Bergische Universität Wuppertal

Fachbereich Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 19/43

Andreas Frommer and Behnam Hashemi

Computing enclosures for the matrix exponential

December 27, 2019

http://www.math.uni-wuppertal.de

COMPUTING ENCLOSURES FOR THE MATRIX EXPONENTIAL*

ANDREAS FROMMER† AND BEHNAM HASHEMI‡

Abstract. We present a review of old and develop new interval arithmetic techniques for computing enclosures for all entries of the *exact* exponential of a matrix. This means that all the rounding and truncation errors committed in the course of computation are rigorously taken into account and the result is mathematically guaranteed to contain the correct matrix exponential. We consider algorithms relying on verified spectral decomposition, two variants relying on Taylor series expansion, a Padé approximation and a contour integration approach together with a Chebyshev approximation based method which is designed for Hermitian matrices. Most of our methods use the scaling and squaring framework and are examined when applied to both the original matrix as well as to an approximate diagonalization. In addition to a comparative study of algorithms, several illustrative numerical examples are given.

Key words. matrix exponential, interval arithmetic, automatic result verification, INTLAB, scaling and squaring

AMS subject classifications. 65F60, 65F30, 65G20

1. Introduction.

The task of computing the exponential $\exp(A)$ of a matrix $A \in \mathbb{C}^{n\times n}$ arises in a variety of applications such as in exponential integrators for ODEs and semi-discretizations of PDEs, in network analysis or in continuous-time Markov models. The development of stable and efficient methods for computing $\exp(A)$ has thus been a topic of intensive research, see the survey paper [26] from 1978 and its update [27] from 2003. Presently, a Padé approximation type method [13] combined with a scaling and squaring approach recently improved in [1] may be considered state of the art. This approach is, in particular, implemented in MATLAB's `expm` function. Roughly speaking, the approach determines a scaling parameter s and a degree q with the ultimate goal that the *backward* error in computing $\exp(A)$ via squaring a (q, q) type Padé approximation of the scaled matrix is in the order of the unit round-off u[1]. Specifically, it is shown in [1] how to choose s and $q \in \{3, 5, 7, 9, 13\}$ to achieve this goal in double precision.

An important question is now how well the result of a computation indeed approximates $\exp(A)$. In this paper, we develop new approaches which, together with the approximation to $\exp(A)$, also compute mathematically guaranteed error bounds for each entry of the matrix. Such approaches will be termed *verified* computations. We compare them with existing ones with respect to the tightness of the bounds obtained and the computational complexity. Conceptually, all these verified computations rely on theoretical results on approximation errors as well as on the use of (machine) interval arithmetic to control the rounding errors related to the use of floating point arithmetic.

The paper is organized as follows: Section 2 briefly reviews those properties of interval arithmetic which matter in our setting. Section 3 presents known and develops new approaches to the verified computation of $\exp(A)$. Section 4 shows how these

*Submitted to the editors DATE.

†Department of Mathematics, University of Wuppertal, 42097 Wuppertal, Germany (frommer@math.uni-wuppertal.de).

‡Department of Mathematics, Faculty of Basic Sciences, Shiraz University of Technology, Modares Boulevard, Shiraz 71555-313, Iran (hashemi@sutech.ac.ir).

[1]Denoting $\varepsilon_{\mathrm{mach}}$ the difference between the smallest floating point number > 1 and 1, the unit roundoff u is $\varepsilon_{\mathrm{mach}}/2$ when the rounding mode is rounding to nearest and $\varepsilon_{\mathrm{mach}}$ for directed roundings.

approaches can be combined with approximate diagonalization. We then present a variety of numerical experiments and comparisons in Section 5. Some conclusions are formulated in Section 6.

2. Interval arithmetic. This section summarizes those aspects of interval arithmetic which are most important for this paper. For a more thorough treatment we refer to the textbooks [23, 28] and the review paper [39].

Let $\mathbb{IR}$ denote the set of all compact intervals $\boldsymbol{x} = [\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$ on the real line. (Interval quantities will always be denoted in boldface.) The (standard) interval arithmetic operations $+, -, \cdot, /$ on $\mathbb{IR}$ are defined in the set theoretic sense. They again yield an element from $\mathbb{IR}$, the bounds of which can be obtained from the bounds of the interval operands. One way to extend the interval concept to the complex plane is to take $\mathbb{IC}_{\text{disc}}$ as the set of all compact disks $\boldsymbol{z}$ in the complex plane with center $\text{mid}\,(\boldsymbol{z})$ and radius $\text{rad}\,(\boldsymbol{z})$ and to define the result of an arithmetic operation $\boldsymbol{z}_1 \circ \boldsymbol{z}_2$ as the disk with center $\text{mid}\,(\boldsymbol{z}_1) \circ \text{mid}\,(\boldsymbol{z}_2)$ and smallest radius such that it still contains $\{z_1 \circ z_2 : z_1 \in \boldsymbol{z}_1, z_2 \in \boldsymbol{z}_2\}$. This radius can be computed from the midpoints and radii of the operands. This *circular* interval arithmetic can also be used on $\mathbb{IR}$ by restriction to the real axis. The results of multiplication and division are then, in general, supersets of what one gets from the standard interval arithmetic on $\mathbb{IR}$.

In a floating point environment, it is important that for any arithmetic operation $\circ \in \{+, -, \cdot, /\}$ interval arithmetic preserves the very crucial *enclosure property*

$$\{x \circ y : x \in \boldsymbol{x}, y \in \boldsymbol{y}\} \subseteq \boldsymbol{x} \circ \boldsymbol{y}. \tag{2.1}$$

This means that in the floating point computation of the lower and upper bound of the result (or its midpoint and radius), we have to use different directed rounding modes. On a given modern hardware, changing the rounding mode is a very time consuming operation as compared to the floating point computation itself. Efficient implementations of machine interval arithmetic as in the MATLAB Toolbox INTLAB [37] or the C++ library C-XSC [18, 16] therefore try to do as few changes of the rounding mode as possible, and this can be achieved by using an operator concept which works on whole arrays in the same spirit as the well-known BLAS (basic linear algebra subprograms). On $\mathbb{IR}$, circular arithmetic has then to be used, see [36]. It cannot be emphasized enough that these savings in switchings of the rounding modes affect run times very substantially: Interval computations then perform comparably fast than floating point computations, whereas without these techniques they are likely to be slower by at least two orders of magnitude.

Trivially, the enclosure property (2.1) carries over to any rational expression $r(x_1, \ldots, x_n)$,

$$\{r(x_1), \ldots, r(x_n) : x_i \in \boldsymbol{x}_i \text{ for } i = 1, \ldots, n\} \subseteq r(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n).$$

If any of the variables x_i appears several times in r we typically encounter the phenomenon of *overestimation* inherent in the use of interval arithmetic, which treats each occurence of a variable as being independent of its other occurrences. A very simple case is the expression $r(x) = x * x$, which for an interval $\boldsymbol{x} \in \mathbb{IR}$ with $0 \in \boldsymbol{x}$ gives

$$r(\boldsymbol{x}) = [-|\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}|, \max\{\underline{\boldsymbol{x}}^2, \overline{\boldsymbol{x}}^2\}] \supsetneq [0, \max\{\underline{\boldsymbol{x}}^2, \overline{\boldsymbol{x}}^2\}] = \{r(x) : x \in \boldsymbol{x}\}.$$

We face a similar situation when we use interval arithmetic to compute the square

$\boldsymbol{B} = \boldsymbol{A} \cdot \boldsymbol{A}$ of an *interval matrix* $\boldsymbol{A} = (\boldsymbol{a}_{ij})_{i,j=1}^n$. In the expression

$$\boldsymbol{b}_{ij} = \sum_{k=1}^{n} \boldsymbol{a}_{ik} \cdot \boldsymbol{a}_{kj}$$

the entry $\boldsymbol{a}_{ij}$ is the only one which occurs more than once, either in $\boldsymbol{a}_{ij}\boldsymbol{a}_{jj}$ and $\boldsymbol{a}_{ii}\boldsymbol{a}_{ij}$ if $i \neq j$ or in $\boldsymbol{a}_{ii}\boldsymbol{a}_{ii}$ if $i = j$. INTLAB as well as virtually any other interval software provides a function $(\cdot)^2$ for intervals which returns (up to roundings) the exact range of the second power for any interval argument. Using this and replacing $\boldsymbol{a}_{ij}\boldsymbol{a}_{jj} + \boldsymbol{a}_{ii}\boldsymbol{a}_{ij}$ by $(\boldsymbol{a}_{ii} + \boldsymbol{a}_{jj})\boldsymbol{a}_{ij}$ in case $i \neq j$ will thus give, in general, narrower intervals for the diagonal of $\boldsymbol{B}$, but for computational efficiency it is important that the whole computation can still be cast into operations on arrays without explicit loops and case distinctions. The following self-explaining MATLAB-INTLAB code shows how this can be achieved using pointwise multiplication:

```
function S = square(A)
n = size(A,1);
c = diag(A);
A(1:n+1:end) = intval(0); % A(i,i) = intval(0) = [0,0];

C = ones(n,1)*c' + c * ones(1,n);
C = C.*A;
C(1:n+1:end) = c.^2;      % C(i,i) = c(i)^2;

S = A*A + C;
end
```

As was observed in [19], proceeding this way we obtain, up to roundings, the *interval hull* $\boldsymbol{S}$ of the set $\mathcal{S} := \{A^2 : A \in \boldsymbol{A}\}$, i.e. the intersection of all interval matrices containing $\mathcal{S}$. Note that $\mathcal{S}$ itself is not an interval matrix. So if we perform another squaring with $\boldsymbol{S}$, we will get the interval hull of all the squares of matrices from $\boldsymbol{S}$ which is in general *more* than the interval hull of the squares of the matrices from $\mathcal{S}$ and is thus larger than the interval hull of the set $\{A^4 : A \in \boldsymbol{A}\}$. It will be important to be aware of this *wrapping effect* when considering scaling and squaring approaches in this paper.

If the two end-points of an interval coincide it is termed a *point interval.* Performing machine interval arithmetic with point intervals yields non-point intervals which contain the exact value of the computation. Interval arithmetic can thus be used as a tool for an automated forward error analysis yielding lower and upper bounds for (arithmetic) expressions involving point quantities. For a more involved computation, though, such a naive use of interval arithmetic will typically end up with quite wide intervals. To obtain narrow enclosures, specific interval methods have to be used. For example, rather than just performing Gaussian elimination in interval arithmetic to solve a linear system $Ax = b$, a narrow enclosure for the solution is obtained by a correction $\boldsymbol{x}$, an interval vector, to an approximate solution $\tilde{x}$, obtained via some floating point computation. The vector $\boldsymbol{x}$ is determined in a such a way that

$$-R(A\tilde{x} - b) + (I - RA)\boldsymbol{x} \subseteq \operatorname{int}\boldsymbol{x} \tag{2.2}$$

with R being an approximate inverse for A, again computed in standard floating point arithmetic. By a result from [20, 35], based on Brouwer's fixed point theorem, A is non-singular then and $A^{-1}b \in \tilde{x} + \boldsymbol{x}$. As a side remark let us mention that it was recently shown in [2] that if one uses the restriction of circular arithmetic to the reals to evaluate the left hand side in (2.2), a much simpler fixed point theorem than Brouwer's can be used to show that $A^{-1}b \in \tilde{x} + \boldsymbol{x}$. The outlined method is the basis of the INTLAB function `verifylss.m`, see also [41], which will be heavily used in our algorithms.

As a final remark in this section, let us note that when computing a higher power $\boldsymbol{A}^k, k \geq 3$ of an interval matrix, the result will typically depend on the order that we choose for its evaluation. This means that in general we have

$$(\boldsymbol{A} \cdot \boldsymbol{A}) \cdot \boldsymbol{A} \neq \boldsymbol{A} \cdot (\boldsymbol{A} \cdot \boldsymbol{A}),$$

with both sets containing $\{A^3; A \in \boldsymbol{A}\}$. In order to reduce wrapping effects, higher powers of interval matrices should be computed in a way to minimize the number of matrix multiplications. For example, for $k = 2^s$ we need just s multiplications if we work recursively $\boldsymbol{S} = \boldsymbol{A}, \boldsymbol{S} \leftarrow \boldsymbol{S}^2$ for $i = 1, \dots, s$, and if k is not a power of 2, the Patterson-Stockmeyer approach [32] also aims at keeping the number of matrix multiplications small.

