
Numerical Algorithms **34:** 13–26, 2003.

Computing the matrix cosine

Nicholas J. Higham * and Matthew I. Smith **

Department of Mathematics, University of Manchester, Manchester, M13 9PL, England
E-mail: {higham;msmith}@ma.man.ac.uk

Received 23 September 2002; accepted 9 May 2003
Communicated by C. Brezinski

An algorithm is developed for computing the matrix cosine, building on a proposal of Serbin and Blalock. The algorithm scales the matrix by a power of 2 to make the ∞-norm less than or equal to 1, evaluates a Padé approximant, and then uses the double angle formula $\cos(2A) = 2\cos(A)^2 - I$ to recover the cosine of the original matrix. In addition, argument reduction and balancing is used initially to decrease the norm. We give truncation and rounding error analyses to show that an [8,8] Padé approximant produces the cosine of the scaled matrix correct to machine accuracy in IEEE double precision arithmetic, and we show that this Padé approximant can be more efficiently evaluated than a corresponding Taylor series approximation. We also provide error analysis to bound the propagation of errors in the double angle recurrence. Numerical experiments show that our algorithm is competitive in accuracy with the Schur–Parlett method of Davies and Higham, which is designed for general matrix functions, and it is substantially less expensive than that method for matrices of ∞-norm of order 1. The dominant computational kernels in the algorithm are matrix multiplication and solution of a linear system with multiple right-hand sides, so the algorithm is well suited to modern computer architectures.

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

AMS subject classification: 65F30

1. Introduction

Motivated by the need to solve numerically a certain class of second order ordinary differential equations, Serbin and Blalock [19] propose an algorithm for computing the cosine of a matrix $A \in \mathbb{R}^{n\times n}$. The algorithm makes use of rational approximations of $\cos(A)$, of which we will consider Taylor series and Padé approximations here. Since these approximations are accurate only near the origin, the norm of A is initially reduced using a technique analogous to that employed in the scaling and squaring method for the matrix exponential [6, section 11.3; 20].

* This work was supported by Engineering and Physical Sciences Research Council grant GR/R22612.
** This work was supported by an Engineering and Physical Sciences Research Council Ph.D. Studentship.

Let us define

$$C_i = \cos\bigl(2^{i-m}A\bigr).$$

The value m is chosen so that $2^{-m}A$ has norm small enough that we can obtain a good approximation of $C_0 = \cos(2^{-m}A)$ via rational approximation. By applying the cosine double angle formula[1], $\cos(2A) = 2\cos^2(A) - I$, we can compute $C_m = \cos(A)$ from C_0 using the recurrence

$$C_{i+1} = 2C_i^2 - I, \quad i = 0, \ldots, m-1. \tag{1.1}$$

The algorithm of Serbin and Blalock [19] can be summarized as follows.

Algorithm 1.1. Given a matrix $A \in \mathbb{R}^{n\times n}$ and a parameter $\alpha > 0$ this algorithm approximates $\cos(A)$.
Choose m such that $2^{-m}\|A\| \approx \alpha$.
$C_0 =$ Taylor or Padé approximation to $\cos(A/2^m)$.
for $i = 0, \ldots, m-1$
  $C_{i+1} = 2C_i^2 - I$
end.

No specific choice of α is suggested by Serbin and Blalock [19]; the value $\alpha = 1$ is proposed for the ∞-norm by Golub and Van Loan [6, section 11.2.3].

The purpose of this work is to develop algorithm 1.1 into a practical algorithm with an automatic choice of both m and the rational approximation that balances the conflicting requirements of minimizing the computational cost and maximizing the accuracy. To do so we develop preprocessing to initially reduce the norm (section 2); derive truncation and rounding error bounds for Taylor series approximants and the [8,8] Padé approximant (sections 3 and 4); identify the most efficient ways of evaluating these approximants (sections 4 and 6); and give a rounding error analysis of the double angle recurrence (section 5). Our algorithm is given in section 6. The new algorithm is substantially less expensive for matrices with ∞-norm of order 1 than the Schur–Parlett algorithm of Davies and Higham [5] (applicable to general matrix functions), and in section 7 we present numerical experiments that show the new algorithm to be competitive in accuracy, too.

In our error analysis we will use the standard model of floating point arithmetic [8, section 2.2]. We denote by u the unit roundoff and write

$$\gamma_k = \frac{ku}{1-ku}, \qquad \widetilde{\gamma}_k = \frac{cku}{1-cku},$$

with c a small integer constant whose exact value is unimportant. The most convenient norm to work with in our analysis is the ∞-norm, so we will use this norm throughout.

[1] For matrices the addition formula $\cos((A+B)t) = \cos(At)\cos(Bt) - \sin(At)\sin(Bt)$ holds for all t if and only if $A = B$, the latter case giving the double angle formula.

