Computing the matrix sine and cosine simultaneously with a reduced number of products

Muaz Seydaoğlu[a,c,*], Philipp Bader[b], Sergio Blanes[c], Fernando Casas[d]

[a] *Faculty of Art and Science, Department of Mathematics, 49100 Mus, Turkey.*
[b] *Departament de Matemàtiques, Universitat Jaume I, 12071 Castellón, Spain.*
[c] *Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, E-46022 Valencia, Spain.*
[d] *IMAC and Departament de Matemàtiques, Universitat Jaume I, 12071 Castellón, Spain.*

Abstract

A new procedure is presented for computing the matrix cosine and sine simultaneously by means of Taylor polynomial approximations. These are factorized so as to reduce the number of matrix products involved. Two versions are developed to be used in single and double precision arithmetic. The resulting algorithms are more efficient than schemes based on Padé approximations for a wide range of norm matrices.

Keywords: Matrix sine, Matrix cosine, Taylor series, Padé approximation, Matrix polynomials

1. Introduction

Many dynamical systems are modeled by differential equations in which finding closed solutions is not possible and so one has to compute approximating solutions. These differential equations usually preserve some underlying geometric structure which reflects the qualitative nature of the phenomena they describe. It is then relevant that the approximations share with the exact solution of the differential equation these qualitative properties to render a right description. The design and analysis of numerical integrators preserving some of these geometric structures constitutes the realm of Geometric Numerical Integration, an active and interdisciplinary research area and the subject of intensive development during the last decades [6, 10, 15, 18, 20, 23].

Exponential integrators can be considered as a class of geometric integrators tailored to stiff and oscillatory equations [7, 8, 13, 14, 16]. For large systems of equations these schemes usually require to compute the action of the exponential of a matrix on a vector [13, 14]. However, for problems of moderate size it may be more appropriate to compute directly the exponential of the matrices involved.

When the problem is oscillatory, very often the formal solution involves both the sine and cosine of a matrix. Thus, for example, consider the Schrödinger equation in quantum mechanics,

$$i\frac{d\psi}{dt} = \mathcal{H}(t)\psi, \qquad \psi(t_0) = \psi_0,$$

where $\mathcal{H}(t)$ is a Hermitian operator and ψ is a complex wave function. A usual procedure to get numerical approximations involves first a spatial discretisation or

*Corresponding author

Email addresses: m.seydaoglu@alparslan.edu.tr (Muaz Seydaoğlu), bader@uji.es (Philipp Bader), serblaza@imm.upv.es (Sergio Blanes), casas@uji.es (Fernando Casas)

working on a finite dimensional representation. In any event, one ends up with a matrix equation with a similar structure,

$$i\frac{du}{dt} = Au, \qquad u(t_0) = u_0 \in \mathbb{C}^N.$$

If A is a real and constant matrix, the unitary evolution operator is given by

$$U(t) = \mathrm{e}^{-itA} = \cos(tA) - i\sin(tA). \tag{1}$$

There are different techniques to compute efficiently the exponential of a matrix [2, 3, 5, 12, 21, 25–27]. However, using any of these general algorithms to approximate the unitary matrix e^{-itA} in (1) involves products of complex matrices making them computationally expensive. Alternatively, we propose an efficient procedure to compute the matrix sine and cosine that only involves a small number of products of real matrices. The algorithm is used in combination with squaring as

$$\cos(2A) = 2\cos^2(A) - I = I - 2\sin^2(A), \qquad \sin(2A) = 2\sin(A)\cos(A).$$

In this way, it only requires two products per squaring (instead of four products when considering the square of complex matrices), thus making the overall procedure more efficient.

There are other examples where the computation of the sine and cosine of a matrix can be of interest. For example, for wave equations given by the generic second order system

$$y'' + Ay = f(y,t),$$

with $y \in \mathbb{R}^N$, exponential integrators frequently require to solve separately the linear homogeneous problem

$$y'' + Ay = 0, \qquad y(0) = y_0, \quad y'(0) = y_0'. \tag{2}$$

Writing (2) as a first order system, the solution is given by

$$\begin{pmatrix} y(t) \\ y'(t) \end{pmatrix} = \mathrm{e}^{tM} \begin{pmatrix} y_0 \\ y_0' \end{pmatrix}, \qquad \text{with} \quad M = \begin{pmatrix} 0 & I \\ -A & 0 \end{pmatrix} \tag{3}$$

and

$$\mathrm{e}^{tM} = \begin{pmatrix} \cos(t\sqrt{A}) & (\sqrt{A})^{-1}\sin(t\sqrt{A}) \\ -\sqrt{A}\sin(t\sqrt{A}) & \cos(t\sqrt{A}) \end{pmatrix} \equiv \begin{pmatrix} c(t^2A) & s(t,A) \\ -As(t,A) & c(t^2A) \end{pmatrix}. \tag{4}$$

Notice that the dimension of M is twice the dimension of A and so the cost of matrix-matrix multiplications grows, in general, by a factor of eight.

On the other hand, a closer look to the functions to be approximated clearly indicates that the algorithm used to evaluate the matrix sine and cosine for the unitary matrix (1) should not be used directly for this problem, since it requires computing first the square root of the matrix, $B = \sqrt{A}$, in addition to a multiplication and an inversion of this matrix. As a matter of fact, an efficient approximation to the exponential (4) was already presented in [4]. We propose in this case an improved algorithm based on a modification of the methods to compute the matrix sine and cosine with the goal of computing simultaneously the functions

$$c(t^2A) \equiv \cos(\sqrt{t^2A}) \qquad \text{and} \qquad s(t,A) \equiv (\sqrt{A})^{-1}\sin(\sqrt{t^2A}).$$

For the double angle we will take into account that

$$c(4t^2A) = 2c^2(t^2A) - I, \qquad s(2t,A) = 2s(t,A)c(t^2A),$$

thus requiring only two products per squaring. Notice that we do not use the property $\cos(2A) = I - 2\sin^2(A)$, since the function $\sin(A)$ is not computed in this case.

In summary, the purpose of this paper consists in developing algorithms that allow one to compute $\cos(A)$ and $\sin(A)$ or $c(t^2A)$ and $s(t, A)$ simultaneously and providing full accuracy up to single or double precision with a reduced computational cost. Thus, in particular, we propose an algorithm that, with only four products, reproduces exactly the series defining $\cos(A)$ up to the power A^{16} and the series of $\sin(A)$ up to A^{17}. In this sense, the following definition will be helpful for the rest of the paper:

Definition 1. *We say that a given function $f(A)$ satisfies $f(A) = \mathcal{O}(A^n)$ if it can be written as a convergent Taylor expansion, $f(A) = \sum_{k=n}^{\infty} c_k A^k$, for $\|A\| < \alpha$, with α a positive constant.*

Thus, our algorithm allows one to write $\cos(A)$ with and error $\mathcal{O}(A^{18})$. The same procedure allows one, with one extra product (seven products in total), to approximate $\cos(A)$ and $\sin(A)$ with errors $\mathcal{O}(A^{26})$ and $\mathcal{O}(A^{25})$, respectively.

Although one can find in the literature several algorithms to compute $\cos(A)$ (see [25] and references therein), only few of them are designed to do so in a simultaneous way (see [1] and references therein). As our analysis shows and several numerical examples confirm, the technique we propose here outperform all of them.

