Efficient orthogonal matrix polynomial based method for computing matrix exponential

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

[a]*Instituto de Telecomunicaciones y Aplicaciones Multimedia,*
[b]*Instituto de Instrumentación para Imagen Molecular,*
[c]*Instituto de Matemática Multidisciplinar,*
Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia (Spain)

Abstract

The matrix exponential plays a fundamental role in the solution of differential systems which appear in different science fields. This paper presents an efficient method for computing matrix exponentials based on Hermite matrix polynomial expansions. Hermite series truncation together with scaling and squaring and the application of floating point arithmetic bounds to the intermediate results provide excellent accuracy results compared with the best acknowledged computational methods. A backward-error analysis of the approximation in exact arithmetic is given. This analysis is used to provide a theoretical estimate for the optimal scaling of matrices. Two algorithms based on this method have been implemented as MATLAB functions. They have been compared with MATLAB functions `funm` and `expm` obtaining greater accuracy in the majority of tests. A careful cost comparison analysis with `expm` is provided showing that the proposed algorithms have lower maximum cost for some matrix norm intervals. Numerical tests show that the application of floating point arithmetic bounds to the intermediate results may reduce considerably computational costs, reaching in numerical tests relative higher average costs than `expm` of only 4.43% for the final Hermite selected order, and obtaining better accuracy results in the 77.36% of the test matrices. The MATLAB implementation of the best Hermite matrix polynomial based algorithm has been made available online.

Keywords: differential equations, matrix exponential, Hermite matrix

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

polynomial approximation, matrix polynomial evaluation, error analysis
PACS: 87.64.Aa

---

1. Introduction

Many engineering processes are described by systems of linear first-order ordinary differential equations. Solving this kind of systems involves the evaluation of the exponential of square matrices, and the same occurs with the partial differential case when using the semi-discretization method [1]–[5].

A survey of methods for computing the exponential matrix was made by Moler and Van Loan in [4], which was updated in [6]. This paper and recent researches [7, 8] show that probably one of the more promising methods is the scaling and squaring technique, which is also the most widely used. This technique exploits the relation $\exp(A) = (\exp(2^{-s}A))^{2^s}$, for a square matrix A and a nonnegative integer scaling parameter s. Most approximations to the exponential of the scaled matrix $\exp(2^{-s}A)$ are based on Padé expansions. [7] provides a scaling and squaring Padé method with excellent results of efficiency and accuracy, which is the method implemented in the last MATLAB versions, and [8] has recently proposed a new scaling and squaring algorithm that alleviates the overscaling problem for nonnormal matrices.

In this paper we use Hermite matrix polynomial expansions of the matrix exponential together with scaling and squaring and the application of floating point arithmetic bounds to the intermediate results in order to perform a competitive method for computing the matrix exponential. This method has the advantage that it does not need the solution of multiple linear systems or the computation of matrix inversions. Moreover, adapting the analysis made in [7] to the Hermite matrix series, a backward-error bound in exact arithmetic has been provided to find the optimal matrix scaling, i.e. the optimal value of scaling parameter s. A careful theoretical cost comparison with the method in [7] has been provided, showing that Hermite method has lower cost for some matrix norm intervals.

This paper is organized as follows. Section 2 summarizes notation and previous results about Hermite matrix polynomials and includes the Hermite series expansion of the matrix exponential to be considered. Section 3 deals with the approximation error analysis and optimal scaling. Section 4 is addressed to study the cost of the method, and to present numerical tests in order to check the method accuracy performance. Finally conclusions are given in Section 5.

2. Hermite matrix polynomial series expansions of matrix exponential

Throughout this paper, for a complex number z, $\Re(z)$ and $\Im(z)$ denote its real and imaginary parts, respectively, and $\|\cdot\|$ denotes any subordinate matrix norm. $\mathbb{R}^{r\times r}$ and $\mathbb{C}^{r\times r}$ denote the set of real and complex matrices of size $r\times r$, respectively, and I denotes the identity matrix for these sets. For a matrix $A\in\mathbb{C}^{r\times r}$, its spectrum $\sigma(A)$ denotes the set of all the eigenvalues of A. If $f(z)$ and $g(z)$ are holomorphic functions of the complex variable z, which are defined in an open set Ω of the complex plane, and B is a matrix in $\mathbb{C}^{r\times r}$ with $\sigma(B)\subset\Omega$, then from the properties of the matrix functional calculus [9, p. 558], it follows that $f(B)g(B)=g(B)f(B)$. If $\mathbb{D}_0$ is the complex plane cut along the negative real axis and $\log(z)$ denotes the principal logarithm of z, [10, p. 72], then $z^{\frac{1}{2}}$ represents $\exp\left(\frac{1}{2}\log(z)\right)$. If B is a matrix with $\sigma(B)\subset\mathbb{D}_0$, then $B^{\frac{1}{2}}=\sqrt{B}$ denotes the image by $z^{\frac{1}{2}}$ of the matrix functional calculus acting on the matrix B. We say that matrix A in $\mathbb{C}^{r\times r}$ is a positive stable matrix if $Re(z)>0$ for all $z\in\sigma(A)$. $[x]$ denotes the entire part of x, $\lceil x\rceil$ denotes the least integer not less than x and $\lfloor x\rfloor$ denotes the greatest integer not exceeding x.

For the sake of clarity in the presentation of the next results we recall some properties and results about Hermite matrix polynomials that have been established in [11] and [12]. If B is a positive stable matrix in $\mathbb{C}^{r\times r}$, the nth Hermite matrix polynomial is defined in (3.4) of [12, p. 25] by

$$H_n(x,B)=n!\sum_{k=0}^{[\frac{n}{2}]}\frac{(-1)^k\left(x\sqrt{2B}\right)^{n-2k}}{k!(n-2k)!}\ , \tag{1}$$

and from (3.1) and (3.2) of [12, p. 24] one gets its generating function

$$G(x,t)=e^{xt\sqrt{2B}-t^2I}=\sum_{n\geq 0}H_n(x,B)t^n/n!,\quad |t|<\infty, \tag{2}$$

where $x\in\mathbb{C}$ and $t\in\mathbb{C}$. Taking $A=\sqrt{2B}$, $y=tx$ and $\lambda=1/t$ in (2) it follows that

$$e^{Ay}=e^{\frac{1}{\lambda^2}}\sum_{n\geq 0}\frac{1}{\lambda^n n!}H_n\left(\lambda y,\frac{1}{2}A^2\right),\ \lambda\in\mathbb{C},\quad y\in\mathbb{C},\quad A\in\mathbb{C}^{r\times r}, \tag{3}$$

without restrictions on $\sigma(A)$. From (1) one gets

$$H_n(\lambda y, A^2/2) = n! \sum_{k=0}^{[\frac{n}{2}]} \frac{(-1)^k (\lambda y A)^{n-2k}}{k!(n-2k)!} \ , \tag{4}$$

also without restrictions on $\sigma(A)$. Denoting by $h_m(\lambda y, A)$ the $m-$th partial sum of series (3), one gets

$$h_m(\lambda y, A) = e^{\frac{1}{\lambda^2}} \sum_{n=0}^{m} \frac{1}{\lambda^n n!} H_n\left(\lambda y, \frac{1}{2}A^2\right) \approx e^{Ay}, \quad \lambda\,, y \in \mathbb{C}, \quad A \in \mathbb{C}^{r\times r}. \tag{5}$$

