Accurate evaluation of divided differences for polynomial interpolation of exponential propagators

M. Caliari, Padua

Abstract

In this paper, we propose an approach to the computation of more accurate divided differences for the interpolation in the Newton form of the matrix exponential propagator $\varphi(hA)\boldsymbol{v}$, $\varphi(z) = (e^z - 1)/z$. In this way, it is possible to approximate $\varphi(hA)\boldsymbol{v}$ with larger time step size h than with traditionally computed divided differences, as confirmed by numerical examples. The technique can be also extended to "higher" order φ_k functions, $k \geq 0$.

AMS Subject Classifications: 65D05, 65D20, 65M20.

Keywords: Accurate divided differences, Newton interpolation, exponential integrators.

1. Introduction

Exponential integrators for large stiff systems of ODEs

$$\begin{cases} \dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}(t)) \\ \boldsymbol{y}(0) = \boldsymbol{y}_0 \end{cases}, \tag{1}$$

where $\boldsymbol{y}(t) = [y_1(t), \ldots, y_n(t)]^{\mathrm{T}}$, are based on the efficient evaluation of $\varphi_k(hA)\boldsymbol{v}$, where h is a scalar related to the time step, $A \in \mathbb{R}^{n\times n}$ a matrix (e.g., the linear part [21], [13], [12], [19], [9] or the Jacobian [11], [6], [24] of $\boldsymbol{f}$), $\boldsymbol{v} \in \mathbb{R}^n$ a vector and

$$\varphi_0(z) = \exp(z), \quad \varphi_k(z) = \int_0^1 e^{(1-s)z} \frac{s^{k-1}}{(k-1)!} \mathrm{d}s, \quad k = 1, 2, \ldots \tag{2}$$

In particular, for a time-independent non-homogeneous linear system

$$\begin{cases} \dot{\boldsymbol{y}} = A\boldsymbol{y} + \boldsymbol{b} \\ \boldsymbol{y}(0) = \boldsymbol{y}_0 \end{cases} \tag{3}$$

it is possible to obtain an exact and explicit integrator

$$\begin{cases} \boldsymbol{y}(t_{j+1}) = \boldsymbol{y}_{j+1} = \boldsymbol{y}_j + \Delta t_j \varphi(\Delta t_j A)\boldsymbol{v}, \quad \boldsymbol{v} = (A\boldsymbol{y}_j + \boldsymbol{b}) \\ \boldsymbol{y}_0 = \boldsymbol{y}(0) \end{cases}, \tag{4}$$

where

$$\varphi(z) = \varphi_1(z) = \begin{cases} \frac{e^z-1}{z} & z \neq 0 \\ 1 & z = 0 \end{cases}.$$

Besides well-known Krylov methods (cf. [8], [22], [11]), other polynomial methods, based on the direct series expansion (cf. [16], [3]) or interpolation (cf. [23], [17], [18], [5]) of the scalar function φ_1 (or exp, for the homogeneous linear case), have been proposed. As pointed out by some authors (cf. [21], [23], [15], [5]), a difficult task for the latter methods is the accurate computation of the expansion or interpolation coefficients for high degrees, which leads to the use of higher precision floating point arithmetics, e.g., quadruple precision, or to the reduction of the maximum allowed time step size $\Delta t_j = t_{j+1} - t_j$, since the degree of the approximation must be increased with the time step size.

In particular, the interpolation at degree m in the Newton form

$$\varphi_1(hA)\boldsymbol{v} \approx \sum_{i=0}^{m} d_i \prod_{s=0}^{i-1} (hA - z_s I_n)\boldsymbol{v}, \tag{5}$$

where d_i are the divided differences of the function φ_1 at the points z_i and I_n the identity matrix of order n, requires divided differences with high relative accuracy, since the norm of the products of matrices can rapidly increase.

In this paper, following [15] for the function exp, we present an algorithm for the accurate computation of the divided differences for the function φ_1, based on the Taylor expansion of the matrix function $\varphi_1(Z_m)$, Z_m a bidiagonal matrix, with the points z_i on the main diagonal and ones below of dimension $m + 1$, that is much smaller than the dimension n of system (3). The scaling and squaring technique usually needed for the good accuracy of the Taylor expansion of exp (cf. [15]) or φ_1 is here replaced by a time-marching scheme based on (4). Moreover, with respect to [15], we also introduce an improvement (described in Sect. 3.2) which allows the computation of a larger number of divided differences without the loss of accuracy due to underflow issues. Numerical examples are performed using the ReLPM (Real Leja Points Method) for the approximation of $\varphi_1(hA)\boldsymbol{v}$ [4], [5], [6], [2], which is based on the interpolation in the Newton form at Leja points (see Sect. 2) of a real interval $[a, b]$ related to the spectrum (or the field of values) of the matrix A. Numerical experiments reported in Sect. 4.1 show that, for "large" h, the Newton polynomial interpolation converges only with our accurate computation of divided differences, whereas it does not in the case of standard computation of the divided differences, both in double and in quadruple precision.

For nonlinear systems (3), higher-order exponential integrators have been proposed (cf. [13], [12], [19], [24]), which require the approximation of $\varphi_k(hA)\boldsymbol{v}$ also for $k > 1$. In Sect. 3.1, we describe how to compute accurate divided differences for a general φ_k function, $k \geq 0$.

