THE LEJA METHOD REVISITED: BACKWARD ERROR ANALYSIS FOR THE MATRIX EXPONENTIAL

MARCO CALIARI[‡], PETER KANDOLF[*†], ALEXANDER OSTERMANN[†], AND STEFAN RAINER[†]

Abstract. The Leja method is a polynomial interpolation procedure that can be used to compute matrix functions. In particular, computing the action of the matrix exponential on a given vector is a typical application. This quantity is required, e.g., in exponential integrators.

The Leja method essentially depends on three parameters: the scaling parameter, the location of the interpolation points, and the degree of interpolation. We present here a backward error analysis that allows us to determine these three parameters as a function of the prescribed accuracy. Additional aspects that are required for an efficient and reliable implementation are discussed. Numerical examples illustrating the performance of our Matlab code are included.

Key words. Leja interpolation, backward error analysis, action of matrix exponential, exponential integrators, φ functions, Taylor series

AMS subject classifications. 65F60, 65D05, 65F30

1. Introduction.

In many fields of science the computation of the action of the matrix exponential is of great importance. As one example amongst others we highlight exponential integrators. These methods constitute a competitive tool for the numerical solution of stiff and highly oscillatory problems, see [9]. Their efficient implementation heavily relies on the fast computation of the action of certain matrix functions among those the matrix exponential is the most prominent one.

Given a square matrix A and a vector v the action of the matrix exponential is denoted by $\mathrm{e}^A v$. In general, the exponential of a sparse matrix A is a full matrix. Therefore, it is not appropriate to form e^A and multiply by v for large scale matrices. The aim of this paper is to define a backward stable method to compute the action of the matrix exponential based on the Leja interpolation. The performed backward error analysis allows one to predict and reduce the cost of the algorithm resulting in a more robust and efficient method.

For a given matrix A and vector v, one chooses a positive integer s so that the exponential $\mathrm{e}^{s^{-1}A}v$ can be well approximated. Due to the functional equation of the exponential we can then exploit the relation

$$\mathrm{e}^A v = \left(\mathrm{e}^{s^{-1}A}\right)^s v. \tag{1}$$

This results in an recursive approximation of $\mathrm{e}^A v = v^{(s)}$ by

$$v^{(i)} = \mathrm{e}^{s^{-1}A} v^{(i-1)}, \quad v^{(0)} = v. \tag{2}$$

There are various possibilities to compute the stages $v^{(i)}$. Usually, this computation is based on interpolation techniques. The best studied methods comprise Krylov subspace methods (see [14] and [12]), truncated Taylor series expansion [1], and interpolation at Leja points (see [5, 3]). In this paper we take a closer look on the Leja method (cf. (4) and (5) below) for approximating $v^{(i)} \approx L_{m,c}(s^{-1}A)v^{(i-1)}$.

[‡]Dipartimento di Informatica, Università di Verona, Italy, marco.caliari@univr.it
[†]Institut für Mathematik, Universität Innsbruck, Austria, peter.kandolf@uibk.ac.at, alexander.ostermann@uibk.ac.at, stefan.rainer@uibk.ac.at
[*]Corresponding author, Recipient of a DOC Fellowship of the Austrian Academy of Science at the Department of Mathematics, University of Innsbruck, Austria

Below we present two different ways of performing a backward error analysis of the Leja method. Our analysis indicates how the scaling parameter s, the degree of interpolation m and the interpolation interval $[-c, c]$ can be selected in order to obtain an appropriately bounded backward error by still keeping the cost of the algorithm at a minimum. Furthermore, we discuss how the method benefits from a shift of the matrix and we show how an early termination of the Leja interpolation can help to reduce the cost in an actual computation. As a last step we illustrate the stability and behavior of the method by some numerical experiments.

The paper is structured in the following way. In section 3 we introduce the backward error analysis and draw some conclusions from it. In particular we show how this analysis helps us to select the parameters s, m, and c. In section 4 we discuss some additional aspects for a successful implementation based on the Leja method. Section 5 presents some numerical examples dealing with different features and benchmarks for the method. In section 6 we finally give a discussion of the presented results.

For a reader not familiar with the Leja method we included a brief description in section 2.

2. The Leja method. Like every polynomial interpolation, the Leja method essentially depends on the interpolation interval and the position and number of interpolation points. The choice of these parameters directly influences the error of the interpolation and the cost. In this section we introduce the Leja method based on a sequence of Leja points in a real interval. The extension to a symmetrized complex sequence of points can be found in subsection 3.3.

Given an interval $[a, b]$, the Leja points are commonly defined as

$$\zeta_m \in \arg\max_{\zeta\in[a,b]} \prod_{j=0}^{m-1} |\zeta - \zeta_j|, \quad m \geq 1, \quad \zeta_0 \in \arg\max_{\zeta\in[a,b]} |\zeta|. \tag{3}$$

The interpolation polynomial of the exponential function is then given by

$$L_m(x; [a, b]) = \sum_{j=0}^{m} \exp[\zeta_0, \ldots, \zeta_j] \prod_{i=0}^{j-1} (x - \zeta_i)\,, \tag{4}$$

where $\exp[\zeta_0, \ldots, \zeta_j]$ denotes the jth divided difference. The scalar interpolation can be extended to the matrix case and rewritten into a two term recurrence relation for the actual computation, see [5, 3].

Due to the functional equation of the exponential it is always possible to shift the argument and perform the interpolation on a symmetric interval with zero as its center. This allows us to optimize the algorithm for symmetric intervals.

Let ζ_i be the Leja points in $[a, b]$ and ξ_i the Leja points in the symmetric interval $[-c, c]$ with same length. Then the relation $\zeta_i = \xi_i + \ell$ with $\ell = (a + b)/2$ and $c = (b - a)/2$ is valid. In practice, we use precomputed points on the interval $[-2, 2]$ and scale them to $[-c, c]$. Due to the functional equation of the exponential function

the shift can be singled out of the divided differences and we get

$$\begin{aligned} L_m(x;[a,b]) &= \sum_{j=0}^{m} \exp[\zeta_0,\ldots,\zeta_j] \prod_{i=0}^{j-1} (x-\zeta_i) \\ &= \sum_{j=0}^{m} \mathrm{e}^{\ell} \exp[\xi_0,\ldots,\xi_j] \prod_{i=0}^{j-1} ((x-\ell)-\xi_i) \\ &= \mathrm{e}^{\ell} L_m(x-\ell;[-c,c]). \end{aligned}$$

Therefore, it is always possible to interpolate on a symmetric interval around zero and apply the appropriate shifts to the argument and solution, respectively. In the following we will always select $\xi_0 = -c$ and consequently we get $\xi_1 = c$ and $\xi_2 = 0$. We denote the Leja interpolation polynomial of degree m on the interval $[-c,c]$ interpolating the exponential by

$$L_{m,c}(x) = L_m(x;[-c,c]). \tag{5}$$

Note that it is not necessary to shift the matrix in order to use a symmetric interval. Nevertheless, a well chosen shift can lead to faster convergence and can help to avoid round-off and overflow errors.

In order to determine a possible shift we define a rectangle $R = [\alpha,\nu] + \mathrm{i}[\eta,\beta]$ in the complex plane. We do this by splitting up the matrix into its Hermitian part A_{H} and skew Hermitian part A_{SH}. Furthermore, we find bounds for the field of values and the eigenvalues of these matrices with the help of Gerschgorin's disk theorem, i.e.

$$\begin{aligned} \sigma(A) = \sigma(A_{\mathrm{H}} + A_{\mathrm{SH}}) &\subseteq \mathcal{W}(A_{\mathrm{H}} + A_{\mathrm{SH}}) \subseteq \mathcal{W}(A_{\mathrm{H}}) + \mathcal{W}(A_{\mathrm{SH}}) \\ &= \mathrm{conv}(\sigma(A_{\mathrm{H}})) + \mathrm{conv}(\sigma(A_{\mathrm{SH}})). \end{aligned}$$

The four real numbers α, ν, η, and β are chosen to satisfy

$$\sigma(A_{\mathrm{H}}) \subseteq [\alpha,\nu] \qquad \text{and} \qquad \sigma(A_{\mathrm{SH}}) \subseteq \mathrm{i}[\eta,\beta]. \tag{6}$$

We note that in former versions of the Leja method ν was always assumed nonpositive and $-\eta = \beta$. These restrictions are no longer required here. In this sense, the method is now more general than previous versions. With the help of the rectangle R the interpolation interval was chosen in [5, 3] as the focal interval of the ellipse with smallest capacity circumscribing R. Here R is used to determine the type of interpolation (real or complex conjugate Leja points) and a possible shift $\mu \in \mathbb{C}$, see subsection 4.1.

We further note that, as stated in [13], the Leja ordering is of great importance for the stability of the method. Reichel suggests the interpolation interval $[-2,2]$. The length of the interpolation interval does not influence the numerical accuracy. Reichel suggests $[-2,2]$ only in order to avoid over- and underflow problems which may arise for large interpolation intervals and/or with very large values of the interpolation degree. In this version of the method we deviate from this choice. This is admissible since the largest interpolation interval and the largest used degree do not give rise to over- or underflow problems.

3. Backward error analysis. This section is devoted to the backward error of the action of the matrix exponential when approximated by the Leja method. We

first focus on the interpolation in a real interval, see subsection 3.3 for the extension to the complex case.

The concept of backward error analysis goes back to Wilkinson, see [16]. The underlying idea is to interpret the result of the interpolation as the exact solution of a perturbed problem $\mathrm{e}^{A+\Delta A}v$. The perturbation ΔA is the absolute backward error and we aim to satisfy $\|\Delta A\| \leq \mathrm{tol}\,\|A\|$ for a user given tolerance tol.

