Available online at www.sciencedirect.com

ScienceDirect

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

www.elsevier.com/locate/cam

The LEM exponential integrator for advection–diffusion–reaction equations☆

Marco Caliari[a], Marco Vianello[a,*], Luca Bergamaschi[b]

[a]Department of Pure and Applied Mathematics, University of Padova, via Trieste 63, 35121 Padova, Italy
[b]Department of Mathematics Methods and Models for Applied Sciences, University of Padova, Italy

Received 4 August 2005; received in revised form 13 February 2006

Abstract

We implement a second-order exponential integrator for semidiscretized advection–diffusion–reaction equations, obtained by coupling exponential-like Euler and Midpoint integrators, and computing the relevant matrix exponentials by polynomial interpolation at Leja points. Numerical tests on 2D models discretized in space by finite differences or finite elements, show that the Leja–Euler–Midpoint (LEM) exponential integrator can be up to 5 times faster than a classical second-order implicit solver.
© 2006 Published by Elsevier B.V.

MSC: 65M20; 65D05; 65L05

Keywords: Advection–diffusion–reaction equations; Exponential integrators; Interpolation; Leja points

1. Semidiscretization of ADR models

In this paper we are concerned with the numerical solution of semilinear parabolic equations of the advection–diffusion–reaction (ADR) type

$$\begin{cases} \dfrac{\partial c}{\partial t} = \operatorname{div}(D(x,t)\nabla c) - \operatorname{div}(c\vec{v}(x,t)) + f(c,x,t), & x \in \Omega, \quad t > 0, \\ c(x,t) = g_{\mathscr{D}}(x,t), & x \in \Gamma_{\mathscr{D}}, \quad t > 0, \\ \langle D\nabla c(x,t), \nu\rangle = g_{\mathscr{N}}(x,t), & x \in \Gamma_{\mathscr{N}}, \quad t > 0, \\ c(x,0) = u_0(x), & x \in \Omega, \end{cases} \tag{1}$$

with mixed (possibly time dependent) Dirichlet and Neumann boundary conditions on $\Gamma_{\mathscr{D}} \cup \Gamma_{\mathscr{N}} = \partial\Omega$, $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$. In (1), $D(x,t)$ is the $d \times d$ (cross-) diffusion matrix, $\vec{v}(x,t) \in \mathbb{R}^d$ the velocity field, and f resembles reaction as well as source and sinks terms; see, e.g., [14].

☆ Work supported by the MIUR PRIN 2003 project "Dynamical systems on matrix manifolds: numerical methods and applications" (co-ordinator L. Lopez, Bari), and by the GNCS-INdAM.
* Corresponding author. Tel.: +39 49 827 1370; fax: +39 49 827 1499.
E-mail address: marcov@math.unipd.it (M. Vianello).

0377-0427/$ - see front matter © 2006 Published by Elsevier B.V.
doi:10.1016/j.cam.2006.10.055

Finite difference (FD) or finite element (FE) spatial discretization of (1) produces a large-scale linear system of ODEs like

$$\begin{cases} P\dot{z} = H(t)z + \boldsymbol{f}(z,t) + \boldsymbol{q}(t), & t > 0, \\ z(0) = \boldsymbol{u}_0, \end{cases} \tag{2}$$

where P is the symmetric positive-definite mass matrix (the identity matrix in the case of FD) and $H(t)$ the (nonsymmetric) stiffness matrix of dimension $N \times N$, $z=[z_1(t), \ldots, z_N(t)]^{\mathrm{T}}$, the vector field $\boldsymbol{f}(z,t)$ is generated by semidiscretization of $f(c,x,t)$, the vector $\boldsymbol{q}(t)$ by the Neumann boundary conditions, and $\boldsymbol{u}_0 = [u_0(x_1), \ldots, u_0(x_N)]^{\mathrm{T}}$, $\{x_j\}$ being the nodes (internal plus boundary) of the mesh or of the grid. Inclusion of boundary nodes is unusual with FD, but allows a unified treatment, which is particularly convenient in the framework of exponential integrators. As it is known, standard Galerkin FE discretization or second-order central FD make sense numerically only on sufficiently fine grids (small Peclét numbers). Otherwise, special stabilization techniques should be adopted, like Petrov–Galerkin upwinding for FE or third-order upwind biased differences for FD; see [14] and references therein.

