Numerical Algorithms (2005) **40:** 383–400
DOI 10.1007/s11075-005-8141-0

Efficient algorithms for the matrix cosine and sine [*]

Gareth I. Hargreaves [**] and Nicholas J. Higham [***]

School of Mathematics, University of Manchester, Sackville Street, Manchester, M60 1QD, UK
E-mail: hargreaves@ma.man.ac.uk, http://www.ma.man.ac.uk/~hargreaves/
higham@ma.man.ac.uk, http://www.ma.man.ac.uk/~higham/

Received 4 February 2005; accepted 16 April 2005
Communicated by C. Brezinski

Several improvements are made to an algorithm of Higham and Smith for computing the matrix cosine. The original algorithm scales the matrix by a power of 2 to bring the ∞-norm to 1 or less, evaluates the $[8/8]$ Padé approximant, then uses the double-angle formula $\cos(2A) = 2\cos^2 A - I$ to recover the cosine of the original matrix. The first improvement is to phrase truncation error bounds in terms of $\|A^2\|^{1/2}$ instead of the (no smaller and potentially much larger quantity) $\|A\|$. The second is to choose the degree of the Padé approximant to minimize the computational cost subject to achieving a desired truncation error. A third improvement is to use an absolute, rather than relative, error criterion in the choice of Padé approximant; this allows the use of higher degree approximants without worsening an a priori error bound. Our theory and experiments show that each of these modifications brings a reduction in computational cost. Moreover, because the modifications tend to reduce the number of double-angle steps they usually result in a more accurate computed cosine in floating point arithmetic. We also derive an algorithm for computing both $\cos(A)$ and $\sin(A)$, by adapting the ideas developed for the cosine and intertwining the cosine and sine double angle recurrences.

Keywords: matrix function, matrix cosine, matrix sine, matrix exponential, Taylor series, Padé approximation, Padé approximant, double-angle formula, rounding error analysis, Schur–Parlett method, MATLAB

AMS subject classification: 65F30

1. Introduction

The matrix exponential, undoubtedly the most-studied matrix function, provides the solution $y(t) = \mathrm{e}^{A} y_0$ to the first order differential system $\mathrm{d}y/\mathrm{d}t = Ay$, $y(0) = y_0$, where $A \in \mathbb{C}^{n\times n}$ and $y \in \mathbb{C}^n$. Trigonometric matrix functions play a similar role in

[*] Numerical Analysis Report 461, Manchester Centre for Computational Mathematics, February 2005.

[**] This work was supported by an Engineering and Physical Sciences Research Council Ph.D. Studentship.

[***] This work was supported by Engineering and Physical Sciences Research Council grant GR/T08739 and by a Royal Society–Wolfson Research Merit Award.

second order differential systems. For example, the problem

$$\frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + Ay = 0, \quad y(0) = y_0,\ y'(0) = y'_0 \tag{1.1}$$

has solution[1]

$$y(t) = \cos\bigl(\sqrt{A}\,t\bigr)y_0 + \bigl(\sqrt{A}\,\bigr)^{-1}\sin\bigl(\sqrt{A}\,t\bigr)y'_0, \tag{1.2}$$

where $\sqrt{A}$ denotes any square root of A. More general problems of this type, with a forcing term $f(t)$ on the right-hand side, arise from semidiscretization of the wave equation and from mechanical systems without damping, and their solutions can be expressed in terms of integrals involving the sine and cosine [10]. Despite the important role played by the matrix sine and cosine in these second order differential systems, their numerical computation has received relatively little attention. As well as methods for computing them individually, methods are needed for simultaneously computing the sine and cosine of the same matrix, as naturally arises in (1.2).

A general algorithm for computing the matrix cosine that employs rational approximations and the double-angle formula $\cos(2A) = 2\cos^2(A) - I$ was proposed by Serbin and Blalock [11]. Higham and Smith [6] developed a particular version of this algorithm based on Padé approximation and supported by truncation and rounding error analysis. In this work we revisit the algorithm of Higham and Smith, making several improvements to increase both its efficiency and its accuracy and adapting it to compute $\cos(A)$ and $\sin(A)$ together.

First, we state the original algorithm [6, algorithm 6.1]. This algorithm, and all those discussed here, are intended for use in IEEE double precision arithmetic, for which the unit roundoff $u = 2^{-53} \approx 1.11 \times 10^{-16}$.

Algorithm 1.1. Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm approximates $X = \cos(A)$.

```
1. Find the smallest nonnegative integer $m$ so that $2^{-m}\|A\|_\infty \leqslant 1$.
2. $C_0 = r_{88}(2^{-m}A)$, where $r_{88}(x)$ is the [8/8] Padé approximant to $\cos(x)$.
3. for $i = 0 : m - 1$
4.     $C_{i+1} = 2C_i^2 - I$
5. end
6. $X = C_m$
```

Cost. $(4 + \mathrm{ceil}(\log_2 \|A\|_\infty))M + D$, where M denotes a matrix multiplication and D the solution of a linear system with n right-hand side vectors.

The algorithm can be explained as follows. Line 1 determines the scaling needed to reduce the ∞-norm of A to 1 or less. Line 2 computes the [8/8] Padé approximant of

[1] This formula is interpreted for singular A by expanding $(\sqrt{A}\,)^{-1}\sin(\sqrt{A}\,t)$ as a power series in A.

the scaled matrix; it is evaluated by the technique described in section 2 (cf. (2.4)) at a cost of $4M + D$. The loop beginning at line 3 uses the double-angle formula $\cos(2A) = 2\cos(A)^2 - I$ to undo the effect of the scaling.

Higham and Smith [6] show that

$$\frac{\|\cos(A) - r_{88}(A)\|_\infty}{\|\cos(A)\|_\infty} \leqslant 3.26 \times 10^{-16} \approx 3u \quad \text{for } \|A\|_\infty \leqslant 1. \tag{1.3}$$

Hence in algorithm 1.1, $r_{88}(2^{-m}A)$ approximates $C_0 = \cos(2^{-m}A)$ to essentially full machine accuracy.

Algorithm 1.1 can optionally make use of preprocessing, which is implemented in the next algorithm.

Algorithm 1.2. Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm computes $X = \cos(A)$ by preprocessing A and then invoking a given algorithm for computing $\cos(A)$.

```
1. $A \leftarrow A - \pi q I$, where $q$ is whichever of 0, floor($\mu$) and ceil($\mu$) yields the smaller
   value of $\|A - \pi q I\|_\infty$, where $\mu = \text{trace}(A)/(n\pi)$.
2. $B = D^{-1}AD$, where $D$ balances $A$.
3. if $\|B\|_\infty < \|A\|_\infty$, $A = B$, end
4. Apply the given algorithm to compute $C = \cos(A)$.
5. $X = (-1)^q C$
6. if balancing was performed, $X = DXD^{-1}$, end
```

Lines 1–3 carry out preprocessing prior to the main computations; they apply a similarity transformation and a shift in an attempt to reduce the norm. Lines 5 and 6 undo the effect of the preprocessing. See [6] for an explanation of the preprocessing.

