Sastre, J.; Ibáñez González, JJ.; Defez Candel, E.; Ruíz Martínez, PA. (2011). Accurate matrix exponential computation to solve coupled differential models in engineering. Mathematical and Computer Modelling. 54(7-8):1835-1840. doi:10.1016/j.mcm.2010.12.049.

Accurate matrix exponential computation to solve coupled differential models in Engineering[☆]

J. Sastre[a,*], J. Ibáñez[b], E. Defez[c], P. Ruiz[b]

Universidad Politécnica de Valencia, Spain

[a] *Instituto de Telecomunicaciones y Aplicaciones Multimedia*
[b] *Instituto de Instrumentación para Imagen Molecular*
[c] *Instituto de Matemática Multidisciplinar*

Abstract

The matrix exponential plays a fundamental role in linear systems arising in engineering, mechanics and control theory. This work presents a new scaling-squaring algorithm for matrix exponential computation. It uses forward and backward error analysis with improved bounds for normal and nonnormal matrices. Applied to the Taylor method, it has presented a lower or similar cost compared to the state-of-the-art Padé algorithms with better accuracy results in the majority of test matrices, avoiding Padé's denominator condition problems.

Keywords: Matrix exponential, scaling and squaring, Taylor series.

1. Introduction

Many engineering processes are described by systems of linear first-order ordinary differential equations with constant coefficients, whose solutions are given in terms of the matrix exponential, and a large number of methods for its computation have been proposed [1, 2]. This work presents a competitive new scaling and squaring algorithm for matrix exponential computation. Throughout this paper $\mathbb{C}^{n\times n}$ denotes the set of complex matrices of size $n\times n$,

[☆]This work has been supported by *Universidad Politécnica de Valencia* grants PAID-05-09-4338, PAID-06-08-3307 and Spanish *Ministerio de Educación* grant MTM2009-08587.

[*]Corresponding author
Email address: jorsasma@iteam.upv.es (J. Sastre)

I denotes the identity matrix for this set, $\rho(A)$ is the spectral radius of matrix A, and $\mathbb{N}$ denotes the set of positive integers. The matrix norm $\|\cdot\|$ denotes any subordinate matrix norm; in particular $\|\cdot\|_1$ is the 1-norm. This paper is organized as follows. Section 2 presents the scaling and squaring error analysis and the developed algorithm, and Section 3 deals with numerical tests and conclusions. Next theorem will be used in next section to bound the norm of matrix power series.

Theorem 1. *Let $h_l(x) = \sum_{k\geq l} b_k x^k$ be a power series with radius of convergence w, and let $\tilde{h}_l(x) = \sum_{k\geq l} |b_k| x^k$. For any matrix $A \in \mathbb{C}^{n\times n}$ with $\rho(A) < w$, if a_k is an upper bound for $||A^k||$ ($||A^k|| \leq a_k$), $p \in \mathbb{N}$, $1 \leq p \leq l$, and $\alpha_p = \max\{(a_k)^{\frac{1}{k}} : k = p, l, l+1, \ldots, l+p-1\}$, then $||h_l(A)|| \leq \tilde{h}_l(\alpha_p)$. If $p = 2$ and l is odd the same bound holds taking $\alpha_2 = \max\{(a_k)^{\frac{1}{k}} : k = 2, l\}$.*

Proof. For the first part note that

$$||h_l(A)|| \leq \sum_{j\geq 0} \sum_{i=l}^{l+p-1} |b_{i+jp}| ||A^p||^j ||A^i|| \leq \sum_{j\geq 0} \sum_{i=l}^{l+p-1} |b_{i+jp}| \alpha_p^{i+jp} = \sum_{k\geq l} |b_k| \alpha_p^k = \tilde{h}_l(\alpha_p). \tag{1}$$

If $p = 2$ and l is odd note that if $k \geq l$ is odd then $k = 2j+1$, $j \in \mathbb{N}$, and one gets $||A^k|| = ||A^{2j+1}|| \leq ||A^l|| ||A^2||^{\frac{2j+1-l}{2}} \leq a_l a_2^{\frac{2j+1-l}{2}} = \left(a_l^{1/l}\right)^l \left(a_2^{1/2}\right)^{2j+1-l} \leq \max\left\{a_l^{1/l}, a_2^{1/2}\right\}^{2j+1} = \alpha_2^k$, and for even $k > l$, $k = 2j$, $j \in \mathbb{N}$, $||A^k|| \leq ||A^2||^j \leq \left(a_2^{1/2}\right)^{2j} \leq \alpha_2^k$. Hence

$$||h_l(A)|| \leq \sum_{k\geq l} |b_k| ||A^k|| \leq \sum_{k\geq l} |b_k| \alpha_2^k = \tilde{h}_l(\alpha_2). \quad \square \tag{2}$$

