Exponential Integrators for Semilinear Problems

University of Bergen

29 October 2004, Bergen.

Borislav V. Minchev

`Borko.Minchev@ii.uib.no`

`http://www.ii.uib.no/∼borko`

Department of computer science

University of Bergen, Norway

Outline

- Introduction and Motivation
- Main classes of exponential integrators
  - Exponential linear multistep methods
  - Exponentila Runge–Kutta methods
  - Exponential general linear methods
- Exponential integrators and Lie group methods
- Implementation issues
- Numerical experiments
- Conclusions
- Open problems

What are exponential integrators ?

What are exponential integrators ?

These are integrators which use the exponential and often functions which are closely related to the exponential function inside a numerical method.

Introduction and Motivation

First exponential integrators:

- Certain’60 - multistep type
- Lawson’69 - multistage type

Introduction and Motivation

First exponential integrators:

- Certain'60 - multistep type
- Lawson'69 - multistage type

A new interest in exponential integrators for semilinear problems

$$u' = Lu + N(u,t), \; u(t_0) = u_0, \tag{1}$$

where $u : \mathbb{R} \to \mathbb{R}^d, \; L \in \mathbb{R}^{d\times d}, N : \mathbb{R}^d \to \mathbb{R}^d$ and d is a discretization parameter equal to the number of spatial grid points.

Exponential multistep methods

- IF = Integrating Factor methods (Lawson)

Exponential multistep methods

- IF = Integrating Factor methods (Lawson)
- ETD = Exponential Time Differencing metods (Certain)

Exponential multistep methods

- IF = Integrating Factor methods (Lawson)

Recall

$$u' \;=\; Lu + N(u,t),\; u(t_0) = u_0. \tag{1}$$

Exponential multistep methods

- IF = Integrating Factor methods (Lawson)

Recall

$$u' = Lu + N(u,t),\ u(t_0) = u_0. \tag{1}$$

Solve exactly the linear part and then make a change of variables (also known as Lawson transformation)

$$v(t) = e^{-tL}u(t).$$

Exponential multistep methods

- IF = Integrating Factor methods (Lawson)

Recall

$$u' = Lu + N(u,t), \; u(t_0) = u_0. \tag{1}$$

Solve exactly the linear part and then make a change of variables
(also known as Lawson transformation)

$$v(t) = e^{-tL}u(t).$$

The initial value problem written in the new variable is then given by

$$v'(t) = e^{-tL}N(e^{tL}v(t), t) = \mathrm{g}(v,t), \qquad v(t_0) = v_0,$$

where $v_0 = e^{-t_0 L}u_0$.

Integrating Factor

Consider the Jacobian of the transformed equation

$$\frac{\partial \mathrm{g}}{\partial v} = e^{-tL}\,\frac{\partial N}{\partial u}\,e^{tL},$$

Since $e^{-tL} = (e^{tL})^{-1}$, it follows that the eigenvalues of $\partial \mathrm{g}/\partial v$ are those of $\partial N/\partial u$.

Integrating Factor

Consider the Jacobian of the transformed equation

$$\frac{\partial \mathrm{g}}{\partial v} = e^{-tL}\,\frac{\partial N}{\partial u}\,e^{tL},$$

Since $e^{-tL} = (e^{tL})^{-1}$, it follows that the eigenvalues of $\partial \mathrm{g}/\partial v$ are those of $\partial N/\partial u$. The idea now is to apply any numerical method on the transformed equation and then to transform back the result into the original variable.

Integrating Factor

Consider the Jacobian of the transformed equation

$$\frac{\partial \mathrm{g}}{\partial v} = e^{-tL}\,\frac{\partial N}{\partial u}\,e^{tL},$$

Since $e^{-tL} = (e^{tL})^{-1}$, it follows that the eigenvalues of $\partial \mathrm{g}/\partial v$ are those of $\partial N/\partial u$.
For example:
IF Euler method is

$$u_n = e^{hL}u_{n-1} + e^{hL}hN_{n-1},$$

where h represents the stepsize of the method and $N_{n-1} = N(u_{n-1}, t_{n-1})$.

Integrating Factor

Consider the Jacobian of the transformed equation

$$\frac{\partial \mathrm{g}}{\partial v} = e^{-tL}\,\frac{\partial N}{\partial u}\,e^{tL},$$

Since $e^{-tL} = (e^{tL})^{-1}$, it follows that the eigenvalues of $\partial \mathrm{g}/\partial v$ are those of $\partial N/\partial u$.
For example:
IF Euler method is

$$u_n = e^{hL}u_{n-1} + e^{hL}hN_{n-1},$$

where h represents the stepsize of the method and $N_{n-1} = N(u_{n-1}, t_{n-1})$.
IF implicit Euler method is

$$u_n = e^{hL}u_{n-1} + e^{hL}hN_n.$$

More IF multistep methods

Similarly k-step IF Adams methods are defined as

$$u_n = e^{hL} u_{n-1} + \sum_{i=0}^{k} \beta_i e^{ihL} h N_{n-i},$$

where β_i are the coefficients of the Adams method and $N_{n-i} = N(u_{n-i}, t_{n-i})$ for $i = 0, 1, 2, \ldots, k$.

More IF multistep methods

Similarly k-step IF Adams methods are defined as

$$u_n = e^{hL}u_{n-1} + \sum_{i=0}^{k} \beta_i e^{ihL} hN_{n-i},$$

where β_i are the coefficients of the Adams method and $N_{n-i} = N(u_{n-i}, t_{n-i})$ for $i = 0, 1, 2, \ldots, k$.
IF BDF methods are defined as

$$u_n = \sum_{i=1}^{k} \alpha_i e^{ihL} u_{n-i} + \beta_0 hN_n,$$

where β_0 and α_i are the coefficients of the underlying BDF method.

ETD multistep methods

Similar approach to the IF methods, but we do not make a complete change of variables. Premultiplying the original problem (1) by the integrating factor e^{-tL} we get

$$\begin{aligned} e^{-tL}u' &= e^{-tL}Lu + e^{-tL}N(u,t), \\ (e^{-tL}u)' &= e^{-tL}N(u,t). \end{aligned}$$

ETD multistep methods

Similar approach to the IF methods, but we do not make a complete change of variables. Premultiplying the original problem (1) by the integrating factor e^{-tL} we get

$$\begin{aligned} e^{-tL}u' &= e^{-tL}Lu + e^{-tL}N(u,t), \\ (e^{-tL}u)' &= e^{-tL}N(u,t). \end{aligned}$$

Integrating the last equation between t_{n-1} and $t_n = t_{n-1} + h$, we obtain

$$\text{(vcf)} \quad u(t_{n-1}+h) = e^{hL}u_{n-1} + \int_0^h e^{(h-\tau)L} N(u(t_{n-1}+\tau), t_{n-1}+\tau)\mathrm{d}\tau.$$

The approach now is to replace the nonlinear term in the variation of constants formulae by a Newton interpolation polynomial and then solve the resulting integral exactly.

ETD multistep methods

Similar approach to the IF methods, but we do not make a complete change of variables. Premultiplying the original problem (1) by the integrating factor e^{-tL} we get

$$\begin{aligned} e^{-tL}u' &= e^{-tL}Lu + e^{-tL}N(u,t), \\ (e^{-tL}u)' &= e^{-tL}N(u,t). \end{aligned}$$

Integrating the last equation between t_{n-1} and $t_n = t_{n-1} + h$, we obtain

$$\text{(vcf)} \quad u(t_{n-1}+h) = e^{hL}u_{n-1} + \int_0^h e^{(h-\tau)L}N(u(t_{n-1}+\tau), t_{n-1}+\tau)\mathrm{d}\tau.$$

The approach now is to replace the nonlinear term in the variation of constants formulae by a Newton interpolation polynomial and then solve the resulting integral exactly.

- When $N(u(t_{n-1}+\tau), t_{n-1}+\tau) \approx N_{n-1}$ we obtain the ETD Euler method

$$u_n = e^{hL}u_{n-1} + \phi^{[1]}(hL)hN_{n-1},$$

where $\phi^{[1]}(z)$ is

$$\phi^{[1]}(z) = \frac{e^z - 1}{z}.$$

ETD multistep methods

Similar approach to the IF methods, but we do not make a complete change of variables. Premultiplying the original problem (1) by the integrating factor e^{-tL} we get

$$\begin{aligned} e^{-tL}u' &= e^{-tL}Lu + e^{-tL}N(u,t), \\ (e^{-tL}u)' &= e^{-tL}N(u,t). \end{aligned}$$

Integrating the last equation between t_{n-1} and $t_n = t_{n-1} + h$, we obtain

$$\text{(vcf)} \quad u(t_{n-1}+h) = e^{hL}u_{n-1} + \int_0^h e^{(h-\tau)L}N(u(t_{n-1}+\tau), t_{n-1}+\tau)\mathrm{d}\tau.$$

The approach now is to replace the nonlinear term in the variation of constants formulae by a Newton interpolation polynomial and then solve the resulting integral exactly.

- When $N(u(t_{n-1}+\tau), t_{n-1}+\tau) \approx N_{n-1}$ we obtain the ETD Euler method

$$u_n = e^{hL}u_{n-1} + \phi^{[1]}(hL)hN_{n-1},$$

- In general, using higher order approximations to the nonlinear part N we obtain

  ETD Adams–Bashforth (Nørsett'69,..., Cox–Matthews'02)

  ETD Adams–Moulton (Verwer and Houwen'74,..., Beylkin et al.'98)

Exponential Runge–Kuta methods

IF Runge–Kuta methods (Lawson'69)

For simplicity, we represent the initial value problem (1) in autonomous form

$$u' = Lu + N(u(t)), \quad u(t_0) = u_0.$$

Similarly to the the multistep case, the idea now is to apply an arbitrary s-stage Runge–Kutta method to the transformed equation

$$v'(t) = e^{-tL} N(e^{tL} v(t)) = \mathrm{g}(v), \qquad v(t_0) = v_0,$$

and then to transform back the result into the original variable. If $\mathcal{A} = (\alpha_{ij})$, $b = (\beta_i)$ and $c = (c_i)$ are the coefficients of the underlying multistage method then in terms of the original variable the computations performed are

Exponential Runge–Kuta methods

IF Runge–Kuta methods (Lawson'69)

$$\begin{aligned}
U_1 &= u_0 \\
U_2 &= e^{c_2 hL}(u_0 + \alpha_{21} hN(U_1)) \\
U_3 &= e^{c_3 hL}(u_0 + \alpha_{31} hN(U_1) + \alpha_{32} he^{-c_2 hL} N(U_2)) \\
U_4 &= e^{c_4 hL}(u_0 + \alpha_{41} hN(U_1) + \alpha_{42} he^{-c_2 hL} N(U_2) \\
&\quad + \alpha_{43} he^{-c_3 hL} N(U_3)) \\
u_1 &= e^{hL}(u_0 + \beta_1 hN(U_1) + \beta_2 he^{-c_2 hL} N(U_2) \\
&\quad + \beta_3 he^{-c_3 hL} N(U_3) + \beta_4 he^{-c_4 hL} N(U_4))
\end{aligned}$$

Exponential Runge–Kuta methods

IF Runge–Kuta methods (Lawson'69)

General form of an order 4 integrating factor method is

$$\left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & e^{c_1 hL} \\ \alpha_{21} e^{c_2 hL} & 0 & 0 & 0 & e^{c_2 hL} \\ \alpha_{31} e^{c_3 hL} & \alpha_{32} e^{(c_3 - c2) hL} & 0 & 0 & e^{c_3 hL} \\ \alpha_{41} e^{c_4 hL} & \alpha_{42} e^{(c_3 - c2) hL} & \alpha_{43} e^{(c_4 - c3) hL} & 0 & e^{c_4 hL} \\ \hline \beta_1 e^{hL} & \beta_2 e^{(1-c_2) hL} & \beta_3 e^{(1-c_3) hL} & \beta_4 e^{(1-c_4) hL} & e^{hL} \end{array}\right]$$

- Uniformly distributed c vector provides cheapest methods.
- This structure requires only classical order conditions.
- IF RK methods perform poorly for stiff problems.