The here presented backward error analysis exploits a variation of the analysis given in [1]. For this, we define the set

$$\Omega_{m,c} = \{X \in \mathbb{C}^{n\times n} \colon \rho(\mathrm{e}^{-X}L_{m,c}(X) - I) < 1\},$$

where ρ denotes the spectral radius and $L_{m,c}$ is the Leja interpolation polynomial of degree m on the symmetric interval $[-c, c]$, see (4) and (5). Note that $\Omega_{m,c}$ is open in $\mathbb{C}^{n\times n}$ and contains a neighborhood of 0 for $m \geq 2$, since $\xi_2 = 0$. For $X \in \Omega_{m,c}$ we define the function

$$h_{m+1,c}(X) = \log(\mathrm{e}^{-X}L_{m,c}(X)). \tag{7}$$

Here log denotes the principal logarithm [8, Thm. 1.31]. As $h_{m+1,c}(X)$ commutes with X we get $L_{m,c}(X) = \mathrm{e}^{X+h_{m+1,c}(X)}$ for $X \in \Omega_{m,c}$. By introducing a scaling factor s such that $s^{-1}A \in \Omega_{m,c}$ for $A \in \mathbb{C}^{n\times n}$ we obtain

$$L_{m,c}(s^{-1}A)^s = \mathrm{e}^{A+sh_{m+1,c}(s^{-1}A)} =: \mathrm{e}^{A+\Delta A}, \tag{8}$$

where $\Delta A = sh_{m+1,c}(s^{-1}A)$ is the backward error resulting from the approximation of e^A by the Leja method $L_{m,c}(s^{-1}A)^s$.

On the set $\Omega_{m,c}$ the function $h_{m+1,c}$ has a series expansion of the form

$$h_{m+1,c}(X) = \sum_{k=0}^{\infty} a_{k,c}X^k. \tag{9}$$

In order to bound the backward error by a specified tolerance tol we need to ensure

$$\frac{\|\Delta A\|}{\|A\|} = \frac{\|h_{m+1,c}(s^{-1}A)\|}{\|s^{-1}A\|} \leq \mathrm{tol} \tag{10}$$

for a given matrix norm. As a consequence of this bound, one can select the scaling factor s (always a positive integer) such that (10) is satisfied for a chosen degree of interpolation m.

In contrast to [1] we have the endpoint c of the interpolation interval as an additional parameter to m and s for our analysis. In the following, we are going to introduce two different ways of bounding the backward error. The first one is closely related to the analysis presented in [1]. We study how (10) can be used to select the interpolation parameters when we perform a power-series expansion of the backward error. In the second approach we consider a contour integral formulation of $h_{m+1,c}$ and estimate the error along the contour.

3.1. Power-series expansion of the backward error. In this section we investigate bounds on the backward error represented by $h_{m+1,c}$. The analysis is based on a power-series expansion of $h_{m+1,c}$.

Starting from (9) we can bound $h_{m+1,c}(X)$ by

$$\|h_{m+1,c}(X)\| = \left\|\sum_{k=0}^{\infty} a_{k,c}X^k\right\| \leq \sum_{k=0}^{\infty} |a_{k,c}|\,\|X\|^k =: \widetilde{h}_{m+1,c}(\|X\|). \tag{11}$$

By inserting this estimate into (10) we get

$$\frac{\|\Delta A\|}{\|A\|} = \frac{\|h_{m+1,c}(s^{-1}A)\|}{\|s^{-1}A\|} \leq \frac{\widetilde{h}_{m+1,c}(s^{-1}\|A\|)}{s^{-1}\|A\|}. \tag{12}$$

Since zero is among the interpolation points for $m \geq 2$ we get

$$\frac{\widetilde{h}_{m+1,c}(\theta)}{\theta} = \sum_{k=1}^{\infty} |a_{k,c}|\theta^{k-1}.$$

This is a monotonically increasing function for $\theta \geq 0$. Furthermore, for $c = 0$, the Leja interpolation reduces to the truncated Taylor series at zero. We thus have $a_{1,0} = 0$ for $m \geq 1$. The equation

$$\frac{\widetilde{h}_{m+1,c}(\theta)}{\theta} = \text{tol} \tag{13}$$

therefore has a unique positive root for c sufficiently small. Henceforth, we will call this root $\theta_{m,c}$. The number $\theta_{m,c}$ can be interpreted in the following way: for the interpolation of degree m in $[-c, c]$ the backward error fulfills $\|\Delta A\| \leq \text{tol}\|A\|$, if the positive integer s fulfills $s^{-1}\|A\| \leq \theta_{m,c}$. In other words, if the norm of a matrix is smaller than $\theta_{m,c}$, the interpolation of degree m with points in $[-c, c]$ has an error less than or equal to tol.

In the analysis up to now, we only used that zero is among the interpolation points. However, the following discussion requires the sequence of Leja points.

In order to compute $\widetilde{h}_{m+1,c}$ in a stable manner we expand $h_{m+1,c}$ in the Newton basis for Leja points in $[-c, c]$ as

$$h_{m+1,c}(X) = \sum_{k=m+1}^{\infty} h_{m+1,c}[\xi_0, \ldots, \xi_k] \prod_{j=0}^{k-1} (X - \xi_j I), \tag{14}$$

where $h_{m+1,c}[\xi_0, \ldots, \xi_k]$ denotes the kth divided difference. The above series starts with $k = m + 1$ as $h_{m+1,c}(\xi_j) = 0$ for $j = 0, \ldots, m$. Rewriting this series in the monomial basis we obtain (11) with the according coefficients $a_{k,c}$. In order to get reliable results for these coefficients we use 300 digits in the actual computation.

Figure 1 displays the path of $\theta_{m,c}$ for the Leja interpolation with respect to c for fixed m up to 100 and $\text{tol} = 2^{-53}$. For an actual computation one has to truncate the series (14) at some index M. We always used $M = 3m$.

As we now have a way of bounding the backward error we discuss the choice of the number of scaling steps in (2). We propose to select the integer s depending on m and c in such a way that the cost of the algorithm becomes minimal. We have the limitation that m is bounded by 100 to avoid problems with over- and underflow. However, we get several possibilities to select the free parameter c describing the interval.

The value $\theta_{m,0}$ corresponds to the truncated Taylor series expansion as described in [1]. A second possibility is to choose, for a fixed m, c in such a way that $\theta_{m,c}$ is maximal. This corresponds to the interpolation interval that admits the largest norm of A. A third possibility is to select the interpolation interval such that the right endpoint c coincides with $\theta_{m,c}$. These are the points on the diagonal in Figure 1.

A priori none of the above choices can be seen to be optimal for an arbitrary matrix. The choice $c = 0$ together with the smallest m such that $\theta_{m,0} \geq \|A\|$ is

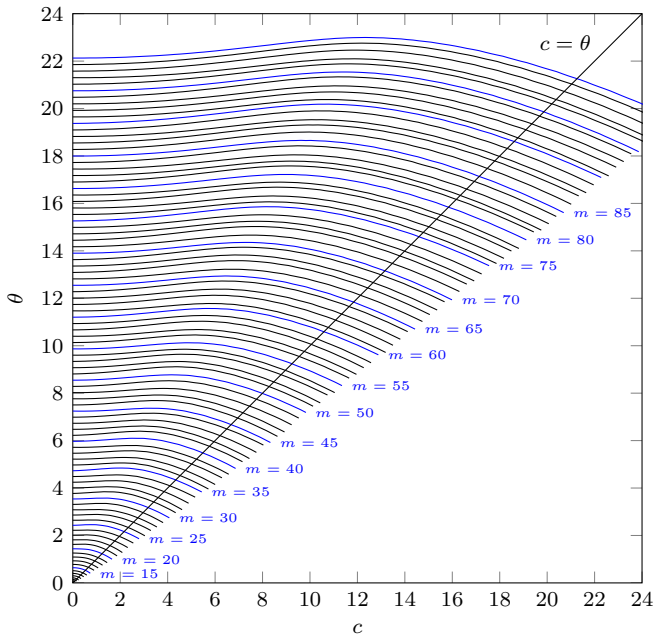


FIG. 1. *The root $\theta = \theta_{m,c}$ as a function of the right endpoint c of the interpolation interval for real Leja points in $[-c, c]$. The tolerance is set to* $\mathrm{tol} = 2^{-53}$. *Along each line the interpolation degree m is kept fixed.*

a good choice for a matrix A with all the eigenvalues clustered in a neighborhood of 0. On the other hand, if the convex hull of the eigenvalues of a matrix A, with $\|A\| \approx \delta$, is the interval $[-\delta, \delta]$, the choice $\theta_{m,\delta}$ with smallest m such that $\theta_{m,\delta} \geq \|A\|$ is preferable.

We choose $\theta_{m,c}$ according to the third option, which favors normal matrices where all eigenvalues lie in an interval. More precisely, we select

$$\theta_m = \min\{c\colon \theta_{m,c} = c\}. \tag{15}$$

This means that θ_m is the first intersection point of the graph $(c, \theta_{m,c})$ with the diagonal, cf. Figure 1.

The behavior of the curves $c \mapsto (c, \theta_{m,c})$ is not unexpected. Let us consider the approximation of e^θ by $L_{m,c}(\theta)$ for $\theta \geq 0$ and $s = 1$. Then $\tilde{h}_{m+1,c}(\theta)/\theta$ is an overestimate of the relative backward error $h_{m+1,c}(\theta)/\theta$. The value $\theta_{m,c}$ represents the maximum value for which $L_{m,c}(\theta_{m,c})$ is an acceptable approximation of $\mathrm{e}^{\theta_{m,c}}$. The value $\theta_{m,0}$ corresponds to interpolation at a set of confluent points at $c = 0$, i.e. the truncated Taylor series approximation. If we slightly increase the interpolation interval $[-c, c]$, about half of the interpolation points lie in $[0, c]$. Therefore, it is possible to have an acceptable interpolation up to $\theta_{m,c} \geq \theta_{m,0}$. If we continue to increase c, the mutual distance between interpolation points increases as well. When the interval gets too large, the number of interpolation points is too small to achieve the desired accuracy and the value $\theta_{m,c}$ starts to decrease.