2. Standard divided differences

Let $\{x_i\}_{i=0}^m$ be a set of distinct real points in the interval $[a, b]$, with $x_0 = a$ and $x_m = b$. As it is well known, the divided differences for the function φ_1 are $d_i = \varphi_1[x_0, x_1, \ldots, x_{i-1}, x_i]$, $i = 0, \ldots, m$, where

$$\begin{aligned} \varphi_1[x] &= \varphi_1(x), \\ \varphi_1[x_0, x] &= \tfrac{\varphi_1[x]-\varphi_1[x_0]}{x-x_0} \end{aligned} \tag{6a}$$

and, for $j = 0, \ldots, m-1$,

$$\varphi_1[x_0, \ldots, x_{j-1}, x_j, x] = \frac{\varphi_1[x_0, \ldots, x_{j-1}, x] - \varphi_1[x_0, \ldots, x_{j-1}, x_j]}{x - x_j}. \tag{6b}$$

Hence, divided differences can be computed recursively following (6), even if this approach is very vulnerable to roundoff errors.

First of all, the simple computation of the function $\varphi_1(x)$ can lead to a less and less accurate value when x approaches 0. A more accurate computation of $\varphi_1(x)$, suggested in [10, pp. 22–24], can be achieved by

$$\varphi_1(x) = \begin{cases} 1 & \text{if } |x| < \varepsilon \\ \frac{e^x-1}{\log e^x} & \text{if } \varepsilon \le |x| < 1, \\ \frac{e^x-1}{x} & \text{if } 1 \le |x| \end{cases} \tag{7}$$

where ε is the machine precision, or using its Taylor series expansion for $|x| < 1$.

Moreover, as pointed out in [20], [23], in order to prevent numerical difficulties related to overflow or underflow, it is convenient to interpolate, by a change of variables, the function $\varphi_1(c+\gamma\xi)$ of the independent variable $\xi, \xi \in [-2, 2], c = (b+a)/2$ and $\gamma = (b-a)/4$. The Standard Recurrence scheme (SR, based on (6), of total cost $\mathcal{O}(m^2/2)$) for the computation of the divided differences for the interpolation of φ_1 $(h(c+\gamma\xi))$ on $[-2, 2]$ is reported in Table 1. It is useful to introduce the scalar parameter h which is related to the time step of the exponential integrator (see the numerical experiments in Sect. 4.1).

Table 1. Standard Recurrence scheme (SR) for divided differences

- INPUT: m, $\{\xi_i\}_{i=0}^m \in [-2, 2]$, h, c, γ
- FOR $i = 0, \ldots, m$
  - $d_i := \varphi_1\,(h(c+\gamma\xi_i))$
  - FOR $j = 1, \ldots, i$
    - $d_i := (d_i - d_{j-1})/(\xi_i - \xi_{j-1})$

  END FOR

  END FOR
- OUTPUT: Divided differences $\{d_i\}_{i=0}^m$, for $\varphi_1\,(h(c+\gamma\xi))$ at $\{\xi_i\}_{i=0}^m$

Finally, the sensibility of the Newton interpolation form to perturbations depends on the distribution and ordering of the interpolation points: in [20] it is shown that *Leja points* for the interval $[-2, 2]$ assure optimal accuracy. If $\xi_0 \in [-2, 2]$ with $|\xi_0| = 2$ and $\{\xi_i\}_{i=0}^{m-1}$ are the first m Leja points, the $(m+1)$-th Leja point ξ_m is defined recursively in such a way that

$$\prod_{i=0}^{m-1} |\xi_m - \xi_i| = \max_{\xi \in [-2,2]} \prod_{i=0}^{m-1} |\xi - \xi_i|. \tag{8}$$

The first m Leja points for $[-2, 2]$ can be extracted from a sufficiently larger set of uniform distributed points on the interval and stored once and for all. Alternatively, Fast Leja Points [1] can be used. Usually, ξ_0 is chosen equal to 2 and thus ξ_1 is the other extreme point -2.

For high degree interpolation, a large number of divided differences are required. Even with Leja points in $[-2, 2]$, due to cancellation errors the SR cannot produce accurate divided differences with magnitude smaller than machine precision (see Table 3 and [21] for an analogous with the coefficients of the Chebyshev series expansion of φ_1). This can lead to non-convergence of the polynomial interpolation in the matrix case, where the divided differences are the coefficients of products of matrices, which usually have very large norms.

3. Divided differences via matrix function

It can be shown (cf. [15]) that the divided differences $\{d_i\}_{i=0}^{m}$ for a function $f(h(c + \gamma\xi))$ of the independent variable ξ at points $\{\xi_i\}_{i=0}^{m} \in [-2, 2]$ are the first column of the matrix function $f(H_m)$, where

$$H_m = h(cI_{m+1} + \gamma\,\Xi_m), \quad \Xi_m = \begin{pmatrix} \xi_0 & & & & \\ 1 & \xi_1 & & & \\ & 1 & \ddots & & \\ & & \ddots & \ddots & \\ & & & 1 & \xi_m \end{pmatrix}.$$

In order to compute the first column of $\varphi_1(H_m)$, that is $\varphi_1(H_m)e_1$, we can consider approaches based on Taylor expansion of order p with scaling and squaring (cf. [15], $p = 16$), rational (p, q) Padé approximation with scaling and squaring (cf. [11], $p = q = 6$) or rational (p, q) Chebyshev (cf. [14], [8], $p = q = 14$ or $p = q = 16$), where p and q are respectively the polynomial degree of the numerator and of the denominator of the approximation.