2. Preprocessing

Before carrying out the steps of algorithm 1.1 it is important to reduce the norm of A by any available transformations of low computational cost. Two norm-reducing techniques implemented by Ward [20] for the exponential are appropriate. First, we can exploit the periodicity relation

$$\cos(A - \pi q I) = (-1)^q \cos(A), \quad q \in \mathbb{Z}.$$

Since we are working with the ∞-norm, we would like to choose q to minimize $\|A - \pi q I\|_\infty$. How to compute efficiently the optimal q is not clear. However, the shift θ that minimizes the Frobenius norm measure $\|A - \theta I\|_F$ over all θ is easily seen (by solving this mini-least squares problem) to be $\theta = n^{-1}\,\mathrm{trace}(A)$. We therefore take q to be whichever of the two integers nearest to $\mathrm{trace}(A)/(n\pi)$ gives the smaller value of $\|A - \pi q I\|_\infty$, or do not shift at all if neither choice reduces the ∞-norm of A.

This procedure is argument reduction. It is well known that in the computation of scalar elementary functions special techniques must be used to avoid severe loss of accuracy in argument reduction for large arguments [16, chapter 8]. The standard techniques are not directly applicable to the matrix case. Therefore we will implement argument reduction in the straightforward way described above, while recognizing that it is a potentially significant source of error. We note that a general treatment of matrix argument reduction is given by Ng [17], but the computational approach there uses the Schur form, which we are eschewing in this work.

Another technique that can help in norm reduction is balancing. The balancing algorithms in LAPACK [1] and MATLAB [13] compute $B = D^{-1}AD$, where D is a permuted diagonal matrix chosen so that the 1-norms of the ith row and ith column of B are of similar magnitude for all i. Balancing is an $O(n^2)$ calculation that can be performed without roundoff. Balancing tends to reduce the norm, though this is not guaranteed, so our strategy is to replace A by the balanced B only if $\|B\|_\infty < \|A\|_\infty$.

To summarize, the preprocessing consists of computing

$$\widetilde{A} = D^{-1}(A - \pi q I)D,$$

and we eventually recover $\cos A$ from

$$\cos(A) = (-1)^q D \cos\bigl(\widetilde{A}\bigr) D^{-1}.$$

3. Taylor series

The matrix cosine $\cos(A)$ has a Taylor series valid for all A, and by truncating the series at the $(2k+1)$st term we obtain the approximation

$$T_{2k}(A) = \sum_{i=0}^{2k} \frac{(-1)^i}{(2i)!} A^{2i}. \tag{3.1}$$

The error in this approximation can be bounded with the aid of the following result from [6, theorem 11.2.4].

Theorem 3.1. If $f(z)$ has the Taylor series

$$f(z) = \sum_{i=0}^{\infty} \alpha_i z^i$$

on an open disk containing the eigenvalues of $A \in \mathbb{C}^{n\times n}$ then

$$\left\| f(A) - \sum_{i=0}^{k} \alpha_i A^i \right\|_\infty \leqslant \frac{n}{(k+1)!} \max_{0\leqslant s\leqslant 1} \left\| A^{k+1} f^{(k+1)}(sA) \right\|_\infty.$$

For the cosine function the bound of theorem 3.1 becomes

$$\begin{aligned} \left\|\cos(A) - T_{2k}(A)\right\|_\infty &\leqslant \frac{n}{(2k+2)!} \max_{0\leqslant s\leqslant 1} \left\| A^{2k+2} \cos^{(2k+2)}(sA) \right\|_\infty \\ &\leqslant \frac{n}{(2k+2)!} \left\| A^{2k+2} \right\|_\infty \max_{0\leqslant s\leqslant 1} \left\| \cos^{(2k+2)}(sA) \right\|_\infty. \end{aligned}$$

Now

$$\begin{aligned} \max_{0\leqslant s\leqslant 1} \left\|\cos^{(2k+2)}(sA)\right\|_\infty &= \max_{0\leqslant s\leqslant 1} \left\|\cos(sA)\right\|_\infty \\ &\leqslant 1 + \frac{\|A\|_\infty^2}{2!} + \frac{\|A\|_\infty^4}{4!} + \cdots = \cosh\bigl(\|A\|_\infty\bigr), \end{aligned}$$

and so the error in the truncated Taylor series approximation to the matrix cosine satisfies the bound

$$\left\|\cos(A) - T_{2k}(A)\right\|_\infty \leqslant \frac{n\|A^{2k+2}\|_\infty}{(2k+2)!} \cosh\bigl(\|A\|_\infty\bigr). \tag{3.2}$$

We also need to bound the error in evaluating $T_{2k}(A)$ in floating point arithmetic. It is easy to show that whether $T_{2k}(A)$ is evaluated by Horner's rule, or by explicitly forming the powers of A, the computed $\widehat{T}_{2k}$ satisfies

$$\left\| T_{2k} - \widehat{T}_{2k} \right\|_\infty \leqslant \widetilde{\gamma}_{kn} \cosh\bigl(\|A\|_\infty\bigr).$$