System (2) can be rewritten in the form $\dot{z} = P^{-1}[H(t)z + \boldsymbol{f}(z,t) + \boldsymbol{q}(t)]$, which is suitable for the application of exponential integrators (cf. [8,21,11]). Clearly, availability of the mass matrix P^{-1} is a computationally expensive task in FE instances. However, we can apply the well-known mass-lumping technique (sum on the diagonal of all the row elements) to P in order to obtain a diagonal matrix P_{L} (and, immediately, P_{L}^{-1}); cf. [14]. In order to impose the lacking Dirichlet boundary conditions, if $x_i \in \Gamma_{\mathscr{D}}$, we can perform the following steps: replace the ith row of P_{L}^{-1} with $\boldsymbol{e}_i$ (the ith canonical basis vector); make vanishing the ith row of $H(t)$; replace $f_i(z,t)$ with $\partial g_{\mathscr{D}}(x_i,t)/\partial t$; replace u_{0i} with $g_{\mathscr{D}}(x_i, 0)$. In practice, the system above is modified into

$$\begin{cases} \dot{\boldsymbol{c}} = \boldsymbol{\phi}(\boldsymbol{c},t) = \widehat{P_{\mathrm{L}}^{-1}}[\widehat{H}(t)\boldsymbol{c} + \widehat{\boldsymbol{f}}(\boldsymbol{c},t) + \boldsymbol{q}(t)], & t > 0, \\ \boldsymbol{c}(0) = \widehat{\boldsymbol{u}}_0. \end{cases} \tag{3}$$

The semilinear system of ODEs (3) is now the discrete approximation of the PDE (1), where the Dirichlet boundary condition have been artificially imposed also to the initial data.

2. The exponential Euler–Midpoint integrator

We consider the following second-order exponential time-marching scheme for the semilinear system of ODEs (3):

$$\begin{cases} \boldsymbol{c}_{k+1} = \boldsymbol{c}_k + \Delta t_k \varphi(\Delta t_k J(\boldsymbol{c}_k, t_{k+1/2}))\boldsymbol{\phi}(\boldsymbol{c}_k, t_{k+1/2}), & k = 0, 1, 2, \ldots, \\ \boldsymbol{c}_0 = \widehat{\boldsymbol{u}_0}, \end{cases} \tag{4}$$

where $\varphi(z)$ is the entire function $\varphi(z) = (\mathrm{e}^z - 1)/z$ if $z \neq 0$, $\varphi(0) = 1$, and $J(\boldsymbol{c},t)$ is the Jacobian matrix defined by

$$J(\boldsymbol{c},t) = \partial_{\mathbf{c}}\boldsymbol{\phi}(\boldsymbol{c},t) = \widehat{P_{\mathrm{L}}^{-1}}[\widehat{H}(t) + \partial_{\mathbf{c}}\widehat{\boldsymbol{f}}(\boldsymbol{c},t)]. \tag{5}$$

We observe that (4) is obtained by coupling the exponentially fitted Euler integrator (see, e.g., [11]) with the exponential Midpoint integrator (see, e.g., [21,10,9]). Indeed, as in the construction of the exponentially fitted Euler method we first linearize (3) locally in time

$$\begin{cases} \dot{\boldsymbol{u}} = \boldsymbol{\phi}(\boldsymbol{c}_k, t) + \partial_{\mathbf{c}}\boldsymbol{\phi}(\boldsymbol{c}_k, t)(\boldsymbol{u} - \boldsymbol{c}_k), & t \in [t_k, t_{k+1}], \\ \boldsymbol{u}(t_k) = \boldsymbol{c}_k, \end{cases} \tag{6}$$

with a local error $\mathcal{O}((\Delta t_k)^3)$. Then, we apply to the nonautonomous linear system (6) the basic Magnus-like integrator, i.e., the Midpoint exponential rule [10], which again is locally of the third order. It is worth stressing that the resulting second-order exponential integrator (4) coincides with the exponential Midpoint rule in the linear case, and with the exponentially fitted Euler method in autonomous instances. In particular, it is an exact integrator in the linear autonomous case.

A possible difficulty with the scheme (4), is that it involves computation of Jacobians, differently from other recent exponential integrators for semilinear problems (cf., e.g., [7,12,22]). Nevertheless, in our applications this is not a real drawback, since we deal with single ADR equations, and with the discretizations adopted the Jacobian of $\widehat{\boldsymbol{f}}$ is a diagonal matrix, immediately computable when the derivative $\partial f/\partial c$ of the reaction is explicitly known (see below). In addition, using the Jacobian of the whole semilinear vector field $\boldsymbol{\phi}$, we have at hand an explicit exponential integrator which naturally works with stiff reaction terms, cf. [11]. For example, with FD (trivially), or with FE when using an approximate projection like $\langle f(c(x,t),x,t),\varphi_i\rangle \approx \sum_j f(c(x_j,t),x_j,t)\,\langle\varphi_i,\varphi_j\rangle$, where $\{\varphi_i\}$ are the FE trial and test functions, we have that $\boldsymbol{f}(\boldsymbol{c},t) = P\{f(c_j,x_j,t)\}_j$, where $P=\{\langle\varphi_i,\varphi_j\rangle\}$ is the symmetric mass matrix. Thus,

