An improved algorithm to compute the exponential of a matrix

Philipp Bader† Sergio Blanes‡ Fernando Casas‡

October 31, 2017

Abstract

In this work, we present a new way to compute the Taylor polynomial of the matrix exponential which reduces the number of matrix multiplications in comparison with the de-facto standard Patterson–Stockmeyer method. This reduction is sufficient to make the method superior in performance to Padé approximants by 10-30% over a range of values for the matrix norms and thus we propose its replacement in standard software kits. Numerical experiments show the performance of the method and illustrate its stability.

Keywords: exponential of matrices, scaling, squaring, matrix polynomials.

1 Introduction

The search for efficient algorithms to compute the exponential of an square matrix has a tremendous interest, given the wide range of its applications in many branches of science. Its importance is clearly revealed by the great impact achieved by various reference reviews devoted to the subject, e.g. [8, 15, 16, 22]. The problem has been tackled of course by many authors and a wide variety of techniques have been proposed. Among them, the scaling and squaring procedure is perhaps the most popular when the dimension of the corresponding matrix runs well into the hundreds. As a matter of fact, this technique is incorporated in popular computing packages such as MATLAB (expm) and MATHEMATICA (MatrixExp) in combination with Padé approximants [2, 8, 10].

Specifically, for a given matrix $A \in \mathbb{C}^{N \times N}$, the scaling and squaring technique is based on the key property

$$e^{A} = \left(e^{A/2^{s}}\right)^{2^{s}}, \quad s \in \mathbb{N}. \quad (1)$$
The exponential $e^{A/2^s}$ is then replaced by a rational approximation, namely a $2m$th-order diagonal Padé approximant, $r_{2m}(A/2^s)$. The optimal choice of both parameters, $s$ and $2m$, depends on the estimation of a bound of $\|A\|$ and the desired accuracy. Diagonal Padé approximants have the form

$$r_{2m}(A) = \frac{p_m(A)}{p_m(-A)},$$

where the polynomial $p_m(x)$ is given by

$$p_m(x) = \sum_{j=0}^{m} \frac{(2m-j)!m!}{(2m)!(m-j)!} x^j$$

so that $r_{2m}(A) = e^A + O(A^{2m+1})$ and the evaluation of $p_m(A), p_m(-A)$ is carried out with a reduced number of matrix products. Finally, the cost of the inverse is taken as $4/3$ the cost of one product: it requires one $LU$ factorization at the cost of $1/3$ products plus $N$ solutions of upper and lower triangular systems by forward and backward substitution at the cost of one product. In particular, $r_{10}(A)$ and $r_{26}(A)$ are considered among the optimal choices (with respect to accuracy and computational cost) in this approach when high accuracy is desired and the norm $\|A\|$ is relatively large. The corresponding algorithm for $r_{10}(A)$ is given by [8]

$$u_5 = A(b_5A_4 + b_3A_2 + b_1I),$$
$$v_5 = b_4A_4 + b_2A_2 + b_0I,$$
$$(-u_5 + v_5) r_{10}(A) = u_5 + v_5,$$

whereas for $r_{26}(A)$ one has

$$u_{13} = A\left(A_6(b_{13}A_6 + b_{11}A_4 + b_9A_2) + b_7A_6 + b_5A_4 + b_3A_2 + b_1I\right),$$
$$v_{13} = A\left(A_{12}A_4 + b_{10}A_4 + b_8A_2\right) + b_6A_6 + b_4A_4 + b_2A_2 + b_0I,$$
$$(-u_{13} + v_{13}) r_{26}(A) = u_{13} + v_{13}.$$  

Here $A_2 = A^2$, $A_4 = A_2^2$ and $A_6 = A_2A_4$. Written in this form, it is clear that only three and six matrix multiplications and one inversion are required to obtain approximations of order 10 and 26, respectively. These two methods are used in fact in the current implementation of the function \texttt{expm} in \textsc{Matlab}.

