Génération et validation des méthodes de Runge-Kutta

Overview of four years of study
Alexandre Chapoutot, Julien Alexandre dit Sandretto
Contents

Context

Interval Analysis
  Arithmetic and sets
  Constraint Satisfaction Problem

Validated Simulation
  Simulation of IVP
  Validated simulation

Validated Runge-Kutta Methods
  Validated schemes
  Local Truncation Error
  Computation of validated RK
  Examples

Differential constraint satisfaction problems

Constraint Programming and Runge-Kutta
  CSP to define RK
  Experimentations
  Cost function to define optimal schemes
  Experimentations
  Properties

Conclusion
Robot’s behavior

A mobile robot...

- ...moves! ⇒ Dynamical system (discrete, continuous, switched, hybrid, etc)
- drived by actuators ⇒ control
- w.r.t. sensors and design parameters ⇒ uncertainties
- for critical tasks! (in our case)

Control: synthesis, analysis, verification

Two antinomic facts: reliable results under uncertainties!
Interval Analysis: the suitable tool

$[x] = [x, \bar{x}]$ stands for the set of reals $x$ s.t. $x \leq x \leq \bar{x}$

Arithmetic

Extension of operators ($+, -, *, /, \sin, \cos, ...$), e.g. $[-1, 1] + [1, 3] = [0, 4]$

Rounding error handled ($1/3 \in [0.33333333, 0.33333334]$)

Extension of function

$[f](\langle x \rangle) \supset f([x]) = \{f(y) | y \in [x]\}$

Interval Integral

Rectangle rule: $\int_{[x]} f(x')dx' \in [f([x])] \cdot w([x])$
Interval Analysis: the suitable tool

\[ [x] = [x, \bar{x}] \text{ stands for the set of reals } x \text{ s.t. } x \leq x \leq \bar{x} \]

Arithmetic

Extension of operators (\(+, -, \times, /, \sin, \cos, \ldots\) ), e.g. \([-1, 1] + [1, 3] = [0, 4]\)

Rounding error handled (1/3 \in 0.33333333[3, 4])

Extension of function

\[ [f](\{[x]\}) \supset f([x]) = \{f(y) | y \in [x]\} \]

Interval Integral

Rectangle rule: \( \int_{[x]} f(x')dx' \in [f([x])] \cdot w([x]) \)
Interval Analysis: the suitable tool

\[ [x] = [x, \bar{x}] \] stands for the set of reals \( x \) s.t. \( x \leq x \leq \bar{x} \)

Arithmetic

Extension of operators (\(+, -, *, /, \sin, \cos, \ldots\))

\([-1, 1] + [1, 3] = [0, 4]\)

Rounding error handled (\(1/3 \in 0.33333333[3, 4]\))

Extension of function

\[[f]([x]) \supset f([x]) = \{ f(y) | y \in [x] \}\]

Interval Integral

Rectangle rule: \( \int_{[x]} f(x') dx' \in [f([x])].w([x]) \)
**Interval Analysis: the suitable tool**

$[x] = [\underline{x}, \overline{x}]$ stands for the set of reals $x$ s.t. $\underline{x} \leq x \leq \overline{x}$

**Arithmetic**

Extension of operators ($+, -, \times, \div, \sin, \cos, ...$), e.g. $[-1, 1] + [1, 3] = [0, 4]$

Rounding error handled ($1/3 \in 0.33333333[3, 4]$)

**Extension of function**

$[f](\{x\}) \supset f([x]) = \{f(y) | y \in [x]\}$

**Interval Integral**

Rectangle rule: $\int_{[x]} f(x')dx' \in [f([x])).w([x])$
A classical problem
find $x$ s.t. $f(x) = 0$ assuming $x \in [x]$.

More generally, a CSP is
- a set of variables $\mathcal{V}$
- a set of domains $\mathcal{D}$
- a set of constraints $\mathcal{C}$. 