Figure 1 shows that $\theta_{m,c} \geq \theta_m$ for all $0 \leq c \leq \theta_m$. Therefore, we can safely interpolate with degree m for all intervals $[-c, c]$ with $0 \leq c \leq \theta_m$. We will use this fact in subsection 4.3.

We compute θ_m by a combination of two root finding algorithms based on Newton's method. The inner equation (13) for computing $\theta_{m,c}$ is solved by an exact Newton iteration with an accuracy of 10^{-20}. The outer equation $\theta_{m,c} = c$ is also solved by Newton's method. This time, however, the necessary derivative is approximated by numerical differentiation. We compute the result up to an accuracy of 10^{-18}. The resulting values are truncated to 16 digits (double precision) and used henceforth as the θ_m values. In Table 1 we listed selected (rounded) values of θ_m for various m and certain tolerances.

TABLE 1
Samples of the (rounded) values θ_m for tolerances half *(*tol $= 2^{-10}$*),* single *(*tol $= 2^{-24}$*) and* double *(*tol $= 2^{-53}$*) for the real Leja interpolation.*

m	5	10	15	20	25	30	35
half	6.43e-01	2.12e+00	3.55e+00	5.00e+00	6.37e+00	7.51e+00	8.91e+00
single	9.62e-02	8.33e-01	1.96e+00	3.26e+00	4.69e+00	5.96e+00	7.44e+00
double	1.74e-03	1.14e-01	5.31e-01	1.23e+00	2.16e+00	3.18e+00	4.34e+00
m	40	45	50	55	60	65	70
half	1.00e+01	1.10e+01	1.23e+01	1.35e+01	1.48e+01	1.59e+01	1.71e+01
single	8.71e+00	1.00e+01	1.15e+01	1.27e+01	1.40e+01	1.52e+01	1.64e+01
double	5.48e+00	6.67e+00	7.99e+00	9.24e+00	1.06e+01	1.18e+01	1.32e+01
m	75	80	85	90	95	100	
half	1.84e+01	1.94e+01	2.07e+01	2.20e+01	2.30e+01	2.42e+01	
single	1.76e+01	1.87e+01	1.99e+01	2.12e+01	2.23e+01	2.35e+01	
double	1.46e+01	1.58e+01	1.71e+01	1.86e+01	1.99e+01	2.13e+01	

We next describe the choice of the parameters used in our implementation. For each m the optimal value of the integer s is given by

$$s = \lceil \|A\|/\theta_m \rceil. \tag{16}$$

We recall that we have chosen $m_{\max} = 100$. The cost of the interpolation is dominated by the number of matrix-vector products computed during Newton interpolation. Therefore, the cost is at most

$$C_m(A) := sm = m\lceil \|A\|/\theta_m \rceil, \tag{17}$$

resulting in the optimal m_* and corresponding s_* and c_* as

$$m_* = \underset{2\le m\le m_{\max}}{\arg\min} \left\{ m \left\lceil \frac{\|A\|}{\theta_m} \right\rceil \right\}, \quad s_* = \left\lceil \frac{\|A\|}{\theta_{m_*}} \right\rceil, \quad c_* = \theta_{m_*}. \tag{18}$$

Remark 3.1 (precomputed divided differences). We now have a fixed discrete set of interpolation intervals, given by θ_m. Therefore, the associated divided differences can be precomputed, once and for all.

Remark 3.2 (nonnormal matrices). The truncated Taylor series method is able to exploit the values $d_p = \|A^p\|^{1/p}$. As shown in [1, Eq. (3.6)], the backward error satisfies

$$\frac{\|\Delta A\|}{\|A\|} \le \frac{\widetilde{h}_{m+1,0}(s^{-1}\alpha_p(A))}{s^{-1}\alpha_p(A)},$$

where $\widetilde{h}_{m+1,0}(\theta) = \sum_{k=m+1}^{\infty} |a_{k,0}|\theta^k$ and $\alpha_p(A) = \min(d_p, d_{p+1})$ with arbitrary p subject to $p(p-1) \le m+1$. For nonnormal matrices, this can be a sharper bound as

$\alpha_p(A) \ll \|A\|$ is possible. Note that our method is not able to use this relation right away. This is due to the fact that the series representation of $\widetilde{h}_{m+1,c}$ starts at $k = 1$ for $c \neq 0$, see (11). As a result, we might use more scaling steps for such problems, see section 5 for some experiments. Nevertheless, the values d_p can be favorably used also for the Leja method, see subsection 4.3.

3.2. Contour integral expansion of the backward error. The selection of the scaling step and the length of the interpolation interval based on the norm of the matrix does not take into account the distribution of the eigenvalues. In this section we investigate bounds of the backward error, based on a contour integral expansion along ellipses that enclose the spectrum of the matrix. This introduces more flexibility as an ellipse can vary its shape from an interval to a circle. By this we can better capture the distribution of the eigenvalues of a matrix A than by simply taking the number $\|A\|$.

Again our aim is to find a bound for $h_{m+1,c}$. Due to the fact that zero is among the interpolation points for $m \geq 2$ it is convenient to write $h_{m+1,c}$ as

$$h_{m+1,c}(X) = X g_{m+1,c}(X). \tag{19}$$

For fixed $\varepsilon > 0$ we can rewrite (10) in the Euclidean norm as

$$\begin{aligned} \frac{\|\Delta A\|_2}{\|A\|_2} &= \frac{\|h_{m+1,c}(s^{-1}A)\|_2}{\|s^{-1}A\|_2} \\ &\leq \|g_{m+1,c}(s^{-1}A)\|_2 \\ &= \left\| \frac{1}{2\pi \mathrm{i}} \int_\Gamma g_{m+1,c}(z)(zI - s^{-1}A)^{-1}\,\mathrm{d}z \right\|_2 \\ &\leq \frac{\mathcal{L}(\Gamma)}{2\pi\varepsilon} \|g_{m+1,c}\|_\Gamma. \end{aligned} \tag{20}$$

Here $\Gamma = \partial K$ denotes the boundary of the domain K that contains $\Lambda_\varepsilon(s^{-1}A)$, the ε-pseudospectrum of $s^{-1}A$. The ε-pseudospectrum of a matrix X is given by

$$\Lambda_\varepsilon(X) = \left\{ z\colon \left\| (zI - X)^{-1} \right\|_2 \geq \varepsilon^{-1} \right\}.$$

The length of Γ is denoted by $\mathcal{L}(\Gamma)$ and $\|\cdot\|_\Gamma$ is the maximum norm on Γ. For given m, c, and K the last term in (20) can be computed in high precision. We use 300 digits and sample the contour in any performed computation.

For the time being, let us fix m. Furthermore, we assume that Γ is an ellipse, with focal interval equal to the interpolation interval $[-c, c]$, and its convex hull K encloses $\Lambda_\varepsilon(s^{-1}A)$. As a result of these assumptions, there is not only one ellipse but rather a two parameter family of ellipses $\Gamma_{\gamma,c}$. The parameters are the right endpoint c of the interpolation (focal) interval and the capacity γ of the ellipse, that is the half sum of the semi-axes. In the following we describe how to extract a discrete set of ellipses from the two parameter family of ellipses. This discrete set can then be stored and used in the algorithm.

We start by reducing the two parameter family of ellipses $\Gamma_{\gamma,c}$ to a one parameter family, only depending on the focal interval $[-c, c]$. For fixed c we have a family of confocal ellipses that are described by

$$\Gamma_{\gamma,c} = \left\{ z \in \mathbb{C}\colon z = \gamma w + \frac{c^2}{4\gamma w}, \quad |w| = 1 \right\}. \tag{21}$$

An ellipse $\Gamma_{\gamma,c}$ will be considered valid for interpolation if

$$\frac{\mathcal{L}(\Gamma_{\gamma,c})}{2\pi\varepsilon}\|g_{m+1,c}\|_{\Gamma_{\gamma,c}} \leq \text{tol} \tag{22}$$

is satisfied. For every c there exists an ellipse with largest capacity $\gamma =: \gamma_{m,c}$ satisfying (22) as tolerance equality, if m is sufficiently large. We single out this ellipse and thereby link the capacity directly with the focal interval and construct a one parameter family of ellipses.

We further reduce the number of ellipses by selecting only a discrete set of focal intervals for every m. More precisely, we use the known values θ_j for $j \geq m$ from (15) and Table 1, respectively. For these values we already have precomputed divided differences at hand and no extra storage is needed.

The overall procedure is as follows. For each interpolation (focal) interval $[-\theta_j, \theta_j]$, $j \geq m$ we compute the ellipse with largest capacity fulfilling (22) with $\varepsilon = \frac{1}{50}$ and store its semi-axes. We increase j as long as there is a $\gamma =: \gamma_{m,\theta_j}$ satisfying (22). Furthermore, we enforce the upper limit $j \leq 120$. With this selection we allow at most 20 ellipses for the maximal degree of interpolation $m_{\max} = 100$.

Figure 2 shows the stored ellipses for $m = 35$ and $\text{tol} = 2^{-53}$. The dashed circle has radius $\theta_{32} = 3.60$ corresponding to the largest circle with radius θ_j that fulfills (22) if a circle is used instead of an ellipse; see subsection 4.3 for further reasoning why to include this circle.