The choice of the optimal order of the approximation for a given value of $\|A\|$ is based on the control of the backward error. To be more specific, given a prescribed accuracy $u$, the parameter $m$ is selected according to $\theta_{2m}(u)$, the largest value of $\|A\|$ such that the Padé scheme $r_{2m}$ has relative backward-error smaller than $u$, i.e.,

$$\forall A, \quad \|A\| \leq \theta_{2m}(u) \quad : \quad r_{2m}(A) = e^{A+E}, \quad \text{s.t.} \quad \frac{\|E\|}{\|A\|} \leq u.$$  

The values of $\theta_{2m}$ for $u = 2^{-24}$ and $u = 2^{-53}$, corresponding to single and double precision, respectively, are collected in Table[1]. They have been obtained
Table 1: Values of $\theta_{2m}$ for diagonal Padé of order $2m$ with the minimum number of products $\pi$ for single and double precision.

| $\pi$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|-------|---|---|---|---|---|---|---|
| $2m$  | 2 | 4 | 6 | 10| 14| 18| 26|
| $u \leq 2^{-24}$ | 8.46e-4 | 8.09e-2 | 4.26e-1 | 1.88 | 3.93 | 6.25 | 11.2 |
| $u \leq 2^{-53}$ | 3.65e-8 | 5.32e-4 | 1.50e-2 | 2.54e-1 | 9.50e-1 | 2.10 | 5.37 |

according to [8, Theorem 2.1] by defining the majorant of

$$\rho(\theta) = \log(e^{-\theta R_{2m}}) = \sum_{i=2m}^{\infty} c_i \theta^i$$

to $f(\theta) = \sum_{i=2m}^{\infty} |c_i| \theta^i$ and solving $f(\theta)/\theta = u$, where the series is truncated after 150 terms.

Another technique that has recently been proposed consists of using, instead of a diagonal Padé approximant, the Taylor polynomial of degree $m$, $T_m(A) = \sum_{k=0}^{m} A^k/k!$, evaluated according with the Paterson–Stockmeyer procedure (see [9, 10, 17]). In that way the number of matrix products is reduced and the overall performance is improved for matrices of small norm, although it is less efficient for matrices with large norm [10, 18, 19, 20, 21].

If the Paterson–Stockmeyer (PS) technique is carried out to compute $T_m(A)$ in a Horner-like fashion, the maximal attainable degree is $m = (k + 1)^2$ by using $2k$ matrix products. The optimal choices for most cases then correspond to $k = 2$ (four products) and $k = 3$ (six products), i.e, to degree $m = 9$ and $m = 16$, respectively. The corresponding polynomials are then computed as follows:

$$T_9(A) = \sum_{i=0}^{9} c_i A^i = f_0 + (f_1 + (f_2 + c_3 A_3)A_3)A_3,$$

$$T_{16}(A) = \sum_{i=0}^{16} c_i A^i = g_0 + (g_1 + (g_2 + (g_3 + c_{16} A_4)A_4)A_4)A_4,$$

where $c_i = 1/i!$ and

$$f_i = \sum_{k=0}^{2} c_{3i+k} A_k, \quad i = 0, 1, 2$$

$$g_i = \sum_{k=0}^{3} c_{4i+k} A_k, \quad i = 0, 1, 2, 3.$$

Here, as before, $A_2 = A^2$, $A_3 = A^2A$, $A_4 = A_2 A_2$. In Table 2 we collect the values for the corresponding thresholds $\theta_{m}(u)$ needed to select the best scheme for a given accuracy.

