BIT manuscript No.
(will be inserted by the editor)

Backward error analysis of polynomial approximations for computing the action of the matrix exponential

Marco Caliari · Peter Kandolf · Franco Zivcovich

the date of receipt and acceptance should be inserted later

Abstract We describe how to perform the backward error analysis for the approximation of $\exp(A)v$ by $p(s^{-1}A)^s v$, for any given polynomial $p(x)$. The result of this analysis is an optimal choice of the scaling parameter s which assures a bound on the backward error, i.e. the equivalence of the approximation with the exponential of a slightly perturbed matrix. Thanks to the SageMath package `expbea` we have developed, one can optimize the performance of the given polynomial approximation. On the other hand, we employ the package for the analysis of polynomials interpolating the exponential function at so called Leja–Hermite points. The resulting method for the action of the matrix exponential can be considered an extension of both Taylor series approximation and Leja point interpolation. We illustrate the behavior of the new approximation with several numerical examples.

Keywords backward error analysis · action of matrix exponential · Leja–Hermite interpolation · Taylor series

Mathematics Subject Classification (2000) 65D05 · 65F30 · 65F60

1 Introduction

We are interested in approximating the action of the matrix exponential to a vector, i.e. $\exp(A)v$, where $A \in \mathbb{C}^{n\times n}$ and $v \in \mathbb{C}^{n\times 1}$. When A is sparse

M. Caliari
Department of Computer Science, University of Verona, Italy
E-mail: marco.caliari@univr.it

P. Kandolf
Department of Mathematics, University of Innsbruck, Austria
E-mail: kandolfp@gmail.com

F. Zivcovich
Department of Mathematics, University of Trento, Italy
E-mail: franco.zivcovich@unitn.it

and of large scale it is not favorable to compute $\exp(A)$ and then multiply by v, since the matrix exponential is, in general, full. It is instead preferable to approximate the exponential function directly by a polynomial. Ideally this should only require matrix-vector products. This allows the computation of $\exp(A)V$, for a matrix $V \in \mathbb{C}^{n\times n_0}$ with $1 \le n_0 \ll n$, by the same method. Moreover, the process is easily parallelizable, since it does not require solving linear systems (see [19]).

The starting point of our analysis is the consideration that for every polynomial $p(x)$ of degree m such that $p(0) = 1$, there exist a positive integer s and a value $\theta_m \ge 0$ such that $p(s^{-1}A)^s v$ is sufficiently close to $\exp(A)v$ if $s^{-1}\|A\| \le \theta_m$. In section 2 we extend the backward error analysis introduced in [2] for truncated Taylor series to any given polynomial $p(x) = 1 + a_1 x + \ldots + a_m x^m$. This allows us to determine an appropriate value for s (if it exists) that guarantees *a priori* a satisfying approximation. This general result can be used for existing polynomial approximations whose parameters are currently determined only in a heuristic or a posteriori ways. Moreover, any new polynomial interpolation will benefit from the analysis carried out here.

Then in section 3 we analyse the important case of *interpolation* polynomials. In particular, we show how to compute in a stable way the divided differences needed for the polynomial interpolation in Newton form (best suited for computing the action of the matrix exponential), how to perform interpolation at complex points in real arithmetic for real input matrices, and how to sort the interpolation points in order to facilitate the triggering of an early termination criterion. In section 4 we introduce and perform the backward error analysis of newly defined approximation polynomials for $\exp(A)v$. They are based on polynomials which interpolate the exponential function at the so called Leja–Hermite points (see [5]) and extend both Taylor series approximation and Leja point interpolation [7]. In particular, they allow us to select the scaling parameter s by considering not only $\|A\|$, but also the values $\|A^q\|^{1/q} \le \|A\|$, $q \ge 1$, in order to reduce the risk of over-scaling the input matrix. The main advantage of interpolation at Leja–Hermite points with respect to existing polynomial methods is its flexibility. In fact, it can be adapted to matrices with (nearly) real or purely imaginary spectra, to matrices with different behaviors of the sequence $\{\|A^q\|^{1/q}\}_q$, and to different requirements by the user, such as best expected performance or best worst case scenario.

In section 4, moreover, we briefly describe the SageMath code `expbea` we have developed in order to perform the backward error analysis for arbitrary polynomials and tolerances. Finally in section 5 we perform several numerical experiments and compare different approximation or interpolation polynomials in the task of approximating $\exp(A)v$ and draw some conclusions in section 6.

2 Backward error analysis

We consider a polynomial

$$p(x) = a_0 + a_1 x + \ldots + a_m x^m, \quad a_0 = 1 \tag{2.1}$$

of degree m. In this section we show how to compute the value θ_m. To a first approximation, this value represents the maximal norm that a matrix A can have so that its exponential can be accurately approximated by the given polynomial. If $p(x)$ allows for $\theta_m > 0$, then any matrix A with norm larger than θ_m has to be scaled down by a positive integer s, so that $s^{-1}\|A\| \leq \theta_m$. Consequently, the approximation $v^{(s)}$ of $\exp(A)v$ is computed by the iterative scheme

$$v^{(l+1)} = p(s^{-1}A)v^{(l)}, \quad l = 0, 1, \ldots, s-1, \quad v^{(0)} = v. \tag{2.2}$$

This recursion can be carried out with only two auxiliary vectors, as it is illustrated by the SageMath code[1] reported in Algorithm 1. The main cost of such

Algorithm 1 Evaluation of scheme (2.2).

```
for l in [0 .. s - 1]:
    pA = a[0] * v
    for i in [1 .. m]:
        v = (A * v) / s
        pA = pA + a[i] * v
    v = pA
```

an algorithm is the number of performed matrix-vector products. Therefore, for given scaling parameter s and degree of approximation m, the cost is $s \cdot m$. For a specific vector v it might not be necessary to perform m iterations of the inner loop to achieve a given tolerance tol. For instance, if v is the zero vector, it is enough to perform only the initial assignment. To detect such cases an early termination criterion is employed. Common (heuristic) early termination criteria (see [2,7]) allow to break the inner loop at step $k < m$ whenever the following estimate of the norm of the tail

$$\left\| a_{k-1}(s^{-1}A)^{k-1}v^{(l)} \right\|_\infty + \left\| a_k(s^{-1}A)^k v^{(l)} \right\|_\infty$$

is smaller than

$$\mathrm{tol} \cdot \left\| \sum_{i=0}^{k} a_i(s^{-1}A)^i v^{(l)} \right\|_\infty .$$

We note that such an early termination, during the construction of the approximation polynomial, is not available if Horner's scheme is used for the evaluation of the polynomial.

In order to reduce the risk of over-scaling, the parameter s should not be chosen too large (see [1,2]). We refer to [1, § 1] for an explicit 2×2 matrix A for which choosing s too large deteriorates the approximation of $\exp(A)$. We therefore follow the backward error analysis of Al-Mohy and Higham [1]. The

[1] The excerpts of code reported here do work in SageMath. Furthermore, we feel they are short, easy to understand, and can be regarded as pseudo-codes.

main idea is to assume that the interpolation polynomial p provides the exact solution to a slightly perturbed matrix, i.e.

$$p(s^{-1}A)^s = \exp(A + \Delta A).$$

We consider ourselves satisfied whenever the relative backward error does not exceed the prescribed tolerance tol, i.e.

$$\|\Delta A\| \leq \text{tol} \cdot \|A\| . \tag{2.3}$$

In order to achieve this we represent ΔA as a function of A by exploiting the relation

$$\begin{aligned} \exp(A + \Delta A) = p(s^{-1}A)^s = \exp(A)\exp(-A)p(s^{-1}A)^s = \\ = \exp\left(A + s\log\left(\exp\left(-s^{-1}A\right)p(s^{-1}A)\right)\right) \end{aligned}$$

and by setting

$$h(s^{-1}A) := \log(\exp\left(-s^{-1}A\right)p(s^{-1}A)) = s^{-1}\Delta A.$$

Let us now investigate the relation between A and ΔA in more depth: for every matrix X in the set

$$\Omega = \{X \in \mathbb{C}^{n\times n} \colon \rho(\exp(-X)p(X) - I) < 1\},$$

where ρ denotes the spectral radius, the function $h(X)$ has the power series expansion

$$h(X) = \sum_{k=\ell+1}^{\infty} c_k X^k, \tag{2.4}$$

where $\ell \geq 0$ is the largest integer such that

$$\frac{\mathrm{d}^j p}{\mathrm{d}x^j}(0) = \mathrm{e}^0 = 1, \quad j = 0, 1, \ldots, \ell.$$

This means that the coefficients of the polynomial $p(x)$ satisfy

$$a_j = \frac{1}{j!}, \quad j = 0, 1, \ldots, \ell. \tag{2.5}$$

Hence, for those values of s such that $s^{-1}A \in \Omega$, we obtain the inequality

$$\begin{aligned} s^{-1}\|\Delta A\| = \left\|h(s^{-1}A)\right\| = \left\|\sum_{k=\ell+1}^{\infty} c_k (s^{-1}A)^k\right\| \leq \\ \leq \sum_{k=\ell+1}^{\infty} |c_k| \left\|(s^{-1}A)^k\right\| \leq \sum_{k=\ell+1}^{\infty} |c_k| \left\|s^{-1}A\right\|^k = \tilde{h}(s^{-1}\|A\|). \end{aligned} \tag{2.6}$$