Interval Analysis  Constraint Satisfaction Problem
Constraint Satisfaction Problem (CSP)

Branch & Prune for $f(x) = 0$

A simple method
Interval arithmetic + bisection strategy
- if $0 \not\in [f([x])]$ then no possible solution in $[x]$
- if $0 \in [f([x])]$ then maybe one solution in $[x]$

Bisection up to a given limit
Constraint Satisfaction Problem (CSP)

Branch & Prune for $f(x) = 0$

A simple method
Interval arithmetic + bisection strategy
- if $0 \not\in [f](x)$ then no possible solution in $x$
- if $0 \in [f](x)$ then maybe one solution in $x$

Bisection up to a given limit
Constraint Satisfaction Problem (CSP)

Branch & Prune for $f(x) = 0$

A simple method

Interval arithmetic + bisection strategy

- if $0 \not\in [f([x])]$ then no possible solution in $[x]$
- if $0 \in [f([x])]$ then maybe one solution in $[x]$

Bisection up to a given limit
Constraint Satisfaction Problem (CSP)

Branch & Prune for $f(x) = 0$

A simple method

Interval arithmetic + bisection strategy

- if $0 \notin [f]([x])$ then no possible solution in $[x]$
- if $0 \in [f]([x])$ then maybe one solution in $[x]$

Bisection up to a given limit
Constraint Satisfaction Problem (CSP)

Branch & Prune for \( f(x) = 0 \)

A simple method

Interval arithmetic + bisection strategy

- if \( 0 \not\in [f([x])] \) then no possible solution in \([x]\)
- if \( 0 \in [f([x])] \) then maybe one solution in \([x]\)

Bisection up to a given limit
Constraint Satisfaction Problem (CSP)

*Branch & Prune for* $f(x) = 0$

A simple method

Interval arithmetic + bisection strategy

- if $0 \notin [f([x])]$ then no possible solution in $[x]$
- if $0 \in [f([x])]$ then maybe one solution in $[x]$

Bisection up to a given limit

Some improvements are available [1]

Simulation of IVP

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 $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is Lipschitz in $y$
but for our purpose we suppose $f$ smooth enough, i.e., of class $C^k$

Numerical integration

Approximate the solution:

► Compute a sequence of time instants: $t_0 = 0 < t_1 < \cdots < t_n = T$
  (with a stepsize controller)
► Compute a sequence of values: $y_0, y_1, \ldots, y_n$ such that

$$\forall i \in \{0, \ldots, n\}, \quad y_i \approx y(t_i; y_0)$$
Validated Simulation

Goal of validated numerical integration

- Same discretization approach
- Compute a sequence $[\tilde{y}_0], [\tilde{y}_1], \ldots, [\tilde{y}_{n-1}]$ such that

$$\forall i \in \{0, \ldots, n\}, \quad y(t; y_0) \in [\tilde{y}_i], \forall t \in [t_i, t_{i+1}],$$

- and a sequence of values: $[y_0], [y_1], \ldots, [y_n]$ such that

$$\forall i \in \{0, \ldots, n\}, \quad y(t_i; y_0) \in [y_i].$$
Validated Simulation

A two-step approach

Exact solution of
\[ \dot{y} = f(y(t)), y(0) \in [y_0] \]

Safe approximation at discrete time instants \([y_i]\), obtained with Taylor (Capd, Vnode) or Runge-Kutta (DynIbex) validated methods

Safe approximation between time instants \([\tilde{y}_i]\), obtained with a Picard-Lindelöf operator
Validated Simulation

Picard-Lindelöf operator

Formal solution of ODE: \( y_{n+1} = y_n + \int_0^h f(s)ds \)

Following the rectangle rule, based on Brouwer’s theorem, the Picard-Lindelöf operator is defined such that:

\[
P([x]) = y_n + [0, h][f][[x]]
\]

- If \( P([x]) \subset Int([x]) \), then ODE admits one and only one solution and this solution is in \( [x], \forall s \in [0, h] \) (even in \( P([x]) \))
  \[\Rightarrow \text{ and } [\tilde{y}] = [x]\]
- Otherwise \( [x] \) is inflated, or \( h \) is reduced

Remarks: the rectangle rule can be replaced by any validated scheme (Taylor series \(^1\) or RK)

State of the art

Taylor methods

They have been developed since 60’s (Moore, Lohner, Makino and Berz, Corliss and Rhim, Neher et al., 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)
History on Interval Runge-Kutta methods

- Andrzej Marciniak et al. work on this topic since 1999
  "The form of $\psi(t, y(t))$ is very complicated and cannot be written in a general form for an arbitrary $p$"

  The implementation OOIRK is not freely available.

- Hartmann and Petras, ICIAM 1999
  No more information than an abstract of 5 lines.

- Bouissou and Martel, SCAN 2006 (only RK4 method)
  Implementation GRKLib is not available.

- Bouissou, Chapoutot and Djoudi, NFM 2013 (any explicit RK)
  Implementation is not available.