2. The algorithms

The search of fast algorithms for evaluating matrix polynomials has received considerable interest in the literature [2, 3, 17, 19, 22, 24, 28, 29]. We next briefly summarize how to approximate the matrix sine and cosine functions by means of certain polynomials involving a reduced number of matrix products. This reduction essentially follows the same approach used in [9] to minimise the number of commutators appearing in different Lie-group integrators and was successfully adapted to the Taylor expansion of the exponential matrix in [2] and especially in [3].

Generally speaking, the strategy consists first in elaborating a recursive procedure to compute the polynomial approximating the matrix cosine with the minimum number of products and then using these same products to approximate the matrix sine as accurately as possible in the cheapest possible way.

Clearly, the most economic way to construct polynomials of degree 2^k is by applying the following sequence, which requires the evaluation of only k products. First we form the intermediate matrices

$$\begin{aligned}
A_0 &:= I, \quad A_1 := A, \\
A_2 &:= z_{2,0}I + z_{2,1}A_1 + (x_1 I + x_2 A_1)(x_3 I + x_4 A_1), \\
A_4 &:= z_{4,0}I + z_{4,1}A_1 + z_{4,2}A_2 + (x_5 I + x_6 A_1 + x_7 A_2)(x_8 I + x_9 A_1 + x_{10} A_2), \\
A_8 &:= \sum_{k=0}^{3} z_{8,2^{k-1}} A_{2^{k-1}} + (x_{11} I + \cdots + x_{14} A_4)(x_{15} I + \cdots + x_{18} A_4), \qquad (5) \\
&\ \vdots
\end{aligned}$$

and finally we take

$$P_{2^k} = A_{2^k},$$

where the coefficients $z_{i,j}, x_k \in \mathbb{R}$ have to be chosen in such a way that P_{2^k} coincides with the desired polynomial. Here the indices 2^k are chosen to indicate the highest attainable power, i.e., $A_{2^k} \in \mathbb{P}_{2^k}(A)$, where $\mathbb{P}_n(x)$ is the set of polynomials of degree

$\leq n$ in the variable x. Of course, there are many redundancies in the coefficients since some of them can be absorbed by others. Thus, for instance, we may take just $A_2 = A_1A_1$, since any quadratic polynomial can be written in terms of A_0, A_1 and A_2. By the same token, we can take $A_4 = A_2(x_1A_1+x_2A_2)$ because any polynomial of degree four can be expressed in terms of A_0, A_1, A_2 and A_4 for an appropriate choice of the coefficients x_1 and x_2. However, the problem is much more involved for higher degree polynomials.

According to the previous considerations, any polynomial of degree up to four can be computed with two products, whereas polynomials up to degree eight can be computed with only three products. This does not mean, however, that all such polynomials can be written with just three products. Thus, in particular, $P_7(A) = A^7$ requires at least four products where one of the intermediate results must be a polynomial of odd order (e.g. $A_2 = A^2$, $A_3 = AA_2$, $A_4 = A_2A_2$, $A_7 = A_4A_3 = P_7(A)$).

When a given polynomial cannot be reproduced by following the previous approach, new terms have to be incorporated. Thus, in particular

$$\begin{aligned} &A_0 = I, \quad A_1 = A, \quad A_2 = A^2, \quad A_3 = AA_2 \\ &A_6 = B_{3,1} + B_{3,2}B_{3,3}, \qquad B_{3,i} = \sum_{k=0}^{3} x_{i,k}A_k \\ &\quad\vdots \end{aligned} \tag{6}$$

and this generalises the procedure.

We use this technique in the sequel to approximate first $\cos(A)$ and $\sin(A)$ simultaneously with the minimum number of products, and then we apply the same procedure to $c(t^2A)$ and $s(t, A)$.

2.1. *Computing* $\cos(A)$ *and* $\sin(A)$ *simultaneously*

Let us denote by

$$T_{2m}^c(A) = \sum_{k=0}^{m} \frac{(-1)^k (A^2)^k}{(2k)!}, \qquad T_{2m+1}^s(A) = A\sum_{k=0}^{m} \frac{(-1)^k (A^2)^k}{(2k+1)!}$$

the Taylor polynomial approximations of $\cos(A)$ and $\sin(A)$ up to order $2m$ and $2m+1$ in A, respectively, and by $T_{2m+1,\ell}^s$ with $\ell > 2m+1$, any polynomial of degree ℓ such that

$$T_{2m+1,\ell}^s(A) = T_{2m+1}^s(A) + \mathcal{O}(A^{2m+2}).$$

We next collect the best approximations we have obtained to the cosine and sine functions by polynomial functions when using an increasing number of products k.

$k = 3$ **products.** This constitutes a trivial problem, but it nevertheless illustrates the general procedure. With two products we can compute T_4^c as

$$\begin{aligned} &A_2 = A^2, \qquad A_4 = A^4, \\ &T_4^c(A) = I - \frac{1}{2!}A_2 + \frac{1}{4!}A_4, \end{aligned} \tag{7}$$

and with one extra product we get

$$T_5^s(A) = A(I - \frac{1}{3!}A_2 + \frac{1}{5!}A_4). \tag{8}$$

$k=4$ **products.** With three products we can compute T_8^c as

$$\begin{aligned} A_2 &= A^2, \qquad A_4 = A_2^2, \qquad A_8 = A_4\left(-\frac{1}{6!}A_2 + \frac{1}{8!}A_4\right), \\ T_8^c(A) &= I - \frac{1}{2!}A_2 + \frac{1}{4!}A_4 + A_8. \end{aligned} \tag{9}$$

With one extra product we can approximate the matrix sine, but only up to order seven:

$$T_{7,9}^s(A) = A\left(I - \frac{1}{3!}A_2 + \frac{1}{5!}A_4 + \frac{6!}{7!}A_8\right). \tag{10}$$

According with the previous notation, $T_{7,9}^s(A) = T_7^s(A) + \mathcal{O}(A^9)$.

The order of approximation of the matrix sine can be increased by incorporating one extra product as follows:

$$\begin{aligned} A_8 &= A_4\left(-\frac{1}{7!}A_2 + \frac{1}{9!}A_4\right), \\ T_9^s(A) &= A\left(I - \frac{1}{3!}A_2 + \frac{1}{5!}A_4 + A_8\right). \end{aligned} \tag{11}$$

In this way, $\sin(A) - T_9^s(A) = \mathcal{O}(A^{11})$.

$k=6$ **products.** The following scheme allows one to express $T_{16}^c(A)$ with only four products:

$$\begin{aligned} A_2 &= A^2, \qquad A_4 = A_2^2, \qquad A_8 = A_4(x_1A_2 + x_2A_4), \\ A_{16} &= (x_3A_4 + A_8)(x_4I + x_5A_2 + x_6A_4 + x_7A_8), \\ T_{16}^c(A) &= I - \frac{1}{2}A_2 + x_8A_4 + A_{16}. \end{aligned} \tag{12}$$

In fact, we get two families of solutions depending on a free parameter that reproduce $T_{16}^c(A)$ in exact arithmetic. Since floating-point arithmetic has to be used in practice, we choose this free parameter x_1 so as to minimize the 1-norm of the vector $(x_1,\ldots,x_8)$ in order to avoid large round off errors. This results in

$$\begin{aligned} x_1 &= \frac{7}{500}, \qquad x_2 = -\frac{7}{60000}, \qquad x_3 = \frac{1}{2500}(-1533 + 7\sqrt{36681}), \\ x_4 &= -\frac{5(124581 + 391\sqrt{36681})}{10594584}, \qquad x_5 = \frac{9775}{10594584}, \qquad x_6 = -\frac{5(1001 + \sqrt{36681})}{508540032}, \\ x_7 &= \frac{3125}{889945056}, \qquad x_8 = \frac{1549211 + 3246\sqrt{36681}}{63063000}. \end{aligned} \tag{13}$$

Some of the coefficients are irrational numbers because they correspond to solutions of a nonlinear system of equations.

