produce the most accurate result is unclear.

To assess the cost of Algorithm 6.4 we compare it with Algorithm 6.1 and with a "Kronecker–Sylvester scaling and squaring algorithm" of Kenney and Laub [11], which is based on a Kronecker sum representation of the Fréchet derivative. In the form detailed in [6, section 10.6.2], this latter algorithm scales to obtain $\|2^{-t}A\| \le 1$, evaluates the [8/8] Padé approximant to $\tanh(x)/x$ at the scaled Kronecker sum, and then uses the recurrence (6.4) or the variant (6.3) that explicitly computes $X_i = e^{2^{-i}A}$

in each step. It requires one matrix exponential, $(17 + 3t)M$, and the solution of 8 Sylvester equations if (6.4) is used, or s matrix exponentials, $(18 + 2t)M$, and the same number of Sylvester equation solutions if (6.3) is used.

To compare the algorithms, assume that the Padé degree $m = 13$ is used in Algorithms 6.1 and 6.4. Then Algorithm 6.4 costs $(19+3s)M+2D$ and Algorithm 6.1 costs $(6+s)M+D$. Two conclusions can be drawn. First, Algorithm 6.4 costs about three times as much as just computing e^A. Second, since the cost of solving a Sylvester equation is about $60n^3$ flops, which is the cost of 30 matrix multiplications, the Kronecker–Sylvester algorithm is an order of magnitude more expensive than Algorithm 6.4. To be more specific, consider the case where $\|A\| = 9$, so that $s = 1$ in Algorithms 6.1 and 6.4 and $t = 4$, and ignore the cost of computing the matrix exponential in the less expensive "squaring" variant of the Kronecker–Sylvester algorithm. Then the operation counts in flops are approximately $48n^3$ for Algorithm 6.4 (e^A and $L_{\exp}(A, E)$), $16n^3$ for Algorithm 6.1 (e^A only), and $538n^3$ for the Kronecker–Sylvester algorithm ($L_{\exp}(A, E)$ only). A further drawback of the Kronecker–Sylvester algorithm is that it requires complex arithmetic, so the effective flop count is even higher.

Other algorithms for $L_{\exp}(A, E)$ are those of Kenney and Laub [9] and Mathias [12] (see also [6, section 10.6.1]), which apply quadrature to an integral representation of the Fréchet derivative. These algorithms are intended only for low accuracy approximations and do not lend themselves to combination with Algorithm 6.1.

We describe a numerical experiment, modeled on that in [5], that tests the accuracy of Algorithm 6.4. We took 74 test matrices, which include some from MATLAB (in particular, from the `gallery` function), some from the Matrix Computation Toolbox [3], and test matrices from the e^A literature; most matrices are 10×10, with a few having smaller dimension. We evaluated the normwise relative errors of the computed Fréchet derivatives $L_{\exp}(A, E)$, using a different E, generated as `randn(n)`, for each A. The "exact" Fréchet derivative is obtained using (2.1) with the exponential evaluated at 100 digit precision via MATLAB's Symbolic Math Toolbox. Figure 6.1 displays

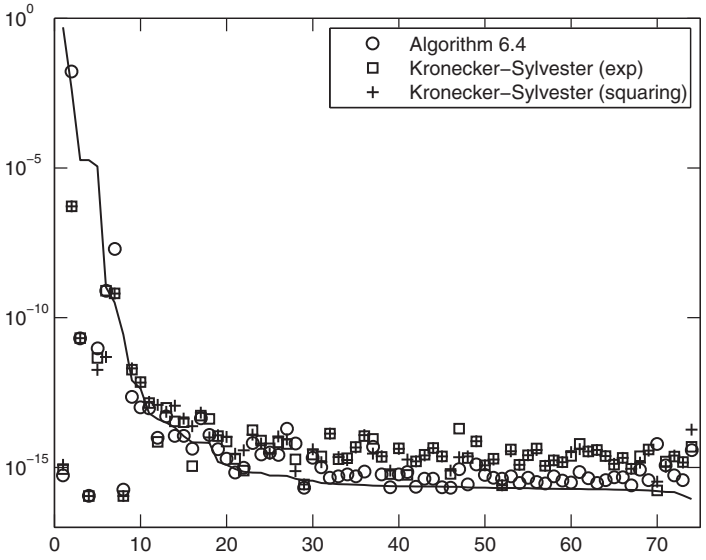

FIG. 6.1. *Normwise relative errors in Fréchet derivatives $L_{\exp}(A, E)$ computed by Algorithm 6.4 and two variants of the Kronecker–Sylvester algorithm for 74 matrices A with a different random E for each A, along with estimate of $\mathrm{cond}(L_{\exp}, A)u$ (solid line).*

