Aggressively truncated Taylor series method for accurate computation of exponentials of essentially nonnegative matrices

By

Meiyue Shao, Weiguo Gao, and Jungong Xue

UMINF-12/04

UMEÅ UNIVERSITY
DEPARTMENT OF COMPUTING SCIENCE
SE-901 87 UMEÅ
Sweden

Aggressively truncated Taylor series method for accurate computation of exponentials of essentially nonnegative matrices*

Meiyue Shao†, Weiguo Gao‡, and Jungong Xue‡

Abstract

Small relative perturbations to the entries of an essentially nonnegative matrix introduce small relative errors to entries of its exponential. It is thus desirable to compute the exponential with high componentwise relative accuracy. Taylor series approximation coupled with scaling and squaring is used to compute the exponential of an essentially nonnegative matrix. An a priori componentwise relative error bound of truncation is established, from which one can choose the degree of Taylor series expansion and the scale factor so that the exponential is computed with desired componentwise relative accuracy. To reduce the computational cost, the degree of the Taylor series expansion is chosen small, while the scale factor is chosen sufficiently large to achieve the desired accuracy. The rounding errors in the squaring stage are not serious as squaring is forward stable for nonnegative matrices. We also establish a posteriori componentwise error bounds and derive a novel interval algorithm for the matrix exponential. Rounding error analysis and numerical experiments demonstrate the efficiency and accuracy of the proposed methods.

Key words. Matrix exponential, Taylor series, essentially nonnegative matrix, high relative accuracy algorithms

AMS subject classifications. 65F60, 65G20

1 Introduction

The matrix exponential is one of the most well-studied matrix functions and has many applications in physics, biology, finance and engineering, especially those related to the solution of dynamical systems. Many methods for computing the matrix exponential have been proposed, see the classical paper [18] and the monograph [14] for a survey. In addition, there also exist interval algorithms for the matrix exponential problem, see, e.g., [6, 11, 22, 23]. The scaling and squaring method coupled with Padé approximation is the most popular method for calculating the matrix exponential [18] and used in MATLAB (function `expm`) as a general purpose solver. As this method ensures no more than normwise backward stability, the computed exponential has small relative errors in norm if the condition number for normwise perturbation is well bounded. Even in this case the componentwise relative accuracy of the computed exponential can not be guaranteed, as tiny entries of the exponential may be computed with low or even no relative accuracy at all. However, it turns out to be quite

*A pre-print appears as Report UMINF-12.04. An early version of this manuscript was entitled "Componentwise high relative accuracy algorithms for the exponential of an essentially nonnegative matrix". The scientific responsibility rests with the authors.

†Department of Computing Science and HPC2N, Umeå University, SE-901 87 Umeå, Sweden (myshao@cs.umu.se). Current address: MATHICSE, EPF Lausanne, CH-1015 Lausanne, Switzerland. The work of M. Shao is supported by the Swedish Research Council under grant VR 70625701, UMIT Research Lab via an EU Mål 2 project, and eSSENCE, a strategic collaborative e-Science programme funded by the Swedish Research Council.

‡School of Mathematical Sciences and Laboratory of Mathematics for Nonlinear Science, Fudan University, Shanghai 200433, China ({wggao,xuej}@fudan.edu.cn). The work of W. Gao is supported by the National Natural Science Foundation of China under grant 11071047, the Science and Technology Commission of Shanghai Municipality under grant 09ZR1401900 and MOE Key Laboratory of Computational Physical Sciences. The work of J. Xue is partly supported by the National Natural Science Foundation of China under grant 10970136.

common for exponentials of matrices to have some very small entries relative to others. For example, the entries of the exponential of a sparse matrix can have great variations in their order of magnitude.

A real $N\times N$ matrix is called *essentially nonnegative* if all its off-diagonal entries are nonnegative [5, 14]. This type of matrices is also widely known as *Metzler matrices* [12, 20]. The exponentials of essentially nonnegative matrices are componentwise nonnegative and have important applications in continuous-time Markov processes and positive linear dynamical systems [12, 20, 29]. Recently, the exponential of the adjacency matrix (which is symmetric and nonnegative) is shown to be involved in the measurement analysis of networks [9]. It is shown [35] that small relative perturbations to the entries of an essentially nonnegative matrix introduce small relative errors to entries of its exponential. It is thus desirable to compute the exponential with high componentwise relative accuracy. In addition, in some applications it is required to compute all entries of the exponential accurately, see [36].

The computation of the exponential of an essentially nonnegative matrix has been studied in, e.g., [17, 22, 23, 36]. In [36] two algorithms are developed to compute all the entries of the exponential of an essentially nonnegative matrix accurately. Both algorithms are in the framework of the scaling and squaring method. More detailed, they are based on the approximation $\mathrm{e}^A \approx \mathrm{e}^{\alpha}\big[T\big((A-\alpha I)/n\big)\big]^n$, where α is a shift such that $B := A-\alpha I$ is componentwise nonnegative, n is a power of 2 and $T(x)$ is some polynomial approximating e^x. In both algorithms, n is chosen to satisfy $\rho(B)/n < 1$, where $\rho(\cdot)$ denotes the spectral radius of a square matrix, and the desirable componentwise accuracy is achieved by making $T(B/n)$ approximate $\mathrm{e}^{B/n}$ with high componentwise relative accuracy. Both algorithms require $\mathcal{O}(N)$ matrix multiplications in $T(B/n)$ when A is sparse (e.g., A is narrow banded). However, high componentwise relative accuracy of $T(B/n)$ to $\mathrm{e}^{B/n}$ is not necessary for high componentwise relative accuracy of $\mathrm{e}^{\alpha}\big[T(B/n)\big]^n$ to e^A. Just consider the fact that

$$\lim_{n\to\infty}\Big(I+\frac{B}{n}\Big)^n = \mathrm{e}^B.$$

Though $I+B/n$ may have no componentwise relative accuracy to $\mathrm{e}^{B/n}$ at all, $(I+B/n)^n$ can have high componentwise relative accuracy to e^B with n sufficiently large. This suggests us to select the scale factor n a bit large to improve the efficiency of the algorithm. Let

$$s(A) = \min_i A(i,i), \quad \widehat{A} = A - s(A)I. \tag{1.1}$$

In this paper we always adopt the shift $\alpha = s(A)$ and use

$$L_{m,n}(A) = \mathrm{e}^{s(A)}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \tag{1.2}$$

to approximate e^A. Here $T_m(x)$ is the m-th order Taylor expansion of e^x, i.e.,

$$T_m(x) = \sum_{k=0}^{m}\frac{x^k}{k!} = 1 + x + \cdots + \frac{x^{m-1}}{(m-1)!} + \frac{x^m}{m!}. \tag{1.3}$$

We provide an a priori componentwise relative error bound of $L_{m,n}(A)$ to e^A, which suggests that in exact arithmetic the desired accuracy can be achieved by making either n or m sufficiently large. In existing scaling and squaring methods for exponentials of general matrices, n is chosen small to avoid potentially serious rounding errors in the squaring procedure. However, the squaring procedure is forward stable for nonnegative matrices [2, 10, 14], which suggests that when A is essentially nonnegative, the scale factor n could be chosen a bit large to improve the efficiency of the algorithm without sacrificing the accuracy. In this spirit, based on the a priori componentwise relative error bound we present a strategy to choose m and n such that only $\mathcal{O}(\log N)$ matrix multiplications are needed for desired componentwise relative accuracy, while the componentwise error caused by inexact arithmetic is comparable to that already made in rounding A to its floating point representation $\mathrm{fl}(A)$. In our algorithm, m is chosen small and $T_m\big(\widehat{A}/n\big)$ may have no componentwise relative accuracy to $\mathrm{e}^{\widehat{A}/n}$. In this sense, we call it *aggressively truncated Taylor series expansion.*

Throughout this paper, we use $\mathbb{R}_+ = \{x \in \mathbb{R}\colon x \geq 0\}$ to denote the set of all nonnegative real numbers. For any matrices A, $B \in \mathbb{R}^{M\times N}$, we denote by $A \geq B$ if $A(i,j) \geq B(i,j)$ for all i and j. $|A|$ is the matrix with entries $|A(i,j)|$, and we let $\mathrm{struct}(A)$ denote the nonzero pattern of any given matrix A, i.e.,

$$\mathrm{struct}(A) = \{(i,j)\colon A(i,j) \neq 0\}.$$

For any $A \in \mathbb{R}^{N\times N}$, let $s(A)$ and $\widehat{A}$ be defined as in (1.1). Then A is essentially nonnegative if and only if $\widehat{A} \geq 0$. Concerning the floating point arithmetic, we denote by $R_{\max}$ and $R_{\min}$ the largest and smallest normalized positive finite floating point numbers, respectively, on a certain architecture. $\mathrm{fl}(x)$ is used to represent the quantity x calculated with floating point arithmetic, where x can be either a number or a matrix. Rounding error analyses are based on the standard rounding model [13]

$$\mathrm{fl}(a \circ b) = (a \circ b)(1+\epsilon), \quad |\epsilon| \leq \mathbf{u}, \tag{1.4}$$

where "$\circ$" is $+$, $-$, $\times$, or $\div$, and $\mathbf{u}$ is the unit roundoff, as long as there are no overflows or (gradual) underflows in the calculation. Sometimes we are able to obtain true lower bounds and upper bounds of x by switching the rounding mode [19] during the computation. The corresponding bounds are denoted by $\underline{\mathrm{fl}}(x)$ and $\overline{\mathrm{fl}}(x)$, respectively. Then these bounds (so long as they exist) satisfy

$$\underline{\mathrm{fl}}(x) \leq x \leq \overline{\mathrm{fl}}(x). \tag{1.5}$$

The rest of this paper is organized as follows. In Section 2, we discuss the conditioning for the computation of exponentials of essentially nonnegative matrices. The a priori componentwise relative error bound is obtained in Section 3.1. Section 3.2 develops the aggressive truncated Taylor series method based on the componentwise truncation error bound. Some further variants of the algorithm are discussed in Section 4. Finally in Section 5, numerical examples are presented to demonstrate the efficiency and accuracy of the algorithm.