This expansion is the one to be used in the method for computing the matrix exponential, and we will refer to m as the order of the approximation. Using (4) and (5) by induction it is easy to show that

$$h_m(\lambda y, A) = \begin{cases} e^{\frac{1}{\lambda^2}} \Big[E_0^{\frac{m-1}{2}} (I + Ay) + E_0^{\frac{m-1}{2}-1} \left(\frac{(Ay)^2}{2!} + \frac{(Ay)^3}{3!} \right) + \cdots \\ \quad \cdots + E_0^1 \left(\frac{(Ay)^{m-3}}{(m-3)!} + \frac{(Ay)^{m-2}}{(m-2)!} \right) + \frac{(Ay)^{m-1}}{(m-1)!} + \frac{(Ay)^m}{m!} \Big] \quad , \quad \text{odd} \quad m \ , \\ \\ e^{\frac{1}{\lambda^2}} \Big[E_0^{\frac{m}{2}} I + E_0^{\frac{m}{2}-1} \left(Ay + \frac{(Ay)^2}{2!} \right) + E_0^{\frac{m}{2}-2} \left(\frac{(Ay)^3}{3!} + \frac{(Ay)^4}{4!} \right) + \cdots \\ \quad \cdots + E_0^1 \left(\frac{(Ay)^{m-3}}{(m-3)!} + \frac{(Ay)^{m-2}}{(m-2)!} \right) + \frac{(Ay)^{m-1}}{(m-1)!} + \frac{(Ay)^m}{m!} \Big] \quad , \quad \text{even} \quad m \ , \end{cases} \tag{6}$$

where

$$E_i^j = \sum_{k=i}^{j} \frac{\left(-\frac{1}{\lambda^2}\right)^k}{k!} \ , \ i, j \in \mathbb{N}, j \geq i \ . \tag{7}$$

Note that for $|\lambda| \to \infty$, $e^{\frac{1}{\lambda^2}} = 1$, $E_0^j = 1$, and $h_m(\lambda y, A)$ tends to the Taylor series of order m of matrix exponential e^{Ay}.

3. Error analysis.

Using (6) with $y = 1$ and taking into account Taylor series of $\exp\left(-\frac{1}{\lambda^2}\right)$, for odd m it follows that

$$\begin{aligned} e^A - h_m(\lambda, A) &= \lim_{M\to\infty} \left(h_M(\lambda, A) - h_m(\lambda, A) \right) \\ &= e^{\frac{1}{\lambda^2}} \Bigg[\sum_{k \geq \frac{m+1}{2}} \frac{\left(-\frac{1}{\lambda^2}\right)^k}{k!} (I + A) + \sum_{k \geq \frac{m-1}{2}} \frac{\left(-\frac{1}{\lambda^2}\right)^k}{k!} \left(\frac{A^2}{2!} + \frac{A^3}{3!} \right) + \cdots \\ &\quad \cdots + \sum_{k\geq 1} \frac{\left(-\frac{1}{\lambda^2}\right)^k}{k!} \left(\frac{A^{m-1}}{(m-1)!} + \frac{A^m}{m!} \right) + \sum_{k\geq 0} \frac{\left(-\frac{1}{\lambda^2}\right)^k}{k!} \sum_{j\geq m+1} \frac{A^j}{j!} \Bigg] \end{aligned}$$

$$= e^{\frac{1}{\lambda^2}} \left[\sum_{k \geq \frac{m+1}{2}} \frac{\left(-\frac{1}{\lambda^2}\right)^k}{k!} e^A + E_{\frac{m-1}{2}}^{\frac{m-1}{2}} \left(\frac{A^2}{2!} + \frac{A^3}{3!} \right) + \right.$$

$$\left. + E_{\frac{m-3}{2}}^{\frac{m-1}{2}} \left(\frac{A^4}{4!} + \frac{A^5}{5!} \right) + \cdots + E_1^{\frac{m-1}{2}} \left(\frac{A^{m-1}}{(m-1)!} + \frac{A^m}{m!} \right) + E_0^{\frac{m-1}{2}} \sum_{j \geq m+1} \frac{A^j}{j!} \right]$$

$$= \left(1 - e^{\frac{1}{\lambda^2}} E_0^{\frac{m-1}{2}}\right) e^A + e^{\frac{1}{\lambda^2}} \left[E_{\frac{m-1}{2}}^{\frac{m-1}{2}} \left(\frac{A^2}{2!} + \frac{A^3}{3!} \right) + E_{\frac{m-3}{2}}^{\frac{m-1}{2}} \left(\frac{A^4}{4!} + \frac{A^5}{5!} \right) + \cdots \right.$$

$$\left. \cdots + E_1^{\frac{m-1}{2}} \left(\frac{A^{m-1}}{(m-1)!} + \frac{A^m}{m!} \right) + E_0^{\frac{m-1}{2}} \sum_{j \geq m+1} \frac{A^j}{j!} \right] . \tag{8}$$

Analogously, using (6), for even m one gets

$$e^A - h_m(\lambda, A) = \left(1 - e^{\frac{1}{\lambda^2}} E_0^{\frac{m}{2}}\right) e^A + e^{\frac{1}{\lambda^2}} \left[E_{\frac{m}{2}}^{\frac{m}{2}} \left(A + \frac{A^2}{2!} \right) + E_{\frac{m}{2}-1}^{\frac{m}{2}} \left(\frac{A^3}{3!} + \frac{A^4}{4!} \right) \right.$$

$$\left. + \cdots + E_1^{\frac{m}{2}} \left(\frac{A^{m-1}}{(m-1)!} + \frac{A^m}{m!} \right) + E_0^{\frac{m}{2}} \sum_{j \geq m+1} \frac{A^j}{j!} \right] . \tag{9}$$

In a similar way to the demonstration of Theorem 2.1 of [7, p. 1182] we have the following result:

Theorem 3.1. *Assume that the matrix exponential Hermite approximation $h_m(\lambda, A)$ of (5) satisfies*

$$e^{-2^{-s}A} h_m(\lambda, 2^{-s}A) = I + G , \tag{10}$$

where $\|G\| < 1$. Then there exists a matrix E that commutes with A such that

$$[h_m(\lambda, 2^{-s}A)]^{2^s} = e^{A+E} , \tag{11}$$

and

$$\frac{\|E\|}{\|A\|} \leq \frac{-\log(1 - \|G\|)}{\|2^{-s}A\|} . \tag{12}$$

We seek to bound the norm of G in (10) in terms of $\|2^{-s}A\|$. Define the function

$$\rho(\lambda, A) = e^{-A} h_m(\lambda, A) - I, \tag{13}$$

and note that using (8), (9) and (13) one gets

$$\rho(\lambda, A) = -e^{-A}(e^A - h_m(\lambda, A)) \tag{14}$$

$$= \begin{cases} \left(e^{\frac{1}{\lambda^2}} E_0^{\frac{m-1}{2}} - 1\right) I - e^{-A} e^{\frac{1}{\lambda^2}} \left[E_{\frac{m-1}{2}}^{\frac{m-1}{2}} \left(\frac{A^2}{2!} + \frac{A^3}{3!}\right) + E_{\frac{m-3}{2}}^{\frac{m-1}{2}} \left(\frac{A^4}{4!} + \frac{A^5}{5!}\right) + \right. \\ \left. + \cdots + E_1^{\frac{m-1}{2}} \left(\frac{A^{m-1}}{(m-1)!} + \frac{A^m}{m!}\right) + E_0^{\frac{m-1}{2}} \sum_{j \geq m+1} \frac{A^j}{j!} \right] , \text{ odd } m \\ \\ \left(e^{\frac{1}{\lambda^2}} E_0^{\frac{m}{2}} - 1\right) I - e^{-A} e^{\frac{1}{\lambda^2}} \left[E_{\frac{m}{2}}^{\frac{m}{2}} \left(A + \frac{A^2}{2!}\right) + E_{\frac{m}{2}-1}^{\frac{m}{2}} \left(\frac{A^3}{3!} + \frac{A^4}{4!}\right) + \right. \\ \left. + \cdots + E_1^{\frac{m}{2}} \left(\frac{A^{m-1}}{(m-1)!} + \frac{A^m}{m!}\right) + E_0^{\frac{m}{2}} \sum_{j \geq m+1} \frac{A^j}{j!} \right] , \text{ even } m \end{cases} \tag{15}$$