The main goal of this work is to show that it is indeed possible to organize the computation of the Taylor polynomial of the matrix exponential function.
Table 2: Values of $\theta_m$ for Taylor of degree $m$ the with minimum number of products $\pi$ for single and double precision. We have included degree 24 to illustrate that the gain is marginal over scaling and squaring for double precision and negative for single precision. The $\theta_m$ values should at least double per extra product to be favorable over scaling and squaring.

| $\pi$ | 0 | 1 | 2 | 4 | 8 | 12 | 18 | 24 |
|-------|---|---|---|---|---|----|----|----|
| $m$   |   |   |   |   |   |    |    |    |
| $u \leq 2^{-24}$ | 1.19E-7 | 5.98E-4 | 5.12E-2 | 5.80E-1 | 1.46 | 3.01 | 4.65 |   |
| $u \leq 2^{-53}$ | 2.22E-16 | 2.58E-8 | 3.40E-4 | 4.99E-2 | 2.99E-1 | 1.09 | 2.22 |   |

in a more efficient way than the Paterson–Stockmeyer technique, so that with the same number of matrix products one can construct a polynomial of higher degree. In this way, a method for computing $e^A$ based on scaling and squaring together with the new implementation of the Taylor polynomial is proposed which is more efficient than the same procedure based on Padé approximants for a wide range of values of $\|A\|$.

2 A generalized recursive algorithm

Clearly, the most economic way to construct polynomials of degree $2^k$ is by applying the following sequence, which involves only $k$ products:

$$
A_1 = A \\
A_2 = (\delta_1 I + x_1 A_1)(\delta_2 I + x_2 A_1) \\
A_4 = (\delta_3 I + x_3 A_1 + x_4 A_2)(\delta_4 I + x_5 A_1 + x_6 A_2) \\
A_8 = (\delta_5 I + x_7 A_1 + x_8 A_2 + x_9 A_4)(\delta_6 I + x_{10} A_1 + x_{11} A_2 + x_{12} A_4),
$$

(7)

with $\delta_i = 0, 1$. These polynomials are then linearly combined to form

$$
T_{2^k} = y_0 I + y_1 A_1 + y_2 A_2 + y_3 A_4 + y_4 A_8 + \cdots + y_{k+1} A_{2^k}.
$$

Notice that the indices in $A$, $A_{2^k}$, are chosen to indicate the highest attainable power, i.e., $A_{2^k} = O(A^{2^k})$. A simple counting tells us that with $k$ products one has $(k + 1)^2 + 1$ parameters to construct a polynomial of degree $2^k$ that contains $2^k + 1$ coefficients. It is then clear that the number of coefficients grows faster than the number of parameters, so that this procedure cannot be used to obtain high degree polynomials, as already noticed in [17]. Even worse, in general, not all parameters are independent and this simple estimate does not suffice to guarantee the existence of solutions with real coefficients.

Nevertheless, this procedure can be modified in such a way that additional parameters are introduced, at the price, of course, of including some extra products. In particular, we could include new terms of the form

$$(\gamma_1 I + z_1 A_1)(\gamma_2 I + z_2 A_1 + z_3 A_2),$$

not only in the previous $A_k$, $k > 1$, but also in $T_{2k}$, which would allow us to introduce a cubic term and an additional parameter.

Although the Paterson-Stockmeyer (PS) technique is arguably the most efficient procedure to evaluate a general polynomial, there are relevant classes of polynomials for which the PS rule involves more products than strictly necessary. To illustrate this feature, let us consider the evaluation of

$$\Psi(k, A) = I + A + \cdots + A^{k-1},$$

a problem addressed in [24]. Polynomial (8) appears in connection with the integral of the state transition matrix and the analysis of multirate sampled data systems. In [24] it is shown that with three matrix products one can evaluate $\Psi(7, A)$ (as with the PS rule), whereas with four products it is possible to compute $\Psi(11, A)$ (one degree higher than using the PS rule). In general, the savings with respect to the PS technique grow with the degree $k$. The procedure was further improved and analysed in [13], where the following conjecture was posed: the minimum number of products to evaluate $\Psi(k, A)$ is $2\lfloor \log_2 k \rfloor - 2 + i_j - 1$ where $N = (i_j, i_{j-1}, \ldots, i_1, i_0)_2$ (written in binary), i.e., $i_{j-1}$ is the second most significant bit.