General format of Exp. RK methods

Aims:

- Construct a general class of exponential integrators which includes as special cases all known exponential Runge–Kutta methods
- Derive the nonstiff order theory for this class of method

The ϕ functions

- The IF ϕ functions are

$$\phi^{[i]}(c_j)(hL) = e^{(c_i - c_j)hL} \quad i = 1, 2, \ldots .$$

The ϕ functions

- The IF ϕ functions are

$$\phi^{[i]}(c_j)(hL) = e^{(c_i - c_j)hL} \quad i = 1, 2, \ldots .$$

- The ETD ϕ functions are

$$\begin{aligned} \phi^{[0]}(c_j)(hL) &= e^{c_j hL}, \\ \phi^{[i]}(c_j)(hL) &= \frac{\phi^{[i-1]}(c_j)(hL) - \frac{1}{(i-1)!}}{c_j hL} \end{aligned}$$

The ϕ functions

In general, for $l \in \mathbb{N}$ and $\lambda \in \mathbb{R}$, the $\phi^{[l]}$ functions could be

$$\phi^{[l]}(\lambda)(hL) = \sum_{j \geq 0} \phi_j^{[l]}(\lambda)(hL)^j,$$

The ϕ functions

In general, for $l \in \mathbb{N}$ and $\lambda \in \mathbb{R}$, the $\phi^{[l]}$ functions could be

$$\phi^{[l]}(\lambda)(hL) = \sum_{j\geq 0} \phi_j^{[l]}(\lambda)(hL)^j,$$

and must:

- Be computed exactly or to arbitrary high order cheaply
- Map the spectrum of hL to a bounded region

Given the IF and ETD ϕ functions as basis elements then
- linear combinations
- products
- inverses
produce mehods.

The ϕ functions

In general, for $l \in \mathbb{N}$ and $\lambda \in \mathbb{R}$, the $\phi^{[l]}$ functions could be

$$\phi^{[l]}(\lambda)(hL) = \sum_{j \geq 0} \phi_j^{[l]}(\lambda)(hL)^j,$$

and must:

- Be computed exactly or to arbitrary high order cheaply
- Map the spectrum of hL to a bounded region

Given the IF and ETD ϕ functions as basis elements then
- linear combinations
- products
- inverses
produce mehods.
Other choices are also posible (approximations with trigonometric polynomials in vcf).
The exact structure of $\phi^{[l]}$, which leads to methods is still unclear!

Formulation of the methods

The computations performed are

$$\begin{aligned} U_i &= \sum_{j=1}^{s}\sum_{l=1}^{m} \alpha_{ij}^{[l]} \phi^{[l]}(c_i)(hL) hN(U_j) + e^{c_i hL} u_{n-1}, \\ u_n &= \sum_{j=1}^{s}\sum_{l=1}^{m} \beta_{j}^{[l]} \phi^{[l]}(1)(hL) hN(U_j) + e^{hL} u_{n-1}, \end{aligned}$$

where m puts a limit on the number of $\phi^{[l]}$ functions which can be computed, h represents the stepsize and U_i denotes the internal stage approximation.

Formulation of the methods

The computations performed are

$$
\begin{aligned}
U_i &= \sum_{j=1}^{s}\sum_{l=1}^{m} \alpha_{ij}^{[l]} \phi^{[l]}(c_i)(hL) hN(U_j) + e^{c_i hL} u_{n-1}, \\
u_n &= \sum_{j=1}^{s}\sum_{l=1}^{m} \beta_{j}^{[l]} \phi^{[l]}(1)(hL) hN(U_j) + e^{hL} u_{n-1},
\end{aligned}
$$

where m puts a limit on the number of $\phi^{[l]}$ functions which can be computed, h represents the stepsize and U_i denotes the internal stage approximation.

Interpreted in a Runge–Kutta type tableau

$$
\begin{array}{c|c|c|c|c|c}
 & \phi^{[1]} & \phi^{[2]} & & \phi^{[m-1]} & \phi^{[m]} \\
c & \alpha^{[1]} & \alpha^{[2]} & \dots & \alpha^{[m-1]} & \alpha^{[m]} \\
\hline
 & \beta^{[1]^T} & \beta^{[2]^T} & \dots & \beta^{[m-1]^T} & \beta^{[m]^T}
\end{array}
$$

Nonstiff order conditions

Use rooted trees and B-series.
Represent the elementary differentials using trees:

- Associate a closed node with L and an open node with N
- 2T* - Bi-coloured rooted trees with one child closed nodes

Nonstiff order conditions

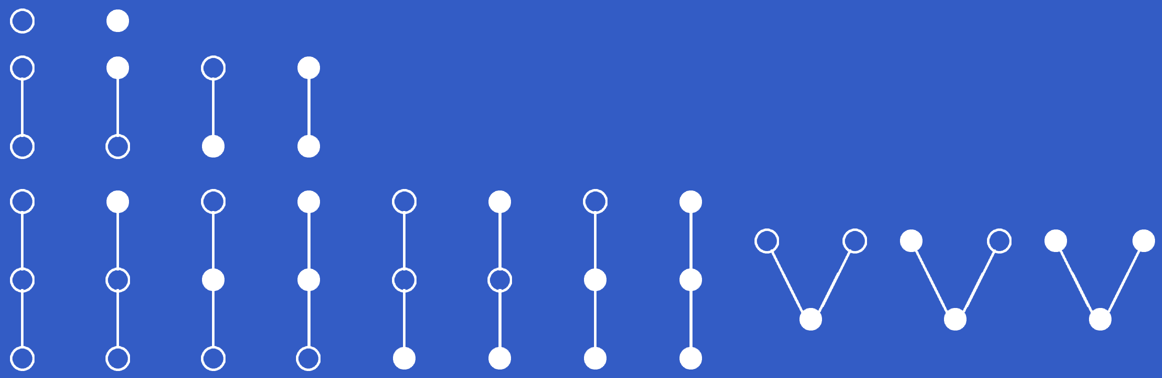
Use rooted trees and B-series.
Represent the elementary differentials using trees:

- Associate a closed node with L and an open node with N
- 2T* - Bi-coloured rooted trees with one child closed nodes

Order 1:

Order 2:

Order 3:



Nonstiff order conditions

Use rooted trees and B-series.
Represent the elementary differentials using trees:

- Associate a closed node with L and an open node with N
- 2T* - Bi-coloured rooted trees with one child closed nodes

n	1	2	3	4	5	6	7	8	9	10
θ_n	2	4	11	34	117	421	1589	6162	24507	99268
$\Theta = \sum_i^n \theta_n$	2	6	17	51	168	589	2178	8340	32847	132115

The number of rooted trees in $2\mathrm{T}^*$ for all orders up to ten.

Elementary differentials and B-series

The elementary differentials are recursively generated as

$$F(\tau)(u) = \begin{cases} LF(\tau_1)(u) & \text{if } \tau = [\circ; \tau_1] \\ N^{(\ell)}(u)\big(F(\tau_1)(u), \ldots, F(\tau_\ell)(u)\big) & \text{if } \tau = [\bullet; \tau_1, \ldots, \tau_\ell] \end{cases}$$

Elementary differentials and B-series

The elementary differentials are recursively generated as

$$F(\tau)(u) = \begin{cases} LF(\tau_1)(u) & \text{if } \tau = [\circ; \tau_1] \\ N^{(\ell)}(u)(F(\tau_1)(u), \dots, F(\tau_\ell)(u)) & \text{if } \tau = [\bullet; \tau_1, \dots, \tau_\ell] \end{cases}$$

For an elementary weight function $a : 2\mathrm{T}^* \to \mathbb{R}$ the B-series is

$$B(a,u) = a(\emptyset)u + \sum_{\tau \in 2\mathrm{T}^*} h^{|\tau|} \frac{a(\tau)}{\sigma(\tau)} F(\tau)(u)$$

Elementary differentials and B-series

The elementary differentials are recursively generated as

$$F(\tau)(u) = \begin{cases} LF(\tau_1)(u) & \text{if } \tau = [\circ; \tau_1] \\ N^{(\ell)}(u)(F(\tau_1)(u), \ldots, F(\tau_\ell)(u)) & \text{if } \tau = [\bullet; \tau_1, \ldots, \tau_\ell] \end{cases}$$

For an elementary weight function $a : 2\mathrm{T}^* \to \mathbb{R}$ the B-series is

$$B(a,u) = a(\emptyset)u + \sum_{\tau \in 2\mathrm{T}^*} h^{|\tau|} \frac{a(\tau)}{\sigma(\tau)} F(\tau)(u)$$

The elementary weight function for the exact solution is

$$a(\tau) = \frac{1}{\gamma(\tau)},$$

where γ is the density of single coloured tree

The lemmas

To obtain B-series expansions of the numerical solution we need three Lemmas.

The lemmas

To obtain B-series expansions of the numerical solution we need three Lemmas.

Lemma 2. *Let* $a : 2T^* \to \mathbb{R}$, *with* $a(\emptyset) = 1$, *then*

$$hN(B(a,u)) = B(a', u),$$

where

$$a'(\tau) = \begin{cases} 0 & \text{if } \tau = [\circ; \tau_1] \\ a(\tau_1)\dots a(\tau_\ell) & \text{if } \tau = [\bullet; \tau_1, \dots, \tau_\ell] \end{cases}$$

The lemmas

To obtain B-series expansions of the numerical solution we need three Lemmas.

Lemma 3. *Let* $a : 2\mathrm{T}^* \to \mathbb{R}$, *with* $a(\emptyset) = 1$, *then*

$$hN(B(a,u)) = B(a',u),$$

where

$$a'(\tau) = \begin{cases} 0 & \text{if } \tau = [\circ; \tau_1] \\ a(\tau_1)\dots a(\tau_\ell) & \text{if } \tau = [\bullet; \tau_1, \dots, \tau_\ell] \end{cases}$$

Lemma 3. *Let* $a : 2\mathrm{T}^* \to \mathbb{R}$, *then*

$$(hL)^l B(a,u) = B(\mathcal{L}^l a, u),$$

where

$$(\mathcal{L}^l a)(\tau) = \begin{cases} (\mathcal{L}^{l-1} a)(\tau_1) & \text{if } \tau = [\circ; \tau_1] \\ 0 & \text{if } \tau = [\bullet; \tau_1, \dots, \tau_\ell] \end{cases}$$

The lemmas

Lemma 4. *Let $\psi_x(z)$ be a power series*

$$\psi_x(z) = \sum_{l\geq 0} x^{[l]} z^l$$

and let $a : 2\mathrm{T}^* \to \mathbb{R}$, then

$$\psi_x(hL)B(a,u) = B(\psi_x(\mathcal{L})a, u),$$

where the elementary weight function satisfies, $(\psi_x(\mathcal{L})a)(\emptyset) = x^{[0]}a(\emptyset)$, and

$$(\psi_x(\mathcal{L})a)(\tau) = \sum_{l\geq 0} x^{[l]}(\mathcal{L}^l a)(\tau)$$

Exp. RK methods for parabolic PDEs

- Stiff order theory for ETD RK methods for parabolic PDEs (Hochbruck–Ostermann'04)
  - Abstract ODEs on a Banach spaces
  - Sectorial operators
  - Locally Lipschitz continuous functions

Exp. RK methods for parabolic PDEs

- Stiff order theory for ETD RK methods for parabolic PDEs (Hochbruck–Ostermann'04)
  - Abstract ODEs on a Banach spaces
  - Sectorial operators
  - Locally Lipschitz continuous functions

The error bounds depend form the space where the solution evolves!