With two extra products we can approximate the matrix sine up to order $\mathcal{O}(A^{19})$ as follows:

$$\begin{aligned} C_{24} &= (z_5I + z_5A_2 + z_6A_4 + z_7A_8 + z_8T_{16}^c(A))A_8, \\ T_{17,25}^s(A) &= A\left(z_0I + z_1A_2 + z_2A_4 + z_3A_8 + z_4T_{16}^c(A) + C_{24}\right) \end{aligned} \tag{14}$$

with

$$
\begin{aligned}
z_0 &= \frac{8887}{4794}, & z_1 &= -\frac{1897}{3196}, & z_2 &= \frac{25259}{575280}, \\
z_3 &= -\frac{965093875}{9674368704}, & z_4 &= -\frac{4093}{4794}, & z_5 &= \frac{25698275}{29023106112}, \\
z_6 &= -\frac{3907675}{348277273344}, & z_7 &= \frac{11865625}{3656911370112}, & z_8 &= \frac{25}{308756448},
\end{aligned} \tag{15}
$$

In other words, (14)-(15) approximates the matrix sine up to a higher order than the matrix cosine.

$k = 7$ **products.** With five products we can compute T_{24}^c:

$$
\begin{aligned}
& A_2 = A^2, \qquad A_4 = A_2^2, \qquad A_6 = A_4 A_2, \\
& C_1 = a_{0,1} I + a_{1,1} A_2 + a_{2,1} A_4 + a_{3,1} A_6, \quad C_2 = a_{0,2} I + a_{1,2} A_2 + a_{2,2} A_4 + a_{3,2} A_6, \\
& C_3 = a_{0,3} I + a_{1,3} A_2 + a_{2,3} A_4 + a_{3,3} A_6, \quad C_4 = a_{0,4} I + a_{1,4} A_2 + a_{2,4} A_4 + a_{3,4} A_6, \\
& A_{12} = C_3 + C_4^2, \qquad A_{24} = (C_2 + A_{12}) A_{12} \\
& T_{24}^c(A) = C_1 + A_{24}.
\end{aligned} \tag{16}
$$

The best solution we have obtained is:

$$
\begin{aligned}
a_{0,1} &= 0, & a_{1,1} &= 0, \\
a_{2,1} &= 0.02264979811206039519, & a_{3,1} &= -0.00013110924142135755, \\
a_{0,2} &= 0.55751443809990408029, & a_{1,2} &= -0.61577924683458386455, \\
a_{2,2} &= 0.00747198841446687051, & a_{3,2} &= -0.00003362444420476012, \\
a_{0,3} &= 0.75936877868464999248, & a_{1,3} &= -0.01560333979813817129, \\
a_{2,3} &= 0.00010936989591908396, & a_{3,3} &= -1.03893360877457159499 \cdot 10^{-6}, \\
a_{0,4} &= 0, & a_{1,4} &= -0.03964996874347447309 1, \\
a_{2,4} &= 0.00015549007350382146 3, & a_{3,4} &= -1.12673966307117002248 8 \cdot 10^{-6}.
\end{aligned} \tag{17}
$$

Although we report here 20 digits for the coefficients, they can be in fact determined with arbitrary accuracy.

With two extra products we can approximate the matrix sine up to order $\mathcal{O}(A^{23})$ as follows:

$$
\begin{aligned}
& C_{48} = (z_6 I + z_7 A_2 + z_8 A_4 + z_9 A_6 + z_{10} A_{12} + z_{11} T_{24}^c(A)) T_{24}^c(A), \\
& T_{23,49}^s(B) = A \left(z_0 I + z_1 A_2 + z_2 A_4 + z_3 A_6 + z_4 A_{12} + z_5 T_{24}^c(A) + C_{48} \right),
\end{aligned} \tag{18}
$$

with

$$
\begin{aligned}
z_0 &= 0.10090808375109885598, & z_1 &= -0.07668753546445299316, \\
z_2 &= 0.00084924846993243257, & z_3 &= -0.00001220406904464391, \\
z_4 &= 0.98499703159318860027, & z_5 &= -0.84925233648155398756, \\
z_6 &= 1, & z_7 &= 0.00095544138280925799, \\
z_8 &= 4.56337109377154270633 \cdot 10^{-6}, & z_9 &= 2.73461259403000427141 \cdot 10^{-8}, \\
z_{10} &= 0.00048550288474842477 & z_{11} &= -4.15891109384923342531 \cdot 10^{-7}.
\end{aligned} \tag{19}
$$

2.2. Computing $c(t^2 A)$ and $s(t, A)$ simultaneously

Given a real matrix A, let us denote by

$$
P_m^c(t^2 A) = \sum_{k=0}^{m} \frac{(-1)^k (t^2 A)^k}{(2k)!}, \qquad P_m^s(t, A) = t \sum_{k=0}^{m} \frac{(-1)^k (t^2 A)^k}{(2k+1)!}
$$

the Taylor expansions of the functions

$$c(t^2A) = \cos(\sqrt{t^2A}), \qquad \text{and} \qquad s(t,A) = (\sqrt{A})^{-1}\sin(\sqrt{t^2A})$$

up to order m in A, respectively. Notice that they are approximations up to order $2m$ and $2m+1$ in t to the respective functions. Analogously, we will denote by $P^s_{m,\ell}$, $\ell > m$, any polynomial of degree ℓ such that $P^s_{m,\ell}(A) = P^s_m(A) + \mathcal{O}(A^{m+1})$.

Next we show how the previous algorithms to approximate the sine and cosine functions can be adjusted to approximate $c(t^2A)$ and $s(t,A)$. As before, we proceed according with the number of products involved.

$k=3$ **products.** With two products we can compute $P^c_4(t^2A)$:

$$\begin{aligned} &B = t^2A, \qquad B_2 = B^2, \qquad B_4 = B^2(-\frac{1}{6!}B + \frac{1}{8!}B_2), \\ &P^c_4(B) = I - \frac{1}{2!}B + \frac{1}{4!}B_2 + B_4. \end{aligned} \tag{20}$$

With the same number of products we can also evaluate $P^s_{3,4}(t,A)$,

$$P^s_{3,4}(t,A) = t\left(I - \frac{1}{3!}B + \frac{1}{5!}B_2 - \frac{6!}{7!}B_4\right), \tag{21}$$

whereas with one extra product we get

$$P^s_4(t,A) = t\left(I - \frac{1}{3!}B + \frac{1}{5!}B_2 + B_2\left(-\frac{1}{7!}B + \frac{1}{9!}B_2\right)\right). \tag{22}$$

$k=4$ **products.** With three products we can compute $P^c_8(t^2A)$:

$$\begin{aligned} &B = t^2A, \qquad B_2 = B^2, \qquad B_4 = B_2(x_1B + x_2B_2), \\ &B_8 = (x_3B_2 + B_4)(x_4I + x_5B + x_6B_2 + x_7B_4), \\ &P^c_8(B) = y_0I + y_1B + y_2B_2 + B_8, \end{aligned} \tag{23}$$

whose coefficients are the same as those given in (13).