2. Error analysis and algorithm

If we denote the truncated matrix exponential Taylor series as $T_m(A) = \sum_{i=0}^m A^i/i!$, and its remainder as $R_m(A) = \sum_{i\geq m+1} A^i/i!$, for a scaled matrix $2^{-s}A$, $s \in \mathbb{N} \cup \{0\}$, see [3], we can write

$$\left(T_m(2^{-s}A)\right)^{2^s} = e^A \left(I + g_{m+1}(2^{-s}A)\right)^{2^s} = e^{A + 2^s h_{m+1}\left(2^{-s}A\right)}, \tag{3}$$

$$g_{m+1}(2^{-s}A) = -e^{-2^{-s}A} R_m(2^{-s}A), \; h_{m+1}\left(2^{-s}A\right) = \log\left(I + g_{m+1}(2^{-s}A)\right), \tag{4}$$

where log denotes the principal logarithm, $h_{m+1}(X)$ is defined in the set $\Omega_m = \left\{X \in \mathbb{C}^{n\times n} : \rho\left(e^{-X}T_m(X) - I\right) < 1\right\}$, see [4, sec. 3], and both $g_{m+1}(2^{-s}A)$ and $h_{m+1}\left(2^{-s}A\right)$ are holomorphic functions of A in Ω_m and then commute with A. If we choose s so that $2^{-s}A \in \Omega_m$, then from (3) one gets that $\Delta A = 2^s h_{m+1}\left(2^{-s}A\right)$ and $\Delta E = e^A\left[\left(I + g_{m+1}\left(2^{-s}A\right)\right)^{2^s} - I\right]$ represent the backward and forward errors in exact arithmetic from the approximation of e^A by the Taylor series with scaling and squaring, respectively. If s is chosen so that

$$\left\|h_{m+1}\left(2^{-s}A\right)\right\| \leq \max\left\{1, \left\|2^{-s}A\right\|\right\} u, \tag{5}$$

where $u = 2^{-53}$ is the unit roundoff in IEEE double precision arithmetic, then: if $2^{-s}\|A\| \geq 1$, then $\Delta A \leq \|A\| u$ and using (3) one gets $\left(T_m\left(2^{-s}A\right)\right)^{2^s} = e^{A+\Delta A} \approx e^A$, and if $2^{-s}\|A\| < 1$, using (3)-(5) and the Taylor series one gets

$$\begin{aligned}\left\|R_m\left(2^{-s}A\right)\right\| &= \left\|e^{2^{-s}A} g_{m+1}\left(2^{-s}A\right)\right\| = \left\|e^{2^{-s}A}\left(e^{h_{m+1}\left(2^{-s}A\right)} - I\right)\right\| \\ &= \left\|e^{2^{-s}A}\sum_{k\geq 1}\left(h_{m+1}\left(2^{-s}A\right)\right)^k/k!\right\| \leq \left\|e^{2^{-s}A}\right\|\sum_{k\geq 1} u^k/k! \\ &\approx \left\|T_m\left(2^{-s}A\right)\right\| u\left(1+u/2!+u^2/3!+\cdots\right) \approx \left\|T_m\left(2^{-s}A\right)\right\| u. \end{aligned} \tag{6}$$

Hence, as we will evaluate explicitly $T_m\left(2^{-s}A\right)$, by (6) one gets $T_m\left(2^{-s}A\right) + R_m\left(2^{-s}A\right) \approx T_m\left(2^{-s}A\right)$, and there is no need to increase m or the scaling parameter s to try to get better accuracy. Using the Taylor series in (4) one gets

$$g_{m+1}(x) = \sum_{k\geq m+1} b_k^{(m)} x^k, \quad h_{m+1}(x) = \sum_{k\geq 1}\frac{(-1)^{k+1}(g_{m+1}(x))^k}{k} = \sum_{k\geq m+1} c_k^{(m)} x^k, \tag{7}$$