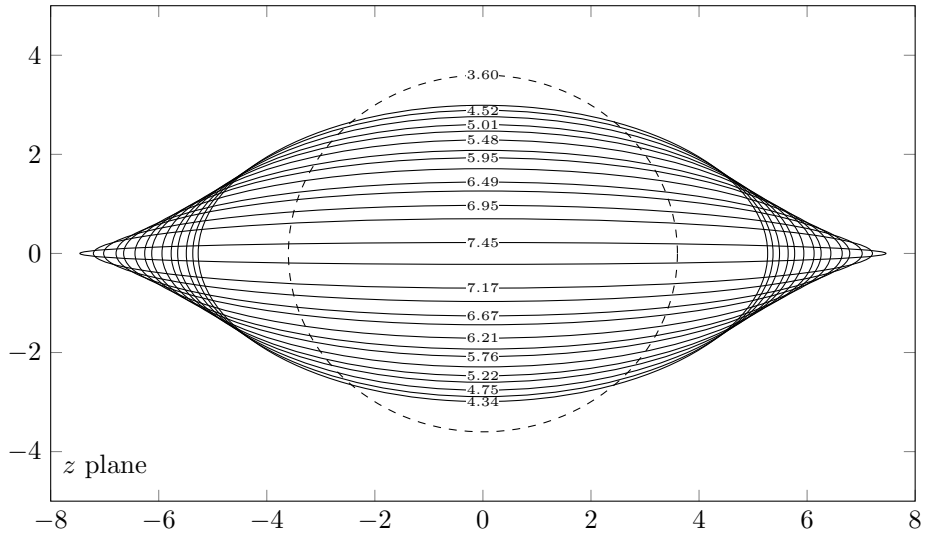


FIG. 2. *For $\varepsilon = \frac{1}{50}$, $m = 35$ and $\text{tol} = 2^{-53}$ the ellipses (21) satisfying (22) are shown for various focal intervals $[-\theta_j, \theta_j]$ with $j = 35, \ldots, 48$. The value θ_j (see Table 1) is indicated on the ellipse. The dashed circle has radius $\theta_{32} = 3.60$. It is the largest circle in lieu of the ellipse $\Gamma_{\gamma,c}$ that fulfills (22) for $m = 35$.*

We can see that, for larger focal intervals, the semi-minor axis decreases until we end up, in the limit, with an interval on the real axis. As we have additional information on the spectrum of the matrix at hand, it is possible to interpolate the exponential of certain matrices with fewer scaling steps than predicted by our power-series estimate.

Remark 3.3. If we reduce the size of the focal interval of our ellipses $\Gamma_{m,c}$ to a point, we end up with a circle. In fact, for a fixed m this circle is slightly smaller than the one obtained by the estimate θ_m.

For our backward error analysis we can interpret the value γ_{m,θ_j} in the following way. If we prescribe an interval $[-\theta_j, \theta_j]$ and select $m+1$ Leja points in this interval, we have $\|\Delta A\| \leq \operatorname{tol} \|A\|$ under the assumption that $s \geq 1$ is selected such that $\Lambda_\varepsilon(s^{-1}A) \subseteq \operatorname{conv}(\Gamma_{\gamma_{m,\theta_j},\theta_j})$.

Before discussing how to select the optimal ellipse for m, we show how to compute s for a given matrix A and an ellipse Γ. We recall that, with the help of Gershgorin's disk theorem, we can enclose the spectrum of A in a rectangle R with vertices $(\alpha, \beta), (\alpha, \eta), (\nu, \beta), (\nu, \eta)$, see (6). Furthermore, we assume that this rectangle is centered in zero ($-\alpha = \nu$ and $-\eta = \beta$), otherwise we shift the matrix accordingly. In order to keep the notation simple we consider a single ellipse Γ with focal interval $[-c, c]$ and capacity γ for which (22) is satisfied. As before we denote the convex hull of Γ by K. Let Δ_ε denote the open disc of radius ε around the origin. Hence, we have the following chain of inclusions

$$\Lambda_\varepsilon(A) \subseteq \mathcal{W}(A) + \Delta_\varepsilon \subseteq R + \Delta_\varepsilon.$$

The first inclusion connecting the pseudospectrum and the field of values can be found in [15]. The above inclusions state that if $R + \Delta_\varepsilon \subset K$ then $\|\Delta A\|_2 \leq \operatorname{tol} \|A\|_2$. Our aim is to determine the correct scaling factor s such that the inclusion $s^{-1}(R+\Delta_\varepsilon) \subseteq K$ is valid. We do this by computing the intersection of Γ with the straight line through zero and $r_\varepsilon = (\nu+\varepsilon, \beta+\varepsilon)$, the upper right vertex of the rectangle extended by ε. The procedure is illustrated in Figure 3.

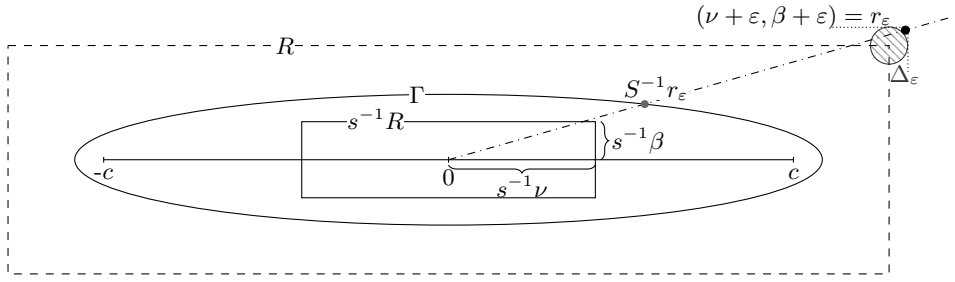


FIG. 3. *Illustration on the selection of the correct scaling factor s to fit the scaled and extended estimate of the pseudospectrum $s^{-1}(R+\Delta_\varepsilon)$ inside the ellipse Γ with convex hull $K \subset \mathbb{R}^2$. We have $S = \sqrt{(\nu+\varepsilon)^2 a^{-2} + (\beta+\varepsilon)^2 b^{-2}}$ and $s = \lceil S \rceil$.*

For

$$a = \gamma + \frac{c^2}{4\gamma}, \qquad b = \gamma - \frac{c^2}{4\gamma}$$

denoting the semi-axes of Γ we have

$$s = \left\lceil \sqrt{\frac{(\nu+\varepsilon)^2}{a^2} + \frac{(\beta+\varepsilon)^2}{b^2}} \right\rceil. \tag{23}$$

Due to our choice of r_ε it holds that $s^{-1}(R+\Delta_\varepsilon) \subseteq K$ for $s^{-1}r_\varepsilon \in K$.

We can now use the the degree of interpolation m to minimize the cost of the interpolation. This is done in the following way. As discussed above and illustrated in Figure 2, for every m, we get a family of ellipses with semi-axes

$$a_{m,\theta_j} = \gamma_{m,\theta_j} + \frac{\theta_j^2}{4\gamma_{m,\theta_j}} \quad \text{and} \quad b_{m,\theta_j} = \gamma_{m,\theta_j} - \frac{\theta_j^2}{4\gamma_{m,\theta_j}}, \tag{24}$$

where γ_{m,θ_j} fulfills (22). Recall that we have chosen $m_{\max} = 100$. We now use (23) to select the optimal ellipse for each m. In this family the optimal ellipse is identified as the one with the fewest scaling steps. For these optimal ellipses the number of scaling steps s_m is given by

$$S_{m,j} = \sqrt{\left(\frac{\nu+\varepsilon}{a_{m,\theta_j}}\right)^2 + \left(\frac{\beta+\varepsilon}{b_{m,\theta_j}}\right)^2}, \quad j_m = \operatorname*{arg\,min}_{j \geq m} \left\{ \lceil S_{m,j} \rceil \right\}, \quad s_m = \lceil S_{m,j_m} \rceil .$$

Now we can minimize with a similar cost function as in (17) over m and obtain our optimal degree of interpolation m_* and scaling factor s_* as

$$m_* = \operatorname*{arg\,min}_{2 \leq m \leq m_{\max}} \{ m s_m \}, \quad s_* = s_{m_*}. \tag{25}$$

The corresponding c_* is given by θ_j with $j = j_{m_*}$

3.3. Symmetrized complex Leja points. All the statements made in the previous sections remain valid if we use complex conjugate Leja points [4]. The advantage of such points lies in the better handling of matrices that have eigenvalues with dominant imaginary parts. This situation is characterized by a height-to-width ratio of more than one for the rectangle R. Examples include the (real) discretization matrices of transport equations or the discretization of the Schrödinger operator (a complex matrix) which has eigenvalues on the negative imaginary axis.

On the interval $\mathrm{i}\,[-c, c]$ on the imaginary axis, symmetrized or conjugate complex Leja points are defined as

$$\xi_m \in \operatorname*{arg\,max}_{\xi \in \mathrm{i}[-c,c]} \prod_{j=0}^{m-1} |\xi - \xi_j|, \quad \xi_{m+1} = -\xi_m \quad \text{for } m \geq 1 \text{ odd, and } \xi_0 = 0.$$

We use conjugate complex pairs of points rather than standard Leja points in an interval along the imaginary axis as this allows real arithmetic for real arguments. This gives rise to polynomials of even degree.

To allow conjugate complex Leja points in our backward error analysis we only need to change the actual computation of the values $\theta_{m,c}, \theta_m$ and $\gamma_{m,c}$. The theory itself stays the same. Figure 4 displays the path of $\theta_{m,c}$ for complex conjugate Leja points in $\mathrm{i}\,[-c, c]$. Table 2 displays a selection of (rounded) θ_m values for various tolerances.

If we apply complex conjugate Leja points in the framework of subsection 3.2 we get ellipses for which the major axis is on the imaginary axis.