With one extra product we can approximate the matrix sine up to order eight as

$$\begin{aligned} &C_{12} = (z_5I + z_5B + z_6B_2 + z_7B_4 + z_8P^c_8(B))B_4, \\ &P^s_{8,12}(t,A) = t\left(z_0I + z_1B + z_2B_2 + z_3B_4 + z_4P^c_8(B) + C_{12}\right), \end{aligned} \tag{24}$$

with the same values for the coefficients z_i as before.

$k=5$ **products.** With four products we can compute $P^c_{12}(t^2A)$:

$$\begin{aligned} &B = t^2A, \qquad B_2 = B^2, \qquad B_3 = B_2B, \\ &D_1 = a_{0,1}I + a_{1,1}B + a_{2,1}B_2 + a_{3,1}B_3, \quad D_2 = a_{0,2}I + a_{1,2}B + a_{2,2}B_2 + a_{3,2}B_3, \\ &D_3 = a_{0,3}I + a_{1,3}B + a_{2,3}B_2 + a_{3,3}B_3, \quad D_4 = a_{0,4}I + a_{1,4}B + a_{2,4}B_2 + a_{3,4}B_3, \\ &B_6 = D_3 + D_4^2 \\ &P^c_{12}(B) = D_1 + (D_2 + B_6)B_6, \end{aligned} \tag{25}$$

with solution for the coefficients $a_{i,j}$ given in (17), whereas with one extra product we can approximate $P^s_{11}(t,A)$ as

$$\begin{aligned} &C_{24} = (z_6I + z_7B + z_8B_2 + z_9B_3 + z_{10}B_6 + z_{11}P^c_{12}(A))P^c_{12}(B), \\ &P^s_{11,24}(t,A) = t\left(z_0I + z_1B + z_2B_2 + z_3B_3 + z_4B_6 + z_5P^c_{12}(B) + C_{24}\right), \end{aligned} \tag{26}$$

with the same coefficients as in (19).

2.3. *Padé approximations*

At this point it is useful to briefly review the schemes presented in [1] to compute the matrix sine and cosine simultaneously, since they will be compared in section 4 with our own procedure.

The methods presented in [1] are based on the identities

$$\cos(A) = \frac{\mathrm{e}^{iA} + \mathrm{e}^{-iA}}{2}, \qquad \sin(A) = \frac{\mathrm{e}^{iA} - \mathrm{e}^{-iA}}{2i},$$

and the use of Padé approximations of the exponential e^{iA}. For instance, taking a diagonal Padé of order eight for approximating e^{iA}, i.e. $r_4(iA) = [p_4(-iA)]^{-1}p_4(iA) = \mathrm{e}^{iA} + \mathcal{O}(A^9)$ one gets

$$\begin{aligned} s_4 &= \frac{A\left(I - \frac{11}{8}A^2 + \frac{37}{1176}A^4 - \frac{1}{70560}A^6\right)}{I + \frac{1}{28}A^2 + \frac{3}{3920}A^4 + \frac{1}{8}A^6 + \frac{1}{2822400}A^8}, \\ c_4 &= \frac{I - \frac{13}{28}A^2 + \frac{289}{11760}A^4 - \frac{19}{70560}A^6 + \frac{19}{2822400}A^8}{I + \frac{1}{28}A^2 + \frac{3}{3920}A^4 + \frac{1}{8}A^6 + \frac{1}{2822400}A^8}, \end{aligned}$$

with

$$s_4 = \sin(A) + \mathcal{O}(A^9), \qquad c_4 = \cos(A) + \mathcal{O}(A^{10}).$$

It is clear that s_4, c_4 can be computed simultaneously with 5 products (A^2, A^4, A^6, A^8, and the extra product for the numerator in s_4) and the computation of two inverse matrices. Since both denominators are the same, only one LU factorization is necessary. The totals cost is $(7 + \frac{1}{3})$ products. Notice that the same order (with very similar accuracy as we will see) is obtained with our novel approach at the cost of only 4 products (and a smaller number of matrices need to be stored).

3. Error analysis

Next we analyse how to bound the truncation errors of the previously considered Taylor polynomial approximations of order $2m$ and $2\tilde{m}+1$ for the cosine and sine functions, respectively. They have the form

$$\begin{aligned} \cos(A) - T_{2m}^c &= \sum_{k=m+1}^{\infty} \alpha_{2k} A^{2k}, \qquad 2m \in \{4, 8, 16, 24\} \\ \sin(A) - T_{2\tilde{m}+1,\ell}^s &= \sum_{k=\tilde{m}+1}^{\infty} \tilde{\alpha}_{2k+1} A^{2k+1}, \qquad 2\tilde{m}+1 \in \{5, 7, 17, 23\}. \end{aligned} \tag{27}$$

On the other hand, the truncation errors of the approximations of the cosine and sine functions obtained by using Padé approximants for e^{iA} [1] can be written as

$$\cos(A) - c_m = \sum_{k=m+1}^{\infty} \gamma_{2k} A^{2k}, \qquad \sin(A) - s_m = \sum_{k=m}^{\infty} \hat{\gamma}_{2k+1} A^{2k+1}. \tag{28}$$

Clearly, the series (27) and (28) can be bounded in terms of $\|A\|$ as

$$\|\cos(A) - T_{2m}^c\| \le \sum_{k=m+1}^{\infty} |\alpha_{2k}| \, \|A\|^{2k}, \qquad \left\|\sin(A) - T_{2\tilde{m}+1,\ell}^s\right\| \le \sum_{k=\tilde{m}+1}^{\infty} |\tilde{\alpha}_{2k+1}| \, \|A\|^{2k+1}, \tag{29}$$

and

$$\|\cos(A) - c_m\| \leq \sum_{k=m+1}^{\infty} |\gamma_{2k}| \, \|A\|^{2k}, \qquad \|\sin(A) - s_m\| \leq \sum_{k=m}^{\infty} |\hat{\gamma}_{2k+1}| \, \|A\|^{2k+1}. \tag{30}$$

We denote by θ_{2m}^{M} the largest value of $\|A\|$ such that the bounds (29), (30) do not exceed a prescribed accuracy, u, for each method $M \equiv T_{2m}^{c}, T_{2\tilde{m}+1}^{s}, c_m, s_m$. To achieve maximum accuracy, we bound the previous forward absolute errors with the unit round off $u = 2^{-53}$, $u = 2^{-24}$ in double and single precision floating-point arithmetic, respectively. We have truncated the series of the corresponding functions after 150 terms to find θ_{2m}^{M}. The corresponding values for the new Taylor approximations of the cosine and sine functions are collected in Tables 1 and 2, together with the total number of matrix products corresponding to each procedure, Π_{2m}. For completeness, we have also included the values of θ_{2m}^{M} for the Padé approximations, as given in [1]. In this case we have also added to the total number of matrix products π_m the cost of evaluating two inverse matrices sharing the same LU factorization, i.e $(2 + \frac{1}{3})$ products.