Hence, taking into account (15), and using Taylor series of e^{-A} one gets

$$\left\|\rho(\lambda, 2^{-s}A)\right\| \leq \begin{cases} \left|e^{\frac{1}{\lambda^2}} E_0^{\frac{m-1}{2}} - 1\right| + e^{\Re\left(\frac{1}{\lambda^2}\right)} \sum_{j \geq 2} |c_j| \, \theta^j , \text{ odd } m \\ \left|e^{\frac{1}{\lambda^2}} E_0^{\frac{m}{2}} - 1\right| + e^{\Re\left(\frac{1}{\lambda^2}\right)} \sum_{j \geq 1} \left|c'_j\right| \theta^j , \text{ even } m \end{cases} \tag{16}$$

where $\theta = \|2^{-s}A\|$. Note that using (15) for $|\lambda| \to \infty$ it follows that

$$\|\rho(\lambda, A)\| = \left\| -e^{-A} \sum_{j \geq m+1} \frac{A^j}{j!} \right\| = \left\| e^{-A} \sum_{j=0}^{m} \frac{A^j}{j!} - I \right\| , \ |\lambda| \to \infty , \tag{17}$$

which is the corresponding bound for Taylor series approximation of matrix exponential. Using (10), (15) and (16) it follows that

$$\|G\| = \left\|\rho(\lambda, 2^{-s}A)\right\| \leq f_m(\lambda, \theta) , \tag{18}$$

where $f_m(\lambda, \theta)$ is given by the expressions on the right hand side of (16) depending on m being odd or even. Note that taking $|\lambda| \to \infty$, using (17), (18) and Taylor expansion of e^{-A}, it follows that

$$\|G\| = \left\|\rho(\lambda, 2^{-s}A)\right\| \leq f_m(|\lambda| \to \infty, \theta) \tag{19}$$

where

$$f_m(|\lambda| \to \infty, \theta) = \sum_{j \geq m+1} \left|c''_j\right| \theta^j , \tag{20}$$

which is the corresponding bound for Taylor approximation of matrix exponential. Combining (12) with (18) one gets

$$\frac{\|E\|}{\|A\|} \leq \frac{-\log(1 - f_m(\lambda, \theta))}{\theta} . \tag{21}$$

m	1	2	3	4	5	6	7	8
θ_m^∞	2.220e-16	2.581e-8	1.386e-5	3.397e-4	2.401e-3	9.066e-3	2.384e-2	4.991e-2
m	9	10	11	12	13	14	15	16
θ_m^∞	8.958e-2	1.448e-1	2.142e-1	2.996e-1	3.998e-1	5.139e-1	6.411e-1	7.803e-1
m	17	18	19	20	21	22	23	24
θ_m^∞	9.305e-1	1.091	1.260	1.438	1.624	1.816	2.015	2.219
m	25	26	27	28	29	30		
θ_m^∞	2.429	2.643	2.861	3.084	3.310	3.540		

Table 1: Maximal values θ_m^∞ of $\|2^{-s}A\|$ such that the backward error bound (21) does not exceed $u = 2^{-53}$ for $|\lambda| \to \infty$, i.e. Taylor expansion of matrix exponential.

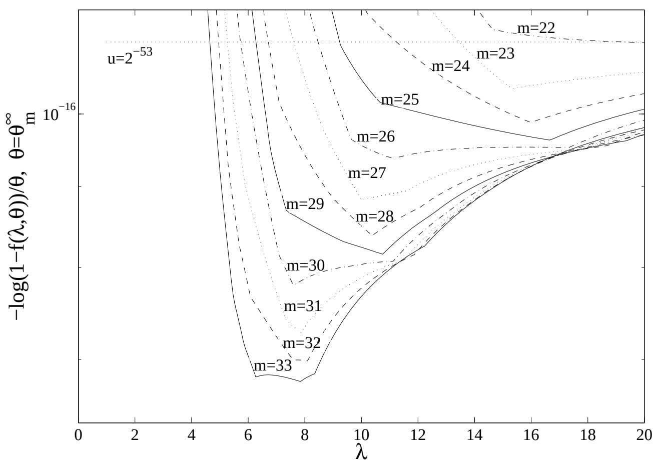


Figure 1: Bound (21) for $m = 22, 23, \dots, 33$ and $\theta = \theta_m^\infty$, vs. parameter λ.

Using MATLAB's Symbolic Math Toolbox we have evaluated $f_m(\lambda, \theta)$ for (18) and (20), in 250 decimal digit arithmetic, summing the first 150 series terms in both expressions, where the coefficients c_j, c'_j and c''_j have been obtained symbolically, c_j, c'_j being functions of parameter λ.

For $m = 1, 2, \dots, 30$ we have used a zero-finder to determine the largest value of θ, denoted by θ_m^∞, such that the backward error bound (21) for $|\lambda| \to \infty$ does not exceed the unit roundoff u in IEEE double precision arithmetic, $u = 2^{-53}$. These values correspond to Taylor approximation of matrix exponential. Table 1 presents the results with 4 significant digits.

Substituting these values in (21) and varying parameter λ, experimen-

m	22	23	24	25	26	27	28	29	30
θ_m	1.816	2.020	2.229	2.441	2.659	2.884	3.113	3.342	3.579
λ	26.904	15.311	15.986	16.661	11.074	10.009	10.381	10.753	7.596

Table 2: Maximal values θ_m of $\|2^{-s}A\|$ such that the backward error bound (21) does not exceed $u = 2^{-53}$ and values of λ for which this is accomplished.

m	$h_m(\lambda, A)$
4	$(F_0^0/4!A^2 + F_0^0/3!A + F_0^1/2I)A^2 + F_0^1 A + F_0^2 I$
6	$((F_0^0/6!A^2 + F_0^0/5!A + F_0^1/4!I)A^2 + F_0^1/3!A + F_0^2/2I)A^2 +$ $F_0^2 A + F_0^3 I$
9	$((F_0^0/9!A^3 + F_0^0/8!A^2 + F_0^1/7!A + F_0^1/6!I)A^3 + F_0^2/5!A^2 +$ $F_0^2/4!A + F_0^3/3!I)A^3 + F_0^3/2A^2 + F_0^4 (A + I)$

Table 3: Evaluation of $h_m(\lambda, A)$, where $F_0^n = e^{\frac{1}{\lambda^2}} E_0^n$. Set $e^{\frac{1}{\lambda^2}} = 1$ and $F_0^n = 1$ to obtain the Taylor approximation.

tally the bound (21) is monotonically decreasing to u for $\lambda > 0$ and $m = 1, 2, \ldots, 21$, and it has minima lower than u for $6 < \lambda < 27$ and $m = 22, 23, \ldots, 30$, see figure 1. So it is possible to determine $\theta_m > \theta_m^\infty$ for $m \geq 22$ searching for those minima. We have used an iterative process and a minimum finder to obtain the new maximum values of θ_m, $m = 22, 23, \ldots, 30$, and the minima of $\lambda > 0$ for which the backward error bound (21) does not exceed u. Table 2 presents the results for the new θ_m values with 4 significant digits. Therefore, we will consider Taylor approximation of matrix exponential for $m \leq 21$ and Hermite approximation for $m \geq 22$.