Therefore, inequality (2.3) holds true if

$$\tilde{h}(s^{-1}\|A\|) \leq \text{tol} \cdot s^{-1}\|A\| .$$

In order to verify this inequality for a given polynomial, we solve for the non-negative real root of the scalar equation

$$\tilde{h}(\theta) = \sum_{k=\ell+1}^{\infty} |c_k| \, \theta^k = \mathrm{tol} \cdot \theta \tag{2.7}$$

and define the largest real solution as θ_m. The equation above has at least the trivial solution $\theta = 0$. We observe that $\tilde{h}(\theta)/\theta$ is a strictly increasing function in θ. If $\ell > 0$, then there exists one additional positive solution θ_m. If instead $\ell = 0$, this property holds provided that $|c_1| < \mathrm{tol}$. Moreover, $\tilde{h}(\theta) \leq \mathrm{tol} \cdot \theta$ for $0 \leq \theta \leq \theta_m$. Therefore, it is possible to find the integer scaling parameter $s \geq 1$ such that

$$\|\Delta A\| \leq \mathrm{tol} \cdot \|A\|, \quad \text{if } s^{-1} \|A\| \leq \theta_m. \tag{2.8}$$

A possible weakness of this approach is that the powers $\left\|s^{-1}A\right\|^k$ might be much larger than $\left\|(s^{-1}A)^k\right\|$ in inequality (2.6). This could lead to a huge over-estimate of $\left\|h(s^{-1}A)\right\|$ and consequently to an over-estimate of the scaling parameter s. In order to tackle this issue, following [1, Thm. 4.2(a)], it is possible to derive the inequality

$$\|h(X)\| \leq \sum_{k=\ell+1}^{\infty} |c_k| \, \alpha_q(X)^k = \tilde{h}(\alpha_q(X)), \quad \text{if } q(q-1) \leq \ell + 1, \quad q \geq 1$$

valid for $X \in \Omega$ and

$$\alpha_q(X) = \max\left(\|X^q\|^{\frac{1}{q}}, \left\|X^{q+1}\right\|^{\frac{1}{q+1}}\right).$$

Clearly, we have

$$\alpha_q(X) \leq \alpha_1(X) = \|X\|, \quad q \geq 1.$$

The constraint $q(q-1) \leq \ell + 1$ can be rewritten as

$$q \leq q_\ell = \left\lfloor \frac{1 + \sqrt{1 + 4(\ell+1)}}{2} \right\rfloor. \tag{2.9}$$

Therefore, in order to assure inequality (2.3) for $q \leq q_\ell$ it is enough to verify

$$\tilde{h}\left(s^{-1}\alpha_q(A)\right) \leq \mathrm{tol} \cdot s^{-1} \|A\|.$$

This is certainly true if

$$\tilde{h}\left(s^{-1}\alpha_q(A)\right) \leq \mathrm{tol} \cdot s^{-1}\alpha_q(A).$$

As result we get

$$\|\Delta A\| \leq \mathrm{tol} \cdot \|A\| \quad \text{if } s^{-1}\alpha_q(A) \leq \theta_m \text{ and } q \leq q_\ell. \tag{2.10}$$

Since $\alpha_q(A) \leq \|A\|$, the condition for s required in (2.10) is usually less restrictive than in (2.8). Hence s is less likely over-estimated. Since the evaluation

of (2.2) requires $s \cdot m$ matrix-vector products, the minimum cost for a given m corresponds to

$$\min_{1 \le q \le q_\ell} \{m \cdot \max\{\lceil \alpha_q(A)/\theta_m \rceil, 1\}\}.$$

As $\alpha_2(A) \le \alpha_1(A)$, the smallest value for q can be assumed to be 2, if $\ell \ge 1$. We note, that we did not take the cost of evaluating or approximating $\alpha_q(A)$ into account. Hence, it is not mandatory to minimize up to q_ℓ.

Remark 2.1 In order to further reduce s, it may be convenient to work with a shifted matrix $B = A - \mu I$, $\mu \in \mathbb{C}$. This is certainly true if the values $\alpha_q(B)$ are smaller then the corresponding values $\alpha_q(A)$. In [2] it was empirically found that the shift $\mu = \mathrm{trace}(A)/n$ often produces smaller values $\alpha_q(B)$.

A sufficient condition on the shift μ in order to preserve the backward error anaylysis discussed above is that $\|B\| \le \|A\|$. In fact, if the scaling parameter s is chosen in order to guarantee

$$p(s^{-1}B)^s = \exp(B + \Delta B), \quad \text{with } \|\Delta B\| \le \mathrm{tol} \cdot \|B\|,$$

then the approximation of $\exp(A)$ is recovered by

$$\begin{cases} \left(\mathrm{e}^{\mu/s} p(s^{-1}B)\right)^s, & \text{if } \Re(\mu) < 0 \\ \mathrm{e}^{\mu} \left(p(s^{-1}B)\right)^s, & \text{if } \Re(\mu) \ge 0. \end{cases}$$

The first expression has to be preferred in order to avoid $p(s^{-1}B)^s$ to overflow (see [13, sect. 10.7.3]). In exact arithmetic, both expressions coincide with

$$\mathrm{e}^{\mu} \exp(B + \Delta B) = \exp(A + \Delta B)$$

and

$$\|\Delta B\| \le \mathrm{tol} \cdot \|B\| \le \mathrm{tol} \cdot \|A\|,$$

as desired.

2.1 Power expansion of the function h

In order to find the positive root solving (2.7), we first need to compute the coefficients c_k of the power series expansion of h in (2.4). First of all, we consider the truncated polynomial of degree M of the series, for a suitable choice of $M > m$, and denote it by $h(x)|_M$. We found that $M = 3m$ is a reasonable choice. We have

$$h(x)|_M = \log(\mathrm{e}^{-x} p(x))|_M = \left(\sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j} (\mathrm{e}^{-x} p(x) - 1)^j \right) \Bigg|_M =$$

$$= \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j} (\mathrm{e}^{-x} p(x) - 1)^j \big|_M = \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j} ((\mathrm{e}^{-x} p(x))|_M - 1)^j \big|_M.$$

Since the polynomial $(\mathrm{e}^{-x}p(x))|_M - 1$ starts from degree $\ell+1$, its $\lceil M/(\ell+1)\rceil$-th power has no monomials of degree less than or equal to M. Therefore,

$$h(x)|_M = \sum_{j=1}^{\lfloor M/(\ell+1)\rfloor} \frac{(-1)^{j-1}}{j}((\mathrm{e}^{-x}p(x))|_M - 1)^j\big|_M = \sum_{k=\ell+1}^{M} c_k x^k.$$

The coefficients of the polynomial

$$(\mathrm{e}^{-x}p(x))|_M = (\mathrm{e}^{-x}|_M p(x))|_M$$

can be obtained by the convolution of the sequences $((-1)^j/j!)_{j=0}^M$ and $(a_j)_{j=0}^m$. By convolving $(\mathrm{e}^{-x}p(x))|_M - 1$ and the coefficients of the $(j-1)$-th power of $(\mathrm{e}^{-x}p(x))|_M - 1$ the coefficients of the j-th power of $(\mathrm{e}^{-x}p(x))|_M - 1$ can be computed iteratively. Finally, with the help of the leading coefficients $|c_k|$ of the power series of $\tilde{h}$ we can approximate the positive zero of $\tilde{h}(\theta)|_M - \mathrm{tol}\cdot\theta$. This allows us to obtain our bound θ_m. In the next sections we compute the bounds θ_m for various approximation polynomials up to degree $m = 55$.

2.2 Example: truncated Taylor series approximation

As example to illustrate the backward error analysis outlined above we recall the results for the truncated Taylor series presented in [2]. The polynomial $p(x)$ of degree m is

$$p(x) = \sum_{i=0}^{m} \frac{1}{i!}x^i,$$

that is the Taylor series expansion of the exponential function about 0. Since all the coefficients a_i of $p(x)$ are exactly $1/i!$, the value ℓ in (2.4) is precisely m. Therefore, we get $p(s^{-1}A)^s = \exp(A + \Delta A)$ with

$$\|\Delta A\| \le \mathrm{tol}\cdot\|A\| \quad \text{if } s^{-1}\alpha_q(A) \le \theta_m \text{ and } q \le q_m,$$

where

$$q_m = \left\lfloor \frac{1+\sqrt{1+4(m+1)}}{2} \right\rfloor.$$

In Table 2.1 we report some values of θ_m up to $m = 55$ for double precision tolerance ($\mathrm{tol} = 2^{-53}$) and the corresponding value q_m. The smallest *maximum* cost, i.e. no early termination criterion in place, is

$$\min_{\substack{1\le m\le 55\\ 1\le q\le \bar{q}\le q_m}} \{m\cdot\max\{\lceil\alpha_q(A)/\theta_m\rceil, 1\}\}, \tag{2.11}$$

where we took into consideration that only the values of $\alpha_q(A)$ for $1 \le q \le \bar{q}$ might be available, for a user defined $\bar{q}$ (because, for instance, it is considered too expensive to compute or approximate them up to q_m). We finally remark that the shift strategy described in Remark 2.1 corresponds to a Taylor expansion about the complex point μ.

Table 2.1 Values of θ_m and q_m for truncated Taylor series approximation.

m	5	10	15	20	25	30	35	40	45	50	55
θ_m	2.4e-3	1.4e-1	6.4e-1	1.4	2.4	3.5	4.7	6.0	7.2	8.5	9.9
q_m	3	3	4	5	5	6	6	6	7	7	8

3 Hermite polynomial interpolation of the exponential function

Although interpolation can be considered a special case of approximation, we reserve it a proper treatment. The above mentioned truncated Taylor series approximation can be interpreted as a special case of Hermite interpolation, where all interpolation points coincide with $x_0 = 0$ and the polynomial $p(x)$ satisfies the following conditions

$$\frac{\mathrm{d}^j p}{\mathrm{d}x^j}(0) = 1, \quad j = 0, 1, \ldots, m.$$

Moreover, the matrix exponential $\exp(A)$ itself can be represented through the polynomial that interpolates the exponential function at the eigenvalues $\{\lambda_i\}_{i=0}^{n-1}$ of A (see [13]). In Newton form it writes as

$$\exp(A) = \sum_{i=0}^{n-1} \exp[\lambda_0, \lambda_1, \ldots, \lambda_i] \prod_{j=0}^{i-1} (A - \lambda_j I), \tag{3.1}$$

where $\exp[\lambda_0, \lambda_1, \ldots, \lambda_i]$ denotes the divided difference of order i of the function exp on the set $\{\lambda_j\}_{j=0}^{i}$. Of course one is interested in a polynomial of degree m much smaller than $n - 1$, which still gives a sufficiently accurate approximation of $\exp(A)$.

