Implementation of exponential Rosenbrock-type methods

M. Caliari*  A. Ostermann*

3rd Austrian Numerical Analysis Day, TU Wien
April 26–27, 2007

*Universität Innsbruck, Institut für Mathematik
Exponential Rosenbrock-type methods: the idea

Let us consider

\[ u'(t) = f(u(t)) = Au(t) + g(u(t)), \quad u(t_0) = u_0 \]
Exponential Rosenbrock-type methods: the idea

Let us consider

\[ u'(t) = f(u(t)) = Au(t) + g(u(t)), \quad u(t_0) = u_0 \]

For the numerical approximation \( u_n \) of \( u(t_n) \) at time \( t_n \), first we linearise at each step

\[ u'(t) = J_n u(t) + g_n(u(t)), \quad t_n \leq t \leq t_{n+1} \]

where

\[ J_n = D_u f(u_n), \quad g_n(u(t)) = f(u(t)) - J_n u(t) \]
Let us consider

\[ u'(t) = f(u(t)) = Au(t) + g(u(t)), \quad u(t_0) = u_0 \]

For the numerical approximation \( u_n \) of \( u(t_n) \) at time \( t_n \), first we linearise at each step

\[ u'(t) = J_n u(t) + g_n(u(t)), \quad t_n \leq t \leq t_{n+1} \]

where

\[ J_n = D_u f(u_n), \quad g_n(u(t)) = f(u(t)) - J_n u(t) \]

and then we consider the variation-of-constants formula.
Exponential Rosenbrock–Euler method

\[ u(t_{n+1}) = \exp(h_n J_n) u_n + \int_0^{h_n} \exp((h_n - \tau) J_n) g_n(u(t_n + \tau)) \, d\tau \]
Exponential Rosenbrock–Euler method

\[
u(t_{n+1}) = \exp(h_n J_n) u_n + \int_0^{h_n} \exp((h_n - \tau)J_n) g_n(u(t_n + \tau)) d\tau\]

and with \(g_n(u(t_n + \tau)) \approx g_n(u_n)\) we get the exponential Rosenbrock–Euler method

\[
u_{n+1} = \exp(h_n J_n) u_n + h_n \varphi_1(h_n J_n) g_n(u_n)
\]

where

\[
\varphi_1(h_n J_n) = \frac{1}{h_n} \int_0^{h_n} \exp((h_n - \tau)J_n) d\tau, \quad \varphi_1(z) = \frac{e^z - 1}{z}
\]
More generally, we consider the \textit{s-stage} exponential method

\begin{align*}
U_{ni} &= \exp(c_i h_n J_n) u_n + h_n \sum_{j=1}^{i-1} a_{ij}(h_n J_n) g_n(U_{nj}), \\
u_{n+1} &= \exp(h_n J_n) u_n + h_n \sum_{i=1}^{s} b_i(h_n J_n) g_n(U_{ni}).
\end{align*}

where $b_i$ and $a_{ij}$ are linear combinations of $\varphi_k(h_n J_n)$,

\begin{align*}
\varphi_k(h_n J_n) &= \frac{1}{h_n^k} \int_0^{h_n} \exp((h_n - \tau) J_n) \frac{\tau^{k-1}}{(k-1)!} d\tau.
\end{align*}
Methods of order 2, 3 and 4

It is easy to construct a 2-stage third order method (with a second order error estimator embedded) and a 3-stage fourth order method (with a third order error estimator embedded).
Methods of order 2, 3 and 4

It is easy to construct a 2-stage third order method (with a second order error estimator embedded) and a 3-stage fourth order method (with a third order error estimator embedded).

The exponential Rosenbrock–Euler method

\[ u_{n+1} = \exp(h_n J_n) u_n + h_n \varphi_1(h_n J_n) g_n(u_n) \]

is computationally attractive since it achieves second order with one stage only. With the additional \( \varphi_1(h_n J_n) g_n(u_{n+1}) \) evaluation, it is possible to have a third order error estimator.
In the applications, these methods need the **efficient evaluation** of the underlying matrix functions \( \varphi_k(hJ)v \) (\( J \in \mathbb{R}^{N \times N}, \ v \in \mathbb{R}^{N} \)).
In the applications, these methods need the **efficient evaluation** of the underlying matrix functions $\varphi_k(hJ)v$ ($J \in \mathbb{R}^{N \times N}$, $v \in \mathbb{R}^N$).