$$P^{-1}\partial_{\mathbf{c}}\boldsymbol{f}(\boldsymbol{c},t) = \partial_{\mathbf{c}}P^{-1}\boldsymbol{f}(\boldsymbol{c},t) = \operatorname{diag}\left\{\frac{\partial f}{\partial c}(c_j,x_j,t)\right\}_j, \tag{7}$$

with the easy adaptation that $\widehat{P_{\mathrm{L}}^{-1}}\partial_{\mathbf{c}}\widehat{\boldsymbol{f}}(\boldsymbol{c},t)$ is the same diagonal matrix, with zero entries on the diagonal correspondingly to Dirichlet boundary nodes.

An important comment on the construction of $\widehat{\boldsymbol{f}}$ at boundary nodes has to be made. Instead of using the definition $\widehat{f}_i(\boldsymbol{c},t) = \partial g_{\mathscr{D}}(x_i,t)/\partial t$, x_i being a Dirichlet boundary node, we can impose directly the exact time evolution of the boundary conditions, by setting $\widehat{f}_i(\boldsymbol{c},t) = (g_{\mathscr{D}}(x_i,t_{k+1}) - g_{\mathscr{D}}(x_i,t_k))/\Delta t_k$. The latter is the second-order central difference of $g_{\mathscr{D}}(x_i,t)$ in $t=t_{k+1/2}$. In fact, by recursion, the ith component of scheme (4) becomes

$$(\boldsymbol{c}_{k+1})_i = g_{\mathscr{D}}(x_i,t_k) + \Delta t_k\frac{g_{\mathscr{D}}(x_i,t_{k+1}) - g_{\mathscr{D}}(x_i,t_k)}{\Delta t_k} = g_{\mathscr{D}}(x_i,t_{k+1}),$$

since the ith row of J is the zero vector, and the ith row of $\varphi(\Delta t_k J)$ is $\boldsymbol{e}_i$ (φ being analytical).

3. The real Leja points method (ReLPM)

The practical application of (4) rests on the possibility of approximating efficiently the exponential propagator $\varphi(\Delta t J)\boldsymbol{v}$, where J is a Jacobian matrix as in (5), and $\boldsymbol{v}\in\mathbb{R}^N$. To this aim, two classes of polynomial methods are currently used. We have Krylov-like methods, which are based on the idea of projecting the propagator on a "small" Krylov subspace of the matrix via the Arnoldi process, and typically involve long-term recurrences in the nonsymmetric case; see, e.g., [8,11]. The second class consists of methods based on polynomial interpolation or series expansion of the entire function φ on a suitable compact subset containing the spectrum (or in general the field of values) of the matrix. They typically require some preliminary estimate of the underlying spectral structure, but, despite of this preprocessing stage, this second class of methods turned out to be competitive with polynomial Krylov-based approaches, especially on very large nonsymmetric matrices, cf., e.g., [4,3,17,18].

In this work we adopt the Real Leja Points Method (ReLPM), recently proposed in the framework of FD spatial discretization of advection–diffusion equations [6], and extended to FE in [5]. For the reader's convenience, we briefly recall the definition and properties of Leja points.

Sequences of Leja points $\{\xi_j\}_{j=0}^{\infty}$ for the compact K are defined recursively as follows [16]: if ξ_0 is an arbitrary fixed point in K (usually such as $|\xi_0| = \max_{\xi\in K}|\xi|$, cf. [20]), the ξ_j are chosen in such a way that

$$\prod_{k=0}^{j-1}|\xi_j-\xi_k| = \max_{\xi\in K}\prod_{k=0}^{j-1}|\xi-\xi_k|, \quad j=1,2,\ldots. \tag{8}$$

By the maximum principle, Leja points for K lie on ∂K; an efficient algorithm for computing a sequence of Leja points, the so-called "Fast Leja points", has been proposed in [1].