The impetus for this work comes from two observations. First, the analysis of Higham and Smith focuses on the [8/8] Padé approximant, but the use of an approximant of a different, A-dependent degree could potentially yield a more efficient algorithm. The recent work of Higham [5] on the scaling and squaring method for the matrix exponential shows how to choose the degree of the Padé approximant and the norm of the scaled matrix at which the approximant is evaluated in order to obtain an optimally efficient algorithm, and the same approach is applicable to the double-angle algorithm for the cosine. The second relevant observation is that the double-angle steps in algorithm 1.1 can potentially magnify both truncation and rounding errors substantially, so reducing the number of such steps (while not sacrificing the efficiency of the whole algorithm) could bring an important improvement in accuracy. Indeed it is shown in [6] that the computed $\widehat{C}_i =: C_i + E_i$ satisfies

$$\begin{aligned}\|E_i\|_\infty \leqslant{}& (4.1)^i \|E_0\|_\infty \|C_0\|_\infty \|C_1\|_\infty \dots \|C_{i-1}\|_\infty \\ &+ \gamma_{n+1} \sum_{j=0}^{i-1} 4.1^{i-j-1} \big(2.21\|C_j\|_\infty^2 + 1\big) \|C_{j+1}\|_\infty \dots \|C_{i-1}\|_\infty,\end{aligned} \tag{1.4}$$

where $\gamma_k = ku/(1-ku)$, which warns of error growth exponential in the number of double-angle steps, but algorithm 1.1 does not attempt to minimize the number of such steps.

In this work we show how to choose the degree of the Padé approximant to minimize the computational effort while at the same time (approximately) minimizing the number of double-angle steps, and where minimization is subject to retaining numerical stability in evaluation of the Padé approximant. We also show how to exploit the fact that the cosine is an even function to reduce the work, possibly by a large amount.

In section 2 we develop an improved version of algorithm 1.1 that incorporates these ideas. In section 3 we argue that imposing an absolute, rather than relative, error criterion on the Padé approximant leads to a more efficient algorithm whose accuracy is in general no worse. The numerical experiments of section 4 compare algorithm 1.1 with the two new algorithms derived in this paper and also with MATLAB's `funm` applied to the cosine. These sections concentrate on the cosine. There is no natural analogue of algorithm 1.1 for the sine, because the corresponding double-angle recurrence $\sin(2A) = 2\sin(A)\cos(A)$ would require cosines. However, computing the sine reduces to computing the cosine through $\sin(A) = \cos(A - \frac{\pi}{2}I)$.

Building on the new algorithms for the cosine, in section 5 we develop an algorithm for simultaneously computing $\cos(A)$ and $\sin(A)$ at lower cost than if they were computed independently, which is useful when evaluating (1.2), for example. Concluding remarks are given in section 6.

Throughout this paper an unsubscripted norm denotes an arbitrary subordinate matrix norm.

2. An algorithm with variable degree Padé approximants

We denote by $r_m(x) = p_m(x)/q_m(x)$ an $[m/m]$ Padé approximant of a given function $f(x)$. By definition, p_m and q_m are polynomials in x of degree at most m and

$$f(x) - r_m(x) = \mathrm{O}\big(x^{2m+1}\big).$$

We will normalize so that p_m and q_m have no common zeros and $q_m(0) = 1$. For later reference we write

$$p_m(x) = \sum_{i=0}^{m} a_i x^i, \qquad q_m(x) = \sum_{i=0}^{m} b_i x^i. \tag{2.1}$$

As discussed in [6], it is not known whether Padé approximants of $\cos(x)$ exist for all m, though formulae of Magnus and Wynn [8] are available that give the coefficients of p_m and q_m in terms of ratios of determinants of matrices whose entries involve binomial coefficients. Since cos is an even function we need consider only even degrees $2m$. Both p_{2m} and q_{2m} are even polynomials and

$$\cos(x) - r_{2m}(x) = \mathrm{O}\big(x^{4m+2}\big).$$

Our first task is to bound the truncation error, which has the form

$$\cos(A) - r_{2m}(A) = \sum_{i=2m+1}^{\infty} c_{2i} A^{2i}.$$

Hence

$$\left\|\cos(A) - r_{2m}(A)\right\| \leqslant \sum_{i=2m+1}^{\infty} |c_{2i}| \theta^{2i}, \tag{2.2}$$

where

$$\theta = \theta(A) = \left\|A^2\right\|^{1/2}.$$

Note that we have expressed the bound in terms of $\|A^2\|^{1/2}$ instead of the (no smaller) quantity $\|A\|$. The reason is that $\|A^2\|^{1/2} \ll \|A\|$ is possible for nonnormal A. Since our algorithm will require the matrix A^2 to be computed, it makes sense to use knowledge of its norm in the derivation; this was not done in [6] and so is one way in which we gain an improvement over algorithm 1.1.

It is easy to see that

$$\left\|\cos(A)\right\| \geqslant 1 - \frac{\|A^2\|}{2!} - \frac{\|A^2\|^2}{4!} - \cdots = 1 - \left(\cosh\left(\left\|A^2\right\|^{1/2}\right) - 1\right) = 2 - \cosh(\theta).$$

Combining this bound with (2.2), we conclude that

$$\frac{\|\cos(A) - r_{2m}(A)\|}{\|\cos(A)\|} \leqslant \frac{\sum_{i=2m+1}^{\infty} |c_{2i}| \theta^{2i}}{2 - \cosh(\theta)} \quad \text{for } \theta < \cosh^{-1}(2) \approx 1.317. \tag{2.3}$$

To design the algorithm we need to know for each m how small $\theta = \|A^2\|^{1/2}$ must be in order for r_{2m} to deliver the required accuracy. Adopting the approach used by Higham [5] for the exponential, we therefore determine the largest value of θ, denoted by θ_{2m}, such that the relative error bound in (2.3) does not exceed u. To do so we compute the c_{2i} symbolically and evaluate the bound (2.3) in 250 decimal digit arithmetic, summing the first 150 terms of the series, all with MATLAB's Symbolic Math Toolbox. We use a zero-finder to find θ_{2m}, obtaining the values shown in table 2.1. We see that θ_{2m} rapidly approaches $\cosh^{-1}(2)$ as m increases: θ_{14} and θ_{12} differ by about 10^{-9}.