Most authors regard **Krylov-like** as the methods of choice.
In the applications, these methods need the efficient evaluation of the underlying matrix functions $\varphi_k(hJ)v$ ($J \in \mathbb{R}^{N \times N}$, $v \in \mathbb{R}^N$).

Most authors regard Krylov-like as the methods of choice.

An alternative class of polynomial methods is based on direct interpolation or approximation of the $\varphi_k$ functions on the spectrum (or the field of values) of the relevant matrix.
Among others, the ReLPM has shown very attractive computational features:
Among others, the ReLPM has shown very attractive computational features:

- it rests on Newton interpolation of the exponential functions at a sequence of real Leja points;
Among others, the ReLPM has shown very attractive computational features:

- it rests on Newton interpolation of the exponential functions at a sequence of real Leja points;
- they guarantee maximal convergence of the interpolant and thus superlinear convergence of the corresponding matrix polynomials;
Among others, the ReLPM has shown very attractive computational features:

- it rests on Newton interpolation of the exponential functions at a sequence of real Leja points;
- they guarantee maximal convergence of the interpolant and thus superlinear convergence of the corresponding matrix polynomials;
- with respect to other sets of interpolation points, (Chebyshev, e.g.), the Leja points allow to increase the degree of interpolation just by adding new nodes of the same sequence.
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on \([-2, 2]\)
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on \([-2, 2]\)
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on \([-2, 2]\)
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
The first 13 Leja points on $[-2, 2]$
A key step is a cheap estimate of a real focal interval \([a, b]\) such that the minimal ellipse of the confocal family which contains the spectrum has a small capacity (half sum of the semi-axes).
Estimating the spectrum: Gershgorin’s circles

A key step is a cheap estimate of a real focal interval \([a, b]\) such that the minimal ellipse of the confocal family which contains the spectrum has a small capacity (half sum of the semi-axes).

The numerical experience with matrices arising from stable spatial discretizations of parabolic equations has shown that good results can be obtained simply by intersecting the Gershgorin’s circles of the matrix with the real axis.
Kernel of ReLPM

The kernel of the ReLPM code is given by the interpolation of \( \varphi_k(hz) \) at Leja points of the real focal interval \([a, b] = [c - 2\gamma, c + 2\gamma] \).
The kernel of the ReLPM code is given by the interpolation of \( \varphi_k(hz) \) at Leja points of the real focal interval 
\[ [a, b] = [c - 2\gamma, c + 2\gamma]. \]

In practice, it is numerically convenient to interpolate the function 
\( \varphi_k(h(c + \gamma\xi)) \) at Leja points \( \{\xi_s\} \) of the reference interval \([-2, 2]\).
Kernel of ReLPM

The kernel of the ReLPM code is given by the interpolation of \( \varphi_k(hz) \) at Leja points of the real focal interval 
\([a, b] = [c - 2\gamma, c + 2\gamma]\).

In practice, it is numerically convenient to interpolate the function 
\( \varphi_k(h(c + \gamma \xi)) \) at Leja points \( \{\xi_s\} \) of the reference interval \([-2, 2]\).

Then, given the corresponding divided differences \( \{d_i\} \) for such a function, the matrix Newton polynomial of degree \( m \) is

\[
\varphi_k(hJ)v \approx p_m(J)v = \sum_{i=0}^{m} d_i \Omega_i, \quad \Omega_i = \left( (J - cl)/\gamma - \xi_{i-1}l \right) \Omega_{i-1}
\]

\[
\Omega_0 = v
\]
ReLPM: key features

- real arithmetic;
ReLPM: key features

- real arithmetic;
- no Krylov subspace to store;
ReLPM: key features

- real arithmetic;
- no Krylov subspace to store;
- linear complexity due to the 2-term recurrence;
ReLPM: key features