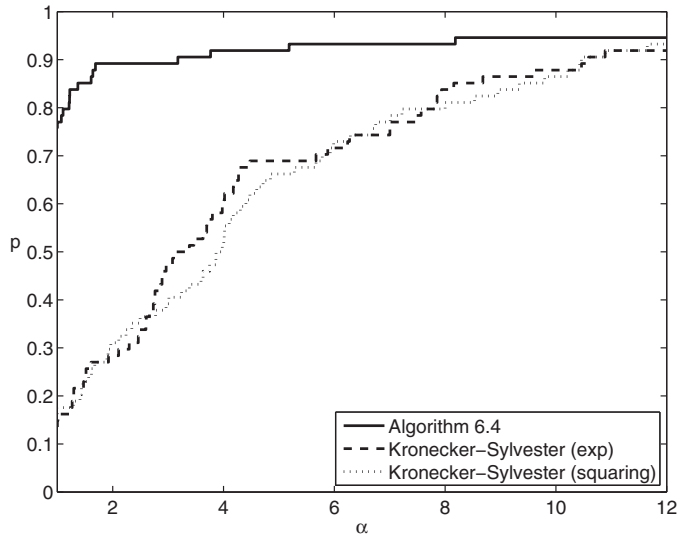

FIG. 6.2. *Same data as in Figure* 6.1 *presented as a performance profile.*

the Frobenius norm relative errors for Algorithm 6.4 and for the Kronecker–Sylvester algorithm in both "squaring" and "exponential" variants. Also shown is a solid line representing a finite difference approximation to $\mathrm{cond}(L_{\exp}, A)u$, where $\mathrm{cond}(L_{\exp}, A)$ is a condition number defined in terms of the Jacobian of the map L regarded as a function of A and E (we use (1.2) with a small, random E); this line indicates the accuracy we would expect from a forward stable algorithm for computing the Fréchet derivative. Figure 6.1 shows that all the methods are performing in a reasonably forward stable manner but does not clearly reveal differences between the methods.

Figure 6.2 plots the same data as a performance profile: for a given α the corresponding point on each curve indicates the fraction p of problems on which the method had error at most a factor α times that of the smallest error over all three methods. The results show clear superiority of Algorithm 6.4 over the Kronecker–Sylvester algorithm in terms of accuracy, for both variants of the latter algorithm. Since Algorithm 6.4 is also by far the more efficient, as explained above, it is clearly the preferred method.

7. Condition number estimation. We now turn our attention to estimating the condition number of the matrix exponential, which from (1.2) is

$$\kappa_{\exp}(A) = \frac{\|L_{\exp}(A)\|\,\|A\|}{\|e^A\|}.$$

Algorithm 6.4 can compute $L_{\exp}(A, E)$ for any direction E, but to obtain the norm $\|L_{\exp}(A)\|$ we need to maximize $L_{\exp}(A, E)$ over all E of unit norm.

For the moment we will consider general f. We can write

$$\mathrm{vec}(L(A, E)) = K(A)\mathrm{vec}(E), \tag{7.1}$$

where $K(A) \in \mathbb{C}^{n^2 \times n^2}$ is independent of E and $\mathrm{vec}(E) \in \mathbb{C}^{n^2}$ denotes the vector comprising the columns of E strung one on top of the other from first to last. We refer to $K(A)$ as the Kronecker form of the Fréchet derivative. From (7.1) we have

$\|L(A,E)\|_F = \|K(A)\mathrm{vec}(E)\|_2$, and on dividing by $\|E\|_F = \|\mathrm{vec}(E)\|_2$ and maximizing over all E it follows that

$$\|L(A)\|_F = \|K(A)\|_2. \tag{7.2}$$

Therefore we can compute $\|L(A)\|_F$ exactly by forming $K(A)$, whose columns are $\mathrm{vec}(L(A, e_i e_j^T))$ for $i,j = 1\colon n$, and then taking the 2-norm. This is a prohibitively expensive computation, typically requiring $O(n^5)$ flops. However, in practice only an estimate of the correct order of magnitude is needed. For this purpose it is appropriate to use matrix norm estimation techniques.