From pages 73-74 and Table 4.1 of [14, p. 74], using Horner's and Paterson-Stockmeyer's methods [13], we can evaluate a matrix polynomial of degree m

$$P_m(A) = \sum_{k=0}^{m} p_k A^k \,, \tag{22}$$

maximizing the degree of the polynomial for a given number of matrix product evaluations, for $m^* = \{1, 2, 4, 6, 9, 12, 16, 20, 25, 30, 36, \ldots\}$, i.e. for $m = k^2$, $k = 1, 2, 3, \ldots$, and $m = k^2 - k$, $k = 2, 3, \ldots$, using next evaluation formula

$$\begin{aligned}P_m(A) = &\big(\cdots\big(A^q p_m + A^{q-1}p_{m-1} + \cdots + Ap_{m-q+1} + Ip_{m-q}\big)\\ &\times\ A^q + A^{q-1}p_{m-q-1} + A^{q-2}p_{m-q-2} + \cdots + Ap_{m-2q+1} + Ip_{m-2q}\big)\\ &\ \ \cdots\cdots\\ &\times\ A^q + A^{q-1}p_{q-1} + A^{q-2}p_{q-2} + \cdots + Ap_1 + Ip_0,\end{aligned} \tag{23}$$

calculating and saving previously the matrix powers $A^2, A^3, \ldots, A^q$, where one can take $q = \lceil\sqrt{m}\rceil$ or $q = \lfloor\sqrt{m}\rfloor$. The even matrix powers can be calculated as $A^2 = AA$, $A^4 = A^2A^2$, $A^6 = A^4A^2, \ldots$, and the odd matrix powers as $A^{2k+1} = AA^{2k}$, $k = 1, 2, \ldots$ Using (6), (7) and (23), Table 3 presents the evaluation formula of $h_m(\lambda, A)$ for some $m \in m^*$, where we use the coefficients $F_0^n = e^{\frac{1}{\lambda^2}}E_0^n$ and we selected $q = \lfloor\sqrt{m}\rfloor$ to minimize the number of matrix powers in memory. Note that one can obtain Taylor approximation from this table setting $e^{\frac{1}{\lambda^2}} = 1$ and $F_0^n = 1$. On the other hand, given parameter λ, the coefficients of the matrix powers in the approximations $h_m(\lambda, A)$ can be evaluated only once in 250 digit precision arithmetic before being rounded to IEEE double precision arithmetic. Evaluating $h_m(\lambda, A)$ for the rest of values of m in a similar way, we determine the cost of evaluating $h_m(\lambda, A)$, in terms of matrix products. Selecting the optimal scaling parameter as $s = \lceil\log_2 ||A||/\theta_m\rceil$ if $||A|| \geq \theta_m$, and $s = 0$ otherwise, the cost of the algorithm in matrix multiplications is

$$\pi_m + s = \pi_m + \max(\lceil\log_2 ||A|| - \log_2 \theta_m\rceil, 0)\,, \tag{24}$$

where π_m denotes the number of matrix products evaluated to obtain Taylor or Hermite matrix polynomial approximations. It is important to note that the lower the θ_m values are, the larger the number of final squaring steps are necessary. Considering $||A|| \geq \theta_m$ and ignoring the constant shift $||A||$ we have to minimize

$$C_m = \pi_m - \log_2 \theta_m\,, \tag{25}$$

in order to obtain the best choice for m in the Hermite approximation, see [7, p. 1184]. Considering Taylor approximation of matrix exponential for $m < 22$ and therefore $\theta_m = \theta_m^\infty$, $m < 22$, table 4 presents the values C_m and π_m for $m = 1, 2, \ldots, 30$.

From table 4, C_m has an absolute minimum for $m = 16$, and local minima for $m = 9, 12, 20, 25, 30$, which correspond with optimal values in terms of number of matrix products. Thus, considering this analysis the optimal

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
π_m	0	1	2	2	3	3	4	4	4	5	5	5	6	6	6
C_m	52.00	26.21	18.14	13.52	11.70	9.79	9.39	8.32	7.48	7.79	7.22	6.74	7.32	6.96	6.64
m	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
π_m	6	7	7	7	7	8	8	8	8	8	9	9	9	9	9
C_m	6.36	7.10	6.87	6.67	6.48	7.30	7.14	6.99	6.84	6.71	7.59	7.47	7.36	7.26	7.16

Table 4: Number of matrix multiplications, π_m. Measure of overall cost C_m.

Hermite				Padé		
m	θ_m	λ	π_m	m	θ'_m	π'_m
4	3.397168839976962e-4	–	2	3	1.495585217958292e-2	2
6	9.065656407595101e-3	–	3	5	2.539398330063230e-1	3
9	8.957760203223343e-2	–	4	7	9.504178996162932e-1	4
12	2.996158913811581e-1	–	5	9	2.097847961257068	5
16	7.802874256626574e-1	–	6	13	5.371920351148152	6
20	1.438252596804337	–	7			
25	2.441356829252848	16.66121324200387	8			
30	3.578700513755017	7.596210771817034	9			

Table 5: Theoretical optimal values of θ_m for Hermite method (m = order of approximation, π_m = number of matrix products). For $m \leq 20$, $\theta_m = \theta_m^\infty$, i.e. the values relating to Taylor approximation. For $m = 25$, 30, θ_m have been obtained for the Hermite approximation, with the corresponding optimal values of λ. Parameter comparison with Padé `expm` method.

order, meaning with minimal cost, would be $m = 16$. Table 5 presents the corresponding values of θ_m in IEEE double precision arithmetic and a comparison with the same values for Padé method proposed in [7]. From Table 5 one gets that $\theta_{20} < 2\theta_{16}$. Thus, if $m = 20$ and the resulting scaling parameter is $s \geq 1$ then there exist matrices such that $\theta_{20}/2 < ||A/2^s|| \leq \theta_{16}$, and for such matrices one can use order $m = 16$ instead of 20 in the approximation, saving one matrix product. The same occurs with orders $m = 25$ and 20, and $m = 30$ and 25 and we will take this into account in the Hermite based algorithms. We will show in section 4 that with this modification the optimal maximum order is $m = 20$ instead of $m = 16$.

With respect to rounding errors, we rule out $m = 1$ and 2 as maximum orders, as Taylor approximation can suffer from loss of significance in floating point arithmetic for those orders taking into account the values of θ_1^∞ and θ_2^∞ from Table 1, see [7, p. 1184].

The effect of rounding errors on the evaluation of the matrix polynomial $h_m(\lambda, A)$ can be bounded analogously to the numerator of Padé approximants in [7, p. 1185]. Using Theorem 2.2 of [7, p. 1184], taking into account that $\|A\|_1 \leq \theta_m$, $e^{-\|A\|} \leq \|e^A\|$, and noting that $h_m(\lambda, A)$ has all positive coefficients for $\lambda \geq 1$, it follows that

$$\begin{aligned}\left\|h_m(\lambda, A) - \hat{h}_m(\lambda, A)\right\|_1 &\leq \widetilde{\gamma}_{mn} h_m\left(\lambda, \|A\|_1\right) \approx \widetilde{\gamma}_{mn} e^{\|A\|_1} \leq \widetilde{\gamma}_{mn} \left\|e^A\right\|_1 e^{2\|A\|_1} \\ &\simeq \widetilde{\gamma}_{mn} \left\|h_m(\lambda, A)\right\|_1 e^{2\|A\|_1} \leq \widetilde{\gamma}_{mn} \left\|h_m(\lambda, A)\right\|_1 e^{2\theta_m}, \quad (26)\end{aligned}$$

where $A \in \mathbb{C}^{n\times n}$, $\hat{h}_m(\lambda, A)$ is the computed Hermite approximation using explicit formation of matrix powers as in (23), and $\widetilde{\gamma}_k = cku/(1 - cku)$ with c a small integer constant [15]. Hence, the relative error is bounded approximately by $\widetilde{\gamma}_{mn} e^{2\theta_m}$, which is a satisfactory bound taking into account the values of θ_m given in Tables 1 and 2. Analogously, a similar bound can be obtained for Taylor matrix polynomial.

We have implemented two algorithms for computing the matrix exponential by the Hermite method presented in this paper. The first algorithm (`dgeexfher`) computes, for <u>d</u>ouble precision <u>ge</u>neral matrices, the <u>ex</u>ponential <u>f</u>unction by a <u>Her</u>mite approximation using the classical Horner's and Paterson-Stockmeyer methods [13], [16, p. 568], [14, p. 73].