3. Enclosure methods for the matrix exponential.

We start this section with a detailed discussion in the scaling and squaring approach which turns out to be crucial for the enclosure methods, too. We then proceed by introducing the different enclosure methods, grouping them by the respective approaches they use to approximate the exponential of the scaled matrix and discussing the variants resulting from different ways to perform the interval arithmetic operations involved.

3.1. The scaling and squaring framework.

It is easier to well approximate $\exp(A)$ when $\|A\|$ is small. This is why scaling and squaring is an ingredient to the majority of methods to compute $\exp(A)$. It relies on the simple identity

$$\exp(A) = \left(\exp\left(\frac{A}{2^s}\right)\right)^{2^s}.$$

Higham [15] notes that the main issue in the accuracy of a scaling and squaring method is the significant rounding errors which might occur as a result of severe numerical cancellation in the squaring phase. The fundamental problem can be seen in the result [12, sec. 3.5]

$$\|A^2 - fl(A^2)\|_p \leq \gamma_n \|A\|_p^2,$$

in which $p \in \{1, \infty, F\}$ and $\gamma_n := \frac{nu}{1-nu}$. Here u is the unit roundoff and fl denotes the result obtained in floating point arithmetic. This shows that the errors in the computed squared matrix are small compared with the square of the norm of the original matrix but not necessarily small compared with the matrix being computed. It is therefore important to keep the number s of squaring steps small. The current state of the art is the algorithm called `expm_new` from [1] used in MATLAB's `expm` function. It improves over the classical approach which chooses s based on $\|A\|$ alone by now also involving $\|A^k\|^{1/k}$ for modest powers k. Since $\rho(A) \leq \|A^k\|^{1/k} \leq \|A\|$ for $k = 1, \dots, \infty$ this new approach tends to yield smaller values for s.

From our experiments and the results in [10] it is apparent that enclosure methods for the matrix exponential have to rely on the scaling and squaring technique, too. For two reasons we use the classical scaling and squaring strategy, based solely on $\|A\|$ in our enclosure methods: First, in a guaranteed, interval-arithmetic method we also have to involve bounds on the approximation error. The second reason is related to the fact that we consider two variations of each algorithm as explained in the beginning of section 4 below; we have observed that in the second variation, where we apply our algorithms to a transformation $\tilde{\boldsymbol{A}}$ of A, the old and new scaling strategies compute the *same* scaling factor s, and so the second variation of our algorithms already prevents overscaling.

3.2. Spectral decomposition for diagonalizable matrices. Assume that A is diagonalizable, i.e.

$$A = VDW,$$

where $VW = I$ and $D = \operatorname{diag}(d_1, \ldots, d_n)$ is diagonal. The exponential of A is then given as

$$\exp(A) = V \exp(D) W.$$

An enclosure method results if we are able to compute interval matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ such that $V \in \boldsymbol{V}, W \in \boldsymbol{W}$ and intervals $\boldsymbol{d}_i$ containing the eigenvalues d_i. We then have

$$\exp(A) \in \boldsymbol{V} \cdot \operatorname{diag}(\exp(\boldsymbol{d}_1), \ldots, \exp(\boldsymbol{d}_n)) \cdot \boldsymbol{W}, \tag{3.1}$$

where we assume that we are able to evaluate the exponential function on intervals in a way that the result is guaranteed to contain its range over that interval. This is possible with the standard function implementations for intervals present in INTLAB or C-XSC, e.g.

The approach outlined here is taken by the `vermatfun.m` routine of the VERSOFT package [34], which, by calling the `verifyeig.m` function of INTLAB [37], uses interval arithmetic to first compute enclosures $(\boldsymbol{v}_i, \boldsymbol{d}_i)$ for all eigenpairs $(v_i, d_i), i = 1, \ldots, n$ and then obtains $\boldsymbol{W}$ by computing an interval matrix $\boldsymbol{W}$ which is guaranteed to contain all solutions $\widetilde{W}$ to all linear systems of the form $\tilde{V}\tilde{W} = I$ for $\tilde{V} \in \boldsymbol{V} := [\boldsymbol{v}_1 \mid \cdots \mid \boldsymbol{v}_n]$. This is done using the INTLAB function `verifylss.m`. Note that `vermatfun.m` is applicable to general matrix functions, not just the exponential.

This approach has two drawbacks. First, since computing an enclosure for just one eigenpair has complexity $\mathcal{O}(n^3)$, its overall complexity is $\mathcal{O}(n^4)$. Second, if the eigenvector matrix is ill conditioned, $\boldsymbol{W}$ will consist of relatively wide intervals such that the right hand side of (3.1) will have wide interval entries, too. As illustrated in section 5, the same issue arises in the presence of eigenvalue clusters.

Recently, Miyajima [25] presented an enclosure method which requires an approximate spectral decomposition $A \approx VDV^{-1}$ only, and then constructs an interval matrix $\boldsymbol{M}$ which uses an enclosure $\boldsymbol{S}$ for the residual quantity $V^{-1}(AV - VA)$ obtained using interval arithmetic and additional bounds for other quantities to obtain the enclosure $\exp(A) \in V^{-1} \exp(D) \boldsymbol{M} V$. This algorithm has complexity $\mathcal{O}(n^3)$, and its accuracy crucially depends on the quality of the enclosure $\boldsymbol{S}$, for which evaluating $AV - VA$ in interval arithmetic is not sufficient. We will discuss this somewhat more in section 5. The paper [25] also presents an extension to defective matrices, where the spectral decomposition is replaced by what is called a *numerical Jordan decomposition*. The complexity then increases to $\mathcal{O}(n^4)$.

3.3. Taylor approximations. Since for any matrix $A \in \mathbb{C}^{n \times n}$ we have

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k,$$

we can use the first $d+1$ terms of this Taylor expansion to obtain the approximation

$$T_d(A) := I + A + \frac{1}{2!}A^2 + \cdots + \frac{1}{d!}A^d. \tag{3.2}$$

The following results on bounds for the truncation error hold, where the first part is due to Liou [21] and the second to Suzuki, see [14, 45].

THEOREM 3.1. *Let $\|\cdot\|$ be the operator 1-, 2- or ∞-norm ad assume $d+2 > \|A\|$. Then we have*

$$\|\exp(A) - T_d(A)\| \le \vartheta(d, \|A\|) := \frac{\|A\|^{d+1}}{(d+1)!(1 - \frac{\|A\|}{d+2})}. \tag{3.3}$$

Moreover, $T_{d,s}(A) := \big(T_d(A/s)\big)^s$ for $s \in \mathbb{N}$ satisfies

$$\|\exp(A) - T_{d,s}(A)\| \le \frac{\|A\|^{d+1}}{s^d(d+1)!}\exp(\|A\|).$$

A consequence of (3.3) is

$$\exp(A) \in T_d(A) + \vartheta(d, \|A\|)\boldsymbol{E}, \tag{3.4}$$

where here as in the sequel $\boldsymbol{E}$ denotes the interval matrix with all entries equal to $[-1.1]$.

In Oppenheimer's PhD thesis [30], it was suggested to use the centered form of the truncated Taylor series (3.2) in order to enclose $\exp(A)$. The algorithm, published later in [31], bounds the truncation error by Liou's error bound (3.3). Taylor series are also used in [43] for the accurate computation of the exponential of essentially nonnegative matrices.

Goldsztejn and Neumaier [10] proposed an enclosure method using scaling and squaring based on the truncated Taylor series, the enclosure (3.4) and a variant of Horner's scheme to evaluate $T_d(A)$ in interval arithmetic according to

$$T_d(A) = I + A\big(I + \tfrac{1}{2}A(I + \cdots + \tfrac{1}{d-1}A(I + \tfrac{1}{d}A)\cdots)\big). \tag{3.5}$$

We formulate their method as Algorithm 3.1.

Algorithm 3.1 Outline of the truncated Taylor series based enclosure method [10]

1: Scale the matrix so that $\|\frac{1}{2^s}A\| \le 0.1$, i.e. $s = \max\{0, \lceil \log_2(10 \cdot \|A\|)\rceil\}$
2: Determine the smallest integer d such that the truncation error bound from Theorem 3.1 is less than $\varepsilon_{\text{mach}}$. ($d = 9$ in double precision.)
3: Obtain the interval matrix $\boldsymbol{T}_d$ by evaluating T_d for the (scaled) matrix using interval arithmetic (to account fo rounding errors).
4: Use interval arithmetic to compute an upper bound $\overline{\vartheta}$ for for $\vartheta(d, \frac{1}{2^s}\|A\|)$ from (3.4) and compute $\boldsymbol{C} = \boldsymbol{T}_d + \overline{\vartheta}\boldsymbol{E}$.
5: Perform s repeated squarings starting with $\boldsymbol{C}$. The final result is an enclosure for $\exp(A)$.

In [10], $\|\cdot\|$ is taken to be the ∞-norm and $\boldsymbol{T}_d$ is obtained via Horner's scheme (3.5). The squarings are done in an optimal way according to the function `square` from section 2. The choice for the ∞-norm is in particular motivated by the fact that for this norm one can show that the radii of the computed enclosures decrease monotonically with d, the degree of the truncated Taylor approximation. Interestingly, if the norm of the scaled matrix is less than 0.1, $d = 9$ already achieves $\overline{\vartheta} < \varepsilon_{\text{mach}}$ in double precision. It is also shown in [10] that the Horner scheme (3.5) yields substantially narrower intervals as compared to a "standard" interval arithmetic evaluation of $T_d(A)$ which first computes all powers of A and then their scaled sum.

In an attempt to obtain smaller radii for the computed enclosures, we implemented Algorithm 3.1 with the following two modifications:

$$\left\{ \begin{array}{l} \text{replace the } \infty\text{-norm by the 2-norm} \\ \text{evaluate } T_d(A) \text{ using the Paterson-Stockmeyer approach} \end{array} \right. \tag{3.6}$$

Using the 2-norm is motivated by the fact that, typically, the 2-norm is smaller than the ∞-norm – it is certainly not larger than the ∞-norm for Hermitian matrices – so that using $\|A\|_2$ is likely to require less scalings and also to yield a smaller value for $\overline{\vartheta}$ from (3.3). In INTLAB, an interval enclosure, and thus an upper bound, for $\|A\|_2$ is computed with $\mathcal{O}(n^3)$ operations, see [40], and thus at a cost comparable to the other computations of the algorithm. Using the Paterson-Stockmeyer approach reduces the number of matrix-matrix multiplications and thus the number of wrappings in interval arithmetic. Details on the Paterson-Stockmeyer approach can be found in [32]; here we just give it for the case $d = 9$ where $T_9(A)$ is evaluated according to

$$T_9(A) = I + A + \tfrac{1}{2!}A^2 + A^3\Big(\big(\tfrac{1}{3!}I + \tfrac{1}{4!}A + \tfrac{1}{5!}A^2\big) + A^3\left(\tfrac{1}{6!}I + \tfrac{1}{7!}A + \tfrac{1}{8!}A^2 + \tfrac{1}{9!}A^3\right) \Big),$$

which requires just one squaring (for A^2) and three matrix-matrix multiplications (including one for $A^3 = A \cdot A^2$). Note that evaluation of $T_9(A)$ according to Horner's scheme (3.5) requires nine matrix-matrix multiplications.

3.4. Padé approximation. The type (k, m) Padé approximation to the scalar exponential function is given as

$$\exp(z) = \frac{p_k(z)}{q_m(z)} + r^{km}(z),$$

where $p_k(z)$ and $q_m(z)$ are polynomials of degree k and m, respectively, with $q_m(0) = 1$, and the remainder term $r^{km}(z)$ satisfies $r^{km}(x) = \mathcal{O}(x^{k+m+1})$, see [14, p. 79].

In the matrix case, the type (k, m) Padé approximation to $\exp(A)$ is thus

$$\exp(A) \approx P_{km}(A) = q_m(A)^{-1}p_k(A),$$

and we have

$$q_m(A) \cdot \exp(A) = p_k(A) + t^{km}(A),$$

where $t^{km}(A) = q_k(A)r^{km}(A)$. The following theorem gives bounds for every entry of the matrix $T^{km} := t^{km}(A)$.