When we rerun the computation with $2m = 8$, aiming for a relative error bound 3.26×10^{-16}, we find that $\theta_8 = 1.005$, which shows that the bound (1.3) is close to optimal (modulo its use of $\|A\|$ in place of $\|A^2\|^{1/2}$).

Now we need to determine the cost of evaluating r_{2m}. Given the absence of any convenient continued fraction or partial fraction forms, we will explicitly evaluate p_{2m} and q_{2m} and then solve the multiple right-hand side system $q_{2m}r_{2m} = p_{2m}$. The most efficient evaluation scheme we have found is to treat p_{2m} and q_{2m} as degree m polynomials in A^2 and apply the Paterson–Stockmeyer method [9], adapted for the simultaneous eval-

Table 2.1
Maximum value θ_{2m} of $\theta = \|A^2\|^{1/2}$ such that the relative error bound (2.3) does not exceed $u = 2^{-53}$.

$2m$	2	4	6	8	10	12	14
θ_{2m}	6.1e−3	1.1e−1	4.3e−1	9.5e−1	1.315	1.317	1.317

Table 2.2
Number of matrix multiplications π_{2m} required to evaluate $p_{2m}(A)$ and $q_{2m}(A)$.

$2m$	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
π_{2m}	1	2	3	4	5	5	6	6	7	7	8	8	9	9	9

uation of two polynomials of the same argument and degree as suggested by Higham [4]. Of equal cost for $8 \leqslant 2m \leqslant 28$ are the schemes of the form illustrated for $2m = 12$ by

$$\begin{aligned} A_2 &= A^2, \quad A_4 = A_2^2, \quad A_6 = A_2A_4, \\ p_{12} &= a_0I + a_2A_2 + a_4A_4 + a_6A_6 + A_6(a_8A_2 + a_{10}A_4 + a_{12}A_6), \\ q_{12} &= b_0I + b_2A_2 + b_4A_4 + b_6A_6 + A_6(b_8A_2 + b_{10}A_4 + b_{12}A_6). \end{aligned} \tag{2.4}$$

Table 2.2 summarizes the cost of evaluating p_{2m} and q_{2m} for $2m = 2:2:30$.

In view of table 2.1 we can restrict to $2m \leqslant 12$, since θ_{14} is only slightly larger than θ_{12}. Since table 2.2 shows that r_{12} can be evaluated at the same cost as the less accurate r_{10}, we can remove $2m = 10$ from consideration. Hence we need consider only $2m = 2, 4, 6, 8, 12$.

If $\theta = \|A^2\|^{1/2} \leqslant \theta_{2m}$, for $2m = 2, 4, 6, 8$ or 12, then we should take $r_{2m}(A)$ with the smallest such m as our approximation to $\cos(A)$. Otherwise, we will need to scale: we simply divide A by 2^s, with s chosen minimally so that $\|(2^{-s}A)^2\|_\infty^{1/2} \leqslant \theta_{2m}$ for some m, with $2m = 8$ and $2m = 12$ being the only possibilities (since $\theta_6 < \theta_{12}/2$, $2m = 6$ offers no computational saving over $2m = 12$). This strategy minimizes the number of double-angle steps, with their potential error magnification, while at the same time minimizing the total work.

We now need to consider the effects of rounding errors on the evaluation of r_{2m}. Consider, first, the evaluation of p_{2m} and q_{2m}, and assume initially that A^2 is evaluated exactly. Let $g_{2m}(A^2)$ denote either of the even polynomials $p_{2m}(A)$ and $q_{2m}(A)$. It follows from a general result in [5, theorem 2.2] that

$$\left\|g_{2m}(A^2) - fl\big(g_{2m}(A^2)\big)\right\| \leqslant \tilde{\gamma}_{mn}\tilde{g}_{2m}\big(\|A^2\|\big), \tag{2.5}$$

where $\tilde{g}_{2m}$ denotes g_{2m} with its coefficients replaced by their absolute values. We have determined numerically that $\|\tilde{g}_{2m}(A^2)\| \leqslant 2$ for $\theta(A) \leqslant \theta_{2m}$ and $2m \leqslant 16$, so the bound (2.5) is suitably small. However, when we take into account the error in forming A^2 we find that the bound (2.5) is multiplied by a term that is approximately $\mu(A) = \||A|^2\|/\|A^2\| \geqslant 1$. The quantity μ can be arbitrarily large. However, μ is large precisely

Table 2.3
Upper bound for $\kappa(q_{2m}(A))$ when $\theta \leqslant \theta_{2m}$, based on (2.6) and (2.7), where the θ_{2m} are given in table 2.1.

$2m$	2	4	6	8	12
Bound	1.0	1.0	1.0	1.0	1.1

when basing the algorithm on $\theta(A)$ rather than $\|A\|$ produces a smaller s, so potentially increased rounding errors in the evaluation of p_{2m} and q_{2m} are balanced by potentially decreased error propagation in the double angle phase.

Since we obtain r_{2m} by solving a linear system with coefficient matrix $q_{2m}(A)$, we require $q_{2m}(A)$ to be well conditioned to be sure that the system is solved accurately. From (2.1), we have

$$\left\|q_{2m}(A)\right\| \leqslant \sum_{k=0}^{m} |b_{2k}|\theta^{2k}. \tag{2.6}$$

Using the inequality $\|(I+E)^{-1}\| \leqslant (1-\|E\|)^{-1}$ for $\|E\| < 1$ gives

$$\left\|q_{2m}(A)^{-1}\right\| \leqslant \frac{1}{|b_0| - \|\sum_{k=1}^{m} b_{2k}A^{2k}\|} \leqslant \frac{1}{|b_0| - \sum_{k=1}^{m} |b_{2k}|\theta^{2k}}. \tag{2.7}$$

Table 2.3 tabulates the bound for $\kappa(q_{2m}(A)) = \|q_{2m}(A)\|\,\|q_{2m}(A)^{-1}\|$ obtained by combining (2.6) and (2.7). It shows that q_{2m} is well conditioned for all the m of interest.

The algorithm that we have derived is as follows.

Algorithm 2.1. Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm approximates $C = \cos(A)$. It uses the constants θ_{2m} given in table 2.1. The matrix A can optionally be preprocessed using algorithm 1.2.

```
1. $B = A^2$
2. $\theta = \|B\|_\infty^{1/2}$
3. for $d = [2\ 4\ 6\ 8\ 12]$
4.    if $\theta \leqslant \theta_d$
5.       $C = r_d(A)$   % Compute Padé approximant, making use of $B$.
6.       quit
7.    end
8. end
9. $s = \mathrm{ceil}(\log_2(\theta/\theta_{12}))$   % Find minimal integer $s$ such that $2^{-s}\theta \leqslant \theta_{12}$.
10. $B \leftarrow 4^{-s}B$
11. if $\|B\|_\infty^{1/2} \leqslant \theta_8$, $d = 8$, else $d = 12$, end
12. $C = r_d(2^{-s}A)$   % Compute Padé approximant, making use of $B = (2^{-s}A)^2$.
```

```
for $i = 1 : s$
    $C \leftarrow 2C^2 - I$
end
```

Cost. $(\pi_d + \text{ceil}(\log_2(\|A\|_\infty/\theta_d)))M + D$, where d is the degree of Padé approximant used and θ_d and π_d are tabulated in tables 2.1 and 2.2, respectively.