The second algorithm saves computational cost taking into account the relative accuracy bounds in IEEE double precision arithmetic. The underlying idea is that if the contribution of the highest degree terms of the Hermites series to the exponential of the scaled matrix is negligible taking into account floating point arithmetic bounds, then we can save the evaluation of matrix products without substantial changes in the final result. For example, for $m = 9$, let

$$F = p_9 A^3 + p_8 A^2 + p_7 A, \quad (27)$$

where p_i, $i = 0, 1, \cdots, 9$, are the coefficients of the matrix powers A^i of the corresponding Hermite expansion in Table 3. Since

$$\frac{\|p_9 A^9 + p_8 A^8 + p_7 A^7\|}{\|e^A\|} \leq \frac{\|F\| \, \|A^3\|^2}{e^{-\|A\|}}, \quad (28)$$

if

$$\frac{\min\{\|F\|, \|F + p_6 I\|\} \|A^3\|^2}{e^{-\|A\|}} < u, \quad (29)$$

or

$$\|F\| < |p_6|u, \tag{30}$$

then matrix F or matrix $F+p_6I$ can be neglected saving one matrix product. This is likely to occur for instance if the norm of matrix A was only slightly greater than θ_6. Similar tests can be devised and applied recursively, eliminating sets of 3 terms each time. It is important to note that if the condition is accomplished more than once or twice in the evaluation of $h_m(\lambda, A)$ for a scaled matrix A, it might be a sign of overscaling.

With respect to the computational cost of the bound test, if we denote B the original unscaled matrix, then $A = B/2^s$. Therefore, when doing the tests, the norm of the original unscaled matrix and the matrix power A^3 are already obtained. Hence, making the recursive tests only involves in practice to evaluate once at the beginning the norm of A^3, the expression $ue^{-||A||_1} = ue^{-||B||_1/2^s}$, and two matrix norms for each test. The cost of these operations for $r \times r$ matrices is of order $O(r^2)$, negligible when compared to a matrix product, whose cost is of order $O(r^3)$.

Similar tests can be applied to the evaluation of Taylor approximation of orders $m = 4, 6, 9, 16, 20$, and $h_m(\lambda, A)$ with $m = 25, 30$, giving Algorithm 2 (`dgeexfhrp`) which computes, for $\underline{\text{d}}$ouble precision $\underline{\text{ge}}$neral matrices, the $\underline{\text{ex}}$ponential $\underline{\text{f}}$unction by scaling-squaring $\underline{\text{H}}$ermites approximation $\underline{\text{r}}$educing the number of matrix $\underline{\text{p}}$roducts when possible. This algorithm has essentially the same stages as Algorithm 1, checking the tests when needed and potentially saving matrix products. Numerical tests will show that in practically the 100% of the test matrices the savings are obtained with the same accuracy in double precision arithmetic. Both algorithms 1 and 2 do not consider orders $m = 1$ and 2 because $||A||$ should be very small to use those orders, given the low values of θ_1^∞ and θ_2^∞, see Table 1.

Algorithm 1 can be divided into the following stages (algorithm 2 can be divided in analogous stages):

1. Preprocessing of matrix A using the techniques proposed by Ward in [17] (steps 1-4). Note that in numerical tests we did not use preprocessing because turning it on provided similar comparison results to those turning it off.
2. After preprocessing, the optimal value of the scaling parameter s is calculated (steps 5-20).
3. In steps 17 and 35-37 the matrix scaling and the squaring of the approximation is done, respectively.

4. Finally in step 38 the postprocessing is applied. In the same way as preprocessing, this step has not been applied in numerical tests.

We have made available online a MATLAB sequential version of the complete algorithm `dgeexfhrp` at

http://personales.upv.es/∼jorsasma/dgeexfhrp.zip,

which implements the rest of cases $m = 9, 12, \dots, 30$ from lines $31 - 34$ of the algorithm, and offers the possibility to select the maximum order m.

4. Numerical examples

The main objective of this section is to compare MATLAB implementations of the algorithms developed in Section 3 with other efficient algorithms implemented in MATLAB that compute matrix exponential. MATLAB 7.7 (R2008b) implementations were tested on an Intel Core 2 Duo processor at 2.52 GHz with 4 GB main memory. In the comparative the following MATLAB functions were used:

- `Expm` is a MATLAB function that uses Padé approximants of exponential function with scaling and squaring proposed by Higham in [7].
- `Funm` is a built-in MATLAB 7.7 function that enables computation of general matrix functions at square matrices. The matrix function must have a Taylor series with an infinite radius of convergence, except for the matrix logarithm, which is treated as a special case. The exponential, cosine, sine, hyperbolic sine, hyperbolic cosine and the logarithm of a matrix are all allowed. This function implements the Schur-Parlett algorithm of Davies and Higham [18].

Algorithm accuracy was tested by computing the relative error

$$\mathrm{E} = \frac{\|e^A - \tilde{Y}\|_1}{\|e^A\|_1},$$

where $\tilde{Y}$ is the computed solution and e^A the exact solution.

As it is mentioned above, in the tests we did not use any preprocessing/postprocessing in the Hermite implemented algorithms. Analogously to the experiments in [7], we found that turning on preprocessing in this algorithm

Algorithm 1 computes the exponential of a matrix by Hermite series with scaling and squaring and maximum order $m = 30$.

```
    Function $F =$ dgeexfher($A$)
    Input: Matrix $A \in \mathbb{C}^{r\times r}$        Output: Matrix $F \cong e^{A} \in \mathbb{C}^{r\times r}$
 1: $\mu = \mathrm{trace}(A)/r$
 2: $A \leftarrow A - \mu I$
 3: Determine a diagonal matrix $D$ and a permutation matrix $P$ such that $D^{-1}P^{T}APD$
    is balanced
 4: $A \leftarrow D^{-1}P^{T}APD$                                        ▷ Preprocessing of $A$
 5: Initialize $\theta$ with values of Tables 5, and coefficients $p_i$ for each approximation order
    $m = 4, 6, 9, \cdots, 30$.
 6: $normA = ||A||_1$
 7: for $m = [4\ 6\ 9\ 12\ 16\ 20\ 25\ 30]$ do
 8:     if $normA \leq \theta_m$ then
 9:         break
10:     end if
11: end for
12: if $normA > \theta_{30}$ then
13:     $s = \lceil \log_2(normA/\theta_{30}) \rceil$
14:     if $normA/2^s \leq \theta_{25}$ then       ▷ This condition can occur because $\theta_{30}/2 < \theta_{25}$
15:         $m = 25$
16:     end if
17:     $A \leftarrow A/2^s$                              ▷ Scaling Phase: matrix $A$ is scaled
18: else
19:     $s = 0$
20: end if
21: $A_2 = A^2$
22: if $m == 4$ then                   ▷ Taylor approximation for $m =$ 4, 6, 9, 12, 16, 20
23:     $F = (p_4A_2 + p_3A + p_2I)\,A_2 + p_1A + p_0I$
24: else if $m == 6$ then
25:     $F = ((p_6A_2 + p_5A + p_4I)\,A_2 + p_3A + p_2I)\,A_2 + p_1A + p_0I$
26: else if $m == 9$ then
27:     $A_3 = A \cdot A_2$
28:     $F = ((p_9A_3 + p_8A_2 + p_7A + p_6I)\,A_3 + p_5A_2 + p_4A + p_3I)\,A_3 + p_2A_2 + p_1A + p_0I$
29: else if $m == 12$ then
30:     .............................
31:     .............................
32: else if $m == 30$ then
33:     .............................
34: end if
35: for $k = 1 : s$ do                  ▷ Squaring Phase: Repeated squaring of matrix $F$
36:     $F = F^2$
37: end for
38: $F \leftarrow e^{\mu}PDFD^{-1}P^{T}$                                 ▷ Postprocessing of $F$
```