Hence

$$\frac{\|\cos(A) - \widehat{T}_{2k}\|_\infty}{\|\cos(A)\|_\infty} \leqslant \left(\frac{n\|A^{2k+2}\|_\infty}{(2k+2)!} + \widetilde{\gamma}_{kn} \right) \frac{\cosh(\|A\|_\infty)}{\|\cos(A)\|_\infty}. \tag{3.3}$$

Note that $\|\cos(A)\|_\infty \geqslant 1 - (\cosh(\|A\|_\infty) - 1) = 2 - \cosh(\|A\|_\infty)$, so for $\|A\|_\infty \leqslant 1$ we have

$$0.45 \leqslant 2 - \cosh(1) \leqslant \left\|\cos(A)\right\|_\infty \leqslant \cosh(1) \leqslant 1.55, \tag{3.4}$$

which gives

$$\frac{\cosh(\|A\|_\infty)}{\|\cos(A)\|_\infty} \leqslant 3.4.$$

We conclude that a relative error $\|\cos(A) - \widehat{T}_{2k}\|_\infty / \|\cos(A)\|_\infty$ of order $\widetilde{\gamma}_{kn}$ is guaranteed if $\|A\|_\infty \leqslant 1$ and k is sufficiently large. In fact, since $18! \approx 6 \times 10^{15}$, $k = 8$ suffices in IEEE standard double precision arithmetic, for which the unit roundoff $u \approx 1.1 \times 10^{-16}$.

4. Padé approximants

For a given scalar function $f(x) = \sum_{i=0}^{\infty} \alpha_i x^i$ the rational function

$$r_{km}(x) = \frac{p_{km}(x)}{q_{km}(x)}$$

is a $[k, m]$ Padé approximant of f if p_{km} is a polynomial in x of degree at most k, q_{km} is a polynomial in x of degree at most m and

$$f(x) - r_{km}(x) = \mathrm{O}\big(x^{k+m+1}\big).$$

In addition, we usually require that p_{km} and q_{km} have no common zeros and that $q_{km}(0) = 1$. These conditions ensure that if a $[k, m]$ Padé approximant exists then it is unique; see [2,3]. We restrict our attention to the main diagonal Padé approximants ($k = m$) of the cosine function, $f(x) = \cos(x)$.

In comparison with the exponential and logarithm functions, relatively few results are available concerning Padé approximants of the cosine function. In particular, we are not aware of a proof of existence of the Padé approximants for all k and m. Nevertheless, particular approximants are readily calculated with the aid of the Padé approximation functions in symbolic manipulation packages such as Maple [11] and Mathematica [12], or via formulae of Magnus and Wynn [10] that give the coefficients of p_{km} and q_{km} in terms of determinants of matrices whose entries are binomial coefficients. The first two Padé approximants are

$$r_{22}(x) = \frac{1 - (5/12)x^2}{1 + (1/12)x^2},$$
$$r_{44}(x) = \frac{1 - (115/252)x^2 + (313/15120)x^4}{1 + (11/252)x^2 + (13/15120)x^4}.$$

Thereafter the numerators and denominators of the rational coefficients grow rapidly in size.

For a Padé approximant to be suitable for use in algorithm 1.1 it must provably have relative error of order u for the parameter α of interest and the error in evaluating the approximant must also be of order u. We are not aware of a useful expression or bound for the error $\cos(x) - r_{mm}(x)$ for arbitrary m. Therefore we will adopt an ad hoc

approach whereby we show that a particular approximant has the desired properties. We consider the approximant

$$r_{88}(x) = \frac{1 - \frac{260735}{545628}x^2 + \frac{4375409}{141863280}x^4 - \frac{7696415}{13108167072}x^6 + \frac{80737373}{23594700729600}x^8}{1 + \frac{12079}{545628}x^2 + \frac{34709}{141863280}x^4 + \frac{109247}{65540835360}x^6 + \frac{11321}{1814976979200}x^8}$$
$$=: \frac{\sum_{i=0}^{8} \pi_i x^i}{\sum_{i=0}^{8} \mu_i x^i} = \frac{p_{88}(x)}{q_{88}(x)}, \tag{4.1}$$

and we take $\alpha = 1$ in algorithm 1.1, that is, we require that $\|A\|_\infty \leqslant 1$. Key to the analysis of r_{88} is the fact that the coefficients μ_i ($i \geqslant 2$) of q_{88} are small:

$$\begin{bmatrix} \mu_2 & \mu_4 & \mu_6 & \mu_8 \end{bmatrix} = \begin{bmatrix} 2.2\,\mathrm{e}{-2} & 2.5\,\mathrm{e}{-4} & 1.7\,\mathrm{e}{-6} & 6.2\,\mathrm{e}{-9} \end{bmatrix}.$$