Given a matrix A and a vector $\boldsymbol{v}$, the ReLPM approximates the exponential propagator as

$$\varphi(A)\boldsymbol{v} \approx p_m(A)\boldsymbol{v}, \quad p_m(A) = \sum_{j=0}^{m} d_j\prod_{k=0}^{j-1}(A-\xi_k I), \quad m=1,2,\ldots, \tag{9}$$

with $p_m(z)$ Newton interpolating polynomial of $\varphi(z)$ at a sequence of Leja points $\{\xi_k\}$, in a compact subset of the complex plane containing the spectrum (or the field of values) of the matrix A. As it is well known, Leja points give a stable interpolant [20], superlinearly convergent to entire matrix functions due to the analogous scalar property

Table 1
Algorithm ReLPM (real Leja points method)

```
• INPUT: $J$, $\boldsymbol{v}$, $\Delta t$, tol
• Estimate the "spectral" focal interval $[\alpha, \beta]$ for $\Delta t J$ by Gershgorin's theorem, and compute a sequence of fast Leja points $\{\xi_j\}$ in $[\alpha, \beta]$ as in [1]
• $d_0 := \varphi(\xi_0)$, $\boldsymbol{w}_0 := \boldsymbol{v}$, $\boldsymbol{p}_0 := d_0\boldsymbol{w}_0$, $m := 0$
• WHILE $e_m^{\text{Leja}} := |d_m| \cdot \|\boldsymbol{w}_m\|_2 > \text{tol}$
    ◦ $\boldsymbol{z} := J\boldsymbol{w}_m$, $\boldsymbol{w}_{m+1} := \Delta t\boldsymbol{z} - \xi_m\boldsymbol{w}_m$
    ◦ $m := m + 1$; compute the next divided difference $d_m = \varphi[\xi_0, \ldots, \xi_m]$
    ◦ $\boldsymbol{p}_m := \boldsymbol{p}_{m-1} + d_m\boldsymbol{w}_m$
• OUTPUT: the vector $\boldsymbol{p}_m$ : $\|\boldsymbol{p}_m - \varphi(\Delta t J)\boldsymbol{v}\|_2 \approx e_m^{\text{Leja}} \leqslant \text{tol}$
```

[24,20,6], i.e.,

$$\limsup_{m\to\infty} \|f(A)\boldsymbol{v} - p_m(A)\boldsymbol{v}\|_2^{1/m} = 0. \tag{10}$$

Following [6,5], we can simply interpolate φ at a sequence of Leja points on the focal interval, say $[\alpha, \beta]$, of a suitable ellipse containing the spectrum of $A = \Delta t J$. From complex approximation theory [24], we get maximal and thus superlinear convergence on such an ellipse. This entails that the corresponding sequence of matrix polynomial operators converges superlinearly to $\varphi(\Delta t J)\boldsymbol{v}$, and that we can work with real arithmetic. Clearly, a key step in this procedure is given by estimating at low cost the reference focal interval for the spectrum of $\Delta t J$. As in [21,6,5], we adopt the simplest estimate given directly by Gershgorin's theorem [23].

The algorithm for real Leja points interpolation (ReLPM) of the advection–diffusion propagator $\varphi(\Delta t J)\boldsymbol{v}$ is sketched in Table 1. In practice, the implementation is slightly more refined. A problem is that convergence may not take place when the estimated focal interval is too large, due to the effect of significant cancellation errors in the computation of the divided differences. In such cases, it is necessary to subdivide a posteriori the time step. For this and other nontrivial implementation details we refer to [6].

The ReLPM algorithm turns out to be quite simple and efficient. Indeed, being based on two-term vector recurrences in real arithmetic, its storage occupancy and computational cost are very small, and it results more efficient than Krylov method on large-scale problems [3]. In addition, ReLPM is very well structured for a parallel implementation, as it has been shown in [2].

4. Numerical examples

The main purpose of this paper is to make a first test of the performance of the ReLPM polynomial approximation method, in the context of exponential integrators for large-scale systems of ODEs generated by spatial discretization of multidimensional semilinear parabolic PDEs. To this aim, albeit in the recent literature several high order exponential integrators have been proposed (cf., e.g., [7,11,12,15]), we have chosen the exponential Euler–Midpoint integrator, one of the simplest when dealing with single semidiscretized ADR equations where the diagonal Jacobian generated by the reaction term is analytically known (cf. (7)). Among second-order exponential integrators, Euler–Midpoint has a low computational cost, since other methods as [7,12,22] require computing at least three matrix exponential-like functions, and it is stable even with stiff reaction terms (see [11]).