Algorithm 2 computes the exponential of a matrix by scaling and squaring Hermite approach with maximum order $m = 30$ reducing the number of matrix products when possible.

```
    **Function** $F =$ dgeexfhrp$(A)$
    **Input**: Matrix $A \in \mathbb{C}^{r\times r}$        **Output**: Matrix $F \cong e^A \in \mathbb{C}^{r\times r}$
 1: Same as dgeexfher(A) lines 1-21
 2: $u = 2^{-53}$
 3: $LowBound = ue^{-normA/2^s}$
 4: **if** $m == 4$ **then**
 5:     $F = p_4A_2 + p_3A$
 6:     $aux = ||F||_1$
 7:     $F = F + p_2I$
 8:     **if** $(aux \leq |p_2|u)$ or $(\min\{aux, ||F||_1\} \cdot ||A_2||_1 \leq LowBound)$ **then**
 9:         $F = p_2A_2 + p_1A + p_0I$                    ▷ One matrix product is saved
10:     **else**
11:         $F = F \cdot A_2 + p_1A + p_0I$
12:     **end if**
13: **else if** $m == 6$ **then**
14:     $normA_2 = ||A_2||_1$
15:     $F = p_6A_2 + p_5A$
16:     $aux = ||F||_1$
17:     $F = F + p_4I$
18:     **if** $(aux \leq |p_4|u)$ or $(\min\{aux, ||F||_1\} \cdot ||A_2||_1^2 \leq LowBound)$ **then**
19:         $F = p_4A_2 + p_3A$                    ▷ One matrix product is saved
20:     **else**
21:         $F = F \cdot A_2 + p_3A$
22:     **end if**
23:     $aux = ||F||_1$
24:     $F = F + p_2I$
25:     **if** $(aux \leq |p_2|u)$ or $(\min\{aux, ||F||_1\} \cdot ||A_2||_1 \leq LowBound)$ **then**
26:         $F = p_2A_2 + p_1A + p_0I$                    ▷ One matrix product is saved
27:     **else**
28:         $F = F \cdot A_2 + p_1A + p_0I$
29:     **end if**
30: **else if** $m == 9$ **then**
31:     ..............................
32:     ..............................
33: **else if** $m == 30$ **then**
34:     ..............................
35: **end if**
36: Same as dgeexfher(A) lines 35-38
```

provided similar results to those presented in this section without preprocessing.

Regarding memory issues, it is important to note that Algorithm 1 needs the same matrices in memory as `expm` when both methods use their maximum orders, $m = 30$ and $m = 13$ respectively: $A, A^2, \ldots, A^5$ plus one to perform the calculation for Hermite method, and A, A^2, A^4, A^6 plus two for the numerator and denominator for Padé method, taking into account that the final rational approximation can be performed re-using the memory allocated for the power of A involved in the numerator and denominator computation.

Regarding computational cost, from Table 4 and Table 2.3 of [7], Table 5 presents the orders of the approximation, the θ_m and θ'_m values and the number of matrix products π_m and π'_m required for Hermite `dgeexfher` function, and Padé `expm` function, respectively. Note that the number of matrix products for `dgeexfher` is a maximum bound of the matrix products for `dgeexfhrp`. Then, using (24), for matrices with $||A|| > \theta'_{13} = 5.37$ (showing three significant digits), the cost of `dgeexfher` in terms of matrix products, denoted by C^H_m, and representing the maximum cost of `dgeexfhrp`, is

$$C^H_{30} = 9+s_H = 9+\lceil\log_2||A||-\log_2(3.58)\rceil, \text{ if } \frac{||A||}{2^{s_H}} > 2.44, \tag{31}$$

$$C^H_{25} = 8+s_H = 8+\lceil\log_2||A||-\log_2(3.58)\rceil, \text{ if } \frac{||A||}{2^{s_H}} \leq 2.44, \tag{32}$$

where s_H denotes the scaling in Hermite methods. On the other hand, the cost of `expm` Padé method, denoted by C^P_m, is

$$C^P_{13} = 6 + C_{LS} + s_P = 6 + C_{LS} + \lceil\log_2||A|| - \log_2(5.37)\rceil\,, \tag{33}$$

where

$$s_P = \lceil\log_2||A|| - \log_2(5.37)\rceil, \tag{34}$$

denotes the scaling in `expm` and C_{LS} denotes the cost of solving the multiple right-hand sides linear system in Padé method, in terms of matrix products. From [19] the cost of the matrix product in $\mathbb{R}^{r\times r}$ and the solution of the multiple right-hand sides of the same size with Padé approximants is $2r^3 - r^2$ and $\frac{8r^3}{3} - \frac{r^2}{2} + \frac{5r}{6}$ flops, respectively. Therefore, asymptotically $C_{LS} \approx 4/3$.

Taking into account that θ_{30} for Hermite methods is greater than $\theta'_{13}/2$ for Padé method [7, p. 1186], taking `expm`'s matrix scaling by 2^{s_P}, for matrices with

$$\theta'_{13}/2 = 2.68 < \frac{||A||}{2^{s_P}} \leq 3.58, \tag{35}$$

Hermite methods use $m = 30$ and $s_H = s_P$, therefore `dgeexfhrp` needs a maximum of $3 - C_{LS}$ more matrix products than `expm`, which results in a maximum relative higher cost of $(1 + 2/3)/(7 + 1/3 + s_P) \times 100\% \leq 20\%$ in matrix products, which decreases with the matrix norm because of the increasing scaling. For matrices such that

$$3.58 < \frac{||A||}{2^{s_P}} \leq 2 \times \theta_{25} = 4.88, \tag{36}$$

Hermite methods use $m = 25$ and $s_H = s_P + 1$, therefore `dgeexfhrp` needs a maximum of $3 - C_{LS}$ more matrix products than `expm` again, resulting in the same relative higher cost $(1 + 2/3)/(7 + 1/3 + s_P) \times 100\% \leq 20\%$ in matrix products, decreasing with the matrix norm. In fact, using (35) and (36), for instance for matrices with norm

$$343.80 < ||A|| \leq 624.98, \tag{37}$$

it follows that $s_P = 7$ and the maximum relative higher cost of `dgeexfhrp` decreases to 11.63%. Finally, for matrices such that

$$4.88 < \frac{||A||}{2^{s_P}} \leq 5.37, \tag{38}$$

Hermite methods use $m = 30$ and $s_H = s_P + 1$, therefore `dgeexfhrp` needs a maximum of $4 - C_{LS}$ more matrix products than `expm`, resulting in a maximum relative higher cost of $(2 + 2/3)/(7 + 1/3 + s_P) \times 100\% \leq 32\%$ in matrix products. Note that this norm interval represents only the 18.21% of the total interval considered in the three cases (35), (36) and (38).

The cost comparison for matrices with $||A|| \leq 5.37$ is presented in Table 6, where the θ_m values are presented with three significant digits, and C_m^H represents a bound on the maximum cost of `dgeexfhrp`. Note that there are some cases where the maximum cost for `dgeexfhrp` is lower than the cost for `expm`, i.e. $2.53e-1 < ||A|| \leq 2.99e-1$, $1.49e-2 < ||A|| \leq 8.95e-2$ and $||A|| \leq 9.06e-3$, reaching relative efficiency gains from 6.25% up to 40%. For matrices satisfying $4.88 \leq ||A|| \leq 5.37$ the maximum cost of `dgeexfhrp` is 36.36% higher and in the rest of cases `dgeexfhrp` maximum cost exceeds `expm` cost in 2/3 or 1+2/3 matrix products. Once again the interval where the cost difference is higher is a small part of all the interval considered, representing only the 9.11% of the total. One more final squaring step than in `expm` is required for matrices satisfying $3.58 < ||A||/2^{s_P} \leq 5.37$ with $||A|| > 5.37$, and $3.58 < ||A|| \leq 5.37$. This might be a possible source of error, especially if A