To summarize, algorithm 2.1 differs from algorithm 1.1 in two main ways:

1. It supports variable degree Padé approximation, with the degree chosen to minimize both the work and the number of double-angle steps.
2. It bases its decisions on $\|A^2\|_\infty^{1/2}$ rather than $\|A\|_\infty$ and so uses truncation error estimates that are potentially much sharper, though the bounds for the effect of rounding errors on the evaluation of the Padé approximant can be larger.

Note that algorithm 2.1 requires as input only A^2, not A. Consequently, if it is used to compute the cosine term in (1.2) there is no need for a square root to be computed.

3. Absolute error-based algorithm

Algorithm 2.1 uses a Padé approximant of maximal degree 12. The limitation on degree comes from requiring the relative error bound (2.3) to be no larger than u: the need to ensure $\cos(A) \neq 0$ enforces the restriction $\theta < \cosh^{-1}(2)$, which makes higher degree Padé approximants uneconomical. If this restriction could be removed then larger degrees, which would allow fewer double-angle steps and hence potentially more accurate results, would be competitive in cost. Imposing an absolute error bound on the Padé approximant achieves this goal, and it can be justified with the aid of the error bound (1.4) for the computed $\widehat{C}_i =: C_i + E_i$.

In both algorithms 1.1 and 2.1, C_0 is a Padé approximant evaluated at $2^{-m}A$, and from $\|C_0\|_\infty = \|\cos(2^{-m}A)\|_\infty \leqslant \cosh(\theta(2^{-m}A))$ we have $\|C_0\|_\infty \leqslant \cosh(1) \approx 1.54$ in algorithm 1.1 and $\|C_0\|_\infty \leqslant \cosh(\theta_{12}) \approx 2$ in algorithm 2.1. Hence, ignoring the rounding errors in evaluating the Padé approximant, we have $\|E_0\|_\infty \lesssim u\|C_0\|_\infty \lesssim u$, and (1.4) can be written

$$\begin{aligned}\|E_m\|_\infty &\lesssim (4.1)^m u\|C_0\|_\infty\|C_1\|_\infty \dots \|C_{m-1}\|_\infty \\ &\quad + \gamma_{n+1}\sum_{j=0}^{m-1} 4.1^{m-j-1}\big(2.21\|C_j\|_\infty^2 + 1\big)\|C_{j+1}\|_\infty \dots \|C_{m-1}\|_\infty. \qquad (3.1)\end{aligned}$$

But this is exactly the form that (1.4) takes with an absolute error bound $\|E_0\| \leqslant u$. Therefore comparing the effect of absolute and relative bounds on C_0 reduces to comparing the norms of the matrices $C_0, \dots, C_{m-1}$ in the two cases. This is difficult in general, because these matrices depend on the choice of scaling and on m. Since the aim of using an absolute criterion is to allow the norm of the scaled matrix to be larger, we

Table 3.1
Maximum value θ_{2m} of θ such that the absolute error bound (3.2) does not exceed $u = 2^{-53}$.

$2m$	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
θ_{2m}	6.1e−3	0.11	0.43	0.98	1.7	2.6	3.6	4.7	5.9	7.1	8.3	9.6	10.9	12.2	13.6

Table 3.2
Upper bound for $\kappa(q_{2m}(A))$ when $\theta \leqslant \theta_{2m}$, based on (2.6) and (2.7), where the θ_{2m} are given in table 3.1. Bound does not exist for $2m \geqslant 26$.

$2m$	2	4	6	8	10	12	14	16	18	20	22	24
Bound	1.0	1.0	1.0	1.0	1.1	1.2	1.4	1.8	2.4	3.5	7.0	9.0e1

Table 3.3
Upper bounds for $\|\tilde{p}_{2m}\|_\infty$ and $\|\tilde{q}_{2m}\|_\infty$ for $\theta \leqslant \theta_{2m}$.

$2m$	2	4	6	8	10	12	14	16	18	20	22	24
$\|\tilde{p}_{2m}\|_\infty$	1.0	1.0	1.1	1.5	2.7	6.2	1.6e1	4.3e1	1.2e2	3.7e2	1.2e3	3.7e3
$\|\tilde{q}_{2m}\|_\infty$	1.0	1.0	1.0	1.0	1.1	1.1	1.2	1.3	1.4	1.6	1.7	2.0

can expect an upper bound for $\|C_0\|$ to be larger in the absolute case. But if the absolute criterion permits fewer double-angle steps (a smaller m) then, as is clear from (3.1), significant gains in accuracy could accrue. In summary, the error analysis provides support for the use of an absolute error criterion if $\|C_0\|$ is not too large. We now develop an algorithm based on an absolute error bound.

Define θ_{2m} to be the largest value of θ such that the absolute error bound in

$$\big\|\cos(A) - r_{2m}(A)\big\| \leqslant \sum_{i=2m+1}^{\infty} |c_{2i}|\theta^{2i} \tag{3.2}$$

(a restatement of (2.2)) does not exceed u. Using the same method of determining the θ_{2m} as in the previous section we find the values listed in table 3.1. The corresponding bounds for the condition number of q_{2m}, which are finite only for $2m \leqslant 24$, are given in table 3.2.

Now we consider the choice of m. In view of table 3.2, we will restrict to $2m \leqslant 24$. Table 3.3, which concerns error bounds for the evaluation of p_{2m} and q_{2m}, as discussed in the previous section, suggests further restricting $2m \leqslant 20$, say. From table 2.2 it is then clear that we need consider only $2m = 2, 4, 6, 8, 12, 16, 20$. Recall that dividing A (and hence θ) by 2 results in one extra matrix multiplication in the double-angle phase, whereas for $\theta \leqslant \theta_{2m}$ the cost of evaluating the Padé approximant increases by one matrix multiplication with each increase in m in our list of considered values. Since the numbers $\theta_{12}, \theta_{16}, \theta_{20}$ differ successively by less than a factor 2, the value of m that gives the minimal work depends on θ. For example, if $\theta = 7$ then $d = 20$ is best, because nothing would be gained by a further scaling by $1/2$, but if $\theta = 5$ then scaling by $1/2$ enables us to use $d = 12$, and the whole computation then requires one less matrix multiplication than if we immediately applied $d = 20$. Table 3.4 summarizes the

Table 3.4
Logic for choice of scaling and Padé approximant degree. Assuming A has already been scaled, if necessary, so that $\theta \leqslant \theta_{20} = 7.1$, further scaling should be done to bring θ within the range for the indicated value of d.