The comparison of the theoretical performance of the new Taylor polynomial approximations $T_{2m}^{c}, T_{2\tilde{m}+1}^{s}$ (with orders $\{4, 8, 16, 24\}$ and $\{5, 7, 17, 23\}$ respectively) and the Padé approximations c_m, s_m [1] (with orders $\{4, 8, 16, 24\}$) has been illustrated in Figure 1: here we plot $\|A\|$ versus the number of matrix products required for each approximation of $\cos(A)$ and $\sin(A)$ simultaneously, both in double (left) and single (right) precision. From the figure the improvement achieved by the proposed Taylor polynomial approximations is apparent.

Table 1: Number of matrix multiplications Π_{2m} and forward absolute error bounds θ_{2m}^{M} in double precision floating-point arithmetic, $u \leq 2^{-53}$, for the new Taylor algorithms T_{2m}^{c}, $T_{2\tilde{m}+1}^{s}$ and Padé approximations c_m, s_m [1]. The cost of the computation of two inverse matrices sharing the same LU factorization, i.e $(2 + \frac{1}{3})$, has been included in the cost π_m for the Padé approximations.

$2\tilde{m}$	4	6	16	22
$2m$	4	8	16	24
$\theta_{2m}^{c_m}$	6.5633e-3	1.3959e-1	1.3879	3.7288
$\theta_{2m}^{s_m}$	2.4019e-3	1.1213e-1	1.3784	3.7287
π_m [1]	$\mathbf{5 + \frac{1}{3}}$	$\mathbf{7 + \frac{1}{3}}$	$\mathbf{10 + \frac{1}{3}}$	$\mathbf{12 + \frac{1}{3}}$
$\theta_{2m}^{T_{2m}^{c}}$	6.5633e-3	1.1495e-1	9.8108e-1	2.5675
$\theta_{2m}^{T_{2\tilde{m}+1}^{s}}$	1.777e-2	8.0438e-2	1.1184	1.97
Π_{2m}	**3**	**4**	**6**	**7**

Table 2: Same as Table 3, but now in single precision floating-point arithmetic.

$2\tilde{m}$	4	6	16	22
$2m$	4	8	16	24
$\theta_{2m}^{c_m}$	1.8687e-1	1.0218	3.8571	7.1575
$\theta_{2m}^{s_m}$	1.3355e-1	9.9511e-1	3.8569	7.1575
π_m [1]	$\mathbf{5 + \frac{1}{3}}$	$\mathbf{7 + \frac{1}{3}}$	$\mathbf{10 + \frac{1}{3}}$	$\mathbf{12 + \frac{1}{3}}$
$\theta_{2m}^{T_{2m}^{c}}$	1.8709e-1	8.5756e-1	2.9935	5.5555
$\theta_{2m}^{T_{2\tilde{m}+1}^{s}}$	3.1386e-1	7.492e-1	3.2152	4.3819
Π_{2m}	**3**	**4**	**6**	**7**

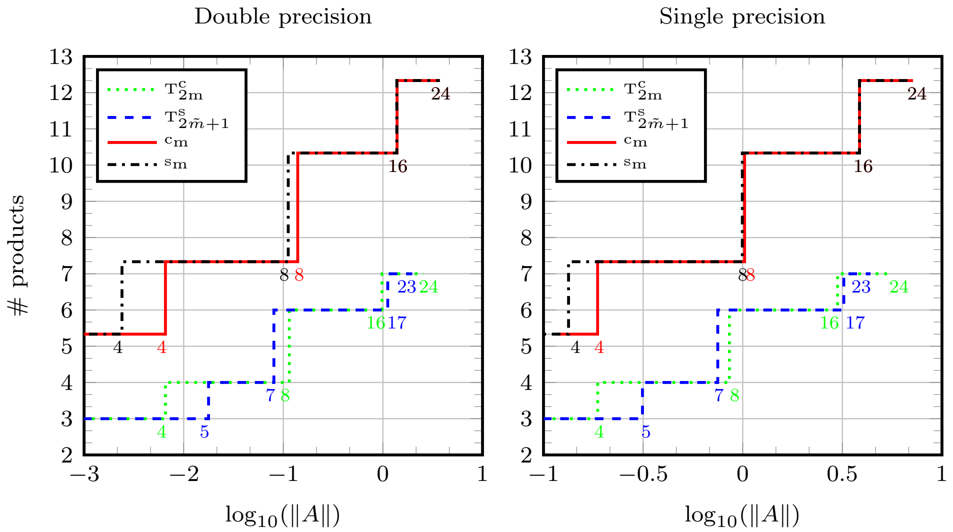

Figure 1: Orders and corresponding number of products of each method versus $\|A\|$ in double and single precision floating-point arithmetic.

Remark 2. *Notice that the previous error analysis is also valid for the approximations to the function $c(t^2A)$ just by replacing $\|A\|$ by $t\sqrt{\|A\|}$ in the first expression of eq. (29). Since this result depends on both t and $\|A\|$, it is not appropriate to get error bounds simultaneously for both $s(t,A)$ and $As(t,A)$ of eq. (4). Nevertheless, in the frequent case in which A is a real and symmetric matrix, then e^{tM} in (4) is a symplectic matrix and $\|\sin(t\sqrt{A})\| \le 1$. In this case the values of $\tilde{\alpha}_{2k+1}$ appearing in the second expression of (29) are the same when considering the relative error for both functions $s(t,A)$ and $As(t,A)$.*

4. Numerical experiments

We measure the performance of new Taylor polynomials (denoted as 'cosmsinmT') and the Padé approximations (denoted as 'cosmsinmP') [1] to compute the matrix cosine and sine functions simultaneously. The platform of all numerical experiments is MATLAB R2013a and the matrix 1-norm has been used in implementing the algorithms. The experiments have been carried out for 2500 matrices of dimension $\le 16\times16$, $\le 64\times64$, $\le 1024\times1024$ (adjusted in order to have different norms) in Figs. 2, 3, Fig. 4 and Fig. 5, respectively, of the following cases: EXPLAI THE SIZE OF THESE 2500 MATRICES. iN FIGS. 2 AND 3 THEY ARE OF DIMENSION $\le 16\times16$, IN FIG. 4 THEY ARE OF DIMENSION $\le 64\times64$ AND IN FIG. 5 THEY ARE OF DIMENSION $\le 1024\times1024$.