	`dgeexfhrp`			`expm`			
Norm ranges	m	s_H	C_m^H	m	s_P	C_m^P	$\frac{C_m^H - C_m^P}{C_m^P}$ %
$\|\|A\|\| \leq 3.39e-4$	4	0	2	3	0	2+C_{LS}	-40
$3.39e-4 < \|\|A\|\| \leq 9.06e-3$	6	0	3	3	0	2+C_{LS}	-10
$9.06e-3 < \|\|A\|\| \leq 1.49e-2$	9	0	4	3	0	2+C_{LS}	20
$1.49e-2 < \|\|A\|\| \leq 8.95e-2$	9	0	4	5	0	3+C_{LS}	-7.69
$8.95e-2 < \|\|A\|\| \leq 2.53e-1$	12	0	5	5	0	3+C_{LS}	15.38
$2.53e-1 < \|\|A\|\| \leq 2.99e-1$	12	0	5	7	0	4+C_{LS}	-6.25
$2.99e-1 < \|\|A\|\| \leq 7.80e-1$	16	0	6	7	0	4+C_{LS}	12.50
$7.80e-1 < \|\|A\|\| \leq 9.50e-1$	20	0	7	7	0	4+C_{LS}	31.25
$9.50e-1 < \|\|A\|\| \leq 1.43$	20	0	7	9	0	5+C_{LS}	10.53
$1.43 < \|\|A\|\| \leq 2.09$	25	0	8	9	0	5+C_{LS}	26.32
$2.09 < \|\|A\|\| \leq 2.44$	25	0	8	13	0	6+C_{LS}	9.09
$2.44 < \|\|A\|\| \leq 3.58$	30	0	9	13	0	6+C_{LS}	22.73
$3.58 < \|\|A\|\| \leq 2 \times 2.44$	25	1	9	13	0	6+C_{LS}	22.73
$2 \times 2.44 < \|\|A\|\| \leq 5.37$	30	1	10	13	0	6+C_{LS}	36.36

Table 6: Comparison of maximum theoretical cost C_m^H in terms of matrix products for `dgeexfhrp` with maximum order $m = 30$, and cost C_m^P for `expm`, for $||A|| \leq 5.37$.

is ill-conditioned. However, Hermite methods with maximum order $m = 30$ obtained better accuracy than `expm` in a high percentage of cases in numerical tests (see Table 8).

In a similar way, it is easy to show that, for any matrix $A \in \mathbb{C}^{r\times r}$, the maximum cost of using `dgeexfhrp` with maximum order $m = 16$ is the same as using $m = 20$. Thus, maximum order $m = 20$ should be used instead of 16 because taking into account that $\theta_{20} > \theta_{16}$, the scaling parameter s with $m = 20$ is lower in the majority of matrix norm intervals, and the squaring process might be a possible source of error. Analogously, it is also easy to check that if we use maximum orders $m = 20$ or 25 then `dgeexfhrp` presents a maximum higher cost than `expm` of $1 + 2/3$ matrix multiplications, instead of $2 + 2/3$ that `dgeexfhrp` presented with $m = 30$. On the other hand, `dgeexfhrp` with maximum orders $m = 20$ and 25 presents lower maximum cost than `expm` for the same matrix norms as `dgeexfhrp` with $m = 30$.

It is important to note that all the costs of `dgeexfhrp` presented in this analysis are maximum costs and, as we will see in tests, they may decrease considerably in practice.

For tests, 105 matrices were used: 49 matrices from the Matrix Computation Toolbox [20], 24 matrices from the Eigtool MATLAB package [21], 18

Table 7: Relative error comparison (%) between `dgeexfher` and `dgeexfhrp`.

maximum order	m=16	m=20	m=25	m=30
$E_{\texttt{dgeexfher}} < E_{\texttt{dgeexfhrp}}$	3.77	0	0	0
$E_{\texttt{dgeexfher}} = E_{\texttt{dgeexfhrp}}$	95.28	99.06	100.00	100.00
$E_{\texttt{dgeexfher}} > E_{\texttt{dgeexfhrp}}$	0.94	0.94	0.00	0.00
$\mathrm{P}_{\texttt{dgeexfher}}/\mathrm{P}_{\texttt{dgeexfhrp}}$	105.52	108.43	108.58	109.91

matrices from papers of the state-of-the-art of matrix functions [6, 17, 18, 22, 23, 24, 25, 26], and 14 special matrices such as matrices of Vandermonde, Hankel, Toeplitz, Wilkinson, symmetric matrices, defective matrices and non defective matrices.

In the examples the matrix exponentials were calculated analytically, when it was possible, or by using [33/33] diagonal Padé method with scaling and squaring with 1000-digit precision in an iterative way: different increasing scalings starting from that provided in [7] for `expm` were used, until the norm of the relative difference between the approximations converted to IEEE double precision arithmetic was zero in four iterations. The [33/33] diagonal Padé approximation was evaluated with matrix power aggregation similar to that proposed in [7, p. 1183].

Table 7 shows a comparative between the implementations `dgeexfher` and `dgeexfhrp`. The three first rows contain the percentage of times that relative error of a function is lower, equal or greater than relative error of the other function. The last row of Table 7 contains the ratio of the total number of matrix products evaluated for the two functions over all test matrices, denoted by $\mathrm{P}_{\texttt{dgeexfher}}$ and $\mathrm{P}_{\texttt{dgeexfhrp}}$. Both functions presented practically the same accuracy, however the total number of matrix products evaluated for `dgeexfhrp` is lower than those for `dgeexfher`.

Table 8 presents the relative error and matrix product comparison of `dgeexfhrp` and `dgeexfher` with `expm`. `dgeexfher` obtained the same comparative results of accuracy as `dgeexfhrp` at a higher cost, see the last two rows. `dgeexfhrp` relative error is lower than `expm` relative error for $m \geq 20$ (68.87%-77.36%), with a slightly higher cost, varying the ratio of matrix products between 103.76% and 104.43% for $m = 20, 25, 30$.

In Table 9 the relative errors of `dgeexfhrp` and `dgeexfher` are compared to `funm`. As shown in these tables, once again `dgeexfher` and `dgeexfhrp` obtained the same comparative results of accuracy, and in general their relative

Table 8: Comparison of relative error and total number of matrix products (%) between `dgeexfhrp` and `expm`. The error comparison results were exactly the same with `dgeexfher` at a greater cost (see last table row).

maximum order	m=16	m=20	m=25	m=30
$\mathrm{E}_{\texttt{dgeexfhrp}} < \mathrm{E}_{\texttt{expm}}$	44.34	68.87	77.36	77.36
$\mathrm{E}_{\texttt{dgeexfhrp}} = \mathrm{E}_{\texttt{expm}}$	1.89	2.83	1.89	1.89
$\mathrm{E}_{\texttt{dgeexfhrp}} > \mathrm{E}_{\texttt{expm}}$	53.77	28.30	20.75	20.75
$\mathrm{P}_{\texttt{dgeexfhrp}}/\mathrm{P}_{\texttt{expm}}$	106.62	103.76	104.01	104.43
$\mathrm{P}_{\texttt{dgeexfher}}/\mathrm{P}_{\texttt{expm}}$	112.51	112.51	112.93	114.78

Table 9: Relative error comparison between `dgeexfhrp` and `funm`. The results were exactly the same with `dgeexfher`.

maximum order	m=16	m=20	m=25	m=30
$\mathrm{E}_{\texttt{dgeexfhrp}} < \mathrm{E}_{\texttt{funm}}$	71.70	77.36	79.25	80.19
$\mathrm{E}_{\texttt{dgeexfhrp}} = \mathrm{E}_{\texttt{funm}}$	0.94	0	0	0
$\mathrm{E}_{\texttt{dgeexfhrp}} > \mathrm{E}_{\texttt{funm}}$	28.30	22.64	20.75	19.81

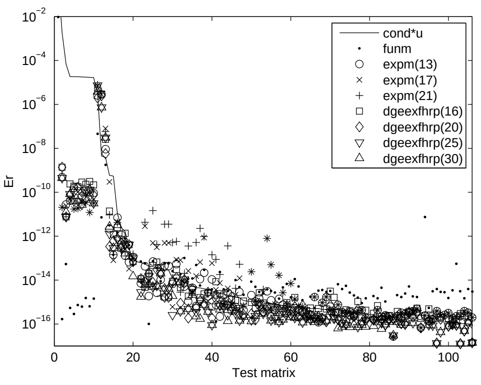
errors were lower than the relative errors of `funm` (71.70%-80.19%).