This conjecture is not true in general, as is illustrated by the following algorithm of type (7), that allows one to compute $\Psi(9, A)$ by using only three matrix products:

$$\begin{align*}
A_2 &= A^2, \\
B &= x_1 I + x_2 A + x_3 A^2 \\
A_4 &= x_4 I + x_5 A + B^2 \\
A_8 &= x_6 A_2 + A_4^2 \\
\Psi(9, A) &= x_7 I + x_8 A + x_9 A_2 + A_8,
\end{align*}$$

with

$$\begin{align*}
x_1 &= \frac{-5 + 6\sqrt{7}}{32}, \\
x_2 &= -\frac{1}{4}, \\
x_3 &= -1, \\
x_4 &= \frac{3(169 + 20\sqrt{7})}{1024}, \\
x_5 &= \frac{3(5 + 2\sqrt{7})}{64}, \\
x_6 &= \frac{1695}{4096}, \\
x_7 &= \frac{267}{512}, \\
x_8 &= \frac{21}{64}, \\
x_9 &= \frac{3\sqrt{7}}{4}.
\end{align*}$$

Notice that, since the coefficients of the algorithm must satisfy a system of nonlinear equations, they are irrational numbers.

Although by following this approach it is not possible to achieve degree 16 with four products, there are other polynomials of degree 16 that can indeed be computed with only four products. This is the case, in particular, of the truncated Taylor expansion of the function $\cos(A)$:

$$T_{16} = \sum_{i=0}^{8} \frac{(-1)^i A^{2i}}{(2i)!} = \cos(A) + O(A^{17}).$$

Taking $B = A^2$ we obtain a polynomial of degree 8 in $B$ that can be evaluated with three additional products in a similar way as in the computation of $\Psi(9, A)$, but with different coefficients.
A new algorithm to compute the matrix exponential

Algorithm (7) can be conveniently modified along the lines exposed in the previous section to compute the truncated Taylor expansion of the matrix exponential function

\[ T_n(A) = \sum_{i=0}^{n} \frac{A^i}{i!} = e^A + O(A^{n+1}) \]  

for different values of \( n \) using the minimum number of products. In practice, we proceed in the reverse order: given a number \( k \), we find a convenient (modified) sequence of type (7) that allows one to construct the highest degree polynomial \( T_n(A) \) using only \( k \) matrix products. The coefficients in the sequence satisfy a relatively large system of algebraic nonlinear equations. Here several possibilities may occur: (i) the system has no solution; (ii) there is a finite number of real and/or complex solutions, or (iii) there are families of solutions depending on parameters. For our purposes, we only need one solution with real coefficients. In addition, if there are several solutions we take a solution with small coefficients to avoid large round off errors due to products of large and small numbers.

With \( k = 0, 1, 2 \) products we can evaluate \( T_n \) for \( n = 1, 2, 4 \), in a similar way as the PS rule, whereas for \( k = 3, 4, 5 \) and 6 products the situation is detailed next.

\[ \begin{align*}
A_2 &= A^2 \\
A_4 &= A_2(x_1A + x_2A_2) \\
A_8 &= (x_3A_2 + A_4)(x_4I + x_5A + x_6A_2 + x_7A_4) \\
T_8(A) &= y_0I + y_1A + y_2A_2 + A_8.
\end{align*} \]  

(11)