Range of θ	d
$[0, \theta_{16}] = [0, 4.7]$	smallest $d \in \{2, 4, 6, 8, 12, 16\}$ such that $\theta \leqslant \theta_d$
$(\theta_{16}, 2\theta_{12}] = (4.7, 5.2]$	12 (scale by 1/2)
$(2\theta_{12}, \theta_{20}] = (5.2, 7.1]$	20 (no scaling)

relevant logic. The tactic, then, is to scale so that $\theta \leqslant \theta_{20}$ and to scale further only if a reduction in work is achieved.

We find, computationally, that with this scaling strategy, $\|C_0\|_\infty \leqslant 583$. Since this bound is not too much larger than 1, the argument at the beginning of this section provides justification for the following algorithm.

Algorithm 3.1. Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm approximates $C = \cos(A)$. It uses the constants θ_{2m} given in table 3.1. The matrix A can optionally be preprocessed using algorithm 1.2.

```
$B = A^2$
$\theta = \|B\|_\infty^{1/2}$
for $d = [2\ 4\ 6\ 8\ 12\ 16]$
    if $\theta \leqslant \theta_d$
        $C = r_d(A)$    % Compute Padé approximant, making use of $B$.
        quit
    end
end
$s = \mathrm{ceil}(\log_2(\theta/\theta_{20}))$    % Find minimal integer $s$ such that $2^{-s}\theta \leqslant \theta_{20}$.
Determine optimal $d$ from table 3.4 (with $\theta \leftarrow 2^{-s}\theta$) and increase $s$ as necessary.
$B \leftarrow 4^{-s}B$
$C = r_d(2^{-s}A)$    % Compute Padé approximant, making use of $B = (2^{-s}A)^2$.
for $i = 1 : s$
    $C \leftarrow 2C^2 - I$
end
```

Cost. $(\pi_d + \mathrm{ceil}(\log_2(\|A\|_\infty/\theta_d)))M + D$, where d is the degree of Padé approximant used and θ_d and π_d are tabulated in tables 3.1 and 2.2, respectively.

Algorithm 3.1 allows the norm $\|(2^{-s}A)^2\|_\infty^{1/2}$ for the scaled matrix $2^{-s}A$ to be as large as 7.1, compared with just 1.3 for algorithm 2.1.

4. Numerical experiments

Testing of algorithms 1.1, 2.1, and 3.1 was performed in MATLAB 7 in IEEE double precision arithmetic. We used a set of 54 test matrices that includes 50 10 × 10 matrices obtained from the function `matrix` in the Matrix Computation Toolbox [3] (which includes test matrices from MATLAB itself), together with the four test matrices from [6]. The norms of these matrices range from order 1 to 10^7, though more than half have ∞-norm 10 or less. For comparison, we also applied MATLAB's `funm` function (invoked as `funm(A,@cos)`), which implements the Schur–Parlett method [1]. This method uses Taylor series evaluations of any diagonal Schur blocks of size greater than 1. It requires roughly between $28n^3$ flops and $n^4/3$ flops, so is significantly more expensive than algorithms 1.1, 2.1, and 3.1 except, possibly, when $\|A^2\|_\infty^{1/2}$ is large: say of order 10^3.

We evaluated the relative error

$$\frac{\|\widehat{C} - C\|_\infty}{\|C\|_\infty},$$

where $\widehat{C}$ is the computed approximation to C, and the exact $C = \cos(A)$ is computed in 50 significant decimal digit arithmetic using MATLAB's Symbolic Math Toolbox. The algorithms were applied both with and without preprocessing. The results for no preprocessing are shown in figure 4.1; those for preprocessing are very similar so are

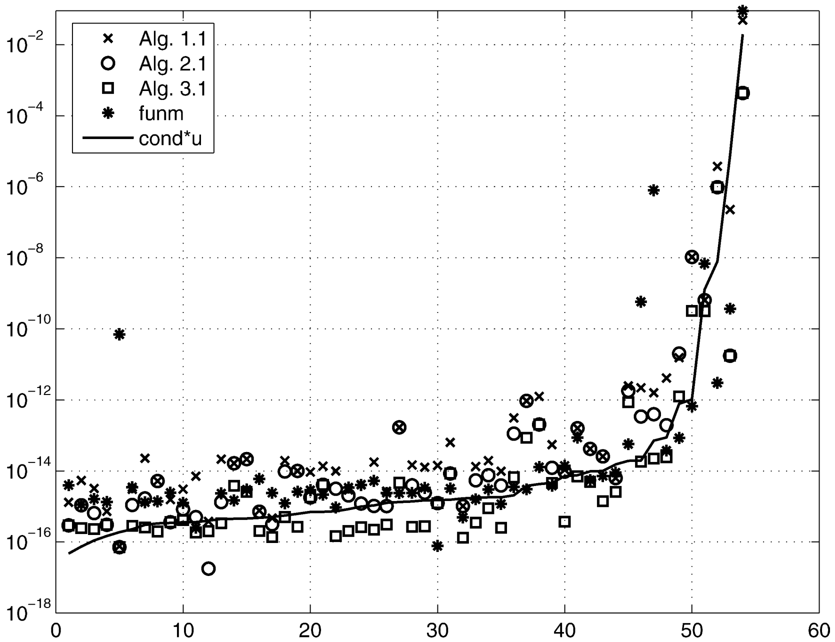

Figure 4.1. Errors for algorithms 1.1, 2.1 and 3.1 without preprocessing.