Taylor and rational Padé expansions are accurate only when all $x_i = h(c + \gamma\xi_i)$ are close to zero. The matrix H_m has to be scaled by a factor τ in such a way that $\max_i |\tau x_i| < 1.59$ for Taylor expansion of $\varphi_1(x)$ (cf. [15]) or $\|\tau H_m\|_\infty < 0.5$ for rational Padé expansion of $\varphi_1(x)$ (cf. [11]).

The standard technique of scaling and squaring for $\exp(z)$, with $\tau = 2^{-J}$ for some J, could be extended to the function $\varphi_1(z)$ by applying the recurrence (cf. [11])

$$\varphi_1(2z) = \varphi_1(z)\left(\frac{z\varphi_1(z)}{2} + 1\right). \tag{9}$$

Since $\varphi_1(\tau H_m)$ is triangular, it would require $\mathcal{O}(m^3/6)$ operations. Here we prefer to exploit the exact time-marching scheme (4) in the following way: if one considers the ODE

$$\begin{cases} \dot{y}(t) = H_m y(t) + e_1 \\ y(0) = 0 \end{cases} \tag{10}$$

then $[d_0, d_1, \ldots, d_m]^T = \varphi_1(H_m)e_1 = y_J$, where

$$\begin{cases} y_{j+1} = y_j + \tau\varphi_1(\tau H_m)(H_m y_j + e_1), \ j = 0, \ldots, J-1, \quad \tau = 1/J \\ y_0 = 0, \end{cases} . \tag{11}$$

It is then possible to compute the whole matrix function $\varphi_1(\tau H_m)$ (we used the Taylor expansion of order $p = 16$, which does not require the solution of linear systems) and then to recover $\varphi_1(H_m)e_1$ with the cost of triangular matrix vector products $\mathcal{O}(m^2/2)$. The procedure for the computation of $\varphi_1(H_m)e_1$ can be split into the 3 steps:

(1) scale the matrix H_m by a factor $\tau = 1/J$ such that $\max_i|\tau x_i| < 1.59$;
(2) compute $\varphi_1(\tau H_m)$ by the Taylor expansion;
(3) recover, in J steps, $\varphi_1(H_m)e_1$ via (11).

Due to the special structure of H_m, the computation of $\varphi_1(\tau H_m)$ can be carried out essentially by Algorithm TS(II) in [15], and given as Algorithm TS(φ_k) in Table 2, adapted for the generic function $\varphi_k(\tau h(c + \gamma\xi))$. The upper triangular part of $\varphi_1(\tau H_m)$ is empty and can be used to store an auxiliary vector for the computation of $H_m y_j + e_1$ in (11): it is then possible to apply the time-marching scheme storing only a matrix of dimension $(m+1) \times (m+1)$.

3.1. Higher-order φ_k functions

For higher-order φ_k functions, $k > 1$, we need a time-marching scheme similar to (11) to recover $\varphi_k(H_m)e_1$ from $\varphi_k(\tau H_m)$. It is possible to write the time-marching scheme for the generic φ_k function, $k \geq 0$. First, notice that

$$\varphi_k(z) = \frac{1}{k!} + \varphi_{k+1}(z)z \tag{12}$$

and

$$h^k\varphi_k(hz) = \int_0^h e^{(h-s)z}\frac{s^{k-1}}{(k-1)!}\mathrm{d}s.$$

Table 2. Algorithm TS(φ_k) (Taylor Series for φ_k functions)

- INPUT: m, $\{\xi_i\}_{i=0}^m \in [-2, 2]$, h, c, γ, τ
- FOR $0 \le j \le i \le m$
  - $F_{i,j} := F_{j,i} := (\tau h \gamma)^i/(i + k - j)!$

  END FOR
- FOR $l = 2, \dots, 17$
  - FOR $j = 0, \dots, m - 1$
    - $F_{j,j} := \tau h(c + \gamma\xi_j) \cdot F(j, j)/(l + k - 1)$
    - FOR $i = j + 1, \dots, m$
      - $F_{j,i} := \tau h \left((c + \gamma\xi_i) \cdot F_{j,i} + \gamma \cdot F_{j,i-1})\right)/(l + i - j + k - 1)$
      - $F_{i,j} := F_{i,j} + F_{j,i}$

      END FOR

    END FOR

  END FOR
- FOR $j = 0, \dots, m$
  - $F_{j,j} := \varphi_k(\tau h(c + \gamma\xi_j))$

  END FOR
- OUTPUT: $F \approx \varphi_k(\tau H_m)$, where Fe_1 are the divided differences for $\varphi_k(\tau h(c + \gamma\xi))$ at $\{\xi_i\}_{i=0}^m$

