Validated Explicit and Implicit Runge-Kutta Methods

Alexandre Chapoutot

joint work with Julien Alexandre dit Sandretto and Olivier Mullier
U2IS, ENSTA ParisTech

8th Small Workshop on Interval Methods, Praha
June 11, 2015
Initial Value Problem of Ordinary Differential Equations

Consider an IVP for ODE, over the time interval $[0, T]$

$$\dot{y} = f(y) \quad \text{with} \quad y(0) = y_0$$

IVP has a unique solution $y(t; y_0)$ if $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz in $y$ but for our purpose we suppose $f$ smooth enough i.e., of class $C^k$

Goal of numerical integration

- Compute a sequence of time instants: $t_0 = 0 < t_1 < \cdots < t_n = T$
- Compute a sequence of values: $y_0, y_1, \ldots, y_n$ such that
  $$\forall i \in [0, n], \quad y_i \approx y(t_i; y_0).$$
- s.t. $y_{n+1} \approx y(t_n + h; y_n)$ with an error $O(h^{p+1})$ where
  - $h$ is the integration step-size
  - $p$ is the order of the method
  - true with localization assumption i.e., $y_n = y(t_n; y_0)$. 
Validated solution of IVP for ODE

Goal of validated numerical integration

- Compute a sequence of time instants: \( t_0 = 0 < t_1 < \cdots < t_n = T \)
- Compute a sequence of values: \([y_0], [y_1], \ldots, [y_n]\) such that
  \[
  \forall i \in [0, n],\ [y_i] \ni y(t_i; y_0).
  \]

A two-step approach

- **Exact solution** of \( \dot{y} = f(y(t)) \) with \( y(0) \in \mathcal{Y}_0 \)
- **Safe approximation** at discrete time instants
- Safe approximation between time instants
State of the art

Taylor methods
They have been developed since 60’s (Moore, Lohner, Makino and Berz, Rhim, Jackson and Nedialkov, etc.)

- prove the existence and uniqueness: high order interval Picard-Lindelöf
- works very well on various kinds of problems:
  - non stiff and moderately stiff linear and non-linear systems,
  - with thin uncertainties on initial conditions
  - with (a writing process) thin uncertainties on parameters
- very efficient with automatic differentiation techniques
- wrapping effect fighting: interval centered form and QR decomposition
- many software: AWA, COSY infinity, VNODE-LP, CAPD, etc.

Some extensions
- Taylor polynomial with Hermite-Obreskov (Jackson and Nedialkov)
- Taylor polynomial in Chebyshev basis (T. Dzetkulic)
One question

Why bother to define new methods?
Answer 1: it may fail

A chemical reaction simulated with VNODE-LP

\[
\begin{align*}
\dot{y} &= z \\
\dot{z} &= z^2 - \frac{3}{0.001 + y^2}
\end{align*}
\]

with \[\begin{align*}
y(0) &= 10 \\
z(0) &= 0
\end{align*}\] and \(t \in [0, 50]\)

Result: it is stuck around \(t = 1\) with various order between 5 and 40.

With validated Lobatto-3C (order 4) method with tolerance \(10^{-10}\), we get in about 7.6s (Intel i7 3.4Ghz)

- \(\text{width}(y_1(50.0)) = 7.67807 \cdot 10^{-5}\)
- \(\text{width}(y_2(50.0)) = 2.338 \cdot 10^{-6}\)

Note: CAPD can solve this problem
Numerical solutions of IVP for ODEs are produced by

- Adams-Bashworth/Moulton methods
- BDF methods
- Runge-Kutta methods
- etc.

Each of these methods is adapted to a particular class of ODEs.

Runge-Kutta methods

- have strong stability properties for various kinds of problems (A-stable, L-stable, algebraic stability, etc.)
- may preserve quadratic algebraic invariant (symplectic methods)
- can produce continuous output (polynomial approximation of $y(t)$)

Can we benefit these properties in validated computations?
Examples of Runge-Kutta methods

Single-step fixed step-size explicit Runge-Kutta method

e.g. explicit Trapzoidal method (or Heun’s method)\(^1\) is defined by:

\[
k_1 = f(t_n, y_n), \quad k_2 = f(t_n + h, y_n + h k_1)
\]

\[
y_{n+1} = y_n + h \left( \frac{1}{2} k_1 + \frac{1}{2} k_2 \right)
\]

Intuition

- \(\dot{y} = t^2 + y^2\)
- \(y_0 = 0.46\)
- \(h = 1.0\)

dotted line is the exact solution.