2 Conditioning

For a given essentially nonnegative matrix A, we would like to compute every entry of e^A accurately. Therefore, a lot of existing normwise condition estimates (e.g., see [16, 17, 31]) are not sufficient for this purpose. Recently, it is shown in [35] that small relative errors in the entries of A cause small relative errors in the entries of e^A. This property guarantees that the componentwise relative error produced by rounding A to $\mathrm{fl}(A)$ is small and thus makes it possible to compute e^A with high componentwise relative accuracy.

Theorem 2.1. [35] *Assume A is an $N \times N$ essentially nonnegative matrix. Let $\Delta A \in \mathbb{R}^{N\times N}$ be a perturbation to A satisfying $|\Delta A(i,j)| \leq \varepsilon_1 A(i,j)$ $(i \neq j)$ and $|\Delta A(i,i)| \leq \varepsilon_2$, where $0 \leq \varepsilon_1 < 1$, $\varepsilon_2 \geq 0$. Then for any $t \geq 0$ we have*

$$\left|\mathrm{e}^{t(A+\Delta A)} - \mathrm{e}^{tA}\right| \leq \delta(t)\,\mathrm{e}^{\delta(t)}\mathrm{e}^{tA},$$

where

$$\delta(t) = t\varepsilon_2 + \left[N - 1 + \rho(\widehat{A})t\right]\frac{\varepsilon_1}{1-\varepsilon_1}.$$

Here we are interested in the case $t = 1$. We denote

$$C(A) = N - 1 + \rho(\widehat{A}) \tag{2.1}$$

as the *condition number* of e^A, in the sense of relative componentwise perturbation. This condition number depends on the quantity $\rho(\widehat{A})$. The following result provides a bound for $\rho(\widehat{A})$ under the assumption that there is no overflow in $\mathrm{fl}\big(\mathrm{e}^A\big)$.

Theorem 2.2. *Let $A \in \mathbb{R}^{N\times N}$ be essentially nonnegative. If there is no overflow in $\mathrm{fl}(\mathrm{e}^A)$, then*

$$\rho(\widehat{A}) \le \ln N + \ln R_{\max} - s(A). \tag{2.2}$$

Proof. Let $\|X\|_{\max} := \max_{i,j} |X(i,j)|$. Then

$$R_{\max} \ge \left\|\mathrm{e}^A\right\|_{\max} \ge \frac{1}{N}\left\|\mathrm{e}^A\right\|_\infty = \frac{\mathrm{e}^{s(A)}}{N}\left\|\mathrm{e}^{\widehat{A}}\right\|_\infty \ge \frac{\mathrm{e}^{s(A)}}{N}\rho\big(\mathrm{e}^{\widehat{A}}\big) = \frac{\mathrm{e}^{s(A)}}{N}\mathrm{e}^{\rho(\widehat{A})},$$

which implies $s(A) + \rho(\widehat{A}) \le \ln N + \ln R_{\max}$. □

***Remark* 1.** If $s(A)$ is not small, $\rho(\widehat{A})$ cannot be too large. For example, if $A \in \mathbb{R}_+^{10000\times 10000}$ and e^A can be represented by IEEE-754 double precision floating point numbers, then

$$\rho(\widehat{A}) \le \rho(A) \le \ln 10000 + \ln R_{\max} < 719.$$

Therefore the matrix exponential problem is always well-conditioned for an essentially nonnegative matrix A if $s(A)$ is not small, under the assumption that e^A does not cause overflow. Some difficult problems, e.g., when $\mathrm{fl}(\mathrm{e}^A)$ overflows or $|s(A)| = \omega(N)$, are beyond the scope of this paper.

3 The Aggressively Truncated Taylor Series Method

In this section, we develop the main algorithm of this paper, based on aggressive truncation in $T_m(x)$ defined in (1.3). To ensure the feasibility of this approach, some a priori estimates of the truncation error are required, which were not available in many traditional Taylor series approaches [7, 17, 36]. To this end, we begin with establishing these a priori componentwise truncation error estimates.

3.1 Componentwise relative truncation error bounds

Several normwise a priori error bounds for $L_{m,n}(A)$ can been found in some existing literatures (e.g., see [7, 10, 28]). However, to our knowledge, there is no a priori componentwise error bound available for essentially nonnegative matrices. In the following, we provide such an a priori componentwise error bound. We first bound the error $\mathrm{e}^A - L_{m,n}(A)$ in terms of $\widehat{A}^{m+1}\mathrm{e}^A$.

Lemma 3.1. *If $A \in \mathbb{R}^{N\times N}$ is essentially nonnegative, then for any positive integers m and n we have*

$$\lim_{n\to\infty} L_{m,n}(A) = \lim_{m\to\infty} L_{m,n}(A) = \mathrm{e}^A. \tag{3.1}$$

Furthermore, the truncation error is bounded by

$$\mathrm{e}^{s(A)}\frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^{n-1} \le \mathrm{e}^A - L_{m,n}(A) \le \frac{\widehat{A}^{m+1}}{n^m(m+1)!}\mathrm{e}^A. \tag{3.2}$$

Proof. Notice that (3.1) is an immediate consequence of (3.2). Therefore, it is sufficient to verify (3.2) only. Let $R_m(x) = \mathrm{e}^x - T_m(x)$. Using the identity $x^n - y^n = (x-y)\sum_{k=0}^{n-1} x^k y^{n-1-k}$ when $xy = yx$, we have

$$\mathrm{e}^A - L_{m,n}(A) = \mathrm{e}^{s(A)}\left[\mathrm{e}^{\widehat{A}} - T_m\Big(\frac{\widehat{A}}{n}\Big)^n\right] = \mathrm{e}^{s(A)}R_m\Big(\frac{\widehat{A}}{n}\Big)\sum_{k=0}^{n-1}\mathrm{e}^{\frac{k}{n}\widehat{A}}T_m\Big(\frac{\widehat{A}}{n}\Big)^{n-1-k}.$$

Since

$$\frac{(\widehat{A}/n)^{m+1}}{(m+1)!} \le R_m\Big(\frac{\widehat{A}}{n}\Big) \le \frac{(\widehat{A}/n)^{m+1}}{(m+1)!}\mathrm{e}^{\widehat{A}/n}$$

and

$$T_m\Big(\frac{\widehat{A}}{n}\Big) \le \mathrm{e}^{\widehat{A}/n},$$

we obtain

$$\frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^{n-1} \le \mathrm{e}^{\widehat{A}} - L_{m,n}(\widehat{A}) \le \frac{\widehat{A}^{m+1}}{n^m(m+1)!}\mathrm{e}^{\widehat{A}}.$$

which is equivalent to (3.2). □

Lemma 3.1 confirms the convergence of $L_{m,n}(A)$ to e^A and provides a componentwise error bound. But it is still not an a priori componentwise relative truncation error for $L_{m,n}(A)$. To this end, we are to bound $\widehat{A}^{m+1}\mathrm{e}^A$ in terms of e^A and need the following lemma.

Lemma 3.2. [35] *Let $A \in \mathbb{R}_+^{N\times N}$. Then*

$$A\mathrm{e}^A \le \big[N - 1 + \rho(A)\big]\mathrm{e}^A.$$

A direct corollary of Lemma 3.2 is

$$\widehat{A}\mathrm{e}^A = \mathrm{e}^{s(A)}\widehat{A}\mathrm{e}^{\widehat{A}} \le \mathrm{e}^{s(A)}C(\widehat{A})\mathrm{e}^{\widehat{A}} = C(\widehat{A})\mathrm{e}^A = C(A)\mathrm{e}^A$$

for an essentially nonnegative matrix A. Consequently,

$$\widehat{A}^k\mathrm{e}^A \le C(A)^k\mathrm{e}^A, \qquad (k \in \mathbb{N}), \tag{3.3}$$

and more generally, $f(\widehat{A})\mathrm{e}^A \le f[C(A)]\mathrm{e}^A$ if $f(x) = \sum_{k=0}^{\infty} a_k x^k$ has nonnegative coefficients (i.e., $a_k \ge 0$ for $k = 0, 1, \dots$) and $C(A)$ is less than the radius of convergence of $f(x)$. Applying (3.3) to (3.2), we obtain the componentwise truncation error estimate as follows.