In the general case, we consider a finite set of distinct complex interpolation points $P = \{x_0, x_1, \ldots, x_k\}$, with $x_0 = 0$. Then there exists a unique polynomial of degree m as in (2.1), called Hermite interpolation polynomial, satisfying

$$\frac{\mathrm{d}^j p}{\mathrm{d}x^j}(x_i) = \mathrm{e}^{x_i},$$

for any choice of the indexes $i = 0, 1, \ldots, k$ and $j = 0, 1, \ldots, m_i$ such that the degree condition

$$m = k + m_0 + m_1 + \ldots + m_k$$

holds true. Here $m_i + 1$ corresponds to the number of copies of x_i used for the interpolation. The equivalence of the leading coefficients with the inverse of the factorials (2.5) now depends on the multiplicity $m_0 + 1$ of the interpolation point $x_0 = 0$, which we therefore also denote by $\ell + 1$.

With confluent or nearly confluent points the computation of the coefficients of $p(x)$ through the Vandermonde matrix or the evaluation of the elementary Lagrange polynomials are highly unstable ways to evaluate the polynomial (see [4, § 2.]). The barycentric formula should also be discarded since, in the matrix case, it requires the solution of linear systems (see [4, end

of § 3.]). Hence we consider the Newton form, which is also an elegant way for considering information on derivatives (see, again, [4, end of § 3.]). For any permutation $(z_0, z_1, \ldots, z_m)$ of the elements of the vector

$$(\underbrace{0, \ldots, 0}_{\ell+1}, \underbrace{x_1, \ldots, x_1}_{m_1+1}, \ldots, \underbrace{x_k, \ldots, x_k}_{m_k+1})$$

the polynomial $p(x)$ can be written in terms of the Newton basis as

$$p(x) = \sum_{i=0}^{m} \underbrace{\exp[z_0, z_1, \ldots, z_i]}_{d_i} \prod_{j=0}^{i-1} (x - z_j)$$

where the coefficient d_i denotes the divided difference of order i. When the points $\{z_j\}_{j=0}^{i}$ are ordered such that equal points are adjacent (see [17]), there are simple recursive formulas for the explicit computation of the divided differences. Otherwise, the Genocchi–Hermite formula can be used (see [13, § B.16]). But there is another representation of divided differences, which turns out to be extremely important for the approximation of the matrix exponential.

Theorem 3.1 (Opitz's divided differences [20, 24]) *Let f be a function for which*

$$\frac{\mathrm{d}^j f}{\mathrm{d}x^j}(x_i), \quad i = 0, 1, \ldots, k, \quad j = 0, 1, \ldots, m_i$$

exist and let $(z_0, z_1, \ldots, z_m)$ be as above. Let $f_{i,j}$, $j \leq i$ be the divided difference of order $i - j$ for the function f on the set $\{z_j, z_{j+1}, \ldots, z_i\}$ and let finally

$$Z = \begin{bmatrix} z_0 & & & & \\ 1 & z_1 & & & \\ & 1 & z_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & z_m \end{bmatrix}.$$

Then

$$\begin{bmatrix} f_{0,0} & & & \\ f_{1,0} & f_{1,1} & & \\ \vdots & \vdots & \ddots & \\ f_{m,0} & f_{m,1} & \cdots & f_{m,m} \end{bmatrix} = f(Z).$$

In particular, the vector $(d_0, d_1, \ldots, d_m)^T$ is the first column of $\exp(Z)$. Due to the special structure of Z there is an efficient and accurate implementation of Taylor series approximation for the matrix exponential, see algorithm TS(II) in [21]. In fact, classical formulas for the computation of the divided differences suffer from severe cancellation errors and are not suitable for matrix interpolation, see [6]. Furthermore, it is not enough to approximate $\exp(Z)$ by standard Padé formulas [1] or directly $\exp(Z)e_1$ by the truncated Taylor series method [2], since these algorithms only bound the norm of the backward error, while here each coefficient d_i is required to have a relative error of

the order of machine precision. In fact, if we use the function `expm` in Matlab R2017a to compute $\exp(Z)$ for the case $z_0 = z_1 = \ldots = z_m = 0$ and extract the first column, we see in Table 3.1 that for $m = 30$ the result d_m is four orders of magnitude different from the exact value $1/30!$. Finally, the algorithm

Table 3.1 Computation of divided differences of the exponential function at the points $z_0 = z_1 = \ldots = z_m = 0$ with algorithm TS(II) and `expm` in Matlab R2017a.

m	d_m with TS(II)	d_m with `expm`
0	1.000000000000000e0	1.000000000000000e0
1	1.000000000000000e0	1.000000000000000e0
5	8.333333333333333e-3	8.333333333333337e-3
10	2.755731922398589e-7	2.755731922398613e-7
15	7.647163731819816e-13	7.647163731820674e-13
20	4.110317623312165e-19	4.112008416252581e-19
25	6.446950284384474e-26	4.001414947655047e-25
30	3.769987628815906e-33	1.372830672184864e-29

described in [21] works for any permutation of the sequence of interpolation points.

Once the divided differences are accurately computed, the matrix polynomial $p(s^{-1}A)^s v$ is computed by a simple two-term recurrence which is sketched in Algorithm 2. As in the case of polynomial approximation, we can break the

Algorithm 2 Evaluation of scheme (2.2), for $p(x)$ in Newton form.

```
for l in [0 .. s - 1]:
    pA = d[0] * v
    for i in [1 .. m]:
        v = (A * v) / s - z[i - 1] * v
        pA = pA + d[i] * v
    v = pA
```

inner loop at step $k < m$ if

$$\left\| d_{k-1} \prod_{j=0}^{k-2} (s^{-1}A - z_j I) v^{(l)} \right\|_\infty + \left\| d_k \prod_{j=0}^{k-1} (s^{-1}A - z_j I) v^{(l)} \right\|_\infty$$

is smaller than

$$\mathrm{tol} \cdot \left\| \sum_{i=0}^{k} d_i \prod_{j=0}^{i-1} (s^{-1}A - z_j I) v^{(l)} \right\|_\infty .$$

A suitable choice for the scaling parameter s is made by means of the backward error analysis framework introduced above. In order to do so, the coefficients $\{a_i\}_{i=0}^m$ of the polynomial $p(x)$ in the monomial basis have to be recovered. This is possible using Stetekluh's algorithm [27] written in Algorithm 3. We remark that the computation of the coefficients $\{a_i\}_{i=0}^m$ is only meant for the

Algorithm 3 Stetekluh's algorithm.

```
for j in [0 .. m]:
    for i in [m - 1 .. j, step = -1]:
        d[i] = d[i] - z[i - j] * d[i + 1]
a = d
```

backward error analysis which should be performed a priori and once and for all. In fact, if the early termination at step k is triggered, the monomial form

$$\sum_{i=0}^{k} a_i x^i$$

does not interpolate the exponential function at the set $\{z_0, z_1, \ldots, z_k\}$, in general, whereas the Newton form

$$\sum_{i=0}^{k} d_i \prod_{j=0}^{i-1} (x - z_j)$$

does.

3.1 Real interpolation at complex points

When A and v are real, so is $\exp(A)v$. In such a case we would like to employ real arithmetic only. This is, in general, not possible when the interpolation points are complex. On the other hand, if for each complex interpolation point z_j the conjugate $\overline{z_j}$ point is also among the interpolation points, it is possible to rewrite Newton interpolation in real arithmetic (see [28]). In order to do that it is necessary to reorder the points in such a way that every complex point is followed by its complex conjugate. If we consider the sequence

$$(z_0, z_1, \ldots, z_r, z_{r+1}, \ldots, z_m) \in \mathbb{C}^{m+1}$$

ordered in such a way that the first $r+1$ points are real, r and m with the same parity, and the next satisfy $z_{r+j+1} = \overline{z_{r+j}}$, for $j = 1, 3, 5, \ldots, m-1-r$, then the Newton interpolation can again be written as a two-term recurrence in real arithmetic, see Algorithm 4. We note that the divided differences $\{d_i\}_{i=0}^{r}$ are real, and so are $\{d_{r+2i}\}_{i=1}^{(m-r)/2}$. Moreover, from the updating of the polynomial in the second last line, it is clear that an early termination criterion similar to that described above is available. For the above sequences of points, this evaluation scheme coincides with the classical form of the Newton interpolation in exact arithmetic, even if A or v are complex. It turns out to be faster when A and v are real, since it does not use complex arithmetic. Moreover, it reduces roundoff errors in the imaginary parts when A or v are complex. Therefore, this scheme is our default choice when dealing with such interpolation points.