The following algorithm is essentially the usual power method applied to $K(A)$, and exploits the relation (7.2) [6, section 3.4], [9].

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

```
1  Choose a nonzero starting matrix Z_0 ∈ C^{n×n}.
2  for k = 0:∞
3        W_{k+1} = L(A, Z_k)
4        Z_{k+1} = L*(A, W_{k+1})
5        η_{k+1} = ‖Z_{k+1}‖_F / ‖W_{k+1}‖_F
6        if converged, η = η_{k+1}, quit, end
7  end
```

Here, $\star$ denotes the adjoint, and for the exponential, $L^\star_{\exp}(A,W) \equiv L_{\exp}(A^*, W)$.

We do not specify Algorithm 7.1 in any more detail because we prefer a 1-norm variant of the power method. For the 1-norm there is no analogue of (7.2), but the next lemma shows how $\|K(A)\|_1$ compares with $\|L(A)\|_1$.

LEMMA 7.2 ([6, Lemma 3.18]). *For $A \in \mathbb{C}^{n\times n}$ and any function f,*

$$\frac{\|L(A)\|_1}{n} \le \|K(A)\|_1 \le n\|L(A)\|_1. \qquad \square \tag{7.3}$$

The following algorithm, which again needs the ability to evaluate $L(A,E)$ and $L^\star(A,E)$, is essentially [6, Algorithm 3.22]; it employs a block 1-norm estimation algorithm of Higham and Tisseur [7], which for an $n \times n$ matrix carries out a 1-norm power iteration whose iterates are $n \times t$ matrices, where t is a parameter.

ALGORITHM 7.3 (block 1-norm estimator for Fréchet derivative). *Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm uses a block 1-norm estimator to produce an estimate η of $\|L(A)\|_1$. More precisely, $\eta \le \|K(A)\|_1$, where $\|K(A)\|_1$ satisfies* (7.3).

```
1  Apply Algorithm 2.4 from Higham and Tisseur [7] with parameter t = 2 to
   the Kronecker matrix representation B := K(A) of L(A), noting that
   By ≡ vec(L(A,E)) and B*y ≡ vec(L*(A,E)), where vec(E) = y.
```

Key properties of Algorithm 7.3 are that it typically requires about $4t$ Fréchet derivative evaluations and it almost invariably produces an estimate of $\|K(A)\|_1$ correct to within a factor 3. A factor n of uncertainty is added when we take η as an estimate of $\|L(A)\|_1$, but just changing the norm from the 1-norm to the ∞-norm can introduce such a factor, so it is not a serious weakness. Overall, the algorithm is a very reliable means of estimating $\|L(A)\|_1$ to within a factor $3n$.

Returning to the exponential, our interest is in how to combine Algorithms 6.4 and 7.3 in the most efficient manner. We need to evaluate $L(A,E)$ and $L(A^*,E)$ for a fixed A and several different E, without knowing all the E at the start of the

TABLE 7.1

Matrices that must be computed and stored during the initial e^A evaluation, to be reused during the Fréchet derivative evaluations. "LU fact" stands for LU factorization of $-u_m + v_m$, and $B = A/2^s$.

m						
3	$r_3(A)$				LU fact.	$W_3(A)$
5	$r_5(A)$	A^2			LU fact.	$W_5(A)$
7	$r_7(A)$	A^2	A^4		LU fact.	$W_7(A)$
9	$r_9(A)$	A^2	A^4		LU fact.	$W_9(A)$
13	$r_{13}(B)^{2^i}$, $i = 0\colon s-1$	B^2	B^4	B^6	LU fact.	W

computation. To do so we will store matrices accrued during the initial computation of e^A and reuse them in the Fréchet derivative evaluations. This of course assumes the availability of extra storage, but in modern computing environments ample storage is usually available.