where $b_k^{(m)}$ and $c_k^{(m)}$ depend on order m, and $b_k^{(m)} = c_k^{(m)}, k = m+1, m+2, \ldots, 2m+1$. Using MATLAB symbolic Math Toolbox, high precision arithmetic, 200 series terms and a zero finder we obtained the maximal values Θ_m of $\Theta = \|2^{-s}A\|$, shown in Table 1, such that, using the notation of Theorem 1

$$||h_{m+1}\left(2^{-s}A\right)|| \leq \tilde{h}_{m+1}(\Theta) = \textstyle\sum_{k\geq m+1} c_k^{(m)}\Theta^k \leq \max\{1,\Theta\}\, u. \tag{8}$$

Hence, if $||2^{-s}A|| \leq \Theta_m$ then (5) holds. For the cases where $\Theta_m > 1$, note that $f(\Theta) = \tilde{h}_{m+1}(\Theta) - \Theta u$ is a continuous function in $[0, \Theta_m]$ and $f(\Theta_m) = 0$, $f(0) = 0$. For $m = 20, 25, 30$ we have checked that there are no other zeros in $[0, \Theta_m]$, and $f(\Theta) < 0$, $\Theta \in]0, \Theta_m[$. Thus, for those orders the next bound

holds

$$||h_{m+1}\left(2^{-s}A\right)|| \leq \tilde{h}_{m+1}\left(||2^{-s}A||\right) = \tilde{h}_{m+1}\left(\Theta\right) \leq \Theta u, \quad 0 \leq \Theta \leq \Theta_m. \quad (9)$$

In Section 2.2 we will obtain an initial maximum value of the scaling parameter s, denoted by s_0, using values Θ_m, Theorem 1, (9) and the powers of A computed for the evaluation of $T_m\left(2^{-s}A\right)$ which we analyze in next subsection.

2.1. Taylor matrix polynomial evaluation

For the evaluation of $T_m\left(2^{-s}A\right)$ we have improved the Horner's and Paterson-Stockmeyer's method of [3] calculating matrix powers $A_i = A^i, i = 2, 3, \ldots, q$ in the same way, but including the scaling in the Taylor series coefficients and saving some divisions of matrix A by scalar as follows:

$$\begin{aligned} T_m\left(2^{-s}A\right) = & \left\{\left\{\cdots\left\{\frac{A_q}{2^s m}+A_{q-1}\right\}/[2^s(m-1)]+A_{q-2}\right\}/[2^s(m-2)]+\cdots+A_2\right\}/[2^s(m-q+2)]+A \\ & +2^s(m-q+1)I\Bigg\}\frac{A_q}{2^{2s}(m-q+1)(m-q)}+A_{q-1}\Bigg\}/[2^s(m-q-1)]+A_{q-2}\Bigg\} \\ & /\left[2^s(m-q-2)\right]+\cdots+A_2\Bigg\}/[2^s(m-2q+2)]+A+2^s(m-2q+1)I\Bigg\} \\ & \times\frac{A_q}{2^{2s}(m-2q+1)(m-2q)}+\cdots+A_2\Bigg\}/[2^s(q+2)]+A+2^s(q+1)I\Bigg\} \\ & \times\frac{A_q}{2^{2s}(q+1)q}+A_{q-1}\Bigg\}/[2^s(q-1)]+\cdots+A_2\Bigg\}/[2^s 2]+A\Bigg\}/2^s+I. \end{aligned} \quad (10)$$

Note that the matrix powers A_i will be obtained before the optimal scaling s is calculated and with this formula it is not necessary to calculate explicitly and save scaled matrices $A_i/2^{si} \rightarrow A_i, i = 1, 2, \ldots, q$. We will use the optimal values of m in terms of the number of evaluations of matrix products $m_k = [1, 2, 4, 6, 9, 12, 16, 20, 25, 30]$, $k = 0, 1, \ldots, 9$, respectively, see [3]. If the maximum allowed order, denoted by m_M, is 25 or 30, we will take $q = [1, 2, 2, 3, 3, 4, 4, 5, 5, 5]$ for each value of m_k, respectively, because in the scaling algorithm it will be necessary that the two last orders m_{M-1} and m_M use the same matrix powers of A, i.e. $A^i, i = 2, 3, \ldots, q$. If the maximum allowed order is $m_M = 20$ we will use $q = 4$ for that order for the same reason. Counting the number of evaluations of matrix products in (10), denoted by Π_{m_k}, including those for obtaining matrix powers A^i, $i = 2, 3, \ldots, q$, for the proposed values of m_k and q we have that using (10), $T_{m_k}\left(2^{-s}A\right)$ is evaluated in $\Pi_{m_k} = k$ matrix products. Similar rounding error bounds to those in [3] could be applied to the intermediate results in $T_{m_k}\left(2^{-s}A\right)$ to try to save matrix products.