- Alexandre dit Sandretto and Chapoutot, 2016 (any explicit and implicit RK)
  Implementation DynIBEX is open-source, combine with IBEX.
Validated Runge-Kutta methods

A validated algorithm

\[ [y_{\ell+1}] = [\Phi](h, [y_\ell]) + \text{Error of } \Phi . \]
Validated Runge-Kutta Methods

How validate a RK method?
Order of Runge-Kutta methods and Local Truncation Error (LTE)

\[ LTE = y(t_\ell; y_{\ell-1}) - y_\ell = C \cdot h^{p+1} \] with \( C \in \mathbb{R} \).

We need to bound the LTE with guarantee...

Order condition
This condition states that a method of RK 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 2 Taylor expansions
Validated Runge-Kutta Methods

Theorem 1 (Butcher, 1963)

The qth derivative of the exact solution is given by

\[ y^{(q)} = \sum_{r(\tau) = q} \alpha(\tau) F(\tau)(y_0) \]

with \( r(\tau) \) the order of the rooted tree \( \tau \)

\( \alpha(\tau) \) a positive integer

\( F(\tau)(.) \) elementary differential for \( \tau \)

We can do the same for the numerical solution:

Theorem 2 (Butcher, 1963)

The qth derivative of the numerical solution is given by

\[ y_{1}^{(q)} = \sum_{r(\tau) = q} \gamma(\tau) \phi(\tau) \alpha(\tau) F(\tau)(y_0) \]

with \( \gamma(\tau) \) a positive integer

\( \phi(\tau) \) depending on a Butcher tableau

Theorem 3, order condition (Butcher, 1963)

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

\[ \phi(\tau) = \frac{1}{\gamma(\tau)} \quad \forall \tau, r(\tau) \leq p \]
LTE formula for **explicit and implicit** Runge-Kutta

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

\[
y(t_1; y_0) - y_1 = \frac{h^{p+1}}{(p+1)!} \sum_{r(\tau) = p+1} \alpha(\tau) \left[ 1 - \gamma(\tau) \phi(\tau) \right] F(\tau)(y(\xi)), \quad \xi \in [t_1, t_0]
\]

**Remark**

In theory, bound the LTE of a Runge-Kutta is a simpler problem:

- for each method the Butcher tableau and the order available
- \( y(\xi) \) is enclosed by \([\tilde{y}]\) using Picard-Lindelöf operator

**But complex in practice !**

Two methods: direct form (symbolic derivatives and trees\(^1\)) or factorized (automatic differentiation and graphs\(^2,3\))

---

\(^1\) Alexandre dit Sandretto et al., “Validated explicit and implicit Runge-Kutta methods”, Reliable Computing 2016
\(^2\) Bartha et al., “Computing of B-series by automatic differentiation”, Discrete and continuous dynamical systems, 2014
\(^3\) Mullier et al., “Validated Computation of the Local Truncation Error of Runge-Kutta Methods with Automatic Differentiation”, AD 2016
Validated Runge-Kutta Methods

⇒ LTE can be bounded, but...

It remains to compute the RK scheme itself:

- **Explicit RK**: evaluation of $f$ with intervals
  Heun’s scheme:

  $[k_1] = [f](t_n, [y_n])$,  $[k_2] = [f](t_n + 1h, [y_n] + h1[k_1])$

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

- **Implicit RK**: need to solve a system
  Gauss order 4:

  $[k_1] = [f] \left( t_n + \left( \frac{1}{2} - \frac{\sqrt{3}}{6} \right) h \right)$,  $[y_n] + h \left( \frac{1}{4} [k_1] + \left( \frac{1}{4} - \frac{\sqrt{3}}{6} \right) [k_2] \right)$

  $[k_2] = [f] \left( t_n + \left( \frac{1}{2} + \frac{\sqrt{3}}{6} \right) h \right)$,  $[y_n] + h \left( \left( \frac{1}{4} + \frac{\sqrt{3}}{6} \right) [k_1] + \frac{1}{4} [k_2] \right)$

  $[y_{n+1}] = [y_n] + h \left( \frac{1}{2} [k_1] + \frac{1}{2} [k_2] \right)$
Validated Runge-Kutta Methods

Solve a problem with interval analysis: contraction technique**!

\[ [k_1] = [k_2] = [k_3] = [k_4] = [\tilde{y}] \]

