SIAM J. Sci. Comput.
Vol. 37, No. 1, pp. A439–A455

NEW SCALING-SQUARING TAYLOR ALGORITHMS FOR COMPUTING THE MATRIX EXPONENTIAL[*]

J. SASTRE[†], J. IBÁÑEZ[‡], E. DEFEZ[§], AND P. RUIZ[‡]

Abstract. The matrix exponential plays a fundamental role in linear differential equations arising in engineering, mechanics, and control theory. The most widely used, and the most generally efficient, technique for calculating the matrix exponential is a combination of "scaling and squaring" with a Padé approximation. For alternative scaling and squaring methods based on Taylor series, we present two modifications that provably reduce the number of matrix multiplications needed to satisfy the required accuracy bounds, and a detailed comparison of the several algorithmic variants is provided.

Key words. matrix exponential, Taylor series, Paterson–Stockmeyer method, backward error analysis, computational cost analysis

AMS subject classification. 65F30

DOI. 10.1137/090763202

1. Introduction.

Many engineering and physics phenomena are governed by systems of linear first-order ordinary differential equations with constant coefficients, whose solution is given in terms of the matrix exponential $\exp(A)$, $A \in \mathbb{C}^{n\times n}$. Thus, the matrix exponential plays an important role in many areas of science and technology: control theory, electrodynamic theory of stratified media, the theory of multimode electric power lines, etc. [HLS98, CM02, KT05, IHAR09]. Numerous methods have been proposed for matrix exponential computation [MV03, Hig08]. Of all the methods, Padé approximation in combination with the scaling and squaring technique is the most popular general method [Hig05]. This paper presents two modifications of a Taylor-based scaling and squaring algorithm that are designed to reduce computational costs while preserving accuracy. A previous unpublished and extended version of this work can be found in [SIDR09].

Throughout this paper $\mathbb{R}^{n\times n}$ and $\mathbb{C}^{n\times n}$ denote the sets of real and complex matrices of size $n \times n$, respectively, and I denotes the identity matrix for both sets. $\mathbb{N}$ denotes the set of positive integers and the matrix norm $\|\cdot\|$ denotes any subordinate matrix norm, and in particular $\|\cdot\|_1$, the 1-norm. $\lceil x \rceil$ denotes the lowest integer not less than x and $\lfloor x \rfloor$ denotes the highest integer not exceeding x. This paper is organized as follows. Section 2 presents a state of the art on matrix exponential computation. Section 3 deals with the proposed improvements. Numerical experiments are presented in section 4. Finally, the conclusions are given in section 5.

2. State of the art.

Given a complex matrix A, $\exp(A)$ can be well approximated by either a Padé or Taylor series approximation if $\|A\|$ is small enough. This

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section June 21, 2009; accepted for publication (in revised form) December 1, 2014; published electronically February 12, 2015. This work was supported by the *Generalitat Valenciana* project GVPRE/2008/340.
http://www.siam.org/journals/sisc/37-1/76320.html

[†]Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de València, Camino de Vera s/n, 46022-Valencia, Spain (jorsasma@iteam.upv.es).

[‡]Instituto de Instrumentación para Imagen Molecular, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, Spain (jjibanez@dsic.upv.es, pruiz@dsic.upv.es).

[§]Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, Spain (edefez@imm.upv.es).

suggests exploiting the relation

$$\exp(A) = \left(\exp(2^{-s}A)\right)^{2^s}, \tag{2.1}$$

where s is a nonnegative integer scaling parameter. Based on (2.1), the idea of a scaling and squaring method is to choose s so that $\|2^{-s}A\|$ is sufficiently small, to use Padé or Taylor series to approximate $\exp(2^{-s}A)$, and finally to square the result s times, in the so-called squaring phase. It is well known that the squaring phase can be badly affected by rounding errors (see [Hig08, p. 248]) and it is therefore desirable to keep s as small as possible.

2.1. Taylor and Padé series.

The Taylor series approximation to $\exp(A)$, discussed in [Wes90, GV96, MV03], [PCK91, p. 210] is

$$T_m(A) = \sum_{i=0}^{m} \frac{A^i}{i!}, \tag{2.2}$$

where m is the degree of the matrix polynomial $T_m(A)$, and the total number of terms in the series is $m+1$. In general, a larger value of m will improve accuracy, i.e., reduce the absolute error. From now on, we refer to m as the order of the approximation. A naive use of Taylor series is well known to produce serious cancellation error [GV96, p. 567], [MV03, p. 10]. Taylor series approximations, the topic of this paper, are discussed in more detail in section 2.2.

Padé approximants [War77, GV96, MV03, Hig05, Hig08] are basic tools for computing the matrix exponential. The $[k/l]$ Padé approximant $r_{kl}(A)$ of the matrix exponential is defined by

$$r_{kl}(A) = p_{kl}(A)\left(q_{kl}(A)\right)^{-1},$$

where

$$p_{kl}(A) = \sum_{j=0}^{k} \frac{(k+l-j)!k!}{(k+l)!(k-j)!j!}A^j, \quad q_{kl}(A) = \sum_{j=0}^{k} \frac{(-1)^j(k+l-j)!l!}{(k+l)!(l-j)!j!}A^j.$$

Diagonal approximants ($k = l$) are preferred because r_{mm} is more accurate than r_{kl} with $k \neq l$ but can be evaluated at the same cost. A Padé-based scaling and squaring algorithm is given in [Hig05].

For reasons discussed in [Hig05] and [HiAM10], the most popular techniques for computing the matrix exponential are based on scaling and squaring using a Padé approximation, which, in general, produces acceptable accuracy with less work than a Taylor-series method.

2.2. Details about Taylor series approximants.

Algorithm GSQT summarizes the general structure of a Taylor-based scaling and squaring algorithm for computing the matrix exponential, where preprocessing and postprocessing (see [War77]) have been omitted.

The proposed modifications affect Algorithms Order-scale (see section 3.1) and Taylor-eval (section 3.2).

Algorithm GSQT (general scaling and squaring Taylor algorithm).
% **Input:** An $n \times n$ matrix A, preprocessed if appropriate;
% K, the maximum allowed number of matrix products; and
% $\{m_k\}, k = 1 : K$, the orders of the associated polynomials, from Table 2.
Step 1. Execute Algorithm Order-scale, which selects the order and scaling parameter of the Taylor polynomial.
Step 2. Execute Algorithm Taylor-eval to evaluate the Taylor polynomial in the scaled matrix.
Step 3. Execute the appropriate number of squaring steps of the Taylor polynomial.

2.2.1. Relations between the order and the computational effort. The principles that underlie implementation of the Order-scale choice in Algorithm GSQT are based on [PS73, GV96, Hig05], and [Hig08, pp. 72–74]. Given a matrix A, a scaling parameter s, and relative machine precision u (typically, IEEE double precision), [Hig05] shows how to define a sequence $\{\theta_m\}$ such that an (m, m) diagonal Padé approximation to $\exp(A)$ produces an acceptable backward error bound if $\|2^{-s}A\| \leq \theta_m$. Applying an analogous procedure to Taylor series, [SIDR09] calculated a sequence $\{\Theta_m\}$ such that, if

$$\left\|2^{-s}A\right\| \leq \Theta_m, \tag{2.3}$$

using an order-m Taylor series approximation gives an acceptable relative backward error bound using IEEE double precision. Table 1 displays values of $\{\theta_m\}$ and $\{\Theta_m\}$.

The work associated with approximating the matrix exponential using Padé or Taylor series is measured by convention as the number of matrix products. A matrix polynomial can be evaluated in a straightforward way using Horner's nested multiplication method (see, e.g., [GV96, p. 574]). [Hig05] and [Hig08] show how Horner's technique can be combined with a method due to Paterson and Stockmeyer [PS73] to produce the smallest number of matrix products, denoted by π_m, needed to calculate the (m, m) diagonal Padé approximant.