2.2. Scaling algorithm

For all norms appearing in the scaling algorithm we will use the 1-norm. Let m_M be the maximum allowed Taylor order. Using the same bounds and process that we will use in the proposed scaling algorithm described below, we will first check if any of the Taylor optimal orders $m_k = 1, 2, 4, \ldots, m_{M-1}$ satisfy (5) without scaling, i.e. with $s = 0$. If not, we will calculate the optimal scaling s for order m_M as follows: First, we will compute the 1-norm estimate of $||A^{m_M+1}||$ using the block 1-norm estimation algorithm of [5]. For a $n \times n$ matrix this algorithm carries out a 1-norm power iteration whose iterates are $n \times t$ matrices, where t is a parameter that has been taken to be 2, see [4, p. 983]. Hence, the estimation algorithm has $O(n^2)$ computational cost, negligible compared to matrix products, whose cost is $O(n^3)$. The bounds a_k for $||A^k||$ needed to apply Theorem 1 in (8) will be obtained using the products of norms of matrix powers estimated for previous and current tested orders, $||A^{m_k+1}||,\ k = 0, 1, 2, \ldots, M$, and the powers of A computed for the evaluation of $T_{m_M}(2^{-s}A)$, $A^i,\ i = 1, 2, \ldots, q$, as

$$\begin{aligned}\left\|A^k\right\| \le a_k = \min\Big\{ & \|A\|^{i_1} \left\|A^2\right\|^{i_2} \cdots \|A^q\|^{i_q} \left\|A^{m_1+1}\right\|^{i_{m_1+1}} \left\|A^{m_2+1}\right\|^{i_{m_2+1}} \cdots \\ & \times \left\|A^{m_M+1}\right\|^{i_{m_M+1}} : i_1 + 2i_2 + \cdots + qi_q + (m_1+1)i_{m_1+1} \\ & + (m_2+1)i_{m_2+1} + \cdots + (m_M+1)i_{m_M+1} = k\Big\}, \end{aligned} \tag{11}$$

where the minimum is desirable, but not necessary. We will obtain successively α_p value of Theorem 1 with $l = m_M + 1$ for $p = 2, 3, \ldots, q, m_1 + 1, m_2+1, \ldots, m_M+1$, stopping the process when $(a_p)^{1/p} \le \max\{(a_k)^{1/k} : k = m+1, m+2, \ldots, m+p\}$. We will select the minimum value of all values α_p, denoted by α_{min}. Then we will take the appropriate initial minimum scaling parameter $s_0 \ge 0$ so that $2^{-s_0}\alpha_{min} \le \Theta_{m_M}$, i.e. if $\alpha_{min} \le \Theta_{m_M}$ then $s_0 = 0$, and otherwise $s_0 = \lceil \log_2(\alpha_{min}/\Theta_{m_M}) \rceil$. Then, if $\Theta_{m_M} \le 1$ using Theorem 1 and (8), and taking for simplicity in the rest of the algorithm description $m = m_M$, it follows that

$$||h_{m+1}\left(2^{-s_0}A\right)|| \le \tilde{h}_{m+1}\left(2^{-s_0}\alpha_{min}\right) \le \tilde{h}_{m+1}\left(\Theta_m\right) \le u, \tag{12}$$

and (5) holds. Taking into account that $||A^k||^{1/k} \le ||A||$, from (11) it follows that $a_k^{1/k} \le (||A||^k)^{1/k} = ||A||$. Thus, α_{min} from Theorem 1 satisfies $\alpha_{min} \le ||A||$. Hence, if $m = 20, 25$ or 30, where $\Theta_m > 1$, using (9) one gets

$$||h_{m+1}\left(2^{-s_0}A\right)|| \le \tilde{h}_{m+1}\left(2^{-s_0}\alpha_{min}\right) \le 2^{-s_0}\alpha_{min}\, u \le 2^{-s_0}||A||u, \tag{13}$$

Table 1: Maximal values $\Theta_m = \|2^{-s}A\|$ such that $\tilde{h}_{m+1}(\Theta_m) \leq \max\{1, \Theta_m\}u$, coefficient ratios $c_k^{(m)}/c_{m+2}^{(m)}$ for the first values of $k \geq m+1$, and values $u/c_{m+2}^{(m)}$.