- 52 test matrices have been chosen from the MATLAB gallery function [11] (blue). 690 sampled matrices with different norms were tested. If the gallery does not return valid results for some dimension then the dimension is reduced to 64×64. Furthermore, the dimension is reduced to 4×4 and 8×8 for matrices with norm larger than 10^4. TO EXPLAIN HOW THE 690 MATRICES ARE CHOSEN AND THEIR SIZES

- Using `rand()` and `randn()` functions in MATLAB to randomly generate matrices with entries drawn from different distributions. 400 matrices normally distributed, 500 matrices uniformly distributed in the interval $(0,1)$ and 501

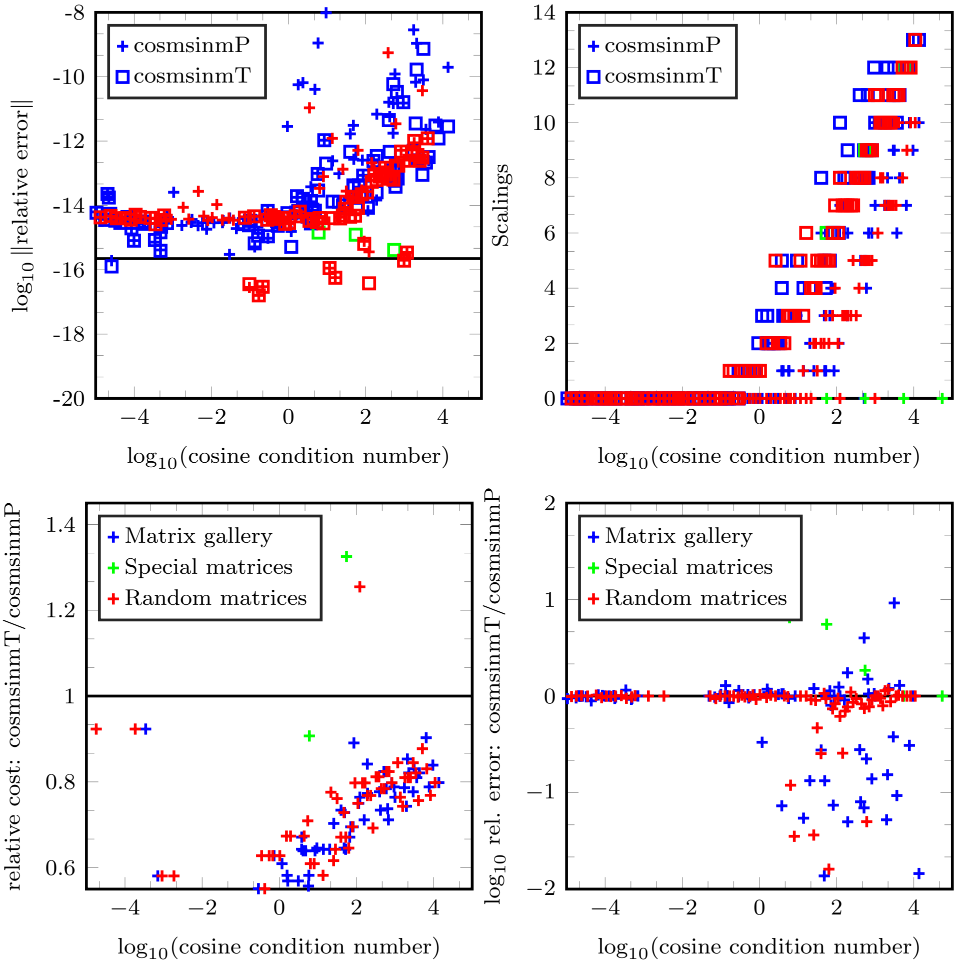

Figure 2: Comparison of cosmsinmP and cosmsinmT by using results of the cosine function of the matrices of dimensions $\leq 16 \times 16$. Test Matrices are specified as Matrix gallery, Random matrices and Special matrices with their corresponding colors blue, red and green. Top left panel: The relative errors of two methods. Top right panel: Utilization of scaling parameter. Bottom left panel: The ratio of the costs. Bottom right panel: The ratio of the relative errors is shown.

matrices in the interval $(-0.5, 0.5)$ (red). TO INDICATE WHICH COLOR CORRESPOND TO THESE MATRICES

- Using `spdiags()` and `rand()` functions in MATLAB to construct 400 triangular nilpotent matrices with random rank (red).
- 9 matrices of the form

$$A = \begin{pmatrix} 1 & \lambda \\ 0 & -1 \end{pmatrix}, \tag{31}$$

where $\lambda = 1, 10, \ldots, 10^8$ (green), possibly leading to *overscaling* (utilization of large value of scaling parameter s).

The same test matrices have been generated as in Remark 5 of [3] and all matrices are adjusted to have 1-norms over $(10^{-4}, 10^{4.1})$ in all numerical experiments. Notice that, given a matrix A, we first compute $\|A\|_1$ in order to decide, according to the results shown in Table 1 (for double precision) of Table 2 (for single precision), which

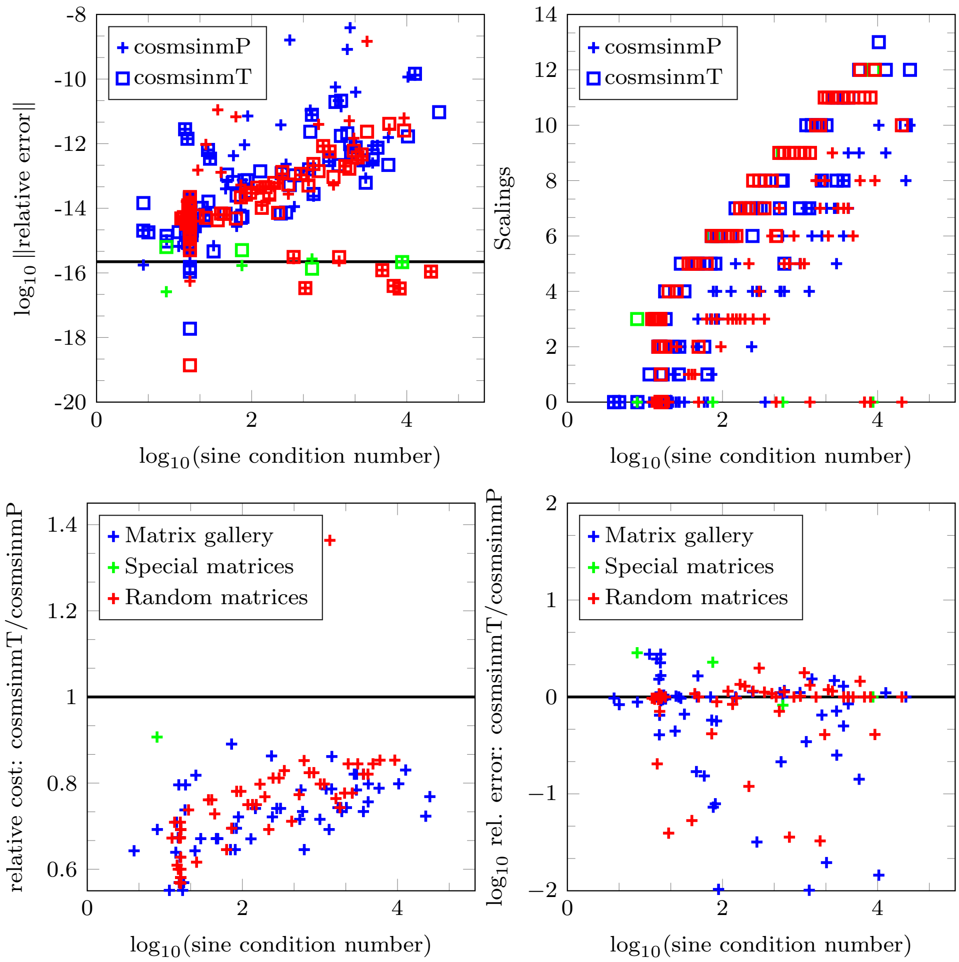

Figure 3: Comparison of cosmsinmP and cosmsinmT by using results of the sine function of the matrices of dimensions $\leq 16 \times 16$. Cf. Figure 2.

method is used to approximate the functions as well as if the scaling-squaring is required.