3.4. Extension to φ functions. The presented backward error analysis extends in a straightforward way to the φ functions which play an important role in exponential integrators, see [9]. We illustrate this here for the φ_1 function. For $A \in \mathbb{C}^{n \times n}$ and $w \in \mathbb{C}^n$ we observe that

$$\varphi_1(A)w = [I, 0] \exp\left(\begin{bmatrix} A & w \\ 0 & 0 \end{bmatrix} \right) \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

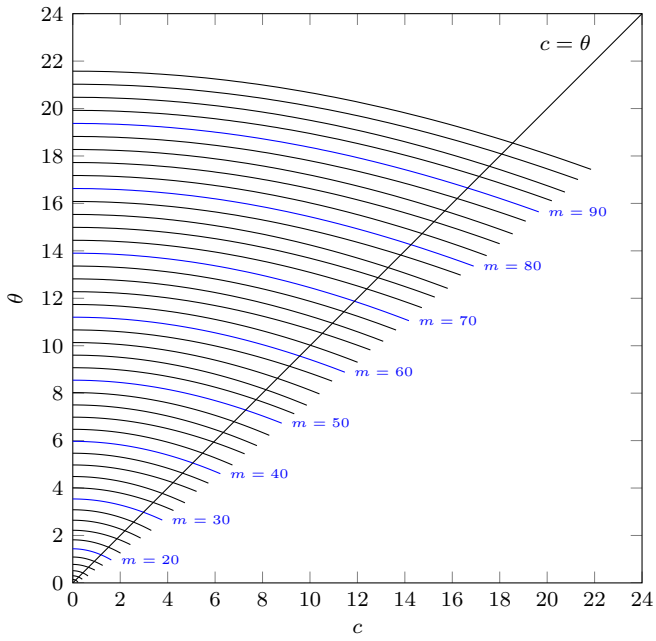


FIG. 4. *The root $\theta = \theta_{m,c}$ as a function of the right endpoint c of the interpolation interval for complex conjugate Leja points in* $\mathrm{i}[-c, c]$. *The tolerance is set to* $\mathrm{tol} = 2^{-53}$. *Along each line the interpolation degree m is kept fixed.*

TABLE 2

Samples of the (rounded) values θ_m with tolerances half *(*$\mathrm{tol} = 2^{-10}$*),* single *(*$\mathrm{tol} = 2^{-24}$*) and* double *(*$\mathrm{tol} = 2^{-53}$*) for complex conjugate Leja interpolation.*

m	10	20	30	40	50
half	1.94e+00	4.53e+00	7.11e+00	9.62e+00	1.21e+01
single	8.11e-01	2.99e+00	5.41e+00	7.85e+00	1.03e+01
double	1.16e-01	1.19e+00	2.98e+00	5.06e+00	7.29e+00
m	60	70	80	90	100
half	1.46e+01	1.70e+01	1.95e+01	2.20e+01	2.44e+01
single	1.27e+01	1.52e+01	1.77e+01	2.01e+01	2.25e+01
double	9.57e+00	1.19e+01	1.43e+01	1.67e+01	1.90e+01

see [14]. For the choice

$$\mathcal{A} = \begin{bmatrix} A & w \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

the backward error is preserving the structure, i.e.

$$\Delta\mathcal{A} = \begin{bmatrix} \Delta A & \Delta w \\ 0 & 0 \end{bmatrix}.$$

Thus the above analysis applies. For general φ functions we can extend our approach with the help of [1, Thm. 2.1].

4. Additional aspects of interpolation. By using the previously described backward error analysis to compute the values m_*, s_* and c_* a working algorithm can be defined. Nevertheless, the performance of the algorithm can be significantly

improved by some preprocessing and by introducing an early termination criterion. Moreover, interpolation in nonexact arithmetic will suffer from roundoff errors, in particular in combination with the hump phenomenon. We address all these issues in this section.

4.1. Spectral bounds and shift. In the above backward error analysis, it was assumed that the rectangle R lies symmetrically about the origin. In general, this requires a shift of A. On the other hand, it is clear that a well chosen shift μ satisfying $\|A-\mu I\| \leq \|A\|$ is beneficial for the interpolation (a lower degree or less scaling steps will be required). For the exponential function such a shift can easily be compensated by scaling. More precisely, if the shift μ is selected, we use

$$[\mathrm{e}^{\mu/s} L_{m,c}(s^{-1}(A-\mu I))]^s$$

as approximation of e^A.

For our algorithm a straightforward shift is to center the rectangle R at the origin, namely

$$\mu = \frac{\alpha+\nu}{2} + \mathrm{i}\frac{\eta+\beta}{2}. \tag{26}$$

If A is real then $\eta = -\beta$ and $\mu \in \mathbb{R}$. Therefore, a complex shift is only applied to complex matrices.

It is easy to see that for a Hermitian or skew Hermitian matrix A the proposed shift (26) coincides with the norm-minimizing shift presented in [8, Thm. 4.21(b)]. For a general matrix, the shift somewhat symmetrizes the spectrum of the matrix with regard to its estimated field of values.

The shift $\mu = n^{-1}\operatorname{trace} A$ used in [1] is a transformation that centers the spectrum of the matrix around the average eigenvalue. In many cases the two shifts are similar. Nevertheless, it is possible to find examples where one shift leads to better results than the other. The matrix `one-sided` of Example 4 is one of these cases. For the trace shift a symmetrization of the rectangle R might be required, resulting in a potential increase of scaling steps for the estimate based on (20). For the method proposed here, we always use (26) as shift.

4.2. Early termination criterion. The estimates based on (15) and (22) are worst case estimates and in particular do not take v into account. As a result, the choice of m_* is likely to be an overestimate and can be reduced in the actual computation. By limiting m in the computation of $L_{m,c}(s^{-1}A)v^{(i)}$ in (2) we introduce a relative forward error that again should be bounded by the tolerance tol. We propose to take

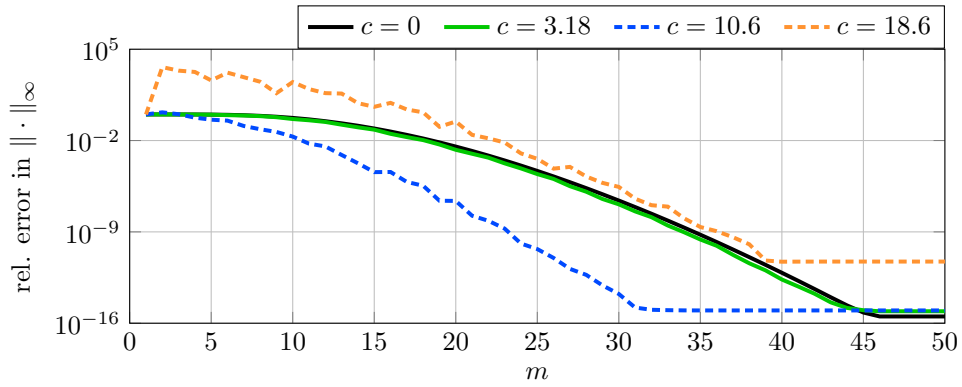
$$\begin{aligned} \|e_k\| &= \|L_{k,c}(s^{-1}A)v^{(i)} - L_{k-1,c}(s^{-1}A)v^{(i)}\| \\ &= |\exp[\xi_0,\ldots,\xi_k]| \left\| \prod_{j=0}^{k-1} (s^{-1}A - \xi_j I)v^{(i)} \right\| \leq \frac{\mathrm{tol}}{s}\|L_{k,c}(s^{-1}A)v^{(i)}\| \end{aligned} \tag{27}$$

as an a posteriori error estimate for the Leja method in the kth step. Experiments show that (27) turns out to be a good choice. In contrast to the method described in [1] we divide the tolerance by the amount of scaling steps. This potentially increases the number of iterations per step but in practice results in a more stable computation for normal matrices, see section 5. Nevertheless, it sometimes leads to results of

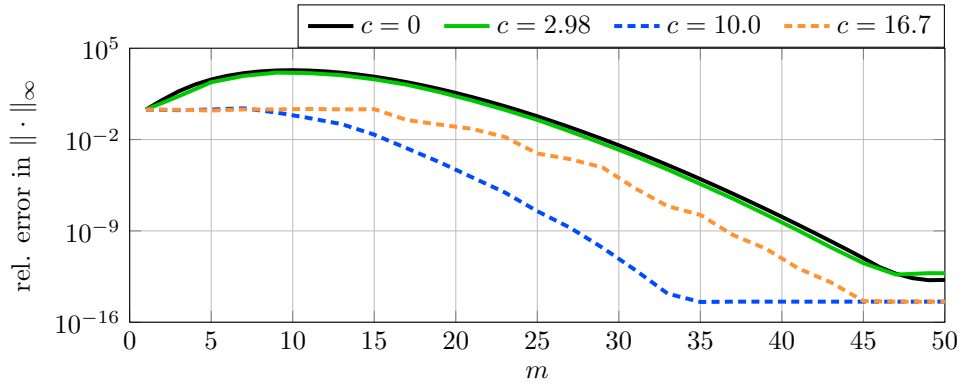
higher accuracy than prescribed. In practice it is advisable to take the sum of two or three successive approximation steps for the estimate as this captures the behavior better. On the other hand, it can also be beneficial to make the error estimate only every couple of iterations rather than in each step to save computational cost, see [3]. A second approach for an a posteriori error estimate for the Leja method based on the computation of a residual can be found in [10]. This procedure can also be used here. Furthermore, it is possible to adapt the early termination criterion to complex conjugate Leja points. With the help of an early termination criterion computational cost can be saved for certain matrices, see section 5.

4.3. Handling the hump phenomenon. In general, the interpolation error does not decrease monotonically with the degree of interpolation. Even worse, a distinct hump can occur in certain situations, see Figure 5. This hump can significantly reduce the accuracy of the interpolation due to roundoff errors. The phenomenon is linked to the distribution of the eigenvalues of a matrix with respect to interpolation interval. Note that the hump we are describing here is not the same as the one described in [11] for nonnormal matrices.

Figure 5 illustrates the problem for the matrix $A = \texttt{diag(linspace(-10,10,10))}$ and vector $v = [1, \dots, 1]^{\mathrm{T}}$. Figure 5a shows the real case. For $c = 0$ (i.e. the truncated



(a) Relative error vs. degree of interpolation m for $L_{m,c}(A)v$; real case.



(b) Relative error vs. degree of interpolation m for $L_{m,c}(\mathrm{i}\,A)v$; complex case.

FIG. 5. *Illustration of the hump phenomenon for the real and complex case. The used matrix* $A = \texttt{diag(linspace(-10,10,10))}$ *and* $v = [1, \dots, 1]^{\mathrm{T}}$.

Taylor series method) the final error is low and no hump is formed. If we increase the interpolation interval $[-c, c]$, we observe that the necessary degree of the interpolation gradually decreases, while the final error stays approximately constant. The *optimal* interpolation interval is reached when c approaches the largest eigenvalue. In the figure this is the case for $c = 10.6$. When the interval is increased further, however, a hump starts to form. This is due to the fact that the divided differences are significantly larger than the result, which has size e^{10}.