An immediate implication is that the denominator polynomial q_{88} is perfectly conditioned: we have

$$\left\| q_{88}(A) \right\|_\infty \leqslant \sum_{i=0}^{8} |\mu_i| = 1.02$$

and, using the inequality $\|(I + E)^{-1}\|_\infty \leqslant (1 - \|E\|_\infty)^{-1}$ for $\|E\|_\infty < 1$,

$$\left\| q_{88}(A)^{-1} \right\|_\infty \leqslant \frac{1}{1 - \|\mu_2 A^2 + \mu_4 A^4 + \mu_6 A^6 + \mu_8 A^8\|_\infty}$$
$$\leqslant \frac{1}{1 - |\mu_2| - |\mu_4| - |\mu_6| - |\mu_8|} = 1.02,$$

so $\kappa(q_{88}) \leqslant 1.04$. To investigate the accuracy of r_{88} we write

$$e(A) := \cos(A) - r_{88}(A) = \big(\cos(A) q_{88}(A) - p_{88}(A)\big) q_{88}(A)^{-1},$$

from which

$$\left\| e(A) \right\|_\infty \leqslant 1.02 \left\| \cos(A) q_{88}(A) - p_{88}(A) \right\|_\infty.$$

Since r_{88} is the [8,8] Padé approximant,

$$\cos(A) q_{88}(A) - p_{88}(A) = \sum_{i=9}^{\infty} c_{2i} A^{2i}.$$

The question is whether the constants c_{2i} are large. Using MATLAB's Symbolic Math Toolbox [14] we find

$$\sum_{i=9}^{32} |c_{2i}| = 1.46 \times 10^{-16},$$

with c_{64} of order 10^{-80}. By bounding the tail of the sum it is not hard to prove that $\sum_{i=9}^{\infty} |c_{2i}| = 1.46 \times 10^{-16}$, correct to 3 significant figures, and so

$$\left\| e(A) \right\|_\infty \leqslant 1.02 \times 1.46 \times 10^{-16} = 1.49 \times 10^{-16}.$$

Using (3.4), we have, finally,

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

Given that errors of order at least nu will inevitably be introduced in numerical evaluation of r_{88}, we conclude that this Padé approximant is accurate enough for use in algorithm 1.1 with $\alpha = 1$ in IEEE double precision arithmetic.

Now we consider the cost of evaluating r_{88}. As explained in [7] in the context of the matrix logarithm, several techniques are available for evaluating Padé approximants at a matrix argument. Two of the most interesting techniques are evaluation of continued fraction and partial fraction expansions. In the natural continued fraction expansion of r_{88} the numerators and denominators of the rational coefficients inconveniently have up to 67 decimal digits, which makes this expansion unattractive. The denominator polynomial q_{88} does not factor into linear factors over $\mathbb{R}$ (it has roots $\pm 6.44 \pm 8.74\,\mathrm{i}$, $\pm 2.09 \pm 1.02\,\mathrm{i}$), which rules out a linear partial fraction form, and we have not found any other convenient partial fraction expansion. Our favoured way of evaluating r_{88} is perhaps the most obvious.

Algorithm 4.1. Given a matrix $A \in \mathbb{R}^{n\times n}$ this algorithm evaluates $R = r_{88}(A)$, where r_{88} is defined in (4.1).

$A_2 = A^2$
$A_4 = A_2^2$
$A_6 = A_2 A_4$
$A_8 = A_4^2$
$P = \pi_0 I + \pi_2 A_2 + \pi_4 A_4 + \pi_6 A_6 + \pi_8 A_8$
$Q = \mu_0 I + \mu_2 A_2 + \mu_4 A_4 + \mu_6 A_6 + \mu_8 A_8$
Solve $QR = P$.

The cost of algorithm 4.1 is $4M + D$, where M denotes a matrix multiplication and D the solution of a linear system with n right-hand side vectors. If Horner's method (nested multiplication) is used to evaluate p_{88} and q_{88} the cost is significantly more: $7M + D$. A method of Paterson and Stockmeyer [6, section 11.2.4; 18] offers a reduction in cost over Horner's method at the expense of using more storage, but it brings only a modest reduction here, to $6M + D$, so algorithm 4.1 remains the least expensive option.

The modestly sized coefficients π_i and μ_i, and the perfect conditioning of q_{88} for $\|A\|_\infty \leqslant 1$, cause the influence of rounding errors on algorithm 4.1 to be as small as could be hoped: the relative error due to roundoff is easily shown to be of order nu.

5. Error analysis

We now analyze the stability of the double angle recurrence (1.1). The following analysis incorporates the error in the initial approximation of $C_0 = \cos(2^{-m}A)$ and the rounding errors introduced on each step of the recurrence.