With this sequence we get two families of solutions depending on a free parameter, \( x_3 \), which is chosen to (approximately) minimize the 1-norm of the vector of parameters. The reasoning behind this approach is to achieve numerical stability by avoiding the multiplication of high powers of \( A \) with large coefficients, in a similar vein as the Horner procedure. The coefficients in (11) are given by

\[ \begin{align*}
x_1 &= x_3 \frac{1 + \sqrt{177}}{88}, & x_2 &= \frac{1 + \sqrt{177}}{352}x_3, & x_4 &= \frac{-271 + 29\sqrt{177}}{315x_3}, \\
x_5 &= \frac{11(-1 + \sqrt{177})}{1260x_3}, & x_6 &= \frac{11(-9 + \sqrt{177})}{5040x_3}, & x_7 &= \frac{89 - \sqrt{177}}{5040x_3^2}, \\
y_0 &= 1, & y_1 &= 1, & y_2 &= \frac{857 - 58\sqrt{177}}{630}, \\
x_3 &= \frac{2}{3}.
\end{align*} \]

Perhaps surprisingly, \( T_7(A) \) requires at least 4 products, so \( T_8 \) may be considered as a singular polynomial.
$k = 4$ products Although polynomials up to degree 16 can in principle be constructed, our analysis suggests that the Taylor polynomial (10) corresponding to $\exp(A)$ does not belong to that family. To determine the highest degree one can achieve with $k = 4$, we proceed as follows. We take $T_n(A)$ for a given value of $n$ and decompose it as a product of two polynomials of lower degree plus a lower degree polynomial (that will be used to evaluate the higher degree polynomials). The highest value we have managed to reach is $n = 12$. It is important to note that our ansatz gives many different ways to write the sought polynomial, one of which is given by

$$
A_2 = A^2, \\
A_3 = A_2 A, \\
B_1 = a_{0,1} I + a_{1,1} A + a_{2,1} A^2 + a_{3,1} A^3, \\
B_2 = a_{0,2} I + a_{1,2} A + a_{2,2} A^2 + a_{3,2} A^3, \\
B_3 = a_{0,3} I + a_{1,3} A + a_{2,3} A^2 + a_{3,3} A^3, \\
B_4 = a_{0,4} I + a_{1,4} A + a_{2,4} A^2 + a_{3,4} A^3, \\
A_6 = B_3 + B_4^2
$$

(12)

This approach, however, leads to a much less stable numerical behaviour. We propose instead to use the following sequence, which has been empirically shown to be comparable in stability to Padé and Horner methods.

$$
A_2 = A^2, \\
A_3 = A_2 A, \\
B_1 = a_{0,1} I + a_{1,1} A + a_{2,1} A^2 + a_{3,1} A^3, \\
B_2 = a_{0,2} I + a_{1,2} A + a_{2,2} A^2 + a_{3,2} A^3, \\
B_3 = a_{0,3} I + a_{1,3} A + a_{2,3} A^2 + a_{3,3} A^3, \\
B_4 = a_{0,4} I + a_{1,4} A + a_{2,4} A^2 + a_{3,4} A^3, \\
A_6 = B_3 + B_4^2
$$

$$
T_{12}(A) = B_1 + (B_2 + A_6)A_6.
$$

The algebraic expressions which minimize the 1-norm of the vector formed by the parameters do not give additional insight and hence we show their numeric values only:

$$
a_{0,1} = 9.0198 \cdot 10^{-16}, \quad a_{0,2} = 5.31597895759871264183, \\
a_{0,3} = 0.18188869982170434744, \quad a_{0,4} = -2.0861320 \cdot 10^{-13}, \\
a_{1,1} = 0.46932117595418237389, \quad a_{1,2} = 1.19926790417132231573, \\
a_{1,3} = 0.05502798439925399070, \quad a_{1,4} = -0.13181061013830184015, \\
a_{2,1} = -0.20099424927047284052, \quad a_{2,2} = 0.0117929624099297031, \\
a_{2,3} = 0.0935159070535414968, \quad a_{2,4} = -0.02027855540589259079, \\
a_{3,1} = -0.04623946134063071740, \quad a_{3,2} = 0.01108844528519167989, \\
a_{3,3} = 0.00610700528898058230, \quad a_{3,4} = -0.00675951846863086359.
$$