Then, we repeat:

\[
[k_1] = [k_1] \cap [f] \left( [y_n] + h \left( \frac{1}{4} [k_1] + \left( \frac{1}{4} - \frac{\sqrt{3}}{6} \right) [k_2] \right) \right)
\]

\[
[k_2] = [k_2] \cap [f] \left( [y_n] + h \left( \left( \frac{1}{4} + \frac{\sqrt{3}}{6} \right) [k_1] + \frac{1}{4} [k_2] \right) \right)
\]

**f is contracting on [\tilde{y}] because of Picard-Lindelöf success (if \( c_i \leq 1 \))...
Examples

Provides a tube, abstracted by a list of boxes \(([y_i], [\dot{y}_i])\):
Initial states: \(y(0) = (0; -10.3; 0.03)\), some parameters:
\(a = 0.2, b = 0.2, c = 5.7\)

The differential system: \(\dot{y} = \begin{cases} 
-(y_1 + y_2) \\
y_0 + a \cdot y_1 \\
b + y_2 \cdot (y_0 - c)
\end{cases}\)
Examples

A chemical reaction simulated (stiff)

\[
\begin{cases}
\dot{y} = z \\
\dot{z} = z^2 - \frac{3}{0.0001 + y^2}
\end{cases}
\]

with

\[
\begin{cases}
y(0) = 10 \\
z(0) = 0
\end{cases}
\]

and \( t \in [0, 50] \)

Result: Taylor based tools fail around \( t = 1 \) (order 5 to 40).

With validated Lobatto-IIIC (order 4), tolerance \( 10^{-10} \), solved in 7.6s
Examples

Van Der Pol 50s

Initial states: \( y(0) = (2, 0) \), One parameter: \( \mu = 1.0 \text{ or } 2.0 \)

\[
\dot{y} = \begin{cases} 
    y_1 \\
    \mu \ast (1 - y_0^2) \ast y_1 - y_0 
\end{cases}
\]
Examples

Volterra 6s

Initial states: \( y(0) = (1.0; 3.0) \)

The differential system: \( \dot{y} = \begin{cases} 2 \times y_0 \times (1 - y_1) \\ -y_1 \times (1 - y_0) \end{cases} \)
Examples

Circle 100s

Initial states: \( y(0) = ([0, 0.1]; [0.95, 1.05]) \)

The differential system: \( \dot{y} = \begin{cases} -y_1 \\ y_0 \end{cases} \)
A general settings of dynamical systems

\[ S \equiv \left\{ \begin{align*}
\dot{y}(t) &= f(t, y(t), x(t), p), \\
0 &= g(t, y(t), x(t)) \\
0 &= h(y(t), x(t))
\end{align*} \right. \]

we denote by

\[ Y(T, Y_0, P) = \{ y(t; y_0, p) : t \in T, y_0 \in Y_0, p \in P \} \]

the set of solutions
Example of ODEs with constraints

Production-Destruction systems based on an ODE with parameter $a = 0.3$

\[
\begin{pmatrix}
\dot{y}_0 \\
\dot{y}_1 \\
\dot{y}_2
\end{pmatrix} = \begin{pmatrix}
\frac{-y_0 y_1}{1 + y_0} \\
\frac{y_0 y_1}{1 + y_0} - ay_1 \\
\end{pmatrix}
\]

and associated to constraints:

\[
y_0 + y_1 + y_2 = 10.0
\]

\[
y_0 \geq 0
\]

\[
y_1 \geq 0
\]

\[
y_2 \geq 0
\]

Initial values, for $t \in [0, 100]$, are

\[
\begin{pmatrix}
y_0(0) \\
y_1(0) \\
y_2(0)
\end{pmatrix} = \begin{pmatrix}9.98 \\0.01 \\
0.01
\end{pmatrix}
\]
ODEs with constraints in DynIBEX – results
Constraint Satisfaction Differential Problems

CSDP

Let $S$ be a differential system and $t_{\text{end}} \in \mathbb{R}^+$ the time limit. A CSDP is a NCSP defined by

- a finite set of variables $\mathcal{V}$ including the parameters of the differential systems $S_i$, i.e., $(y_0, p)$, a time variable $t$ and some other algebraic variables $q$;

- a domain $\mathcal{D}$ made of the domain of parameters $p : \mathcal{D}_p$, of initial values $y_0 : \mathcal{D}_y$, of the time horizon $t : \mathcal{D}_t$, and the domains of algebraic variables $\mathcal{D}_q$;