The computed quantities $\widehat{C}_i$ satisfy

$$\widehat{C}_{i+1} = fl\big(2\widehat{C}_i^2 - I\big) = 2\widehat{C}_i^2 - I + R_i, \tag{5.1}$$

where

$$\|R_i\|_\infty \leqslant \gamma_{n+1}\big(2\big\|\widehat{C}_i\big\|_\infty^2 + 1\big). \tag{5.2}$$

Write $\widehat{C}_i = C_i + E_i$, where $C_0 = \cos(2^{-m}A)$ and the C_i satisfy (1.1). Then (5.1) gives

$$E_{i+1} = 2\big(E_i^2 + E_iC_i + C_iE_i\big) + R_i.$$

By taking norms we obtain

$$\|E_{i+1}\|_\infty \leqslant 2\|E_i\|_\infty\big(\|E_i\|_\infty + 2\|C_i\|_\infty\big) + \|R_i\|_\infty. \tag{5.3}$$

For simplicity, we now make the reasonable assumption that $\|E_i\|_\infty \leqslant 0.05\|C_i\|_\infty$. Then (5.3) gives

$$\|E_{i+1}\|_\infty \leqslant 4.1\|E_i\|_\infty\|C_i\|_\infty + \|R_i\|_\infty. \tag{5.4}$$

This recurrence is easily solved to give

$$\begin{aligned}
\|E_{i+1}\|_\infty &\leqslant (4.1)^{i+1}\|E_0\|_\infty\|C_0\|_\infty\|C_1\|_\infty\cdots\|C_i\|_\infty \\
&\quad + \sum_{j=0}^{i}(4.1)^{i-j}\|R_j\|_\infty\|C_{j+1}\|_\infty\cdots\|C_i\|_\infty \\
&\leqslant (4.1)^{i+1}\|E_0\|_\infty\|C_0\|_\infty\|C_1\|_\infty\cdots\|C_i\|_\infty \\
&\quad + \gamma_{n+1}\sum_{j=0}^{i}4.1^{i-j}\big(2.21\|C_j\|_\infty^2 + 1\big)\|C_{j+1}\|_\infty\cdots\|C_i\|_\infty.
\end{aligned} \tag{5.5}$$

Recall that bounds for $\|E_0\|_\infty$ are given in sections 3 and 4.

Suppose, first, that A is normal with real eigenvalues. Then $\|C_i\|_\infty \leqslant 1$ for all i and (5.5) yields

$$\|E_m\|_\infty \leqslant (4.1)^m\big(\|E_0\|_\infty + \gamma_{n+1}\big).$$

With the rounding term omitted this is essentially the bound obtained by Serbin and Blalock [19]. This bound reflects the fact that, because the double angle recurrence multiplies the square of the previous iterate by 2 at each stage, the errors could grow by a factor 4 at each stage, though this worst-case growth is clearly extremely unlikely.

Turning now to general A, we will simplify the bound (5.5) slightly. Since we are computing C_0 to essentially full precision, $\|E_0\|_\infty \leqslant \gamma_{n+1}\|C_0\|_\infty$, and we can rewrite (5.5) as

$$\frac{\|E_m\|_\infty}{\|C_m\|_\infty} \lesssim \gamma_{n+1}\, g(4.1), \qquad g(\theta) = \max_{j=0:m-1}\theta^{m-j}\,\frac{\|C_j\|_\infty\|C_{j+1}\|_\infty\cdots\|C_{m-1}\|_\infty}{\|C_m\|_\infty}. \tag{5.6}$$

We have introduced the parameter θ in g because while the analysis requires $\theta = 4.1$, in practice a much smaller value of θ will usually give more realistic bounds. The key quantity is the ratio of norms in $g(\theta)$. We know that $\|C_0\|_\infty \approx 1$, by (3.4). But little can be said about the behaviour of the sequence $\|C_0\|_\infty, \|C_1\|_\infty, \ldots, \|C_m\|_\infty$. First, note that the C_i are unbounded in norm: cos is unbounded in the complex plane and, moreover, nonnormality of A can cause $\cos(A)$ to greatly exceed $\max\{|\cos(\lambda)|\colon \lambda \in \lambda(A)\}$, where $\lambda(A)$ denotes the set of eigenvalues of A. It is easy to see that $g(\theta)$ is unbounded: the denominator can be arbitrarily close to zero while the numerator is bounded away from zero. For example, for $n = 1$ and $A = \pi/2 + \varepsilon$, with $0 < \varepsilon \ll 1$, algorithm 1.1 with $\alpha = 1$ takes $m = 1$ and then

$$g(\theta) = \frac{\|C_0\|_\infty}{\|C_1\|_\infty} = \frac{\cos(\pi/4 + \varepsilon/2)}{\cos(\pi/2 + \varepsilon)} \to \infty \quad \text{as } \varepsilon \to 0.$$

Therefore large $g(\theta)$ can occur even for normal matrices – unlike the hump phenomenon for the matrix exponential [15], whereby in the identity $\exp(A) = \exp(A/m)^m$ the unwanted circumstance that $\|\exp(A)\|_\infty \ll \|\exp(A/m)\|_\infty^m$ can happen only for nonnormal A.

In summary, the error bound for the double angle recurrence contains two main parts:

1. Powers of 4.1 up to the mth, which are independent of A. They are innocuous if m is small, and are likely to be very pessimistic, otherwise.
2. A term that depends in a complicated way on the norms of the intermediate C_i, and which is difficult to bound a priori.