THEOREM 3.2. *Let $\|\cdot\|$ be any submultiplicative matrix norm. Then*

$$\|T^{km}\| \le \pi(k, m, \|A\|) := \frac{k!\ m!}{(k+m)!\ (k+m+1)!}\|A\|^{k+m+1}\exp(\|A\|), \tag{3.7}$$

and if $\|\cdot\|$ is the 1-, 2- or ∞-norm, then T^{km} satisfies

$$T^{km} \in \pi(k, m, \|A\|)\boldsymbol{E}.$$

Proof. A classical result from [49] (see also [14, p. 241]) establishes the representation

$$r^{km}(A) = \frac{(-1)^m}{(k+m)!} A^{k+m+1}\; q_m(A)^{-1} \int_0^1 \exp(tA)(1-t)^k t^m dt. \tag{3.8}$$

Multiplying both sides of (3.8) by $q_m(A)$ and using the fact that rational functions of the same matrix commute (see, e.g., [14, Thm. 1.13]), we get

$$T^{km} = q_m(A)\; r_{km}(A) = \frac{(-1)^m}{(k+m)!} A^{k+m+1} \int_0^1 \exp(tA)(1-t)^k t^m dt. \tag{3.9}$$

Since the Taylor series of the exponential has positive coefficients only, we have that $\|\exp(A)\| \leq \exp(\|A\|)$ for any submultiplicative matrix norm. Taking norms in (3.9) thus gives

$$\begin{aligned}
\|T\| &\leq \frac{\|A\|^{k+m+1}}{(k+m)!} \int_0^1 \exp(t\|A\|)(1-t)^k t^m dt \\
&= \frac{\|A\|^{k+m+1}}{(k+m)!} \exp(\theta\|A\|) \int_0^1 (1-t)^k t^m dt, \quad \theta \in [0,1] \\
&= \frac{\|A\|^{k+m+1}}{(k+m)!} \exp(\theta\|A\|) \frac{k!\; m!}{(k+m+1)!} \\
&\leq \frac{k!\; m!}{(k+m)!\; (k+m+1)!} \|A\|^{k+m+1} \exp(\|A\|) = \pi(k,m,\|A\|).
\end{aligned}$$

The equality in the second line holds by the generalized mean value theorem. We thus have established the first part of the theorem, and its second part holds because the 1-, 2- and ∞ norms of a matrix are all larger than or equal to the absolute value of any of the matrix entries. ∎

An important aspect of Padé approximation is its efficiency compared to Taylor series, see the discussion in [14], e.g. Since we work with IEEE double precision in all our numerical computations, we choose $m = k = 7$ which gives $\pi \approx 6.06 \times 10^{-16}$ for $\|A\| \leq 1$ which is our target value for the scaling phase. The coefficients $b = (b_0, \ldots, b_7)$ of the polynomial $p_7(x) = \sum_{i=0}^{7} b_i x^i$ are the integers

$$b = \begin{bmatrix} 17{,}297{,}280 & 8{,}648{,}640 & 1{,}995{,}840 & 277{,}200 & 25{,}200 & 1{,}512 & 56 & 1 \end{bmatrix},$$

see [14, p. 246]. Moreover, $q_m(x) = p_m(-x)$ for all m. Our implementation represents an interval arithmetic extension of the method outlined in [14, p. 244] to compute the interval matrices $\boldsymbol{P} \ni p_k(A)$ and $\boldsymbol{Q} \ni q_m(A)$. To be specific, we take $\boldsymbol{P} = \boldsymbol{V} + \boldsymbol{U}$ and $\boldsymbol{Q} = \boldsymbol{V} - \boldsymbol{U}$ where

$$\left\{ \begin{array}{l} \boldsymbol{U} := A(b_7\boldsymbol{A}_6 + b_5\boldsymbol{A}_4 + b_3\boldsymbol{A}_2 + b_1 I), \\ \boldsymbol{V} := b_6\boldsymbol{A}_6 + b_4\boldsymbol{A}_4 + b_2\boldsymbol{A}_2 + b_0 I, \end{array} \right. \tag{3.10}$$

and $\boldsymbol{A}_2$ is the (optimal) enclosure for A^2 obtained via the function square from section 2 applied to A, and, similarly, $\boldsymbol{A}_4$ is an enclosure for A^4 computed as the square of $\boldsymbol{A}_2$, while $\boldsymbol{A}_6$ is an enclosure for A^6 computed as $\boldsymbol{A}_2 \cdot \boldsymbol{A}_4$. In this way, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are computed with only two interval matrix-matrix multiplications and two squarings.

Once $\boldsymbol{P}$ and $\boldsymbol{Q}$ are computed, we use Theorem 3.2, which shows that $\exp(A)$ is contained in the solution set of the interval linear system

$$\boldsymbol{Q}X = \boldsymbol{P} + \overline{\pi}\boldsymbol{E}, \tag{3.11}$$

where $\overline{\pi}$ is an upper bound for $\pi(k, m, \|A\|)$ from (3.7) obtained using an interval arithmetic evaluation. We take INTLAB's function `verifylss.m` to compute an interval enclosure for the solution set of (3.11).

Algorithm 3.2 summarizes the enclosure method based on Padé approximation when working in double precision where a $(7, 7)$ Padé approximation is sufficient to bound the approximation error by $\varepsilon_{\text{mach}}$ for matrices with norm ≤ 1.

Algorithm 3.2 Outline of the Padé approximation based enclosure method

1: Scale the matrix so that $\|\frac{1}{2^s}A\| \le 1$, i.e. $s = \max\{0, \lceil \log_2(\|A\|) \rceil\}$
2: Compute enclosures $\boldsymbol{P} = \boldsymbol{U} + \boldsymbol{V}$ and $\boldsymbol{Q} = \boldsymbol{U} - \boldsymbol{V}$ for the two polynomials in the $(7, 7)$ Padé approximation, with $\boldsymbol{U}, \boldsymbol{V}$ computed according to (3.10)
3: Compute an upper bound $\overline{\pi}$ for $\pi(7, 7, \frac{1}{2^s}\|A\|)$ via an interval arithmetic evaluation of (3.7)
4: Use INTLAB's function `verifylss.m` to obtain an interval matrix $\boldsymbol{C}$ containing the solution set of the interval linear system (3.11)
5: Perform s repeated squarings starting with $\boldsymbol{C}$. The final result is an enclosure for $\exp(A)$.

For the same reasons as in the Taylor series approach, we chose the norm to be the 2-norm in our computations.

It should be noted that Bochev [5] has already applied a Padé-based algorithm for enclosing $\exp(A)$ which also uses the representation (3.8) of the error. The Padé approach we present here is different in at least two important aspects: Our approach relies directly on Theorem 3.2 and does thus not need to compute a rough enclosure for $\exp(A)$ which is required in [6, 5] in a preliminary step. Moreover, [6, 5] has to use the computationally expensive staggered correction format [44] to accurately bound the polynomials p_k and q_k and to enclose solutions of the interval linear system (3.11). Staggered correction formats or other (expensive) means to enhance floating point accuracy are not required in our approach.

3.5. Contour integration. For the exponential as for any other analytic function, Cauchy's formula

$$\exp(a) = \frac{1}{2\pi i}\int_\Gamma \frac{\exp(z)}{z-a}dz, \tag{3.12}$$

where Γ is a contour in the complex plane that encloses the point a, carries over to the matrix function case as

$$\exp(A) = \frac{1}{2\pi i}\int_\Gamma \exp(z)(zI - A)^{-1}dz, \tag{3.13}$$

provided the spectrum of A is enclosed by Γ, see [14]. We now develop an enclosure method based on quadrature for (3.12) and a rigorous bound for the remainder term. We will scale A such that $\|A\| < 1$, and therefore take Γ to be the unit circle

$$e^{i\theta}, \theta \in [0, 2\pi].$$

Then (3.12) becomes

$$\exp(a) = \int_0^{2\pi} v(\theta)d\theta, \text{ where } v(\theta) = \frac{\exp(e^{i\theta})}{2\pi(e^{i\theta} - a)}e^{i\theta} \text{ (with } |a| < 1).$$

The function v is 2π-periodic, which is why the standard trapezoidal rule

$$\exp(a) = \underbrace{\frac{2\pi}{N}\sum_{k=1}^{N} v(\theta_k)}_{:=I_N(v)} + R_N(v)$$

with $N \in \mathbb{N}$ and $\theta_k = 2\pi k/N, k = 1,\ldots,N$, has a small error R_N. Indeed, the following result holds, see [48, Thm. 3.2],[51].

Lemma 3.3. *Suppose v is 2π-periodic and analytic and satisfies $|v(\theta)| \le M(c)$ in the strip $-c < \Im(\theta) < c$ for some $c > 0$. Then for any $N \ge 1$,*

$$\left|\int_0^{2\pi} v(\theta)d\theta - I_N(v)\right| \le \frac{4\pi M(c)}{e^{cN}-1},$$

and the constant 4π is as small as possible.

In order to use this result for the matrix case, recall that by Banach's lemma (see [7, p. 33], e.g.), we have that for any operator norm $\|\cdot\|$

$$|z| > \|A\| \Rightarrow \|(zI - A)^{-1}\| \le \frac{1}{|z| - \|A\|}. \tag{3.14}$$

If $\|\cdot\|$ is the 1-, 2- or ∞-norm, this implies in particular

$$\left|[(zI - A)^{-1}]_{ij}\right| \le \frac{1}{|z| - \|A\|},\ i,j = 1,\ldots,n \quad \text{for } |z| > \|A\|. \tag{3.15}$$

Theorem 3.4. *Let $\|A\| < 1$ where $\|\cdot\|$ is the 1-,2- or ∞-norm, let c be such that $e^{-c} > \|A\|$, and let $N \in \mathbb{N}$. Let $z_k = e^{2\pi ik/N}, k = 1,\ldots,N$. Then with*

$$\gamma(c, N, \|A\|) := \frac{2e^c \exp(e^c)}{(e^{cN}-1)(e^{-c} - \|A\|)} \tag{3.16}$$

we have

$$\exp(A) \in \frac{1}{N}\sum_{k=1}^{N} z_k \exp(z_k)(z_k I - A)^{-1} + \gamma(c, N, \|A\|)\cdot \boldsymbol{E}, \tag{3.17}$$

In particular, if $e^{-c} > 2\|A\|$, we have

$$\exp(A) \in \frac{1}{N}\sum_{k=1}^{N} z_k \exp(z_k)(z_k I - A)^{-1} + \gamma(c, N)\cdot \boldsymbol{E}, \tag{3.18}$$

where

$$\gamma(c, N) := \frac{4e^{2c}\exp(e^c)}{e^{cN}-1}. \tag{3.19}$$

Proof. We first note that (3.18) follows directly from (3.17), since $e^{-c} > 2\|A\|$ implies $\frac{1}{e^{-c} - \|A\|} \le 2e^c$. With Γ the unit circle $z = e^{i\theta}, \theta \in [0, 2\pi]$, Cauchy's formula (3.13) expresses each entry $[\exp(A)]_{ij}$ of the matrix $\exp(A)$ as

$$[\exp(A)]_{ij} = \int_0^{2\pi} \underbrace{\frac{1}{2\pi}\exp(e^{i\theta})[(e^{i\theta}I - A)^{-1}]_{ij}e^{i\theta}}_{=:v_{ij}(\theta)}\, d\theta.$$

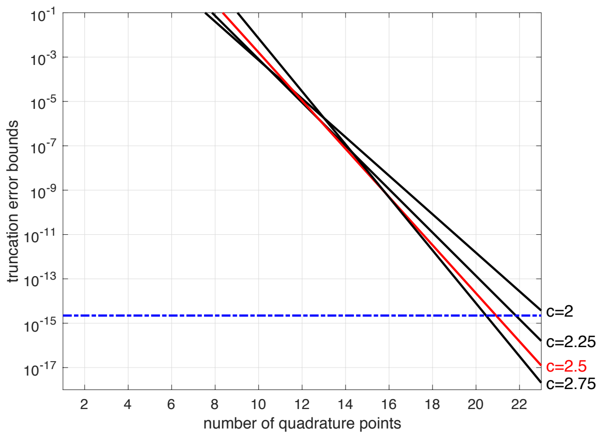

FIG. 1. *Bounds for the quadrature error of the periodic trapezoidal rule. The blue line represents* 2.2×10^{-15}.