In view of the evaluation schemes (6.11)–(6.13) and (6.15), for $m \in \{3, 5, 7, 9\}$ we need to store A^{2k}, $k = 1\colon d_{(m-1)/2}$, where $d = [0\ 1\ 2\ 2]$, along with $W_m(A) = \sum_{k=0}^{(m-1)/2} b_{2k+1}A^{2k}$, $r_m(A)$, and the LU factorization of $(-u_m + v_m)(A)$. For $m = 13$, the matrix A needs to be scaled, to $B = A/2^s$. According to the scheme used in Algorithm 6.4 we need to store B^{2k}, $k = 1\colon 3$, $W \equiv w(B)$, the LU factorization of $(-u_m + v_m)(B)$, and $r_m(B)^{2^i}$, $i = 0\colon s-1$. Table 7.1 summarizes the matrices that need to be stored for each m.

The following algorithm computes the matrix exponential and estimates its condition number. Since the condition number is not needed to high accuracy we use the parameters θ_m in Table 6.1 (designed for e^A) instead of ℓ_m (designed for $L(A, E)$). The bound in (6.10) for the Fréchet derivative backward error $\|\Delta E\|/\|E\|$ does not exceed $28u$ for $m \le 13$ when we use the θ_m, so the loss in backward stability for the Fréchet derivative evaluation is negligible. If the condition estimate is omitted, the algorithm reduces to Algorithm 6.1. The algorithm exploits the relation $L_f(A^*, E) = L_f(A, E^*)^*$, which holds for any f with a power series expansion with real coefficients, by (3.1).

ALGORITHM 7.4 (scaling and squaring algorithm for the matrix exponential with 1-norm condition estimation). *Given $A \in \mathbb{C}^{n\times n}$ this algorithm computes $X = e^A$ by the scaling and squaring method (Algorithm 6.1) and an estimate $\gamma \approx \kappa_{\exp}(A)$ using the block 1-norm estimator (Algorithm 7.1). It uses the values θ_m listed in Table 6.1. The algorithm is intended for IEEE double precision arithmetic.*

```
1  α = ‖A‖_1
2  for m = [3 5 7 9]
3      if α ≤ θ_m
4          Evaluate U = u_m(A) and V = v_m(A), using (6.11).
5          Solve (−U + V)X = U + V for X.
6          Store the needed matrices (see Table 7.1).
7          Use Algorithm 7.3 to produce an estimate η ≈ ‖L_exp(A)‖_1.
           ..... To compute L_exp(A, E) for a given E:
8                  Evaluate M_{2k} = L_{x^{2^k}}(A, E), k = 1 : (m − 1)/2.
9                  L_u ← A (Σ_{k=1}^{(m−1)/2} b_{2k+1} M_{2k}) + E W_m(A)
10                 L_v ← Σ_{k=1}^{(m−1)/2} b_{2k} M_{2k}
11                 Solve (−U + V)L = L_u + L_v + (L_u − L_v)X for L.
           ..... To compute L*_exp(A, E) for a given E:
```