It is not possible to construct stiff fourth order
explicit exponential Runge–Kutta method with only four stages

Exp. RK methods for parabolic PDEs

- Stiff order theory for ETD RK methods for parabolic PDEs (Hochbruck–Ostermann'04)
  - Abstract ODEs on a Banach spaces
  - Sectorial operators
  - Locally Lipschitz continuous functions

- Implicit Exp RK methods of *collocation type*

The methods converge at least with their stage order. Higher and even fractional order of convergence is possible if additional temporal and spatial regularity are required Hochbruck–Ostermann'04.

General linear methods

Consider $u' = f(u(t)), \quad u(t_0) = u_0, \quad f(u(t)) : \mathbb{R}^d \to \mathbb{R}^d.$
Assume that at the beginning of step number n, r quantities

$$u_1^{[n-1]}, u_2^{[n-1]}, \dots, u_r^{[n-1]},$$

are available from approximations computed in the previous steps. If

$$U_1, U_2, \dots, U_s$$

are the internal stage approximations to the solution at points near the current time step, then then the quantities imported into and evaluated in step number n are related by the equations

$$U_i = \sum_{j=1}^{s} a_{ij} h f(Uj) + \sum_{j=1}^{r} d_{ij} u_j^{[n-1]}, \quad i = 1, 2, \dots s,$$

$$u_i^{[n]} = \sum_{j=1}^{s} b_{ij} h f(Uj) + \sum_{j=1}^{r} v_{ij} u_j^{[n-1]}, \quad i = 1, 2, \dots r,$$

Vector notations

$$U_i = \sum_{j=1}^{s} a_{ij} h f(Uj) + \sum_{j=1}^{r} d_{ij} u_j^{[n-1]}, \quad i = 1, 2, \dots s,$$

$$u_i^{[n]} = \sum_{j=1}^{s} b_{ij} h f(Uj) + \sum_{j=1}^{r} v_{ij} u_j^{[n-1]}, \quad i = 1, 2, \dots r,$$

Vector notations

$$U_i = \sum_{j=1}^{s} a_{ij} h f(Uj) + \sum_{j=1}^{r} d_{ij} u_j^{[n-1]}, \quad i = 1, 2, \dots s,$$

$$u_i^{[n]} = \sum_{j=1}^{s} b_{ij} h f(Uj) + \sum_{j=1}^{r} v_{ij} u_j^{[n-1]}, \quad i = 1, 2, \dots r,$$

Introducing the vector notations

$$U = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_s \end{bmatrix}, \quad f(U) = \begin{bmatrix} f(U_1) \\ f(U_2) \\ \vdots \\ f(U_s) \end{bmatrix}, \quad u^{[n-1]} = \begin{bmatrix} u_1^{[n-1]} \\ u_2^{[n-1]} \\ \vdots \\ u_r^{[n-1]} \end{bmatrix}, \quad u^{[n]} = \begin{bmatrix} u_1^{[n]} \\ u_2^{[n]} \\ \vdots \\ u_r^{[n]} \end{bmatrix},$$

Vector notations

$$U_i = \sum_{j=1}^{s} a_{ij} h f(Uj) + \sum_{j=1}^{r} d_{ij} u_j^{[n-1]}, \quad i = 1, 2, \ldots s,$$

$$u_i^{[n]} = \sum_{j=1}^{s} b_{ij} h f(Uj) + \sum_{j=1}^{r} v_{ij} u_j^{[n-1]}, \quad i = 1, 2, \ldots r,$$

allows us to rewrite the above method in the following more compact form

$$\left[\begin{array}{c} U \\ \hline u^{[n]} \end{array}\right] = \left[\begin{array}{c|c} A \otimes I_d & D \otimes I_d \\ \hline B \otimes I_d & V \otimes I_d \end{array}\right] \left[\begin{array}{c} hf(U) \\ \hline u^{[n-1]} \end{array}\right],$$

where $\otimes$ is the Kronecker product and I_d is the $d \times d$ identity matrix.

Examples of GLMs

Consider k-step linear multistep methods of Adams type

$$u_n = u_{n-1} + h \sum_{i=0}^{k} \beta_i f(u_{n-i}).$$

Examples of GLMs

Consider k-step linear multistep methods of Adams type

$$u_n = u_{n-1} + h \sum_{i=0}^{k} \beta_i f(u_{n-i}).$$

In general linear form

$$\left[\begin{array}{c} U_1 \\ \hline u_n \\ hf(U_1) \\ hf(u_{n-1}) \\ \vdots \\ hf(u_{n-k-1}) \end{array}\right] = \left[\begin{array}{c|cccccc} \beta_0 & 1 & \beta_1 & \cdots & \beta_{k-1} & \beta_k \\ \hline \beta_0 & 1 & \beta_1 & \cdots & \beta_{k-1} & \beta_k \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{array}\right] \left[\begin{array}{c} hf(U_1) \\ \hline u_{n-1} \\ hf(u_{n-1}) \\ hf(u_{n-2}) \\ \vdots \\ hf(u_{n-k}) \end{array}\right].$$

Examples of GLMs

The classical fourth order Runge–Kutta method

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array},$$

Examples of GLMs

The classical fourth order Runge–Kutta method

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array},$$

can be written as

$$\left[\begin{array}{c} U_1 \\ U_2 \\ U_3 \\ U_4 \\ \hline u_n \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 & 0 & 1 \\ 0 & \frac{1}{2} & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ \hline \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & 1 \end{array}\right] \left[\begin{array}{c} hf(U_1) \\ hf(U_2) \\ hf(U_3) \\ hf(U_4) \\ \hline u_{n-1} \end{array}\right].$$

Examples of GLMs

It is not always appropriate to represent a Runge–Kutta method like a general liner method with $r = 1$. Example is the Lobatto IIIA method

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ \frac{1}{2} & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

Examples of GLMs

It is not always appropriate to represent a Runge–Kutta method like a general liner method with $r = 1$. Example is the Lobatto IIIA method

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ \frac{1}{2} & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}.$$

It has the following general linear form

$$\left[\begin{array}{c} U_1 \\ U_2 \\ \hline u_n \\ hf(U_2) \end{array}\right] = \left[\begin{array}{cc|cc} \frac{1}{3} & -\frac{1}{24} & 1 & \frac{5}{24} \\ \frac{2}{3} & \frac{1}{6} & 1 & \frac{1}{6} \\ \hline \frac{2}{3} & \frac{1}{6} & 1 & \frac{1}{6} \\ 0 & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} hf(U_1) \\ hf(U_2) \\ \hline u_{n-1} \\ hf(u_{n-1}) \end{array}\right].$$

Exp. general linear methods

Consider the following unified format of Exp. GLMs

$$U_i = \sum_{j=1}^{s}\sum_{l=1}^{m} \alpha_{ij}^{[l]}\, \phi^{[l]}(c_i)(hL)\, hN(U_j) + \sum_{j=1}^{r}\sum_{l=1}^{m} \delta_{ij}^{[l]}\, \phi^{[l]}(c_i)(hL)\, u_j^{[n-1]},$$

$$u_i^{[n]} = \sum_{j=1}^{s}\sum_{l=1}^{m} \beta_{ij}^{[l]}\, \phi^{[l]}(1)(hL)\, hN(U_j) + \sum_{j=1}^{r}\sum_{l=1}^{m} \nu_{ij}^{[l]}\, \phi^{[l]}(1)(hL)\, u_j^{[n-1]}.$$

Exp. general linear methods

Consider the following unified format of Exp. GLMs

$$U_i = \sum_{j=1}^{s}\sum_{l=1}^{m} \alpha_{ij}^{[l]}\, \phi^{[l]}(c_i)(hL)\, hN(U_j) + \sum_{j=1}^{r}\sum_{l=1}^{m} \delta_{ij}^{[l]}\, \phi^{[l]}(c_i)(hL)\, u_j^{[n-1]},$$

$$u_i^{[n]} = \sum_{j=1}^{s}\sum_{l=1}^{m} \beta_{ij}^{[l]}\, \phi^{[l]}(1)(hL)\, hN(U_j) + \sum_{j=1}^{r}\sum_{l=1}^{m} \nu_{ij}^{[l]}\, \phi^{[l]}(1)(hL)\, u_j^{[n-1]}.$$

Exp. general linear methods

Consider the following unified format of Exp. GLMs

$$U_i = \sum_{j=1}^{s}\sum_{l=1}^{m} \alpha_{ij}^{[l]}\, \phi^{[l]}(c_i)(hL)\, hN(U_j) + \sum_{j=1}^{r}\sum_{l=1}^{m} \delta_{ij}^{[l]}\, \phi^{[l]}(c_i)(hL)\, u_j^{[n-1]},$$

$$u_i^{[n]} = \sum_{j=1}^{s}\sum_{l=1}^{m} \beta_{ij}^{[l]}\, \phi^{[l]}(1)(hL)\, hN(U_j) + \sum_{j=1}^{r}\sum_{l=1}^{m} \nu_{ij}^{[l]}\, \phi^{[l]}(1)(hL)\, u_j^{[n-1]}.$$

Similarly, to the traditional GLMs, we can represent the method in the following matrix form

$$\left[\begin{array}{c} U \\ \hline u^{[n]} \end{array}\right] = \left[\begin{array}{c|c} A(\phi) & D(\phi) \\ \hline B(\phi) & V(\phi) \end{array}\right] \left[\begin{array}{c} hN(U) \\ \hline u^{[n-1]} \end{array}\right],$$

where each of the coefficient matrices $A(\phi), B(\phi), D(\phi), V(\phi)$ has entries which are linear combinations of the $\phi^{[l]}$ functions.

Generalized IF methods

Seek for a solution of the form

$$u(t_n + t) = \phi_{t,\widehat{F}}(v(t)),$$

where $\phi_{t,\widehat{F}}$ is the flow of the differential equation

$$u' = \widehat{F}(u,t), \qquad u(0) = u_n.$$

Generalized IF methods

Seek for a solution of the form

$$u(t_n + t) = \phi_{t,\widehat{F}}(v(t)),$$

where $\phi_{t,\widehat{F}}$ is the flow of the differential equation

$$u' = \widehat{F}(u,t), \qquad u(0) = u_n.$$

The vector field $\widehat{F}(u,t)$ must:

- Approximates the original vector field $f(u, t_n + t)$ around the point u_n.
- Have a flow $\phi_{t,\widehat{F}}$ which is easy to compute exactly.

Generalized IF methods

Seek for a solution of the form

$$u(t_n + t) = \phi_{t,\widehat{F}}(v(t)),$$

where $\phi_{t,\widehat{F}}$ is the flow of the differential equation

$$u' = \widehat{F}(u,t), \qquad u(0) = u_n.$$

The vector field $\widehat{F}(u,t)$ must:

- Approximates the original vector field $f(u, t_n + t)$ around the point u_n.
- Have a flow $\phi_{t,\widehat{F}}$ which is easy to compute exactly.

The corresponding differential equation for the v variable is

$$v'(t) = \left(\frac{\partial u}{\partial v}\right)^{-1} \left(f(u, t_n + t) - \widehat{F}(u,t),\right), \qquad v(0) = u_n.$$

Use numerical method on the transformed equation and then transform back.

GIF for semilinear problems

For the semilinear problem (1), we can choice $\widehat{F}(u,t) = Lu + \mathsf{L}_k(N,t)$, where $\mathsf{L}_k(N,t)$ is the Lagrange interpolating polynomial of degree $k-1$ for the function $N(u(t_n+t),\, t_n+t)$, which passes through the k points $N_n, N_{n-1}, \ldots, N_{n-k+1}$.

The transformed equation is

$$v'(t) = e^{-tL}\big(N(u(t_n+t),\, t_n+t) - \mathsf{L}_k(N,t)\big) \qquad v(0) = u_n.$$

GIF for semilinear problems

For the semilinear problem (1), we can choice $\widehat{F}(u,t) = Lu + \mathsf{L}_k(N,t)$,
where $\mathsf{L}_k(N,t)$ is the Lagrange interpolating polynomial of degree $k-1$ for the function $N(u(t_n+t),\, t_n+t)$, which passes through the k points $N_n, N_{n-1}, \ldots, N_{n-k+1}$.