Algorithm 4 Evaluation of scheme (2.2), for $p(x)$ in real Newton form.

```
for l in [0 .. s - 1]:
    pA = d[0] * v
    for i in [1 .. r]:
        v = (A * v) / s - z[i - 1] * v
        pA = pA + d[i] * v
    v = (A * v) / s - z[r] * v
    w = (A * v) / s - real (z[r + 1]) * v
    pA = pA + real (d[r + 1]) * v + d[r + 2] * w
    for i in [r + 2 .. m - 2, step = 2]:
        v = (A * w) / s - real (z[i - 1]) * w + imag (z[i - 1]) ^ 2 * v
        w = (A * v) / s - real (z[i + 1]) * v
        pA = pA + real (d[i + 1]) * v + d[i + 2] * w
    v = pA
```

3.2 Ordering of the interpolation points

Even though we can accurately compute divided differences for any permutation of the interpolation points, a good order of the points in the sequence $(z_0, z_1, \ldots, z_m)$ is a crucial requirement for triggering an early termination criterion in the above scheme. In [25] Reichel suggests the following ordering, called Leja ordering, of the points of $P = \{x_0, x_1, \ldots, x_k\}$: for fixed initial point $y_0 \in P$, y_{i+1} is recursively chosen as

$$y_{i+1} \in \arg\max_{x \in P} \prod_{j=0}^{i} |x - y_j|^{m_j+1}, \quad i = 0, 1, \ldots, k-1,$$

and the sequence of interpolation points is given by

$$(z_0, z_1, \ldots, z_m) = (\underbrace{y_0, \ldots, y_0}_{m_0+1}, \underbrace{y_1, \ldots, y_1}_{m_1+1}, \ldots, \underbrace{y_k, \ldots, y_k}_{m_k+1}).$$

Since the interpolation error $\mathrm{e}^x - p(x)$ for $p(x)$ of degree i is proportional to $\pi_i(x) = (x - z_0)(x - z_1) \cdot \ldots \cdot (x - z_i)$, when z_0, z_1, …, z_i are all distinct, the procedure above selects a point z_{i+1} which maximizes $|\pi_i(x)|$ among the points in P. This should greedily reduce the interpolation error. For repeated points the above procedure diverges from this idea, that we would like to pursue in the following. Hence, fixing the first point $z_0 \in P = \{x_0, x_1, \ldots, x_k\}$, z_{i+1} is recursively picked from the set

$$z_{i+1} \in \arg\max_{x \in P} \prod_{j=0}^{i} |x - z_j| = \arg\max_{x \in P} |\pi_i(x)|, \quad i = 0, 1, \ldots, k-1.$$

Of course, when i equals k then

$$\max_{x \in P} |\pi_k(x)| = 0,$$

since all the points in P have been already selected once. In order to complete the sequence of interpolation points up to m, let us suppose that we can look for the next point in a perturbed set $P+\nu$, $\nu \in \mathbb{C}$, that is

$$z_{k+1}^{\nu} \in \arg\max_{x\in P+\nu} |\pi_k(x)| = \arg\max_{x\in P} |\pi_k(x+\nu)| + \nu.$$

We can express $\pi_k(x+\nu)$ by means of Taylor's formula

$$\pi_k(x+\nu) = \pi_k(x) + \pi_k'(x)\nu + o(\nu),$$

and therefore

$$\arg\max_{x\in P} |\pi_k(x) + \pi_k'(x)\nu + o(\nu)| + \nu = \arg\max_{x\in P} \frac{|\pi_k'(x)\nu + o(\nu)|}{\nu} + \nu.$$

Finally, letting ν tend to 0, our suggested choice for z_{k+1} is therefore

$$z_{k+1} \in \arg\max_{x\in P} |\pi_k'(x)|.$$

Now, let $P_1 \subseteq P$ be the set of points with multiplicity at least two and k_1 its cardinality. Then, we can define

$$z_{k+j} \in \arg\max_{x\in P_1} \left|\pi_{k+j-1}'(x)\right|, \quad j = 1, 2, \ldots, k_1.$$

If for $k + k_1 < m$ it is

$$\max_{x\in P_1} \left|\pi_{k+k_1}'(x)\right| = 0$$

following the reasoning above, we select

$$z_{k+k_1+j} \in \arg\max_{x\in P_2} \left|\pi_{k+k_1+j-1}''(x)\right|, \quad j = 1, 2, \ldots, k_2,$$

where $P_2 \subseteq P_1$ is the set of points with multiplicity at least three and k_2 is its cardinality. We iterate this reasoning until we complete the sequence $(z_0, z_1, \ldots, z_m)$. In case of complex conjugate pairs of points, such an ordering algorithm has to be slightly adjusted in order to keep them contiguous.

4 Examples of sets of interpolation points

In this section we consider some sets of interpolation points suited for the approximation of the matrix exponential through a polynomial interpolation. In particular, we extend some sets already used in the literature and analyse their properties. From formula (3.1) above, it is clear that it is desirable that the interpolation points $\{z_j\}_{j=0}^m$ lie in a neighbourhood of the convex hull of the spectrum $\sigma(A)$. If we shift the matrix A by $\mu = \text{trace}(A)/n$, then the eigenvalues of the matrix $B = A - \mu I$ have arithmetic average 0. Another possible shift which requires some information about the field of values of A is used in [7], with the purpose to have the field of values of B inscribed into a rectangle symmetric with respect to the origin of the complex plane.

Therefore, without loss of generality, we may assume that the interpolation points $\{z_j\}_{j=0}^m$ are distributed around the origin. Since we would like to work in real arithmetic whenever the input matrix A is real, we consider only sets of real interpolation points or sets with real and complex conjugate pairs of interpolation points.

We start with interpolation at real Leja points, already used and analysed in [7]. Because of the shift strategy described above, it is not restrictive to consider a real interval $[-c, c]$ symmetric with respect to the origin. A sequence of Leja points for the real interval $[-c, c]$ is defined by $z_0 = 0$ and

$$z_{i+1} \in \arg \max_{x\in[-c,c]} \prod_{j=0}^{i} |x - z_j|, \quad i = 0, 1, \ldots, m-1.$$

We select the first three points as $z_1 = c$, $z_2 = -c$, and $z_3 = c/\sqrt{3}$. The next points are univocally determined. The related unique polynomial $p(x)$ of degree m satisfying

$$p(z_i) = \mathrm{e}^{z_i}, \quad i = 0, 1, \ldots, m$$

is called Leja interpolation polynomial. Differently from the truncated Taylor series method, here there is the interpolation interval as a free parameter to be chosen. Therefore, for given m and c it is possible to compute, by means of the backward error analysis, the corresponding value $\theta_{m,c}$. For each m, the curve $c \mapsto \theta_{m,c}$ has been drawn for m up to 100 in [7]. For some values of m up to 55, we report in Table 4.1, the value

$$\theta_m = \max_{0\le c\le \bar{c}_m} \theta_{m,c}, \quad \bar{c}_m = \min\{\gamma \colon \gamma = \theta_{m,\gamma}\}, \tag{4.1}$$

computed by taking the maximum of $\theta_{m,c}$ over 200 values of c between 0 and $\bar{c}_m$[2]. It is an approximation of the maximum of the curve $c \mapsto \theta_{m,c}$ before its first intersection with the curve $c \mapsto c$. Additionally, we obtain the interpolation interval $[-c_m, c_m]$ corresponding to the maximum θ_m. Since c is

Table 4.1 Values of c_m, θ_m, ℓ_m, and q_{ℓ_m} for the Leja interpolation.

m	5	10	15	20	25	30	35	40	45	50	55
c_m	0.0	0.0	0.0	0.0	7.4e-1	1.4	2.1	2.5	3.1	4.2	4.8
θ_m	2.4e-3	1.4e-1	6.4e-1	1.4	2.5	3.6	4.8	6.1	7.4	8.8	1.0e1
ℓ_m	5	10	15	20	0	0	0	0	0	0	0
q_{ℓ_m}	3	3	4	5	1	1	1	1	1	1	1

allowed to range from 0 (in which case Leja interpolation does coincide with the truncated Taylor approximation) to $\bar{c}_m$, for each degree m the value found for θ_m is never smaller than the corresponding value for the truncated Taylor approximation. It turns out that for degree m up to 16 and degrees 19 and

[2] In [7], θ_m was selected equal to $\bar{c}_m$ and therefore the values reported in Table 4.1 are larger than those in [7, Table 1].

20, the maximum in (4.1) is in fact attained at $c_m = 0$. Therefore, Leja interpolation does coincide with Taylor approximation and it is in fact a Hermite interpolation. For these degrees m the interpolation points are $\ell_m + 1 = m + 1$ copies of 0 and the value

$$q_{\ell_m} = \left\lfloor \frac{1 + \sqrt{1 + 4(\ell_m + 1)}}{2} \right\rfloor$$

is greater than 1. For the degrees larger than 20, the value θ_m is larger than the corresponding value in the Taylor method. For instance, for degree $m = 50$, it is about 8.78 versus 8.55. Therefore, if $p(x)$ is the polynomial of degree m which interpolates the exponential function at the Leja points of $[-c_m, c_m]$, then $p(s^{-1}B)^s = \exp(B + \Delta B)$ with

$$\|\Delta B\| \leq \mathrm{tol} \cdot \|B\| \quad \text{if } s^{-1}\alpha_q(B) \leq \theta_m \text{ and } q \leq q_{\ell_m}.$$

The Leja points in the intervals $[-c_m, c_m]$ (for any degree m) as well as the corresponding divided differences can be computed once and for all and stored for an actual implementation of the method. Therefore, the method has no extra cost with respect to a standard two-terms recurrence algorithm, such as Taylor approximation.

Despite the slightly improved boundaries θ_m presented by this method, there exists a drawback given by the fact that the interpolation point $z_0 = 0$ has in general multiplicity 1 (i.e. $\ell = 0$). Therefore, it is not possible to employ the values of $\alpha_q(B)$ for q different from 1. Nevertheless, this interpolation method can be effective whenever the matrix B has a thin spectrum distributed along the real axis. This property has been already exploited in [7].