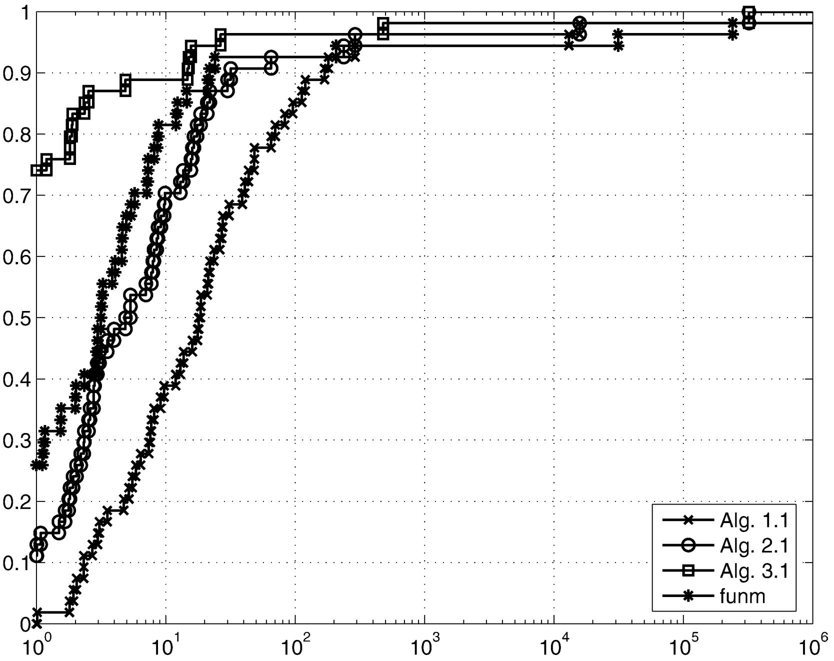

Figure 4.2. Performance profile for the four methods, without preprocessing, on the test set.

omitted. The solid line is the unit roundoff multiplied by an estimate of the relative condition number

$$\operatorname{cond}(A) = \lim_{\epsilon \to 0} \max_{\|E\|_2 \leqslant \epsilon \|A\|_2} \frac{\|\cos(A+E) - \cos(A)\|_2}{\epsilon \|\cos(A)\|_2},$$

which we estimate using the finite-difference power method of Kenney and Laub [2,7]. A method that is forward stable should produce errors not lying far above this line on the graph. Figure 4.2 shows performance profile curves for the four solvers. For a given α on the x-axis, the y coordinate of the corresponding point on the curve is the probability that the method in question has an error within a factor α of the smallest error over all the methods on the given test set.

The results show a clear ordering of the methods for this set of test problems, with algorithm 3.1 in first place, followed by funm, algorithm 2.1, and finally algorithm 1.1.

The mean of the total number of matrix multiplications and multiple right-hand side linear system solves over the test set is 10, 9.1 and 8.6 for algorithms 1.1, 2.1 and 3.1, respectively, without preprocessing, and 9.8, 8.9 and 8.4 with preprocessing. For the involutory matrix gallery('invol',8)*8*pi from [6], algorithm 1.1 requires 29 multiplies and solves, versus only 10 for algorithms 2.1 and 3.1.

MATLAB's funm is generally competitive in accuracy with algorithm 3.1. The worst case for funm – the matrix giving error about 10^{-10} on the left of figure 4.1 – is the Forsythe matrix, which is a 10^{-8} perturbation of a Jordan block. The computed eigenvalues lie approximately on a circle, and this is known to be

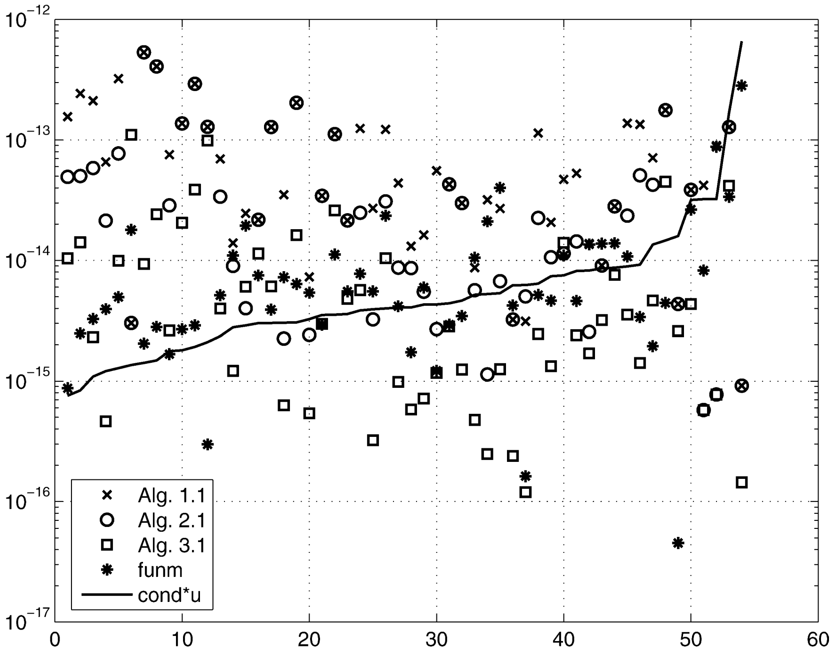

Figure 4.3. Errors for algorithms 1.1, 2.1 and 3.1 without preprocessing on matrices scaled so that $\|A\|_\infty = 25$.

a difficult case for `funm` [1]. Increasing the blocking tolerance, through the call `funm(A,@cos,struct('TolBlk',0.2))` results in an accurate evaluation.

We repeated the experiment with every matrix scaled so that $\|A\|_\infty = 25$. The results without preprocessing are shown in figure 4.3; those with preprocessing are very similar, with just a modest reduction of up to a factor 3 or so of the maximum and mean error for algorithms 1.1, 2.1 and 3.1. The performance profile is shown in figure 4.4. Clearly the (generally) larger norm causes difficulty for all the methods, but much less so for algorithm 3.1 than for algorithms 1.1 and 2.1. In this case, the means costs are 10, 9.4 and 9.1 without preprocessing and 9.6, 9.1 and 8.6 with preprocessing.

5. Computing the sine and cosine of a matrix

Suppose now that we wish to compute both $\sin(A)$ and $\cos(A)$. Since $\cos(A) = (e^{iA} + e^{-iA})/2$ and $\sin(A) = (e^{iA} - e^{-iA})/(2i)$, we can obtain both functions from two matrix exponential evaluations. However, when A is real the arguments of the exponential are complex, so this approach will not be competitive in cost even with computing $\sin(A)$ and $\cos(A)$ separately. A further disadvantage is that these formulas can suffer badly from cancellation in floating point arithmetic, as shown in [6].