The condition numbers of each matrix function are computed by executing the function `funm_condest1` from the Matrix Function Toolbox [11]. The reference solutions of the matrix cosine and sine have been calculated with *Mathematica* with 100 digits of precision. We have computed the relative error

$$\frac{\|F - f(A)\|_2}{\|f(A)\|_2},$$

where F is an approximated value of $f(A)$. In the following we show the results for double precision (similar results are obtained for single precision). We have simulated the results for the 2500 matrices of dimension $\leq 16 \times 16$ in Figs. 2, 3. From the top left of the Figs. 2, 3, in general, the relative errors of both cosmsinmP and cosmsinmT methods produced in approximating the matrix cosine and sine functions change between $1.0e-12$ and $1.0e-15$ and they drop below the machine accuracy for few matrices. It can be observed from the top right of the Figs. 2, 3, the cosmsinmT method involves more scalings, particularly the cosmsinmP and cosmsinmT methods have leaded to the scaling for 741 and 1180 matrices respectively. Regarding to the bottom left of the Figs. 2, 3, the ratios of the cost

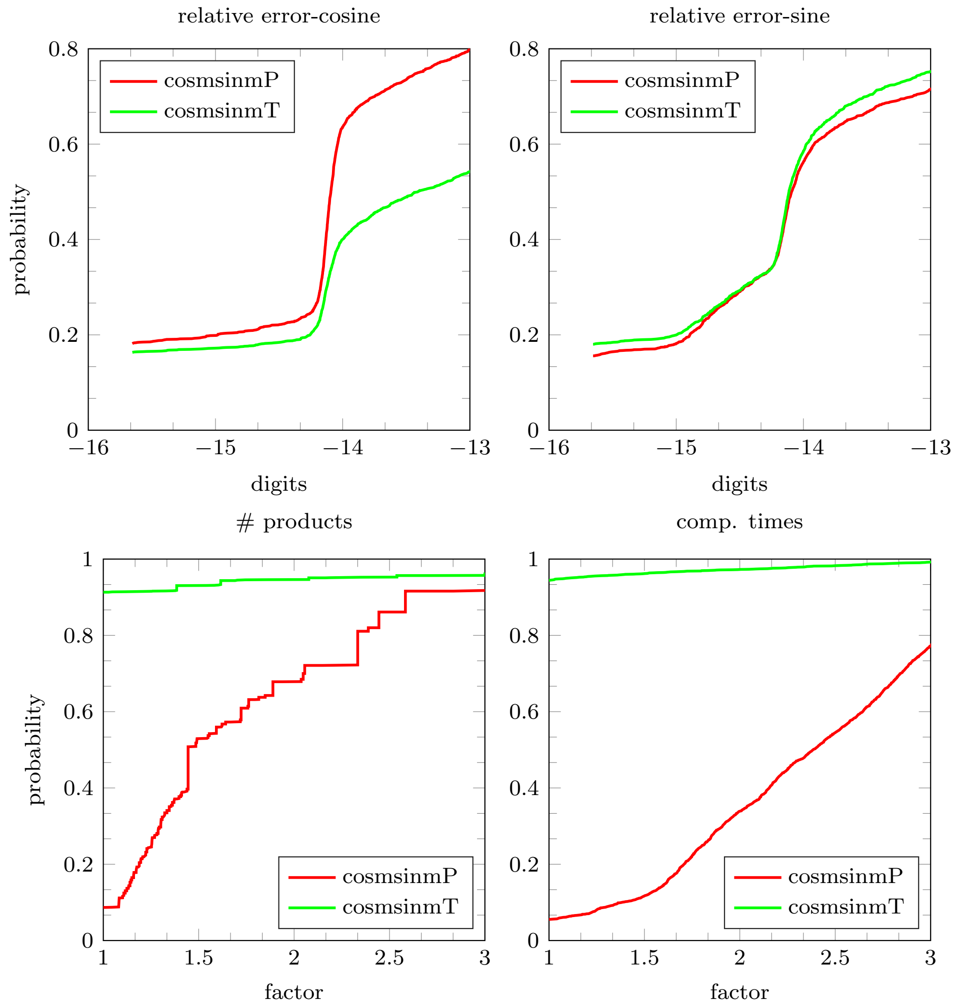

Figure 4: Performance profiles of cosmsinmP and cosmsinmT by using results of the cosine and sine function of the matrices of dimensions $\leq 64 \times 64$. The top panels: the percentage of matrices among the test set for which the relative logarithmic errors are lower than the horizontal axis. Bottom left panel: the percentage of matrices among the test set for which the method under consideration has a number of product lower than *factor*-times the smallest number of product across all algorithms. Bottom right panel: Restatement of bottom left panel in terms of the computation times averaged across 10 cycles.

cosmsinmT/cosmsinmP are in general below 1, it also has been concluded from Tables 1, 2 and Fig.1, the new method cosmsinmT requires less number of matrix products. As can be seen from the bottom right of the Figs. 2, 3, the accuracy of both methods is in good agreement with the theoretical results we have obtained. In these cases, some of the values of the relative errors have been replaced by machine accuracy (if these are lower) in the results of both methods. Furthermore, we plot performance profiles of the algorithms on a set of the test matrices exemplified for Figs. 2, 3 in terms of the relative errors, number of products and computational times.

The performance profile figure presents the probability that a measure is within a given factor of the best measure across all algorithms. The performance measures which are used for comparison in the figures are : relative errors, number of products and computational times. The probability is plotted versus the digits (roundoff

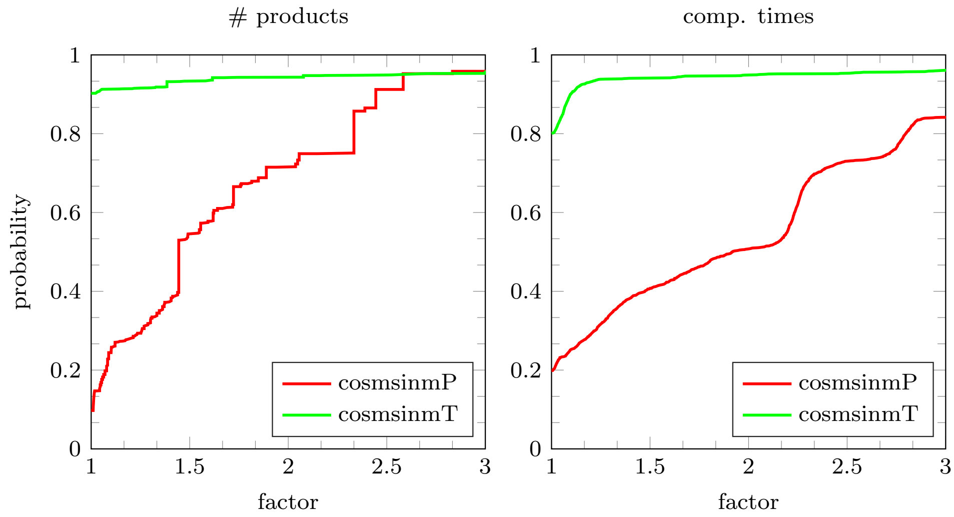

Figure 5: Performance profiles of cosmsinmP and cosmsinmT by using results of the cosine and sine function of the matrices of dimensions $\leq 1024 \times 1024$. Cf. Figure 4 bottom panels.

error) in the comparison of the relative errors on the top of the Fig. 4 and versus the factor on the interval $[1, 3]$ in the comparison of the number of products and computational times on the bottom of Fig. 4 and on the Fig. 5.