The transformed equation is

$$v'(t) = e^{-tL}\big(N(u(t_n+t),\, t_n+t) - \mathsf{L}_k(N,t)\big) \qquad v(0) = u_n.$$

- Applying a multistep method to the transformed equation leads to a class of methods which includes as a special cases all exponential multistep methods considered so far.
- Applying a multistage method to the transformed equation leads to a new class of methods known as GIR/RK methods (Krogstad'03).

GIF/RK methods

$$\mathsf{L}_0(N,t) = 0 \quad - \quad \text{IF RK (Lawson)},$$

$$\mathsf{L}_1(N,t) = N_n \quad - \quad \text{GIF1/RK (Krogstad)},$$

$$\mathsf{L}_2(N,t) = N_n + t\left(\frac{N_n - N_{n-1}}{h}\right) \quad - \quad \text{GIF2/RK (Krogstad)},$$

GIF/RK methods

$$
\begin{aligned}
\mathsf{L}_0(N,t) &= 0 && - \quad \text{IF RK (Lawson)},\\
\mathsf{L}_1(N,t) &= N_n && - \quad \text{GIF1/RK (Krogstad)},\\
\mathsf{L}_2(N,t) &= N_n + t\left(\frac{N_n - N_{n-1}}{h}\right) && - \quad \text{GIF2/RK (Krogstad)},
\end{aligned}
$$

$$
\left[\begin{array}{cccc|cc}
0 & 0 & 0 & 0 & I & 0 \\
a_{21}(hL) & 0 & 0 & 0 & e^{c_2 hL} & d_{22}(hL) \\
a_{31}(hL) & \alpha_{32}e^{(c_3-c_2)hL} & 0 & 0 & e^{c_3 hL} & d_{32}(hL) \\
a_{41}(hL) & \alpha_{42}e^{(c_4-c_2)hL} & \alpha_{43}e^{(c_4-c_3)hL} & 0 & e^{c_4 hL} & d_{42}(hL) \\
\hline
b_1(hL) & \beta_2 e^{(1-c_2)hL} & \beta_3 e^{(1-c_3)hL} & \beta_4 e^{(1-c_4)hL} & e^{hL} & v_{12}(hL) \\
I & 0 & 0 & 0 & 0 & 0
\end{array}\right],
$$

Fourth order GIF2/RK method

GIF/RK methods

$$
\begin{aligned}
\mathsf{L}_0(N,t) &= 0 && - \quad \text{IF RK (Lawson)},\\
\mathsf{L}_1(N,t) &= N_n && - \quad \text{GIF1/RK (Krogstad)},\\
\mathsf{L}_2(N,t) &= N_n + t\left(\frac{N_n - N_{n-1}}{h}\right) && - \quad \text{GIF2/RK (Krogstad)},
\end{aligned}
$$

where

$$a_{21}(hL) = c_2\phi^{[1]} + c_2^2\phi^{[2]},$$

$$a_{31}(hL) = c_3\phi^{[1]} + c_3^2\phi^{[2]} - \alpha_{32}\,(1+c_2)\,e^{(c_3-c_2)hL},$$

$$a_{41}(hL) = c_4\phi^{[1]} + c_4^2\phi^{[2]} - \alpha_{42}\,(1+c_2)\,e^{(c_4-c_2)hL} - \alpha_{43}\,(1+c_3)\,e^{(c_4-c_3)hL},$$

$$b_1(hL) = \phi^{[1]}+\phi^{[2]}-\beta_2(1+c_2)e^{(1-c_2)hL}-\beta_3(1+c_3)e^{(1-c_3)hL}-\beta_4(1+c_4)e^{(1-c_4)hL},$$

$$d_{22}(hL) = -c_2^2\phi^{[2]},$$

$$d_{32}(hL) = -c_3^2\phi^{[2]} + c_2\alpha_{32}e^{(c_3-c_2)hL},$$

$$d_{42}(hL) = -c_4^2\phi^{[2]} + c_2\alpha_{42}e^{(c_4-c_2)hL} + c_3\alpha_{43}e^{(c_4-c_3)hL},$$

$$v_{12}(hL) = -\phi^{[2]} + c_2\beta_2e^{(1-c_2)hL} + c_3\beta_3e^{(1-c_3)hL} + c_4\beta_4e^{(1-c_4)hL}.$$

GIF/RK methods

Similarly, if $\widehat{F}(u,t) = Lu + \mathsf{T}_k(N,t)$, where

$$\mathsf{T}_k(N,t) = b + \sum_{\alpha=1}^{k} \left(c_\alpha \sin(\alpha t) + d_\alpha \cos(\alpha t)\right); \qquad k \in \mathbb{N},\ c_\alpha, d_\alpha \in,$$

it is possible to construct new GIF methods.

This approach leads to the followig $\phi^{[l]}$ funactions

$$\begin{aligned} \phi^{[\alpha]}_{\sin}(hL) &= \frac{e^{hL}\alpha - L\sin(\alpha h) - \alpha\cos(\alpha h)I}{\alpha^2 I + L^2}, \\ \phi^{[\alpha]}_{\cos}(hL) &= \frac{e^{hL}L - L\cos(\alpha h) + \alpha\sin(\alpha h)I}{\alpha^2 I + L^2}. \end{aligned}$$

Exp. int. and Lie group methods

Lie group methods

- Designed to preserve certain qualitatively properties of the exact flow

Exp. int. and Lie group methods

Lie group methods

- Designed to preserve certain qualitatively properties of the exact flow
- The freedom in the choice of the action allows to define the basic motions on the manifold to be given by approximations of the exact flow.

Generic presentation of a diff. eq.

Every differential equation evolving on a homogeneous space $\mathcal{M}$ can always be written as

$$u'(t) = F(u) \circledast u, \qquad u(t_0) = u_0,$$

where $F : \mathcal{M} \to \mathfrak{g}$ and $\circledast : \mathfrak{g} \times \mathcal{M} \to T\mathcal{M}$.

Generic presentation of a diff. eq.

Every differential equation evolving on a homogeneous space $\mathcal{M}$ can always be written as

$$u'(t) = F(u) \circledast u, \qquad u(t_0) = u_0,$$

where $F : \mathcal{M} \to \mathfrak{g}$ and $\circledast : \mathfrak{g} \times \mathcal{M} \to T\mathcal{M}$.

For a fix $\Theta \in \mathfrak{g}$ the product $\Theta \circledast p$ gives a vector field on $\mathcal{M}$

$$\mathcal{F}_\Theta(p) = \Theta \circledast p$$

The vector field $\mathcal{F}_\Theta$ is called a frozen vector field.

Generic presentation of a diff. eq.

Every differential equation evolving on a homogeneous space $\mathcal{M}$ can always be written as

$$u'(t) = F(u) \circledast u, \qquad u(t_0) = u_0,$$

where $F : \mathcal{M} \to \mathfrak{g}$ and $\circledast : \mathfrak{g} \times \mathcal{M} \to T\mathcal{M}$.

For a fix $\Theta \in \mathfrak{g}$ the product $\Theta \circledast p$ gives a vector field on $\mathcal{M}$

$$\mathcal{F}_\Theta(p) = \Theta \circledast p$$

The vector field $\mathcal{F}_\Theta$ is called a frozen vector field.

In order to implement any Lie group integrator we need to know:

- The generic presentation F
- The structure of the Lie algebra $\mathfrak{g}$
- The Lie algebra action $*$ on $\mathcal{M}$

- How the commutators between elements in $\mathfrak{g}$ are defined (RKMK methods)

Lie group integrators

Algotithm. *(Crouch–Grossman'93)*

```
for $i = 1, \ldots, s$ do
    $U_i = (h\alpha_{is} F_s) * \cdots * (h\alpha_{i1} F_1) * u_n$
    $F_i = F(U_i)$
end
$u_{n+1} = (h\beta_s F_s) * \cdots * (h\beta_1 F_1) * u_n$
```

Lie group integrators

Algotithm. *(Crouch–Grossman'93)*

```
for $i = 1, \dots, s$ do
    $U_i = (h\alpha_{is}F_s) * \cdots * (h\alpha_{i1}F_1) * u_n$
    $F_i = F(U_i)$
end
$u_{n+1} = (h\beta_s F_s) * \cdots * (h\beta_1 F_1) * u_n$
```

Algotithm. *(Runge–Kutta Munthe-Kaas'99)*

```
for $i = 1, \dots, s$ do
    $\Theta_i = h \sum_{j=1}^{s} \alpha_{ij} K_j$
    $F_i = F((\Theta_i) * u_n)$
    $K_i = d\Psi^{-1}(F_i)$
end
$u_{n+1} = (h \sum_{i=1}^{s} \beta_i K_i) * u_n$
```

Lie group integrators

Algotithm. *(Crouch–Grossman'93)*

```
for $i = 1, \dots, s$ do
    $U_i = (h\alpha_{is}F_s) * \cdots * (h\alpha_{i1}F_1) * u_n$
    $F_i = F(U_i)$
end
$u_{n+1} = (h\beta_s F_s) * \cdots * (h\beta_1 F_1) * u_n$
```

Algotithm. *Commutator-free Lie group method (Celledoni, Marthinsen, Owren'03)*

```
for $i = 1, \dots, s$ do
    $U_i = (h\sum_{k=1}^{s} \alpha_{iJ}^{k} F_k) * \cdots * (h\sum_{k=1}^{s} \alpha_{i1}^{k} F_k) * u_n$
    $F_i = F(U_i)$
end
$u_{n+1} = (h\sum_{k=1}^{s} \beta_J^k F_k) * \cdots * (h\sum_{k=1}^{s} \beta_1^k F_k) * u_n$
```

Basic motions on $\mathcal{M}$

Consider the following nonautonomous problem defined on $\mathbb{R}^d$

$$u' = f(u,t), \qquad u(t_0) = u_0.$$

Basic motions on $\mathcal{M}$

Consider the following nonautonomous problem defined on $\mathbb{R}^d$

$$u' = f(u,t), \qquad u(t_0) = u_0.$$

Define:

- the basic movements on $\mathcal{M}$ to be given by the solution of a simpler diff. equation

$$u' = \mathcal{F}_\Theta(u,t), \qquad u(t_0) = u_0, \tag{2}$$

where $\mathcal{F}_\Theta(u,t)$ approximates $f(u,t)$.

- the Lie algebra $\mathfrak{g}$ to be the set of all coefficients Θ of the *frozen* vector fields $\mathcal{F}_\Theta$.

- the algebra action $h\Theta * u_0$ to be the solution of (2) at time $t_0 + h$.

The choice of action

- The simplest case
  - $\mathfrak{g} = \{b^{[0]} \in \mathbb{R}^d\}$.
  - The frozen vector field is $\mathcal{F}_{b^{[0]}}(u,t) = b^{[0]}$.
  - The algebra action is $hb^{[0]} * u_0 = u_0 + hb^{[0]}$ (translations).
  - The commutators are given by $[\Theta_1, \Theta_2] = (0, 0)$.

In this case we recover the traditional integration schemes.

The choice of action

- The simplest case
  - $\mathfrak{g} = \{b^{[0]} \in \mathbb{R}^d\}$.
  - The frozen vector field is $\mathcal{F}_{b^{[0]}}(u,t) = b^{[0]}$.
  - The algebra action is $hb^{[0]} * u_0 = u_0 + hb^{[0]}$ (translations).
  - The commutators are given by $[\Theta_1, \Theta_2] = (0, 0)$.

  In this case we recover the traditional integration schemes.

- When $f(u,t) = L(u,t)u + N(u,t)$ (such a representation is always possible)
  - $\mathfrak{g} = \{(A, b^{[0]}) : A \in \mathbb{R}^{d\times d}, b^{[0]} \in \mathbb{R}^d\}$.
  - The frozen vector field is $\mathcal{F}_{(A,b^{[0]})}(u,t) = Au + b^{[0]}$.
  - The algebra action is given by $h(A, b^{[0]}) * u_0 = e^{hA}u_0 + hb^{[0]}\phi^{[1]}(hA)$, where e^{hA} denotes the matrix exponential and $\phi^{[1]}$ is the first ETD $\phi^{[i]}$ function.
  - The commutators are given by $[\Theta_1, \Theta_2] = (A_1A_2 - A_2A_1, A_1b_2^{[0]} - A_2b_1^{[0]})$.