$k = 5$ products With 5 products, $n = 18$ is the highest value we have been able to achieve. We write $T_{18}$ as the product of two polynomials of degree 9, that are further decomposed into polynomials of lower degree. The polynomial
is evaluated through the following sequence:

\[ A_2 = A^2, \quad A_3 = A_2 A, \quad A_6 = A_3^2, \]

\[ B_1 = a_{0,1} I + a_{1,1} A + a_{2,1} A^2 + a_{3,1} A_3, \]

\[ B_2 = b_{0,1} I + b_{1,1} A + b_{2,1} A^2 + b_{3,1} A_3 + b_{6,1} A_6, \]

\[ B_3 = b_{0,2} I + b_{1,2} A + b_{2,2} A^2 + b_{3,2} A_3 + b_{6,2} A_6, \]

\[ B_4 = b_{0,3} I + b_{1,3} A + b_{2,3} A^2 + b_{3,3} A_3 + b_{6,3} A_6, \]

\[ B_5 = b_{0,4} I + b_{1,4} A + b_{2,4} A^2 + b_{3,4} A_3 + b_{6,4} A_6, \]

\[ A_9 = B_1 B_5 + B_4, \]

\[ T_{18}(A) = B_2 + (B_3 + A_9) A_9, \]

with coefficients

\[
\begin{align*}
  a_{0,1} &= 0, & a_{1,1} &= -0.100365581030144620, \\
  a_{2,1} &= -0.008029246824115696, & a_{3,1} &= -0.0008921384980457299, \\
  b_{0,1} &= 0, & b_{1,1} &= 0.39784974949964507614, \\
  b_{2,1} &= 1.36783778460411719922, & b_{3,1} &= 0.49828962252538267755, \\
  b_{6,1} &= -0.10967639605296206259, & b_{6,2} &= 0.0571779846478655127, \\
  b_{1,2} &= 1.68015813878906197182, & b_{2,2} &= 0.00003349750170860705, \\
  b_{3,2} &= -0.0069821012248805208, & b_{6,2} &= -0.00676404519071381907, \\
  b_{9,3} &= -0.0904316832390810561, & b_{1,3} &= -0.02955525704293155274, \\
  b_{2,3} &= 0.06759613017704596460, & b_{3,3} &= -0.0000139180257516060, \\
  b_{6,3} &= -0.0000139180257516060, & b_{6,4} &= 0, \\
  b_{1,4} &= 0, & b_{2,4} &= -0.0923364619367118592, \\
  b_{3,4} &= -0.0169364939002081717, & b_{6,4} &= -0.0000140086798182036.
\end{align*}
\]

**k = 6 products** With six products we can reconstruct the Taylor polynomial up to degree \( n = 22 \) by applying the same strategy. We have also explored different alternatives, considering decompositions based on the previous computation of low powers of the matrix, such as \( A^2, A^3, A^4, A^6, \) etc. to achieve degree \( n = 24, \) but all our attempts have been in vain. Nevertheless, we should remark that even if one could construct \( T_{24}(A) \) with only six products, this would not lead to a significant advantage with respect to considering one scaling and squaring \( (s = 1 \text{ in eq. (1))} \) and using the previous decomposition for \( T_{18}. \)

In Table 3 we show the number of products required to evaluate \( T_n \) following the PS rule and the new decomposition strategy. The improvement for \( k \geq 3 \) products is apparent.

### 4 Numerical performance and stability

We next try to assess the performance of the various approximations by comparing the efficiency of the new procedure to compute the Taylor polynomial with the PS rule and also with diagonal Padé approximants, all in combination with the scaling and squaring method for the matrix exponential.
Table 3: Minimal number of products to evaluate a polynomial of a given degree.

| Method              | Products | Degree |
|---------------------|----------|--------|
| Paterson-Stockmeyer | 1 2 3 4 5 6 | 2 4 6 9 12 16 |
| New decomposition   | 1 2 3 4 5 6 | 2 4 8 12 18 22 |

All methods are based on an estimate of the matrix norm $\|A\|$, which is then used to choose the optimal order of the approximant together with the scaling parameter, based on the use of backward error bounds.