		$c_k^{(m)}/c_{m+2}^{(m)}$				$\frac{u}{c_{m+2}^{(m)}}$
m	Θ_m	$m+1$	$m+3$	$m+4$	$m+5$	
1	1.490116111983279e-8	-3/2	-3/4	3/5	-1/2	3.3e-16
2	8.733457513635361e-6	-4/3	-2/5	0	1/7	8.9e-16
4	1.678018844321752e-3	-6/5	-3/7	1/8	-1/36	1.6e-14
6	1.773082199654024e-2	-8/7	-4/9	2/15	-1/33	6.4e-13
9	1.137689245787824e-1	-11/10	-11/24	11/78	-11/336	4.4e-10
12	3.280542018037257e-1	-14/13	-7/15	7/48	-7/204	7.5e-7
16	7.912740176600240e-1	-18/17	-9/19	3/20	-1/28	4.2e-2
20	1.438252596804337	-22/21	-11/23	11/72	-11/300	5.9e03
25	2.428582524442827	-27/26	-27/56	9/58	-3/80	4.7e10
30	3.539666348743690	-32/31	-16/33	8/51	-4/105	9.4e17

and (5) also holds. Once obtained s_0, if $s_0 \geq 1$ check if (5) holds reducing the scaling $s = s_0 - 1$, and using the bounds for $\left\|A^k\right\| \leq a_k$ to test if bound

$$\frac{\|h_{m+1}(2^{-s}A)\|}{|c_{m+2}^{(m)}|} \leq \sum_{k \geq m+1} \left| \frac{c_k^{(m)}}{c_{m+2}^{(m)}} \right| \frac{a_k}{2^{sk}} \leq \max\left\{1, \left\|2^{-s}A\right\|\right\} \frac{u}{|c_{m+2}^{(m)}|}, \tag{14}$$

holds, truncating the series. Note that we will stop the series summation if after summing one term the sum is greater than $\max\{1, ||2^{-s}A||\}u/|c_{m+2}^{(m)}|$. This has been the case many times in numerical tests after calculating just the first series term. If the sum of one or more terms is lower than the bound but the complete truncated series sum is not, we can estimate $||A^{m+2}||$ to improve the bound a_{m+2} and check if (14) holds then. This has improved the computational cost in numerical tests. If (14) does not hold with $s = s_0 - 1$, then we will check if next bound holds

$$\begin{aligned} \frac{\|h_{m+1}(2^{-s}A)\|}{|c_{m+2}^{(m)}|} \leq & \frac{\left\|A^{m+1}\right\|}{2^{s(m+2)}} \left\| \frac{c_{m+1}^{(m)} 2^s I}{c_{m+2}^{(m)}} + A + \frac{c_{m+3}^{(m)} A_2}{c_{m+2}^{(m)} 2^s} + \frac{c_{m+4}^{(m)} A_3}{c_{m+2}^{(m)} 2^{2s}} + \cdots + \frac{c_{m+q+1}^{(m)} A_q}{c_{m+2}^{(m)} 2^{s(q-1)}} \right\| \\ & + \sum_{k \geq m+2+q} \left| \frac{c_k^{(m)}}{c_{m+2}^{(m)}} \right| \frac{a_k}{2^{sk}} \leq \max\left\{1, \left\|2^{-s}A\right\|\right\} \frac{u}{|c_{m+2}^{(m)}|} \end{aligned} \tag{15}$$

where we will truncate the series by $k = m+N$, with $N \geq 2+q$, and we have divided by the coefficient of A to save the product of matrix A by a scalar.