Theorem 3.3. *Let $A \in \mathbb{R}^{N\times N}$ be an essentially nonnegative matrix. Then for any positive integers m and n,*

$$0 \le \mathrm{e}^A - L_{m,n}(A) \le \frac{C(A)^{m+1}}{n^m(m+1)!}\mathrm{e}^A. \tag{3.4}$$

3.2 The aggressively truncated Taylor series method

Theorem 3.3 offers a useful a priori componentwise error estimate for the algorithm development. Let τ denote the desired accuracy provided by the user. Then it is natural to choose m and n satisfying

$$\frac{C(A)^{m+1}}{n^m(m+1)!} \le \tau \tag{3.5}$$

and then use $L_{m,n}(A)$ to approximate e^A. Figure 1 shows some sample values of the coefficient in (3.4), where $C(A) = 32$, 128, 512, and 2048, respectively. The computational cost for evaluating $L_{m,n}(A)$ as defined in (1.2) is roughly $m + \log_2 n$ matrix multiplications. Loosely speaking, if we would like to minimize the computational cost, it is sensible to choose the tangent line with slope equals one and pick the point of tangency on the contour line. A better criterion for choosing the parameters will be presented soon.

***Remark* 2.** From the plots in Figure 1, the optimal value (for the worst case) of both m and $\log_2 n$ are of the magnitude 10^1 for a wide range of matrices. Preferable values on m and $\log_2 n$ are close to the boundary of the white area.

Some similar suggestions for choosing m and n have been proposed in [18] based on normwise error estimates. Here we conclude that both m and $\log_2 n$ are still small even if componentwise accuracy is desired. For example, if $m = 13$ is chosen, $n = 4C(A)$ is usually sufficient for attaining double precision accuracy which indicates that overscaling never occurs in practice. Moreover, both Theorem 3.3 and the rounding error analysis for repeated squaring are pessimistic [18], the true error can be even smaller.

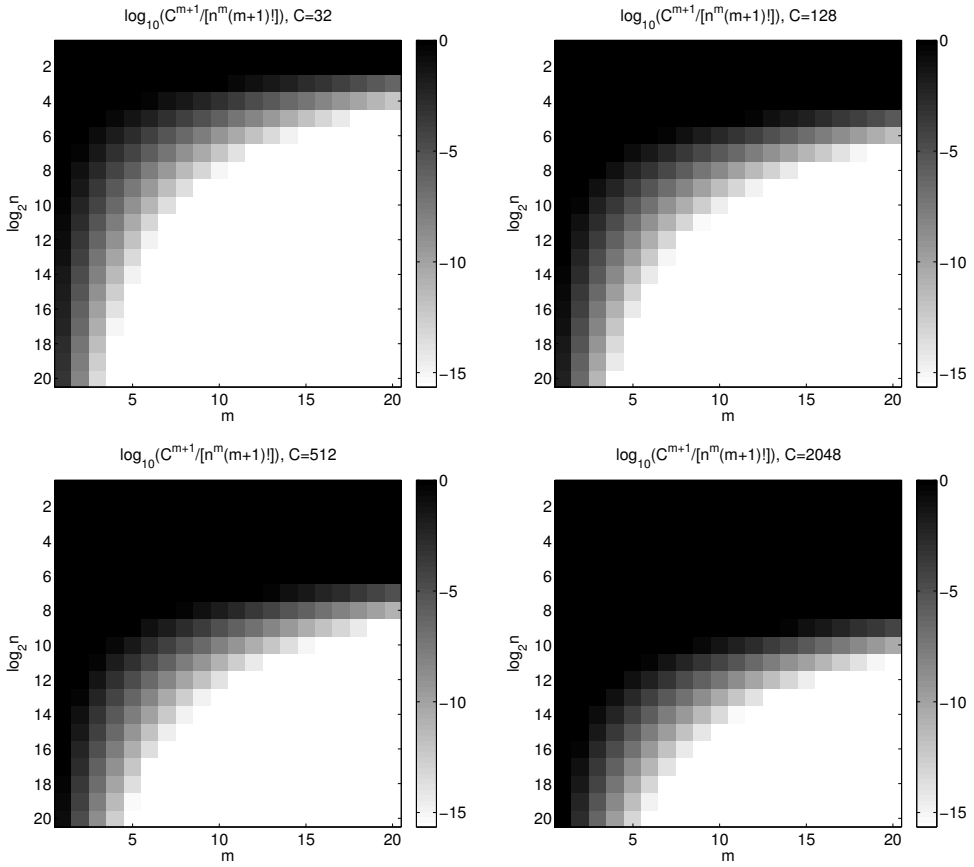

Figure 1: Sample values of $C(A)^{m+1}/[n^m(m+1)!]$.
The color is a one to one correspondence with $\log_{10}\big[C(A)^{m+1}/[n^m(m+1)!]\big]$. Bright values correspond to small truncation errors, whereas dark values correspond to large errors.

Compared to traditional Taylor series methods, an advantage of knowing m and n a priori is that there are cheaper alternatives for evaluating $T_m(X)$ (see, e.g., [24]). For example,

$$\begin{aligned} T_6(X) &= \left(I + X + \frac{X^2}{2!}\right) + X^3\left(\frac{I}{3!} + \frac{X}{4!} + \frac{X^2}{5!} + \frac{X^3}{6!}\right), \\ T_7(X) &= \left(I + \frac{X^2}{2!} + \frac{X^4}{4!} + \frac{X^6}{6!}\right) + X\left(I + \frac{X^2}{3!} + \frac{X^4}{5!} + \frac{X^6}{7!}\right). \end{aligned} \tag{3.6}$$

They both require two less matrix multiplications than the naive approach. The upper bound of computational cost for $1 \le m \le 21$ is summarized in Table 1. We remark that the numbers in Table 1 are smaller than those values in [15, Table 2.2], since there is no need to treat odd powers and even powers separately when evaluating the truncated Taylor series. With the reformulation technique in (3.6), it is sensible to choose m and n minimizing $\pi(m) + \log_2 n$ so that the computational cost of evaluating $L_{m,n}(A)$ is minimized. Certainly, by taking into account the finite precision arithmetic, a smaller n is preferred once there are multiple choices of the optimal parameter setting. However, if the same idea is applied to traditional Taylor series methods, the a priori estimate of convergence rate becomes crucial; the algorithm will become quite complicated too.

Now we are ready to derive our aggressively truncated Taylor series method (shown in Algorithm 1) based on the above discussions. Some remarks on the algorithm are also provided below.

Table 1: Number of matrix multiplications, π_m, required for evaluating $T_m(x)$.

m	2	3	4	5	6	7	8	9	10	11
π_m	1	2	2	3	3	4	4	4	5	5
m	12	13	14	15	16	17	18	19	20	21
π_m	5	6	6	6	6	7	7	7	7	8

Algorithm 1 Aggressively truncated Taylor series method for e^A, with A essentially nonnegative

Require: $\widehat{A} \in \mathbb{R}_+^{N\times N}$, $\tau > \mathbf{u}$

1: (optional) Balancing.
2: Estimate $\rho(\widehat{A})$ (e.g., via power method).
3: Solve the discrete optimization problem

$$\pi(m) + \log_2 n = \min, \quad \text{s.t. } \frac{C(A)^{m+1}}{n^m(m+1)!} \le \tau$$

by enumerating over m and $\log_2 n$.
4: $E \leftarrow L_{m,n}(A)$.
5: (optional) Reversed balancing.

***Remark* 3.** An important feature of Algorithm 1 is that $\pi(m) + \log_2 n$ is of order $\mathcal{O}(\log N)$ when $\mathrm{fl}\big(\mathrm{e}^A\big)$ does not overflow and $|s(A)| = \mathcal{O}(N)$. To see this, we first conclude from (2.1) and (2.2) that $C(A) = \mathcal{O}(N)$. Then by Theorem 3.3, it is sufficient to choose $m_0 = 1$ and $n_0 = \lceil C(A)^2/(2\tau)\rceil$ in order to fulfill (3.5). Hence

$$\min\Big[\pi(m) + \log_2 n\Big] \le \pi(m_0) + \log_2 n_0 = \mathcal{O}(\log C(A)^2) = \mathcal{O}(\log N).$$

As a consequence, the complexity of Algorithm 1 is $\mathcal{O}(N^3 \log N)$. However, to achieve the componentwise high relative accuracy, the traditional Taylor series method for essentially nonnegative matrices requires $\mathcal{O}(N^4)$ operations as $m = \mathcal{O}(N)$ and $\log_2 n = \mathcal{O}(\log\log N)$. This is the main advantage of our new method. When the required accuracy is modestly low (e.g., only four correct digits are of interest), the new method becomes even more attractive while the traditional Taylor series method still requires $\mathcal{O}(N^4)$ operations to produce all nonzero entries in e^A.

***Remark* 4.** The rounding error of Algorithm 1 can be bounded by

$$\big|\mathrm{fl}[L_{m,n}(A)] - \mathrm{e}^A\big| \le \big[\tau + n(m+2)(N+3)\mathbf{u} + \mathcal{O}(\mathbf{u}^2)\big]\mathrm{e}^A, \tag{3.7}$$

using existing results (see, e.g., [10]) for the traditional Taylor series method directly. By substituting the order of m and n into the above bound, both methods have

$$\big|\mathrm{fl}[L_{m,n}(A)] - \mathrm{e}^A\big| \le \big[\tau + \mathcal{O}(N^{2+\varepsilon})\mathbf{u} + \mathcal{O}(\mathbf{u}^2)\big]\mathrm{e}^A.$$

Therefore, Algorithm 1 improves the efficiency of the traditional Taylor series method without sacrificing the accuracy.