THE PERFORMANCE PLOTS NEED TO BE EXPLAINED TO READERS NOT FAMILIAR WITH THIS. TO EXPLAIN IN MORE DETAIL WHAT IS PLOTTED IN THESE FIGURES.

The performance plot shows the percentage of problems (y-axis) that are within a given factor (x-axis) of the best method [30]. In the experiments illustrated by the performance profiles in Fig. 4, the 2500 matrices of dimension $\leq 64 \times 64$ have been tested. We have observed that the cosmsinmT method has a lower relative error for 877 and 1257 of the 2500 matrices than the cosmsinmP method for computing the approximate values of the matrix cosine and sine functions respectively (358 and 348 results are equal). These results are evident from the Fig. 4 on the top. It is seen clearly from the bottom of Fig. 4 that the cosmsinmT method is less expensive than cosmsinmP.

The performance profiles in Fig. 5 resulted from demonstrating the returns from the 2500 matrices of dimension $\leq 1024 \times 1024$ confirm the superiority of the cosmsinmT method in the sense of computational cost.

TO EXPLAIN IN THE CAPTION THE FIGURES SO, IT WILL NOT BE NECESSARY TO LOOK FOR IT IN THE TEXT

5. Conclusions

We have presented a new algorithm to compute the matrix cosine and sine. The algorithm contains several methods that are optimised for different values of the norm of the matrix and the desired accuracy, and can be combined with the scaling and squaring technique. Each of these methods is obtained by following a sequence in which each stage uses the results from all previous ones. An error analysis is also carried out and we have shown both theoretically as well as in the numerical experiments that the new algorithm is superior to other procedures from the literature that are based on Padé approximations to the matrix cosine and sine.

The new algorithm only involves matrix-matrix products and does not require to compute the inverse of matrices as is the case for Padé approximations. The cost of computing the inverse of a dense matrix can be taken as 4/3 the cost of the product of two dense matrices. However, for sparse matrices, the computational cost of the proposed algorithms grow almost linearly, whereas the cost of Padé approximations grows much faster because, in general, the inverse of a sparse matrix is a dense matrix. We thus expect a further increase in the efficiency of the new algorithms in this case.

Acknowledgments

The work of MS has been funded by the Scientific and Technological Research Council of Turkey (TUBITAK) with Grand Number 1059B191802292. PB, SB and FC acknowledge financial support from Ministerio de Economía, Industria y Competitividad (Spain) through projects MTM2016-77660-P and PID2019-104927GB-C21 (AEI/FEDER, UE).

References

[1] A.H. Al-Mohy, N.J. Higham and S.D. Relton, New algorithms for computing the matrix sine and cosine separately or simultaneously, SIAM J. Sci. Compu. 37 (2015) A456 - A487.

[2] P. Bader, S. Blanes, and F. Casas, An improved algorithm to compute the exponential of a matrix, arXiv:1710.10989 [math.NA] (2017) preprint.

[3] P. Bader, S. Blanes, and F. Casas, Computing the matrix exponential with an optimized Taylor polynomial approximation, Mathematics 7 (2019) 1174 doi:10.3390/math7121174.

[4] P. Bader, S. Blanes, E. Ponsoda, and M Seydaoğlu, Symplectic integrators for the matrix Hill's equation and its applications to engineering models, J. Comput. Appl. Math. 316 (2017) 47 - 59.

[5] P. Bader, S. Blanes, and M. Seydaoğlu, The scaling, splitting and squaring method for the exponential of perturbed matrices, SIAM J. Matrix Anal. Appl. 36 (2015) 594 - 614.

[6] S. Blanes, and F. Casas, A Concise Introduction to Geometric Numerical Integration, CRC Press: Boca Raton, FL, USA, 2016.

[7] S. Blanes, F. Casas, and A. Murua, An efficient algorithm based on splitting for the time integration of the Schrödinger equation, J. Comput. Phys. 303 (2015) 396 - 412.

[8] S. Blanes, F. Casas, J.A. Oteo, and J. Ros, The Magnus expansion and some of its applications, Phys. Rep. 470 (2009) 151 - 238.

[9] S. Blanes, F. Casas and J. Ros, High order optimized geometric integrators for linear differential equations, BIT, 42 (2002) 262 - 284.

[10] E. Hairer, C. Lubich, and G. Wanner, Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations, 2nd Ed., Springer, Berlin, 2006.

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

[12] N.J. Higham, and A.H. Al-Mohy, Computing matrix functions, Acta Numerica 19 (2010) 159 - 208.

[13] M. Hochbruck and C. Lubich, On Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 34 (1997) 1911 – 1925.

[14] M. Hochbruck and A. Ostermann, Exponential integrators, Acta Numerica 19 (2010) 209 - 286.

[15] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, Cambridge University Press, 2nd ed., 2008

[16] A. Iserles, H.Z. Munthe-Kaas, S.P. Nørsett, and A. Zanna, Lie group methods, Acta Numerica 9 (2000)215 - 365.

[17] L. Lei, and T. Nakamura, A fast algorithm for evaluating the matrix polynomial $I+A+\cdots+A^{N-1}$, IEEE Trans. Circuits Sys.-I: Fund. Theory Appl. 39 (1992) 299 - 300.

[18] B. Leimkuhler and S. Reich, Simulating Hamiltonian Dynamics, Cambridge University Press, 2004.

[19] W. Liang, R. Baer, C. Saravanan, Y. Shao, A.T. Bell, M. Head-Gordon, Fast methods for resumming matrix polynomials and Chebyshev matrix polynomials, J. Comput. Phys. 194 (2004) 575 - 587.

[20] C. Lubich, From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis, European Mathematical Society, 2008.

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

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

[23] J.M. Sanz-Serna and M.P. Calvo, Numerical Hamiltonian Problems, Chapman & Hall, London, 1994.

[24] J. Sastre, Efficient evaluation of matrix polynomials, Linear Algebra Appl.539 (2018) 229 - 250.

[25] J. Sastre, J. Ibáñez, P. Alonso-Jordá, J. Peinado, E. Defez, Fast Taylor polynomial evaluation for the computation of the matrix cosine, J. Comput. Appl. Math. 354 (2019) 641 - 650.

[26] J. Sastre, J. Ibáñez, and E. Defez, Boosting the computation of the matrix exponential, Appl. Math. Comput. 340 (1019) 206 - 220.

[27] R.B. Sidje, Expokit: a software package for computing matrix exponentials, ACM Trans. Math. Software 24 (1998) 130 - 156.

[28] Van Loan, C. A note on the evaluation of matrix polynomials, IEEE Transactions on Automatic Control 24 (1979) 320 - 321.

[29] Westreich, D. Evaluating the matrix polynomial $I + A + \cdots + A^{N-1}$, IEEE Trans. Circuits Sys. 36 (1989) 162 - 164.

[30] E. D. Dolan and J. J. More, Benchmarking optimization software with performance profiles, Math. Programming 91 (2002) 201 - 213.