In order to test the efficiency of the Leja–Euler–Midpoint (LEM) exponential integrator, we compared it with a classical second-order method like Crank–Nicolson (CN). Although CN might not be considered the best choice for time integration of ADR problems, it is an efficient and robust method still widely used on stiff large-scale problems in engineering applications, and a sound baseline benchmark for an advection–diffusion solver (cf. [19]). In the case of the relevant ODEs system (2) it writes as

$$\begin{aligned}&\left(P - \frac{\Delta t_k}{2}H(t_{k+1})\right)\boldsymbol{u}_{k+1} - \frac{\Delta t_k}{2}\boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1})\\ &\quad = \left(P + \frac{\Delta t_k}{2}H(t_k)\right)\boldsymbol{u}_k + \frac{\Delta t_k}{2}\boldsymbol{f}(\boldsymbol{u}_k, t_k) + \frac{\Delta t_k}{2}(\boldsymbol{q}(t_{k+1}) + \boldsymbol{q}(t_k)), \quad k = 0, 1, 2\ldots,\end{aligned} \tag{11}$$

where the nonlinear vector field ***f*** is constructed as in (7). In our applications we solved the nonlinear system (11) by Newton method, with BiCGStab preconditioned by ILU(0) as internal linear solver (imposing the Dirichlet boundary conditions at this stage).

4.1. Example 1: FD discretization of a 2D semilinear problem with a travelling wave solution

We now discuss in detail the autonomous advective Fisher equation (cf. [13])

$$\frac{\partial c}{\partial t} = \varepsilon\nabla^2 c - \operatorname{div}(c\vec{v}) + \gamma c^2(1-c), \quad x=(x_1,x_2)\in\Omega=(0,1)^2, \quad t>0, \tag{12}$$

where $\vec{v}=(\alpha,\alpha)$, $\alpha=-1$. Initial and Dirichlet (time-dependent) boundary conditions are prescribed so as to correspond to the exact "travelling wave" solution $c(x_1,x_2,t)=(1+\exp(a(x_1+x_2-bt)+p))^{-1}$, with $a=\sqrt{\gamma/4\varepsilon}$, $b=2\alpha+\sqrt{\gamma\varepsilon}$ and $p=a(b-1)$. We chose, as in [13], $\varepsilon=0.001$ and $\gamma=100$, which give solutions with a steep gradient. Nonoscillatory spatial discretization is obtained by third-order upwind-biased differences for the advection (cf. [14]), and by standard second-order central differences for the diffusion, on a uniform 160×160 grid.

In Table 2 we compare the CN and LEM integrators (Fortran77 codes) for Eq. (12) at a sequence of decreasing time stepsizes, $\Delta t=\Delta x$, $\Delta x/2$, $\Delta x/4$, $\Delta x/8$, on the time window [0, 1]. The exit tolerances of ReLPM inside LEM, and of Newton method inside CN have been set to $(\Delta x)^2/4$, while that of BiCGStab inside Newton is 10 times smaller.

First, we observe the L_2-errors with respect to the exact solution at final time are very close, as expected dealing with two second-order methods. Moreover, decreasing Δt, they tend to stabilize around the size of the underlying spatial discretization error. It is curious that linear iterations of Crank–Nicolson at the smaller time stepsizes are less than nonlinear iterations, that is the BiCGStab sometimes exits on the initial guess without iterating. This phenomenon is due to the fact that up to a certain time the wave front (see Fig. 1) has not developed, and the solution is practically constant. Observe also that Crank–Nicolson performs about one linear iteration per nonlinear iteration (from 1.4 for $\Delta t=\Delta x$ to 0.8 for $\Delta t=\Delta x/8$).

On the other hand, the number of iterations per time step of ReLPM decreases with Δt, as expected from the reduction of the capacity of the ellipse estimating the spectrum of $\Delta t\,J$ (see the theoretical convergence as estimated in [6]). In Fig. 2 we illustrate how the spectrum of the Jacobian matrix in (5) on a smaller 40×40 grid, computed with the `dgeev` LAPACK subroutine, is deformed with respect to the pure advection–diffusion case ($\gamma=0$) and evolves in time.

In Table 2 we also report the overall CPU times in seconds on an Alpha Station 600, and the corresponding speed-ups. It is worth stressing that the LEM integrator is around 5 times faster than CN, despite of the fact that it performs a larger average number of linear iterations per time step (ReLPM vs. BiCGStab). This is not surprising, even if we have used optimized linear algebra subroutines (DXML library), taking into account that the preconditioned BiCGStab performs the equivalent of four matrix–vector products per iteration plus several vector operations, and many memory allocations for auxiliary vectors. On the contrary, ReLPM performs only one matrix–vector product and two DAXPYs per iteration, and allocates only four auxiliary vectors.