As described in [SIDR09], an analogous procedure can be applied to a degree-m Taylor approximation of $\exp(A)$. In the Paterson–Stockmeyer method, a positive integer $q < m$ is chosen and the polynomial (2.2) is written as a degree-r polynomial in A^q with $r = \lfloor m/q \rfloor$:

$$T_m(A) = \sum_{k=0}^{r} B_k \left(A^q\right)^k, \quad r = \lfloor m/q \rfloor, \tag{2.4}$$

Table 1
Comparison of Padé and Taylor approximations.

Padé			Taylor		
m	θ_m	π_m	m	Θ_m	Π_m
3	1.5e-2	2	3	1.39e-5	2
4	8.5e-2	3	4	3.40e-4	2
5	2.5e-1	3	5	2.40e-3	3
6	5.4e-1	4	6	9.07e-3	3
7	9.5e-1	4	7	2.38e-2	4
8	1.5e0	5	8	5.00e-2	4
9	2.1e0	5	9	8.96e-2	4
10	2.8e0	6	10	1.44e-1	5
11	3.6e0	6	11	2.14e-1	5
12	4.5e0	6	12	3.00e-1	5

where the Paterson–Stockmeyer coefficients $\{B_k\}$, $k = 0, \ldots, r$, are themselves matrix polynomials:

$$B_k = \sum_{j=0}^{q-1} b_{qk+j} A^j, \; k = 0, \ldots, r-1, \; \text{and } B_r = b_{qr} I + \cdots + b_m A^{m-qr}. \tag{2.5}$$

Note that, with this form, each B_k, $k = 0, \ldots, r-1$, contains q terms, whereas B_r contains $m - qr + 1$ terms. The Paterson–Stockmeyer form is recursively evaluated as follows with the Horner technique, where F_0 gives the final result.

```
$F_r = B_r$;
for  $j = r-1 : -1 : 0$,   $F_j = B_j + A^q \times F_{j+1}$;   end for
```

In general, forming a degree-m Taylor polynomial in this way with the Horner–Paterson–Stockmeyer technique requires $q+r-1$ matrix products, with $q-1$ of these used to form $A^2, \ldots, A^q$, plus $r = \lfloor m/q \rfloor$ matrix products to compute the sequence $\{F_j\}$. However, in the special case when q divides m (i.e., when $m = qr$), then only $r-1$ matrix products are needed to apply Horner's technique because the Paterson–Stockmeyer coefficient $B_r = b_m I$ is a multiple of the identity; hence forming F_{r-1} does not involve a matrix product.

Given $q < m$, the number of matrix products needed to compute T_m using the Horner–Paterson–Stockmeyer procedure is thus given by $q + \lfloor m/q \rfloor - 1 - \psi(q, m)$, where $\psi(q, m) = 1$ if q divides m and 0 otherwise. The value of q for which T_m can be computed with the smallest number of matrix products is obtained by approximately minimizing $q + m/q$, giving $q = \lfloor \sqrt{m} \rfloor$. We use Π_m to denote the associated "optimal" number of matrix multiplications:

$$\Pi_m = \lfloor \sqrt{m} \rfloor + \lfloor m/\sqrt{m} \rfloor.$$

Calculation of the Taylor-based sequences $\{\Theta_m\}$ (2.3) and $\{\Pi_m\}$ is summarized in [SIDR09] and in an appendix of [HiAM10], with $\tilde{\theta}_m$ denoting Θ_m and $\tilde{\pi}_m$ denoting Π_m. For selected values of m, Table 1 displays θ_m and π_m for Padé approximants, and Θ_m and Π_m for Taylor approximants.

Comparing π_m and Π_m, keeping in mind that $\theta_m > \Theta_m$ for each m, we see that, as noted in [SIDR09] and the Appendix of [HiAM10], a Padé approximation requires fewer matrix products except when $\|A\|$ lies in the following three intervals:

$$\begin{aligned} \|A\| &\le 9.07e-3 \; (\Theta_6), \\ 1.50e-2 \; (\theta_3) \le \|A\| &\le 8.96e-2 \; (\Theta_9), \quad \text{and} \\ 2.54e-1 \; (\theta_5) \le \|A\| &\le 3.00e-1 \; (\Theta_{12}). \end{aligned} \tag{2.6}$$

2.2.2. Maximizing the order. Combining the analysis of [Hig05] and the results of Table 1 when k is a specified number of matrix products, [SIDR09] shows how to define m_k, the highest-order Taylor approximation that can be obtained with k matrix products when $\|A\| \le \Theta_{m_k}$. For $k = 1 : 11$, Table 2 shows m_k, q_k, and r_k (the associated Paterson–Stockmeyer indices), Π_{m_k}, and Θ_{m_k}.

Assuming that $\|A\| \le \Theta_{m_k}$, Table 2 shows that for even k, the largest Taylor polynomial order achievable with k matrix products is $m_k = (k+2)^2/4$, with $q_k = r_k = (k+2)/2$ and $m_{k+1} = m_k + q_k$, so that allowing one additional matrix product increases the order of the Taylor polynomial by q_k. For an odd number k of matrix

Table 2

k, m_k, q_k, r_k, Θ_{m_k}, and Π_{m_k}.

k	m_k	q_k	r_k	Θ_{m_k}	Π_{m_k}
1	2	1	2	2.58e-8	1
2	4	2	2	3.40e-4	2
3	6	2	3	9.07e-3	3
4	9	3	3	8.96e-2	4
5	12	3	4	3.00e-1	5
6	16	4	4	7.80e-1	6
7	20	4	5	1.44e0	7
8	25	5	5	2.43e0	8
9	30	5	6	3.54e0	9
10	36	6	6	4.97e0	10
11	42	6	7	6.48e0	11

products, where $q_k = (k+1)/2$ and $r_k = q_k + 1$, the order of the Taylor polynomial increases by $q_k + 1$ if $k+1$ matrix products are allowed.

In all cases of interest to us, $m_k = q_k r_k$. This means that the Paterson–Stockmeyer form of T_m (2.4) can be expressed as an equivalent polynomial of degree $r-1$ in A^q, giving the following expression in which the "extra" term $b_0 I$ is added separately:

$$T_m(A) = b_0 I + \sum_{k=0}^{r-1} \bar{B}_k (A^q)^k \quad \text{with} \quad \bar{B}_k = \sum_{j=1}^{q} b_{qk+j} A^j,\ k = 0, \ldots, r-1. \tag{2.7}$$

Note that, as distinct from (2.4), the Paterson–Stockmeyer coefficients are barred, and that every $\bar{B}_k,\ k = 0, \ldots, r-1$, contains q terms.