\(^1\)example coming from “Geometric Numerical Integration”, Hairer, Lubich and Wanner.
Examples of Runge-Kutta methods

Single-step fixed step-size implicit Runge-Kutta method

e.g. Runge-Kutta Gauss method (order 4) is defined by:

\begin{align}
    k_1 &= f \left( t_n + \left( \frac{1}{2} - \frac{\sqrt{3}}{6} \right) h_n, \quad y_n + h \left( \frac{1}{4} k_1 + \left( \frac{1}{4} - \frac{\sqrt{3}}{6} \right) k_2 \right) \right) \quad (1a) \\
    k_2 &= f \left( t_n + \left( \frac{1}{2} + \frac{\sqrt{3}}{6} \right) h_n, \quad y_n + h \left( \left( \frac{1}{4} + \frac{\sqrt{3}}{6} \right) k_1 + \frac{1}{4} k_2 \right) \right) \quad (1b) \\
    y_{n+1} &= y_n + h \left( \frac{1}{2} k_1 + \frac{1}{2} k_2 \right) \quad (1c)
\end{align}

Remark: A non-linear system of equations must be solved at each step.
Runge-Kutta methods

s-stage Runge-Kutta methods are described by a Butcher tableau

| \(c_1\) | \(a_{11}\) \(a_{12}\) \(\cdots\) \(a_{1s}\) |
| \(\vdots\) | \(\vdots\) \(\vdots\) \(\vdots\) |
| \(c_s\) | \(a_{s1}\) \(a_{s2}\) \(\cdots\) \(a_{ss}\) |
| \(b_1\) | \(b_2\) \(\cdots\) \(b_s\) |
| \(b'_1\) | \(b'_2\) \(\cdots\) \(b'_s\) (optional) |

Which induces the following recurrence:

\[
\begin{align*}
    k_i &= f \left( t_n + c_i h_n, \ y_n + h \sum_{j=1}^{s} a_{ij} k_j \right) \\
    y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_i
\end{align*}
\]  \hspace{1cm} (2)

- **Explicit** method (ERK) if \(a_{ij} = 0\) is \(i \leq j\)
- **Diagonal Implicit** method (DIRK) if \(a_{ij} = 0\) is \(i \leq j\) and at least one \(a_{ii} \neq 0\)
- **Implicit** method (IRK) otherwise
Validated Runge-Kutta methods

Challenges

1. Computing with sets of values taking into account dependency problem and wrapping effect;
2. Bounding the approximation error of Runge-Kutta formula.

Our approach

- **Problem 1** is solved using **affine arithmetic** avoiding centered form and QR decomposition
- **Problem 2** is solved by bounding the **Local truncation error** of Runge-Kutta method based on **B-series**

We focus on Problem 2 in this talk
Order condition for Runge-Kutta methods

Method order of Runge-Kutta methods and Local Truncation Error (LTE)

\[ y(t_n; y_{n-1}) - y_n = C \cdot O(h^{p+1}) \quad \text{with} \quad C \in \mathbb{R}. \]

we want to bound this!

Order condition
This condition states that a method of Runge-Kutta family is of order \( p \) iff

- the Taylor expansion of the exact solution
- and the Taylor expansion of the numerical methods

have the same \( p + 1 \) first coefficients.

Consequence
The LTE is the difference of Lagrange remainders of two Taylor expansions

...but how to compute it?
A quick view of Runge-Kutta order condition theory\(^2\)

Starting from \(y^{(q)} = (f(y))^{(q-1)}\) and with the Chain rule, we have

**High order derivatives of exact solution \(y\)**

\[
\begin{align*}
\dot{y} &= f(y) \\
\ddot{y} &= f'(y)\dot{y} \\
y^{(3)} &= f'''(y)(\dot{y}, \dot{y}) + f'(y)\ddot{y} \\
y^{(4)} &= f'''(y)(\dot{y}, \dot{y}, \dot{y}) + 3f''(y)(\ddot{y}, \dot{y}) + f'(y)y^{(3)} \\
y^{(5)} &= f^{(4)}(y)(\dot{y}, \dot{y}, \dot{y}, \dot{y}) + 6f'''(y)(\ddot{y}, \ddot{y}, \dot{y}) \\
&\quad + 4f''(y)(y^{(3)}, \dot{y}) + 3f''(y)(\dddot{y}, \ddot{y}) + f'(y)y^{(4)} \\
&\quad + \cdots
\end{align*}
\]