Table 2
Comparison of the CN and LEM integrators for Eq. (12) at different time stepsizes

Δt	CN				LEM			SU
	Newt	BiCGS	CPU(s)	$E_{t=1}$	ReLPM	CPU(s)	$E_{t=1}$	
Δx	2.8	4.0	103.5	8E − 2	12.0	19.8	8E − 2	5.2
$\Delta x/2$	2.2	2.2	150.3	3E − 2	9.6	32.1	3E − 2	4.7
$\Delta x/4$	2.2	1.9	270.5	2E − 2	8.3	55.4	2E − 2	4.9
$\Delta x/8$	2.2	1.9	477.9	2E − 2	7.5	101.9	2E − 2	4.7

Newt: average Newton its. per time step, BiCGS: average BiCGStab its. per time step, $E_{t=1}$: L_2-error at the final time $t=1$, ReLPM: average ReLPM its. per time step, SU: speed-up = (CPU of CN)/(CPU of LEM).

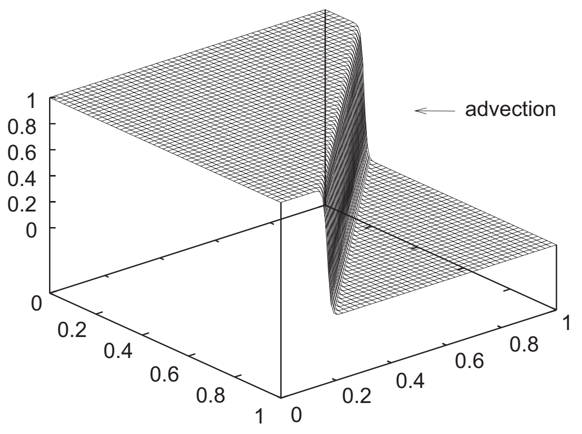

Fig. 1. Plot of the computed solution (travelling wave) at $t = 0.9$.

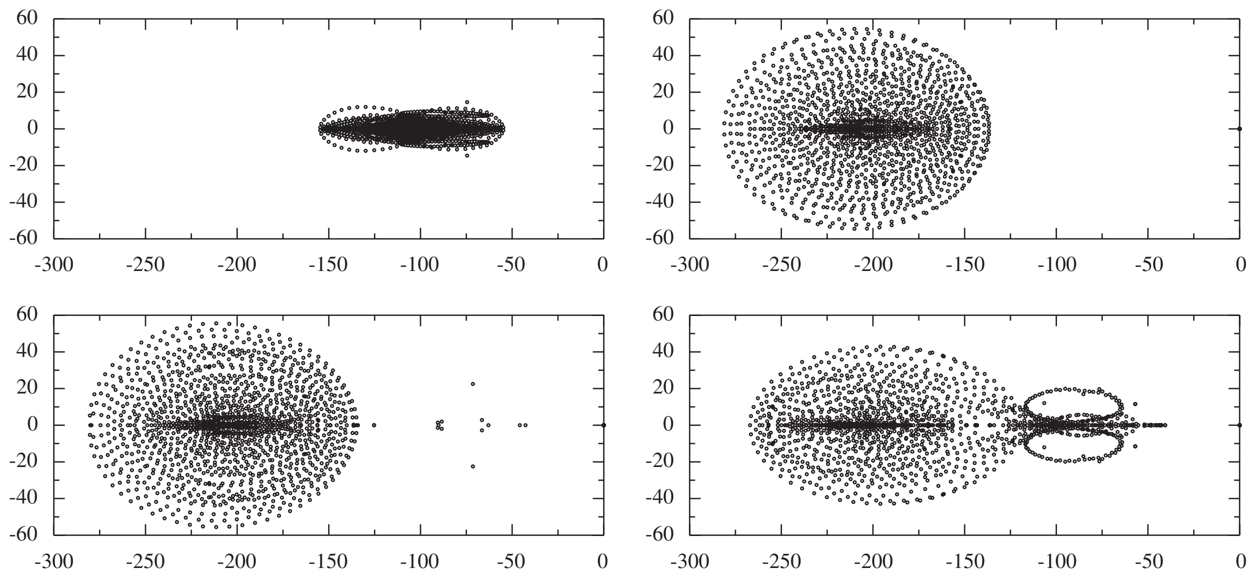

Fig. 2. Computed eigenvalues of the advection–diffusion–reaction Jacobian matrices (40×40 spatial grid) in the Euler–Midpoint exponential integrator applied to (12). First from the top: no reaction ($\gamma = 0$). Second, third and fourth: with reaction ($\gamma = 100$), at the times $t = 0$, $t = 0.5$, $t = 1$.