Ideally, we would relate the error bound to the conditioning of the $\cos(A)$ problem, which can be measured by the condition number

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

Unfortunately, as is the case for the matrix exponential [6, section 11.3] and the matrix logarithm [4], no convenient characterization of $\operatorname{cond}(A)$ is known, so we are unable to determine theoretically whether the method delivers the accuracy that the condition of the problem warrants.

6. Algorithm

We are now in a position to choose the details of our algorithm for computing $\cos(A)$.

First, we must choose between the use of a Taylor series approximant and a Padé approximant in algorithm 1.1. We also need to consider how to scale A and whether to choose an approximation of fixed degree, or whether to vary the degree in order to minimize the total cost, as is done by Cheng et al. [4] in a Padé approximation-based algorithm for the matrix logarithm.

We will assume that the approximations are applied only when $\|A\|_\infty \leqslant 1$ (as our analysis for r_{88} assumed). We know from section 3 that a Taylor series of degree 16 is needed to guarantee the required accuracy. The cost of evaluating this approximation is $8M$ if we use Horner's method, dropping to $5M$ with the use of the Paterson–Stockmeyer method, though the latter method requires $2n^2$ extra elements of storage. Therefore in terms of overall cost and storage the Padé approximation r_{88}, which costs $4M + D$, is preferable to a Taylor series approximation. We can consider additional scaling once $\|A\|_\infty \leqslant 1$ in the hope that a lower degree approximation can be used. However, each scaling requires an extra matrix multiplication in the final double angle recurrence, while the saving in evaluating r_{66} (say) over r_{88} is just one matrix multiplication. Furthermore, as the analysis in the previous section shows, to minimize the overall error bound we need to minimize the number of scalings. We choose, therefore, not to allow the possibility of additional double angle steps. (In the matrix logarithm work in [4] the economics are very different because the analogue of the scaling step is a matrix square root which, when computed iteratively, has a cost that depends strongly on the number of square roots already computed.) Finally, as already shown, both r_{88} and the Taylor series are evaluated accurately given that $\|A\|_\infty \leqslant 1$.

These deliberations lead to the following algorithm.

Algorithm 6.1. Given a matrix $A \in \mathbb{R}^{n\times n}$ this algorithm approximates $X = \cos(A)$. It is intended for use in IEEE double precision arithmetic.

$A \leftarrow A - \pi q I$, where q is whichever of 0, $\lfloor \mathrm{trace}(A)/(n\pi)\rfloor$ and $\lceil \mathrm{trace}(A)/(n\pi)\rceil$ yields the smaller value of $\|A - \pi q I\|_\infty$.
$B = D^{-1}AD$, where D balances A.
if $\|B\|_\infty < \|A\|_\infty$, $A = B$, end
Choose m so that $2^{-m}\|A\|_\infty \leqslant 1 < 2^{-(m-1)}\|A\|_\infty$.
$C_0 = r_{88}(2^{-m}A)$, where r_{88} is the Padé approximant (4.1) evaluated by algorithm 4.1.
for $i = 0 : m-1$
  $C_{i+1} = 2C_i^2 - I$
end
$X = (-1)^q C_m$
if balancing was performed, $X = DXD^{-1}$, end.

The total cost of algorithm 6.1 is $(4 + \lceil \log_2(\|A\|_\infty)\rceil)M + D$.

7. Numerical experiments

We present some numerical experiments designed to provide insight into algorithm 6.1 and to illustrate its performance in comparison with the general purpose $f(A)$ algorithm of Davies and Higham [5]. The latter algorithm computes a Schur decomposition, re-orders and blocks the Schur form, and applies a block form of Parlett's recurrence to the triangular Schur factor, with functions of the nontrivial diagonal blocks evaluated via a Taylor series. The cost of this Schur–Parlett algorithm is roughly between

$28n^3$ flops and $n^4/3$ flops. Algorithm 6.1 has a smaller flop count if $\|A\|_\infty \leqslant 512$, and since the algorithm is dominated by matrix multiplication and the solution of multiple right-hand side systems it can exploit modern computer architectures particularly well.

Another relevant comparison is with the method that evaluates

$$\cos(A) = \mathrm{Re}\, \mathrm{e}^{\mathrm{i}A}, \tag{7.1}$$

with the matrix exponential evaluated by the scaling and squaring method [6, section 11.3; 20]. This method is not, in general, a serious competitor to algorithm 6.1 for two main reasons: it uses about twice as many complex arithmetic flops as algorithm 6.1 uses real flops, and it can clearly suffer badly from cancellation when $\mathrm{e}^{\mathrm{i}A}$ is large but $\cos(A)$ is small.