\(^2\)strongly inspired from “Geometric Numerical Integration”, Hairer, Lubich and Wanner.
Inserting the value of $\dot{y}$, $\ddot{y}$, ..., we have:

**High order derivatives of exact solution $y$**

\[
\begin{align*}
\dot{y} &= f \\
\ddot{y} &= f'(f) \\
y^{(3)} &= f'''(f, f) + f'(f'(f)) \\
y^{(4)} &= f'''(f, f, f) + 3f''(f'f, f) + f''(f''(f, f)) + f'(f'(f'(f))) \\
& \vdots
\end{align*}
\]

- Elementary differentials, are denoted by $F(\tau)$

**Remark** a tree structure is made apparent in these computations

---

\(^2\)strongly inspired from “Geometric Numerical Integration”, Hairer, Lubich and Wanner.
Rooted trees

- $f$ is a leaf
- $f'$ is a tree with one branch, \ldots, $f^{(k)}$ is a tree with $k$ branches

Example

$$f''(f'f, f)$$

is associated to

\[ f \quad f' \quad f'' \]

Remark: this tree is not unique e.g., symmetry

\[\overset{2}{\text{strongly inspired from}} \ “\text{Geometric Numerical Integration}, \text{ Hairer, Lubich and Wanner.} \]
A quick view of Runge-Kutta order condition theory\textsuperscript{2}

**Theorem 1 (Butcher, 1963)**

The \( q \)th derivative of the \textbf{exact solution} is given by

\[
y^{(q)} = \sum_{r(\tau) = q} \alpha(\tau) F(\tau)(y_0) \quad \text{with} \quad r(\tau) \text{ the order of } \tau \text{ i.e., number of nodes}
\]

\[
\alpha(\tau) \text{ a positive integer}
\]

We can do the same for the numerical solution

**Theorem 2 (Butcher, 1963)**

The \( q \)th derivative of the \textbf{numerical solution} is given by

\[
y_{1}^{(q)} = \sum_{r(\tau) = q} \gamma(\tau) \phi(\tau) \alpha(\tau) F(\tau)(y_0) \quad \text{with} \quad \gamma(\tau) \text{ a positive integer}
\]

\[
\phi(\tau) \text{ depending on a Butcher tableau}
\]

**Theorem 3, order condition (Butcher, 1963)**

A Runge-Kutta method has order \( p \) iff

\[
\phi(\tau) = \frac{1}{\gamma(\tau)} \quad \forall \tau, r(\tau) \leq p
\]

\textsuperscript{2}strongly inspired from “Geometric Numerical Integration”, Hairer, Lubich and Wanner.
LTE formula for explicit and implicit Runge-Kutta

From Theorem 1 and Theorem 2, if a Runge-Kutta has order \( p \) then

\[
y(t_n; y_{n-1}) - y_n = \frac{h^{p+1}}{(p + 1)!} \sum_{r(\tau) = p+1} \alpha(\tau) [1 - \gamma(\tau) \phi(\tau)] F(\tau)(y(\xi))
\]

\[\xi \in [t_{n-1}, t_n]\]

- \( \alpha(\tau) \) and \( \gamma(\tau) \) are positive integer (with some combinatorial meaning)
- \( \phi(\tau) \) function of the coefficients of the RK method,

Example

\[\phi(\cdot)\] is associated to

\[
\sum_{i,j=1}^{s} b_i a_{ij} c_j \quad \text{with} \quad c_j = \sum_{k=1}^{s} a_{jk}
\]

Note: \( y(\xi) \) may be over-approximated using Interval Picard-Lindelöf operator.
Implementation of LTE formula