We propose using at least one term of the infinite series because the norm of the previous matrix polynomial in (15) might vanish in some cases where the infinite series might be large, e.g. scalar A when A is a zero of the resulting scalar polynomial. For convenience we will also truncate the series in (14) by the same value of k, i.e. $k = m + N$. For the proposed orders m_k and the first 1000 series terms we have observed in (7) that $b_{m+j}^{(m)} = (-1)^j |b_{m+j}^{(m)}|$ and $1/(j+1) < |b_{m+1+j}^{(m)}/b_{m+j}^{(m)}| < 1/j$, $j \geq 1$, and then bounds for the last terms of $g_{m+1}(||2^{-s}A||)$ and $h_{m+1}(||2^{-s}A||)$ can be obtained. Table 1 presents some values of $c_k^{(m)}/c_{m+2}^{(m)}$, and the values $u/|c_{m+2}^{(m)}|$. Next we obtain lower bounds for expression (15) to avoid its unnecessary evaluation: Taking

$$T_{max} = \max\left\{ \left| \frac{c_{m+1}^{(m)}}{c_{m+2}^{(m)}} \right| 2^s, \|A\|, \left| \frac{c_{m+3}^{(m)}}{c_{m+2}^{(m)}} \right| \|A_2\| / 2^s, \cdots, \left| \frac{c_{m+q+1}^{(m)}}{c_{m+2}^{(m)}} \right| \|A_q\| / 2^{s(q-1)} \right\}, \tag{16}$$

and T_i as the other q elements of the same set, note that

$$\left\| \frac{c_{m+1}^{(m)} 2^s I}{c_{m+2}^{(m)}} + A + \frac{c_{m+3}^{(m)} A_2}{c_{m+2}^{(m)} 2^s} + \frac{c_{m+4}^{(m)} A_3}{c_{m+2}^{(m)} 2^{2s}} + \cdots + \frac{c_{m+q+1}^{(m)} A_q}{c_{m+2}^{(m)} 2^{s(q-1)}} \right\| \geq minsum, \tag{17}$$

where $minsum = \max\{0, T_{max} - \sum_{i=1}^{q} T_i\}$, and if

$$\frac{\left\|A^{m+1}\right\|}{2^{s(m+2)}} \times minsum + \sum_{k=m+2+q}^{m+N} \left| \frac{c_k^{(m)}}{c_{m+2}^{(m)}} \right| \frac{a_k}{2^{sk}} > \max\left\{1, \left\|2^{-s}A\right\|\right\} \frac{u}{|c_{m+2}^{(m)}|}, \tag{18}$$

then there is no need to evaluate bound (15). Using (18) saved many times the evaluation of (15) in numerical tests. Using now the 2-norm and taking into account that for normal matrices $||A^i||_2 = ||A||_2^i$, $i = 2, 3, \ldots$, for any scalar coefficients $d_k \in \mathbb{R}, k = 0, 1, \ldots, q$, one gets

$$\begin{aligned} ||A^{m+1}||_2 ||d_0 I + d_1 A + \cdots + d_q A^q||_2 \leq ||A^{m+1}||_2 (|d_0| + |d_1| ||A||_2 + \cdots \\ + |d_q| ||A^q||_2) = |d_0| ||A^{m+1}||_2 + |d_1| ||A^{m+2}||_2 + \cdots + |d_q| ||A^{m+1+q}||_2, \end{aligned} \tag{19}$$

and then using the 2-norm, the bound in (15) is lower or equal than the bound in (14) for normal matrices. As our algorithm uses 1-norm, any of (14) or (15) may be the lower bound depending on the matrix. In the case of nonnormal matrices any of (14) or (15) may also be the lower bound. Moreover, the first $m + 1$ non-zero coefficients of $h_{m+1}(x)$ have an alternating sign, see Table 1, and then the bound in (15) may be higher for matrices with all negative

elements and lower for matrices with all positive elements. For instance, considering normal matrices B_1 and B_2, and nonnormal matrices B_3, B_4

$$B_1 = \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix},\ B_2 = \begin{pmatrix} 1 & 2 \\ 2 & -1 \end{pmatrix},\ B_3 = \begin{pmatrix} 1 & 25 \\ 0 & -1 \end{pmatrix},\ B_4 = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}, \qquad (20)$$