  In this case we recover the affine algebra action proposed by Munthe-Kaas'99.

Nonautonomous frozen vector fields

- Similarly, when $f(u,t) = L(u,t)u + N^{[0]}(u,t) + tN^{[1]}(u,t)$.
  - $\mathfrak{g} = \{(A, b^{[0]}, b^{[1]}) : A \in \mathbb{R}^{d\times d}, b^{[1]}, b^{[0]} \in \mathbb{R}^d\}$
  - The frozen vector field is $\mathcal{F}_{(A,b^{[1]},b^{[0]})}(u,t) = Au + c_0 + tc_1$, where $c_0 = b^{[0]} + t_0(1-\lambda)b^{[1]}$ and $c_1 = \lambda b^{[1]}$.
  - The algebra action is given by $h(A, b^{[1]}, b^{[0]}) * u_0 = e^{hA}u_0 + h(b^{[0]} + t_0 b^{[1]})\phi^{[1]}(hA) + h^2\lambda b^{[1]}\phi^{[2]}(hA)$.
  - The commutators are given by $[\Theta_1, \Theta_2] = \Big([A_1, A_2], A_1 b_2^{[1]} - A_2 b_1^{[1]}, A_1 b_2^{[0]} - A_2 b_1^{[0]} + \lambda_2 b_1^{[1]} - \lambda_1 b_2^{[1]}, 0\Big)$.

Nonautonomous frozen vector fields

- Similarly, when $f(u,t) = L(u,t)u + N^{[0]}(u,t) + tN^{[1]}(u,t)$.
  - $\mathfrak{g} = \{(A, b^{[0]}, b^{[1]}) : A \in \mathbb{R}^{d\times d}, b^{[1]}, b^{[0]} \in \mathbb{R}^d\}$
  - The frozen vector field is
    $\mathcal{F}_{(A,b^{[1]},b^{[0]})}(u,t) = Au + c_0 + tc_1$, where $c_0 = b^{[0]} + t_0(1-\lambda)b^{[1]}$ and $c_1 = \lambda b^{[1]}$.
  - The algebra action is given by
    $h(A, b^{[1]}, b^{[0]}) * u_0 = e^{hA}u_0 + h(b^{[0]} + t_0 b^{[1]})\phi^{[1]}(hA) + h^2\lambda b^{[1]}\phi^{[2]}(hA)$.
  - The commutators are given by
    $[\Theta_1, \Theta_2] = \Big([A_1, A_2], A_1 b_2^{[1]} - A_2 b_1^{[1]}, A_1 b_2^{[0]} - A_2 b_1^{[0]} + \lambda_2 b_1^{[1]} - \lambda_1 b_2^{[1]}, 0\Big)$.
- Can generalize this approach to the case $f(u,t) = L(u,t)u + \sum_{k=0}^{p} t^k N^{[k]}(u,t)$.

Nonautonomous frozen vector fields

- Similarly, when $f(u,t) = L(u,t)u + N^{[0]}(u,t) + tN^{[1]}(u,t)$.
  - $\mathfrak{g} = \{(A, b^{[0]}, b^{[1]}) : A \in \mathbb{R}^{d\times d}, b^{[1]}, b^{[0]} \in \mathbb{R}^d\}$
  - The frozen vector field is $\mathcal{F}_{(A,b^{[1]},b^{[0]})}(u,t) = Au + c_0 + tc_1$, where $c_0 = b^{[0]} + t_0(1-\lambda)b^{[1]}$ and $c_1 = \lambda b^{[1]}$.
  - The algebra action is given by $h(A, b^{[1]}, b^{[0]}) * u_0 = e^{hA}u_0 + h(b^{[0]} + t_0 b^{[1]})\phi^{[1]}(hA) + h^2\lambda b^{[1]}\phi^{[2]}(hA)$.
  - The commutators are given by $[\Theta_1, \Theta_2] = \Big([A_1, A_2], A_1 b_2^{[1]} - A_2 b_1^{[1]}, A_1 b_2^{[0]} - A_2 b_1^{[0]} + \lambda_2 b_1^{[1]} - \lambda_1 b_2^{[1]}, 0\Big)$.

- Can generalize this approach to the case $f(u,t) = L(u,t)u + \sum_{k=0}^{p} t^k N^{[k]}(u,t)$.
- It is possibel to show that:
  - IF RK methods are RKMK methods (Krogstad'03);
  - GIF/RK methods are also RKMK methods.

Implementation issues

Consider the ETD $\phi^{[i]}$ functions

$$\phi^{[1]}(z) = \frac{e^z - 1}{z}, \qquad \phi^{[i+1]}(z) = \frac{\phi^{[i]}(z) - \frac{1}{i!}}{z}, \quad \text{for} \quad i = 2, 3, \ldots.$$

Implementation issues

Consider the ETD $\phi^{[i]}$ functions

$$\phi^{[1]}(z) = \frac{e^z - 1}{z}, \qquad \phi^{[i+1]}(z) = \frac{\phi^{[i]}(z) - \frac{1}{i!}}{z}, \quad \text{for} \quad i = 2, 3, \ldots .$$

Numerical techniques

- Decomposition methods
- Krylov subspace approximations
- Cauchy integral approach

Decomposition methods

At the heart of all decomposition methods is the similarity transformation

$$A = SBS^{-1},$$

where $A = \gamma h L$. Therefore

$$\phi^{[i]}(A) = S\phi^{[i]}(B)S^{-1}.$$

Decomposition methods

At the heart of all decomposition methods is the similarity transformation

$$A = SBS^{-1},$$

where $A = \gamma h L$. Therefore

$$\phi^{[i]}(A) = S\phi^{[i]}(B)S^{-1}.$$

Two conflicting tasks:

- Make B close to diagonal so that $\phi^{[i]}(B)$ is easy to compute.
- Make S well conditioned so that errors are not magnified.

Decomposition methods

At the heart of all decomposition methods is the similarity transformation

$$A = SBS^{-1},$$

where $A = \gamma h L$. Therefore

$$\phi^{[i]}(A) = S\phi^{[i]}(B)S^{-1}.$$

Algotithm. *(Block Schur–Parlett algorithm)*

- *Compute the Schur decomposition $A = QTQ^*$ (QR algorithm).*

  - the scalar Parlett recurrence fails when the eigenvalues of T are equal or close to each other
- *Reorder T into a block upper triangular matrix $\widetilde{T}$.*
- *Compute $\widetilde{F}_{kk}^{[i]} = \phi^{[i]}(\widetilde{T}_{kk})$ for all diagonal blocks $\widetilde{T}_{kk}$ (Chebyshev rational approximations).*
- *Find $\widetilde{F}^{[i]} = \phi^{[i]}(\widetilde{T})$ by solving the Silvester equation (Parlett recurrence)*
  $$\widetilde{T}_{kk}\widetilde{F}_{kl}^{[i]} - \widetilde{F}_{kl}^{[i]}\widetilde{T}_{ll} = \widetilde{F}_{kk}^{[i]}\widetilde{T}_{kl} - \widetilde{T}_{kl}\widetilde{F}_{ll}^{[i]} + \sum_{j=k+1}^{l-1}\left(\widetilde{F}_{kj}^{[i]}\widetilde{T}_{jl} - \widetilde{T}_{kj}\widetilde{F}_{jl}^{[i]}\right)$$
- *Compute $\phi^{[i]}(A) = Q\phi^{[i]}(\widetilde{T})Q^*$.*

The cost depends from the eigenvalue distribution of A- between $28d^3$ and $d^4/3$ flops.

Decomposition methods

At the heart of all decomposition methods is the similarity transformation

$$A = SBS^{-1},$$

where $A = \gamma hL$. Therefore

$$\phi^{[i]}(A) = S\phi^{[i]}(B)S^{-1}.$$

When A is summetric

Algotithm. *(Tridiagonal Reduction)*

- *Calculate a symmetric tridiagonal reduction* $A = Q\mathsf{T}Q^T$ *(Hauseholder reflections).*
- *Find the largest eigenvalue* λ_1 *of* T*.*
- *Compute* $\phi^{[i]}(\mathsf{T})$ *by a Chebyshev rational approximation with respect to* λ_1*.*
  $e^B \approx e^{\lambda_1} R_p(B - \lambda_1 I),$ *the structure of* B *depends of* $\phi^{[i]}$ *and* λ_1
- *Calculate* $\phi^{[i]}(A)$ *by* $\phi^{[i]}(A) = Q\phi^{[i]}(\mathsf{T})Q^T$*.*

The total number of operations for computing each of the functions $\phi^{[i]}$ is $\mathcal{O}(\frac{10}{3}d^3)$.

Krylov subspace approximations

Approximately project the action of $\phi^{[i]}(A)$ on a state vector $v \in \mathbb{C}^d$, to a small Krylov subspace

$$K_{\mathrm{m}} \equiv \text{span}\{v, Av, \dots, A^{\mathrm{m}-1}v\}.$$

Construct a orthogonal basis $V_{\mathrm{m}} = [v_1, v_2, \dots, v_{\mathrm{m}}]$ of K_{m} (Arnoldi, Lanczos)
If H_{m} is the $\mathrm{m} \times \mathrm{m}$ upper Hessenberg matrix generated by the process then

$$V_{\mathrm{m}}^T A V_{\mathrm{m}} = H_{\mathrm{m}}.$$

Therefore, H_{m} is the orthogonal projection of A to the subspace K_{m} and

$$\phi^{[i]}(A)v \approx V_{\mathrm{m}} V_{\mathrm{m}}^T \phi^{[i]}(A)v = \beta V_{\mathrm{m}} V_{\mathrm{m}}^T \phi^{[i]}(A) V_{\mathrm{m}} e_1 \approx \beta V_{\mathrm{m}} \phi^{[i]}(H_{\mathrm{m}}) e_1,$$

where e_1 is the first unit vector in $\mathbb{R}^{\mathrm{m}}$ and $\beta \equiv ||v||_2$.

Krylov subspace approximations

Approximately project the action of $\phi^{[i]}(A)$ on a state vector $v \in \mathbb{C}^d$, to a small Krylov subspace

$$K_{\mathrm{m}} \equiv \mathrm{span}\{v, Av, \ldots, A^{\mathrm{m}-1}v\}.$$

Construct a orthogonal basis $V_{\mathrm{m}} = [v_1, v_2, \ldots, v_{\mathrm{m}}]$ of K_{m} (Arnoldi, Lanczos)
If H_{m} is the $\mathrm{m} \times \mathrm{m}$ upper Hessenberg matrix generated by the process then

$$V_{\mathrm{m}}^T A V_{\mathrm{m}} = H_{\mathrm{m}}.$$

Therefore, H_{m} is the orthogonal projection of A to the subspace K_{m} and

$$\phi^{[i]}(A)v \approx V_{\mathrm{m}} V_{\mathrm{m}}^T \phi^{[i]}(A)v = \beta V_{\mathrm{m}} V_{\mathrm{m}}^T \phi^{[i]}(A) V_{\mathrm{m}} e_1 \approx \beta V_{\mathrm{m}} \phi^{[i]}(H_{\mathrm{m}}) e_1,$$

where e_1 is the first unit vector in $\mathbb{R}^{\mathrm{m}}$ and $\beta \equiv ||v||_2$.