The following three pseudocode fragments show explicitly how the Taylor polynomial $T_m(A)$ is evaluated using (2.7) when $m = qr$. The reader should note in particular the ranges of the indices k and j.

```
Algorithm Taylor-eval.
% Horner–Paterson–Stockmeyer evaluation of the order-$m$ Taylor polynomial
% of $\exp(A)$ with $m = qr$.
% Input parameters: $m$; $q$; $r$; $\{A_j = A^j\}, j = 1 : q$; $\{b_i\}, i = 0 : m$.
Execute Algorithm PS-coeff;     % calculate $\{\bar{B}_j\}, j = 0 : r-1$;
Execute Algorithm HPS-eval;     % calculate $T_m(A)$;
end % Taylor-eval
```

```
Algorithm PS-coeff.
% calculation of the $r$ Paterson–Stockmeyer coefficients $\{\bar{B}_k\}$ in (2.7), where $b_i = 1/i!$.
% Input: $q$; $r$; $\{A_j = A^j\}, j = 1 : q$; $\{b_i\}, i = 0 : qr$.
for $k = 0 : r-1$
    $\bar{B}_k = 0$;
    for $j = 1 : q$
        $\bar{B}_k = \bar{B}_k + b_{qk+j} A_j$;
    end for $j$
end for $k$
end % PS-coeff
```

Table 3
Overall cost (2.8) *as a function of Taylor order* m.

k	2	3	4	5	6	7	8	9
m_k	4	6	9	12	16	20	25	30
Γ_{m_k}	13.52	9.79	7.48	6.74	6.36	6.48	6.72	7.18

```
Algorithm HPS-eval.
% Horner–Paterson–Stockmeyer evaluation of the order-$m$ Taylor polynomial
% of $\exp(A)$ with $m = qr$, from (2.7).
% Input: $r$; $b_0$; $A_q = A^q$; $\{\bar{B}_k\}, k = 0 : r-1$.
$F = \bar{B}_{r-1}$;
for $j = r-1 : -1 : 1$
    $F = \bar{B}_{j-1} + A_q \times F$;
end for $j$
$F = F + b_0 I$;
end % HPS-eval
```

2.2.3. Estimating the total cost. Based on [Hig05], the following approach can be used to estimate the cost of calculating the matrix exponential as a function of the order m. For Taylor series, if $\|A\| > \Theta_m$ for some order m, the scaling parameter s can be chosen as the smallest value of s such that $\|A/2^s\| \leq \Theta_m$, namely, $s = \lceil \log_2(\|A\|/\Theta_m) \rceil$. After calculating the order-m Taylor approximant of $\exp(A/2^s)$, the result will be squared s times, so that the total number of matrix multiplications required to calculate the order-m Taylor approximation is $\Pi_m + \lceil \log_2(\|A\|/\Theta_m) \rceil$. An approximation of this value that depends only on m, denoted by Γ_m, is obtained by omitting the ceiling operation and the constant $\|A\|$:

$$\Gamma_m = \Pi_m - \log_2(\Theta_m). \tag{2.8}$$

When estimating the total for Padé approximations, [Hig05] rules out $m = 1$ and $m = 2$ because of the unfavorable numerical consequences resulting from the need for a large s to make $\|A/2^s\|$ small enough. The same argument applies to very low-order Taylor polynomials, and we therefore show the values of Γ_{m_k} (2.8) in Table 3 only for $k \geq 2$. Table 3 shows that, as measured by Γ, six matrix products, corresponding to order $m_6 = 16$, produce the minimum computational cost for a Taylor-based approximation.

The usual practice in Taylor-based methods is to decide in advance on the maximum allowed number of matrix products, denoted by K. Based on this practice, Algorithm Order-scale-1 summarizes the procedures for choosing the order $\hat{m}$ and the scaling parameter $\hat{s}$ in a standard Taylor-based method when K is given.

```
Algorithm order-scale-1.
% Standard technique for choosing the order $\hat{m}$ and the scaling parameter $\hat{s}$
% to compute a Taylor approximation to $\exp(A)$.
% Input: $K$, the maximum number of matrix products allowed to evaluate
    the polynomial; $\{m_k\}$ and $\Theta_{m_k}$ from Table 2.
% Output: $\hat{k}$, $\hat{m}$, and $\hat{s}$ (the values needed).
$\hat{m} = m_K$; $\hat{k} = K$;
```

```
if $\|A\| > \Theta_{m_K}$ then
    $\hat{s} = \lceil(\log_2(\|A\|/\Theta_{m_K})\rceil$
else
    $\hat{s} = 0$;
    for $k = 1 : K$
        if $\|A\| \le \Theta_{m_k}$ then $\hat{m} = m_k$; $\hat{k} = k$; break; end if
    end for $k$
end if
end      % Order-scale-1
```

In the remainder of this paper, we consider new strategies designed to reduce the number of matrix multiplications while preserving accuracy.

3. Proposed modifications.

We propose two modifications to the standard Taylor-based method presented in section 2.2. The first modification, described in section 3.1, replaces Algorithm Order-scale-1 with a method that changes the order and the scaling parameter if another combination of these values reduces the number of matrix products. The second modification, discussed in section 3.2, neglects higher-order terms in the Taylor series based on relative error bounds.

3.1. Modification of the choices for order and scaling.

Two modifications are proposed of Algorithm Order-scale-1 for choosing the order and scaling parameter. The motivation for the first change is based on the logic of Algorithm Order-scale-1. If $\|A\| \le \Theta_{m_K}$, $\hat{k}$ is taken as the largest value such that $\|A\| \le \Theta_{m_{\hat{k}}}$ with $\hat{m} = m_{\hat{k}}$ and $\hat{s} = 0$; evaluation of the Taylor polynomial then requires $\hat{k}$ matrix products. When $\|A\| > \Theta_{m_K}$, then $\hat{k} = K$, $\hat{m} = m_K$, and

$$\hat{s} = \left\lceil \log_2\left(\frac{\|A\|}{\Theta_{m_K}}\right)\right\rceil . \tag{3.1}$$

We know in this case that $\hat{s} \ge 1$ and, by definition of $\hat{s}$, that

$$\frac{1}{2}\Theta_{m_K} < \frac{\|A\|}{2^{\hat{s}}} \le \Theta_{m_K}. \tag{3.2}$$

With these latter choices, a Taylor approximation of order m_K to $\exp(A)$ can be computed with $K + \hat{s}$ matrix products.

But it is possible in some cases to reduce this number. If K, $\hat{s}$, and $\|A\|$ are such that $\|A\|/2^{\hat{s}} \le \Theta_{m_{K-1}}$, then we can take $k^* = K - 1$, so that the order of the Taylor polynomial is m_{K-1}, keeping $s^* = \hat{s}$. Thus the number of matrix products is $K-1+\hat{s}$, one smaller than with the original choices. It follows from (3.2) that this situation is possible only when $\Theta_{m_{K-1}} > \frac{1}{2}\Theta_{m_K}$, which is true only for $K \ge 7$ (see Table 2). The strategy just described corresponds to executing Algorithm Order-scale-2, shown below, with option = 1.

If one is prepared to accept additional squarings, the idea can be generalized in a limited way for certain values of $\|A\|$ when $K = 8$ and $K = 9$, as implemented in Algorithm Order-scale-2. The idea is to find $k^* < K - 1$ and $s^* > \hat{s}$ such that $\|A\|/2^{s^*} < \Theta_{m_{k^*}}$ and $k^* + s^* < K - 1 + \hat{s}$, meaning that, using order m_{k^*} and scaling parameter s^*, the total number of matrix products is smaller than with option = 1. Allowing such changes corresponds to option = 2 in Algorithm Order-scale-2.