Figure 2a presents the normwise relative errors of `dgeexfhrp`, `expm` and `funm` (a similar figure is obtained when `dgeexfher` is used). This figure shows the relative error of all implementations sorted in decreasing order, and a solid line that represents the unit roundoff multiplied by the relative condition number of the exponential function at X [14, p. 56],

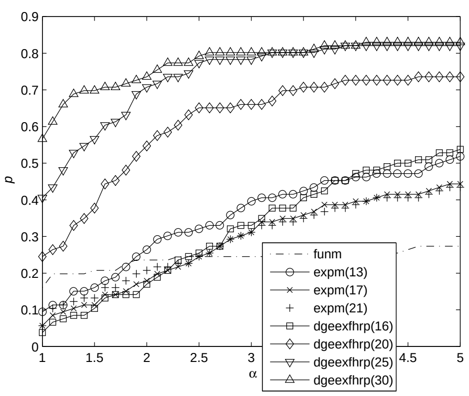
$$\mathrm{cond}_{\exp}(X) = \lim_{\varepsilon\to 0} \sup_{\|E\|\le\varepsilon} \frac{\left\|e^{X+E} - e^{X}\right\|}{\varepsilon} \frac{\|X\|}{\|e^{X}\|}.$$

Relative condition number was computed using the MATLAB function `expm_cond`. This function is incorporated into the Matrix Function Toolbox developed by Higham [14, Appendix D] and available at http://www.ma.man.ac.uk/~higham/mftoolbox.

For a method to perform in a backward and forward stable manner, its error should lie not far above this line on the graph [7, p. 1188]. Figure 2a shows that all functions perform in a numerically stable way on this test, even for matrices 64-70 where there were overscaling problems [26].



(a) Normwise relative errors



(b) Performance profile

Figure 2: Comparative of `dgeexfhrp`, `expm` and `funm`.

Performance profile [27] is presented in Figure 2b. This figure shows the performances of the compared functions, where α coordinate varies between 1 and 5 in steps equal to 0.1, and p coordinate is the probability that the considered function has a relative error lower or equal than α-times the smallest error over all the methods. The probabilities are defined over all matrices considered in the tests. As shown in this figure, `dgeexfhrp` with maximum order $m = 30$ is the most accurate function, and it was achieved with very similar cost to `dgeexfhrp` with maximum orders $m = 16, 20$ or 25. Hence, we consider $m = 30$ as the best choice of maximum order for `dgeexfhrp`.

5. Conclusions

In this work an efficient method to approximate the matrix exponential based on Hermite matrix polynomial expansions has been presented. Following the ideas of [7] we have developed an optimal backward error bound for the scaling and squaring Hermite method in exact arithmetic, which depends on A only through $\|A\|$ and enables to obtain the theoretical optimal scaling for general matrices. The optimal parameter λ of the algorithm has been obtained for each order $m = 21, 22, \dots, 30$, providing greater values of the Hermite scaling parameter θ_m than Taylor methods for those orders, i.e. $\theta_m > \theta_m^\infty$, $m = 21, 22, \dots, 30$, where θ_m^∞ is the scaling parameter for Taylor methods, see Tables 1 and 2. Based on that result, two mixed Hermite-Taylor algorithms have been developed in order to evaluate different order matrix polynomial approximations: a Hermite-Taylor series Paterson-Stockmeyer algorithm, `dgeexfher`, and a modified algorithm, `dgeexfhrp`, which taking into account floating point arithmetic error bounds may reduce the number of matrix product evaluations. This modification (`dgeexfhrp`) presented practically the same accuracy as `dgeexfher` in numerical tests with a lower computational cost of up to 9.91%. Both algorithms use Hermite series for order $m > 20$ and Taylor series for order $m \leq 20$. From experimental results, we have identified the best choice of maximum degree m of Hermite approximation in terms of efficiency and accuracy: $m = 30$.

We have shown that `dgeexfhrp` with maximum order $m = 30$ has lower theoretical maximum cost than `expm` for some matrix norm intervals, and in numerical tests the cost of `dgeexfhrp` was similar to that for `expm`.

`dgeexfhrp` stores the same number of matrices in memory as `expm` when both functions use their maximum orders, $m = 30$ and $m = 13$ respectively, and does not need the solution of multiple linear systems. [7] shows that this solution does not introduce large errors in `expm`. However, `dgeexfhrp` obtained higher accuracy than both `funm` and `expm` in the majority of test matrices, i.e. the 80.19% and 77.36% respectively. These results are based on empirical observations. However, numerical results are promising and further research on Hermite matrix polynomial series for the matrix exponential is being carried out to reduce the computational costs.

6. Acknowledgements

This work has been supported by the Programa de Apoyo a la Investigación y el Desarrollo of the Universidad Politécnica de Valencia PAID-05-09-4338, 2009.

References

[1] R. A. Frazer, W. J. Duncan, A. R. Collar, Elementary Matrix and Some Applications to Dynamics and Differential Equations, Macmillan, New York, 1946.

[2] M. W. Hirsch, S. Smale, Differential Equations, Dynamical Systems and Linear Algebra, Academic Press, New York, 1974.

[3] T. Kailath, Linear Systems, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[4] C. B. Moler, C. V. Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Review 20 (4) (1978) 801–836.

[5] G. D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, 3^{rd} Edition, Oxford University Press, 1985.

[6] 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.

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

[8] 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.

[9] N. Dunford, J. Schwartz, Linear Operators, Part I., Interscience Publishers, New York, 1957.

[10] S. Saks, A. Zygmund, Analytic Functions, Elsevier Science Publishers, Amsterdam, The Netherlands, 1971.

[11] E. Defez, L. Jódar, Some applications of Hermite matrix polynomials series expansions, J. Comput. Appl. Math. 99 (1998) 105–117.

[12] L. Jódar, R. Company, Hermite matrix polynomials and second order matrix differential equations, J. Approximation Theory Appl. 12 (2) (1996) 20–30.

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

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

[15] N. J. Higham, Accuracy and Stability of Numerical Algorithms, 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[16] G. H. Golub, C. V. Loan, Matrix Computations, 3rd Ed., Johns Hopkins Studies in Math. Sci., The Johns Hopkins University Press, 1996.

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

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

[19] S. Blackford, J. Dongarra, Installation guide for LAPACK, LAPACK Working Note 411, Department of Computer Science University of Tenessee (1999).

[20] N. J. Higham, The Test Matrix Toolbox for MATLAB, Numerical Analysis Report No. 237, Manchester Centre for Computational Mathematics, Manchester, England (Dec. 1993).

[21] T. G. Wright, Eigtool, version 2.1, (2009) http://web.comlab.ox.ac.uk/pseudospectra/eigtool/

[22] I. Najfeld, T. F. Havel, Derivatives of the matrix exponential and their computation, Advances in Appl. Math. 16 (1995) 321–375.

[23] C. S. Kenney, A. J. Laub, A Schur–Fréchet algorithm for computing the logarithm and exponential of a matrix, SIAM J. Matrix Anal. Appl. 19 (3) (1998) 640–663.

[24] D. Westreich, A practical method for computing the exponential of a matrix and its integral, Communications in Appl. Numer. Methods 6 (1990) 375–380.

[25] Y. Y. Lu, Computing a matrix function for exponential integrators, J. Comput. Appl. Math. 161 (2003) 203–216.

[26] L. Dieci, A. Papini, Padé approximation for the exponential of a block triangular matrix, Linear Algebra Appl. 308 (2000) 183–202.

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