***Remark* 5.** Some extra care must be taken in the shifting stage. If we compute $\mathrm{e}^{\widehat{A}}$ and $\mathrm{e}^{s(A)}$ straightforwardly, it is possible that $\mathrm{e}^{\widehat{A}}$ causes overflow but $\mathrm{e}^{s(A)}$ is small enough so that the desired result e^A is still representable with floating point numbers. Therefore shifting needs to be handled carefully when $s(A) < 0$. Since

$$\mathrm{e}^A = \big(\mathrm{e}^{A/n}\big)^n = \Big(\mathrm{e}^{s(A)/n}\mathrm{e}^{\widehat{A}/n}\Big)^n,$$

the shift $s(A)/n$ on A/n is much safer when n is large. The same observation can also be applied to $L_{m,n}(A)$, i.e.,

$$L_{m,n}(A) = \left[\mathrm{e}^{s(A)/n} T_m\Big(\frac{\widehat{A}}{n}\Big) \right]^n, \tag{3.8}$$

By applying the scaling before shifting, we are most likely able to avoid unnecessary overflow even when $s(A)$ is small. This trick is commonly used in computations of Markov models (e.g., see [17, 26, 36]). Moreover, we suggest that the shifting strategy (3.8) is always used even for $A \geq 0$ since some entries of $\mathrm{e}^{\widehat{A}}$ might underflow. In case that $|s(A)|$ is extremely large so that n has to be large to avoid overflow/underflow, we conclude from Section 2 that the condition number of the problem is also large. Such problems need special attention, and are not investigated in this paper.

***Remark* 6.** Balancing is a commonly used technique for computing the matrix exponential. It helps slightly to improve the accuracy when summing up poorly scaled numbers. Notice that balancing cannot reduce $\rho(\widehat{A})$ so it does not help reducing the condition number and the computational cost. Therefore, we provide balancing as an optional step. For essentially nonnegative matrices, since $\|A\|_\infty$ can be reduced by balancing, the technique can help avoiding potential overflow/underflow. Although balancing is not always safe for some eigenvalue problems [33], extensive numerical experiments in [27] show that it is unlikely to be harmful when computing matrix functions.

4 Further Refinements of the Algorithm

In this section, we discuss several variants of the aggressively truncated Taylor series method. These variants can fulfill some special kinds of requirements.

4.1 The upper bound algorithm

We have already seen from Theorem 3.3 that $L_{m,n}(A)$ is a lower bound of e^A. Sometimes an upper bound of e^A might be of interest. Hence we define

$$\begin{aligned} \widetilde{T}_m(x) &= T_m(x) + \frac{x^{m+1}}{m!\,m}\left(1 - \frac{x}{m}\right)^{-1} \\ &= 1 + x + \cdots + \frac{x^m}{m!} + \frac{x^{m+1}}{m!\,m} + \frac{x^{m+2}}{m!\,m^2} + \cdots \end{aligned}$$

and

$$U_{m,n}(A) = \mathrm{e}^{s(A)} \widetilde{T}_m\Big(\frac{\widehat{A}}{n}\Big)^n. \tag{4.1}$$

Notice that $\widetilde{T}_m(x)$ is an mth order rational approximation of e^x which can be expressed in the form

$$\widetilde{T}_m(x) = \frac{\left(1 - \dfrac{x}{m}\right) T_{m-2}(x) + \dfrac{x^{m-1}}{(m-1)!}}{1 - \dfrac{x}{m}}.$$

By the uniqueness of the Padé approximant, $\widetilde{T}_m(x)$ is in fact the $(m-1,1)$ Padé approximant of e^x. The (absolute value of) corresponding remainder is denoted as

$$\widetilde{R}_m(x) = \widetilde{T}_m(x) - \mathrm{e}^x.$$

Then for an essentially nonnegative matrix A with $\rho(\widehat{A}) < mn$, we have $\widetilde{R}_m(\widehat{A}/n) \geq 0$ and then $U_{m,n}(A) \geq \mathrm{e}^A$. Similar to Theorem 3.3, we are able to obtain a componentwise a priori error bound for $U_{m,n}(A)$.

Theorem 4.1. *Let $A \in \mathbb{R}^{N\times N}$ be an essentially nonnegative matrix. Then for any positive integers m and n satisfying $mn > C(A)$, we have*

$$0 \le U_{m,n}(A) - \mathrm{e}^A \le \left[\left(1 + \widetilde{R}_m\Big(\frac{C(A)}{n}\Big)\right)^n - 1\right]\mathrm{e}^A. \tag{4.2}$$

Proof. Notice that

$$\mathrm{e}^X \le \widetilde{T}_m(X) = \mathrm{e}^X + \widetilde{R}_m(X) \le \mathrm{e}^X\left[I + \widetilde{R}_m(X)\right]$$

holds for any nonnegative matrix X with $\rho(X/m) < 1$. Then by Lemma 3.2, we have

$$\widetilde{T}_m\Big(\frac{\widehat{A}}{n}\Big)^n - \mathrm{e}^{\widehat{A}} \le \left[\left(I + \widetilde{R}_m\Big(\frac{\widehat{A}}{n}\Big)\right)^n - I\right]\mathrm{e}^{\widehat{A}} \le \left[\left(1 + \widetilde{R}_m\Big(\frac{C(\widehat{A})}{n}\Big)\right)^n - 1\right]\mathrm{e}^{\widehat{A}},$$

based on the fact that $\big[1+\widetilde{R}_m(x/n)\big]^n - 1$ has nonnegative coefficients in its Maclaurin series expansion. Finally, it is sufficient to verify that

$$U_{m,n}(A) - \mathrm{e}^A = \mathrm{e}^{s(A)}\left[U_{m,n}(\widehat{A}) - \mathrm{e}^{\widehat{A}}\right].$$

Hence we obtain the conclusion. □

Notice that

$$\widetilde{R}_m(x) = \frac{x^{m+1}}{(m+1)!\,m} + \mathcal{O}(x^{m+2}).$$

Then once $\widetilde{R}_m\big(C(A)/n\big)$ is small, we have

$$\left[\left(1 + \widetilde{R}_m\Big(\frac{C(A)}{n}\Big)\right)^n - 1\right]\mathrm{e}^A \approx \frac{C(A)^{m+1}}{n^m(m+1)!\,m}\mathrm{e}^A,$$

which is similar to (3.4) in Theorem 3.3.

The algorithm for computing the upper bound is also straightforward. Algorithm 2 is an upper bound version in analogy to Algorithm 1. A notable difference between $U_{m,n}(A)$ and $L_{m,n}(A)$ is that the matrix inverse is needed for evaluating the upper bound. Since $I - \widehat{A}/(mn)$ is an M-matrix, its inverse can be calculated accurately using GTH-type algorithms [1]. Actually by increasing the scale factor to $2n$, $[I - \widehat{A}/(2mn)]^{-1}$ can be safely computed even by the standard Gaussian elimination without pivoting because it is always well-conditioned in the sense of componentwise perturbation (see [34]). The subtractions on the diagonal never cause severe cancellation and hence can be interpreted as small relative backward errors. Both GTH-type algorithms and the Gaussian elimination can adopt block variants so that most computational work is done in Level 3 BLAS. This feature is useful when the performance is important. Finally, we provide the rounding error bounds for $\mathrm{fl}[U_{m,n}(A)]$ in Theorem 4.2. A proof can be found in [25].

Theorem 4.2. *Let A be an $N \times N$ essentially nonnegative matrix and let m, n be positive integers satisfying $mn > 2\rho(\widehat{A})$ and $U_{m,n}(A) - \mathrm{e}^A \le \tau\mathrm{e}^A$. Then*

$$\big|\mathrm{fl}[U_{m,n}(A)] - \mathrm{e}^A\big| \le \big[\tau + n(m+10N)(N+3)\mathbf{u} + \mathcal{O}(\mathbf{u}^2)\big]\mathrm{e}^A. \tag{4.3}$$

4.2 A posteriori error estimates and iterative methods

The a priori error estimates provided in Theorems 3.3 and 4.1 lead to simple and efficient approaches for computing e^A. However, rounding error is not taken into account when determining the parameters. Also, sometimes $\widehat{A}^k\mathrm{e}^A \le C(A)^k\mathrm{e}^A$ can overestimate the truncation error quite a lot. Therefore the choice of m and n might be a bit too large. As an extreme case, if the user has an unfeasibly high

Algorithm 2 A priori upper bound algorithm for e^A, with A essentially nonnegative

Require: $\widehat{A} \in \mathbb{R}_+^{N\times N}$, $\tau > \mathbf{u}$

1: (optional) Balancing.
2: Estimate $\rho(\widehat{A})$ (e.g., via power method).
3: Solve the discrete optimization problem

$$\pi(m-1) + \log_2 n = \min, \quad \text{s.t. } mn > 2\rho(\widehat{A}) \text{ and } \left(1 + \widetilde{R}_m\Big(\frac{C(A)}{n}\Big)\right)^n - 1 \le \tau$$

by enumerating over m and $\log_2 n$.
4: $E \leftarrow U_{m,n}(A)$.
5: (optional) Reversed balancing.

requirement on the accuracy (e.g., $\tau = 10^{-12}$ using IEEE single precision arithmetic), the parameters m and n which bound the truncation error well below τ can cause significant rounding error. A sensible choice might be returning a nearly optimal solution in the working precision and reporting a warning message to the user. In any case, we need to avoid m or n being too large when the accuracy requirement is unfeasibly high. From Section 3, we have already seen that both m and $\log_2 n$ are of order 10^1 in the aggressively truncated Taylor series method. Since the rounding error depends more sensitively on $\log_2 n$ compared to m, we can use a fixed truncation order m (e.g., $m = 13$) and seek for an appropriate scale factor n to avoid overscaling. In the following, we derive such kind of iterative approaches based on more accurate error estimates.