The pseudocode for Algorithm Order-scale-2 assumes that $K \le 9$, since it is straightforward to show (see [SIDR09]) that performing the analogue of option 2 with $K > 9$ does not reduce the number of matrix products.

```
ALGORITHM ORDER-SCALE-2.
% Two alternative ways to choose the order $m^*$ of a Taylor approximation
% to $\exp(A)$ and the scaling parameter $s^*$;
% Input: option, set to 0, 1, or 2; $K$, the maximum number of matrix products
% allowed in evaluating the polynomial; $\{m_k\}$ and $\{\Theta_{m_k}\}$ from Table 2.
% Output: $k^*$, $m^*$, and $s^*$.
$m^* = m_K$; $k^* = K$;
if $\|A\| \leq \Theta_{m_K}$ then
    $s^* = 0$; $\hat{s} = 0$;
    for $k = 1 : K$
        if $\|A\| \leq \Theta_{m_k}$ then $m^* = m_k$; $k^* = k$; break; end if
    end for
    if option $= 2$ then
        if $k \geq 8$ and $\|A\| \leq 2\Theta_{m_{k-2}}$ then $k^* = k - 2$; $m^* = m_{k^*}$; $s^* = 1$;
        else if $k = 9$ and $\|A\| \leq 4\Theta_{m_{k-3}}$ then $k^* = k - 3$; $m^* = m_{k^*}$; $s^* = 2$;
        end if
    end if     % end logic for option $= 2$
else     % here, it must be true that $\|A\| > \Theta_{m_K}$
    $\hat{s} = \lceil \log_2(\|A\|/\Theta_{m_K}) \rceil$; $s^* = \hat{s}$;
    if option $> 0$ then     % possibly change the order and scale
        if $K \geq 7$ and $\|A\|/2^{\hat{s}} \leq \Theta_{m_{K-1}}$ then $k^* = K - 1$; $m^* = m_{k^*}$;
        else if option $= 2$ then
            if $K \geq 8$ and $\|A\|/2^{\hat{s}} \leq 2\Theta_{m_{K-2}}$ then $k^* = K-2$; $m^* = m_{k^*}$; $s^* = \hat{s}+1$;
            else if $K = 9$ and $\|A\|/2^{\hat{s}} \leq 4\Theta_{m_{K-3}}$ then
                $k^* = K - 3$; $m^* = m_{k^*}$; $s^* = \hat{s} + 2$;
            end if
        end if     % end logic for option $= 2$
    end if     % end logic for option $> 0$
end if     % end logic for $\|A\| > \Theta_{m_K}$
end     % Order-scale-2
```

When $K \geq 7$, assuming that $\hat{s}$ is defined by (3.1), Table 4 shows the number of matrix products needed with option 0 (the standard method), option 1, and option 2 to approximate $\exp(A)$ when $\|A\|/2^{\hat{s}}$ lies in the intervals shown.

For certain intervals of $\|A\|$, the total number of matrix products needed to form the Taylor polynomial with option 2 can be lower than with option 1. However, in those cases the number s^* of squarings is larger, possibly leading to more numerical error. Numerical tests that explore this issue are given in section 4.

3.2. Neglecting higher-order terms of the Taylor polynomial. The motivation for the second modification is that, in some circumstances, the highest-order terms in the Taylor polynomial are negligible relative to $\|\exp(A)\|$, where "negligible" is defined in terms of the level of rounding error.

Recall from Algorithm Taylor-eval in section 2.2 that, given a number of multiplications k, the Taylor approximation of order m_k is defined by applying the recursive Horner–Paterson–Stockmeyer procedure, where there are r_k Paterson–Stockmeyer coefficients, $\bar{B}_0, \bar{B}_1, \ldots, \bar{B}_{r_k-1}$, each a matrix polynomial in A of degree q_k (see Algorithm PS-coeff). It follows from the discussion in section 2.2.2 that reducing the number of matrix multiplications from k to $k-1$ implies a reduction by q_k in order of the Taylor polynomial (see Table 2).

TABLE 4

Matrix products needed when using options $0, 1,$ *and* 2 *in Algorithm Order-scale-*2 *for* $K \geq 7$ *to approximate* $\exp(A)$ *when* $\|A\|/2^{\hat{s}}$ *lies in the intervals shown, where* $\hat{s}$ *is defined by* (3.1).

K	Option	Interval of $\|A\|/2^{\hat{s}}$	k^*	m^*	s^*	Matrix products
7	0	$(\frac{1}{2}\Theta_{20}, \Theta_{20}] = (0.72, 1.44]$	7	20	$\hat{s}$	$7 + \hat{s}$
7	1	$(\frac{1}{2}\Theta_{20}, \Theta_{16}] = (0.72, 0.781]$	6	16	$\hat{s}$	$6 + \hat{s}$
7	1	$(\Theta_{16}, \Theta_{20}] = (0.781, 1.44]$	7	20	$\hat{s}$	$7 + \hat{s}$
8	0	$(\frac{1}{2}\Theta_{25}, \Theta_{25}] = (1.215, 2.43]$	8	25	$\hat{s}$	$8 + \hat{s}$
8	1	$(\frac{1}{2}\Theta_{25}, \Theta_{20}] = (1.215, 1.44]$	7	20	$\hat{s}$	$7 + \hat{s}$
8	1	$(\Theta_{20}, \Theta_{25}] = (1.44, 2.43]$	8	25	$\hat{s}$	$8 + \hat{s}$
8	2	$(\frac{1}{2}\Theta_{25}, \Theta_{20}] = (1.215, 1.44]$	7	20	$\hat{s}$	$7 + \hat{s}$
8	2	$(\Theta_{20}, 2\Theta_{16}] = (1.44, 1.562]$	6	16	$\hat{s} + 1$	$7 + \hat{s}$
8	2	$(2\Theta_{16}, \Theta_{25}] = (1.562, 2.43]$	8	25	$\hat{s}$	$8 + \hat{s}$
9	0	$(\frac{1}{2}\Theta_{30}, \Theta_{30}] = (1.77, 3.54]$	9	30	$\hat{s}$	$9 + \hat{s}$
9	1	$(\frac{1}{2}\Theta_{30}, \Theta_{25}] = (1.77, 2.43]$	8	25	$\hat{s}$	$8 + \hat{s}$
9	1	$(\Theta_{25}, \Theta_{30}] = (2.43, 3.54]$	9	30	$\hat{s}$	$9 + \hat{s}$
9	2	$(\frac{1}{2}\Theta_{30}, \Theta_{25}] = (1.77, 2.43]$	8	25	$\hat{s}$	$8 + \hat{s}$
9	2	$(\Theta_{25}, 2\Theta_{20}] = (2.43, 2.88]$	7	20	$\hat{s} + 1$	$8 + \hat{s}$
9	2	$(2\Theta_{20}, 4\Theta_{16}] = (2.88, 3.124]$	6	16	$\hat{s} + 2$	$8 + \hat{s}$
9	2	$(4\Theta_{16}, \Theta_{30}] = (3.124, 3.54]$	9	30	$\hat{s}$	$9 + \hat{s}$

Let $\hat{k}$ denote a number of matrix products, with $\hat{m}$, $\hat{q}$, and $\hat{r}$ the associated values, so that $\hat{m} = \hat{q}\hat{r}$. From Algorithms PS-coeff and HPS-eval, the $\hat{q}$ highest-degree terms in $T_{\hat{m}}(A)$ can be expressed as

$$H_{\hat{q},\hat{r}}(A) = \bar{B}_{\hat{r}-1}(A^{\hat{q}})^{\hat{r}-1}. \tag{3.3}$$

These terms need not be formed or included in the Taylor polynomial if

$$\|e^{-A}H_{\hat{q},\hat{r}}(A)\| \leq u, \tag{3.4}$$

where u is unit roundoff, since then

$$\|H_{\hat{q},\hat{r}}(A)\| = \|e^{A}e^{-A}H_{\hat{q},\hat{r}}(A)\| \leq \|e^{A}\|u. \tag{3.5}$$