Herein, by (3.14), v_{ij} is defined and analytic on an open superset of the strip $D^c = -c \leq \Im(\theta) \leq c$ for c with $e^{-c} > \|A\|$, and it is 2π periodic. Using (3.15), one then obtains

$$\max_{\theta \in D^c} |v_{ij}(\theta)| \leq \frac{\exp(e^c)}{2\pi} \frac{1}{e^{-c} - \|A\|} e^c =: M(c).$$

By Lemma 3.3, we thus get

$$\left| [\exp(A)]_{ij} - \frac{1}{N} \sum_{k=1}^{N} z_k \exp(z_k) \left[(z_k I - A)^{-1} \right]_{ij} \right| \leq \frac{2M(c)}{e^{cN} - 1},$$

which gives (3.17). ∎

The enclosure method based on contour integration will have to compute an enclosure for each of the inverses $(z_k I - A)^{-1}$. Using `verifylss.m` to that purpose will for each k give results where each entry has at least a relative width of $\varepsilon_{\text{mach}}$. Since we expect to have $\mathcal{O}(10)$ of these systems to solve, it is adequate to require the bound on the quadrature error to be approximately $10\varepsilon_{\text{mach}}$. In order to keep the computational cost low, in our algorithm we therefore choose c such that N is minimal under all pairs (N, c) which satisfy

$$\frac{4e^{2c} \exp(e^c)}{(e^{cN} - 1)} \leq 10\varepsilon_{\text{mach}} \approx 2.2 \times 10^{-15}.$$

Figure 1 illustrates that this is (approximately) achieved for $c = 2.5$ with $N = 21$, implying that A is scaled such that $\|\frac{1}{2^s} A\| \leq 0.03$. This might seem restrictive, but note that if we relax the scaling to just satisfy $\|\frac{1}{2^s} A\| \leq 0.18$, for example, then we would need $c = 1$ and $N = 40$, thus doubling the computational cost. Also note that in our numerical experiments other choices than $c = 2.5, N = 21$ gave comparable if not larger radii for the enclosure obtained for $\exp(A)$.

Algorithm 3.3 summarizes our approach based on contour integration. In Step 2 we use INTLAB's `verifypoly.m` to obtain as narrow as possible enclosures $\boldsymbol{z}_k$ for the roots z_k of the polynomial $z^N - 1$; see [29, 38] for an overview of relevant techniques. Let us also note that in case that A is real we have that $z_k I - A = \overline{(z_{N-k-1} I - A)}$,

Algorithm 3.3 Outline of the contour integration based enclosure method

1: Choose $c = 2.5$ and scale the matrix so that $\|\frac{1}{2^s}A\| \leq \frac{1}{2}e^{-c}$, i.e. $s = \max\{0, \lceil\log_2(2e^c\|A\|)\rceil\}$
2: Put $N = 21$ and compute interval enclosures $\boldsymbol{z}_k$ for the roots of unity $z_k = e^{2\pi ik/N}, k = 1, \ldots, N$.
3: Use INTLAB's `verifylss.m` to compute an interval matrix $\boldsymbol{S}_k$ containing $\{(zI - A)^{-1} : z \in \boldsymbol{z}_k\}$.
4: Compute an upper bound $\overline{\gamma}$ for $\gamma(N, c)$ via an interval arithmetic evaluation of (3.19) to get the enclosure $\boldsymbol{C} = \sum_{k=1}^{N} \boldsymbol{S}_k + \overline{\gamma}\boldsymbol{E}$ for the exponential of the scaled matrix
5: Perform s repeated squarings starting with $\boldsymbol{C}$. The final result is an enclosure for $\exp(A)$.

so $(z_kI - A)^{-1} = \overline{(z_{N-k-1}I - A)^{-1})}$ which can be used to approximately halve the computational cost. Specifically, we then invert only 11 interval matrices rather than $N = 21$.

As the Padé approach, the contour integration approach is based on a rational approximation. In the Padé scheme we have to once enclose the solution of the interval linear system (3.11) where all entries of the system matrix and of the right hand side are intervals. In the contour integration approach we have to enclose the inverses of several matrices where only the diagonals contain non-point quantities, and we need more squarings.

3.6. Chebyshev approximation. The Chebyshev polynomials T_k for the interval $[-1, 1]$ are the orthogonal polynomials with respect to the inner product $\langle f, g\rangle = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)g(x)dx$ on the space of continuous functions on $[-1, 1]$. They satisfy $T_k(x) = \cos(k \arccos x)$ for $x \in [-1, 1]$ and obey the recurrence

$$\begin{cases} T_0 = 1,\ T_1 = x, \\ T_k = 2xT_{k-1} - T_{k-2},\ k = 2, 3, \ldots, \end{cases} \tag{3.20}$$

and the (formal) Chebyshev series of a Lipschitz continuous function f on $[-1 - 1]$ is given as

$$\sum_{k=0}^{\infty} \frac{\langle f, T_k\rangle}{\langle T_k, T_k\rangle} T_k.$$

For $f = \exp$ the coefficients of the Chebyshev series are given via the modified Bessel functions of the first kind $I_k(t) = \frac{1}{\pi}\int_0^{\pi} \exp(t\cos(\theta))\cos(k\theta)\, d\theta$ as

$$\frac{\langle \exp, T_0\rangle}{\langle T_0, T_0\rangle} = I_0(1),\quad \frac{\langle \exp, T_k\rangle}{\langle T_k, T_k\rangle} = 2I_k(1), k = 1, 2, \ldots,$$

see [22, p. 109] and [46, p. 23] e.g.

The use of Chebyshev series for approximating $\exp(A)b$ where A is Hermitian and b is a vector was suggested by Druskin and Knizhnerman [9]. In [3], $\exp(A)$ is computed for sparse A using the degree 17 truncated Chebyshev series. Its advantage is that, typically, for the same degree, the polynomial approximation given by the truncated Chebyshev series will give a more accurate approximation than a Taylor polynomial if one considers the whole interval $[-1, 1]$. For interval arithmetic based enclosure methods this means that we can scale less and thus save some squarings and the associated wrappings as compared to the Taylor approach. This motivates our investigation of Chebyshev approximation in this work. Its use is restricted to Hermitian matrices A, however, because we do not have a useable error bound for general A. To state an enclosure result for the Hermitian case, recall that the Bernstein ellipse E_ρ is an ellipse with center at zero and foci at ± 1 whose parameter $\rho > 1$ is the sum of its semi-axis lengths. The following result can be found in [46, Thm. 8.2], e.g.

LEMMA 3.5. *Let f be analytic in $[-1, +1]$ and analytically continuable to the open Bernstein ellipse E_ρ, where it satisfies $|f(z)| \le M(\rho)$. Then, for each $d \ge 0$, the truncated Chebyshev series $p_d = \sum_{k=0}^{d} \frac{\langle f, T_k\rangle}{\langle T_k, T_k\rangle} T_k$ satisfies*

$$|f(x) - p_d(x)| \le \frac{2M(\rho)\rho^{-d}}{\rho - 1} \text{ for } x \in [-1, 1].$$

THEOREM 3.6. *Let A be Hermitian with spectrum in $[-1, 1]$ and let $p_d(A)$ be the degree d truncated Chebyshev series approximation*

$$p_d(A) = I_0(1)I + \sum_{k=1}^{d} 2I_k(1) \cdot T_k(A) \tag{3.21}$$

for $\exp(A)$*. Then, with $\tau(\rho, d)$ defined for $\rho \ge 1$ as*

$$\tau(\rho, d) := 2e^{\frac{\rho+\rho^{-1}}{2}} \frac{\rho^{-d}}{\rho-1} \tag{3.22}$$

we have

$$\exp(A) \in p_d(A) + \tau(\rho, d)\boldsymbol{E}.$$

Proof. Since A is Hermitian we have $A = VDV^{-1}$ with V being orthonormal and $D = \mathrm{diag}(\lambda_i)$ is the diagonal matrix containing the eigenvalues. Also, $T_k(A) = VT_k(D)V^{-1}$ for all k and, thus, $p_d(A) = Vp_d(D)V^{-1}$. We thus have

$$\begin{aligned} |r_{ij}| \le \|R\|_2 &= \|\exp(A) - p_d(A)\|_2 = \|V(\exp(D) - p_d(D))V^{-1}\|_2 \\ &= \|\exp(D) - p_d(D)\|_2 = \|\exp(D) - p_d(D)\|_\infty \\ &= \max_i |\exp(\lambda_i) - p_d(\lambda_i)|. \end{aligned}$$

The fact that the maximum value of the exponential on E_ρ is $e^{\frac{\rho+\rho^{-1}}{2}}$, see [47], together with Lemma 3.5 now completes the proof. ∎

In an algorithm, we want to choose ρ and d such that $\tau(\rho) \approx \varepsilon_{\text{mach}}$ for d as small as possible. Figure 2 reports a Chebfun-enabled [8] experiment. It suggests that $d = 14$ (and $\rho = 32$) is an appropriate choice when working in double precision.

Also, in an interval arithmetic based enclosure method we need to use intervals enclosing the exact values $I_k(1)$ of the Bessel functions, and these should be as narrow as possible. We have used the Arb package [17], a C library for arbitrary-precision interval arithmetic to obtain enclosures $\boldsymbol{I}_k$ of relative radii of about $\varepsilon_{\text{mach}}$ in double precision for the exact values of $I_k(1)$ for $k = 0, 1, \ldots, 14$. Note that those must be computed only once and can then be stored for every later use.

Before summarizing our approach in Algorithm 3.4, we discuss how we evaluate the truncated Chebyshev sum $p_d(A)$. The direct way to evaluate $p_d(A)$ would be to precompute $T_k(A)$ for $k = 0, \ldots, d$ using the recurrence (3.20) and to subsequently

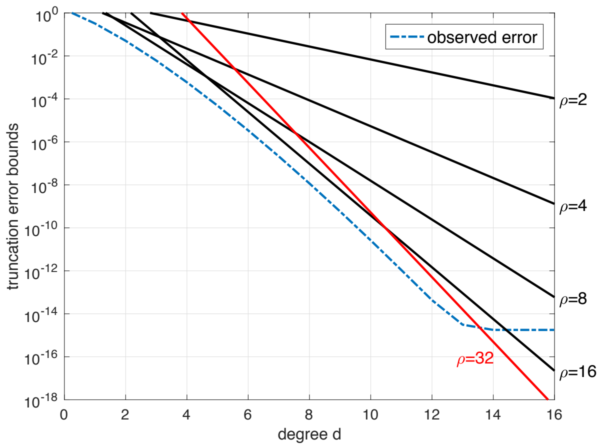

FIG. 2. *Truncation error in chopping Chebyshev series.*

evaluate the sum $I_0(1)I + \sum_{k=1}^{d} 2I_k(1)T_k(A)$. For the scalar case, and a general Chebyshev approximation $p_d(x) = \sum_{k=0}^{d} a_k T_k(x)$, it is known that the use of the *Clenshaw recurrence*, which interleaves the evaluation of the Chebyshev polynomials with the summation,

$$\begin{cases} b_{d+2} = b_{d+1} = 0, \\ b_j = 2xb_{j+1} - b_{j+2} + a_j, \quad j = d, d-1, \cdots, 0, \end{cases}$$

and then $p_d(x) = b_0 - xb_1$, is more stable, see [33, p. 173], e.g. Both, the direct way and the Clenshaw recurrence rely on three-term recurrences. When applied on matrices and using interval arithmetic, we thus not only experience the wrapping effect but also the fact that interval arithmetic treats the same variable occurring several times as being independent variables, thus having the tendency to further increase the width of the computed intervals. Analogous phenomena have been observed when enclosing scalar Chebyshev expansions of high-degree [11]. To alleviate this problem we suggest to use the matrix analogue of the product formulae

$$\begin{cases} T_{2k}(x) &= 2T_k^2(x) - 1 \\ T_{2k+1}(x) &= 2T_{k+1}(x)T_k(x) - x \end{cases}, \quad k = 1, 2, \ldots \tag{3.23}$$

with $T_0 = 1, T_1 = x$ to evaluate $T_k(A)$ using interval arithmetic. Therein, the squares can be computed using the method from section 2. This approach makes the total number of multiplications, and thus of the associated wrappings, small. For example, $T_8(A)$ is computed with just 3 squarings (for T_8, T_4 and T_2), and T_{14} is computed with 3 squarings (for T_{14}, T_4 and T_2) and two matrix multiplications (for T_7 and T_3).