In the traditional Taylor series method, the truncation order m is determined by an a posteriori estimate [36]

$$\frac{(\widehat{A}/n)^{m+1}}{(m+1)!}\left[I - \frac{\widehat{A}}{n(m+2)}\right]^{-1} \le \tau \cdot T_m\Big(\frac{\widehat{A}}{n}\Big). \tag{4.4}$$

This criterion is generally too hard to be satisfied when applying to the aggressively truncated approach since $T_m(\widehat{A}/n)$ might not have any relative accuracy to $\mathrm{e}^{\widehat{A}/n}$. Notice that Lemma 3.1 takes into account both m and n. The error estimate (3.2) is a potential candidate for our purpose since it is more accurate than (3.4). But (3.2) cannot be used directly as a stopping criterion in practice since e^A is not known. To overcome this difficulty, we further assume that

$$\frac{C(A)^{m+1}}{n^m(m+1)!} \le \frac{1}{2},$$

which is reasonably easy to fulfill (since by definition, $C(A) \le N - 1 + \ln N + \ln R_{\max} - s(A)$). By Theorem 3.3, it immediately yields

$$\mathrm{e}^{\widehat{A}} \le 2T_m\Big(\frac{\widehat{A}}{n}\Big)^n,$$

i.e., $\mathrm{e}^{\widehat{A}}$ can be bounded from above by $2T_m(\widehat{A}/n)^n$. Then the a posteriori error estimate

$$\frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{\tau}{2}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \tag{4.5}$$

can be used as a stopping criterion which ensures $\mathrm{e}^A - L_{m,n}(A) \le \tau \mathrm{e}^A$. In fact,

$$\frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{\tau}{1+2\tau}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \tag{4.6}$$

is a better one which also guarantees the same error bound. To see this, from (4.6) and Lemma 3.1 we conclude that

$$\mathrm{e}^{\widehat{A}} - T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{\widehat{A}^{m+1}}{n^m(m+1)!}\mathrm{e}^{\widehat{A}} \le \frac{2\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{2\tau}{1+2\tau}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{2\tau}{1+2\tau}\mathrm{e}^{\widehat{A}}.$$

Repeating the same procedure with the new bound, we eventually obtain

$$\mathrm{e}^{\widehat{A}} - T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{\widehat{A}^{m+1}}{n^m(m+1)!}\mathrm{e}^{\widehat{A}} \le \frac{1}{1-\frac{2\tau}{1+2\tau}}\frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \frac{\frac{\tau}{1+2\tau}}{1-\frac{2\tau}{1+2\tau}}\mathrm{e}^{\widehat{A}} = \tau\mathrm{e}^{\widehat{A}}.$$

The condition (4.6) is relaxed roughly by half compared to (4.5) by the careful analysis. Sometimes this might have some potential benefit to reduce $\log_2 n$ by one.

The condition (4.6) has several advantages over (4.4). For example, (4.6) is inverse free and hence cheaper than (4.4). A more important feature is that even if (4.6) is not satisfied for a given choice of (m, n), say

$$\varepsilon_{m,n} = \min\left\{\varepsilon\colon \frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^n \le \varepsilon T_m\Big(\frac{\widehat{A}}{n}\Big)^n\right\} > \frac{\tau}{1+2\tau},$$

it still suggests the appropriate choice of the parameters. For example, we can expect $\varepsilon_{m,2n} \sim 2^{-m}\varepsilon_{m,n}$ when $\varepsilon_{m,n} < 1$ because the convergence rate provided in Lemma 3.1 is reasonably accurate. This kind of heuristics often leads to a suitable choice of (m, n) rapidly. The typical situation in practice is that appropriate values of the parameter setting can be found immediately after an initial guess. As a consequence of the above discussion, an iterative method based on the a posteriori error estimate is presented in Algorithm 3. We recommend that $m = 13$ is used so that $n < 4C(A)$ is sufficient for $N \le 2048$. Usually, no more than two iterations are enough in Algorithm 3. Therefore the computational cost is also $\mathcal{O}(N^3 \log N)$ which is not much more expensive than Algorithm 1.

***Remark* 7.** If $\mathrm{e}^A v$ (for some vector $v \ge 0$) instead of e^A is of interest, the stopping criterion can be changed to

$$\frac{\widehat{A}^{m+1}}{n^m(m+1)!}T_m\Big(\frac{\widehat{A}}{n}\Big)^n v \le \frac{\tau}{1+2\tau}T_m\Big(\frac{\widehat{A}}{n}\Big)^n v,$$

which might be easier to satisfy than (4.6).

***Remark* 8.** Sometimes e^A contains very tiny nonzero entries which are less than or close to $R_{\min}$. For example, when A is symmetric and banded, the magnitude of off-diagonal entries in e^A decay rapidly along with the distance to the main diagonal since e^x is analytic [4]. Therefore it is advisable to have another absolute tolerance τ_0 to avoid unfeasible accuracy requirements on these tiny entries. The difference between true zeros and underflows can be distinguished by calculating $\mathrm{struct}\,[(I+B)^n]$ with $n = 2^{\lceil \log_2 N\rceil}$, where B is a binary matrix satisfying $\mathrm{struct}(B) = \mathrm{struct}(A)$.

Similarly, we can establish an iterative approach for $U_{m,n}(A)$. In analogy to Lemma 3.1, Lemma 4.3 provides an a posteriori error bound of $U_{m,n}(A)$.

Lemma 4.3. *If $A \in \mathbb{R}^{N\times N}$ is essentially nonnegative, then for positive integers m and n satisfying $mn > \rho(\widehat{A})$ and $m > 2$, we have*

$$\mathrm{e}^{s(A)}\frac{\widehat{A}^{m+1}}{n^m(m+1)!\,m}\mathrm{e}^{\frac{n-1}{n}\widehat{A}} \le U_{m,n}(A) - \mathrm{e}^A \le \frac{\widehat{A}^{m+1}}{n^m(m+1)!\,m}\Big(I - \frac{\widehat{A}}{mn}\Big)^{-1}U_{m,n}(A). \tag{4.7}$$

Based on this lemma, it is very straightforward to derive a stopping criterion. We can simply use

$$\frac{\widehat{A}^{m+1}}{n^m(m+1)!\,m}\Big(I - \frac{\widehat{A}}{mn}\Big)^{-1}U_{m,n}(A) \le \frac{\tau}{1+\tau}U_{m,n}(A), \tag{4.8}$$

which implies

$$U_{m,n}(A) - \mathrm{e}^A \le \tau\mathrm{e}^A.$$

Notice that $\big[I - \widehat{A}/(mn)\big]^{-1}$ has already been calculated when evaluating $U_{m,n}(A)$, the stopping criterion (4.8) only requires matrix multiplications. The iterative algorithm for $U_{m,n}(A)$ is not listed since it is very similar to Algorithm 3.

Algorithm 3 Lower bound iteration for e^A, with A essentially nonnegative

Require: $\widehat{A} \in \mathbb{R}_+^{N\times N}$, $m \in \mathbb{N}$, $\tau > \mathbf{u}$, $\tau_0 > R_{\min}$, $\texttt{MAXITER} \in \mathbb{N}$

(optional) Balancing.

$k \leftarrow \left\lceil \log_2\big(N + \max\{\widehat{A}(i,i)\}\big)\right\rceil + 1$

$\varepsilon \leftarrow \tau + 1$, $\varepsilon_0 \leftarrow \varepsilon$

while $\varepsilon_0 \geq \varepsilon \geq \frac{\tau}{1+2\tau}$ **and** $k \leq \texttt{MAXITER}$ **do**

  $n \leftarrow 2^k$, $\varepsilon_0 \leftarrow \varepsilon$

  $L \leftarrow L_{m,n}(A)$

  $W \leftarrow \dfrac{n(\widehat{A}/n)^{m+1}}{(m+1)!}L$

  $\varepsilon \leftarrow \max\left\{\dfrac{W(i,j)}{L(i,j)} : W(i,j) \geq \tau\tau_0\right\}$

  $k \leftarrow k + \dfrac{1}{m}\log_2\dfrac{\varepsilon}{\tau}$

end while

$E \leftarrow L$

if $\varepsilon \geq \tau$ **then**

  Report failure.

end if

(optional) Reversed balancing.

4.3 Interval algorithms with improved accuracy

When both $L_{m,n}(A)$ and $U_{m,n}(A)$ are computed, it is possible to make use of these bounds to gain a more accurate approximation of e^A. In the following, we demonstrate such a technique which potentially improves the accuracy of the solution using the upper and lower bounds. In addition, we present a novel high accuracy interval-type algorithm based on these bounds.