In order to combine all the interesting properties of the truncated Taylor series and the Leja interpolation method it appears logical to literally mix them. This is done in our second example, the so called Leja–Hermite points. In fact, we consider sets of points P obtained as Leja extension of a core set composed by $\ell + 1$ points at the origin (see [5]). Namely, we take the sequence of points $(z_0, z_1, \ldots, z_\ell) = (0, 0, \ldots, 0)$ and choose

$$z_{i+1} \in \arg \max_{x \in [-c,c]} \prod_{j=0}^{i} |x - z_j|, \quad i = \ell, \ell + 1, \ldots, m - 1.$$

We note that it is possible to extend any set of $\ell + 1$ given points in a similar fashion. For the case of interest, we select $z_{\ell+1} = c$, $z_{\ell+2} = -c$, and $z_{\ell+3} = c\sqrt{(\ell + 1)/(\ell + 3)}$. Consequently, the polynomial interpolating the exponential function at such a sequence of Leja–Hermite points satisfies

$$\begin{cases} \dfrac{\mathrm{d}^j p}{\mathrm{d}x^j}(0) = 1 & j = 0, 1, \ldots, \ell, \\ p(z_i) = \mathrm{e}^{z_i} & i = \ell + 1, \ell + 2, \ldots, m. \end{cases}$$

That gives us the possibility to employ the values $\alpha_q(B)$ for some suitable $q \geq 1$ together with new boundaries θ_m. Now, for each degree of interpolation

m, there are two free parameters, namely the multiplicity $\ell+1$ of 0, $\ell \le m$ and the interpolation interval $[-c, c]$. Therefore, it is possible to compute the boundaries $\theta_{m,\ell,c}$ for each ℓ and each c. A possible choice for the determination of a unique interpolation polynomial of a given degree m is the following. We first consider

$$q_m = \left\lfloor \frac{1+\sqrt{1+4(m+1)}}{2} \right\rfloor$$

(see formula (2.9)) and select θ_m as

$$\theta_m = \max_{0 \le c \le \bar{c}_{m,\ell}} \theta_{m,\ell,c}, \quad \bar{c}_{m,\ell} = \min\{\gamma \colon \gamma = \theta_{m,\ell,\gamma}\}, \tag{4.2}$$

requiring

$$\ell + 1 = q_m(q_m - 1).$$

This choice allows us to use the values $\alpha_q(B)$, while keeping $m-\ell$ points distributed in the interval $[-c_m, c_m]$. In our experience this is beneficial if an early termination criterion is used. Moreover, we obtain values for θ_m not smaller than the corresponding value for the truncated Taylor series. For instance, for degree $m = 50$, the corresponding value θ_m is about 8.64. We note that in some circumstances the above maximum is attained for $c_m = 0$. In these cases, the number of zeros among the interpolation points is exactly $\ell_m + 1 = m + 1 \ge q_m(q_m - 1)$. In Table 4.2 we report some values c_m, θ_m, and q_m for degree m up to 55.

Another possible choice in order to associate a unique interpolation polynomial to a given degree m is the one which maximizes the value θ_m, namely

$$\theta_m = \max_{\substack{0 \le \ell \le m \\ 0 \le c \le \bar{c}_{m,\ell}}} \theta_{m,\ell,c}. \tag{4.3}$$

For instance, for degree $m = 50$ the value θ_m is about 8.84 obtained with an interpolation set with $\ell_m = 37$. In Table 4.3 we report some values c_m, θ_m, and q_m for degree m up to 55. This choice is particularly appealing when the early termination is not necessary and the information about the values $\alpha_q(B)$ for $q \ge 2$ is not available. In this case, in formula (2.10) q_ℓ has to be replaced by q_1. These features can be useful for some exponential integrators (see [16, 18]), which may require the evaluation in parallel of large matrix exponentials.

Table 4.2 Values of c_m, θ_m, ℓ_m, and q_{ℓ_m} for the Leja–Hermite interpolation, with θ_m defined as in (4.2).

m	5	10	15	20	25	30	35	40	45	50	55
c_m	0.0	0.0	0.0	0.0	9.2e-1	0.0	3.0	4.0	6.1	6.3	0.0
θ_m	2.4e-3	1.4e-1	6.4e-1	1.4	2.4	3.5	4.8	6.1	7.4	8.6	9.9
ℓ_m	5	10	15	20	19	30	29	29	41	41	55
q_{ℓ_m}	3	3	4	5	5	6	6	6	7	7	8

Table 4.3 Values of c_m, θ_m, ℓ_m, and q_{ℓ_m} for the Leja–Hermite interpolation, with θ_m defined as in (4.3).

m	5	10	15	20	25	30	35	40	45	50	55
c_m	0.0	0.0	0.0	7.5e-1	1.4	2.3	3.5	3.7	4.8	5.9	5.0
θ_m	2.4e-3	1.4e-1	6.4e-1	1.5	2.5	3.6	4.9	6.1	7.5	8.8	1.0e1
ℓ_m	5	10	15	15	20	25	30	27	32	37	2
q_{ℓ_m}	3	3	4	4	5	5	6	5	6	6	2

As a final example, we analyse an extension of the sequence of complex conjugate Leja points introduced in [5,7,8] with a single initial point $z_0 = 0$. Given the initial sequence $(z_0, z_1, \dots, z_\ell) = (0, 0, \dots, 0)$ we define them in the complex interval $\mathrm{i}[-c, c]$ as

$$z_{i+1} \in \arg \max_{x \in \mathrm{i}[-c,c]} \prod_{j=0}^{i} |x - z_i|, \quad z_{i+2} = \overline{z_{i+1}}, \quad \text{for } i = \ell, \ell+2, \dots, m-1.$$

The first four points after the initial sequence are $z_{\ell+1} = \mathrm{i}c$, $z_{\ell+2} = -\mathrm{i}c$, $z_{\ell+3} = \mathrm{i}c\sqrt{(\ell+1)/(\ell+3)}$, and $z_{\ell+4} = -\mathrm{i}c\sqrt{(\ell+1)/(\ell+3)}$. This set of interpolation points is beneficial for those matrices B with eigenvalues whose convex hull is well described by a vertical skinny rectangle. In order to use a set containing an even number of points, the value ℓ has to be taken odd. Therefore, in order to allow the use of the largest number of values $\alpha_q(B)$, we select the number of repeated initial zeros as

$$\ell_m + 1 = \begin{cases} q_m(q_m - 1) & \text{if } m \text{ is odd} \\ q_m(q_m - 1) + 1 & \text{otherwise.} \end{cases} \tag{4.4}$$

In [7] it was observed that the maximum of the curve $c \mapsto \theta_{m,0,c}$ (which corresponds to $\ell = 0$) is attained at $c = 0$. This fact now turns out to be true for any number of repeated initial zeros in the sequence. We moreover observed the same phenomenon with pure complex Leja–Hermite points, without any artificial conjugation. Therefore, a maximization like (4.2) produces nothing else than the truncated Taylor series of degree m. However, as previously observed, the presence of some point in the interpolation interval generally helps to trigger the early termination criterion. Therefore, unless ℓ_m turns out to be equal to m from formula (4.4) and all the interpolation points concide with zero, we follow the path traced in [7] by taking

$$\theta_m = \bar{c}_{m,\ell_m}, \quad \bar{c}_{m,\ell_m} = \min\{\gamma \colon \gamma = \theta_{m,\ell_m,\gamma}\}. \tag{4.5}$$

The values of ℓ_m, θ_m (obtained by a fixed point iteration of the equation $\gamma = \theta_{m,\ell_m,\gamma}$), and q_{ℓ_m} are reported in Table 4.4. For $m = 50$ we get θ_m about 8.17.

For comparison, we also computed the values θ_m, which correspond to sets of complex conjugate Leja points with the smallest number of repeated zeros $\ell_m + 1$ (one or two), see Table 4.5.

Table 4.4 Values of θ_m, ℓ_m, and q_{ℓ_m} for the Leja–Hermite interpolation at complex conjugate points, with θ_m defined as in (4.5). In this table $c_m = \theta_m$, unless $\ell_m = m$, in which case $c_m = 0$.

m	5	10	15	20	25	30	35	40	45	50	55
θ_m	2.4e-3	1.3e-1	5.9e-1	1.4	2.3	3.5	4.5	5.6	7.1	8.2	9.9
ℓ_m	5	6	11	20	19	30	29	30	41	42	55
q_{ℓ_m}	3	3	4	5	5	6	6	6	7	7	8

Table 4.5 Values of θ_m, ℓ_m, and q_{ℓ_m} for Leja–Hermite interpolation at complex conjugate points, with the smallest number of repeated zeros. In this table $c_m = \theta_m$.

m	5	10	15	20	25	30	35	40	45	50	55
θ_m	1.9e-3	1.2e-1	5.2e-1	1.2	2.0	3.0	4.0	5.1	6.1	7.3	8.4
ℓ_m	1	0	1	0	1	0	1	0	1	0	1
q_{ℓ_m}	2	1	2	1	2	1	2	1	2	1	2

4.1 SageMath code for the backward error analysis

All the previous tables were obtained with the help of the SageMath code mentioned above and described in this section. The code is a very flexible and powerful tool for the analysis of interpolation polynomials for the exponential function. In the following we will highlight some of its main features as well as some possible applications. We will mainly do this by showing how the θ_m values in the above tables can be computed with the help of the code.

The package is essentially comprised of two functions, namely `expbea.sage`, which performs the backward error analysis for a polynomial and `intpol.sage` which computes the coefficients of the polynomial which interpolates the exponential at certain given points. Moreover, we prepared a list containing Leja–Hermite points (`lejaptslist.sage`) up to degree 109 with any number of leading zeros and the corresponding symmetric version (`lejasymptslist.sage`) suitable for interpolation at complex points. Finally, the function `leja.sage` has to be invoked in order to select the proper set of points and scale them to the wanted interval.