In this case, the desired accuracy can be achieved by omitting these $\hat{q}$ terms, thereby using $\hat{k} - 1$ matrix products to form a Taylor polynomial of order $m_{\hat{k}-1} = (\hat{r} - 1)\hat{q}$, since, using (3.5), it follows that the difference between $T_{m_{\hat{k}-1}}(A)$ and $T_{m_{\hat{k}}}(A)$ satisfies

$$\|T_{m_{\hat{k}-1}}(A) - T_{m_{\hat{k}}}(A)\|/\|e^{A}\| = \|H_{\hat{q},\hat{r}}(A)\|/\|e^{A}\| \leq u, \tag{3.6}$$

and then it is not significant for machine precision. A practical test of whether (3.4) holds can be based on obtaining bounds on the separate factors of $\|H_{\hat{q},\hat{r}}(A)\|$:

$$\|H_{\hat{q},\hat{r}}(A)\| \leq \|\bar{B}_{\hat{r}-1}\|\|A^{\hat{q}}\|^{\hat{r}-1}, \tag{3.7}$$

where computing the norm of $\bar{B}_{\hat{r}-1}$ does not involve any matrix products beyond those needed to compute the coefficient itself. Thus satisfaction of the following relationship shows that the desired accuracy can be achieved while reducing the number of matrix products by one:

$$\|e^{-A}\|\|\bar{B}_{\hat{r}-1}\|\|A^{\hat{q}}\|^{\hat{r}-1} \leq u. \tag{3.8}$$

If $\|A\| \leq \Theta_{m_{\hat{k}}}$, since $\|-A\| = \|A\|$, then $e^{-A} \approx T_{m_{\hat{k}}}(-A)$. Thus, using (2.7) we can obtain the following bound for $\|e^{-A}\|$, denoted by b_{exp},

$$\|e^{-A}\| \approx \|T_{m_{\hat{k}}}(-A)\| \leq b_{\text{exp}} = \|b_0 I + \hat{B}_0\| + \sum_{l=1}^{r-1} \|\hat{B}_l\| \|A^{\hat{q}}\|^l, \tag{3.9}$$

where $\hat{B}_l = \sum_{j=1}^{q} (-1)^{ql+j} b_{ql+j} A^j$, $l = 0, \ldots, r-1$. Bound (3.9) can be evaluated with no matrix products reusing the matrix powers A^j, $j = 2, 3, \ldots, \hat{q}$, computed for evaluating $T_{m_{\hat{k}}}(A)$ in the following algorithm.

```
Algorithm bexp-bound.
% Calculation of bound $b_{\text{exp}}$ of $\|\exp(-A)\|$ from (3.9), with $m = qr$.
% **Input:** $q$; $r$; $\{A_j = A^j\}, j = 1 : q$; $\{b_i = (-1)^i/i!\}, i = 0 : qr$.
Execute Algorithm PS-coeff with $b_i = (-1)^i/i!$  % calculate $\{\hat{B}_l\}, l = 0 : r-1$;
$b_{\text{exp}} = \|\hat{B}_{r-1}\|$;
for  $l = r-1 : -1 : 2$
    $b_{\text{exp}} = \|\hat{B}_{l-1}\| + \|A_q\| \times b_{\text{exp}}$;
end for $l$
$b_{\text{exp}} = \|b_0 I + \hat{B}_0\| + \|A_q\| \times b_{\text{exp}}$;
end % bexp-Bound
```

Note that, since $\|A\| \leq \Theta_{m_{\hat{k}}}$ then

$$b_{\text{exp}} \leq \sum_{i=0}^{m_{\hat{k}}} \|A\|^i / i! = T_{m_{\hat{k}}}(\|A\|) \approx \mathrm{e}^{\|A\|}, \tag{3.10}$$

and b_{exp} is strictly lower than $\mathrm{e}^{\|A\|}$ for some matrices, e.g., for matrix

$$A = \begin{pmatrix} 1.25 & 1.25 \\ 1.25 & 1.25 \end{pmatrix}, \tag{3.11}$$

it follows that $\|A\|_1 = 2.5$, $\Theta_{25} < \|A\|_1 < \Theta_{30}$, and then, Algorithm Order-Scale-2 with $K = 9$ and option $= 1$ gives $m^* = 30$ and $s^* = 0$. Then, from Table 2 it follows that $q = 5$ and $r = 6$. Computing symbolically the exact value of e^{-A} and using the 1-norm in (3.9) gives

$$\|\mathrm{e}^{-A}\|_1 = 1 < b_{\text{exp}} = 1.25 < \mathrm{e}^{\|A\|_1} = 12.18, \tag{3.12}$$

and $b_{\text{exp}} \|\bar{B}_5\|_1 \|A^5\|_1^5 = 7.57e-17 < u \approx 1.11e-16$ (IEEE double precision), showing that (3.8) holds and the number of matrix products can be reduced. However, using bound $\|\mathrm{e}^{-A}\| \leq \mathrm{e}^{\|A\|}$ in (3.8) gives $\mathrm{e}^{\|A\|_1} \|\bar{B}_5\|_1 \|A^5\|_1^5 = 7.39e-16 > u$ and condition (3.8) is not guaranteed. Hence, since bound b_{exp} is not greater than $\mathrm{e}^{\|A\|}$ and can be lower for some matrices, we use it in (3.8) to give the final condition

$$b_{\text{exp}} \|\bar{B}_{\hat{r}-1}\| \|A^{\hat{q}}\|^{\hat{r}-1} \leq u. \tag{3.13}$$

Similar tests can be devised and applied recursively, eliminating sets of $\hat{q}$ terms each time: if $b_{\text{exp}} \|\bar{B}_{\hat{r}-j}\| \|A^{\hat{q}}\|^{\hat{r}-j} \leq u$ with $j = 2$, then the number of matrix products can be reduced to $\hat{k} - 2$ computing a Taylor polynomial of order $\hat{m} = (\hat{r} - 2)\hat{q}$, and so on with $j = 3, 4, \ldots, \hat{r} - 1$ computing Taylor polynomials of orders $\hat{m} = (\hat{r} - j)\hat{q}$ with $\hat{k} - j$ matrix products.

TABLE 5
k, m_k, Θ_{m_k}, Θ'_{m_k}, and ϑ_{m_k}.

k	m_k	Θ_{m_k}	Θ'_{m_k}	ϑ_{m_k}
1	2	2.5810e-8	8.7334e-6	8.7334e-6
2	4	3.3972e-4	1.6778e-3	1.6778e-3
3	6	9.0657e-3	1.7720e-2	1.7720e-2
4	9	8.9578e-2	1.1354e-1	1.1354e-1
5	12	2.9962e-1	3.2690e-1	3.2690e-1
6	16	7.8029e-1	7.8738e-1	7.8738e-1
7	20	1.4383e0	1.4070e0	1.4383e0
8	25	2.4286e0	2.3392e0	2.4286e0
9	30	3.5397e0	3.3908e0	3.5397e0

The next theorem establishes that (3.8) holds for matrices whose norm lies in certain intervals, so the number of matrix products can be reduced for those matrices.