If we consider, for $k = 0$,

$$\begin{cases} y'(t) = H_m y(t) \\ y(0) = e_1 \end{cases} \tag{13a}$$

and, for $k > 0$,

$$\begin{cases} \dot{y}(t) = H_m y(t) + \frac{t^{k-1}}{(k-1)!} e_1 \\ y(0) = 0 \end{cases} \tag{13b}$$

then $y(1) = \varphi_k(H_m)e_1$, since $y(t) = t^k \varphi_k(tH_m)e_1$. Then, set $t_0 = 0$, $y_0 = y(0)$, $\tau = 1/J$ and $t_j = j\tau$. From the variation-of-constants formula, we have

$$\begin{aligned} y_{j+1} = y(t_{j+1}) &= \varphi_0(\tau H_m) y_j + \int_{t_j}^{t_{j+1}} \varphi_0((t_{j+1} - s)H_m) \frac{s^{k-1}}{(k-1)!} e_1 \mathrm{d}s \\ &= \varphi_0(\tau H_m) y_j + \int_0^{\tau} \varphi_0((\tau - \zeta)H_m) \frac{(\zeta + t_j)^{k-1}}{(k-1)!} e_1 \mathrm{d}\zeta. \end{aligned} \tag{14}$$

The last integral is

$$\begin{aligned} &\int_0^{\tau} \varphi_0((\tau - \zeta)H_m) \frac{(\zeta + t_j)^{k-1}}{(k-1)!} e_1 \mathrm{d}\zeta \\ &= \sum_{i=0}^{k-1} \binom{k-1}{i} \int_0^{\tau} \varphi_0((\tau - \zeta)H_m) \frac{\zeta^i t_j^{k-1-i}}{(k-1)!} e_1 \mathrm{d}\zeta \end{aligned}$$

$$= \sum_{i=0}^{k-1} \frac{(k-1)!}{(k-1-i)!(k-1)!} \int_0^{\tau} \varphi_0((\tau-\zeta)H_m) \frac{\zeta^i t_j^{k-1-i}}{i!} e_1 \mathrm{d}\zeta$$

$$= \sum_{i=0}^{k-1} \frac{1}{(k-1-i)!} t_j^{k-1-i} \tau^{i+1} \varphi_{i+1}(\tau H_m) e_1. \tag{15}$$

Now, using (12), for $0 \le i+1 \le k$,

$$\varphi_{i+1}(z) = \varphi_k(z) z^{k-i-1} + \sum_{l=i+1}^{k-1} \frac{z^{l-i-1}}{l!}, \tag{16}$$

it is possible to rewrite all the φ_{i+1} functions in (15) in terms of φ_k and putting this inside (15) we get

$$\sum_{i=0}^{k-1} \frac{1}{(k-i-1)!} t_j^{k-1-i} \tau^{i+1} \varphi_{i+1}(\tau H_m) e_1$$

$$= \sum_{i=0}^{k-1} \frac{1}{(k-i-1)!} t_j^{k-1-i} \left(\tau^k \varphi_k(\tau H_m) H_m^{k-i-1} e_1 + \sum_{l=i+1}^{k-1} \frac{\tau^l H_m^{l-i-1}}{l!} e_1 \right).$$

Coming back to (14), we only need to add $\varphi_0(\tau H_m) y_j$ by (16) for $i+1=0$

$$\begin{aligned} y_{j+1} &= \tau^k \varphi_k(\tau H_m) \left(\sum_{i=0}^{k-1} \frac{(t_j H_m)^{k-i-1}}{(k-i-1)!} e_1 + H_m^k y_j \right) \\ &\quad + \sum_{i=0}^{k-1} \left(\frac{t_j^{k-1-i}}{(k-i-1)!} \sum_{l=i+1}^{k-1} \frac{\tau^l H_m^{l-i-1}}{l!} e_1 + \frac{(\tau H_m)^i}{i!} y_j \right) \\ &= \tau^k \varphi_k(\tau H_m) \left(\sum_{l=0}^{k-1} \frac{(t_j H_m)^l}{l!} e_1 + H_m^k y_j \right) \\ &\quad + \sum_{i=0}^{k-1} (\tau H_m)^i \left(\sum_{l=0}^{k-2-i} \frac{t_j^{k-1-l}}{(k-1-l)!} \frac{\tau^{l+1}}{(l+i+1)!} e_1 + \frac{y_j}{i!} \right). \end{aligned} \tag{17}$$

The solution of (13) $y(t_{j+1}) = y_{j+1}$ can be then exactly recovered by the computation of one matrix function $\varphi_k(\tau H_m)$, again by Algorithm TS(φ_k) in Table 2, applied to a vector involving H_m and y_j and $y_J = \varphi_k(H_m)e_1$. Notice that, whereas the φ_1 function can be used in the exact exponential integrator (4), where the time step Δt_j can be arbitrarily large and the computation of $\varphi_1(H_m)e_1$ may require a large number J of sub-steps in (11), higher order φ_k functions are used only for non-exact higher-order exponential integrators (cf. [13], [12], [19], [24]), with a truncation error depending on the time step size.