- Superlinear convergence (Hochbruck, Lubich'98)

Krylov subspace approximations

Approximately project the action of $\phi^{[i]}(A)$ on a state vector $v \in \mathbb{C}^d$, to a small Krylov subspace

$$K_{\mathrm{m}} \equiv \mathrm{span}\{v, Av, \ldots, A^{\mathrm{m}-1}v\}.$$

Construct a orthogonal basis $V_{\mathrm{m}} = [v_1, v_2, \ldots, v_{\mathrm{m}}]$ of K_{m} (Arnoldi, Lanczos)
If H_{m} is the $\mathrm{m} \times \mathrm{m}$ upper Hessenberg matrix generated by the process then

$$V_{\mathrm{m}}^T A V_{\mathrm{m}} = H_{\mathrm{m}}.$$

Therefore, H_{m} is the orthogonal projection of A to the subspace K_{m} and

$$\phi^{[i]}(A)v \approx V_{\mathrm{m}} V_{\mathrm{m}}^T \phi^{[i]}(A)v = \beta V_{\mathrm{m}} V_{\mathrm{m}}^T \phi^{[i]}(A) V_{\mathrm{m}} e_1 \approx \beta V_{\mathrm{m}} \phi^{[i]}(H_{\mathrm{m}}) e_1,$$

where e_1 is the first unit vector in $\mathbb{R}^{\mathrm{m}}$ and $\beta \equiv ||v||_2$.

- Superlinear convergence (Hochbruck, Lubich'98)
- Preconditioning the Lanczos process (Hochbruck,Van der Eshof'04)

Cauchy integral approach

Based on the Cauchy integral formula

$$\phi^{[i]}(A) = \frac{1}{2\pi i} \int_{\Gamma_A} \phi^{[i]}(\lambda)(\lambda I - A)^{-1} d\lambda,$$

where Γ_A is a contour in the complex plane that encloses the eigenvalue of A, and it is also well separated from 0. It is practical to choose the contour Γ_A to be a circle centered on the real axis.
Using the trapezoid rule, we obtain the following approximation

$$\phi^{[i]}(A) \approx \frac{1}{k} \sum_{j=1}^{k} \lambda_j \phi^{[i]}(\lambda_j)(\lambda_j I - A)^{-1},$$

where k is the number of the equally spaced points λ_j along the contour Γ_A.

Cauchy integral approach

To achieve computational savings we can use the formula

$$\phi^{[i]}(A) = \phi^{[i]}(\gamma hL) = \frac{1}{2\pi i}\int_\Gamma \phi^{[i]}(\gamma\lambda)(\lambda I - hL)^{-1} d\lambda,$$

where the contour Γ encloses the eigenvalues of γhL and $\gamma\Gamma$ is well separated from 0 for all γ in the integration process.
As before

$$\phi^{[i]}(A) \approx \frac{1}{k}\sum_{j=1}^{k} \lambda_j \phi^{[i]}(\gamma\lambda_j)(\lambda_j I - hL)^{-1},$$

where now λ_j are the equally spaced points along the contour Γ.

Note: The inverse matrices no longer depend of γ.

Cauchy integral approach

To achieve computational savings we can use the formula

$$\phi^{[i]}(A) = \phi^{[i]}(\gamma hL) = \frac{1}{2\pi i}\int_{\Gamma} \phi^{[i]}(\gamma\lambda)(\lambda I - hL)^{-1} d\lambda,$$

where the contour Γ encloses the eigenvalues of γhL and $\gamma\Gamma$ is well separated from 0 for all γ in the integration process.
As before

$$\phi^{[i]}(A) \approx \frac{1}{k}\sum_{j=1}^{k} \lambda_j \phi^{[i]}(\gamma\lambda_j)(\lambda_j I - hL)^{-1},$$

where now λ_j are the equally spaced points along the contour Γ.

When L arises from a finite difference approximation, we can benefit from its sparse block structure and find the action of the inverse martices to a given vector by:

- iterative methods - *preconditioned conjugate gradient* and *multigrid methods*.

- direct methods - CR, FFT, $FACR$, LU factorization.

A modification of LU facorization

Consider the following linear system

$$Mx = f,$$

where

$$M = \begin{pmatrix} A & B & & & & & \\ B^* & A & B & & 0 & & \\ & B^* & A & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot \\ & 0 & & & & \cdot & \cdot & B \\ & & & & & & B^* & A \end{pmatrix},$$

is an Hermitian tridiagonal block Teoplitz matrix with block size n. A and B are $m \times m$ matrices, x and f are column vectors with size nm.

The idea

First we sove a parametric linear systems of the form

$$Ny = f,$$

where

$$N = \begin{pmatrix} X & B & & & & \\ B^* & A & . & & 0 & \\ & . & . & . & & \\ & & . & . & . & \\ & & & . & . & . \\ & 0 & & & . & . & B \\ & & & & & B^* & A \end{pmatrix},$$

is a block tridiagonal matrix with block size n and X is a parameter.

The idea

First we sove a parametric linear systems of the form

$$Ny = f,$$

The matrix N admits the following LU factorization

$$N = LU = \begin{pmatrix} I & & & & \\ B^*X^{-1} & \ddots & & & 0 \\ & \ddots & \ddots & & \\ 0 & & \ddots & \ddots & \\ & & & B^*X^{-1} & I \end{pmatrix} \begin{pmatrix} X & B & & & \\ & \ddots & \ddots & & 0 \\ & & \ddots & \ddots & \\ 0 & & & \ddots & B \\ & & & & X \end{pmatrix},$$

where I is the $m \times m$ identity matrix.

The idea

First we sove a parametric linear systems of the form

$$Ny = f,$$

The matrix N admits the following LU factorization

$$N = LU = \begin{pmatrix} I & & & & \\ B^*X^{-1} & . & & & 0 \\ & . & . & & \\ 0 & & . & . & \\ & & & B^*X^{-1} & I \end{pmatrix} \begin{pmatrix} X & B & & & \\ & . & . & & 0 \\ & & . & . & \\ 0 & & & . & B \\ & & & & X \end{pmatrix},$$

where I is the $m \times m$ identity matrix.
The parameter X must satisfie the nonlinear matrix equation

$$X + B^*X^{-1}B = A.$$

The idea

First we sove a parametric linear systems of the form

$$Ny = f,$$

The matrix N admits the following LU factorization

$$N = LU = \begin{pmatrix} I & & & & \\ B^* X^{-1} & . & & & 0 \\ & . & . & & \\ 0 & & . & . & \\ & & & B^* X^{-1} & I \end{pmatrix} \begin{pmatrix} X & B & & & \\ & . & . & & 0 \\ & & . & . & \\ 0 & & & . & B \\ & & & & X \end{pmatrix},$$

where I is the $m \times m$ identity matrix.
Therefore, the solution y can be obtained by solving two simpler systems

$$L\,z = f, \quad z = \{z_i\}_{i=1,\dots,n}$$
$$U\,y = z, \quad y = \{y_i\}_{i=1,\dots,n},$$

The idea

The matrices M and N are related by relation

$$M = N + E_1 V_1^T,$$

where $E_1 = \begin{pmatrix} I \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad V_1^T = \begin{pmatrix} A - X & 0 & \dots & 0 \end{pmatrix}.$

The idea

The matrices M and N are related by relation

$$M = N + E_1 V_1^T,$$

where $E_1 = \begin{pmatrix} I \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad V_1^T = \begin{pmatrix} A - X & 0 & \dots & 0 \end{pmatrix}.$

Using Woodbury's formula we have

$$M^{-1} = N^{-1} - N^{-1}E_1(I + V_1^T N^{-1} E_1)^{-1} V_1^T N^{-1}.$$

Therefore, the solution x is obtained from the vector y as follows

$$\begin{aligned} x &= M^{-1} f \\ &= N^{-1} f - N^{-1}E_1(I + V_1^T N^{-1} E_1)^{-1} V_1^T N^{-1} f \\ &= y - N^{-1}E_1(I + (A - X)E_1^T N^{-1} E_1)^{-1} V_1^T y \\ &= y - N^{-1}E_1(I + (A - X)E_1^T N^{-1} E_1)^{-1} (A - X) y_1. \end{aligned}$$

Circulant matrices

Similar approach can also be used to solve linear systems of the form

$$W\,x = f,$$

where

$$W = \begin{pmatrix} M & N & S & & & & S^* & N^* \\ N^* & M & N & S & & & & S^* \\ S^* & N^* & . & . & . & & 0 & \\ & S^* & . & . & . & . & & \\ & & . & . & . & . & . & \\ & & 0 & & . & . & . & N & S \\ S & & & & & S^* & N^* & M & N \\ N & S & & & & & S^* & N^* & M \end{pmatrix}$$

is Hermitian pentadiagonal block circulant matrix

Circulant matrices

The parametric system in this case is

$$T\,y \;=\; f,$$

where

$$T \;=\; \begin{pmatrix} X & Y & S & & & & \\ Y* & Z & N & . & & 0 & \\ S* & N* & M & . & . & & \\ & & . & . & . & . & S \\ & 0 & & . & . & . & N \\ & & & & S* & N* & M \end{pmatrix},$$

is pentadiagonal matrix with block size n.

Circulant matrices

T admits the following LU factorization

$$T = LU = \begin{pmatrix} I_m & & & & \\ Y^*X^{-1} & . & & 0 & \\ S^*X^{-1} & . & . & & \\ & . & . & . & \\ 0 & & S^*X^{-1} & Y^*X^{-1} & I_m \end{pmatrix} \begin{pmatrix} X & Y & S & & 0 \\ & . & . & . & \\ & & . & . & S \\ & 0 & & . & Y \\ & & & & X \end{pmatrix},$$

where I_m is the identity matrix with size $m \times m$ and the paramethers $X = X^*$, Y and $Z = Z^*$ satisfy the following relation

$$F + Q^*F^{-1}Q = R,$$

where

$$F = \begin{pmatrix} X & Y \\ Y^* & Z \end{pmatrix}, \; Q = \begin{pmatrix} S & 0 \\ N & S \end{pmatrix}, \; R = \begin{pmatrix} M & N \\ N^* & M \end{pmatrix}.$$

Circulant matrices

T admits the following LU factorization

$$T = LU = \begin{pmatrix} I_m & & & & 0 \\ Y^* X^{-1} & . & & & \\ S^* X^{-1} & . & . & & \\ & . & . & . & \\ 0 & & S^* X^{-1} & Y^* X^{-1} & I_m \end{pmatrix} \begin{pmatrix} X & Y & S & & 0 \\ & . & . & . & \\ & & . & . & S \\ & 0 & & . & Y \\ & & & & X \end{pmatrix},$$

where I_m is the identity matrix with size $m \times m$ and the paramethers $X = X^*$, Y and $Z = Z^*$ satisfy the following relation

$$F + Q^* F^{-1} Q = R,$$

where

$$F = \begin{pmatrix} X & Y \\ Y^* & Z \end{pmatrix}, \quad Q = \begin{pmatrix} S & 0 \\ N & S \end{pmatrix}, \quad R = \begin{pmatrix} M & N \\ N^* & M \end{pmatrix}.$$

The solution x is obtained from y and the solution of a pentadiagonal block Teoplitz system by applying two times the Woodbury's formula.

Numerical experiments

The methods

- **ETD RK3(CM)** The third order method of Cox–Matthews;
- **ETD2RK3** A modification of the third order method of CM (continuous RK);
- **ETD2CF3** A stiff order three method based on the CF3 (continuous RK);
- **ETD RK4(CM)** The fourth order method of Cox–Matthews;
- **ETD RK4(Kr)** The fourth order method of Krogstad;
- **ETD RK4(Min)** A fourth order method which satisfies half of the order 5 coditions;
- **ETD RK4(HO)** The stiff order four method of Hochbruck–Ostermann;
- **IF RK4** The fourth order Integrating Factor Runge–Kutta methd (classical RK);
- **GIF1/RK4** The fourth order Generalized Integrating Factor Runge–Kutta method;
- **GIF2/RK4** The fourth order Generalized Integrating Factor Runge–Kutta method;
- **GIF3/RK4** The fourth order Generalized Integrating Factor Runge–Kutta method;
- **CF4** The fourth order Commutator Free Lie group method with affine action;
- **CF4A1** A fourth order CF method with action corresponding to nonautonomous FVF.