In order to get started with the package the above mentioned functions need to loaded into SageMath by:

```
load ("expbea.sage")
load ("intpol.sage")
load ("lejaptslist.sage")
load ("lejasymptslist.sage")
load ("leja.sage")
```

Now the function

```
expbea (poly, tol, digits)
```

is available. The three arguments are the polynomial (2.1), the desired tolerance tol in (2.7), and the binary digits used for evaluation. The result, if it exists, is the largest real solution θ_m fulfilling (2.7). As a possible application, not yet mentioned, suppose you have a problem where you are interested in

obtaining a result for a certain tolerance, e.g. quadruple or specifically 2^{-10}. Currently the Taylor as well as Leja method do support the precisions half, single, and double. In addition to the early termination you can obtain the best performance for your example by using the function `expbea` to compute the required parameters θ_m tailored to your tolerance. These values can easily be computed and used for your problem.

To illustrate the functionality of `expbea`, as well as the other methods, we recompute the value θ_{50} for Taylor, Leja, Leja–Hermite, and complex conjugate Leja–Hermite interpolation in several different ways. Furthermore, we also show how parameters for other popular approximation methods can be computed with our code.

For the truncated Taylor method the value θ_{50} (compare with Table 2.1) can be computed by

```
expbea (intpol (leja (51, 50, 0, lejapts), 165), 2 ^ (-53), 165)
```

resulting in

```
8.546902045684933253595836581620611939411332215655
```

The function `leja` selects the set of 51 Leja–Hermite points with $\ell = 50$ from the list `lejapts` (therefore, a set containing 51 repetitions of zero). The coefficients of the interpolation polynomials and the backward error analysis were performed with 165 binary digits. The tolerance 2^{-53} corresponds to double precision. The computational time for the above computation (obtained by the `timeit` functions) is 351 ms on a standard laptop. For such an example, a simpler way to reach the same result is to directly give the coefficient of Taylor series expansion for the exponential function, i.e.

```
expbea ([1 / factorial (i) for i in [0 .. 50]], 2 ^ (-53), 165)
```

with has a computational time of 42.9 ms and provides the same result as before. Now suppose we are interested in quadruple precision, invoking

```
expbea ([1 / factorial (i) for i in [0 .. 50]], 2 ^ (-113), 165)
```

computes the value θ_{50} for $\mathrm{tol} = 2^{-113}$, i.e.

```
4.063015975075497005259133550997831602060074466426
```

with a computational time of 69.8 ms.

Now let us extend this example to the Leja interpolation. For this method we need to include a interpolation interval and obtain the necessary polynomial. Again we recompute the value θ_{50} found in Table 4.1 and therefore the interpolation interval $[-4.2, 4.2]$ is used. By calling

```
expbea (intpol (leja (51, 0, 42 / 10, lejapts),165), 2 ^ (-53), 165)
```

we compute the Leja polynomial of degree 50 with $\ell = 0$ in the interval $[-4.2, 4.2]$ as input for `expbea` and obtain

```
8.773372324142648390974599300196828845892837073024
```

with a computational time of 1.97 s. Again we used 165 binary digits and the unite roundoff for double. To obtain the same coefficient θ_{50} for the Leja–Hermite interpolation in the interval $[-6.3, 6.3]$ (compare with Table 4.2), we invoke

```
expbea (intpol (leja (51, 41, 63 / 10, lejapts), 165), 2 ^ (-53), 165)
```

and obtain

```
8.642710070503132351899676020863358052966697141513
```

Finally, the value θ_{50} for the Leja–Hermite interpolation at complex conjugate points in the interval $\mathrm{i}\cdot[-8.2, 8.2]$ (compare with Table 4.4) is computed by,

```
expbea (intpol (leja (51, 42, I * 82 / 10, lejasympts), 165), 2 ^ (-53), 165)
```

and results in

```
8.172837810334057223553959774976711911401660234870.
```

Other direct polynomial methods are used in the literature for the matrix exponential approximation, see, for instance, [3,11,22,23,26]. We briefly look at Chebyshev approximation in the interval $[-c, c]$ and show how `expbea` can be used here. The truncated Chebyshev series approximation can be written as

$$\mathrm{e}^x = I_0(c) + 2\sum_{i=1}^{\infty} I_i(c)T_i(x/c)$$

where I_i is the modified Bessel function of the first kind of order i and T_i the Chebyshev polynomial of the fist kind of degree i. The truncated polynomial of degree m has the representation

$$p(x) = I_0(c) + 2\sum_{i=1}^{m} I_i(c)T_i(x/c)$$

and does not fulfill the property $p(0) = 1$. Therefore, the above analysis cannot directly be applied. Nevertheless, it is possible to consider the polynomial

$$\tilde{p}(x) = p(x) - p(0) + 1$$

which trivially fulfills the required property. In SageMath it is possible to perform the backward error analysis in the following way:

```
c = 42 / 10
m = 50
p = bessel_I (0, c) * chebyshev_T (0, x / c)+\
2 * sum ([bessel_I (i, c) * chebyshev_T (i, x / c) for i in [1 .. m]])
ptilde = p - p(x = 0) + 1
expbea (p.list(), 2 ^ (-53), 165)
```

The result

```
8.777192038645274619076146787259326379345848575125
```

is very similar to that of the Leja interpolation in the same interval.

With the framework compiled in `expbea.sage` it is also possible to compute the value θ_m for other approximations, such as the rational Padé approximation. For instance, starting from

```
r3 = (-x ^ 3 - 12 * x ^ 2 - 60 * x - 120) / (x ^ 3 - 12 * x ^ 2 + 60 * x - 120)
```

which is the Padé approximant of order [3/3] to the exponential, we get

```
expbea (r3.taylor (x, 0, 14).list(), 2 ^ (-53), 165)
```

0.014955852179582915172769802710889816287320053018 97

which corresponds up to 16 digits with the corresponding value used in the Matlab R2017a function `expm`. The relation between the order of the Padé approximant and the order of the Talyor expansion used for the computation (14 in this example) is not further investigated here.

This little demonstration outlines the main features of or SageMath code. The whole code for the backward error analysis is available at the web page `https://bitbucket.org/expleja/expbea`.

5 Numerical experiments

In this section we report on several numerical experiments. For all experiments we use Matlab R2017a. Furthermore, we denote the truncated Taylor series method by "T", the interpolation at Leja–Hermite points by "L–H", and the Leja–Hermite method with the reordering of the points as suggested at the end of section 3.2 by "L–H reord.". If complex conjugate Leja–Hermite points are used we indicate this by adding "cplx", in this case choice (4.5) is used. Furthermore, A denotes the original matrix and $B = A - \mathrm{trace}(A)/n$. For each of the following experiments, we report the scaling parameter s, the degree m, the interval endpoint c_m, the value θ_m, and the number ℓ_m of repetitions of zero among interpolation points. Here m corresponds to the maximum number of iterations for each scaling parameter, selected by the optimization strategy (2.11). Moreover, the total maximum number of iterations (which corresponds to $s \cdot m$), the actual number of iterations (if the early termination criterion is enabled), and the relative error in the 1-norm, with respect to the solution computed by `expm(A)*v`, are reported. The global accuracy of `expm(A)*v` is not known. On the other hand, for Hermitian, skew-Hermitian, or real and essentially nonnegative matrices the Taylor method approximates the action of the matrix exponential in a forward stable way [12] and the amplification of rounding errors is governed by the condition number of the $\exp(A)v$ problem (which can be estimated as described in [14] or in [10], for large and sparse matrices). Not all of the matrices used in our experiments have the above properties. Nevertheless, we decided to report in parenthesis the error of Leja–Hermite interpolation with respect to the Taylor approximation. We also note that the computational cost of evaluating the sequence $\{\alpha_q(B)\}_{q=1}^{\bar{q}}$ through the estimation algorithm described in [15], which is $4\bar{q}(\bar{q}+3)$ matrix-vector products (see [2]), is not reported in the tables below. In all experiments tol is set to 2^{-53} (which corresponds to the double unit roundoff) if required.

We group the experiments, depending on the behavior of the sequence $\{\alpha_q(B)\}_q$.

5.1 Matrices with non-decreasing α_q values

In the following we are going to present several experiments where the sequence $\{\alpha_q(B)\}_q$ is constant. On the one hand, it means that it is not necessary to

waste matrix-vector products to estimate the sequence. On the other hand, applying [1, Thm. 4.2(a)] to the estimation of s provides no additional savings.

5.1.1 Advection-diffusion matrices

We consider the method of lines applied to the partial differential equation

$$\begin{cases} \dfrac{\partial u}{\partial t}(t,x,y) + b\left(\dfrac{\partial u}{\partial x}(t,x,y) + \dfrac{\partial u}{\partial y}(t,x,y)\right) = d\left(\dfrac{\partial^2 u}{\partial x^2}(t,x,y) + \dfrac{\partial^2 u}{\partial y^2}(t,x,y)\right), \\ u(0,x,y) = u_0(x,y) = 16x(1-x)y(1-y), \end{cases}$$

equipped with homogeneous Dirichlet boundary conditions in the spatial domain $[0,1]^2$. The parameters are chosen as $d = 1/100$, b non-negative, and we discretise in space by standard central second-order finite differences. The space step size h is chosen equal to $1/50$. The grid Péclet number is thus

$$\mathrm{Pe} = \frac{hb}{2d} = b.$$

The resulting matrix A has size $n \times n = 2401 \times 2401$ (we used a discretization with inner nodes). We are interested in the solution at time $t = 1$, where v is the space discretization of u_0. After applying the shift $\mu = \mathrm{trace}(A)/n$, the equality $\|B\|_1 = \alpha_q(B) = 4d/h^2 = 100$ holds for any value of $q \leq 8$ and non-negative b. For $b = 0$ the matrix is symmetric and all the eigenvalues are real and distributed in the interval $[-100, 100]$. The results for this experiment are collected in Table 5.1. For such an example, interpolation at pure Leja points, without repeated zeros, outperforms Taylor method, since the interpolation points are well spread in the convex hull of the matrix of interest and Taylor method cannot take advantage of the values $\alpha_q(B)$.