and $B_5 = -B_4$, for $m = 4$ one gets $\|B_i^5\|_1\, ||c_5^{(4)}/c_6^{(4)} I + B_i + c_7^{(4)}/c_6^{(4)} B_i^2||_1 = 108.3, 475.7, 718.3, 175.4, 712.8$ and $\|B_i^5\|_1\, |c_5^{(4)}/c_6^{(4)}| + \|B_i^6\|_1 + \|B_i^7\|_1\, |c_7^{(4)}/c_6^{(4)}| = 387.4,\ 375.7, 43.3, 605.1, 605.1$, respectively, confirming that any of both bounds may be the best depending on the case. We have also obtained the corresponding α_{min} values from the scaling algorithm, for $m = 4$, using norms of matrix powers $||B_i^2||_1$ and $||B_i^5||_1$, i.e. $\alpha_{min} = 2.53, 2.37, 1.92, 2.60, 2.60$, being lower in all cases than those obtained using theorem 4.2 of [4], i.e. $\max\{||B_i^2||_1^{1/2},\, ||B_i^3||_1^{1/3}\} = 2.65, 2.47, 2.96, 2.65, 2.65$, respectively.

If any of both bounds (14) or (15) is satisfied with $s_0 - 1$ then repeat the process with $s = s_0 - 2, s_0 - 3, \ldots$ Note that the computational cost of evaluating (14) or (15) is $O(n^2)$ and if any of them is satisfied with $0 \leq s < s_0$ their evaluation saves matrix products, whose cost is $O(n^3)$. If the last value of scaling parameter s where (14) or (15) are satisfied is $s \geq 1$ then if $\Theta_{m_M} < 2\Theta_{m_{M-1}}$ it is possible that the same value of scaling s and order m_{M-1} also satisfy (14) or (15), see [3], and this occurred in numerical tests. Thus, if the final resulting scaling is $s \geq 1$ we propose testing bounds (14) and/or (15) with the same value of the scaling parameter s, and order m_{M-1}. Finally, the algorithm will return s and the minimum order satisfying (14) or (15), which may be m_M or m_{M-1}. It is possible to evaluate $T_m(2^{-s}A)$ with both orders with the optimal number of matrix products at this point because we set in its evaluation that both last orders used the same matrix powers of A.

The complete matrix exponential computation algorithm will consist of: using Theorem 1, (14) and (15) check if one of orders $m = 1, 2, 4, \ldots, m_{M-1}$ satisfies (5) with $s = 0$. If not, obtain the values of scaling parameter s and order m using the previous algorithm, and use (10) and squaring to evaluate $\left(T_m\left(2^{-s}A\right)\right)^{2^s}$. We have made available online the commented MATLAB implementation of the algorithm, denoted by `exptayns`, in

http://personales.upv.es/∼jorsasma/exptayns.zip

If $\hat{T}_m(A)$ denotes the computed Taylor approximation, using error analysis techniques for the evaluation of matrix products from [6, sec. 3.5] we have

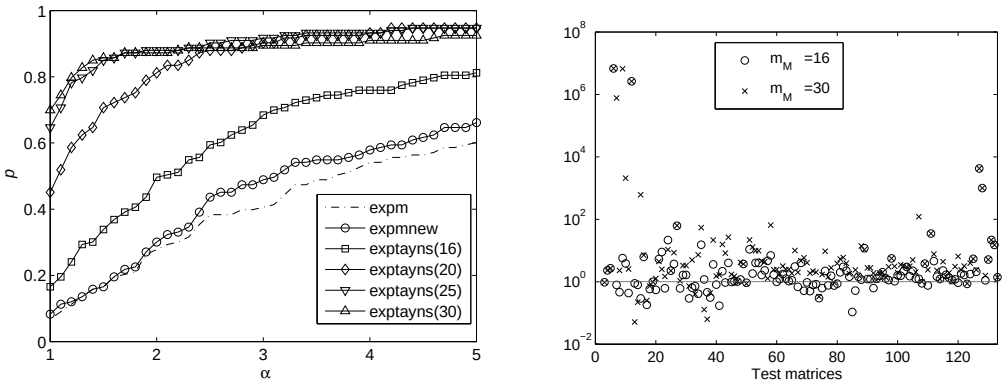
Table 2: Cost in terms of total number of matrix product evaluations (P) and relative error comparison between `exptayns`, `expm` and `expm_new`.