Sometimes both an upper bound and a lower bound of e^A or $\mathrm{e}^A v$ (for some vector $v \geq 0$) are of interest. Certainly we are able to calculate them separately using the algorithms we have presented. But we expect more useful outputs by merging them. Notice that

$$U_{m,n}(A) - L_{m,n}(A) = \big[U_{m,n}(A) - \mathrm{e}^A\big] + \big[\mathrm{e}^A - L_{m,n}(A)\big],$$

The difference between the upper bound and the lower bound is also componentwise small compared to e^A. Thus, once both $L_{m,n}(A)$ and $U_{m,n}(A)$ have already been calculated, a simpler a posteriori error estimate is automatically available. In principle, any matrix X satisfying $L_{m,n}(A) \leq X \leq U_{m,n}(A)$ is a good approximation of e^A when both bounds are sufficiently accurate. We seek for a higher order approximation of the form $\eta L_{m,n}(A) + (1-\eta)U_{m,n}(A)$ where $\eta \in [0,1]$, i.e., a convex combination of $L_{m,n}(A)$ and $U_{m,n}(A)$. It is straightforward to show that

$$\lim_{x\to 0}\frac{\mathrm{e}^x - T_m(x/n)^n}{x^{m+1}} = m\lim_{x\to 0}\frac{\widetilde{T}_m(x/n)^n - \mathrm{e}^x}{x^{m+1}} = \frac{1}{n^m(m+1)!}.$$

Therefore

$$E_{m,n}(A) = \frac{1}{m+1}L_{m,n}(A) + \frac{m}{m+1}U_{m,n}(A)$$

is an $(m+1)$-th order approximation of e^A. Using the same techniques for deriving Theorems 3.3 and 4.1, the following error estimate can be established.

Theorem 4.4. *Let $A \in \mathbb{R}^{N\times N}$ be essentially nonnegative and let m, n be positive integers satisfying $mn > C(A)$. Then*

$$0 \leq E_{m,n}(A) - \mathrm{e}^A \leq \frac{m}{m+1}\left[\left(1 + \widetilde{R}_m\Big(\frac{C(A)}{n}\Big)\right)^n - 1 + \frac{C(A)^{m+2} - C(A)^{m+1}}{n^m(m+1)!\,m}\right]\mathrm{e}^A. \tag{4.9}$$

Proof. Without loss of generality, we can assume $A \geq 0$. On the one hand, from Lemmas 3.1 and 4.3 we have

$$\begin{aligned} E_{m,n}(A) - \mathrm{e}^A &= \frac{m}{m+1}\big[U_{m,n}(A) - \mathrm{e}^A\big] + \frac{1}{m+1}\big[L_{m,n}(A) - \mathrm{e}^A\big] \\ &\geq \frac{1}{m+1}\left[m\widetilde{R}_m\Big(\frac{A}{n}\Big) - R_m\Big(\frac{A}{n}\Big)\right]\mathrm{e}^{\frac{n-1}{n}A}. \end{aligned}$$

Notice that

$$m\widetilde{R}_m(x) - R_m(x) = \sum_{k=1}^{\infty}\left[\frac{1}{m!\,m^{k-1}} - \frac{m+1}{(m+k)!}\right]x^{m+k}$$

has nonnegative coefficients in the Maclaurin series expansion. Hence we obtain

$$m\widetilde{R}_m\Big(\frac{A}{n}\Big) - R_m\Big(\frac{A}{n}\Big) \geq 0,$$

and then

$$E_{m,n}(A) - \mathrm{e}^A \geq 0.$$

On the other hand,

$$L_{m,n}(A) - \mathrm{e}^A \leq -\frac{A^{m+1}}{n^m(m+1)!} = \frac{A^{m+1}}{n^m(m+1)!}\big(\mathrm{e}^A - I\big) - \frac{A^{m+1}}{n^m(m+1)!}\mathrm{e}^A \leq \frac{A^{m+2} - A^{m+1}}{n^m(m+1)!}\mathrm{e}^A.$$

Therefore,

$$\begin{aligned} E_{m,n}(A) - \mathrm{e}^A &= \frac{m}{m+1}\left[\big(U_{m,n}(A) - \mathrm{e}^A\big) + \frac{1}{m}\big(L_{m,n}(A) - \mathrm{e}^A\big)\right] \\ &\leq \frac{m}{m+1}\left[\left(1 + \widetilde{R}_m\Big(\frac{A}{n}\Big)\right)^n - 1 + \frac{A^{m+2} - A^{m+1}}{n^m(m+1)!\,m}\right]\mathrm{e}^A \\ &\leq \frac{m}{m+1}\left[\left(1 + \widetilde{R}_m\Big(\frac{C(A)}{n}\Big)\right)^n - 1 + \frac{C(A)^{m+2} - C(A)^{m+1}}{n^m(m+1)!\,m}\right]\mathrm{e}^A, \end{aligned}$$

based on the fact that

$$\left[1 + \widetilde{R}_m\Big(\frac{x}{n}\Big)\right]^n - 1 + \frac{x^{m+2} - x^{m+1}}{n^m(m+1)!\,m}$$

has nonnegative coefficients in its Maclaurin series expansion. □

***Remark* 9.** From the definition of $\widetilde{T}_m(x)$, we are able to see that the terms up to order m are shared when evaluating $T_m(\widehat{A}/n)$ and $\widetilde{T}_m(\widehat{A}/n)$ simultaneously. This observation is also valid for some alternative approaches such as (3.6).

In applications [3] where accurate bounds are extremely important, it is desirable that

$$\underline{\mathrm{fl}}[L_{m,n}(A)] \leq L_{m,n}(A) \leq \mathrm{e}^A \leq U_{m,n}(A) \leq \overline{\mathrm{fl}}[U_{m,n}(A)] \tag{4.10}$$

is guaranteed regardless of rounding errors. In this case $L_{m,n}(A)$ and $U_{m,n}(A)$ need to be calculated separately using different rounding modes and no computational cost can be saved. The technique of convex combination can still be applied although $\mathrm{fl}[E_{m,n}(A)]$ is not guaranteed to be another upper bound of e^A.

Now it is important to verify that the property (4.10) can indeed be achieved. Obviously, $\underline{\mathrm{fl}}[L_{m,n}(A)] \leq L_{m,n}(A)$ can be easily satisfied by simply switching the rounding mode to *round towards* $-\infty$. For the upper bound, the tricky part is to achieve $\overline{\mathrm{fl}}\big(M^{-1}\big) \geq M^{-1}$, where $M = I - A/(mn)$ is

an M-matrix[1]. Firstly, $\mathrm{fl}(M)$ is obtained by $\mathrm{fl}(M) = -\overline{\mathrm{fl}}\big[\overline{\mathrm{fl}}[A/(mn) - I]\big] \le M$. Consider one step of Gaussian elimination for calculating the LU factorization

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} 1 & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ & S_{22} \end{bmatrix},$$

where S_{22} is the Schur complement. The factorization can be computed through

$$\begin{aligned} \mathrm{fl}(L_{21}) &= -\overline{\mathrm{fl}}(|L_{21}|) = -\overline{\mathrm{fl}} \left[\frac{\overline{\mathrm{fl}}(|M_{21}|)}{\underline{\mathrm{fl}}(M_{11})} \right] \le L_{21}, \\ \mathrm{fl}(S_{22}) &= -\overline{\mathrm{fl}}\big[\overline{\mathrm{fl}}(|L_{21}|)\,\overline{\mathrm{fl}}(|M_{12}|) - \underline{\mathrm{fl}}(M_{22})\big] \le S_{22}. \end{aligned}$$

By applying the same procedure recursively to S_{22}, we are able to conclude that the LU factorization $M = LU$ satisfies

$$\underline{\mathrm{fl}}(L) = \mathrm{fl}(L) \le L, \qquad \underline{\mathrm{fl}}(U) = \mathrm{fl}(U) \le U.$$

Suppose M^{-1} is computed by solving two triangular linear systems under the rounding mode to *round towards* $+\infty$, the solution satisfies

$$\overline{\mathrm{fl}}\big(M^{-1}\big) \ge \underline{\mathrm{fl}}(U)^{-1}\,\underline{\mathrm{fl}}(L)^{-1} \ge U^{-1}L^{-1} = M^{-1}.$$

The conclusion can be generalized to block LU factorization and some algorithmic variants for computing M^{-1} as long as $\mathrm{fl}(L)$ and $\mathrm{fl}(U)$ are true lower bounds of L and U, respectively. However, the analysis does not carry over to GTH-type algorithms since the diagonal entries of S_{22} are computed in a different way. Therefore if the actual upper bound is important, we prefer standard Gaussian elimination rather than GTH-type algorithms. In practice all calculations can be done without switching the rounding mode (under the rounding mode to *round towards* $+\infty$). Notice that the rounding errors can be modeled as

$$\underline{\mathrm{fl}}(a \circ b) = (a \circ b)(1 - \epsilon_1), \quad \overline{\mathrm{fl}}(a \circ b) = (a \circ b)(1 + \epsilon_2), \quad 0 \le \epsilon_1, \epsilon_2 \le \mathbf{u},$$

which both fit the rounding model

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \epsilon), \quad |\epsilon| \le \mathbf{u}.$$

Here the unit roundoff $\mathbf{u}$ is twice as large as the one under the standard rounding mode (i.e., *round towards nearest*) [21]. Therefore, the error bounds (3.7) and (4.3) are still valid under directed rounding modes.

A complete algorithm is summarized as Algorithm 4 which provides both lower and upper bounds and a higher order approximation in between. Because of the property (4.10), this approach can be seen as an interval algorithm [19] but without any explicit use of interval arithmetic. The rounding error analysis demonstrates that this interval algorithm does not severely overestimate the error. Generally the a posteriori error $U_{m,n}(A) - L_{m,n}(A)$ is also much smaller than the a priori error based on truncation and rounding error analysis. In practice, Algorithm 4 is more robust than other algorithms presented in this paper, while it is also more costly.