The experiments were carried out in MATLAB, which uses IEEE double precision arithmetic. In computing errors we take the "exact" $\cos(A)$ to be an approximation obtained by using MATLAB's Symbolic Math Toolbox (which invokes the Maple kernel) to evaluate the formula (7.1) at high precision. We are primarily interested in the relative error

$$\mathrm{err} = \frac{\|X - \widehat{X}\|_\infty}{\|X\|_\infty}.$$

The first test matrix is the 16×16 Frank matrix (MATLAB's `gallery('frank',16)`). In table 1 we show the results for algorithm 6.1 both with and without the preprocessing step. In the fifth column, A_0 is the matrix after the preprocessing step, that is, the matrix to which the 2^{-m} scaling is applied; in the third column, err_bound is the error bound in (5.6) and $g(1)$ is an evaluation of the function g in (5.6). In this test we see that algorithm 6.1 produces a much more accurate result than the Schur–Parlett algorithm. The preprocessing step produces a useful reduction in the norm (most of which is due to the balancing), which reduces the amount of scaling needed and benefits the error. The error bound err_bound is pessimistic, but $g(1)u$ predicts the error well. An estimate of the condition number cond, computed with the finite-difference power method proposed by Kenney and Laub [9], is shown in the table.

The second test matrix is the 8×8 Pascal matrix (MATLAB's `pascal(8)`), which is symmetric positive definite. The results are shown in table 2. The Schur–Parlett algorithm is the clear winner in accuracy, which is not surprising since the algorithm simply reduces to evaluating $\cos(A)$ from the spectral decomposition. Again, preprocessing reduces the error. In this example, $g(1)u$ is much too optimistic as an error estimate. The

Table 1
Results for 16×16 Frank matrix; $\|A\|_\infty = 1.4\,\mathrm{e}2$, $\mathrm{cond}(A) \approx 10^5$.

	Algorithm 6.1					Schur–Parlett	(7.1)
Preprocessing	err	err_bound	$g(1)u$	$\|A_0\|_\infty$	m	err	err
Yes	8.9 e−14	1.8 e−10	2.0 e−15	5.1 e1	6	4.1 e−10	5.9 e−14
No	2.2 e−12	9.2 e−7	6.4 e−13	1.4 e2	8		

Table 2
Results for 8×8 Pascal matrix; $\|A\|_\infty = 6.4\,\mathrm{e}3$, $\mathrm{cond}(A) \approx 10^3$.

Algorithm 6.1						Schur–Parlett	(7.1)
Preprocessing	err	err_bound	$g(1)u$	$\|A_0\|_\infty$	m	err	err
Yes	1.7 e−11	6.9 e−6	7.7 e−15	5.8 e3	13	6.7 e−13	1.8 e−12
No	1.4 e−8	3.6 e−5	4.0 e−14	6.4 e3	13		

Table 3
Results for `gallery('invol',8)*8*pi`; $\|A\|_\infty = 1.1\,\mathrm{e}7$, $\mathrm{cond}(A) \approx 10^{10}$. The results in the third row were obtained by imposing $m = 8$.

Algorithm 6.1						Schur–Parlett	(7.1)
Preprocessing	err	err_bound	$g(1)u$	$\|A_0\|_\infty$	m	err	err
Yes	1.3 e−9	4.3 e−8	1.1 e−16	2.2 e6	22	1.6 e−10	6.6 e2
No	2.3 e−7	1.1 e−5	1.1 e−16	1.1 e7	24		
Yes	5.7 e−12	1.3 e−13	1.1 e−16	2.2 e6	8		

error for algorithm 6.1 exceeds $\mathrm{cond}(A)u$, so the method is performing unstably on this example.

The third matrix is, in MATLAB notation, `gallery('invol',8)*8*pi`; the results are shown in table 3. This matrix has eigenvalues 8π and -8π, each repeated four times, and $\cos(A) = I$. Since the eigenvalues are multiples of π, this example might appear to be one for which for which range reduction would work well. However, since the eigenvalues are symmetric about the origin, a shift that reduces one eigenvalue increases the other, and in any case the high non-normality of the matrix means that consideration of eigenvalues alone is not sufficient. Algorithm 6.1 and the Schur–Parlett algorithm both produce much smaller errors than would be expected, given the condition of the problem. The value of err_bound is smaller than might be expected in view of the fact that $4.1^{22} \approx 10^{13}$; the reason is that the computed C_{m-3} has norm of order 10^{-6}, and this term multiplies the large powers of 4.1 in (5.6). Interestingly, if we force $m = 8$ (which corresponds to evaluating the Padé approximant at a matrix of norm 8.7 e3), algorithm 6.1 produces a much smaller error, as shown in the third row of table 3. The reason is that $\|A^2\|_\infty \approx 6 \times 10^2 \ll \|A\|_\infty \approx 1 \times 10^7$, which make the terms of the Taylor series of $\cos(A)$ decrease much faster than is usual for matrices of such a norm; hence the Padé approximation r_{88} delivers good accuracy even though it is applied here to a matrix with norm much greater than 1.