4.2. *Example 2: FE discretization of a 2D advection–dispersion–reaction model with linear reaction*

In this example we adopt the standard Galerkin FE discretization (linear basis functions) of a simplified linear model for transport of a reactive solute in groundwater flow (advection–dispersion–reaction, cf., e.g., [25])

$$\frac{\partial c}{\partial t} = \operatorname{div}(D\nabla c) - \operatorname{div}(c\vec{v}) - kc, \quad x \in \Omega, \quad t > 0, \tag{13}$$

with initial and Dirichlet ($\partial\Omega = \Gamma_{\mathscr{D}}$) boundary conditions as in (1). Here, c is the solute concentration, D the hydrodynamic dispersion tensor, $D_{ij} = \alpha_T|\vec{v}|\delta_{ij} + (\alpha_L - \alpha_T)v_i v_j/|\vec{v}|$, $1 \leqslant i, j \leqslant d$ (α_L and α_T being the longitudinal and the transverse dispersivity, respectively), $\vec{v}$ the average linear velocity of groundwater flow. We consider the case of a first-order reaction $f(c) = -kc$ with decay rate k. The domain Ω is the unit circle with a mesh consisting of $N = 35\,313$ nodes and $M = 70\,030$ triangular elements (maximum edges length $h = 0.0054$), constructed by a standard

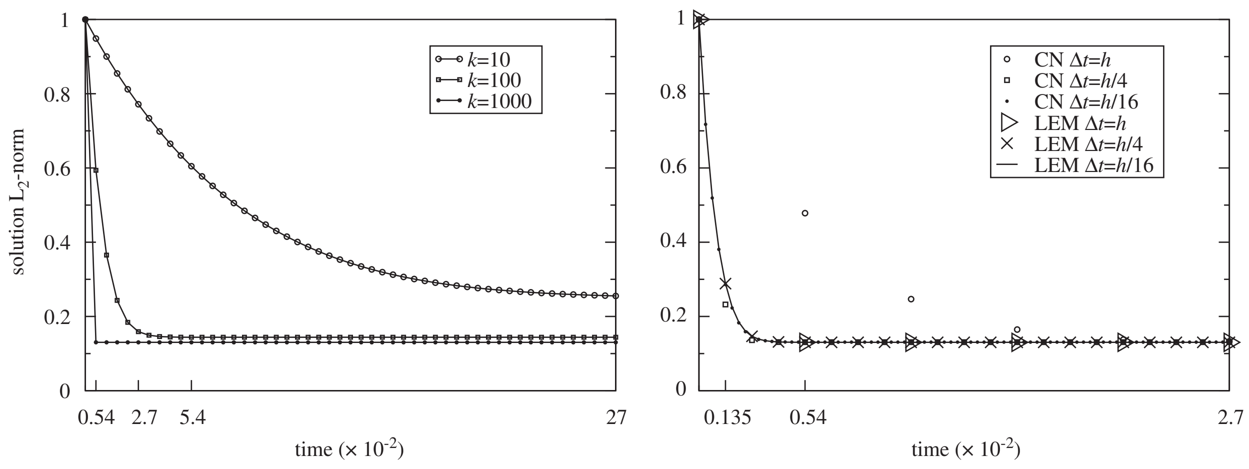

Fig. 3. Left: decay behavior of the solution for different reaction rates k. Right: solution L_2-norm with CN and LEM integrators at different time stepsizes, $k = 1000$.

Table 3
Comparison of the CN and LEM integrators for Eq. (13) with $k = 1000$ at different time stepsizes, up to steady state ($t = 2.7 \times 10^{-2}$)

Δt	CN		LEM		SU
	BiCGS	CPU(s)	ReLPM	CPU(s)	
h	1.0	2.5	14.4	1.4	1.8
$h/4$	1.0	9.0	8.9	9.1	1.0
$h/16$	1.0	36.2	7.1	16.9	2.1

BiCGS: average BiCGStab iterations per time step, ReLPM: average ReLPM iterations per time step, SU: speed-up = (CPU of CN)/(CPU of LEM).

Delaunay triangulator. Dirichlet boundary condition $g_{\mathscr{D}} \equiv 1$ are imposed on the whole boundary. The velocity is $\vec{v} = (v_1, v_2) = (1, 1)$, and $\alpha_L = \alpha_T = 0.00625$; moreover, $u_0 \equiv 1$.

We have considered three values of the reaction rate k with increasing stiffness (see Fig. 3, left). In Table 3, we compare the integrators for Eq. (13) with the stiffest value $k = 1000$, up to the final time $t = 0.027$ where the solution is at the steady state, at the decreasing time stepsizes $\Delta t = h, h/4, h/16$. The exit tolerances of ReLPM inside LEM, and of BiCGStab inside CN have been set to $h^2/4$.

Notice that, due to truncation errors, CN is as accurate as LEM only for the smallest time stepsize (see Fig. 3, right), where LEM performs more than twice faster. In addition, LEM is "exact" and can be already used for $\Delta t = h/4$ with no loss of accuracy with respect to $\Delta t = h/16$: in this case, it still provides an acceptable tracking of the transient state and is 4 times faster than CN.