4. Approximate diagonalization.

If V is non-singular and

$$D = V^{-1}AV \iff A = VDV^{-1},$$

we have, in exact arithmetic, $\exp(A) = V\exp(D)V^{-1}$. In floating point arithmetic, if we compute an enclosure $\boldsymbol{W}$ for V^{-1} and then $\boldsymbol{D}$ as $\boldsymbol{D} = \boldsymbol{W}AV$ using interval arithmetic, we have

$$\exp(A) \in V\boldsymbol{E}\boldsymbol{W}, \tag{4.1}$$

Algorithm 3.4 Outline of the Chebyshev-based enclosure algorithm (A Hermitian)

1: Scale the matrix so that $\|\frac{1}{2^s}A\| \leq 1$, i.e. $s = \max\{0, \lceil \log_2(\|A\|) \rceil\}$
2: Use interval arithmetic to compute enclosures for $T_k(A)$ for $k = 0, \ldots, 14$ via (3.23) and to then subsequently evaluate $\boldsymbol{S} = \boldsymbol{I}_0 + \sum_{k=1}^{14} 2\boldsymbol{I}_k \boldsymbol{T}_k(A)$, an enclosure for the value of the truncated Chebyshev series for the scaled matrix.
3: Compute an upper bound $\overline{\tau}$ for $\tau(32, 14)$ via an interval arithmetic evaluation of (3.22) to get the enclosure $\boldsymbol{C} = \boldsymbol{S} + \overline{\tau}\boldsymbol{E}$ for the exponential of the scaled matrix
4: Perform s repeated squarings starting with $\boldsymbol{C}$. The final result is an enclosure for $\exp(A)$

where $\boldsymbol{E}$ is enclosure for $\{\exp(D) : D \in \boldsymbol{D}\}$, and to compute $\boldsymbol{E}$, we can rely on any of the techniques presented in the previous section, replacing A by the interval matrix $\boldsymbol{D}$.

This observation can be used in an attempt to reduce the wrapping effect. Indeed, if $\boldsymbol{D}$ were diagonal, there would be no wrapping effect at all when computing powers of $\boldsymbol{D}$, and when the off-diagonal elements of $\boldsymbol{D}$ are small compared to the diagonal, the wrapping effect is also small. The price we pay are additional wrappings due to the multiplications with V and $\boldsymbol{W}$, and here the wrapping effect becomes large when V is ill-conditioned.

In our numerical examples we used two variants of this *transformation approach.* The first takes V as a computed approximation of the eigenvector matrix if we can expect A to be diagonalizable and V to have small condition, e.g. when A is Hermitian. The second uses the MATLAB routine `bdschur` from the Control System Toolbox which, for a general matrix A, produces a *block diagonal* matrix D and a well conditioned matrix V, computed in floating point arithmetic, such that $A \approx VDV^{-1}$; see [4]. In either case we ue `verifylss.m` to compute an interval enclosure $\boldsymbol{W}$ for V^{-1}. Note that the matrix $\boldsymbol{D} = \boldsymbol{W}AV$ will in general have small non-zero enries outside its (block) diagonal.

5. Numerical examples. We compare the performance of the various algorithms for different classes of matrices with dimensions ranging from $n = 50$ to $n = 600$. Table 1 lists all methods together with their acronyms used in the figures to come. For the methods `SpecDec` and `PadéBM` we use the MATLAB-INTLAB implementations of Miyajima [24].

We report two quantities for all methods. The first is the *average relative precision* (arp) of $\boldsymbol{X}$ defined by

$$\operatorname{arp}(\boldsymbol{X}) := \Big(\prod_{i,j=1,n} (\operatorname{rp}(\boldsymbol{X}_{ij})) \Big)^{1/n^2}, \tag{5.1}$$

where

$$\operatorname{rp}(\boldsymbol{x}) := \min(\texttt{relerr}(\boldsymbol{x}), 1),$$

is the *relative precision* of an interval $\boldsymbol{a}$ with `relerr` defined as

$$\texttt{relerr}(\boldsymbol{x}) = \begin{cases} \frac{\operatorname{rad}(\boldsymbol{x})}{|\operatorname{mid}(\boldsymbol{x})|}, & 0 \notin \boldsymbol{x}, \\ \operatorname{rad}(\boldsymbol{x}), & 0 \in \boldsymbol{x}. \end{cases}$$

as an indicator of the quality of the computed enclosures. Roughly speaking, the quantity $-\log_{10}(\operatorname{arp}(\boldsymbol{X}))$ represents the average number of known correct digits within $\boldsymbol{X}$.

acronym	corresponding enclosure method
`TayH`	Taylor-Horner: Alg. 3.1, Horner's scheme to evaluate the polynomial [10]
`TayPS`	Taylor-Patterson-Stockmeyer: Alg. 3.1 with the modifications from (3.6)
`Padé`	Alg. 3.2
`Cont`	Contour integration: Alg. 3.3
`Cheb`	Chebyshev: Alg. 3.4
`SpecDec`	Miyajima's method relying on spectral decomposition [25]
`PadéBM`	Padé-based method of Bochev and Markov [6], $q = 7$ as implemented in [24]
`VER`	VERSOFT's routine `vermatfun.m` [34]
`Acronym-ad` is method `Acronym` using approximate diagonalization; see section 4	

TABLE 1
Acronyms used in figures.

The second reported quantity is wall clock time (in seconds) as an indicator for the efficiency of the method. Note that we undertook quite some efforts in our implementations to obtain good (interval) arithmetic performance, using built-in INTALB functions systematically and casting operations as matrix operations whenever possible.

All numerical results were obtained using INTLAB Version 11 and MATLAB R2017a on a Mac OS X with 2.5 GHz Intel Core i7 processor and 16 GB of RAM. The random number generator mode is always fixed by the command `rng(1,'twister')` for all the tests involving matrices with random entries. Most of our examples are from MATLAB's gallery of test matrices, accessible via `gallery.m`.

5.1. Nonsymmetric matrices. We compare the performance of all the eleven methods from Table 1 which are applicable to non-symmetric matrices.

EXAMPLE 1. *A is the $n \times n$ Helmert matrix, which is a permutation of a lower Hessenberg matrix, whose first row is* `ones(1:n)/sqrt(n)`*. It is in MATLAB's gallery as the* `orthog` *matrix of type 4.*

The results are depicted in Figure 3. The most narrow enclosures are obtained by `TayPS` and the second most accurate results are obtained by `Padé`. `TayPS` is not only the most accurate but also among the fastest. Methods applied to the original matrix provide more accuracy as compared to the corresponding method applied to the approximately diagonalized matrix, and the difference is the more pronounced the larger the dimension. While `SpecDec` is ranked third with respect to accuracy, it is up to almost 600 times slower than `TayPS`. This drawback of `SpecDec` and `VER` as well as, to a lesser extent, also `PadéBM` will be visible in all other experiments, too, just as the fact that `VER` typically yields the poorest enclosures. We will not repeat this observation explicitly for the other examples. Note that `SpecDec` has to use INTLAB's accurate dot product `AccDot.m` to obtain sufficiently narrow enclosures for the entries of some matrix-matrix product, an approach which is not recommended in [42] because of the interpretation overhead. For `VER`, the long run times result from its complexity being $\mathcal{O}(n^4)$ rather than, as in all the other methods, $\mathcal{O}(n^3)$. The bad timings for `PadéBM`, finally, result from the method requiring an "exponent safe bound evaluation" step that, as implemented in [24], has to enclose solutions for up to $\sqrt{n}$ linear systems with a matrix right hand side using `verifylss.m`. Finally, we note that `Cont` the contour integral approach is about one order of magnitude slower than the best performing method, and that its accuracy is substantially less than that of `TayPS`, `TayH` and `Padé`.

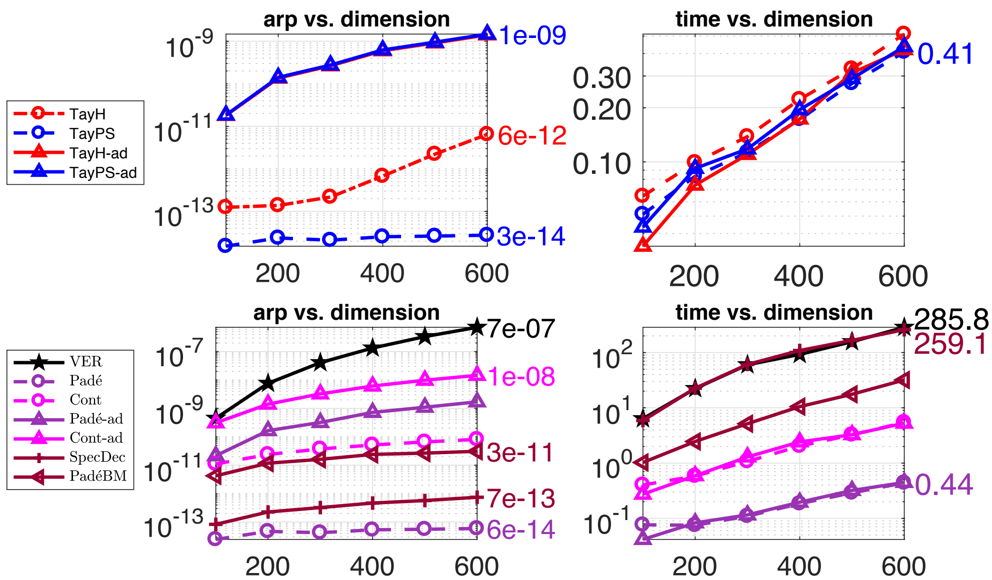

FIG. 3. Average relative precision versus dimension (left) and time versus dimension (right) for the Helmert matrix, Example 1.

EXAMPLE 2. *A is the* `forsythe` *matrix [50] from MATLAB's gallery. A consists of one single $n \times n$ Jordan block with eigenvalue zero except that its $(n, 1)$ entry is equal to $\sqrt{\varepsilon_{\text{mach}}}$.*

This time, `Padé` gives the narrowest enclosures. The second most accurate results are obtained via `TayH` and `TayPS` which perform very similarly and are not easy to distinguish in the middle of Figure 4 (left). `Padé` gives about one more digit of accuracy compared with both `TayH` and `TayPS`. Also, like Example 1, the methods applied to A generally give enclosures that are narrower than those obtained when applied to the approximately diagonalized matrix $\boldsymbol{D}$. The fastest method is `TayH`, but `TayPS` and `Padé` are comparable with respect to speed.

EXAMPLE 3. *A is the* `lesp` *matrix from MATLAB's gallery. It has the property that the condition of its eigenvalues increases exponentially with the dimension n.*

Because of the ill-conditioned eigenvalues `SpecDec` and `VER` fail, returning NaNs for any $n > 50$ and $n > 100$, respectively. Figure 5 shows that the most accurate enclosures are obtained either with `Padé` or `PadéBM` obtaining one to two more digits of accuracy compared with `TayH` and `TayPS` which are the next most accurate approaches. `Padé-ad` looses about one digit in accuracy compared to `Padé`. The computing time of `Padé`, `TayH` and `TayPS` are of the same order.