5 Numerical Experiments

In this section, we present and discuss some experimental results. All experiments are performed on a Linux machine with an Intel quadcore 3.10GHz CPU and 8GB memory. We make use of only a single core for the experiments. Both Algorithm 1 and Algorithm 4 are implemented in C99 and compiled by the GNU C complier[2]. Matrix multiplications are performed by the optimized GEMM routine

[1] Unfortunately, GTH-type algorithms [1] usually do not have this property.

[2] GCC, the GNU Complier Collection: `http://gcc.gnu.org/`.

Algorithm 4 Interval algorithm for e^A, with A essentially nonnegative

Require: $\widehat{A} \in \mathbb{R}_+^{N\times N}$, $m \in \mathbb{N}$, $\tau > \mathbf{u}$, $\tau_0 > R_{\min}$, $\texttt{MAXITER} \in \mathbb{N}$

(optional) Balancing.

$k \leftarrow \left\lceil \log_2\big(N + \max\{\widehat{A}(i,i)\}\big) \right\rceil + 1$

$\varepsilon \leftarrow \tau + 1$, $\varepsilon_0 \leftarrow \varepsilon$

$L \leftarrow A$, $U \leftarrow +\infty$

while $\varepsilon_0 \geq \varepsilon \geq \tau$ **and** $k \leq \texttt{MAXITER}$ **do**

  $n \leftarrow 2^k$, $\varepsilon_0 \leftarrow \varepsilon$

  $L \leftarrow \max\big\{L, \underline{\mathrm{fl}}[L_{m,n}(A)]\big\}$ (*round towards* $-\infty$)

  (Try) $U \leftarrow \min\big\{U, \overline{\mathrm{fl}}[U_{m,n}(A)]\big\}$ (*round towards* $+\infty$)

  $\varepsilon \leftarrow \max\left\{ \dfrac{U(i,j) - L(i,j)}{L(i,j)} : U(i,j) \geq \tau_0 \right\}$

  $k \leftarrow k + \dfrac{1}{m} \log_2 \dfrac{\varepsilon}{\tau}$

end while

$E \leftarrow \dfrac{1}{m+1} L + \dfrac{m}{m+1} U$

if $\varepsilon \geq \tau$ **then**

  Report failure.

end if

(optional) Reversed balancing.

from the OpenBLAS library[3]; while other operations, such as block LU factorization and solving triangular systems, are carefully coded so that they are not ruined by the compiler. For the parameter setting, we choose $m = 8$ as the truncation order in Algorithm 4. It is relatively small compared to the recommended value ($m = 13$) so that we are able to observe some behaviors of this algorithm which rarely occur under the recommended setting. The relative and absolute tolerances are set to $\tau = 1024N\mathbf{u} \approx 2.3 \times 10^{-13} N$ and $\tau_0 = R_{\min}/\mathbf{u} \approx 1.0 \times 10^{-292}$, respectively. The maximum iteration number is set to $\texttt{MAXITER} = 52$ which is more than needed. Balancing is not used in the experiments to show that our algorithm is not sensitive to bad scaling in the original matrix. The switching of rounding modes are done by the standard library function `fesetround()`. As a comparison, we also run the same set of tests on the same machine for MATLAB's `expm` function (version R2012a) which is an excellent general purpose solver. We choose nine test matrices which are commonly used. All entries in the exponentials can be represented by normalized floating point numbers (i.e., no overflow or (gradual) underflow).

Example 1.

$$A = \begin{bmatrix} -0.01 & 10^{15} \\ 0 & -0.01 + 10^{-6} \end{bmatrix}.$$

Example 2. *A matrix modified from an example in C. Moler's blog*[4]*:*

$$A = \begin{bmatrix} 0 & e & 0 \\ a+b & -d & a \\ c & 0 & -c \end{bmatrix},$$

where $a = 2 \times 10^{10}$, $b = \frac{2}{3} \times 10^8$, $c = \frac{200}{3}$, $d = 3$, $e = 10^{-8}$.

[3] OpenBLAS: `http://xianyi.github.io/OpenBLAS/`.

[4] C. Moler, A Balancing Act for the Matrix Exponential, `http://blogs.mathworks.com/cleve/2012/07/23/a-balancing-act-for-the-matrix-exponential/`.

Example 3. [8]

$$A = \begin{bmatrix} -16 & 2^{60} & 2^{60} & 2^{60} \\ 0 & -16 & 2^{60} & 2^{60} \\ 0 & 0 & -1 & 2^{60} \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Example 4. [32] *The* 10×10 *Forsythe matrix*

$$A = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 10^{-10} & & & 0 \end{bmatrix}_{10\times 10}.$$

Example 5. [36] *The* 50×50 *one-dimensional Laplacian matrix*

$$A = -T_{50} = \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}_{50\times 50}.$$

Example 6. *The* 128×128 *Jordan block with zero eigenvalues*

$$A = J_{128}(0) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix}_{128\times 128}.$$

Example 7. [36] *The adjacency matrix of a* 2*-nearest ring network of* 200 *nodes with four extra shortcuts* 16–30*,* 74–85*,* 90–128*,* 138–147*, modelling a small-world network. It is generated by* `smallw(200,2,0.03)`*, where* `smallw.m` *is from the MATLAB toolbox CONTEST*[5] [30].

Example 8. [36] *The two-dimensional Laplacian matrix on a* 42×42 *grid with Dirichlet boundary condition*

$$A = -T_{40,40} = -(T_{40} \otimes I_{40} + I_{40} \otimes T_{40}).$$

Example 9.

$$A = 1400 \times J_{2048}(-1/2) = \begin{bmatrix} -700 & 1400 & & \\ & \ddots & \ddots & \\ & & \ddots & 1400 \\ & & & -700 \end{bmatrix}_{2048\times 2048}.$$

The componentwise errors for these testing examples are listed in Table 2. For Algorithm 4, we present the a posteriori error estimate computed from $U_{m,n}(A) - L_{m,n}(A)$, as well as the true error from the "accurate" solution computed by the MATLAB Symbolic Computation toolbox. Algorithm 1 successfully solves all these problems to desired accuracy. Algorithm 4 also solves all problems except for Example 2. Despite that the true error in Example 2 is already smaller than τ, we regard it as a failure since the algorithm cannot identify the true error by itself. However, Algorithm 4 still provides a useful error estimate which informs the user regarding the quality of the unsatisfactory solution. The difference between the a posteriori error estimate and the true error also confirms that

[5]CONTEST: `http://www.mathstat.strath.ac.uk/research/groups/numerical_analysis/contest` .

Table 2: Componentwise relative errors of the test examples.

Matrix ID	$C(A)$	$\kappa_{\exp}(A)$	τ	Algorithm 1	Algorithm 4		`expm`
					estimated	true	
1	1.0×10^{0}	1.7×10^{29}	4.5×10^{-13}	8.9×10^{-16}	1.2×10^{-14}	6.3×10^{-15}	1.0×10^{-2}
2	8.6×10^{1}	2.4×10^{19}	6.8×10^{-13}	4.7×10^{-14}	2.5×10^{-12}	6.0×10^{-13}	1.0×10^{0}
3	1.8×10^{1}	5.2×10^{69}	9.1×10^{-13}	5.7×10^{-15}	3.8×10^{-13}	1.5×10^{-13}	1.0×10^{0}
4	9.1×10^{0}	1.7×10^{0}	2.3×10^{-12}	3.6×10^{-16}	1.1×10^{-12}	6.8×10^{-14}	1.8×10^{-14}
5	5.1×10^{1}	5.4×10^{0}	1.1×10^{-11}	2.0×10^{-14}	2.9×10^{-12}	2.2×10^{-12}	2.6×10^{6}
6	1.3×10^{2}	1.8×10^{0}	2.9×10^{-11}	9.8×10^{-13}	1.5×10^{-11}	6.9×10^{-14}	9.7×10^{77}
7	2.0×10^{2}	7.3×10^{0}	4.5×10^{-11}	2.8×10^{-14}	2.2×10^{-11}	1.9×10^{-11}	3.3×10^{7}
8	1.6×10^{3}	6.1×10^{1}	3.6×10^{-10}	6.3×10^{-13}	1.2×10^{-10}	1.0×10^{-10}	4.2×10^{6}
9	2.0×10^{3}	2.4×10^{4}	4.6×10^{-10}	6.1×10^{-11}	3.5×10^{-10}	8.4×10^{-12}	1.1×10^{-12}

Table 3: Parameter settings for Algorithm 1 and Algorithm 4.

Matrix ID	Algorithm 1		Algorithm 4		
	m	$\log_2 n$	m	$\log_2 n$	#(`iter`)
1	12	1	8	3	1
2	18	6	8	10	3
3	21	3	8	8	2
4	21	2	8	5	2
5	18	5	8	9	2
6	19	6	8	10	2
7	18	7	8	9	1
8	18	10	8	12	1
9	19	10	8	14	2

the weighted sum $E_{m,n}(A)$ can be more accurate than the upper/lower bounds. As a general purpose solver, MATLAB's `expm` function only produces componentwise accuracy for two of these examples. In fact, for the first three Examples which have large normwise condition numbers[6], the normwise errors of `expm` are also large (1.0×10^{-2}, 1.0×10^{0}, and 1.0×10^{0}, respectively).