- real arithmetic;
- no Krylov subspace to store;
- linear complexity due to the 2-term recurrence;
- no linear system to solve;
ReLPM: key features

- real arithmetic;
- no Krylov subspace to store;
- linear complexity due to the 2-term recurrence;
- no linear system to solve;
- well structured for a parallel implementation.
Time sub-steps

In general, it is not feasible to interpolate with the original time step $h$, which has to be split. This happens, e.g., when the expected degree for convergence is too large.
Time sub-steps

In general, it is not feasible to interpolate with the original time step $h$, which has to be split. This happens, e.g., when the expected degree for convergence is too large.

Setting $K = hJ$, $\varphi_k(K)v$ $(k > 0)$ is the solution at $t = 1$ of

$$y'(t) = Ky(t) + \frac{t^{k-1}}{(k-1)!}v, \quad y(0) = y_0 = 0$$
Time sub-steps

In general, it is not feasible to interpolate with the original time step $h$, which has to be split. This happens, e.g., when the expected degree for convergence is too large.

Setting $K = hJ$, $\varphi_k(K)v$ $(k > 0)$ is the solution at $t = 1$ of

$$y'(t) = Ky(t) + \frac{t^{k-1}}{(k-1)!}v, \quad y(0) = y_0 = 0$$

Given $\delta = 1/L$, then $y_L = \varphi_k(K)v$, where

$$y_{\ell+1} = \exp(\delta K)y_{\ell} + \delta^k \sum_{i=0}^{k-1} \frac{\ell^{k-1-i}}{(k-1-i)!} \varphi_{i+1}(\delta K)v, \quad \ell = 0, \ldots, L-1.$$
Time sub-steps

In general, it is not feasible to interpolate with the original time step \( h \), which has to be split. This happens, e.g., when the expected degree for convergence is too large.

Setting \( K = hJ \), \( \varphi_k(K)v \) (\( k > 0 \)) is the solution at \( t = 1 \) of

\[
y'(t) = Ky(t) + \frac{t^{k-1}}{(k-1)!}v, \quad y(0) = y_0 = 0
\]

Given \( \delta = 1/L \), then \( y_L = \varphi_k(K)v \), where

\[
y_{\ell+1} = \exp(\delta K)y_{\ell} + \delta^k \sum_{i=0}^{k-1} \frac{\ell^{k-1-i}}{(k-1-i)!} \varphi_{i+1}(\delta K)v, \quad \ell = 0, \ldots, L-1.
\]

It requires \( k \) evaluations of a matrix function for \( \ell = 0 \). When increasing \( \ell \) by one, only one additional evaluation, \( \exp(\delta K)y_\ell \), is required. The other terms are scalar multiples of the previous evaluations, which have to be stored.
Diffusion-advection-reaction equations

We consider

\[ \partial_t u = \varepsilon (\partial_{xx} u + \partial_{yy} u) - \alpha (\partial_x u + \partial_y u) + \gamma u \left( u - \frac{1}{2} \right) (1 - u) \]

on the unit square subject to homogeneous Neumann b. c.
Diffusion-advection-reaction equations

We consider

$$\partial_t u = \varepsilon (\partial_{xx} u + \partial_{yy} u) - \alpha (\partial_x u + \partial_y u) + \gamma u \left( u - \frac{1}{2} \right) (1 - u)$$

on the unit square subject to homogeneous Neumann b. c.

Standard finite differences in space with $\Delta x = \Delta y = 0.005$ (d.o.f. 40401), final time $T = 0.3$. 
Time integrators

We consider 5 second order time integrators:
Time integrators

We consider 5 second order time integrators:

- **erow2l** exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by ReLPM;
Time integrators

We consider 5 second order time integrators:

- \texttt{erow2l}: exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by ReLPM;
- \texttt{erow2k}: exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by Krylov method (\texttt{phipro} routine by Y. Saad);
Time integrators

We consider 5 second order time integrators:

- **erow2l**: exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by ReLPM;
- **erow2k**: exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by Krylov method (\texttt{phipro} routine by Y. Saad);
- **rkc**: explicit Runge–Kutta–Chebyshev method (code by Sommeijer, Shampine and Verwer);
We consider 5 second order time integrators:

- **erow2l**: exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by ReLPM;
- **erow2k**: exponential Rosenbrock–Euler method with $\varphi_k$ evaluations by Krylov method (\texttt{phipro} routine by Y. Saad);
- **rkc**: explicit Runge–Kutta–Chebyshev method (code by Sommeijer, Shampine and Verwer);
- **rock2**: explicit Orthogonal Runge–Kutta–Chebyshev (code by Abdulle and Medovikov);
Time integrators

We consider 5 second order time integrators:

- **erow2l**: exponential Rosenbrock–Euler method with \( \varphi_k \) evaluations by ReLPM;
- **erow2k**: exponential Rosenbrock–Euler method with \( \varphi_k \) evaluations by Krylov method (\texttt{phipro} routine by Y. Saad);
- **rkc**: explicit Runge–Kutta–Chebyshev method (code by Sommeijer, Shampine and Verwer);
- **rock2**: explicit Orthogonal Runge–Kutta–Chebyshev (code by Abdulle and Medovikov);
- **cn**: classical Crank–Nicolson scheme (inexact Newton nonlinear solver, BiCGStab linear solver, ILU with no fill-in perconditioner).
Results: small advection, small reaction

\[ \varepsilon = 0.1, \ \alpha = -1, \ \gamma = 1 \] (tolerances: \(10^{-3} - 10^{-6}\))
Results: large reaction

\( \varepsilon = 0.1, \, \alpha = -1, \, \gamma = 100 \) (tolerances: \( 10^{-3} - 10^{-6} \))
Results: large advection

\[ \varepsilon = 0.1, \, \alpha = -10, \, \gamma = 1 \text{ (tolerances: } 10^{-3} - 10^{-6}) \]
Collocation with cardinal functions

Given the PDE

$$\partial_t u(t, x) = Lu(t, x)$$

(with $L$ a second order, linear operator), we consider the interpolation

$$u(t, x) \approx \sum_{j=1}^{N} \phi_j(x) u(t; x_j) = \phi(x) \circ u(t)$$

where $\phi_j(x_i) = \delta_{ij}$ and $u(t; x_j) = u(t, x_j)$. 
Collocation with cardinal functions

Given the PDE

$$\partial_t u(t, x) = Lu(t, x)$$

(with $L$ a second order, linear operator), we consider the interpolation

$$u(t, x) \approx \sum_{j=1}^{N} \phi_j(x) u(t; x_j) = \phi(x) \circ u(t)$$

where $\phi_j(x_i) = \delta_{ij}$ and $u(t; x_j) = u(t, x_j)$. Then,

$$\phi(x) \circ u'(t) = L (\phi(x) \circ u(t))$$
Given the PDE

$$\partial_t u(t, x) = Lu(t, x)$$

(with $L$ a second order, linear operator), we consider the interpolation

$$u(t, x) \approx \sum_{j=1}^{N} \phi_j(x) u(t; x_j) = \phi(x) \circ u(t)$$

where $\phi_j(x_i) = \delta_{ij}$ and $u(t; x_j) = u(t, x_j)$. Then,

$$\phi(x) \circ u'(t) = L (\phi(x) \circ u(t))$$

Now, we can collocate at the same $\{x_i\}$.
We get

\[ u'(t) = L(\phi(x) \circ u(t)) \big|_{(x_1,\ldots,x_N)^T} = (L(\phi)(x) \circ u(t)) \big|_{(x_1,\ldots,x_N)^T} = Au(t) \]

where \( A = a_{ij} = L(\phi_j)(x_i) \).

*In collaboration with M. Vianello, University of Padua*
Exponential meshless method*

We get

\[ u'(t) = L(\phi(x) \circ u(t)) \big|_{(x_1,\ldots,x_N)^T} = (L(\phi)(x) \circ u(t)) \big|_{(x_1,\ldots,x_N)^T} = Au(t) \]

where \( A = a_{ij} = L(\phi_j)(x_i) \).