EXAMPLE 4. *We take $n \times n$ matrices A of the form $A := WDW^{-1}$, where W is a matrix with normally distributed random entries and D is the diagonal matrix with its diagonal entries taken as n equidistant points in the interval $[-1, 1]$.*

Much to the opposite of Example 3, the matrices of this example fit particularly well the `SpecDec` approach. Figure 6 shows that this is indeed the most accurate algorithm. The much faster approaches `TayPS-ad`, `Padé-ad` and `Cont-ad` which are the second most accurate, but their accuracy is significantly lower (up to 7 decimal

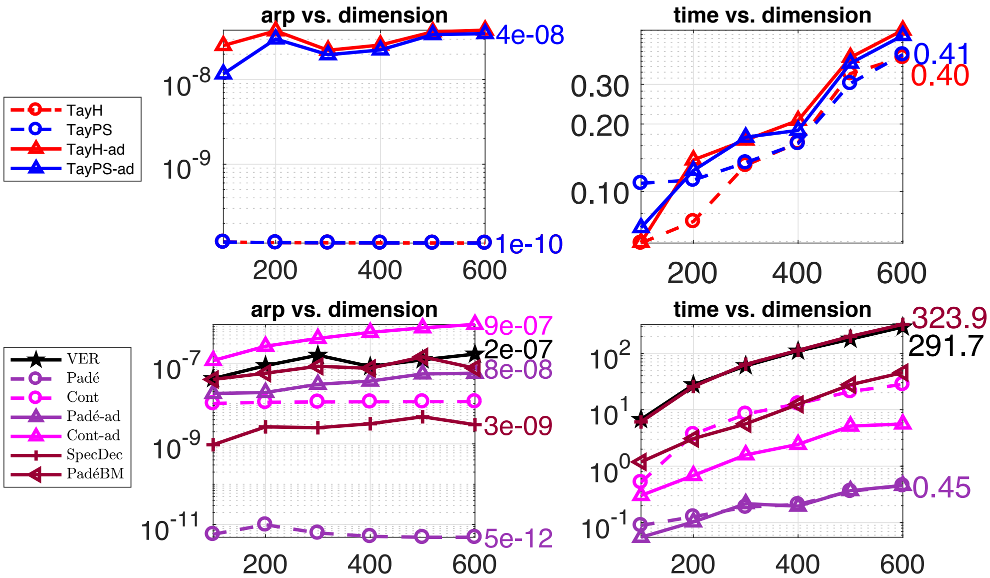

FIG. 4. Average relative precision versus dimension (left) and time versus dimension (right) for the **forsythe** matrix, Example 2.

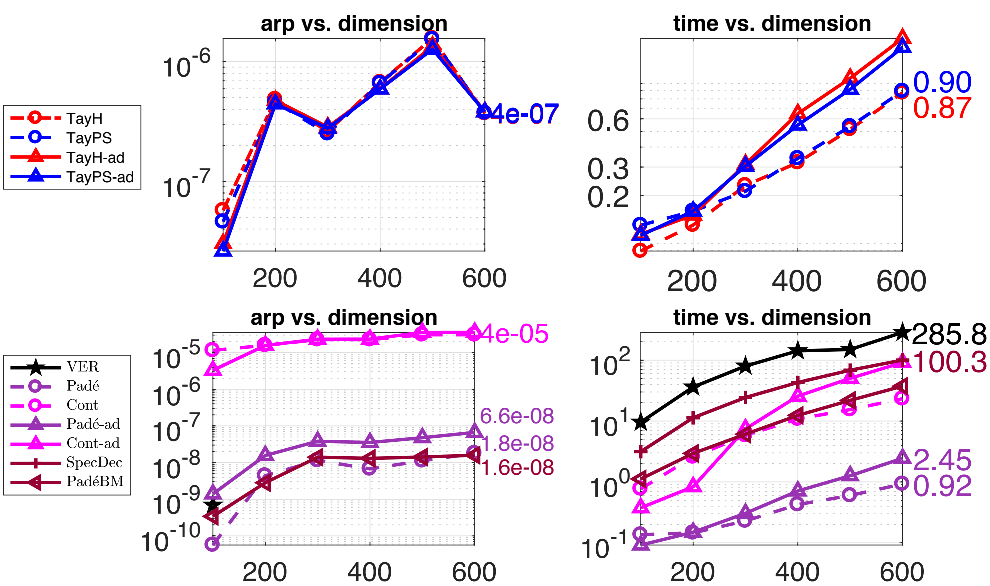

FIG. 5. Average relative precision versus dimension (left) and time versus dimension (right) for the **lesp** matrix, Example 3.

digits). Approaches without approximate diagonalization yield very poor accuracy. The reason is that W has a high condition number, so that $\|A\|_\infty$ and $\|A\|_2$ are large. This implies that the algorithms perform quite many scaling steps ($s = 14, \ldots, 16$ for $n = 400$, e.g.), whereas with approximate diagonalization this goes down to

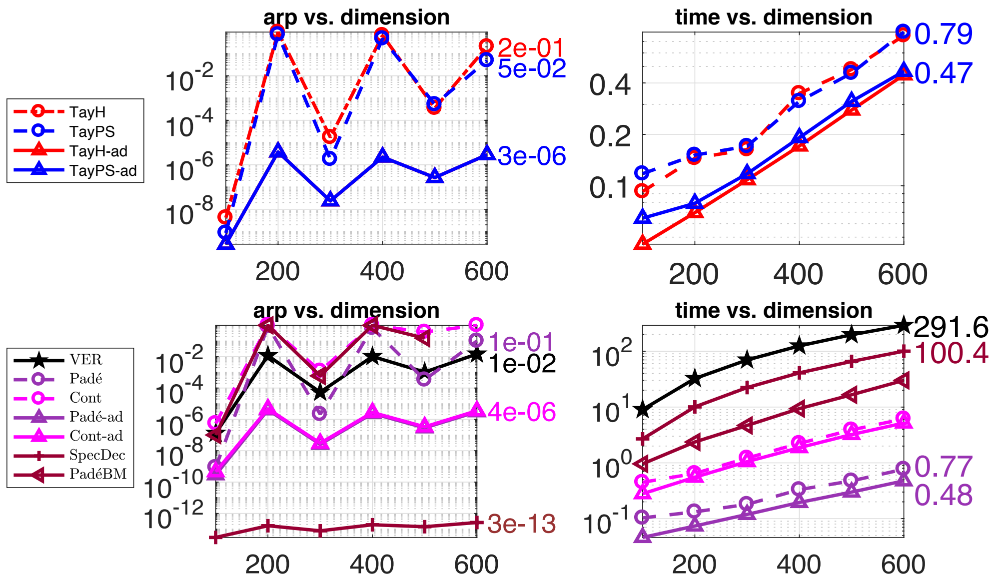

FIG. 6. Average relative precision versus dimension (left) and time versus dimension (right) for the random diagonalizable matrix whose eigenvalues are equispaced points on $[-1, 1]$. See Example 4.

$s = 1, \dots, 4$. So approximate diagonalization allows to save a significant number of squarings and thus reduces the otherwise predominant wrapping effect.

EXAMPLE 5. *A is the* `triw` *matrix from MATLAB's gallery. A is upper triangular and ill-conditioned both with respect to inversion and eigenvalue computation.*

Here, `Padé` gives the narrowest enclosures and most of the time it is the fastest method as well, see Figure 7. The quality of enclosures computed via approximate diagonalization is the same as that obtained when applied to the original matrix. `SpecDec` fails, returning NaNs, for all sizes n due to the ill-conditioning, and similarly for `VER` for $n = 100, \dots, 400$. Since `VER` already takes more than 19 minutes for $n = 400$ we did not run it for $n = 500$ and $n = 600$. `VER` is therefore not at all depicted in Figure 7, while we kept the run times for `SpecDec`.

EXAMPLE 6. *We take the point analogue of the matrices considered in [10] and define $A_n \in \mathbb{R}^{3n\times 3n}$ as the $3n \times 3n$ block diagonal matrix which for each $k = 1, \dots, n$ has one diagonal block of size 1 with entry $2k+1$ and one diagonal block of size 2 with entries $2k \cdot \left[\begin{smallmatrix} 1 & -1 \\ 1 & 1 \end{smallmatrix}\right]$ and eigenvalues $2k(1 \pm i)$. Then A is taken as $A = P^{-1}A_nP$ where P is a random orthogonal matrix, obtained as the Q-factor in the QR decomposition of a random $3n \times 3n$ matrix with normally distributed entries.*

The numerical results in Figure 8 show that the most accurate methods are `TayPS` and `Padé`, and these are the fastest methods as well. This is a quite extreme example for larger dimensions n, since the moduli of the eigenvalues of $\exp(A)$ range from e to e^{2n}.

5.2. Symmetric matrices. If A is symmetric, then so is $\exp(A)$. In our algorithms, whenever we know that a point matrix for which we compute an enclosure is symmetric we can thus "symmetrize", and at the same time narrow, this enclosure

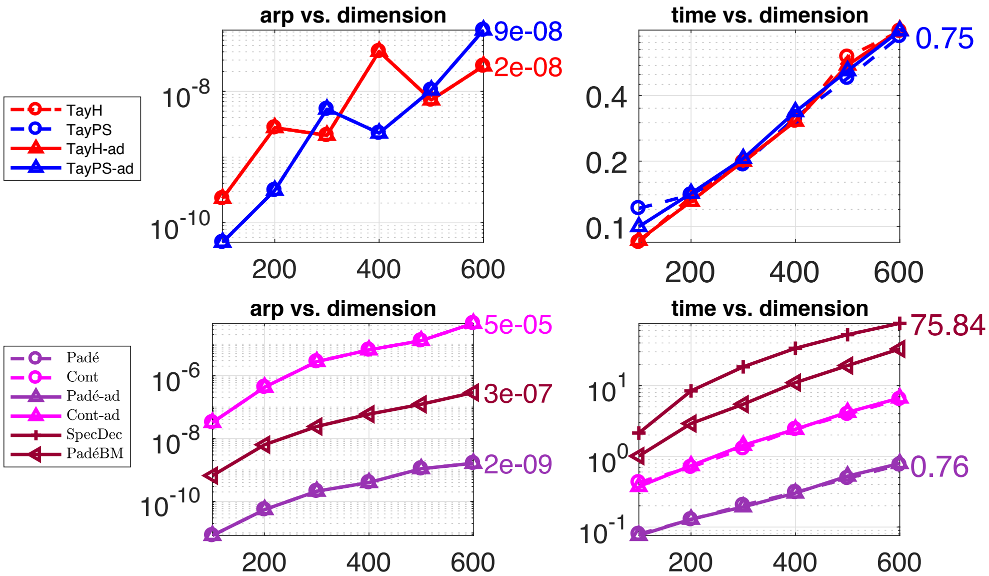

FIG. 7. Average relative precision versus dimension (left) and time versus dimension (right) for the **triw** matrices, Example 5.

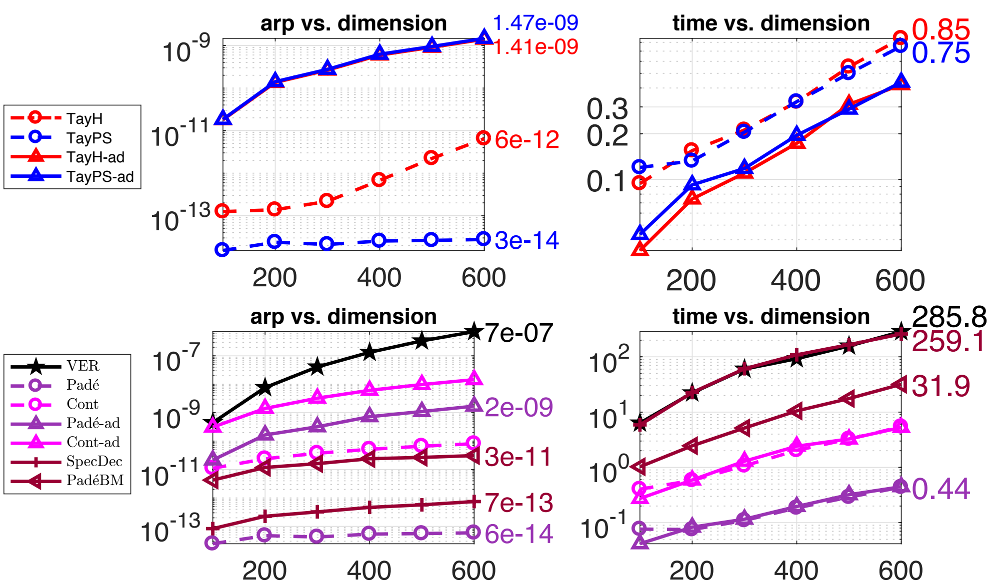

FIG. 8. Average relative precision (left) and time (right) for the (point) matrix from [10], Example 6.

by replacing it by the intersection with its adjoint. We do this whenever adequate in our implementations involving symmetric matrices.

The tested methods for symmetric matrices now also include those based on the truncated Chebyshev series.