3.2. *Comments on algorithm* TS(φ_k)

In the first FOR loop in Algorithm TS(φ_k) in Table 2 (and also in Algorithm TS(II) in [15]), the terms $F_{i,j} = F_{j,i}$, which, at this stage, are the coefficients of the Taylor expansion, can underflow with respect to double precision machine arithmetic for i sufficiently large, and initialized to zero, thus leading to a loss of accuracy of the corresponding divided differences. The easiest way to overcome this problem is to fix a maximum number M_1 of computable divided differences (and, consequently, of the interpolation degree), with the drawback, when applying the interpolation to exponential integrators, of the reduction of the maximum allowed time step. In order to increase the maximum number of divided differences, one can pre-multiply the terms $F_{i,j}$ by a factor S "large" (say, $S = 10^{300}$) and compute them as

$$\begin{aligned}\tilde{F}_{i,j} &= S(\tau h\gamma)^i/(i+k-j)! = \exp\Big[\ln S(\tau h\gamma)^i - \ln\Gamma(i+k+1-j)\Big] \\ &= \exp\left[\ln(S) + i\ln(\tau h\gamma) - \ln\Gamma(i+k+1-j)\right],\end{aligned}$$

where $\Gamma(\cdot)$ is the Euler's Gamma function: clearly, the values $\ln\Gamma(i+k+1-j)$ have to be computed directly using the function $\ln\Gamma$ (cf., e.g., [7]). With this trick, the terms $\tilde{F}_{i,j}$ can again be initialized to zero, when the argument of the exponential function is negative and large in magnitude, but for $i > M_2 \gg M_1$ (see the next section for typical values of M_1 and M_2). At the end of the algorithm, the matrix elements $\tilde{F}_{i,j}$, $i \neq j$ have to be scaled by the factor S^{-1} to get $F_{i,j}$. It is clear that this strategy makes sense only when the final value of $F_{i,j}$ does not underflow.

In order to show the gain of such an approach, let us consider the following example: $1/170! + 1/171! = 172/171! = \exp(\ln(172) - \ln\Gamma(172)) \approx 1.3860 \cdot 10^{-307}$. But a direct computation in double precision arithmetic leads to $1.3779 \cdot 10^{-307} \approx 1/170!$, since $1/171!$ underflows, with a relative error of $1/172 \approx 0.0058$. The computation we suggest, $S^{-1}(\exp(\ln(S) - \ln\Gamma(171)) + \exp(\ln(S) - \ln\Gamma(172)))$, leads to a relative error of about $1.5728 \cdot 10^{-13}$.

4. Matrix polynomial interpolation: the Real Leja Points Method

In order to compute $\varphi_1(hA)\boldsymbol{v}$ we use the *Real Leja Points Method* (ReLPM), proposed in [5] and applied to advection-diffusion-reaction models in [4], [6], which has shown very attractive computational features. It is based on Newton interpolation at a sequence of Leja points on the real focal interval of a family of confocal ellipses in the complex plane. The use of Leja points is suggested, besides the optimal properties in the Newton interpolation form, by the fact that they guarantee maximal and superlinear convergence of the interpolant on every ellipse of the confocal family, and thus superlinear convergence of the corresponding matrix polynomials.

A key step in the approximation procedure is given by estimating cheaply a real focal interval, say $[a, b]$, such that the "minimal" ellipse of the confocal family which contains the spectrum (or the field of values) of the matrix has a *capacity*, the half sum of the semi-axis, that is not too large. The numerical experience with matrices arising

from stable spatial discretizations of parabolic equations, which are the main target of the ReLPM code [2], has shown that good results can be obtained at a very low cost, simply by intersecting the Gerschgorin's disks of the matrix with the real axis.

The kernel of the ReLPM code is given then by the interpolation of $\varphi_1(h(c+\gamma\xi))$, $[a, b] = [c-2\gamma, c+2\gamma]$ at the Leja points $\{\xi_i\}$ of the reference interval $[-2, 2]$. The matrix Newton polynomial of degree m is

$$p_m(hA)\boldsymbol{v} = p_{m-1}(hA)\boldsymbol{v} + d_m[(A - cI_n)/\gamma - \xi_{m-1}I_n]\boldsymbol{q}_{m-1}, \tag{18a}$$

where

$$\begin{aligned} \boldsymbol{q}_{m-1} &= \prod_{s=0}^{m-2}[(A - cI_n)/\gamma - \xi_s I_n]\boldsymbol{v} \\ [d_0, \ldots, d_m]^T &= \varphi_1(H_m)e_1 \\ p_0(hA)\boldsymbol{v} &= d_0\boldsymbol{v}. \end{aligned} \tag{18b}$$

4.1. Numerical example

Let us consider the advection-diffusion equation

$$\begin{cases} \dot{u} = \nabla^2 u - \langle \alpha, \nabla u \rangle + b & \Omega = (0, 1)^2 \\ u(0) = u_0 & t = 0 \\ u = 0 & \partial\Omega \end{cases}, \tag{19}$$

with $\alpha = (100, 100)$, $b \equiv 10$ and $u_0 \equiv 1$. A discretization with central, second order finite difference with $n = 10000$ uniformly distributed internal nodes leads to a system of ODEs of type (3). We are interested in the approximation of $\boldsymbol{y}(\Delta t) = \boldsymbol{y}_0 + \Delta t\varphi_1(\Delta t A)(A\boldsymbol{y}_0 + \boldsymbol{b})$, where the time step is $h = \Delta t = 0.005$, and the spectral interval for hA, estimated by Gerschgorin's disks, is $h \cdot [-81608, 0] = [-408.04, 0]$. Notice that this is a "large" time step, since there is a relative variation of the norm of the solution $\boldsymbol{y}(\Delta t)$ with respect to the norm $\boldsymbol{y}(0)$ of about one half.