In Fig. 1 we plot $\|A\|$ versus the cost measured as the number of matrix products necessary to evaluate Padé and Taylor with the PS rule and the new implementation to approximate $e^A$ in double precision (top) and single precision (bottom). It is clear that for relatively small values of $\|A\|$ the new approach to compute the Taylor approximant shows the best performance, whereas for higher values it has similar performance as Padé. The diagonal lines in the graphs show the asymptotic cost based on scaling and squaring.

To check the numerical stability of the proposed technique we have considered 46 different special matrices from the MATLAB matrix collection, as has been done in [8], sampled at 1000 different norms, as well as 2000 random matrices which are scaled by random scalars to spread them over a range of norms. In Fig. 2 we plot the relative errors in the computation of the corresponding matrix exponential vs. the norm of the matrix in double (top) and single precision (bottom) for the different approximants analysed here. All matrices are of dimension $30 \times 30$, but the same experiments with matrices of larger dimension (up to $60 \times 60$) show virtually identical results.

As a reference, we use a diagonal Padé method of order 26 and the matrices are scaled and squared according to criterion (5) until the resulting norm is below the accuracy threshold from Table 1.

We observe that the accuracy of our method lies well within the range that can be expected for Horner scheme evaluations and the Padé algorithm from Table 1.

5 Concluding remarks

It is important to remark that very often the evaluation of the exponential of a matrix constitutes an intermediate stage in the process of solving differential equations by exponential integrators. This is the case, in particular, for
equations of the form

\[ X' = F(t, X)X , \]
\[ X' = [F(t, X), X] , \]
\[ X' = [[F(t), X], X] , \]

where \( X(t_0) = X_0 \in \mathbb{R}^{n \times n} , t \in [t_0, T] \) and \([\cdot, \cdot]\) denotes the usual matrix commutator. Among the advantages of exponential integrators applied to these equations are that they preserve qualitative properties and their favourable long-time error propagation \([11]\).

Take for example the linear equation \( X' = F(t)X \). If \( F(t) \) belongs to some particular matrix Lie algebra (e.g., \( F \) is skew-symmetric), then the solution \( X(t) \) evolves in the corresponding Lie group (e.g., \( X(t) \) is orthogonal for all \( t \)), and one would like that any numerical approximation to \( X \) obtained by applying an integrator share this important feature with the exact solution. If the integrator is formulated in terms of matrix exponentials, such as with Magnus or Fer integrators \([4]\), this is achieved by construction.

Given a time step, say \( h \), one usually considers a numerical integrator of this class of order \( hp \), for values of \( p \) usually between 2 and 6. Since the matrices involved in the schemes are typically proportional to \( h \) and the truncated polynomial approximation, \( T_n \), is an approximation to order \( O(h^{n+1}) \), then one rarely needs to consider \( n > 18 \).

In some cases one can take advantage of the very particular structure of the matrix \( A \) and then design especially tailored (and very often more efficient) methods for such problems \([6, 7, 12]\). Also, if one can find an additive decomposition \( A = B + C \) such that \( \|C\| \) is small and \( B \) is easy to exponentiate, e.g., \( e^D \) is sparse and exactly solvable (or can be accurately and cheaply approximated numerically), and \( B \) is a dense matrix, then more efficient methods can be found \([3, 16]\). In addition, if \( A \) is an upper or lower triangular matrix, it is shown in \([2]\) that it is advantageous to exploit the fact that the diagonal elements of the exponential are exactly known. It is then more efficient to replace the diagonal elements obtained numerically by the exact solution before squaring the matrix. This technique can also be extended to the first super (or sub-)diagonal elements.