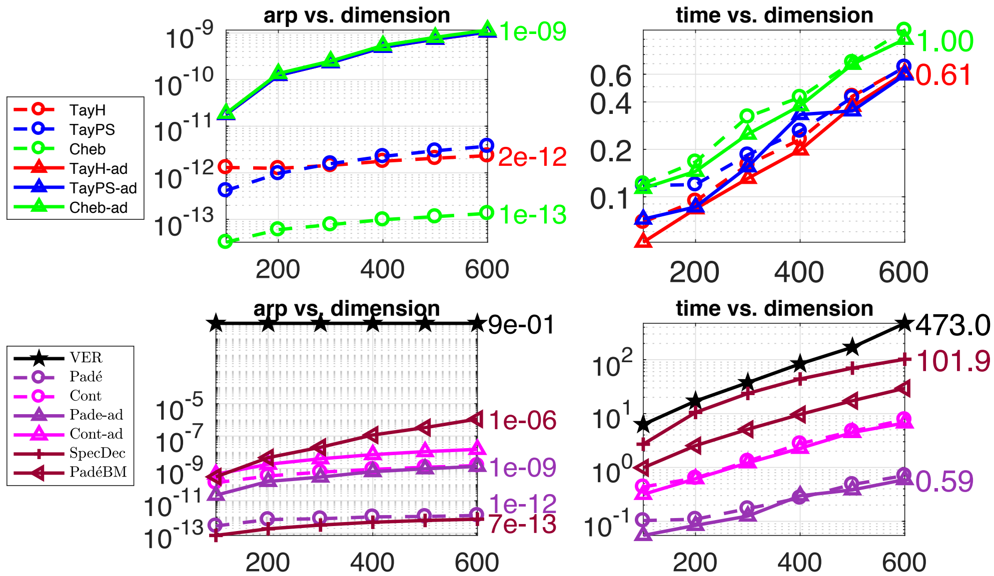

FIG. 9. Average relative precision versus dimension (left) and time versus dimension (right) for the symmetric matrix `ris`. See Example 7.

EXAMPLE 7. *A is the Hankel matrix* `ris` *from MATLAB's gallery. A is a normal matrix whose eigenvalues cluster around* $-\pi/2$ *and* $\pi/2$.

Figure 9 shows that while the narrowest enclosures are obtained by `Cheb`, the fastest method is usually `Padé` with the Taylor approaches, `Cheb` being less than a factor of 2 off. Even though A is symmetric, `VER` gives wide enclosures because, due to the clustering of the eigenvalues, the computed enclosures for the eigenvalues used in `VER` are already wide; see also Example 10. As a further illustration, for $n = 400$ we report the relative radii of all entries of the computed enclosing interval matrix for $\exp(A)$. For six different methods this is depicted in the left part of Figure 10, where the ordinate represents the $160,000$ entries in ascending order of their relative radii. The figure shows that most of the entries have similar relative width and only a few have substantially larger or smaller width. This is a quite typical situation thus justifying that "arp" is indeed a good way of measuring the quality of enclosures.

The right part of Figure 10 gives a histogram reporting a comparison of the number of "konwn correct digits" of the floating point approximation $C = \texttt{expm}(A)$ obtained using MATLAB's `expm` function and of the mid point $\operatorname{mid}\boldsymbol{C}$ of the interval matrix $\boldsymbol{C}$ obtained with `Cheb`. For an entry $\operatorname{mid}\boldsymbol{C}_{ij}$, the number of its known correct digits is the number to which the upper and lower bounds coincide. Similarly, the number of known correct digits of an entry C_{ij} is the number of digits which coincide with those of both, the lower and the upper bound of $\boldsymbol{C}_{ij}$. If an entry C_{ij} of C is not contained in $\boldsymbol{C}_{ij}$, its number of correct digit is guaranteed to be smaller than or at most equal to that of $\operatorname{mid}\boldsymbol{C}_{ij}$, and for these cases we report the difference between the known exact digits of $\operatorname{mid}\boldsymbol{C}_{ij}$ and the number of exact digits of C_{ij} in the right-most histogram. For this example, this difference is 1 or 2 for about 60% of the entries. So the results obtained with the enclosure method not only give intervals which are guaranteed to contain the exact values, but their midpoints are also (slightly) more

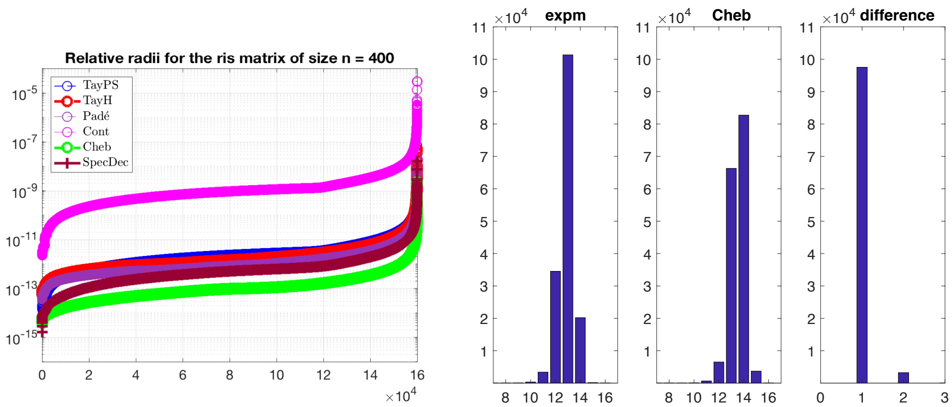

FIG. 10. Relative radii of the 160,000 entries of the computed enclosure for the symmetric matrix **ris** of size 400×400 for six different approaches (left), histogram of known correct digits in MATLAB's `expm`, midpoint of the interval matrix computed via `Cheb`, and increase in correct digits obtained by `Cheb` (right), Example 7

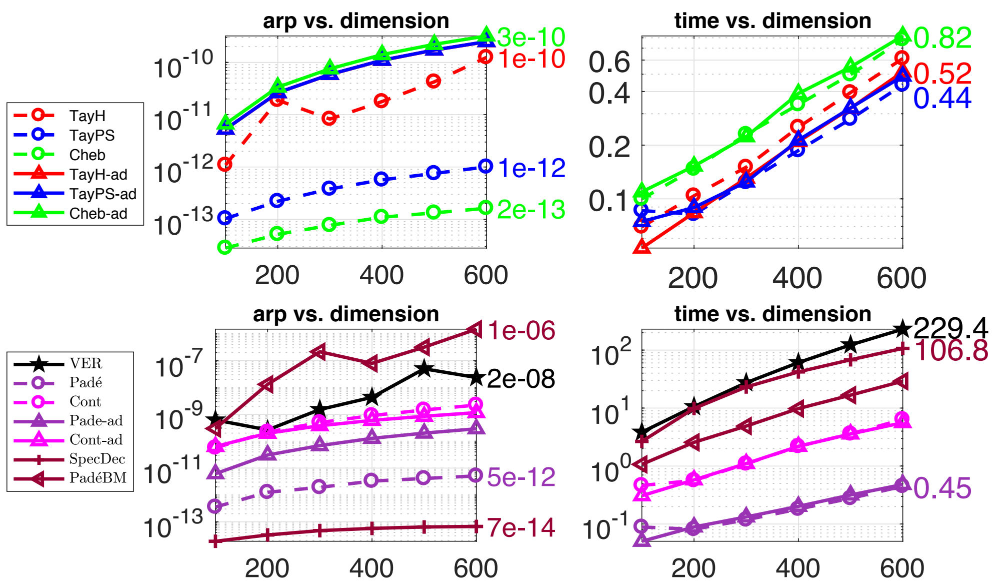

FIG. 11. Average relative precision versus dimension (left) and time versus dimension (right) for the symmetric matrix `orthog` of type two, Example 8.

accurate than the values obtained with `expm`.

EXAMPLE 8. *A is the symmetric* `orthog` *matrix of type 2 from MATLAB's gallery.* ∎

Figure 11 shows that for this example `Cheb` appears as the best method. Its accuracy is second best, only marginally lower than that of `SpecDec`, and slightly better than `Padé`. We also observe a better quality of the enclosures computed by `TayPS` compared with `TayH`.

EXAMPLE 9. *A is generated as a symmetric random matrix by filling the upper triangle of a matrix with normally distributed random entries and complementing the*

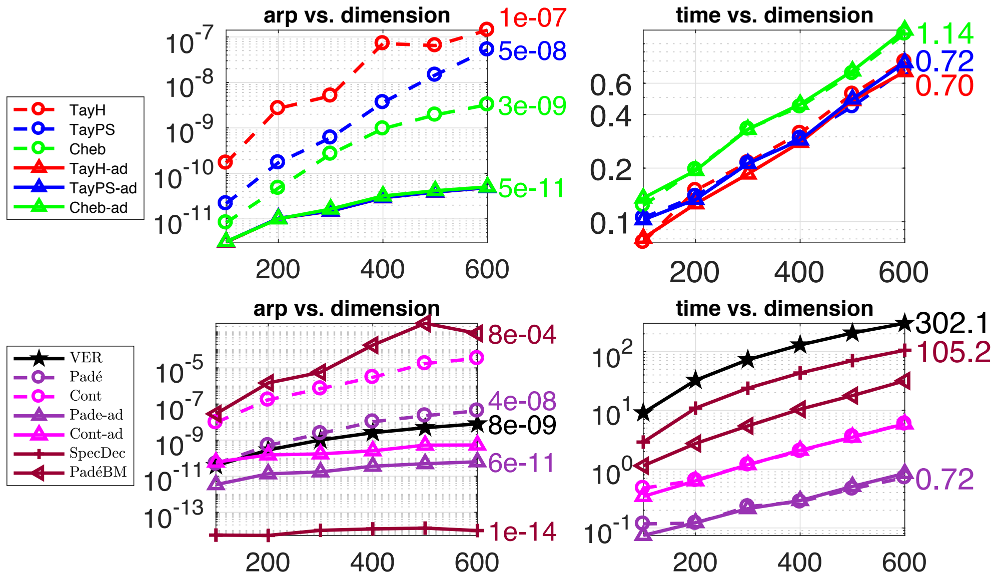

FIG. 12. Average relative precision (left) and time (right) for the symmetric random matrix, Example 9.

lower triangle symmetrically.

Figure 12 shows that among the methods with acceptable run time, those with approximate diagonalization (`Pade-ad`, `TayH-ad`, `TayPS-ad` and `Cheb-ad`) perform similarly and obtain about 2 additional digits of accuracy when compared with their counterparts without approximate diagonalization.

EXAMPLE 10. *A is the symmetric positive definite* `prolate` *matrix from MATLAB's gallery. It was also used as a test case in [25]. The matrix is Toeplitz and perfectly well conditioned with respect to the eigenvalues but ill-conditioned with respect to inversion or matrix multiplication.*

The results in Figure 13 show that this time the most accurate results are obtained either by `Cheb` or `TayPS` which show comparable speed. The eigenvalues of the `prolate` matrix tend to cluster around 0 and 1 which is why, as in Example 7, `VER` obtains poor enclosures. The cluster at 0 also explains why approximate diagonalization deteriorates the quality of the enclosures significantly: When we compute the almost diagonal matrix $\boldsymbol{D}$, the size of the off-diagonal entries is comparable to that of the eigenvalues clustering at 0. Then the computed enclosure for $\exp(\boldsymbol{D})$ will have off-diagonal entries which are not small relative to the diagonal elements, too, and this will spoil the relative accuracy when performing the two matrix-matrix multiplications in the back transformation (4.1).

EXAMPLE 11. *We take A as the symmetric and positive definite* `poisson` *matrix from MATLAB's gallery. It represents the finite difference discretization of the Laplace operator on an equispaced $N \times N$ grid with Dirichlet boundary conditions. This example is also considered in [25]. We took matrices of size $n = 100 = 10^2, 196 = 14^2, 289 = 17^2, 400 = 20^2, 484 = 22^2$ and $n = 625 = 25^2$.*

Figure 14 shows that the most accurate results are obtained with `Padé`. The

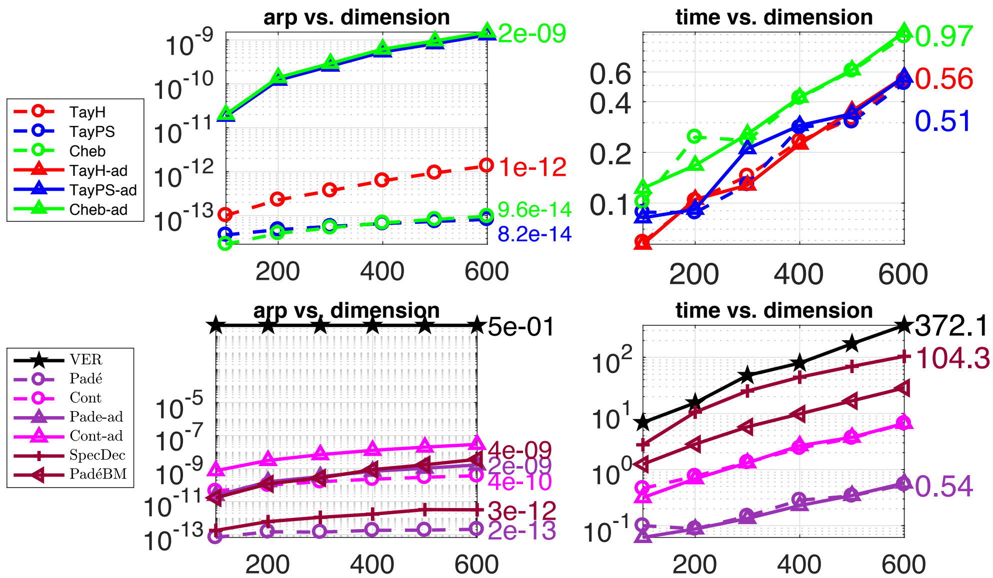

Fig. 13. Average relative precision (left) and time (right) for the `prolate` matrix, Example 10.