Allen-Cahn equation

$$u_t = \varepsilon u_{xx} + u - u^3, \quad x \in [-1, 1]$$

with $\varepsilon = 0.01$ and with boundary and initial conditions

$$u(-1,t) = -1, \;\; u(1,t) = 1, \quad u(x,0) = .53x + .47\sin(-1.5\pi x)$$

Allen-Cahn equation

$$u_t \;=\; \varepsilon u_{xx} + u - u^3, \quad x \in [-1,1]$$

with $\varepsilon = 0.01$ and with boundary and initial conditions

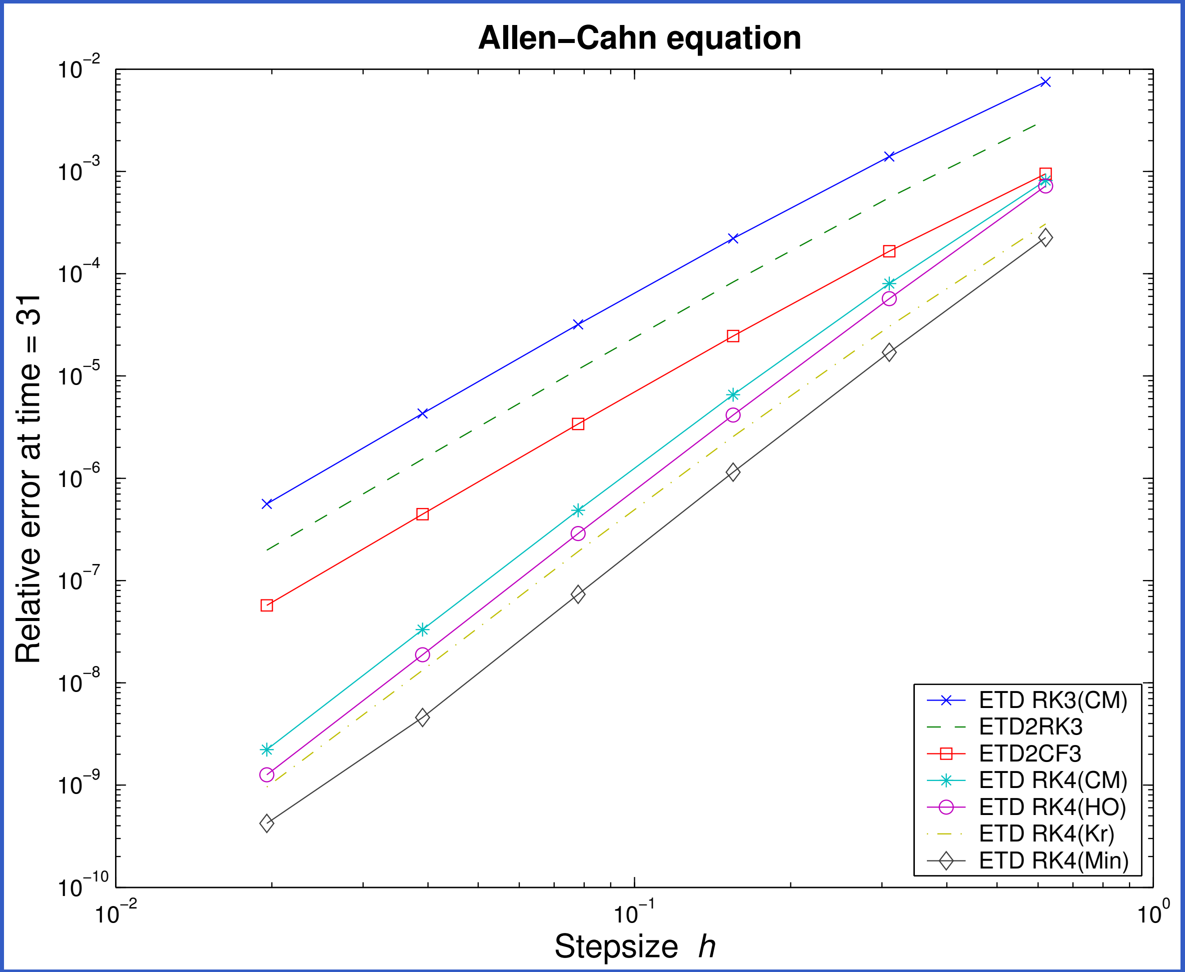
$$u(-1,t) = -1, \;\; u(1,t) = 1, \quad u(x,0) = .53x + .47\sin(-1.5\pi x)$$

After discretisation in space.

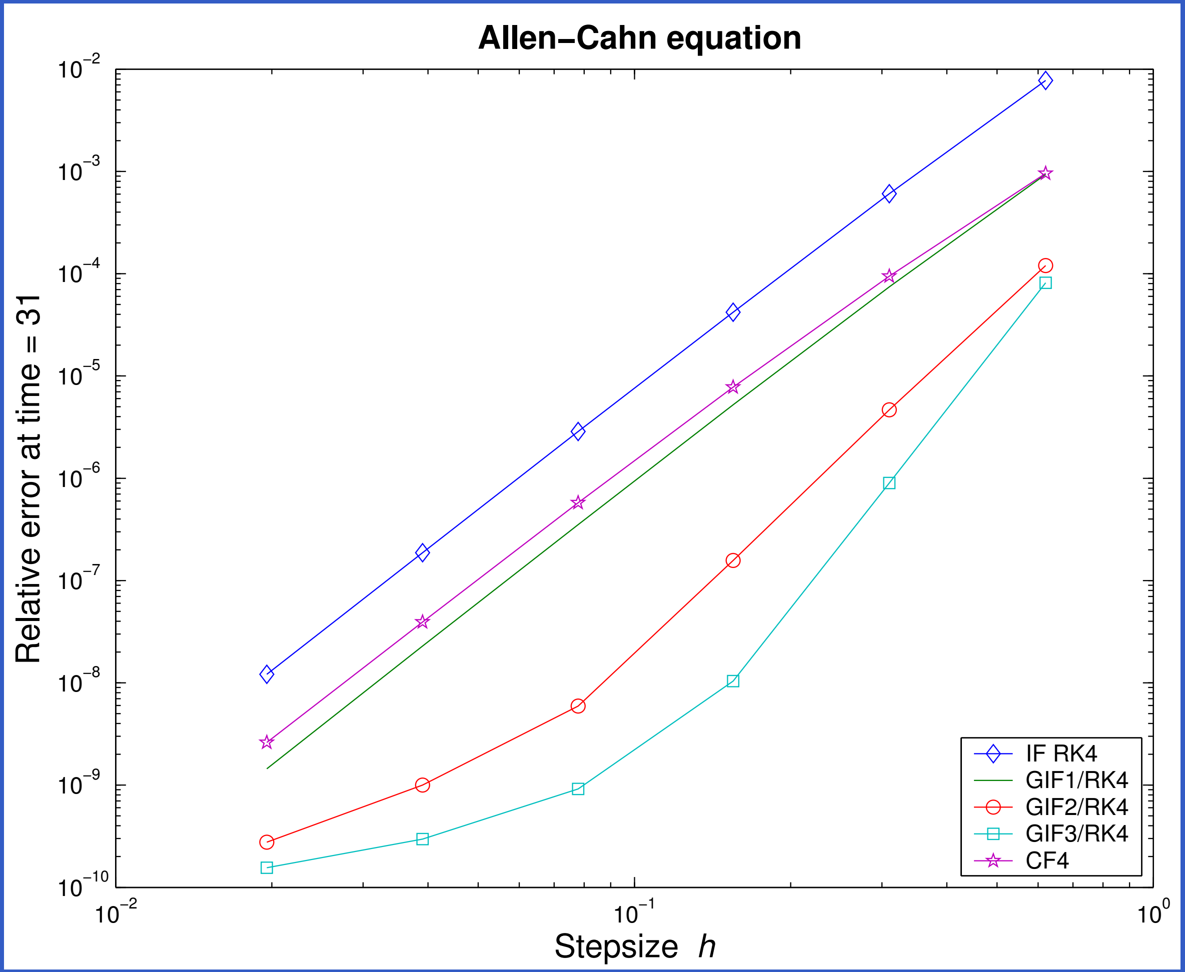
$$u_t \;=\; \mathbf{L}u + \mathbf{N}(u(t))$$

where $\mathbf{L} \;=\; \varepsilon D^2, \;\; \mathbf{N}(u(t)) \;=\; u - u^3$ and D is the Chebyshev differentiation matrix.