[6]The normwise relative condition number $\kappa_{\exp}(A)$ is defined as

$$\kappa_{\exp}(A) = \max_{Z \neq 0} \frac{\|A\|_F}{\|Z\|_F \, \|\mathrm{e}^A\|_F} \left\| \int_0^1 \mathrm{e}^{(1-s)A} Z \mathrm{e}^{sA} \mathrm{d}s \right\|_F ,$$

see, e.g., [14].

Table 4: Execution time (in seconds) of the test examples.

Matrix ID	Algorithm 1	Algorithm 4	`expm`
1	8.4×10^{-5}	1.5×10^{-5}	5.4×10^{-4}
2	8.9×10^{-5}	4.7×10^{-5}	3.7×10^{-4}
3	9.4×10^{-5}	2.9×10^{-5}	6.0×10^{-4}
4	8.3×10^{-5}	3.2×10^{-5}	2.5×10^{-3}
5	4.2×10^{-4}	1.2×10^{-3}	5.5×10^{-4}
6	3.7×10^{-3}	2.0×10^{-2}	3.6×10^{-3}
7	1.3×10^{-2}	2.4×10^{-2}	7.2×10^{-3}
8	8.6×10^{0}	3.5×10^{1}	3.1×10^{0}
9	1.4×10^{1}	1.1×10^{2}	2.4×10^{1}

Example 6 is of our particular interest. The exact solution is known as

$$\left(\mathrm{e}^A\right)(i,j) = \begin{cases} 0, & i > j, \\ \frac{1}{(j-i)!}, & i \le j, \end{cases}$$

whose nonzero entries are the coefficients in the Maclaurin series of e^x. Algorithm 4 terminates successfully with $n = 1024$. Therefore the first 128 terms in the Maclaurin series of $T_8\left(x/1024\right)^{1024}$ and $\widetilde{T}_8\left(x/1024\right)^{1024}$ both agree with those terms in e^x within a relative error of 2×10^{-11}. Since $1/128! < 2.6 \times 10^{-216}$, the remaining terms are very tiny. This is an intuitive way to understand why our methods can always produce high relative accuracy. Example 9 is a very challenging problem because the magnitude of nonzero entries in the solution vary from 10^{-304} to 10^{302}. Our new algorithms still provide componentwise accurate bounds for this extreme case. Interestingly, `expm` also does a good job for this difficult example but fails for Example 6 which should be easier. A brief explanation is that `expm` tries to avoid "unnecessary" scaling in Example 6 and thus loses the opportunity to recover small entries in the solution.

Table 3 lists the parameters used in Algorithm 1 and Algorithm 4, respectively. For Examples 3, 7, and 8, the parameter settings used in Algorithm 4 would be treated as infeasible in Algorithm 1, since they cannot pass the a priori test (3.5). Therefore, Theorem 3.3 indeed overestimates the truncation errors. Interestingly, because of the overestimation, the parameter setting used in Algorithm 4 requires less computational work compared to the "optimal" settings in Algorithm 1 for Examples 7 and 8. Although Algorithm 4 is still more expensive since the upper bound $U_{m,n}(A)$ is also calculated, Algorithm 3 with the same setting might be potentially cheaper than Algorithm 1. These phenomenon confirm our motivation for developing iterative approaches based on a posteriori error estimates. Algorithm 4 uses at most two iterations in almost all examples since the estimated scale factor n based on the convergence rate is quite accurate. Example 2 requires three iterations because it is terminated in the third step where the error stops decreasing.

The computational times are provided in Table 4 for reference also. Algorithm 1 and MATLAB's `expm` have similar performance since they both use a priori estimates to determine the approximation order and the scale factor. As an iterative algorithm, with both upper and lower bounds computed, Algorithm 4 is in general slower. This is the cost we need to pay in order to obtain extra robustness.

***Remark* 10.** In practice, $m = 13$ is recommended as the truncation order for double precision floating point numbers. Algorithm 4 can solve all these examples with this recommended setting. Most of them are solved in one iteration and hence the performance is also largely improved compared to $m = 8$. But the convergence rate can be better understood with a smaller truncation order since we need to predict the scale factor from an improper initial guess. A failure in the testing examples with $m = 8$ also demonstrates the robustness of Algorithm 4.

6 Conclusions

Taylor expansions are usually not the first choice for calculating the exponentials of general matrices. However, by carefully choosing the order of expansion and the scale factor, it is preferred when the matrices are nonnegative. We make use of the nonnegativity and establish accurate a priori estimates of the lower and upper bounds. We also show that the weighted average of the bounds derives a higher-order approximation. These novel theoretical results lead to different variants of the algorithms. Some similar techniques can also be applied to analyze the truncation error for Padé methods. But how to accurately compute the Padé approximant is still a challenging problem. Rounding error analysis ensures the stability of the proposed algorithms. The interval algorithm also provides componentwise error estimates regardless of roundoff. We tested these new algorithms with some commonly used matrices.

Acknowledgements

The authors would like to thank Bo Kågström and Daniel Kressner for fruitful discussions and constructive comments and proposals. They also thank the referees for providing valuable suggestions.

References

[1] A. S. Alfa, J. Xue, and Q. Ye. Accurate computation of the smallest eigenvalue of a diagonally dominant M-matrix. *Math. Comp.*, 71(237):217–236, 2002.

[2] M. Arioli, B. Codenotti, and C. Fassino. The Padé method for computing the matrix exponential. *Linear Algebra Appl.*, 240:111–130, 1996.

[3] R. Baker Kearfott and V. Kreinovich, editors. *Applications of Interval Computations*, volume 3 of *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[4] M. Benzi and G. H. Golub. Bounds for the entries of matrix functions with applications to preconditioning. *BIT*, 39(3):417–438, 1999.

[5] A. Berman and R. J. Plemmons. *Nonnegative matrices in the Mathematical Sciences*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994.

[6] P. Bochev and S. Markov. A self-validating numerical method for the matrix exponential. *Computing*, 43:59–72, 1989.

[7] B. Codenotti and C. Fassino. Error analysis of two algorithms for the computation of the matrix exponential. *Calcolo*, 29(1):1–31, 1992.

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

[9] E. Estrada and D. J. Higham. Network properties revealed through matrix functions. *SIAM Rev.*, 52(4):696–714, 2010.

[10] C. Fassino. *Computation of Matrix Functions*. PhD thesis, University of Pisa, 1993.

[11] A. Goldsztejn. On the exponentiation of interval matrices. Technical report, 2009. Available at `http://arxiv.org/abs/0908.3954`.

[12] L. Gurvits, R. Shorten, and O. Mason. On the stability of switched positive linear systems. *IEEE Trans. Automat. Control*, 52(6):1099–1103, 2007.

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

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

[15] N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Rev.*, 51(4):747–764, 2009.

[16] B. Kågström. Bounds and perturbation bounds for the matrix exponential. *BIT*, 17(1):39–57, 1977.

[17] B. J. Melloy and G. K. Bennett. Computing the exponential of an intensity matrix. *J. Comput. Appl. Math.*, 46(3):405–413, 1993.

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

[19] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[20] L. Moreau. Stability of continuous-time distributed consensus algorithms. In *43rd IEEE Conference on Decision and Control*, volume 4, pages 3998–4003, 2004.

[21] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, Boston, MA, USA, 2010.

[22] E. Nuding. Computing the exponential of an essentially nonnegative matrix. *Interval Mathematics*, pages 449–452, 1980.

[23] E. Nuding. Schrankentreue Berechnung der Exponentialfunktion wesentlich-nichtnegativer Matrizen. *Computing*, 26:57–66, 1981.

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

[25] M. Shao. *Dense and Structured Matrix Computations—the Parallel QR Algorithm and Matrix Exponentials*. PhD thesis, EPF Lausanne, 2013.

[26] R. B. Sidje. Inexact uniformization and GMRES methods for large Markov chains. *Numer. Linear Algebra Appl.*, 18:947–960, 2011.

[27] M. Stadelmann. Matrixfunktionen — Analyse und Implementierung. Master's thesis, ETH Zürich, 2009.

[28] C. Standish. Truncated Taylor series approximation to the state transition matrix of a continuous parameter finite Markov chain. *Linear Algebra Appl.*, 12(2):179–183, 1975.

[29] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, USA, 1994.

[30] A. Taylor and D. J. Higham. CONTEST: A controllable test matrix toolbox for MATLAB. *ACM Trans. Math. Softw.*, 35(4):26:1–26:17, 2009.

[31] C. F. Van Loan. The sensitivity of the matrix exponential. *SIAM J. Numer. Anal.*, 14(6):971–981, 1977.

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

[33] D. S. Watkins. A case where balancing is harmful. *Electron. Trans. Numer. Anal.*, 23:1–4, 2006.

[34] J. Xue and E. Jiang. Entrywise relative perturbation theory for nonsingular M-matrices and applications. *BIT*, 35(3):417–427, 1995.

[35] J. Xue and Q. Ye. Entrywise relative perturbation bounds for exponentials of essentially nonnegative matrices. *Numer. Math.*, 110(3):393–403, 2008.

[36] J. Xue and Q. Ye. Computing exponentials of essentially non-negative matrices entrywise to high relative accuracy. *Math. Comp.*, 82:1577–1596, 2013.