THEOREM 3.1. *Let $m_k = q_k r_k$, where $q_k = r_k = (k+2)/2$ for even $k > 0$, and $q_k = (k+1)/2$ and $r_k = q_k + 1$ for odd $k > 0$. Let $A \in \mathbb{C}^{n\times n}$, let $H_{q_k,r_k}(A)$ be the q_k highest-degree terms in $T_{m_k}(A) = \sum_{i=0}^{m_k} A^i/i!$, let u be the relative machine precision, and let $\Theta'_{m_{k-1}}$ be the values such that*

$$\Theta'_{m_{k-1}} = \max\{\theta, e^{\theta} H_{q_k,r_k}(\theta) \le u\}. \tag{3.14}$$

Then, if $\|A\| \le \Theta'_{m_{k-1}}$ then condition (3.8) *holds and $\|T_{m_k}(A) - T_{m_{k-1}}(A)\|/\|e^A\| \le u$.*

Proof. Using (3.8), (3.3), (2.7), and (3.14), since $\|\mathrm{e}^{-A}\| \le \mathrm{e}^{\|A\|}$, if $\|A\| \le \Theta'_{m_{k-1}}$ it follows that

$$\|\mathrm{e}^{-A}\|\|\bar{B}_{r_k-1}\|\|A^{q_k}\|^{r_k-1} \le \mathrm{e}^{\|A\|} H_{q_k,r_k}(\|A\|) \le \mathrm{e}^{\Theta'_{m_{k-1}}} H_{q_k,r_k}(\Theta'_{m_{k-1}}) \le u, \tag{3.15}$$

and condition (3.8) holds. Hence, from (3.3)–(3.5) it follows that (3.6) holds. □

Using a zero finder we computed the values Θ'_{m_k}, $k = 1, \dots, 9$, presented in Table 5. Note that $\Theta'_{m_k} > \Theta_{m_k}$ for $k = 1, \dots, 6$. Thus, if k and $\|A\|$ are such that $\Theta_{m_{k-1}} < \|A\| \le \Theta'_{m_{k-1}}$, then, Theorem 3.1 states that we can omit the last q terms of $T_{m_k}(A)$ and compute Taylor polynomial $T_{m_{k-1}}(A)$ instead, reducing the number of matrix products by one. Hence, taking $\vartheta_{m_k} = \max\{\Theta_{m_k}, \Theta'_{m_k}\}$, see Table 5, and substituting Θ_{m_k} with ϑ_{m_k} in Algorithm Order-scale-2, the number of matrix products is reduced with respect to the original Algorithm Order-scale-2 for matrices with $\Theta_m < \|A\| \le \Theta'_m$, $m = 2, 4, 6, 9, 12, 16$, and $\Theta_{16} < \|A\|/2^{s^*} \le \Theta'_{16}$, where s^* is the scaling parameter obtained by Order-scale-2 with the new ϑ_{m_k} values.

Taking into account the new values ϑ_{m_k} from Table 5 and proceeding in a way similar to [SIDR09], the intervals of $\|A\|$, where a Taylor approximation requires fewer matrix products than Padé approximation, increase with respect to those given in (2.6): $\|A\| \le 0.11$ (ϑ_9) and 0.25 (θ_5) $< \|A\| \le 0.33$ (ϑ_{12}).

Note that $\Theta_{m_k} > \Theta'_{m_k}$ for $m_k = 20, 25, 30$. However, using (3.10) it follows that condition (3.13) is less restrictive than (3.15), and the example was given above of the matrix from (3.11) where condition (3.13) holds with $m = 30$ for $\|A\|_1 = 2.5 > \Theta_{25} > \Theta'_{25}$.

Algorithm HPS-eval-2 is analogous to HPS-eval, but implements the performance of bound tests based on (3.13) to reduce the number of matrix products.

```
ALGORITHM HPS-EVAL-2.
% Horner–Paterson–Stockmeyer evaluation of the order-$m$ Taylor polynomial
% of $\exp(A)$ with $m = qr$, performing bound tests to check if the higher-order
% terms of the Taylor series can be neglected.
% **Input:** $s;\ q;\ r;\ \{b_i\}, i = 0 : qr;\ \{A_j = A^j\}, j = 1 : q;\ \{\bar{B}_k\}, k = 0 : r - 1;\ u$.
% **Output:** $F = T_m(A)$.
Execute Algorithm bexp-Bound; % Compute bound $b_{exp}$
$F = \bar{B}_{r-1}$;
for $j = r - 1 : -1 : 1$
    if $(b_{exp}\|F\|_1\|A_q\|_1^j \le u)$ then
        $F = \bar{B}_{j-1}$;      % Reduction of one matrix product
    else
        $F = \bar{B}_{j-1} + A_q \times F$;
    end if
end for $j$
$F = F + b_0 I$;
end      % HPS-eval-2
```

Taking into account the two proposed modifications, in the next section we test the following versions of the Taylor Algorithm GSQT presented in section 2.2:

- TSTD (Taylor standard) uses Algorithm Order-scale-2 with option 0 and Taylor-eval. Cost $k^* + s^*$ matrix products (see Order-scale-2).
- TPS (Taylor Paterson–Stockmeyer) uses Algorithm Order-scale-2 with option 1 and Taylor-eval. Cost $k^* + s^*$ matrix products.
- OTPS (Optimal TPS) uses Algorithm Order-scale-2 with option 2 and Taylor-eval. Cost $k^* + s^*$ matrix products.
- TPSBT (TPS performing Bound Tests) is analogous to TPS except for the use of the new ϑ_m values in Algorithm Order-scale-2, and the Algorithm HPS-eval-2 instead of HPS-eval in Taylor-eval to perform the bound tests. Cost less than or equal to $k^* + s^*$ matrix products.
- OTPSBT (Optimal TPSBS) is analogous to TPSBT using option 2 in Algorithm Order-scale-2. Cost less than or equal to $k^* + s^*$ matrix products.

Finally, we analyze the rounding error for the proposed Taylor algorithms. The effect of rounding errors in the evaluation of the Taylor matrix polynomial can be bounded similarly to the numerator of Padé approximants [Hig05, p. 1185]. Since $e^{-\|A\|} \le \|e^A\|$, using Theorem 2.2 of [Hig05, p. 1184] with $\|A\|_1 \le \upsilon_m$, where $\upsilon_m = \Theta_m$ for TSTD, TPS, and OTPS, and $\upsilon_m = \vartheta_m$ for TPSBT and OTPSBT, and noting that all the coefficients in the Taylor matrix polynomial are positive, it follows that

$$\begin{aligned}\left\|T_m(A) - \hat{T}_m(A)\right\|_1 &\le \widetilde{\gamma}_{mn} T_m\left(\|A\|_1\right) \approx \widetilde{\gamma}_{mn} e^{\|A\|_1} \le \widetilde{\gamma}_{mn}\left\|e^A\right\|_1 e^{2\|A\|_1} \\ &\simeq \widetilde{\gamma}_{mn}\left\|T_m(A)\right\|_1 e^{2\|A\|_1} \le \widetilde{\gamma}_{mn}\left\|T_m(A)\right\|_1 e^{2\upsilon_m}, \qquad (3.16)\end{aligned}$$

where $A \in \mathbb{C}^{n\times n}$, $\hat{T}_m(A)$ is the computed Taylor approximation, and $\widetilde{\gamma}_k = cku/(1 - cku)$, where c is a small integer constant [Hig02]. Hence, the relative error is bounded approximately by $\tilde{\gamma}_{mn} e^{2\upsilon_m}$, which is a satisfactory bound taking into account the values of Θ_m and ϑ_m given in Table 5 (see [Hig05, p. 1185]), and there is no rounding error contribution of solving multiple linear systems as in Padé methods.