First, we compute the first 223 divided differences at Leja points in $[-2, 2]$ for $\varphi_1(h(c+\gamma\xi))$, with $c = -40804$ and $\gamma = 20402$ with the standard recurrence scheme SR, in double and quadruple precision, and with the Taylor series approach TS(φ_k), without and with the use of the $\ln\Gamma$ function as described in Sect. 3.2. The results are collected in Table 3. Both double and quadruple precision standard differences stagnate around the corresponding machine precision. The divided differences produced by TS(φ_k) do not stagnate, but, without the use of the $\ln\Gamma$ function, the first terms initialized to zero appear at degree $M_1 = 143$, whereas with the $\ln\Gamma$ function at degree $M_2 = 255$. These values are typical for the discretization adopted, with a spectral interval for the matrix A of the type $[a, 0]$, since $\max_i|\tau h(c+\gamma\xi_i)| = |\tau h(c+\gamma\xi_1)| = |\tau h(-2\gamma + \gamma\cdot(-2))| = \tau h \cdot 4\gamma$, which should be smaller than 1.59

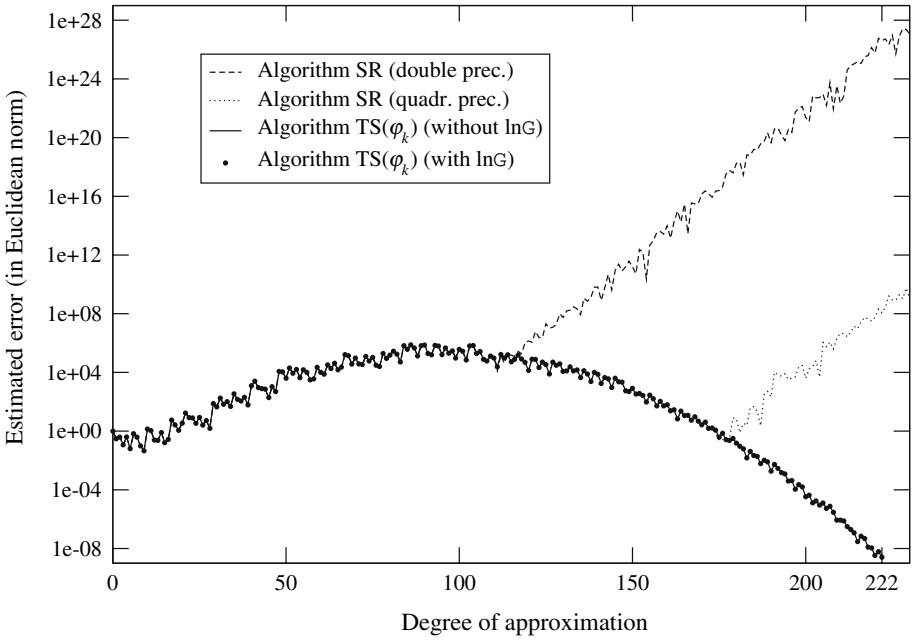

Fig. 1. Estimate relative errors for the matrix interpolation

(see Sect. 3) and the smallest term not initialized to zero in the first loop of TS(φ_k) in Table 2 is

$$\frac{(\tau h \gamma)^{142}}{(142+1)!} < \frac{(1.59/4)^{142}}{(142+1)!} \approx 3.3 \cdot 10^{-305}.$$

With the trick of the $\ln \Gamma$ function it is instead

$$\exp\left(\ln(S) + 254 \cdot \ln(1.59/4) - \ln \Gamma(254+2)\right) \approx 5.1 \cdot 10^{-307}.$$

As in IEEE specifications, here we are assuming that the minimum value in double precision is $2^{-1021-1} \approx 2.2 \cdot 10^{-308}$.

Next, we consider matrix interpolation. In the approximation (18), setting $\boldsymbol{p}_m = p_m(hA)(A\boldsymbol{y}_0 + \boldsymbol{b})$ we can choose

$$\|\boldsymbol{p}_{m+1} - \boldsymbol{p}_m\|_2 = |d_{m+1}| \cdot \|\boldsymbol{q}_{m+1}\|_2 \approx e_m = \|\boldsymbol{y}(h) - (\boldsymbol{y}_0 + h\boldsymbol{p}_m)\|_2 \qquad (20)$$

as an estimate of the approximation error. In Fig. 1, the estimated error relative to $\|\boldsymbol{y}_0\|_2$ computed with the four approaches described above is reported. With divided differences computed with the standard recurrence, the matrix interpolation does not converge, because they stagnate around machine precision (see Table 3) whereas the norm of the $\boldsymbol{q}_m$ increases in m. In this case, it would be necessary to split the time step Δt, choosing $h < \Delta t$ and applying the propagator (4). On the other hand, with