- a set of constraints $\mathcal{C}$ which may be defined by set-based constraints over variables of $\mathcal{V}$ and special variables $\mathcal{Y}_i(\mathcal{D}_t, \mathcal{D}_y, \mathcal{D}_p)$ representing the set of the solution of $S_i$ in $S$.

with set-based constraints considered:

$$ g(A) \subseteq B $$

$$ g(A) \supseteq B $$

$$ g(A) \cap B = \emptyset $$

$$ g(A) \cap B \neq \emptyset $$
Particular problems considered and temporal properties

We focus on particular problems of robotics involving quantifiers

- Robust controller synthesis: $\exists u, \forall p, \forall y_0 + \text{temporal constraints}$
- Parameter synthesis: $\exists p, \forall u, \forall y_0 + \text{temporal constraints}$
- etc.

We also defined a set of temporal constraints useful to analyze/design robotic application.

<table>
<thead>
<tr>
<th>Verbal property</th>
<th>QCSDP translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stay in $\mathcal{A}$</td>
<td>$\forall t \in [0, t_{\text{end}}], [y](t, v') \subseteq \text{Int}(\mathcal{A})$</td>
</tr>
<tr>
<td>In $\mathcal{A}$ at $\tau$</td>
<td>$\exists t \in [0, t_{\text{end}}], [y](t, v') \subseteq \text{Int}(\mathcal{A})$</td>
</tr>
<tr>
<td>Has crossed $\mathcal{A}^*$</td>
<td>$\exists t \in [0, t_{\text{end}}], [y](t, v') \cap \text{Hull}(\mathcal{A}) \neq \emptyset$</td>
</tr>
<tr>
<td>Go out $\mathcal{A}$</td>
<td>$\exists t \in [0, t_{\text{end}}], [y](t, v') \cap \text{Hull}(\mathcal{A}) = \emptyset$</td>
</tr>
<tr>
<td>Has reached $\mathcal{A}^*$</td>
<td>$[y](t_{\text{end}}, v') \cap \text{Hull}(\mathcal{A}) \neq \emptyset$</td>
</tr>
<tr>
<td>Finished in $\mathcal{A}$</td>
<td>$[y](t_{\text{end}}, v') \subseteq \text{Int}(\mathcal{A})$</td>
</tr>
</tbody>
</table>

*: shall be used in negative form
One application: validated path planning
Second part

Context

Interval Analysis
  Arithmetic and sets
  Constraint Satisfaction Problem

Validated Simulation
  Simulation of IVP
  Validated simulation

Validated Runge-Kutta Methods
  Validated schemes
  Local Truncation Error
  Computation of validated RK

Examples

Differential constraint satisfaction problems

Constraint Programming and Runge-Kutta
  CSP to define RK
  Experimentations
  Cost function to define optimal schemes
  Experimentations
  Properties

Conclusion
Is it possible to define new RK schemes with IA tools?

+ Higher order implies smaller LTE
+ Method adapted to a given problem
- Coefficients must be computed with guarantee too!
New scheme: a complex problem

Needs to solve constraints

High order polynomials (till $p$), number of constraints increases rapidly (4 for $p = 3$, 8 for $p = 4, 17, 37, 85, 200$)

1. $\sum_{1}^{s} b_i = 1$
2. $\sum_{1}^{s} b_i c_i = 1/2$
3. $\sum_{1}^{s} b_i c_i^2 = 1/3$  $\sum_{1}^{s} \sum_{1}^{s} b_i a_{ij} c_j = 1/6$

Classical approach

Solved by using polynomials with known exact zeros such as Legendre (for Gauss) or Jacobi (for Radau)

Problems

▶ Discovery of new methods guided by solver and not by requirements
▶ Solved numerically: additive approximations
  ▶ Constraints not satisfied $\Rightarrow$ Method not at order $p$, but lower…
  ▶ Validated methods use LTE: wrong with floating numbers
New scheme: a complex problem

Needs to solve constraints

High order polynomials (till \( p \)), number of constraints increases rapidly (4 for \( p = 3 \), 8 for \( p = 4 \), 17, 37, 85, 200)

1. \[ \sum_{i=1}^{s} b_i = 1 \]
2. \[ \sum_{i=1}^{s} b_i c_i = \frac{1}{2} \]
3. \[ \sum_{i=1}^{s} b_i c_i^2 = \frac{1}{3} \]
   \[ \sum_{i=1}^{s} \sum_{j=1}^{s} b_i a_{ij} c_j = \frac{1}{6} \]