Our final example is a matrix constructed by the MATLAB code

```
randn('seed',1)
D = pi/2*(eye(8)+diag(1:8)*1e-8);
X = gallery('randsvd',8,1e2);
A = X*D/X;
```

Table 4
Results for 8×8 matrix with eigenvalues close to $\pi/2$; $\|A\|_\infty = 1.6\,\mathrm{e}0$, $\mathrm{cond}(A) \approx 10^6$.

	Algorithm 6.1					Schur–Parlett	(7.1)
Preprocessing	err	err_bound	$g(1)u$	$\|A_0\|_\infty$	m	err	err
Yes	2.4 e−10	1.2 e−9	4.2 e−11	1.6 e0	1	1.9 e−9	1.4 e−10
No	3.0 e−10	6.7 e−10	2.3 e−11	1.6 e0	1		

The matrix has eigenvalues very close to $\pi/2$ and a fairly well conditioned eigenvector matrix. This matrix illustrates the phenomenon identified in section 5 whereby an intermediate C_i can be of much larger norm than the final C_m, as can be seen in table 4 by the fact that $g(1)$ in (5.6) is large. Nevertheless, the error is still no larger than $\mathrm{cond}(A)u$, so algorithm 6.1 is performing stably.

The exponential-based method that evaluates (7.1) yields good accuracy on the first, second and fourth matrices, but its flaw is apparent on the third matrix, where massive cancellation results in a computed result with no correct digits.

The error bound in (5.6) can clearly be pessimistic, but for the third and fourth matrices it provides reasonable estimates of the actual error; on the other hand, the values of $g(1)u$ show that ignoring the potential 4.1^m error growth can lead to gross underestimates of the error.

Finally, we note that scaling so that $\|2^{-m}A\|_\infty \gg 1$ and evaluating the Taylor series until numerical convergence, that is, until additional terms do not affect the computed sum, can give acceptable accuracy. To illustrate, we applied algorithm 1.1 with different m, evaluating the Taylor series until numerical convergence. For the Frank matrix, with $m = 4$ ($\|2^{-m}A\|_\infty = 8.5$) we obtained an error of 2.6 e–14 with 27 terms of the Taylor series, this being the minimal error over all m. For the third matrix, a minimal error of 5.7 e–12 was obtained with $m = 2$ ($\|2^{-m}A\|_\infty = 2.8\,\mathrm{e}6$), using 39 terms of the Taylor series. There are two difficulties with trying to develop an algorithm that applies the basic approximation to a matrix of norm greater than 1. First, without knowledge of the conditioning of the problem it is impossible to know how large an error (the combination of rounding and truncation errors) should be allowed. (Recall that algorithm 6.1 evaluates the basic approximation to machine accuracy.) Second, even if the conditioning is known, it is unclear how to develop an algorithm that achieves the (now known) desired relative error with reasonable cost. In particular, $\|\cos(2^{-m}A)\|_\infty$ is not known a priori, which makes using the bound (3.3) to determine a suitable m in in advance of evaluation of the Taylor series of the scaled A problematic.

Acknowledgements

We thank John Coleman for pointing out the reference [10].

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., *Essentials of Padé Approximants* (Academic Press, New York, 1975).

[3] G.A. Baker, Jr. and P. Graves-Morris, *Padé Approximants*, Encyclopedia of Mathematics and Its Applications, 2nd ed. (Cambridge Univ. Press, Cambridge, 1996).

[4] S.H. Cheng, N.J. Higham, C.S. Kenney and A.J. Laub, Approximating the logarithm of a matrix to specified accuracy, SIAM J. Matrix Anal. Appl. 22(4) (2001) 1112–1125.

[5] P.I. Davies and N.J. Higham, A Schur–Parlett algorithm for computing matrix functions, Numerical Analysis Report No. 404, Manchester Centre for Computational Mathematics, Manchester, England (July 2002), revised March 2003, to appear in SIAM J. Matrix Anal. Appl.

[6] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins Univ. Press, Baltimore, MD, 1996).

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

[8] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. (SIAM, Philadelphia, PA, 2002).

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

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

[11] Maple, Waterloo Maple Inc., Waterloo, ON, Canada; `http://www.maplesoft.com`.

[12] Mathematica, Wolfram Research, Inc., Champaign, IL, USA; `http://www.wolfram.com`.

[13] MATLAB, The MathWorks, Inc., Natick, MA, USA; `http://www.mathworks.com`.

[14] *Symbolic Math Toolbox Version 2: User's Guide* (The MathWorks, Natick, MA, USA) Online version.

[15] C.B. Moler and C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Rev. 20(4) (1978) 801–836.

[16] J.-M. Muller, *Elementary Functions: Algorithms and Implementation* (Birkhäuser, Boston, MA, 1997).

[17] K.C. Ng, Contributions to the computation of the matrix exponential, Ph.D. thesis, Technical Report PAM-212, Center for Pure and Applied Mathematics, University of California, Berkeley (1984).

[18] 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.

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

[20] R.C. Ward, Numerical computation of the matrix exponential with accuracy estimate, SIAM J. Numer. Anal. 14(4) (1977) 600–610.