Maximum allowed Taylor order m_M	16	20	25	30
$\mathrm{E}_{\texttt{exptayns}} < \mathrm{E}_{\texttt{expm}}$ %	74.44	90.98	89.47	88.72
$(\mathrm{P}_{\texttt{exptayns}} - \mathrm{P}_{\texttt{expm}})/\mathrm{P}_{\texttt{expm}}$ %	-15.47	-15.69	-14.95	-14.35
$\mathrm{E}_{\texttt{exptayns}} < \mathrm{E}_{\texttt{expm\_new}}$ %	66.17	87.22	87.22	86.47
$(\mathrm{P}_{\texttt{exptayns}} - \mathrm{P}_{\texttt{expm\_new}})/\mathrm{P}_{\texttt{expm\_new}}$ %	1.31	1.04	1.94	2.65

that $\left\| T_m(A) - \hat{T}_m(A) \right\| \leq \tilde{\gamma}_{mn} T_m\left(\|A\|\right) \leq \tilde{\gamma}_{mn} e^{\|A\|}$, where $\tilde{\gamma}_k = \frac{cku}{1-cku}$, with c a small integer constant. This bound might be unsatisfactory taking into account that with the proposed scaling algorithm $\|A\|$ can be large. However, the proposed algorithm behaved in a stable way in all numerical tests.

3. Numerical experiments and conclusions

133 matrices from 2×2 to 10×10 from MATLAB (`gallery` test matrices and other special matrices), Eigtool package [7], and references [3, 8], have been used to compare the proposed algorithm `exptayns` to MATLAB functions `expm` [8], and `expm_new` from [4]. The accuracy was tested by computing relative errors $\mathrm{E} = ||e^A - \tilde{X}||_1 / ||e^A||_1$, where $\tilde{X}$ is the computed approximation. The "exact" value of matrix exponential e^A was computed using MATLAB's Symbolic Math Toolbox and a [33/33] diagonal Padé method with scaling and squaring at 1000 decimal digit precision. We have compared function `exptayns` truncating the series in (14) and (15) by $N = 150$ terms, and truncating them with $N = q + 2$ terms. The same results were obtained in almost the 100% of the test matrices and we used definitely $N = q+2$ series terms in the comparison with `expm` and `expm_new`. Table 2 shows that the cost for `exptayns` is lower than the cost for `expm`, and slightly greater than that for `expm_new`, and that `exptayns` is more accurate than both methods in the majority of test matrices. Figure 1a shows the performance profile [9] of the compared functions, where the α coordinate varies between 1 and 5 in steps equal to 0.1, and p coordinate is the probability that the considered method has a relative error lower than or equal to α-times the smallest error over all the methods, where probabilities are defined over all matrices. Figure 1b shows the ratio of relative errors $\mathrm{E}_{\texttt{expm\_new}}/\mathrm{E}_{\texttt{exptayns}}$ using the Taylor maximum orders $m_M = 16$ and 30. Figures 1a and 1b show that `exptayns`



(a) (b)

Figure 1: (a) Performance profile ($m_M = 16, 20, 25, 30$). (b) Ratio of relative errors $\mathrm{E}_{\texttt{expm\_new}}/\mathrm{E}_{\texttt{exptayns}}$ with Taylor maximum orders $m_M = 16$ and 30.

has better accuracy than the other functions in the majority of test matrices. A normwise relative error study [2, p. 252-253] was also made and showed that the three functions performed in a numerically stable way on this test.

To sum up, a new scaling and squaring competitive algorithm has been proposed. It is based on a mixed backward and forward error analysis which uses improved bounds for normal and nonnormal matrices. Applied to the Taylor method, it has shown to be more accurate than existing state-of-the-art algorithms in the majority of matrices in numerical tests, with lower or similar cost. Its extension to IEEE single precision arithmetic is straightforward. Now, we are applying the new scaling and squaring algorithm to the Padé method, however, denominator condition problems are expected as in [4].

4. References

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

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

[3] J. Sastre, J. Ibáñez, E. Defez, P. Ruiz, Efficient orthogonal matrix poly-

nomial based method for computing matrix exponential, Appl. Math. Comput. (2011) in press, (doi: 10.1016/j.amc.2011.01.004).

[4] A.H. Al-Mohy, N.J. Higham, A new scaling and squaring algorithm for the matrix exponential, SIAM J. Matrix Anal. Appl. 31 (3) (2009) 970–989.

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

[6] N.J. Higham. Accuracy and Stability of Numerical Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[7] G. Wright, www.comlab.ox.ac.uk/pseudospectra/eigtool/, (2002).

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

[9] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Programming 91 (2002) 201–213.