We will develop an analogue of algorithm 3.1 that scales A by a power of 2, computes Padé approximants to both the sine and cosine of the scaled matrix, and then applies the double-angle formulas $\cos(2A) = 2\cos^2(A) - I$ and $\sin(2A) =$

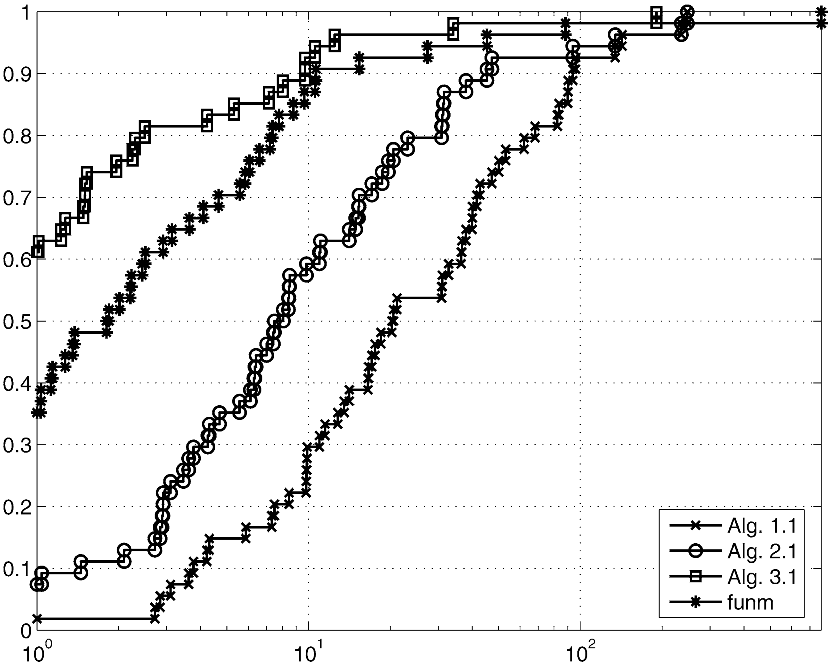

Figure 4.4. Performance profile for the four methods, without preprocessing, on the test set with matrices scaled so that $\|A\|_\infty = 25$.

$2\sin(A)\cos(A)$. Computational savings are possible in the evaluation of the Padé approximants and in the double-angle recurrences by re-using the cos terms.

Denote the $[m/m]$ Padé approximant to the sine function by $\tilde{r}_m(x) = \tilde{p}_m(x)/\tilde{q}_m(x)$. Then the error in $\tilde{r}_m$ has the form

$$\sin(A) - \tilde{r}_m(A) = \sum_{i=m}^{\infty} c_{2i+1} A^{2i+1}.$$

Since this expansion contains only odd powers of A we bound the series in terms of $\|A\|$ instead of $\theta(A)$ (cf. (2.2)):

$$\big\|\sin(A) - \tilde{r}_m(A)\big\| \leqslant \sum_{i=m}^{\infty} |c_{2i+1}|\beta^{2i}, \quad \beta = \|A\|. \tag{5.1}$$

Define β_m to be the largest value of β such that the bound (5.1) does not exceed u. Using the same technique as for the cosine, we computed the values shown in table 5.1. These values of β_m can be compared with the values of θ_{2m} in table 3.1. Although θ_{2m} is defined as the largest value of $\theta(A) = \|A^2\|^{1/2}$ such that the absolute error bound (3.2) for $\|\cos(A) - r_{2m}(A)\|$ does not exceed u, θ_{2m} can also (trivially) be regarded as the largest value of $\|A\|$ such that the bound (3.2), with θ interpreted as $\|A\|$, does not exceed u.

Table 5.1
Maximum value β_m of $\|A\|$ such that the relative error bound (5.1) does not exceed $u = 2^{-53}$.

m	2	3	4	5	6	7	8	9	10	11	12	
β_m	1.4e−3	1.8e−2	6.4e−2	1.7e−1	0.32	0.56	0.81	1.2	1.5	2.0	2.3	
m	13	14	15	16	17	18	19	20	21	22	23	24
β_m	2.9	3.3	3.9	4.4	5.0	5.5	6.2	6.7	7.4	7.9	8.7	9.2

Table 5.2
Number of matrix multiplications $\tilde{\pi}_{2m}$ to evaluate $p_{2m}(A)$, $q_{2m}(A)$, $\tilde{p}_{2m+1}(A)$ and $\tilde{q}_{2m+1}(A)$.

$2m$	2	4	6	8	10	12	14	16	18	20	22	24
$\tilde{\pi}_{2m}$	2	3	4	5	6	7	8	9	10	10	11	11

On comparing table 5.1 with table 3.1 we see that for $4 \leqslant 2m \leqslant 22$ we have $\beta_{2m} < \theta_{2m} < \beta_{2m+1}$. We could therefore scale so that $\|2^{-s}A\| \leqslant \beta_{2m}$ and then use the $[2m/2m]$ Padé approximants to the sine and cosine, or scale so that $\|2^{-s}A\| \leqslant \theta_{2m}$ and use the $[2m/2m]$ Padé approximant to the cosine and the $[2m+1/2m+1]$ Padé approximant to the sine. Since the diagonal Padé approximants to the sine have an odd numerator polynomial and an even denominator polynomial [8], and since we can write an odd polynomial in A as A times an even polynomial of degree one less, it is as cheap to evaluate $\tilde{r}_{2m+1}$ and r_{2m} as to evaluate $\tilde{r}_{2m}$ and r_{2m}. Therefore we will scale so that $\|2^{-s}A\| \leqslant \theta_{2m}$ and then evaluate r_{2m} for the cosine and $\tilde{r}_{2m+1}$ for the sine. Evaluating p_{2m}, q_{2m}, $\tilde{p}_{2m+1}$ and $\tilde{q}_{2m+1}$ reduces to evaluating four even polynomials of degree $2m$ if we write $\tilde{p}_{2m+1}$ as A times an even polynomial of degree $2m$. This can be done by forming the powers $A^2, A^4, \ldots, A^{2m}$, at a total cost of $m+1$ multiplications. However, for $2m \geqslant 20$ it is more efficient to use the schemes of the form (2.4). We summarize the cost of evaluating p_{2m}, q_{2m}, $\tilde{p}_{2m+1}$ and $\tilde{q}_{2m+1}$ for $2m = 2:2:24$ in table 5.2.

Now we consider the choice of degree, $2m$. Bounds analogous to those in table 3.2 show that $\tilde{q}_{2m+1}$ is well conditioned for $2m \leqslant 24$, and bounds for $\tilde{p}_{2m+1}$ and $\tilde{q}_{2m+1}$ analogous to those in table 3.3 suggest restricting to $2m \leqslant 20$ (the same restriction that was made in section 3 for the Padé approximants for the cosine). It is then clear from table 5.2 that we need only consider $2m = 2, 4, 6, 8, 10, 12, 14, 16, 20$. Noting that dividing A by 2 results in two extra multiplications in the double-angle phase and that increasing from one value of $2m$ to the next in our list of considered values increases the cost of evaluating the Padé approximants by one multiplication, we can determine the most efficient choice of $2m$ by a similar argument to that in the previous section. The result is that we should scale so that $\theta \leqslant \theta_{20}$, and scale further according to exactly the same strategy as in table 3.4, except for the fact that in the first line of the table "14" is added to the set of possible d values.

The algorithm can be summarized as follows.