Table 6

Total number of matrix products P computed by each function to evaluate the exponentials of all the test matrices from Case Study 1.

m_K	TSTD	TPS	OTPS	TPSBT	OTPSBT	max. m **expm**	`expm`
20	724	719	719	668	668	13	635.67
25	739	724	719	673	672	17	637.67
30	757	739	719	677	672	21	649.67

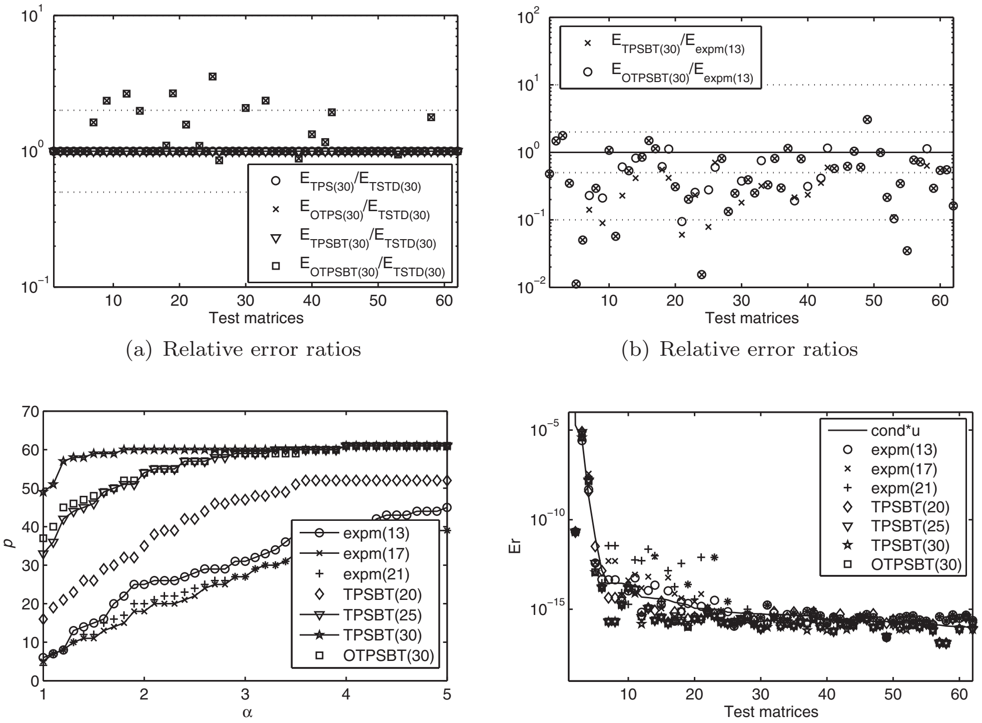
4. Numerical experiments.

MATLAB implementations of Algorithms TPS, TPSBT, OTPS, and OTPSBT with $m_K = 20$, 25, and 30 were compared to the standard Taylor algorithm TSTD, and the MATLAB scaling and squaring Padé function **expm** with maximum orders $m = 13$, 17, and 21; see [Hig05]. Tests were done on an 2.8 GHz Intel Xeon processor with 4 GB main memory and MATLAB 7.7. Accuracy was tested by computing relative error $\mathrm{E} = \|\mathrm{e}^A - \tilde{X}\|_1 / \|\mathrm{e}^A\|_1$, where $\tilde{X}$ is the computed solution, and the "exact" value e^A was computed using the Symbolic Math Toolbox of MATLAB and a [33/33] Padé method with scaling and squaring, at 1000-digit decimal arithmetic for Case Study 1 (small matrices), 250-digit decimal arithmetic for Case Study 2 (matrices 50 × 50), and quadruple precision in Case Study 2 (matrices 500 × 500). An extra output parameter P was added to all tested functions to return the number of matrix products. 4/3 products were added in function **expm** for each solution of a multiple right-hand side linear system [BD99].

4.1. Case Study 1.

In this test we considered the same test matrices as in [Hig05], except for three matrices where **expm** gave infinite results in MATLAB, i.e., matrices 17 "ipjfact," 42 "invhilb," and 44 "pascal" with size 8 × 8, from the **matrix** function given in the Matrix Function Toolbox [Hig08, Appendix D], and matrix 43 "magic," which was repeated as matrix 49. Table 6 shows the total number of matrix products P computed by each function to evaluate the matrix exponential of the 62 test matrices. It shows that the proposed Taylor algorithms, and especially TPSBT and OTPSBT, need fewer matrix products than the standard Taylor algorithm TSTD, and between 5.08% and 6.50% more matrix products than **expm**. The cost of OTPSBT is very similar for $m_K = 20, 25, 30$, as expected; see Algorithm Order-scale-2. Thus, from now on we consider only $m_K = 30$ for OTPSBT. Figure 1(a) shows the relative error ratios of all proposed Taylor functions with respect to the standard TSTD for $m_K = 30$. This figure shows that the TPS and TPSBT errors are similar to that of the standard TSTD for all matrices. However, OTPS and OTPSBT errors are greater for some matrices, confirming that the extra squaring in both algorithms can lead to more numerical error; see subsection 3.1. Figure 1(a) also shows that TPS error and TPSBT error are similar for all matrices. The same happens with OTPS and OTPSBT. This fact is supported by (3.6), which shows that the norm of the neglected terms in functions TPSBT and OTPSBT, relative to the exact value $\exp(A)$, is not significant for machine precision.

Figure 1(b) presents the relative error ratios for the most efficient Taylor functions TPSBT and OTPSBT with $m_K = 30$, and **expm** with optimal maximum order $m = 13$ [Hig05]. Figure 1(b) shows that OTPSBT and TPSBT displayed error comparable to **expm**. Figure 1(c) shows the performances [DM02] of TPSBT and OTPSBT, and **expm** with maximum orders $m = 13, 17, 21$ (see [Hig05]), where the α coordinate varies between 1 and 5 in steps equal to 0.1, and the p coordinate is the probability that the considered algorithm 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.



(a) Relative error ratios

(b) Relative error ratios

(c) Performance profile

(d) Normwise relative error

FIG. 1. *Comparison results, Case Study* 1.

Figure 1(c) shows that in this case study the most accurate function was TPSBT with $m_K = 30$ followed by OTPSBT with $m_K = 30$ confirming that the extra squaring in OTPSBT can yield a lower accuracy.

To test numerical stability we plotted the normwise relative errors of the considered functions. Figure 1(d) shows the relative errors of all implementations, and a solid line that represents the unit roundoff multiplied by the relative condition number of the exponential function at X [Hig08, 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** from the Matrix Function Toolbox [Hig08, Appendix D] (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 [Hig05, p. 1188]. Figure 1(d) shows that all functions performed in a numerically stable way.

4.2. Case Study 2. In this case study 39 matrices of dimension $n = 50$ and 36 matrices of dimension $n = 500$ from MATLAB function **matrix** in the Matrix Computation Toolbox were used as the test battery (matrices whose exponential cannot be represented in double precision because overflow errors were excluded from the 52 total possible matrices). Fortran versions of functions TSTD, TPS, OTPS,

Table 7

Total number of matrix products P computed by each function to evaluate the exponentials of all 50×50 *matrices from Case Study* 2.

m_K	TSTD	TPS	OTPS	TPSBT	OTPSBT	max. **expm** order	`expm`
20	469	465	465	430	430	13	408.67
25	479	469	465	432	431	17	412.33
30	487	479	465	429	431	21	416.33

Table 8

Total number of matrix products P to compute the exponential of all 500×500 *matrices from Case Study* 2. *Mean and standard deviation of total execution time t in seconds for* 100 *repetitions of the experiment, with* $m_K = 30$ *in Taylor functions, and the optimal maximum order* $m = 13$ *in* ***expm****; see* [Hig05].