In summary, we have derived a new algorithm to compute the matrix exponential by reducing the cost of the evaluation of its Taylor polynomial. The method is as stable as the Padé methods implemented in the MATLAB function \texttt{expm} and is cheaper to compute than Padé approximants for a wide range values for the matrix norm. For the convenience of the reader, we provide a MATLAB implementation of the proposed scheme on our website \([1]\).

Acknowledgments

This work has been funded by Ministerio de Economía, Industria y Competitividad (Spain) through project MTM2016-77660-P (AEI/FEDER, UE).
References

[1] http://www.gicas.uji.es/Research/MatrixExp.html

[2] A.H. Al-Mohy and N.J. Higham, A new Scaling and Squaring Algorithm for the Matrix Exponential, SIAM J. Matrix Anal. Appl., 31, (2009), pp. 970–989.

[3] P. Bader, S. Blanes, and M. Seydaoglu, The Scaling, Splitting and Squaring Method for the Exponential of Perturbed Matrices, SIAM J. Matrix Anal. Appl., 36, (2015), pp. 594–614.

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

[5] 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), pp. 396–412.

[6] E. Celledoni and A. Iserles, Approximating the exponential from a Lie algebra to a Lie group, Math. Comput., 69 (2000), pp. 1457–1480.

[7] E. Celledoni and A. Iserles, Methods for the Approximation of the Matrix Exponential in a Lie-Algebraic Setting, IMA J. Numer. Anal., 21 (2001), pp. 463–488.

[8] N.J. Higham, The Scaling and Squaring Method for the Matrix Exponential Revisited, SIAM J. Matrix Anal. Appl., 26, (2005), pp. 1179–1193.

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

[10] N.J. Higham and A.H. Al-Mohy, Computing Matrix Functions, Acta Numerica, 51, (2010), pp. 209–286.

[11] M. Hockbruck and A. Ostermann, Exponential integrators, Acta Numerica, 19, (2010), pp. 209–286.

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

[13] Li 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), pp. 299–300.

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

[15] 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), pp. 3–49.
[16] I. Najfeld and T.F. Havel, Derivatives of the Matrix Exponential and Their Computation Adv. Appl. Math., 16 (1995), pp. 321–375.

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

[18] P. Ruiz, J. Sastre, J. Ibáñez, and E. Defez, High performance computing of the matrix exponential, J. Comput. Appl. Math., 291 (2016), pp. 370–379.

[19] J. Sastre, J. Ibáñez, E. Defez, and P. Ruiz, Accurate matrix exponential computation to solve coupled differential models in Engineering, Math. Compu. Modelling, 54 (2011), pp. 1835–1840.

[20] J. Sastre, J. Ibáñez, P. Ruiz, and E. Defez, Accurate and efficient matrix exponential computation, Int. J. Comput. Math., 91 (2014), pp. 97–112.

[21] J. Sastre, J. Ibáñez, P. Ruiz, and E. Defez, New scaling-squaring Taylor algorithms for computing the matrix exponential, SIAM J. Sci. Comp., 37 (2015), pp. A439–A455.

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

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

[24] D. Westreich, Evaluating the Matrix Polynomial $I + A + \cdots + A^{N-1}$, IEEE Trans. Circuits Sys.,36 (1989), pp. 162–164.
Figure 1: Optimal orders and their respective cost vs. the norm of the matrix exponent. The numbers indicate the order of the approximants. The diagonal lines show the asymptotic cost based on scaling and squaring and the corresponding orders are highlighted in bold-face.

Double precision

Single precision
Figure 2: Relative errors for a range of $30 \times 30$ matrices computed with various algorithms as discussed in the text. Notice that all methods show similar (double) precision which is close to round-off accuracy. The error growth towards larger norms stems from the squaring error and affects all methods alike. The black horizontal line is the backward precision goal of the scaled exponential, cf. (5).