As can be seen in Figure 5b, the behavior is different for the complex case. Here the divided differences have modulus one and a hump forms if the interpolation interval is too small. Note that the smallest necessary degree of interpolation is again obtained by selecting the *optimal* interval.

In both cases the undesired behavior can be improved by obtaining a better estimate of the spectral radius and consequently reducing the interpolation interval. For this we employ the values $d_p = \|A^p\|^{1/p}$ which satisfy the well known relation

$$\rho(A) = \lim_{p\to\infty} \|A^p\|^{1/p}.$$

As long as the sequence of $\{d_p\}$ decreases, we adjust the interpolation interval accordingly.

For a general matrix A this phenomenon will influence the computation whenever $\|A\|$ strongly overestimates $\rho(A)$. In this case our algorithm chooses an interpolation interval that is far too large. Note that this happens in particular for nonnormal matrices.

For the estimate based on (15) the reduction of the interpolation interval is possible due to the behavior of the $\theta_{m,c}$ curve shown in Figure 1. However, if we use the estimate based on (20) the reduction of the interpolation interval is not straightforward. If we fit our rectangle R into an ellipse with semi-axis given by (24) then, in general, R will not fit into an ellipse with a smaller interpolation interval. We overcome this problem by adding a circle to the ellipses. We use the largest circle defined by $a_{m,\theta_k} = b_{m,\theta_k} = \theta_k$ for some $k \leq m$ that fulfills (22), see Figure 2 for an example. In most cases the radius of the circle is not going to be θ_m. If the values d_p indicate a reduction of the interval, we restrict the ellipse estimate to these circles and perform a reduction of the interpolation interval. The validity of this process can be checked in the same manner as for θ_m.

Remark 4.1. In the current version the algorithm does not allow to reduce the number s along with the decay of d_p as in [1]. Nevertheless, if a drastic decay is indicated it is possible to transform our method into Taylor interpolation by simply setting $c = 0$.

5. Numerical examples. In order to illustrate the behavior of our method we provide a variety of numerical examples. We use matrices resulting from the spatial discretization of time dependent partial differential equations already used in [3]. Furthermore, we also utilize examples from [1] and certain prototypical examples to illustrate some specific behavior of the method. All our experiments are carried out with Matlab 2013a. As a measure of the required computational work we use the number of matrix-vector products (mv) performed by the method, without taking into account preprocessing tasks. We will compare our method to the function `expmv` of [1].

Note that the Leja method also employs divided differences. They are computed as described in [2]. The used divided differences are precomputed as the employed

interpolation intervals are fixed.

In the following we are going to compare different variations of our algorithm based on the presented ways to compute the scaling factor s and degree of interpolation m.

`Algorithm 1:` Uses m_* and s_* given by (18).

`Algorithm 2:` Uses m_* and s_* given by (25).

In both algorithms, the early termination criterion (27) is used, as well as the shift and the hump test discussed in the previous section, if not indicated otherwise. For `Alg.2` the hump test procedure is only employed if the estimate of the scaling step is based on circles.

In Example 1 we take a look at the stability of the methods with and without early termination, Example 2 and Example 3 focus on the selection of the degree and the interpolation interval for the different variations of our algorithm, and Example 4 investigates the behavior for multiple scaling steps, i.e., $s > 1$.

Example 1 (early termination). This example is taken from [1, Exp. 2] to show the influence of the early termination criterion for a specific problem. We use the matrix A as given by `gallery('lesp',10)`. This is a nonsymmetric, tridiagonal matrix with real, negative eigenvalues. We compute $\mathrm{e}^{tA}v$ by `Alg.1` and `Alg.2`, respectively, for 50 equally spaced time steps $t \in [0, 100]$. We select the tolerance $\mathrm{tol} = 2^{-53}$ and $v_i = i$. As A is a nonnormal matrix, `Alg.2` is restricted to circles to allow the hump reduction procedure. The results of the experiments can be seen in Figure 6 where

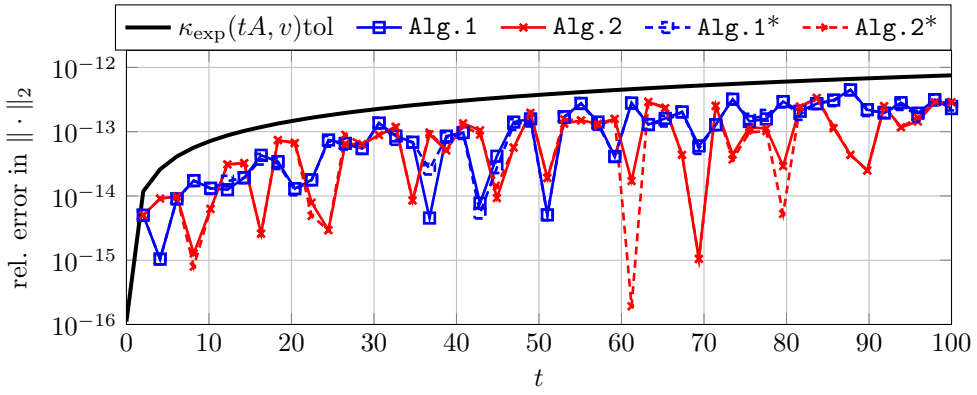


FIG. 6. *Time step t versus relative error in the 2-norm for the computation of $\mathrm{e}^{tA}v$ with tolerance $\mathrm{tol} = 2^{-53}$. The * indicates that no early termination was used. Note that there is almost no visible difference between the methods with and without the early termination in place.*

the solid line corresponds to the condition number (28) multiplied by the tolerance. We can not expect the algorithms to perform much better than indicated by this line. As condition number we use

$$\kappa_{\exp}(tA, v) := \frac{\|\exp(tA)\|_2\|v\|_2}{\|\exp(tA)v\|_2} + \frac{\|(v^{\mathrm{T}} \otimes I)K_{\exp}(tA)\|_2\|\operatorname{vec}(tA)\|_2}{\|\exp(tA)v\|_2}, \tag{28}$$

as defined in [1, Eq. (4.2)]. Here vec denotes the vectorization operator that converts its matrix argument to a vector by traversing the matrix column-wise. Furthermore, let $L(A, \Delta A)$ denote the Fréchet derivative of exp at A in direction ΔA given by

$$\mathrm{e}^{A+\Delta A} = \mathrm{e}^A + L(A, \Delta A) + o(\|\Delta A\|).$$

With the relation $\operatorname{vec}(L(A, \Delta A)) = K_{\exp}(A)\operatorname{vec}(\Delta A)$ the Fréchet derivative is given in its Kronecker form as $K_{\exp}(A)$. In addition we use the relation

$$\operatorname{vec}(L(A, \Delta A)v) = (v^{\mathrm{T}} \otimes I)\operatorname{vec}(L(A, \Delta A)).$$

For the computation we use the function `expm_cond` from the Matrix Function Toolbox, see [7].

Overall we can see that the algorithms behave in a forward stable manner for this example. The early termination criterion shows no significant increase in the error. Both algorithms are well below $\kappa_{\exp}(tA, v)$ for all values of t. For this rather small matrix we used the exact norm and not a norm estimate to allow for a better comparison.

Example 2 (advection-diffusion equation). In order to show the difference between the two suggested processes for selecting the interpolation interval for our algorithms, we use an example that allows us to easily vary the spectral properties of the discretization matrix. Let us consider the advection-diffusion equation

$$\partial_t u = a\Delta u + b(\partial_x u + \partial_y u)$$

on the domain $\Omega = [0, 1]^2$ with homogeneous Dirichlet boundary conditions. This problem is discretized in space by finite differences with grid size $\Delta x = (N + 1)^{-1}$, $N \geq 1$. As a result of the discretization we get a sparse $N^2 \times N^2$ matrix A. We define the grid Péclet number

$$\mathrm{Pe} = \frac{|b|\Delta x}{2a}$$

as the ratio of advection to diffusion, scaled by Δx. By increasing Pe the nonnormality of the discretization matrix can be controlled. In addition, Pe describes the height-to-width ratio of the rectangle R. The estimates for α and ν stay constant but $\eta = -\beta$ increases with Pe.

For the following computations the parameters are chosen as: $N = 20$, $a = 1$ and $b = \frac{2a\mathrm{Pe}}{\Delta x}$. As a result, for $\mathrm{Pe} = 0$ we get that R is an interval on the real axis and for $\mathrm{Pe} = 1$ a square. For $\mathrm{Pe} = 0$ the matrix is equal to `-(N+1)^2*gallery('poisson',N)`. The vector v is given by the discretization of the initial value $u_0(x, y) = 256 \cdot x^2(1 - x)^2y^2(1 - y)^2$. In the following discussion we call the shifted matrix again A.

TABLE 3

For varying grid Péclet number in Example 2 the selection of the degree of interpolation m_*, the actual degree due to the early termination m and the right endpoint c of the interpolation interval are shown. We compute $\exp(tA)v$ with a time step $t = 5e\text{-}3$, discretization parameter $N = 20$ and tolerance $\mathrm{tol} = 2^{-53}$. The error is measured relative to the result of the Matlab built-in function `expm` in the maximum norm.