Algorithm 5.1. Given a matrix $A \in \mathbb{C}^{n\times n}$ this algorithm approximates $C = \cos(A)$ and $S = \sin(A)$. It uses the constants θ_{2m} given in table 3.1. The matrix A can optionally be preprocessed using an obvious modification of algorithm 1.2.

```
for $d = [2\ 4\ 6\ 8\ 12\ 14\ 16]$
    if $\|A\|_\infty \leqslant \theta_d$
        $C = r_d(A)$, $S = \tilde{r}_d(A)$
        quit
    end
end
$s = \mathrm{ceil}(\log_2(\theta/\theta_{20}))$    % Find minimal integer $s$ such that $2^{-s}\theta \leqslant \theta_{20}$.
Determine optimal $d$ from modified table 3.4 (with $\theta \leftarrow 2^{-s}\theta$) and increase $s$ as
    necessary.
$C = r_d(2^{-s}A)$, $S = \tilde{r}_d(2^{-s}A)$
for $i = 1 : s$
    $S \leftarrow 2CS$, $C \leftarrow 2C^2 - I$
end
```

Cost. $(\tilde{\pi}_d + \mathrm{ceil}(\log_2(\|A\|_\infty/\theta_d)))M + D$, where d is the degree of the Padé approximants used and θ_d and $\tilde{\pi}_d$ are tabulated in tables 3.1 and 5.2, respectively.

How much work does algorithm 5.1 save compared with separate computation of $\cos(A)$ and $\sin(A) = \cos(A - \frac{\pi}{2}I)$ by algorithm 3.1? The answer is roughly $2\pi_d - \tilde{\pi}_d$ matrix multiplies, which rises from 1 when $d = 4$ to 4 when $d = 20$; the overall saving is therefore up to about 27%.

We tested algorithm 5.1 on the same set of test matrices as in section 4. Figure 5.1 compares the relative errors for the computed sine and cosine with the corresponding errors from `funm`, invoked as `funm(A,@sin)` and `funm(A,@cos)`. Note that the cost of the latter two computations can be reduced by using the same Schur decomposition in both cases. Algorithm 5.1 provides similar or better accuracy to `funm` on this test set. Its cost varies from 9 matrix multiplies and solves to 54, with an average of 16, so the algorithm can require significantly fewer flops than are needed for a single Schur decomposition.

6. Concluding remarks

We have improved the algorithm of Higham and Smith [6] in two respects: by employing variable degree Padé approximants, with the degree chosen to minimize the computational cost, and by employing truncation error bounds expressed in terms of $\|A^2\|^{1/2}$ in place of $\|A\|$. Our two improved algorithms, algorithms 2.1 and 3.1, both out-perform the Higham and Smith algorithm in accuracy and cost. Of the two, algorithm 3.1, based on an absolute error bound for the Padé approximant, emerges as the clear winner. By its design, it allows larger degree Padé approximants to be evaluated at matrices of significantly larger norm, but in so doing it does not sacrifice accuracy, as

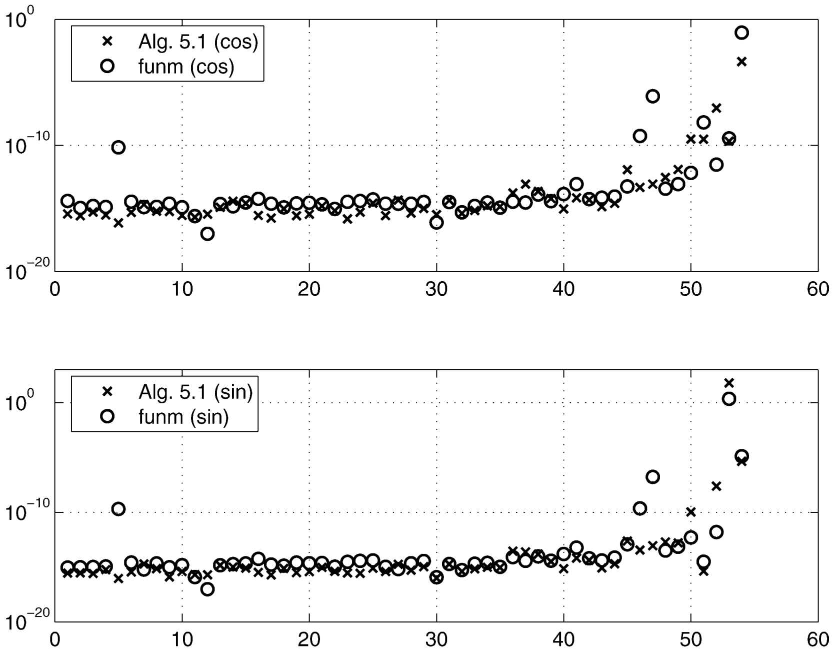

Figure 5.1. Errors for algorithm 5.1 without preprocessing and `funm`.

we have shown by analysis (see (3.1)) and experiment. Analogously to our experience with the matrix exponential [5], designing our algorithms to achieve low cost brings an added benefit of better accuracy through the need for fewer double-angle steps.

We have also shown how, using the Padé double-angle approach, $\cos(A)$ and $\sin(A)$ can be evaluated together at lower cost than if they are evaluated separately.

The design of the algorithms involved making compromises between maximizing efficiency and minimizing the effects of rounding errors. Compared with the Schur–Parlett method applied to the sine and cosine the algorithms require fewer flops unless $\|A^2\|^{1/2}$ is large, and on our test set they are generally more accurate; this provides confidence that the compromises have been well chosen.

References

[1] P.I. Davies and N.J. Higham, A Schur–Parlett algorithm for computing matrix functions, SIAM J. Matrix Anal. Appl. 25(2) (2003) 464–485.

[2] N.J. Higham, *Functions of a Matrix: Theory and Computation*, in preparation.

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

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

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

[6] N.J. Higham and M.I. Smith, Computing the matrix cosine, Numerical Algorithms 34 (2003) 13–26.

[7] C.S. Kenney and A.J. Laub, Condition estimates for matrix functions, SIAM J. Matrix Anal. Appl. 10(2) (1989) 191–209.

[8] A. Magnus and J. Wynn, On the Padé table of cos z, Proc. Amer. Math. Soc. 47(2) (1975) 361–367.

[9] M.S. Paterson and L.J. Stockmeyer, On the number of nonscalar multiplications necessary to evaluate polynomials, SIAM J. Comput. 2(1) (1973) 60–66.

[10] S.M. Serbin, Rational approximations of trigonometric matrices with application to second-order systems of differential equations, Appl. Math. Comput. 5(1) (1979) 75–92.

[11] S.M. Serbin and S.A. Blalock, An algorithm for computing the matrix cosine, SIAM J. Sci. Statist. Comput. 1(2) (1980) 198–204.