References

[1] J. Baglama, D. Calvetti, L. Reichel, Fast Leja points, Electron. Trans. Numer. Anal. 7 (1998) 124–140.
[2] L. Bergamaschi, M. Caliari, A. Martínez, M. Vianello, A parallel exponential integrator for large-scale discretizations of advection–diffusion models, in: Proceedings of Euro PVM MPI 2005, Sorrento, Italy, Lecture Notes in Computer Science, vol. 3666, Springer, Berlin, pp. 483–492.
[3] L. Bergamaschi, M. Caliari, A. Martínez, M. Vianello, Comparing Leja and Krylov approximations of large scale matrix exponentials in: Proceedings of ICCS 2006, Reading, UK, Lecture Notes in Computer Science, vol. 3994, Springer, Berlin, pp. 685–692.
[4] L. Bergamaschi, M. Caliari, M. Vianello, Efficient approximation of the exponential operator for discrete 2D advection–diffusion problems, Numer. Linear Algebra Appl. 10 (2003) 271–289.
[5] L. Bergamaschi, M. Caliari, M. Vianello, The ReLPM exponential integrator for FE discretizations of advection–diffusion equations, in: Proceedings of ICCS 2004, Krakow, Lecture Notes in Computer Science, vol. 3039, Springer, Berlin, part IV, pp. 434–442.

[6] M. Caliari, M. Vianello, L. Bergamaschi, Interpolating discrete advection–diffusion propagators at Leja sequences, J. Comput. Appl. Math. 172 (2004) 79–99.

[7] S.M. Cox, P.C. Matthews, Exponential time differencing for stiff systems, J. Comput. Phys. 176 (2002) 430–455.

[8] E. Gallopoulos, Y. Saad, Efficient solution of parabolic equations by Krylov subspace methods, SIAM J. Sci. Statist. Comput. 13 (1992) 1236–1264.

[9] C. González, A. Ostermann, M. Thalhammer, A second-order Magnus type integrator for non-autonomous parabolic problems, J. Comput. Appl. Math. 189 (2006) 142–156.

[10] M. Hochbruck, C. Lubich, On Magnus integrators for time-dependent Schrödinger equations, SIAM J. Numer. Anal. 41 (2003) 945–963.

[11] M. Hochbruck, C. Lubich, H. Selhofer, Exponential integrators for large systems of differential equations, SIAM J. Sci. Comput. 19 (1998) 1552–1574.

[12] M. Hochbruck, A. Ostermann, Explicit exponential Runge–Kutta methods for semilinear parabolic problems, SIAM J. Numer. Anal. 43 (2005) 1069–1090.

[13] W. Hundsdorfer, Accuracy and stability of splitting with stabilizing corrections, Appl. Numer. Math. 42 (2002) 213–233.

[14] W. Hundsdorfer, J.G. Verwer, Numerical Solution of Time-Dependent Advection–Diffusion–Reaction equations, Springer Series in Computational Mathematics, vol. 33, Springer, Berlin, 2003.

[15] S. Krogstad, Generalized integrating factor methods for stiff PDEs, J. Comput. Phys. 203 (2005) 72–88.

[16] F. Leja, Sur certaines suites liées aux ensembles plas et leur application á la représentation conforme, Ann. Polon. Math. 4 (1957) 8–13.

[17] I. Moret, P. Novati, An interpolatory approximation of the matrix exponential based on Faber polynomials, J. Comput. Appl. Math. 131 (2001) 361–380.

[18] P. Novati, A polynomial method based on Feiér points for the computation of functions of unsymmetric matrices, Appl. Numer. Math. 44 (2003) 201–224.

[19] G. Pini, G. Gambolati, Arnoldi and Crank–Nicolson methods for integration in time of the transport equation, Internat. J. Numer. Methods Fluids 35 (2001) 25–38.

[20] L. Reichel, Newton interpolation at Leja points, BIT 30 (1990) 332–346.

[21] M.J. Schaefer, A polynomial based iterative method for linear parabolic equations, J. Comput. Appl. Math. 29 (1990) 35–50.

[22] M. Thalhammer, A second-order Magnus type integrator for non-autonomous semilinear parabolic problems, preprint, University of Innsbruck, 2005.

[23] R.S. Varga, Gershgorin and His Circles, Springer Series in Computational Mathematics, vol. 36, Springer, Berlin, 2004.

[24] J.L. Walsh, Interpolation and approximation by rational functions in the complex domain, AMS Colloquium Publications, vol. XX, American Mathematical Society, Providence, RI, 1935.

[25] G.T. Yeh, Computational Subsurface Hydrology, Kluwer, Dordrecht, 1999.