Elementary differentials

\[ F(\tau)(y) = f^{(m)}(y) \left( F(\tau_1)(y), \ldots, F(\tau_m)(y) \right) \quad \text{for} \quad \tau = [\tau_1, \ldots, \tau_m] \]

translate as a sum of partial derivatives of \( f \) associated to sub-trees

Notations

- \( n \) the state-space dimension
- \( p \) the order of a Rung-Kutta method

Two ways of computing \( F(\tau) \)

1. **Direct form** (current): complexity \( O(n^{p+1}) \)

2. **Factorized form** (under test): complexity \( O(n(p + 1)^{\frac{5}{2}}) \)
   
   based on the work of Ferenc Bartha and Hans Munthe-Kaas
   "Computing of B-series by automatic differentiation", 2014
Experimentation

Toy example

\[
\begin{pmatrix}
\dot{y}_1 \\
\dot{y}_2
\end{pmatrix} = \begin{pmatrix}
-y_2 \\
y_1
\end{pmatrix}
\text{ with } \begin{pmatrix}
y_1(0) = [0, 0.1] \\
y_2(0) = [0.95, 1.05]
\end{pmatrix}
\]

Validated RK4 method with tolerance $10^{-8}$ we get in about 3s (Intel i7 3.4Ghz)

- $\text{width}(y_1(100.0)) = 0.146808$
- $\text{width}(y_2(100.0)) = 0.146902$
Experimentation

Usefulness of affine arithmetic

\[
\begin{align*}
\dot{y}_1 &= 1, & y_1(0) &= 0 \\
\dot{y}_2 &= y_3, & y_2(0) &= 0 \\
\dot{y}_3 &= \frac{1}{6}y_2^3 - y_2 + 2 \sin(p \cdot y_1) \quad \text{with} \quad p \in [2.78, 2.79], & y_3(0) &= 0.
\end{align*}
\]

Validated RK4 method with tolerance $10^{-6}$ we get in about 2.3s (Intel i7 3.4Ghz)

- $\text{width}(y_1(10.0)) = 7.10543 \cdot 10^{-15}$
- $\text{width}(y_2(10.0)) = 6.11703$
- $\text{width}(y_3(10.0)) = 7.47225$

**Note:** none of the method in the Vericomp benchmark can reach 10s

**Note 2:** CAPD can solve it
Experimentation

Based on Vericomp benchmark\(^3\) (around 70 problems)

\[
\text{IVP} \rightarrow \text{non-stiff (P.I)} \rightarrow \text{complicate (C)} \rightarrow \text{moderate (B)} \rightarrow \text{simple (A)} \rightarrow \text{Uncertain (U) or not}
\]

\[
\text{linear (L)} \rightarrow \text{idem}
\]

with the following metrics:

- c5t: user time taken to simulate the problem for 1 second.
- c5w: the final diameter of the solution (infinity norm is used).
- c6t: the time to breakdown the method with a maximal limit of 10 seconds.
- c6w: the diameter of the solution a the breakdown time.

\(^3\)http://vericomp.inf.uni-due.de/
Vnode-LP: order 15, 20, 25 (tolerances $10^{-14}$)

RK4, LC3, LA3: tolerances $10^{-8}$ to $10^{-14}$ (order 4)
▶ Vnode-LP: order 15, 20, 25 (tolerances $10^{-14}$)
▶ RK4, LC3, LA3: tolerances $10^{-8}$ to $10^{-14}$ (order 4)
Conclusion

We presented a new approach to validate Runge-Kutta methods

- a new formula to compute LTE based on B-series
- fully parametrized by a Butcher tableau
- affine arithmetic avoiding QR decomposition

Implementation as a plugin of IBEX, code name DynIbex, available at http://perso.ensta-paristech.fr/~chapoutot/dynibex/

Future work

- finish testing the implementation of LTE with automatic differentiation
- implement new a priori enclosure methods based on Runge-Kutta
- define new methods mixing different Runge-Kutta in one simulation
- solve new IVP problems such as for DAE (next talk) or DDE
BACKUP
Note on the number of trees (up to order 11 (left)):