	`Alg.1`				`Alg.2`				`expmv`		
	m_*	m	rel. err	c	m_*	m	rel. err	c	m_*	m	rel. err
$\mathrm{Pe} = 0$	54	32	3.66e-15	8.96	40	32	3.66e-15	8.96	52	44	2.88e-15
$\mathrm{Pe} = 0.2$	54	34	5.47e-15	8.96	49	35	4.17e-15	8.28	52	44	3.91e-15
$\mathrm{Pe} = 0.4$	54	35	2.21e-15	8.96	55	36	2.86e-15	9.24	52	44	2.34e-15
$\mathrm{Pe} = 0.6$	54	38	3.63e-15	8.96	62	40	6.36e-15	11.08	52	44	4.93e-15
$\mathrm{Pe} = 0.8$	54	41	2.98e-15	8.96	67	43	9.86e-15	11.34	52	43	2.59e-15
$\mathrm{Pe} = 1$	54	44	3.24e-15	8.96	72	49	4.50e-14	12.99	52	39	1.30e-15

Table 3 gives the results of an experiment where we varied the grid Péclet number. We show the results of the different selection procedures. The time step $t = 5\text{e-}3$ is

chosen such that `expmv` is able to compute the result without scaling the matrix. The actual degree of interpolation m and the relative error with respect to the method `expm` in the maximum norm are shown. As the maximum norm of the matrix stays the same (for fixed N) the parameters of `expmv` and `Alg.1` are always the same. For $\mathrm{Pe} = 1$ the eigenvalues of the matrix tA are in a small circle around zero and therefore the Taylor approximation requires a lower degree of interpolation.

On the other hand we can see that for a small height-to-width ratio (small Pe) the estimate based on ellipses, i.e. `Alg.2`, produces a significantly smaller m_* with the same actual degree m of interpolation and comparable error. This means that less scaling is required for larger t, cf. Example 4. When the rectangle R is closer to a square the algorithm still produces reliable results but is slightly less efficient than `Alg.1`.

Remark 5.1 (θ_m selection). As mentioned in subsection 3.1 it is possible to select the θ_m values differently depending on c. By selecting $\hat{\theta}_m = \max_c \theta_{m,c}$ the computation corresponding to $\mathrm{Pe} = 0$ gives the following results: $m_* = 51$, $m = 44$ and $c = 4.31$. This indicates a slower convergence with a similar error, as the eigenvalues of the shifted matrix are in $[-8.82, 8.82]$ but we interpolate in $[-4.31, 4.31]$.

Example 3 (hump and scaling steps). In order to illustrate the potential gain of testing for a hump in our algorithm, we use the matrix A given by the command `-1/2*gallery('triw',20,4)` and $v_i = \cos i$. This corresponds to [1, Experiment 6] with a single time step of size $t = 0.5$. In the following discussion we call the shifted matrix again A.

The 20×20 matrix A is an upper triangular matrix with 0 in the main diagonal and -2 on the strict upper triangular part. The 1-norm of the matrix is $\|A\|_1 = 38$ and $\rho(A) = 0$. For this example the truncated Taylor series method is optimal and stagnates after 20 iterations, in a single scaling step, with a final error of about 10^{-14}. We use this example to illustrate the hump phenomenon and the procedure to counteract it. This will result in a better performance of the Leja method, even though for this example it is not as efficient as `expmv`.

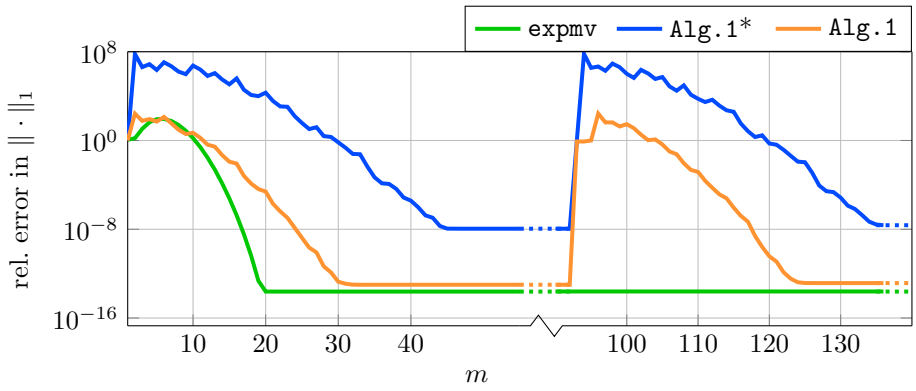


FIG. 7. *Relative error vs. interpolation degree m for the approximation of $\exp(A)v$. The plot illustrates the behavior of the hump reduction procedure. Here `Alg.1*` indicates that no hump reduction was performed. The relative error is measured with respect to $\exp(A/2)v$ in the first scaling step and $\exp(A)v$ in the second. No early termination criterion is used in the computation.*

In Figure 7 we illustrate the behavior of the hump reduction procedure, described in section 4. A first observation is that our method selects two scaling steps ($s =$

2). For this nonnormal matrix the rectangle R is a square and therefore we only show the results for `Alg.1`, as we can expect a better performance for this algorithm (cf. Example 2). We can see quite clearly that a hump of about 8 digits is formed when we use the initial guess of $c = \theta_{92} = 19.10$. As expected we get an error of about 10^{-8} in each scaling step and consequently a final error of the same size. In this experiment we deactivate the early termination criterion and therefore the algorithm uses $m = 92$ in each of the two scaling steps. Furthermore, for `Alg.1*` and `Alg.1` the relative error is measured with respect to $\exp(A/2)v$ in the first scaling step and $\exp(A)v$ in the second. For sake of comparison we run `expmv` without early termination and 182 iterations, and we measure the relative error with respect to $\exp(A)v$.

On the other hand, if we reduce the interval length, the hump is reduced as well. For this (shifted) matrix the values d_p are $(38, 26, 20, 16, 13, \ldots, 0)$ (see subsection 4.3), where $d_{20} = 0$. These values suggest that the interpolation interval should be reduced. In fact, by reducing the interval to $c = 0$ we would recover the truncated Taylor series method and therefore the optimal choice for this example. The cost of the computation of d_p can not be neglected and in a practical implementation p is therefore limited. Experiments have shown that $p \leq 5$ is a practical choice. Therefore, we use the interpolation interval $c = \theta_{45} = 6.67$ in the reduced case.

From Figure 7 we can see that the algorithm to reduce the impact of the hump is working. The hump is significantly reduced and the error is close to the error of the truncated Taylor series method. Nevertheless, the method takes about three times as many iterations (approximately 60) than the truncated Taylor series method, cf. Example 4.

Even though this procedure might not be necessary for accuracy it is still beneficial for the overall cost reduction. If we only require 8 digits of accuracy we do not need to reduce the interpolation interval to achieve this. However, for this example it would still be beneficial to reduce the interpolation interval, as the necessary degree of interpolation is reduced as well. This is related to the observations of Remark 5.1 and subsection 4.3. Here the norm of the matrix is a large overestimate of the spectral radius leading to slow convergence.

Example 4 (behavior for multiple scaling steps). In this experiment we investigate the behavior of the methods for multiple scaling steps. We use the two matrices of Example 2 and Example 3 from above and in addition the matrices `orani676` and `bcspwr10` which are obtained from the University of Florida sparse matrix collection [6], as well as several other matrices, see Table 4a. The sparse matrix `orani676` is real and nonnormal with 90158 nonzero entries, whereas `bcspwr10` is a real and symmetric sparse matrix with 13571 nonzero entries. The matrix `triu` is an upper triangular matrix with entries uniformly distributed on $[-0.5, 0.5]$. For the matrix `onesided` we have a 41×41 upper triangular matrix with one eigenvalue at 10 and 40 eigenvalues uniformly distributed on $[-10.1, -9.9]$, with standard deviation of 0.1. The values in the strict upper triangular part are uniformly distributed on $[-0.5, 0.5]$. Furthermore, we use `S3D` from [3, Example 3], a finite difference discretization of the three dimensional Schrödinger equation with harmonic potential in $[0, 1]^3$. The matrix `Trans1D` is a periodic, symmetric finite difference discretization of the transport equation in $[0, 1]$. For the matrices `orani676`, `S3D` and `Trans1d` complex conjugate Leja points are used in the computation.

As vector v we use $[1, \ldots, 1]^{\mathrm{T}}$ for `orani676`, $[1, 0, \ldots, 0, 1]^{\mathrm{T}}$ for `bcspwr10`, v as specified in Example 2 for `AD`, the discretization of $4096x^2(1-x)^2y^2(1-y)^2z^2(1-z)^2$ is used for `S3D`, the discretization of $\exp(-100(x-0.5)^2)$ for `Trans1D`, and $v_i = \cos i$

#	A	n	t	$\nu-\alpha$	$\beta-\eta$	κ_1	$\|\cdot\|_1$	$\|\cdot\|_2$	$\|\cdot\|_\infty$
1	orani676	2529	100	1.0e+03	1.0e+03	0.002	1.0e+03	3.2e+01	9.4e+00
2	bcspwr10	5300	10	2.6e+01	0.0e+00	0	1.4e+01	6.8e+00	1.4e+01
3	triw	2000	10	8.0e+03	8.0e+03	0.5	8.0e+03	5.0e+03	8.0e+03
4	triu	2000	40	1.0e+02	1.0e+03	0.021	1.0e+03	4.2e+01	1.0e+03
5	AD Pe=0	9801	1/4	8.0e+04	0.0e+00	0	8.0e+04	7.9e+04	8.0e+04
6	AD Pe=0	9801	1	8.0e+04	0.0e+00	0	8.0e+04	7.9e+04	8.0e+04
7	onesided	41	5	3.1e+01	1.2e+01	0.5	2.0e+01	1.1e+01	2.0e+01
8	S3D	27000	1/2	0.0e+00	5.7e+03	0	5.8e+03	5.7e+03	5.8e+03
9	Trans1D	1000	2	0.0e+00	2.0e+03	0	1.0e+03	1.0e+03	1.0e+03

(a) Summary of the spectral properties of the matrices.

		`Alg.1` 1-norm			`Alg.2` 1-norm			`expmv` 1-norm		
#	t	s	mv	rel.err	s	mv	rel.err	s	mv	rel.err
1	100	4639	41751	1.8e-11	3508	31572	2.2e-11	21	526	4.0e-08
2	10	6	157	7.8e-10	6	157	7.8e-10	5	171	7.2e-07
3	10	3408	98840	5.4e-09	2423	87233	1.0e-07	1588	10425	1.1e-09
4	40	1730	22495	1.6e-11	1268	17755	1.6e-12	59	960	4.3e-09
5	1/4	427	14945	1.0e-08	357	13923	1.9e-09	749	29211	2.2e-06
6	1	1705	59675	1.9e-08	1426	55614	3.3e-09	2995	116805	9.0e-06
7	5	5	129	3.0e-09	4	119	1.6e-10	8	296	2.1e-08
8	1/2	65	3185	2.2e-11	57	2793	8.0e-09	108	5400	1.3e-07
9	2	89	4539	9.5e-13	79	3871	1.4e-08	150	5135	3.6e-08

(b) Results for each matrix and the used algorithms, respectively.