An iterative method (such as ReLPM or Krylov method) for the approximation of \( \exp(h_nA)u_n \) is based on successive products

\[ Au_n^{(m)}, \quad m = 0, 1, \ldots, \quad u_n^{(0)} = u_n \]

*In collaboration with M. Vianello, University of Padua
Exponential meshless method*

We get

\[ u'(t) = L(\phi(x) \circ u(t)) |_{(x_1, \ldots, x_N)^T} = (L(\phi)(x) \circ u(t)) |_{(x_1, \ldots, x_N)^T} = Au(t) \]

where \( A = a_{ij} = L(\phi_j)(x_i) \).

An iterative method (such as ReLPM or Krylov method) for the approximation of \( \exp(h_nA)u_n \) is based on successive products

\[ Au_n^{(m)}, \quad m = 0, 1, \ldots, \quad u_n^{(0)} = u_n \]

If the nodes \( \{x_i\} \) are arbitrary and we have a tool to compute \( Au_n^{(m)} \) directly as \( L\left(\phi(x) \circ u_n^{(m)}\right) |_{(x_1, \ldots, x_N)^T} \), we get an exponential meshless method.

*In collaboration with M. Vianello, University of Padua
**2D examples: CShep2D and QShep2D**

**CShep2D**, by R. J. Renka, is among the most accurate and efficient scattered data interpolation algorithms. It constructs a $C^2$ interpolant in a moving least-square fashion. The costs range between $O(N)$ and $O(N^2)$ (preprocessing) and $O(1)$ and $O(N)$ (evaluation), depending on the distribution of the nodes. It can compute also the first and the second derivative. The required storage is $9N$. 
2D examples: CShep2D and QShep2D

CShep2D, by R. J. Renka, is among the most accurate and efficient scattered data interpolation algorithms. It constructs a $C^2$ interpolant in a moving least-square fashion. The costs range between $O(N)$ and $O(N^2)$ (preprocessing) and $O(1)$ and $O(N)$ (evaluation), depending on the distribution of the nodes. It can compute also the first and the second derivative. The required storage is $9N$.

QShep2D constructs a $C^1$ interpolant. It can compute also the first and the second derivative (although not continuous). The required storage is $5N$. 
A monitor for the check points

Since the collocation, or center, points are arbitrary, we can add or remove points at each time step, depending on the behaviour of the solution on a set of check points.
Since the collocation, or center, points are arbitrary, we can add or remove points at each time step, depending on the behaviour of the solution on a set of check points.

For example, we can consider a curvature monitor

$$|\Delta x^2 u_{xx}(t; (\tilde{x}_i, \tilde{y}_i))| + |\Delta y^2 u_{yy}(t; (\tilde{x}_i, \tilde{y}_i))|$$

which has to be evaluated at each check point $(\tilde{x}_i, \tilde{y}_i)$. 

Center and check points

- ● center point
- ✗ check point
Center and check points

- center point
- check point
Unstable derivatives

Derivatives computed directly by \{C,Q\}Shep2D give raise to instabilities during time integration (under investigation).
Unstable derivatives

Derivatives computed directly by \{C,Q\}Shep2D give raise to instabilities during time integration (under investigation).

Idea: derivatives in a **FD fashion**. They are much more stable.
The Molenkamp–Crowley test

Let us consider

$$\partial_t u + (au)_x + (bu)_y = 0$$

with

$$a(x, y) = -2\pi \left(y - \frac{1}{2}\right), \quad b(x, y) = 2\pi \left(x - \frac{1}{2}\right)$$
The Molenkamp–Crowley test

Let us consider

$$\partial_t u + (au)_x + (bu)_y = 0$$

with

$$a(x, y) = -2\pi \left( y - \frac{1}{2} \right), \quad b(x, y) = 2\pi \left( x - \frac{1}{2} \right)$$

The initial profile is rotated around the center of the domain. At time $t = 1$ one rotation will be completed. Homogeneous Dirichlet conditions are prescribed at the inflow boundaries.
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The Molenkamp–Crowley test
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$
The first symmetrized 13 Leja points on $i[-2, 2]$