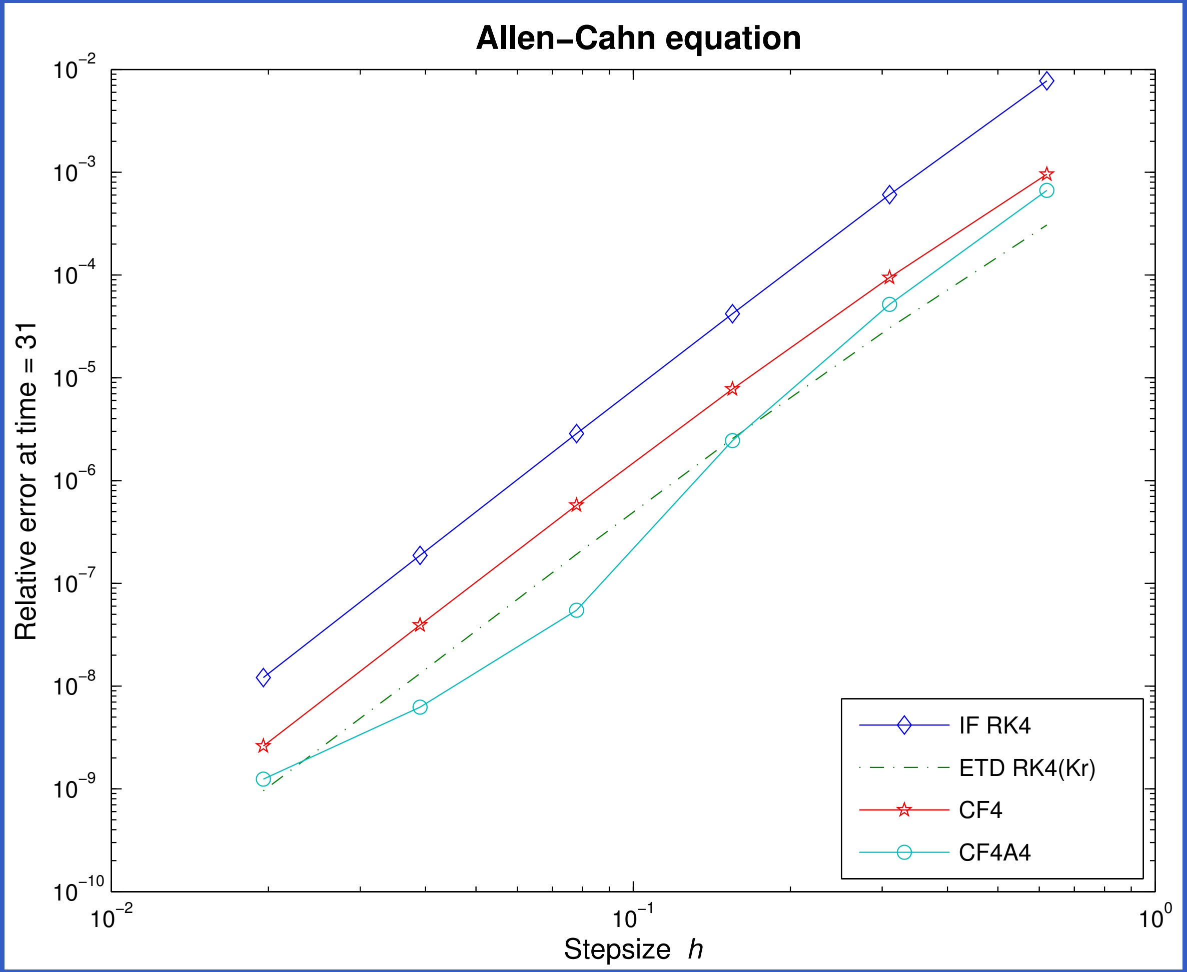
Allen-Cahn equation



Allen-Cahn equation



Allen-Cahn equation



Korteweg de Vries equation

$$u_t = -u_{xxx} - uu_x, \qquad x \in [-\pi, \pi],$$

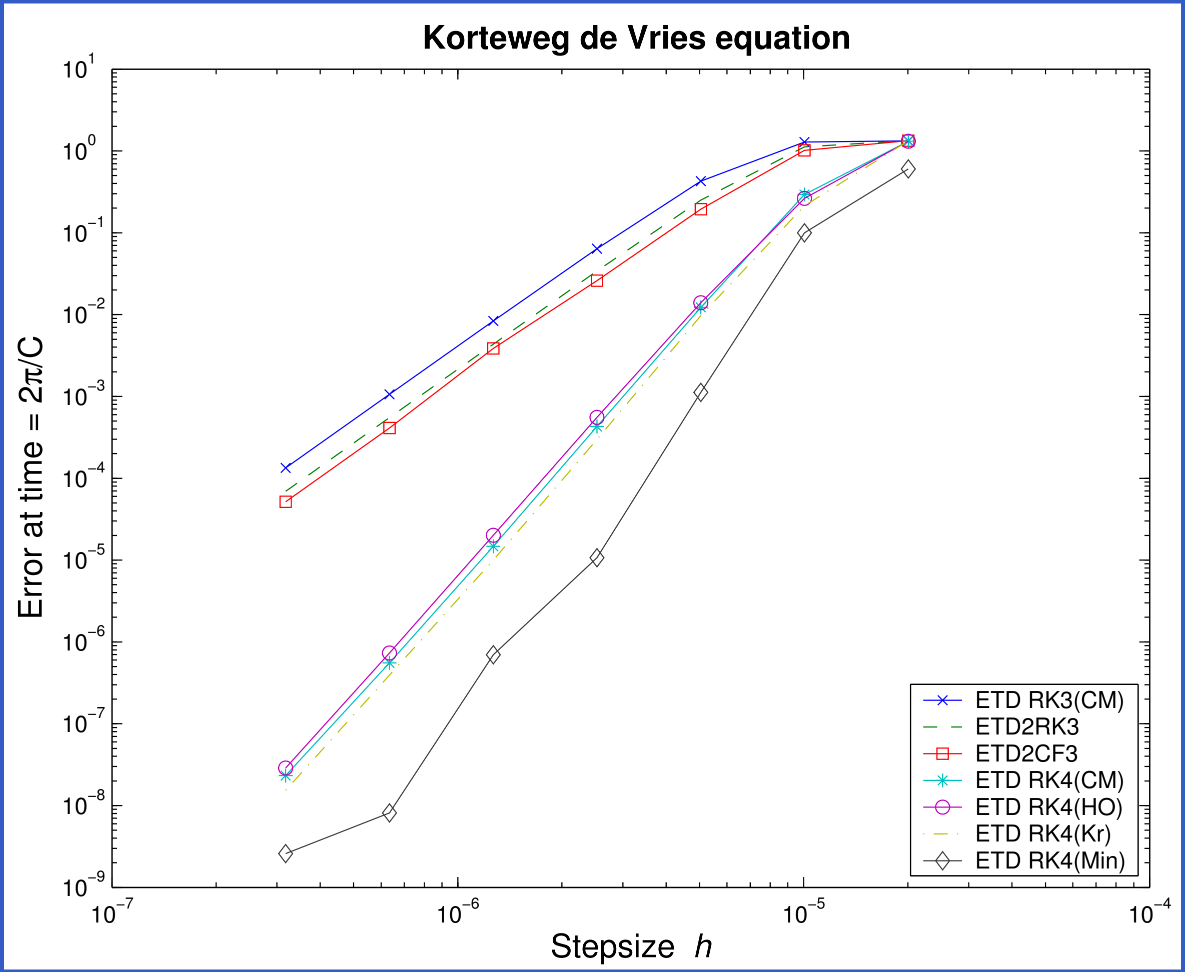
with periodic boundary conditions and with initial condition

$$u(x,0) = 3C/\cosh^2(\sqrt{C}x/2),$$

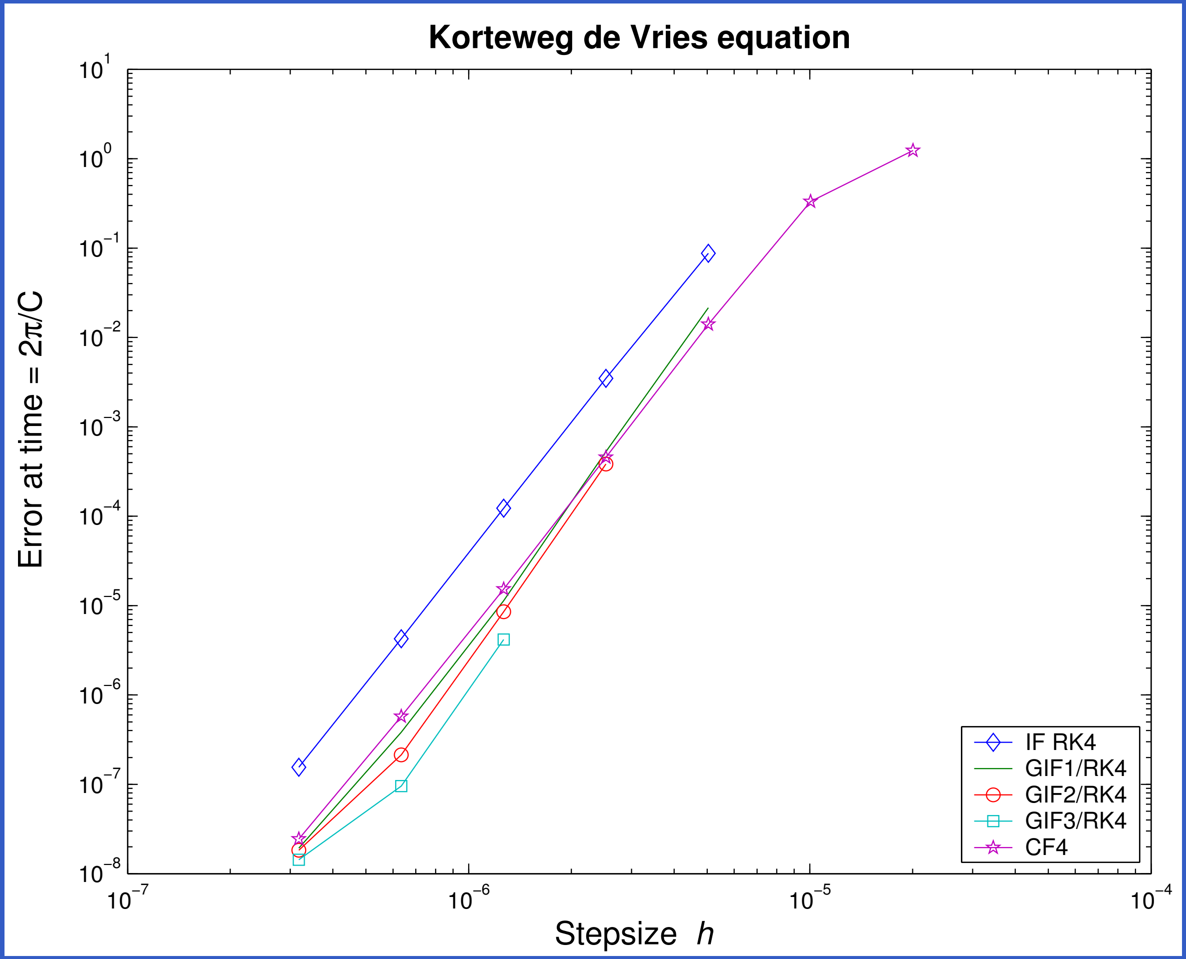
where $C = 625$. The exact solution is $2\pi/C$ periodic and is given by $u(x,t) = u(x - Ct, 0)$.

We use a 256-point Fourier spectral discretization in space. In this case the matrix L is again diagonal. The integration in time is done for one period.

Korteweg de Vries equation



Korteweg de Vries equation



Contributions

- Proposed a general class of exponential Runge-Kutta methods specifically designed for time integration of semilinear problems.
- Rederived the nonstiff order theory

  - a non-recursive rule for generating each nonstiff order condition from its corresponding rooted tree
- Extended the class of GLMs to the exponential settings
- Studied the natural connection between exponential integrators and Lie group methods.

  - $GIF/RK \equiv RKMK$

  - proposed a new approach in the derivation of GIF/RK methods

  - suggested how to construct exponential integrators with algebra action arising from the solutions of differential equations with nonautonomous frozen vector fields
- Studied different nuumerical techniques for computing the ETD $\phi^{[i]}$ functios.
- Generalized the tridiagonal reduction approach.
- Proposed new methods for solving special block linear systems.

Open problems

- Other $\phi^{[i]}$ functions.
- Effective algorithms for their computation.
- Are these methods competitive with variable stepsize - in which cases ?
- Stability analysis.
- Extensive numerical experiments.
- Exponential integrators for oscillatory problems.

References

- J. Certaine,*The solution of ordinary differential equations with large time constants,* Math. Meth. Dig. Comp., 129-132 (1960).
- J. D. Lawson, *Generalized Runge-Kutta processes for satble systems with large Lipschitz constants,* SIAM J. Num. Anal., 4, 372-380 (1967).
- S.P. Nørsett, *An A-Stable modification of the Adams-Bashforth methods,* Lecture Note in Mathematics, 109, 214-219 (1969).
- A. Friedli, *Verallgemeinerte Runge-Kutta Verfahren zur Lösung steifer Differential gleichundssysteme,* Lecture Note in Mathematics, 631, (1978).
- R. Strehmel, R. Weiner, *B-convergence results for linearly implicit one step methods,* BIT, 27, 264-282 (1987).
- G. Beylkin, J. M. Keiser, and L. Vozovoi, *A new class of time discretization schemes for the solution of nonlinear PDEs,* J. Comput. Phys. 147, 362–387 (1998).
- Y. Maday, A. T. Patera and E. M. Rønquist,*An operator-integration-factor splitting method for time dependant problems: Application to incompressible fluid flow,* J. Sci. Comput., 263-292 (1990).

References

- M. Hochbruck, Ch. Lubich, *On Krylov Subspace Approximations to the matrix exponential operator,* SIAM J. Numer.Anal., 34, 1911–1925 (1997).
- M. Hochbruck, Ch. Lubich, H. Selhofer, *Exponential integrators for large systems of differential equations,* SIAM J. Sci. Comput., 19, 1552–1574 (1998).
- H. Munthe-Kaas, High order Runge–Kutta methods on manifolds, Appl. Num. Math., 29, 115-127 (1999)
- S.M. Cox, P.C. Matthews, Exponential time differencing for stiff systems, *J. Comp. Phys.* 176, 430-455 (2002)
- A.K. Kassam, L.N. Trefethen, Fourth-order time stepping for stiff PDEs, Submitted to SIAM J. Sci. Comput. (2002)
- E. Celledoni, A. Marthinsen, B. Owren, Commutator-free lie group methods, *FGCS* 19(3), 341-352 (2003)
- S. Krogstad, Generalized integrating factor methods for stiff PDEs, available at: `http://www.ii.uib.no/∼stein`

References

- M. Hochbruck, A. Osterman, Exponential Runge-Kutta methods for parabolic problems, available at:
  `http://techmath.uibk.ac.at/numbau/alex/publications.html`
- M. Hochbruck, A. Osterman, Explicit Exponential Runge-Kutta methods for semilinear parabolic problems, available at:
  `http://techmath.uibk.ac.at/numbau/alex/publications.html`