Table 5.1 Results for the diffusion case ($b = 0$).

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	11	53	—	9.3	53	583	495	3.0e-14
L–H	10	55	4.8e0	1.0e1	0	550	460	3.3e-14 (6.6e-15)

For the case $b = 0.5$, results are reported in Table 5.2. In this case the convex hull of the spectrum resembles an ellipse inscribed in the skinny rectangle $[-100, 100] + \mathrm{i}[-15, 15]$ (see [9]). The results are comparable to the previous case. Finally, for $b = 1$ the matrix B is lower triangular with zero in the

Table 5.2 Results for the advection-diffusion case ($b = 0.5$).

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	11	53	—	9.3e0	53	583	495	2.7e-14
L–H	10	55	4.8e0	1.0e1	0	550	456	2.1e-14 (6.5e-15)

diagonal and the results are reported in Table 5.3. In this case, the set of Leja–Hermite points was chosen in order to maximize θ_m according to (4.3), independently of the number of repeated zeros (since the sequence $\{\alpha_q(B)\}_q$ does not decrease). As a result, it was selected a smaller s with respect to Taylor method and the number of iterations is smaller. The reordering of the points slightly improves the result.

Table 5.3 Results for the advection-diffusion case ($b = 1$).

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	11	53	—	9.3e0	53	583	474	4.4e-15
L–H	10	55	5.0e0	1.0e1	2	550	422	1.3e-14 (1.1e-14)
L–H reord.	10	55	5.0e0	1.0e1	2	550	420	1.2e-14 (1.0e-14)

5.1.2 Advection matrices

We consider the discretization of the one-dimensional advection operator ∂_x in the space domain $[0, 1]$ with periodic boundary conditions. We consider the first order upwind and the second order central schemes

$$\frac{1}{h}\begin{bmatrix} 1 & & & -1 \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad \frac{1}{2h}\begin{bmatrix} 0 & 1 & & -1 \\ -1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ 1 & & -1 & 0 \end{bmatrix} \in \mathbb{R}^{n\times n},$$

with $h = 1/70$ ($n = 70$). Both matrices are normal, the second is skew-symmetric. After applying the standard shift, the values for $\alpha_q(B)$ are constant and equal to $1/h = 70$, for $q \leq 8$. For $k = 0, 1, \ldots, n-1$ the eigenvalues are $\lambda_k = \mathrm{e}^{2\pi i k h}/h$ and $\lambda_k = \mathrm{i} \cdot \sin(2\pi k h)/h$, respectively. The initial vector has the components $v_i = \mathrm{e}^{-10(ih-1/2)^2/2}$. For the upwind scheme, we can see in Table 5.4 that interpolation at Leja–Hermite points outperforms Taylor method, thanks again to the possibility to select a smaller scaling parameter s. For the central scheme, we tested complex conjugate Leja points. Due to the

Table 5.4 Results for the advection case with upwind finite differences.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	8	51	—	8.8e0	51	408	361	4.0e-13
L–H	7	55	5.0e0	1.0e1	2	385	326	4.1e-13 (1.8e-13)
L–H reord.	7	55	5.0e0	1.0e1	2	385	324	5.6e-13 (2.0e-13)

distribution of the eigenvalues on the complex interval $\mathrm{i} \cdot [-70, 70]$, one can observe in Table 5.5 that real interpolation at complex conjugate points needs the fewer number of iterations.

Table 5.5 Results for the advection case with central finite differences.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	8	51	—	8.8e0	51	408	368	5.6e-15
L–H cplx	9	53	8.0e0	8.0e0	1	477	297	6.2e-15 (1.7e-15)

5.1.3 Schrödinger matrix

We finally consider the discretization of the free Schrödinger equation

$$\begin{cases} \dfrac{\partial u}{\partial t}(t,x) = \mathrm{i}\dfrac{\partial^2 u}{\partial x^2}(t,x), \\ u(0,x) = \mathrm{e}^{-10x^2}, \end{cases}$$

equipped with homogeneous Dirichlet boundary conditions in the spatial domain $[-1,1]$. The space step size h is $1/35$. The resulting matrix A has size $n \times n = 69 \times 69$ and is skew-symmetric. After the usual shift, the matrix B has $\alpha_q(B) = 2/h^2 = 2450$ for any value of $q \leq 8$ and the eigenvalues are distributed in the complex interval $\mathrm{i}[-2450, 2450]$. As initial vector v the discretization of $u(0,x)$ is used. We compute the solution at time $t = 1$. Once again, interpolation at complex conjugate Leja points gives the smallest number of actual iterations. We note that this outcome could not be predicted by the maximum number of iterations given after the initial parameter computation. Moreover, the relative error with respect to the reference solution is two orders of magnitude smaller, see Table 5.6. Due to the large difference of the computed errors with respect to the `expm(A)*v` solution, we decided to compute the reference solution in two other different ways. In the first, we *explicitely* computed the eigenvectors $\{v_j\}_{j=1}^n$ and the eigenvalues $\{\lambda_j\}_{j=1}^n$ of the matrix A, i.e. without using `eig`, and built the reference solution as $V \exp(\Lambda) V^T$. Then, we used the variable precision arithmetic in Matlab R2017a and computed the reference solution by `expm` with 32 decimal digits. In both cases, we found an error of 7.3e-11 for the Taylor approximation and of 3.0e-13 for the complex Leja–Hermite interpolation. A larger error in the Taylor approximation with respect to the interpolation at complex Leja points for a diagonal matrix with pure imaginary eigenvalues was already observed in [7, Section 4.3].

Table 5.6 Results for the Schrödinger matrix.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	249	55	—	9.9e0	55	13695	13197	7.3e-11
L–H cplx	292	55	8.4e0	8.4e0	1	16060	10220	2.7e-13 (7.2e-11)

5.2 Matrices with decreasing α_q values

In the following we are going to present several experiments where the sequence $\alpha_q(B)$ is decreasing. In particular this means applying [1, Thm. 4.2(a)] can be used to reduce the over-estimation of s.

5.2.1 `lesp` *matrix*

We consider the tridiagonal matrix A of size $n \times n = 20 \times 20$, with $(-2k-3)_{k=1}^{20}$ on the main diagonal, $(1/k)_{k=2}^{20}$ on the lower diagonal, and $(k)_{k=2}^{20}$ on the upper diagonal, multiplied by 100. It can be obtained by the MATLAB command `100*gallery('lesp',20)`. After the standard shift, the values of $\alpha_q(B)$ range from $\|B\|_1 = \alpha_1(B) = 3900$ to $\alpha_8(B) \approx 3383$. The eigenvalues of B lie in the real interval $[-2000, 2000]$. The initial vector has components $v_j = j$, $j = 1, 2, \ldots, n$. The results are collected in Table 5.7, and 5.8. In this example,

Table 5.7 Results for the `lesp` matrix.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	343	55	—	9.9e0	55	18865	12355	6.9e-14
L–H	348	54	6.6e0	9.9e0	41	18792	12533	2.0e-13 (2.7e-13)
L–H reord.	348	54	6.6e0	9.9e0	41	18792	10458	2.3e-13 (3.0e-13)

the best performance is reached by interpolation at reordered Leja–Hemite points selected as in (4.2), since the very efficient early termination criterion takes place.

Now we consider the same matrix scaled by 1/25. The scaling parameter and the maximum number of iterations roughly scale accordingly. After the standard shift, we have $\alpha_1(B) = 156$ and $\alpha_2(B) \approx 152.9$. One may think it is not worth to compute further values in the α_q sequence, since they seem to decrease too slowly to give a clear advantage and cost hundreds of matrix-vector products. So, we force the methods to use only these two values. In such a situation, a large number of zeros among interpolation points would probably not help and the choice like (4.3) would at least minimize the maximum number of iterations. This is confirmed as it can be seen in Table 5.8. Both methods optimize their performance with the value $\alpha_2(B)$ and interpolation at Leja–Hermite points allows smaller maximum and actual numbers of iterations.

Table 5.8 Results for the `lesp`/25 matrix, with $1 \le q \le \bar{q} = 2$.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	16	54	—	9.6e0	54	864	548	1.2e-15
L–H	15	55	5.0e0	1.0e1	2	825	437	1.3e-16 (1.3e-15)
L–H reord.	15	55	5.0e0	1.0e1	2	825	425	4.6e-15 (3.4e-15)

5.2.2 i·`lesp` *matrix*

Additionally, we consider the previous matrix multiplied by the complex identity i. In this case, the optimization stratery (2.11) selects degree $m = 55$ which, for the interpolation at complex Leja–Hermite points according to (4.4), means 56 repeated zeros. Therefore, the method coincides with truncated Taylor series. Among all our numerical experiments, in this one we obtained the largest errors with respect to the reference solution. On the other hand, the condition condition number of the $\exp(A)v$ problem, computed as in [2, (4.2)], is very large and takes value 1.0e10.

For this example, we report in Table 5.9 also the result (denoted by "L cplx") obtained with interpolation at complex conjugate Leja points as introduced in [7]. As it can be seen, interpolation at pure complex Leja points performs very bad. In fact, the eigenvalues of B/s are in the complex interval $\mathrm{i}[-4.4, 4.4]$, while the interpolation points are spread in $\mathrm{i}[-8.1, 8.1]$, since $\|B/s\|_1 \approx 8.1$ and it is not possible to take advantage of $\alpha_q(B/s)$ for $q > 1$.