12 Execute lines 8–11 with E replaced by its conjugate transpose and take the conjugate transpose of the result.
13 goto line 44
14 end
15 % Use degree $m = 13$.
16 $s = \lceil \log_2(\alpha/\theta_{13}) \rceil$, the minimal integer such that $2^{-s}\alpha \le \theta_{13}$.
17 $A \leftarrow 2^{-s}A$
18 $A_2 = A^2$, $A_4 = A_2^2$, $A_6 = A_2A_4$
19 $W_1 = b_{13}A_6 + b_{11}A_4 + b_9A_2$
20 $Z_1 = b_{12}A_6 + b_{10}A_4 + b_8A_2$
21 $W = A_6W_1 + b_7A_6 + b_5A_4 + b_3A_2 + b_1I$
22 $U = AW$
23 $V = A_6Z_1 + b_6A_6 + b_4A_4 + b_2A_2 + b_0I$
24 Solve $(-U + V)X_s = U + V$ for X_s
25 for $i = s\colon -1\colon 1$
26 $X_{i-1} = X_i^2$
27 end
28 $X = X_0$
29 Use Algorithm 7.3 to produce an estimate $\eta \approx \|L_{\exp}(\widetilde{A})\|_1$, where $\widetilde{A}$ denotes the original input matrix A.
..... To compute $L_{\exp}(\widetilde{A}, E)$ for a given E:
30 $E \leftarrow 2^{-s}E$
31 $M_2 = AE + EA$, $M_4 = A_2M_2 + M_2A_2$, $M_6 = A_4M_2 + M_4A_2$
32 $L_{w_1} = b_{13}M_6 + b_{11}M_4 + b_9M_2$
33 $L_{w_2} = b_7M_6 + b_5M_4 + b_3M_2$
34 $L_{z_1} = b_{12}M_6 + b_{10}M_4 + b_8M_2$
35 $L_{z_2} = b_6M_6 + b_4M_4 + b_2M_2$
36 $L_w = A_6L_{w_1} + M_6W_1 + L_{w_2}$
37 $L_u = AL_w + EW$
38 $L_v = A_6L_{z_1} + M_6Z_1 + L_{z_2}$
39 Solve $(-U + V)L = L_u + L_v + (L_u - L_v)X_s$ for L.
40 for $i = s\colon -1\colon 1$
41 $L \leftarrow X_iL + LX_i$
42 end
..... To compute $L^{\star}_{\exp}(\widetilde{A}, E)$ for a given E:
43 Execute lines 30–42 with E replaced by its conjugate transpose and take the conjugate transpose of the result.
44 $\gamma = \eta\alpha/\|X\|_1$

The cost of Algorithm 7.4 is the cost of computing e^A plus the cost of about 8 Fréchet derivative evaluations, so obtaining e^A and the condition estimate multiplies the cost of obtaining just e^A by a factor of about 17. This factor can be reduced to 9 if the parameter t in the block 1-norm power method is reduced to 1, at a cost of slightly reduced reliability.

In our MATLAB implementation of Algorithm 7.4 we invoke the function `funm_condest1` from the Matrix Function Toolbox [4], which interfaces to the MATLAB function `normest1` that implements the block 1-norm estimation algorithm of [7].

With the same matrices as in the test of the previous section we used Algorithm 7.4 to estimate $\|K(A)\|_1$ and also computed $\|K(A)\|_1$ exactly by forming $K(A)$ as described at the start of this section. Figure 7.1 plots the norms and the estimates.

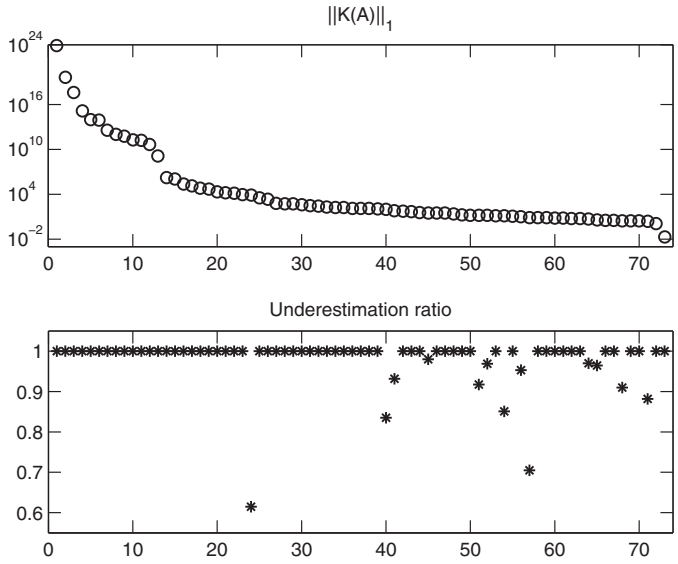

FIG. 7.1. $\|K(A)\|_1$ *and underestimation ratio* $\eta/\|K(A)\|_1$*, where* η *is the estimate of* $\|K(A)\|_1$ *produced by Algorithm* 7.4.