Table 3. Divided differences at Leja points in $[-2, 2]$ for $\varphi_1(h(c+\gamma\xi))$, $h = 0.005$, $c = -40804$, $\gamma = 20402$

i	SR, double precision	SR, quadruple precision	TS(φ_k)	TS(φ_k) with $\ln\Gamma$
0	0.1000000000000000E+01	0.1000000000000000E+01	0.1000000000000000E+01	0.1000000000000000E+01
1	0.2493873149691207E+00	0.2493873149691207E+00	0.2493873160804668E+00	0.2493873160804668E+00
2	0.1240809724536810E+00	0.1240809724536810E+00	0.1240809718939015E+00	0.1240809719021515E+00
3	0.3920872473780410E-01	0.3920872473780412E-01	0.3920872538812127E-01	0.3920872530828995E-01
4	0.5660663373584662E-01	0.5660663373584663E-01	0.5660663354179452E-01	0.5660663357156029E-01
5	0.1534637354851165E-01	0.1534637354851163E-01	0.1534637377241952E-01	0.1534637374323619E-01
6	0.5673617474021447E-01	0.5673617474021448E-01	0.5673617468962416E-01	0.5673617469551290E-01
7	0.4054418942883743E-01	0.4054418942883743E-01	0.4054418944316114E-01	0.4054418944174693E-01
8	0.1527635436587982E-01	0.1527635436587982E-01	0.1527635434538669E-01	0.1527635434957854E-01
9	0.3920433959851063E-02	0.3920433959851067E-02	0.3920434075661181E-02	0.3920434088825579E-02
...	...	...	...	...
112	0.8469614373623688E-16	0.8834183005397521E-16	0.8834183005288245E-16	0.8834183005278012E-16
113	0.1188514364826536E-16	0.2772629311256006E-16	0.2772629311322181E-16	0.2772629311291827E-16
114	0.3240368631730459E-16	0.2911536030336666E-16	0.2911536030321465E-16	0.2911536030334424E-16
115	0.2006953177750741E-16	0.7045993698022770E-17	0.7045993697802649E-17	0.7045993697645099E-17
116	– 0.7063941849691142E-17	0.4488135982782659E-17	0.4488135982878391E-17	0.4488135982913511E-17
117	0.1406463630196744E-16	0.8055985358453566E-17	0.8055985358426929E-17	0.8055985358423188E-17
118	– 0.5457066278115170E-17	0.2295195069895228E-17	0.2295195069914968E-17	0.2295195069912732E-17
119	0.1894414032455443E-16	0.8581934550224545E-18	0.8581934550164591E-18	0.8581934550157033E-18
120	0.2068651308109173E-16	0.2000532127819905E-18	0.2000532127661069E-18	0.2000532127960088E-18
121	– 0.6768996588349689E-17	0.4729811376057897E-18	0.4729811376100871E-18	0.4729811376025084E-18
122	0.2025827616715950E-16	0.2582275241189298E-18	0.2582275241173175E-18	0.2582275241203080E-18
...	...	...	...	...
175	0.1986120969064187E-17	0.1882472583092473E-34	0.3050584249685031E-34	0.3050584249692870E-34
176	0.1691566878614787E-17	0.1034680078634812E-34	0.2720985919917550E-34	0.2720985919917129E-34
177	– 0.7926503640570788E-17	0.5579089093878982E-35	0.6032969908044198E-35	0.6032969908021589E-35
178	0.6173080211273747E-17	– 0.3310704513159364E-35	0.2390790757618338E-35	0.2390790757624817E-35
179	0.4086477088521008E-17	– 0.8344634021247242E-34	0.3332239159156940E-35	0.3332239159155767E-35
180	0.9281494679154254E-17	– 0.3512312081982813E-34	0.7936339610173081E-36	0.7936339610166252E-36
181	– 0.5561550595128862E-17	– 0.2063686990328303E-35	0.2520231111342217E-36	0.2520231111346201E-36
182	0.6311765003586008E-18	– 0.7397395220498419E-35	0.1333876092256326E-36	0.1333876092254925E-36
183	0.1298110279052147E-16	– 0.4831234832409614E-35	0.2852445557935219E-37	0.2852445557907276E-37
184	– 0.5018803462836761E-17	– 0.3156888504586356E-34	0.3272636724983505E-37	0.3272636724991355E-37
185	0.6864899748127804E-17	– 0.2044375199739452E-34	0.9478821463068231E-38	0.9478821463013070E-38

the Taylor expansion approach the estimated relative error is below 10^{-8} at degree $m = 222$. Notice that convergence is achieved both without and with the use of the $\ln\Gamma$ function. However, since without the use of the $\ln\Gamma$ function there is surely a loss of accuracy at degree $M_1 = 143$, it is not safe to use the divided differences beyond that degree.

Finally, notice that the error estimate grows up to about 10^6 before starting to decrease. This is not unexpected, since the decrease of the divided differences starts to overtake the polynomial increase of the norm of the vector $\boldsymbol{q}_m$ only for a sufficiently high degree m.

5. Conclusions

We have shown that with divided differences accurately computed essentially by Algorithm TS(φ_k) in Table 2 and the time-marching scheme (11) (with a total cost of $\mathcal{O}(m^2/2)$), it is possible to interpolate in the Newton form the matrix function $\varphi_1(hA)\boldsymbol{v}$ with larger time step size h than with divided differences computed by the standard recurrence scheme (6) (both in double and in quadruple precision). Here A was the discretization matrix of a linear advection-diffusion PDE (19) and the matrix function $\varphi_1(hA)\boldsymbol{v}$ has been used inside the exact exponential integrator (4). This approach can be extended to higher-order functions φ_k, $k \geq 0$.