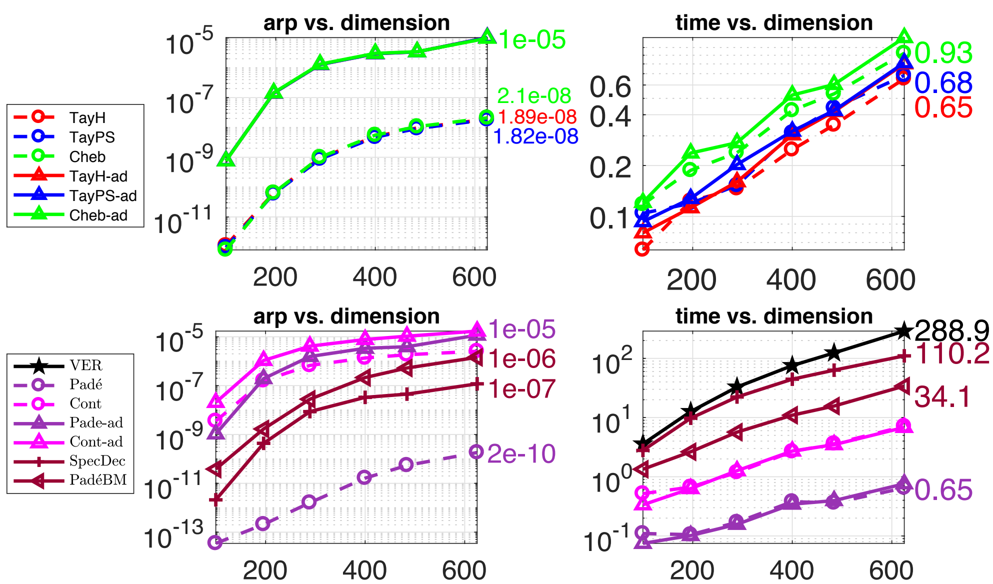

Fig. 14. Average relative precision (left) and time (right) for the `poisson` matrix, Example 11.

results with `SpecDec` are not as accurate as those for `Cheb`, `TayPS` and `TayH`. The fastest methods are Taylor-type techniques, but `Padé` is not significantly slower. `VER` returns NaNs for all dimensions.

6. Conclusions. We have presented improvements of several known and some new methods for computing enclosures for the exponential of a matrix. The methods

`TayH`, `TayPS`, `Cheb` rely exclusively on matrix-matrix multiplications so that, as a rule, the methods which require the less of those yield the tightest enclosures since they reduce the wrapping effect. The methods `Padé` and `Cont` involve a linear system solve with a (interval) matrix right hand side. For this task, state-of-the-art interval methods are available (implemented as `verifylss.m` of INTLAB, e.g.), which compute tight enclosures for the solution set.

We performed a number of numerical experiments comparing the quality of the computed enclosure and the run time of theses methods for a variety of test matrices. For general matrices, our new `Padé` is typically the best compromise in terms of accuracy and speed, closely followed by our Patterson-Stockmeyer variant `TayPS` of the Taylor approximation approach. For symmetric matrices, the new `Cheb` or `Padé` are generally superior to all other methods. The methods which rely on an either verified (`VER`) or approximate (`SpecDec`) spectral decomposition of the matrix suffer from a much higher wall clock time and do not, in general, provide tighter enclosures than `Padé` and `Cheb`. Moreover, these methods may fail completely for non-diagonalizable matrices. Method `Cont` requires the computation of enclosures for several linear systems which results in higher computational cost and less accurate overall enclosures than `Padé`. Finally, approximate diagonalization can be beneficial or detrimental to the quality of the computed enclosures, and it seems hard to characterize classes of matrices for which either of these observations would hold in general.

Acknowledgments. We are grateful to Holger Arndt, Anthony Austin, Bruno Lang, Yuji Nakatsukasa, Siegfried M. Rump, and Nick Trefethen for stimulating discussions. In particular it was Holger Arndt who ran the Arb package [17] and provided accurate enclosures for the values of the modified Bessel function of the first kind.

REFERENCES

[1] A. H. Al-Mohy and N. J. Higham, *A new scaling and squaring algorithm for the matrix exponential*, SIAM Journal on Matrix Analysis and Applications, 31 (2009), pp. 970–989.

[2] G. Alefeld and G. Heindl, *A fixed point theorem based on a modified midpoint–radius interval arithmetic*, Reliable Computing, 26 (2018), pp. 97–108.

[3] T. Auckenthaler, M. Bader, T. Huckle, A. Spörl, and K. Waldherr, *Matrix exponentials and parallel prefix computation in a quantum control problem*, Parallel Computing, 36 (2010), pp. 359–369.

[4] C. A. Bavely and G. Stewart, *An algorithm for computing reducing subspaces by block diagonalization*, SIAM Journal on Numerical Analysis, 16 (1979), pp. 359–367.

[5] P. Bochev, *Simultaneous self-verified computation of* $\exp(a)$ *and* $\int_0^1 \exp(as)ds$, Computing, 45 (1990), pp. 183–191.

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

[7] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[8] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*, Pafnuty Publications, Oxford, 2014.

[9] V. L. Druskin and L. A. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, USSR Computational Mathematics and Mathematical Physics, 29 (1989), pp. 112–121.

[10] A. Goldsztejn and A. Neumaier, *On the exponentiation of interval matrices*, Reliable Computing, 20 (2014), pp. 53–72.

[11] B. Hashemi, *Enclosing Chebyshev expansions in linear time*, ACM Tranactions on Mathematical Software, accepted, (2019), pp. 1–35, https://doi.org/10.1145/3319395, http://bit.ly/2D9brjd.

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

[13] N. J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 1179–1193.

[14] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.
[15] N. J. Higham, *The scaling and squaring method for the matrix exponential revisited*, SIAM Review, 51 (2009), pp. 747–764.
[16] W. Hofschuster and W. Krämer, *C-XSC 2.0–A C++ library for extended scientific computing*, in Numerical Software with Result Verification, R. Alt, A. Frommer, R. B. Kearfott, and W. Luther, eds., Springer, 2004, pp. 15–35.
[17] F. Johansson, *Arb: efficient arbitrary-precision midpoint-radius interval arithmetic*, IEEE Transactions on Computers, 66 (2017), pp. 1281–1292.
[18] R. Klatte, U. Kulisch, A. Wiethoff, and M. Rauch, *C-XSC: A C++ Class Library for Extended Scientific Computing*, Springer-Verlag, Berlin, 1993.
[19] O. Kosheleva, V. Kreinovich, G. Mayer, and H. T. Nguyen, *Computing the cube of an interval matrix is NP-hard*, in Proceedings of the 2005 ACM Symposium on Applied Computing, ACM, 2005, pp. 1449–1453.
[20] R. Krawczyk, *Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken*, Computing, 4 (1969), pp. 187–201.
[21] M. Liou, *A novel method of evaluating transient response*, Proceedings of the IEEE, 54 (1966), pp. 20–23.
[22] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, CRC Press, 2003.
[23] G. Mayer, *Interval Analysis and Automatic Result Verification*, de Gruyter, 2017.
[24] S. Miyajima, *MATLAB-INTLAB implementations of [25]*, 2019, http://web.cc.iwate-u.ac.jp/~miyajima/Mexp.zip.
[25] S. Miyajima, *Verified computation of the matrix exponential*, Advances in Computational Mathematics, (2019), pp. 1–16.
[26] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Review, 20 (1978), pp. 801–836.
[27] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review, 45 (2003), pp. 3–49.
[28] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009.
[29] A. Neumaier, *Enclosing clusters of zeros of polynomials*, Journal of Computational and Applied Mathematics, 156 (2003), pp. 389–401.
[30] E. P. Oppenheimer, *Application of Interval Analysis to Problems of Linear Control Systems*, PhD thesis, Electrical Engineering Department, Iowa State University, 1979.
[31] E. P. Oppenheimer and A. N. Michel, *Application of interval analysis techniques to linear systems II: The interval matrix exponential function*, IEEE Transactions on Circuits and Systems, 35 (1988), pp. 1230–1242.
[32] M. S. Paterson and L. J. Stockmeyer, *On the number of nonscalar multiplications necessary to evaluate polynomials*, SIAM Journal on Computing, 2 (1973), pp. 60–66.
[33] W. Press, S. A. Teukolsky, W. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 2nd ed., 1992.
[34] J. Rohn, *VERSOFT: Verification software in MATLAB/INTLAB*, http://uivtx.cs.cas.cz/~rohn/matlab/.
[35] S. M. Rump, *Kleine Fehlerschranken bei Matrixproblemen*, PhD thesis, Fakultät für Mathematik, Universität Karlsruhe, 1980.
[36] S. M. Rump, *Fast and parallel interval arithmetic*, BIT Numerical Mathematics, 39 (1999), pp. 534–554.
[37] S. M. Rump, *INTLAB–INTerval LABoratory*, in Developments in Reliable Computing, T. Csendes, ed., Kluwer Academic Publishers, 1999, pp. 77–104.
[38] S. M. Rump, *Ten methods to bound multiple roots of polynomials*, Journal of Computational and Applied Mathematics, 156 (2003), pp. 403–432.
[39] S. M. Rump, *Verification methods: Rigorous results using floating-point arithmetic*, Acta Numerica, 19 (2010), pp. 287–449.
[40] S. M. Rump, *Verified bounds for singular values, in particular for the spectral norm of a matrix and its inverse*, BIT Numerical Mathematics, 51 (2011), pp. 367–384.
[41] S. M. Rump, *Accurate solution of dense linear systems, Part II: Algorithms using directed rounding*, Journal of Computational and Applied Mathematics, 242 (2013), pp. 185–212.
[42] S. M. Rump, T. Ogita, and S. Oishi, *Accurate floating-point summation part I: Faithful rounding*, SIAM Journal on Scientific Computing, 31 (2008), pp. 189–224.
[43] M. Shao, W. Gao, and J. Xue, *Aggressively truncated Taylor series method for accurate computation of exponentials of essentially nonnegative matrices*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 317–338.
[44] H. J. Stetter, *Sequential defect correction for high-accuracy floating-point algorithms*, in

Numerical Analysis, D. F. Griffiths, ed., Lecture Notes in Mathematics, Springer, 1984, pp. 186–202.

[45] M. Suzuki, *Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems*, Communications in Mathematical Physics, 51 (1976), pp. 183–190.

[46] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.

[47] L. N. Trefethen, *Convergence bounds for entire functions*. Chebfun example, 2016, http://www.chebfun.org/examples/approx/EntireBound.html (accessed 2017-04-05).

[48] L. N. Trefethen and J. A. C. Weideman, *The exponentially convergent trapezoidal rule*, SIAM Review, 56 (2014), pp. 385–458.

[49] R. S. Varga, *On higher order stable implicit methods for solving parabolic partial differential equations*, Journal of Mathematics and Physics, 14 (1977), pp. 600–610.

[50] R. C. Ward, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM Journal on Numerical Analysis, 40 (1961), pp. 220–231.

[51] J. A. C. Weideman, *Numerical integration of periodic functions: A few examples*, The American Mathematical Monthly, 109 (2002), pp. 21–36.