Number of Rooted Trees
1842 719 286 115 48 20 9 4 2 1 1 (total 3047)
Taylor series development of $y(t)$ (assume $y(t_n) \in [y_n]$)

$$y(t_{n+1}) = y(t_n) + \sum_{i=1}^{N-1} \frac{h^i}{i!} \frac{d^i y}{dt^i}(t_n) + \frac{h_{n+1}^N}{N!} \frac{d^N y}{dt^N}(t')$$

$$\in [y_n] + \sum_{i=1}^{N-1} h^i f[i-1](y(t_n)) + h^N f[N-1](y(t'))$$

$$\in [y_n] + \sum_{i=1}^{N-1} h^i f[i-1](\tilde{y}_n) + h^N f[N-1](\tilde{y}_n) \triangleq [y_{n+1}]$$

Challenges

- Computation of $[\tilde{y}_n]$ such that $\forall t \in [t_n, t_{n+1}], y(t) \in [\tilde{y}_n]$
  **Solution**: interval Picard-Lindelöf operator

- With that formula: $\text{width}([y_{n+1}]) \geq \text{width}([y_n])$
  **Solutions**: interval centered form + QR decomposition
Single-step variable step-size explicit Runge-Kutta method

e.g. Bogacki-Shampine (ode23) is defined by:

\[ k_1 = f(t_n, y_n) \]
\[ k_2 = f\left(t_n + \frac{1}{2} h_n, y_n + \frac{1}{2} h k_1\right) \]
\[ k_3 = f\left(t_n + \frac{3}{4} h_n, y_n + \frac{3}{4} h k_2\right) \]
\[ y_{n+1} = y_n + h \left( \frac{2}{9} k_1 + \frac{1}{3} k_2 + \frac{4}{9} k_3 \right) \]
\[ k_4 = f\left(t_n + 1 h_n, y_{n+1}\right) \]
\[ z_{n+1} = y_n + h \left( \frac{7}{24} k_1 + \frac{1}{4} k_2 + \frac{1}{3} k_3 + \frac{1}{8} k_4 \right) \]

Remark: the step-size \( h \) is adapted following \( \| y_{n+1} - z_{n+1} \| \leq \text{tol} \)
Gauss-Legendre methods

Single-step fixed step-size implicit Runge-Kutta method

e.g. Runge-Kutta Gauss method (order 4) is defined by:

\[
\begin{align*}
    k_1 &= f \left( t_n + \left( \frac{1}{2} - \frac{\sqrt{3}}{6} \right) h_n, \right) \\
    k_2 &= f \left( t_n + \left( \frac{1}{2} + \frac{\sqrt{3}}{6} \right) h_n, \right) \\
    y_{n+1} &= y_n + h \left( \frac{1}{2} k_1 + \frac{1}{2} k_2 \right)
\end{align*}
\]  

(3a)  

\[
\begin{align*}
    y_n + h \left( \frac{1}{4} k_1 + \left( \frac{1}{4} - \frac{\sqrt{3}}{6} \right) k_2 \right) \\
    y_n + h \left( \left( \frac{1}{4} + \frac{\sqrt{3}}{6} \right) k_1 + \frac{1}{4} k_2 \right)
\end{align*}
\]  

(3b)  

\[
\begin{align*}
    y_{n+1} &= y_n + h \left( \frac{1}{2} k_1 + \frac{1}{2} k_2 \right)
\end{align*}
\]  

(3c)

Remark: A non-linear system of equations must be solved at each step.
Note on building IRK Gauss’ method

\[ \dot{y} = f(y) \quad \text{with} \quad y(0) = y_0 \iff y(t) = y_0 + \int_{t_n}^{t_{n+1}} f(y(s)) \, ds \]

We solve this equation using quadrature formula.

IRK Gauss method is associated to a **collocation method** (polynomial approximation of the integral) such that for \( i, j = 1, \ldots, s \):

\[
a_{ij} = \int_0^{c_i} \ell_j(t) \, dt \quad \text{and} \quad b_j = \int_0^1 \ell_j(t) \, dt
\]

with \( \ell_j(t) = \prod_{k \neq j} \frac{t - c_k}{c_j - c_k} \) the **Lagrange polynomial**.

And the \( c_i \) are chosen as the solution of the **Shifted Legendre polynomial** of degree \( s \):

\[
P_s(x) = (-1)^s \sum_{k=0}^{s} \binom{s}{k} \binom{s + k}{s} (-x)^k
\]

Example: 1, 2x – 1, 6x^2 – 6x + 1, 20x^3 – 30x^2 + 12x – 1, etc.