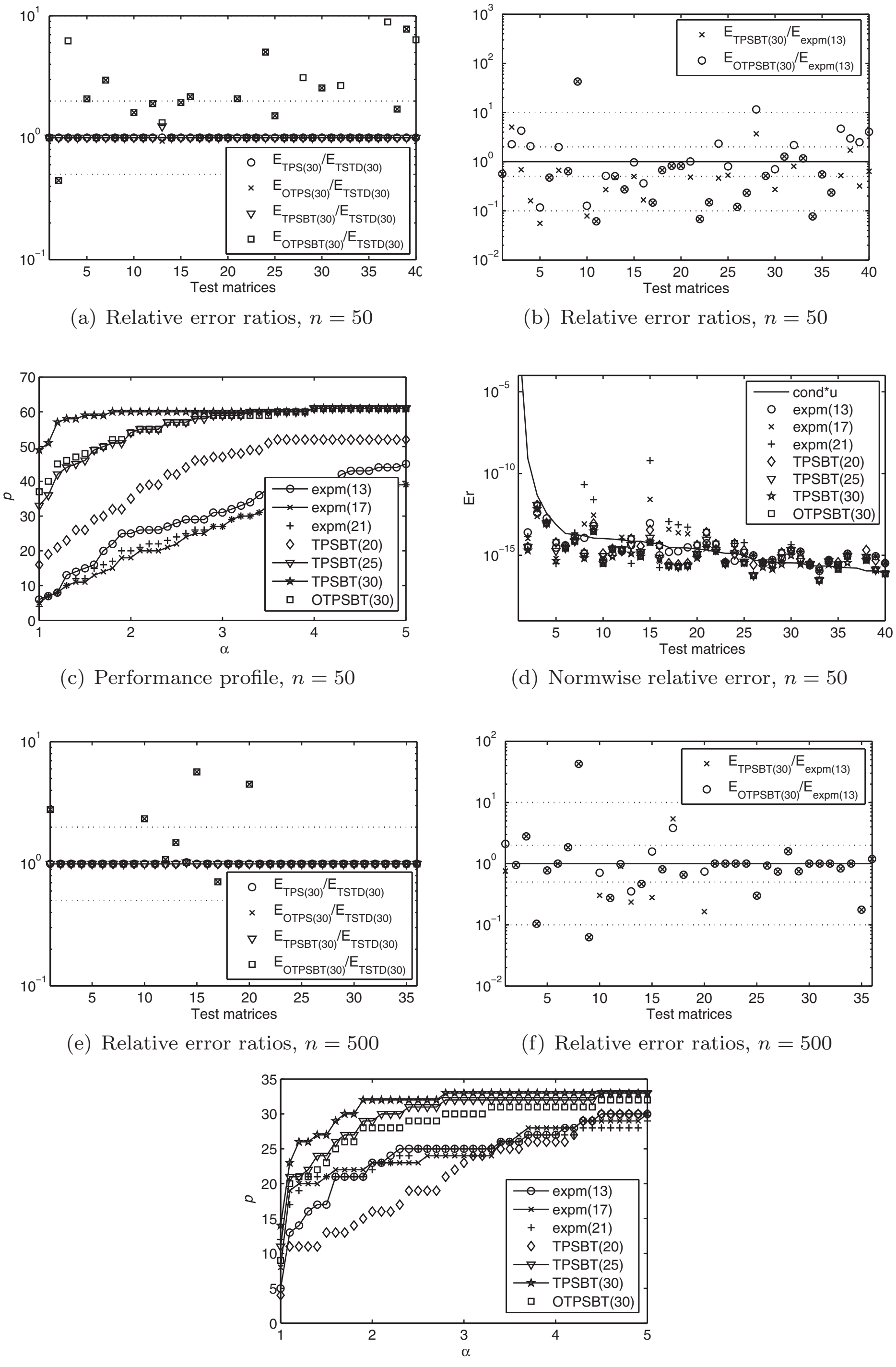
	TSTD	TPS	OTPS	TPSBT	OTPSBT	`expm`
P	506	487	479	440	439	425
mean(t) (seconds)	829.97	787.39	782.67	692.28	694.59	762.58
standard deviation(t)	2.51	2.31	2.33	2.11	1.99	2.26

TPSBT, OTPSBT with $m_K = 30$ and **expm** with maximum order $m = 13$ [Hig05] were implemented, and made available online in [FORT], to measure execution times for the 500×500 matrices. Tables 7 and 8 present the total number of matrix products P computed to evaluate the exponential of all matrices, and Table 8 also presents the mean and standard deviation of the total execution time t in seconds to compute the exponential of all 500×500 matrices with 100 repetitions of the experiment. We omitted plotting the normwise relative errors for the 500×500 matrices because they were too large to compute the relative condition number.

Similar conclusions to those from Case Study 1 are obtained from Tables 7 and 8 and Figure 2: TPSBT and OTPSBT with $m_K = 30$ had a lower cost than TSTD, and between 3.29% and 5.46% more matrix products than **expm** with maximum order $m = 13$, and TPSBT and OTPSBT execution times for the 500×500 matrices were 9.21% and 8.91% lower than **expm**, respectively. TPS and TPSBT errors were similar to TSTD error in all cases. OTPS and OTPSBT errors were greater than TSTD error in some cases, as expected because of the extra squaring. In the majority of tests TPS and OTPS had similar errors to TPSBT and OTPSBT, respectively, as supported by (3.6). The performance profiles show that TPSBT with $m_K = 30$ was the most accurate function in both the 50×50 and the 500×500 matrices.

5. Conclusions. This work developed four new Taylor algorithms TPS, OTPS, TPSBT, and OTPSBT to compute the matrix exponential based on two modifications to the standard algorithm that reduce computational cost and preserve accuracy. The first modification changes the order and the scaling parameter of the standard scaling algorithm if another combination of these values reduces the number of matrix products. The second modification neglects higher-order terms in the Taylor series based on relative error bounds. Finally, a detailed comparison of the several algorithmic variants was provided.

Acknowledgment. We are grateful for the referees' valuable suggestions.



(a) Relative error ratios, $n = 50$

(b) Relative error ratios, $n = 50$

(c) Performance profile, $n = 50$

(d) Normwise relative error, $n = 50$

(e) Relative error ratios, $n = 500$

(f) Relative error ratios, $n = 500$

(g) Performance profile, $n = 500$

FIG. 2. *Comparison results, Case Study* 2.

REFERENCES

[BD99] S. Blackford and J. Dongarra, *Installation Guide for LAPACK*, Technical report, LAPACK Working Note 411, Department of Computer Science University of Tennessee, Knoxville, TN, 1999.

[CM02] S.M. Cox and P.C. Matthews, *Exponential time differencing for stiff systems*, J. Comput. Phys., 176 (2002), pp. 430–455.

[DM02] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[FORT] Fortran Versions of Functions TSTD, TPS, OTPS, TPSBT, OTPSBT, and **expm**, http://personales.upv.es/∼jorsasma/FORTRAN.zip

[GV96] G.H. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Stud. Math. Sci., The Johns Hopkins University Press, Baltimore, MD, 1996.

[Hig02] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[Hig05] N.J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 1179–1193.

[Hig08] N.J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.

[HiAM10] N.J. Higham and A.H. Al-Mohy, *Computing matrix functions*, Acta Numer., 19 (2010), pp. 159–208.

[HLS98] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.

[IHAR09] J. Ibáñez, V. Hernández, E. Arias, and P. Ruiz, *Solving initial value problems for ordinary differential equations by two approaches: BDF and piecewise-linearized methods*, Comput. Phys. Comm., 180 (2009), pp. 712–723.

[KT05] A.-K. Kassam and L.N. Trefethen, *Fourth-order time-stepping for stiff PDEs*, SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.

[MV03] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later**, SIAM Rev., 45 (2003), pp. 3–49.

[PCK91] P.H. Petkov, N.D. Christov, and M.M. Konstantinov, *Computational Methods for Linear Control Systems*, Prentice Hall, Hertfordshire, UK, 1991.

[PS73] 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.

[War77] R.C. Ward, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM J. Numer. Anal., 14 (1977), pp. 600–610.

[Wes90] D. Westreich, *A practical method for computing the exponential of a matrix and its integral*, Commun. Appl. Numer. Methods, 6 (1990), pp. 375–380.

[SIDR09] J. Sastre, J. Ibáñez, E. Defez, and P. Ruiz, *Efficient Scaling-Squaring Taylor Method for Computing the Matrix Exponential*, http://personales.upv.es/∼jorsasma/076320.pdf (2009).