References

[1] Baglama, J., Calvetti, D., Reichel, L.: Fast Leja points. Electron. Trans. Numer. Anal. *7*, 124–140 (1998).
[2] Bergamaschi, L., Caliari, M., Martínez, A., Vianello, M.: Comparing Leja and Krylov approximations of large scale matrix exponentials. In: Computational science – ICCS 2006 (Alexandrov, V. N., van Albada, G. D., Sloot, P. M. A., and Dongarra, J., eds.). Lecture Notes in Computer Science, vol. 3994. Berlin Heidelberg: Springer 2006. 6th Int. Conf., Reading, UK, May 28–31, 2006, Proc., Part IV, pp. 685–692.
[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] Bergamaschi, L., Caliari, M., Vianello, M.: The ReLPM exponential integrator for FE discretizations of advection-diffusion equations. In: Computational science – ICCS 2004 (Bubak, M., Albada, G. D. v., Sloot, P. M. A., Dongarra, J., eds.). Lecture Notes in Computer Science, vol. 3039. Berlin Heidelberg: Springer 2004. 4th Int. Conf., Kraków, Poland, June 6–9, 2004, Proc., Part IV, pp. 434–442.
[5] Caliari, M., Vianello, M., Bergamaschi, L.: Interpolating discrete advection-diffusion propagators at Leja sequences. J. Comput. Appl. Math. *172*, 79–99 (2004).
[6] Caliari, M., Vianello, M., Bergamaschi, L.: The LEM exponential integrator for advection-diffusion-reaction equations. J. Comput. Appl. Math. (2006). In: Proc. of numerical analysis: the state of the art (NAC2005), Rende (CS), Italy, May 19–21, 2005, in press (available online December 5, 2006).
[7] Cody, W. J., Hillstrom, K. E.: Chebyshev approximations for the natural logarithm of the Gamma function. Math. Comp. *21*, 198–203 (1967).
[8] Gallopoulos, E., Saad, Y.: Efficient solution of parabolic equations by Krylov subspace methods. SIAM J. Sci. Statist. Comput. *13*(5), 1236–1264 (1992).
[9] González, C., Ostermann, A., Thalhammer, M.: A second-order Magnus-type integrator for non-autonomous parabolic problems. J. Comput. Appl. Math. *189*, 142–156 (2006).
[10] Higham, N. J.: Accuracy and stability of numerical algorithms, chap.: Principles of finite precision computation. Philadelphia: SIAM, pp. 22–24 (1996).

[11] Hochbruck, M., Lubich, C., Selhofer, H.: Exponential integrators for large systems of differential equations. SIAM J. Sci. Comput. *19*(5), 1552–1574 (1998).
[12] Hochbruck, M., Ostermann, A.: Explicit exponential Runge-Kutta methods for semilinear parabolic problems. SIAM J. Numer. Anal. *43*, 1069–1090 (2005).
[13] Caliari, M., Vianello, M., Bergamaschi, L.: Exponential Runge-Kutta methods for parabolic problems. Appl. Numer. Math. *53*(2–4), 323–339 (2005).
[14] Lu, Y. Y.: Computing a matrix function for exponential integrators. J. Comput. Appl. Math. *161*, 203–216 (2003).
[15] McCurdy, A., Ng, K. C., Parlett, B. N.: Accurate computation of divided differences of the exponential function. Math. Comp. *43*(186), 501–528 (1984).
[16] Moret, I., Novati, P.: The computation of functions of matrices by truncated Faber series. Numer. Funct. Anal. Optim. *22*(5–6), 697–719 (2001).
[17] Moret, I., Novati, P.: An interpolatory approximation of the matrix exponential based on Faber polynomials. J. Comput. Appl. Math. *131*(1–2), 361–380 (2001).
[18] Novati, P.: A polynomial method based on Fejér points for the computation of functions of unsymmetric matrices. Appl. Numer. Math. *44*(1–2), 201–224 (2003).
[19] Ostermann, A., Thalhammer, M., Wright, W.: A class of explicit exponential general linear methods. BIT *46*(2), 409–431 (2006).
[20] Reichel, L.: Newton interpolation at Leja points. BIT *30*(2), 332–346 (1990).
[21] Schaefer, M. J.: A polynomial based iterative method for linear parabolic equations. J. Comput. Appl. Math. *29*(1), 35–50 (1990).
[22] Sidje, R. B.: Expokit. A software package for computing matrix exponentials. ACM Trans. Math. Softw. *24*(1), 130–156 (1998).
[23] Tal-Ezer, H.: High degree polynomial interpolation in Newton form. SIAM J. Sci. Statist. Comput. *12*(1), 648–667 (1991).
[24] Tokman, M.: Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods. J. Comput. Phys. *213*, 748–776 (2006).

M. Caliari
Department of Pure and Applied Mathematics
University of Padova
Via Trieste 63
35121 Padova
Italy
e-mail: mcaliari@math.unipd.it