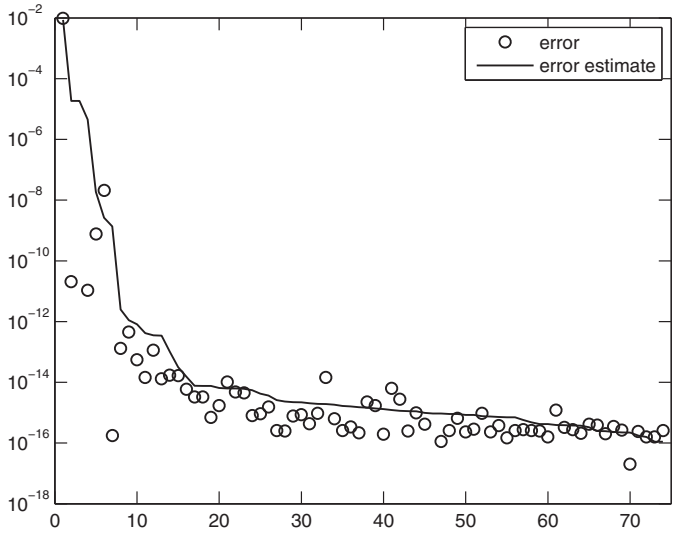

FIG. 7.2. *Normwise relative error for computed exponential and error estimate comprising condition number estimate times unit roundoff.*

The worst underestimation ratio is 0.61, so the estimates are all within a factor 2 of the true 1-norm.

Finally, we invoked Algorithm 7.4 on the same set of test matrices and computed the "exact" exponential in 100 digit precision. Figure 7.2 plots the error in the computed exponential along with the quantity γu: the condition estimate multiplied by the unit roundoff, regarded as an error estimate. If the scaling and squaring algorithm were forward stable and the condition estimate reliable we would expect

the error to be bounded by $\phi(n)\gamma u$ for some low degree polynomial ϕ. The overall numerical stability of the scaling and squaring algorithm is not understood [6], but our experience is that the method usually does behave in a forward stable way. Figure 7.2 indicates that the condition estimate from Algorithm 7.4 provides a useful guide to the accuracy of the computed exponential from the algorithm.

8. Concluding remarks. The LAPACK Users' Guide states [1, p. 77] that "Our goal is to provide error bounds for most quantities computed by LAPACK." This is a desirable goal for any numerical algorithm, and in order to achieve it error analysis must be developed that yields a reasonably sharp error bound that can be efficiently estimated. For matrix function algorithms a complete error analysis is not always available, and for the forward error a bound of the form $\mathrm{cond}(f, A)u$ is the best we can expect in general. To date relatively little attention has been paid to combining evaluation of $f(A)$ with computation of the Fréchet derivative $L(A, E)$ and estimation of the condition number $\mathrm{cond}(f, A)$. We are currently applying and extending the ideas developed here to other transcendental functions such as the logarithm and the sine and cosine and will report on this work in a future paper.

Acknowledgment. We thank Bruno Iannazzo for his helpful comments on a draft of this paper.

REFERENCES

[1] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, and D. C. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, PA, 1999.

[2] G. A. Baker, Jr., and P. Graves-Morris, *Padé Approximants*, Encyclopedia of Mathematics and Its Applications, 2nd ed., Vol. 59, Cambridge University Press, Cambridge, 1996.

[3] N. J. Higham, *The Matrix Computation Toolbox*, http://www.ma.man.ac.uk/~higham/mctoolbox.

[4] N. J. Higham, *The Matrix Function Toolbox*, http://www.ma.man.ac.uk/~higham/mftoolbox.

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

[6] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, PA, 2008.

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

[8] G. Jewitt and J. R. McCrorie, *Computing estimates of continuous time macroeconometric models on the basis of discrete data*, Comput. Statist. Data Anal., 49 (2005), pp. 397–416.

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

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

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

[12] R. Mathias, *Evaluating the Frechet derivative of the matrix exponential*, Numer. Math., 63 (1992), pp. 213–226.

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

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

[15] I. Najfeld and T. F. Havel, *Derivatives of the matrix exponential and their computation*, Adv. Appl. Math., 16 (1995), pp. 321–375.

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

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