Table 5.9 Results for the i·`lesp` matrix.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T, L–H cplx	343	55	—	9.9e0	55	18865	12775	1.1e-09
L cplx	479	54	8.1e0	8.1e0	0	25866	16568	1.1e-09 (4.9e-10)

5.2.3 `triw` *matrix*

Our last example consists of the matrix A of size $n \times n = 20 \times 20$ with -1 on the main diagonal and -4 on the remaining upper triangular part (see [2, Experiment 6]). It corresponds to the MATLAB matrix `-gallery('triw',20,4)`. The non-normality of the matrix, measured as

$$\kappa_1(B) = \frac{\|BB^* - B^*B\|_1}{\|B\|_1^2}$$

after applying the standard shift, is about 0.5. This is in fact not much different from $\kappa_1(B)$ for the shifted matrix B from section 5.1.1, with $b = 1$. However, the sequence of $\alpha_q(B)$ strongly decays, ranging from $\|B\|_1 = \alpha_1(B) = 76$ to $\alpha_8(B) \approx 16.29$. The initial vector has components $v_j = \cos j$, for $j =$

Table 5.10 Results for the `triw` matrix.

Method	s	m	c_m	θ_m	ℓ_m	max. its.	act. its.	rel. err.
T	2	54	—	9.6e0	54	108	42	3.2e-14
L–H	2	53	6.7e0	9.6e0	41	106	42	4.2e-14 (2.1e-14)
L–H cplx	2	55	9.9e0	9.9e0	55	110	42	3.2e-14 (0)
L cplx	10	52	7.7e0	7.7e0	0	520	344	4.5e-15 (3.2e-14)

$1, 2, \ldots, n$. The matrix B is nilpotent and therefore it is preferable to have enough zeros among the interpolation points and keep them in the leading positions. The results can be found in Table 5.10. Taylor truncated series and interpolation at Leja–Hermite points (both real and complex) give the same number of actual iterations, which is the minimum required by the exit criterion in order do discover the nilpotency of the matrix. Also in this case, interpolation at pure complex conjugate Leja points as in [7] would lead to a very bad result, as indicated in the last row of Table 5.10.

6 Discussion and conclusions

In this paper we aim to create a common framework for all those researchers who are approaching the problem of approximating the action of the matrix exponential $\exp(A)v$ through a polynomial. In fact, the effort spent to generalise and to make the backward error analysis tool so elastic and customisable can be particularly useful to people working on the matrix exponential. To this purpose, we developed a SageMath package called `expbea` which is publicly available at `https://bitbucket.org/expleja/expbea`. This code can be used for the analysis of interpolation polynomials for the exponential function, the computation of additional θ_m values for other tolerances or help tailor existing methods to a specific example for best performance.

In this work we consider classical Taylor approximation and several examples of polynomial interpolation. Besides the already known Leja points, we present Leja–Hermite points. They combine key features of both Taylor and Leja approximation. In addition, we describe a new way to extend the Leja ordering for repeated interpolation points. In particular, the Leja–Hermite interpolation has never been used before for matrix exponential approximation. In the following we summarize the results from the presented examples.

1. Interpolation at real Leja–Hermite points allows to fully use the values $\alpha_q(B)$. Moreover, the estimated cost is not larger than the cost predicted by Taylor approximation (compare Table 2.1 with Tables 4.2 and 4.3). For complex conjugate Leja–Hermite points, however, it is not possible to improve the boundaries θ_m of the Taylor approximation. Anyway, choice (4.5) allows to use the values $\alpha_q(B)$. Furthermore, it can outperform Taylor approximation, both in terms of total number of iterations and global accuracy. This is true whenever the spectrum of B has a vertical skinny shape and $\|B\|_1$ is a good estimate of the spectral radius (see Tables 5.5 and 5.6).
2. The effectiveness of sets of interpolation points containing some zeros depends more on the behavior of the sequence of values $\alpha_q(B)$, than on the normality of the matrix B.
   - If the sequence $\{\alpha_q(B)\}_q$ is flat, then the sets of Leja and Leja–Hermite points with reordering (both real and complex conjugate) lead in general to less iterations, thanks to the possibility of early termination. In fact, see Tables 5.1, 5.4, 5.5, and 5.6 (normal matrices), and Tables 5.2

and 5.3 (non-normal matrices), respectively. Of course, the choice between real or complex interpolation points should be based on some information about the spectrum of B. If not available from theory, it can be recovered in a cheap way by Gershgorin disks (see [7]).
- If the sequence $\{\alpha_q(B)\}_q$ strongly decreases, Taylor series approximation or interpolation at Leja–Hermite points lead to the best results (see Table 5.10). In this case the use of pure Leja points results in a much larger number of iterations.
- In the intermediate cases, real Leja–Hermite reordered point interpolation pays off (see Table 5.7). Concerning complex points, Leja–Hermite point interpolation avoids a quite huge amount of wasted iteration with respect to the use of pure complex conjugate Leja points (see Table 5.9). If the sequence of $\{\alpha_q(B)\}_q$ values is considered too slowly decreasing, the choice (4.3) of Leja–Hermite points which maximizes the value θ_m allows to reduce the maximum number of iterations and, in our example, also the actual number (see Table 5.8).

We therefore conclude that interpolation at Leja–Hermite points, being an extension of both Taylor truncated series and Leja point interpolation, can be considered a reliable, powerful and flexible method for the approximation of the matrix exponential applied to a vector. In fact, it has an a priori backward error estimate, a cost proportional to the degree of approximation without requiring the solution of linear systems, and the possibility to select proper sets of interpolation depending on some readly available information on the matrix.

References

1. Al-Mohy, A.H., Higham, N.J.: A new scaling and squaring algorithm for the matrix exponential. SIAM J. Matrix Anal. Appl. **31**(3), 970–989 (2009)
2. Al-Mohy, A.H., Higham, N.J.: Computing the action of the matrix exponential with an application to exponential integrators. SIAM J. Sci. Comput. **33**(2), 488–511 (2011)
3. Bergamaschi, L., Caliari, M., Vianello, M.: Efficient approximation of the exponential operator for discrete 2D advection-diffusion problems. Numer. Linear Algebra Appl. **10**(3), 271–289 (2003)
4. Berrut, J.P., Trefethen, N.: Barycentric Lagrange Interpolation. SIAM Rev. **46**(3), 501–517 (2004)
5. Bos, L.P., Caliari, M.: Application of modified Leja sequences to polynomial interpolation. Dolomites Res. Notes Approx. **8**, 66–74 (2015)
6. Caliari, M.: Accurate evaluation of divided differences for polynomial interpolation of exponential propagators. Computing **80**(2), 189–201 (2007)
7. Caliari, M., Kandolf, P., Ostermann, A., Rainer, S.: The Leja method revisited: backward error analysis for the matrix exponential. SIAM J. Sci. Comput. **38**(3), A1639–A1661 (2016)
8. Caliari, M., Ostermann, A., Rainer, S.: Meshfree exponential integrators. SIAM J. Sci. Comput. **35**(1), A431–A452 (2013)
9. Caliari, M., Vianello, M., Bergamaschi, L.: Interpolating discrete advection-diffusion propagators at Leja sequences. J. Comput. Appl. Math. **172**(1), 79–99 (2004)
10. Deadman, E.: Estimating the condition number of $f(A)b$. Numer. Algorithms **70**, 287–308 (2015)

11. Druskin, V.L., Knizhnerman, L.A.: Two polynomial methods of calculating functions of symmetric matrices. USSR Comput. Math. & Math. Phys. **29**(6), 112–121 (1989)
12. Fischer, T.M.: On the algorithm by Al-Mohy and Higham for computing the action of the matrix exponential: A posteriori roundoff error estimation. Linear Algebra Appl. **531**, 141–168 (2017)
13. Higham, N.J.: Functions of Matrices. SIAM, Philadelphia (2008)
14. Higham, N.J., Relton, S.D.: Higher order Fréchet derivatives of matrix functions and the level-2 condition number SIAM J. Matrix Anal. Appl. **35**(3), 1019–1037 (2014)
15. Higham, N.J., Tisseur, F.: A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. SIAM J. Matrix Anal. Appl. **21**(4), 1185–1201 (2000)
16. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010)
17. Kalantari, B.: Generalization of Taylor's theorem and Newton's method via a new family of determinantal interpolation formulas and its applications. J. Comput. Appl. Math. **126**, 287–318 (2000)
18. Luan, V.T., Ostermann, A.: Parallel exponential Rosenbrock methods. Comp. Math. Appl. **71**, 1137–1150 (2016)
19. Martínez, A., Bergamaschi, L., Caliari, M., Vianello, M.: A massively parallel exponential integrator for advection-diffusion models. J. Comput. Appl. Math. **231**(1), 82–91 (2009)
20. McCurdy, A.: Accurate computation of divided differences. Tech. rep., University of California — ERL (1980)
21. McCurdy, A., Ng, K.C., Parlett, B.N.: Accurate computation of divided differences of the exponential function. Math. Comp. **43**(168), 501–528 (1984)
22. Moret, I., Novati, P.: An interpolatory approximation of the matrix exponential based on Faber polynomials. J. Comput. Appl. Math. **131**, 361–380 (2001)
23. Novati, P.: A polynomial method based on Fejèr points for the computation of functions of unsymmetric matrices. Appl. Numer. Math. **44**, 201–224 (2003)
24. Opitz, V.G.: Steigungsmatrizen. Z. Angew. Math. Mech. **44**, T52–T54 (1964)
25. Reichel, L.: Newton interpolation at Leja points. BIT **30**, 332–346 (1990)
26. Schaefer, M.J.: A polynomial based iterative method for linear parabolic equations. J. Comput. Appl. Math. **29**, 35–50 (1990)
27. Stetekluh, J.: URL `http://stetekluh.com/NewtonPoly.html`
28. Tal-Ezer, H.: High degree polynomial interpolation in Newton form. SIAM J. Sci. Stat. Comput. **12**(3), 648–667 (1991)