TABLE 4

Results for Example 4. For a tolerance of tol $= 2^{-24}$ *we compute* $\exp(tA)v$ *in a single call of the respective algorithm. The value* s *indicates the number of scaling steps and* mv *denotes the number of matrix-vector products without preprocessing. The values* α, ν, η, ν *correspond to* (6).

for all other examples. This corresponds to [1, Exp. 7].

We summarize the properties of all the matrices used in this example in Table 4a. The tolerance is chosen as 2^{-24} and the relative error is computed with respect to `expmv` running with the highest accuracy. Furthermore we use

$$\kappa_1 = \frac{\|AA^* - A^*A\|_1}{\|A\|_1^2}$$

as an indicator for the nonnormality of the matrices. From now on we refer to the matrices by their number given in the first column of Table 4a.

We can see that for the nonnormal matrices $\{1, 3, 4\}$ the algorithm `expmv` is superior in terms of matrix-vector products, in comparison to both variants of our algorithm. This is largely due to the fact that for these matrices the method `expmv` can reduce the number of scaling steps based on the values d_p (see Remark 3.2). As the Leja method is not able to do this, the only way of getting comparable results for these example is by obtaining sharper bounds for the rectangle R in `Alg.2`. This could be achieved using the Matlab routines eig (based on LAPACK — Linear Algebra PACKage and suited for full matrices), eigs (based on ARPACK — ARnoldi PACKage and suited for sparse matrices), or an eigensolver of your choice fitted to the example. However, the computation can be very expensive and therefore is not practicable in a general purpose algorithm.

Furthermore, for these matrices the user specified norm has a relevant influence on the performance, as can be seen in Table 4b and Table 5, respectively. If the problem is considered with the 2- or the maximum norm the number of matrix-vector products is significantly reduced.

TABLE 5

For a tolerance of $\mathrm{tol} = 2^{-24}$ *we compute* $\exp(tA)v$ *in a single call of the algorithm* `Alg.1` *with the* 2- *and maximum norm, respectively. The value* s *indicates the number of scaling steps and* mv *denotes the number of matrix-vector products without preprocessing. The numbers* # *correspond to Table* 4a.

		`Alg.1` 2-norm			`Alg.1` ∞-norm		
#	t	s	mv	rel.err	s	mv	rel.err
1	100	142	2434	1.8e-10	43	2195	2.2e-11
2	10	2	106	7.0e-08	6	154	1.0e-09
3	10	2175	76141	5.8e-08	3408	98847	5.4e-09
4	40	66	2122	7.4e-10	1748	22732	1.5e-11
7	5	3	91	5.5e-10	5	128	3.0e-09

For the matrices $\{2, 5, 6, 7, 8, 9\}$ the results show a different picture. Here, the Leja method is clearly beneficial in terms of matrix-vector products. Furthermore, we also produce a smaller error in comparison to the truncated Taylor series approach. This is due to the fact that we divide the tolerance by s in the early termination criterion, cf. (27). In the case of the complex conjugate Leja points this leads to a higher accuracy than required. On the other hand, for the `AD` problem, the errors of the Leja methods increase by a factor of 2, if we increase t by a factor of 4, whereas the error for `expmv` increases by a factor of 4. This is due to the fact that the used early termination criterion (27) for the Leja method takes the number of scaling steps into account whereas `expmv` does not.

For matrices $\{1, 8, 9\}$ conjugate complex Leja points are used, see subsection 3.3. For the two normal matrices $\{8, 9\}$ the Leja method saves a lot of matrix-vector products in comparison to `expmv`. As here the rectangle R is a line, `Alg.2` is again superior to `Alg.1` as it leads to fewer scaling steps and less matrix-vector products.

Matrix 2 is normal. However, only for the 2-norm we have that $\|A\| = \rho(A)$. This is the reason why the number of scaling steps is only two in the 2-norm.

The final error of the methods is always comparable. The more precautious approach we propose leads sometimes to an increase in accuracy. Nevertheless, in the cases where the Leja method is beneficial it still uses significantly less matrix-vector products than `expmv`.

A comparison of `Alg.1` and `Alg.2` shows that none of the two approaches can be considered superior or the *better* overall choice. Due to the construction, `Alg.2` provides a scaling factor and a degree of interpolation that are independent of the norm, even though the Gerschgorin discs are closely related to the 1- and maximum norm. Nevertheless, the reduction of the interpolation interval is connected to a norm. In fact, this is also the case where the two methods do provide similar estimates for s. In total, if we always select the method with the least (predicted) computational cost we always use the more efficient methods as we save matrix-vector products. This indicates that a combination of the two algorithms, where we always select the one with the least expected cost is beneficial.

Depending on the specified norm, `Alg.1` has some significant fluctuations in performance.

6. Discussion. The backward error analysis presented in this work provides a sound basis for the selection of the scaling parameter s_* and the degree of interpolation m_* for the Leja method. With this information at hand the algorithm becomes in a sense direct, as the maximal number of matrix-vector products is known after the initial preprocessing. The cost of `Alg.1` is determined by the norm of the matrix,

whereas the cost of `Alg.2` is determined by the spectral information of the matrix. The convergence is monitored by the early termination criterion. The practical use of this approach is confirmed by the numerical experiments of section 5.

The algorithm can be adapted in a similar way as the `expmv` method to support dense output and provides essentially the same properties as [1, Algorithm 5.2]. In particular this means that the new algorithm also has some benefits in compared to Krylov subspace methods.

Note that in certain applications one has to compute $\mathrm{e}^{tA}V$ for a scalar t and a $n \times n_0$ matrix V. This problem, however, is not more general since the product tA can always be considered as a new matrix and the performed analysis extends to a matrix V instead of a vector v. This is especially interesting in comparison to Krylov subspace methods as the available implementations would need to be called repeatedly for each column of V.

In comparison to `expmv` the Leja method is especially beneficial for matrices where the values $d_1, \ldots, d_p$ do not vary much. In these cases the method saves matrix-vector products. On the other hand our method makes a higher preprocessing effort than `expmv`. This is due to the more complex selection procedure of m_* and s_* and the fact that we need an estimate of the field of values. As the overall cost is dominated by the matrix-vector products, this fact comes only into play for low-dimensional examples.

The combination of the two algorithms `Alg.1` and `Alg.2`, where we select the scaling parameter and the degree of interpolation based on the minimum of the predicted cost of the two algorithms, seems to be the logical choice for a combined (black box) algorithm. With the changes applied to the method it can be called for any matrix A, it is numerically stable, the costs are predictable and the effort for the implementation is manageable.

In the present version of our algorithm, the knowledge of d_p can not be used to properly scale the interpolation interval. However, and this is the focus of our future work, it is possible to modify the method and repeatedly use zero as interpolation point. These so-called Leja–Hermite methods will then be able to make use of d_p in a suitable fashion as `expmv`.

A Matlab implementation of the algorithm presented in this paper is available on the homepage https://numerical-analysis.uibk.ac.at/exponential-integrators.

Acknowledgment. We thank the referees for their constructive remarks which helped us to improve the presentation of the paper considerably.

REFERENCES

[1] Al-Mohy, A.H., Higham, N.J., 2011. Computing the action of the matrix exponential, with an application to exponential integrators. SIAM J. Sci. Comput. 33 (2), 488–511.
[2] Caliari, M., 2007. Accurate evaluation of divided differences for polynomial interpolation of exponential propagators. Computing 80 (2), 189–201.
[3] Caliari, M., Kandolf, P., Ostermann, A., Rainer, S., 2014. Comparison of methods for computing the action of the matrix exponential. BIT Numer. Math. 52 (1), 113–128.
[4] Caliari, M., Ostermann, A., Rainer, S., 2013. Meshfree exponential integrators, SIAM J. Sci. Comput. 35 (1), A431–A452.
[5] Caliari, M., Vianello, M., Bergamaschi, L., 2004. Interpolating discrete advection-diffusion propagators at Leja sequences. J. Comput. Appl. Math. 172 (1), 79–99.
[6] Davis, T.A., Hu, Y., 2011. The University of Florida sparse matrix collection. ACM Trans. Math. Software, 38 (1), 1–25.
[7] Higham, N.J., The Matrix Function Toolbox. http://www.ma.man.ac.uk/~higham/mftoolbox, Version 1.0, March 6, 2008.
[8] Higham, N.J., 2008. Functions of Matrices. Society for Industrial and Applied Mathematics

(SIAM), Philadelphia.
[9] Hochbruck, M., Ostermann, A., 2010. Exponential integrators. Acta Numerica 19, 209–286.
[10] Kandolf, P., Ostermann, A., Rainer, S., 2014. A residual based error estimate for Leja interpolation of matrix functions. Linear Algebra Appl. 456, 157–173.
[11] Moler, C., Van Loan, C., 2003. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. 45 (1), 3–49.
[12] Niesen, J., Wright, W.M., 2012 Algorithm 919: a Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators. ACM Trans. Math. Software 38 (3), Art. 22, 19 pp.
[13] Reichel, L., 1990. Newton interpolation at Leja points. BIT Numer. Math. 30 (2), 332–346.
[14] Sidje, R.B., 1998. Expokit. A software package for computing matrix exponentials, ACM Trans. Math. Software, 24 (1), 130–156.
[15] Trefethen, L.N., Embree M., 2005. Spectra and Pseudospectra. The Behavior of Nonnormal Matrices and Operators. Princeton University Press, Princeton, Oxford.
[16] Wilkinson, J.H., 1961. Error analysis of direct methods of matrix inversion. J. ACM 8 (3), 281–330.