Classical approach

Solved by using polynomials with known exact zeros such as Legendre (for Gauss) or Jacobi (for Radau)

Problems

- Discovery of new methods guided by solver and not by requirements
- Solved numerically: additive approximations
  - Constraints not satisfied \( \Rightarrow \) Method not at order \( p \), but lower…
  - Validated methods use LTE: wrong with floating numbers
New scheme: a complex problem

Needs to solve constraints

High order polynomials (till $p$), number of constraints increases rapidly (4 for $p = 3$, 8 for $p = 4, 17, 37, 85, 200$)

1. $\sum_{1}^{s} b_i = 1$
2. $\sum_{1}^{s} b_i c_i = 1/2$
3. $\sum_{1}^{s} b_i c_i^2 = 1/3 \quad \sum_{1}^{s} \sum_{1}^{s} b_i a_{ij} c_j = 1/6$

Classical approach

Solved by using polynomials with known exact zeros such as Legendre (for Gauss) or Jacobi (for Radau)

Problems

- Discovery of new methods guided by solver and not by requirements
- Solved numerically: additive approximations
  - Constraints not satisfied $\Rightarrow$ Method not at order $p$, but lower...
  - Validated methods use LTE: wrong with floating numbers
Constraint approach to define new schemes

**Variables:** Butcher tableau coefficients

| $c_1$ | $a_{11}$ | $a_{12}$ | $\cdots$ | $a_{1s}$ |
| : | : | : | : | : |
| $c_s$ | $a_{s1}$ | $a_{s2}$ | $\cdots$ | $a_{ss}$ |
| : | $b_1$ | $b_2$ | $\cdots$ | $b_s$ |

**Domains:** $c_i \in [0, 1]$, $b_j \in [-1, 1]$, $a_{ij} \in [-1, 1]$ (a subpart)

**Constraints:**

**Consistency**

- $c_i = \sum a_{ij}$ with $c_1 < \cdots < c_s$

**Order conditions**

Function of order of desired method, example

$$\sum c_i b_i a_{ij} - 1/6 = 0$$

**Properties by construction**

- Singly diagonal: $a_{1,1} = \cdots = a_{s,s}$
- Explicit: $a_{ij} = 0, \forall j \geq i$
- Diagonal implicit: $a_{ij} = 0, \forall j > i$
- Explicit first line: $a_{1,1} = \cdots = a_{1,s} = 0$
- Stiffly accurate: $a_{s,i} = b_i, \forall i = 1, \ldots, s$
Re-discover the theory

Only one 2-stage method of order 4

Variables
b[2] in [-1,1];
c[2] in [0,1];
a[2][2] in [-1,1];

Constraints
b(1) + b(2) - 1.0 = 0;
b(1)*c(1) + b(2)*c(2) - 1.0/2.0 = 0;
b(1)*(c(1))^2 + b(2)*(c(2))^2 - 1.0/3.0 = 0;
b(1)*a(1)(1)*c(1) + b(1)*a(1)(2)*c(2) +
   b(2)*a(2)(1)*c(1) + b(2)*a(2)(2)*c(2)
   - 1.0/6.0 = 0;
b(1)*(c(1))^3 + b(2)*(c(2))^3 - 1.0/4.0 = 0;
b(1)*a(1)(1)*(c(1))^2 + b(1)*a(1)(2)*(c(2))^2 +
   b(2)*a(2)(1)*(c(1))^2 + b(2)*a(2)(2)*(c(2))^2
   - 1.0/12.0 = 0;
b(1)*a(1)(1)*a(1)(1)*c(1) + b(1)*a(1)(1)*a(1)(2)*c(2) +
   b(1)*a(1)(2)*a(2)(1)*c(1) + b(1)*a(1)(2)*a(2)(2)*c(2) +
   b(2)*a(2)(1)*a(1)(1)*c(1) + b(2)*a(2)(1)*a(1)(2)*c(2) +
   b(2)*a(2)(2)*a(2)(1)*c(1) + b(2)*a(2)(2)*a(2)(2)*c(2)
   - 1.0/24.0 = 0;
a(1)(1)+a(1)(2)-c(1) = 0; a(2)(1)+a(2)(2)-c(2) = 0;
c(1) < c(2);

Solved with Ibex

number of solutions=1
cpu time used=0.013073s.
([0.5, 0.5]; [0.5, 0.5];
0.21132486540[5,6]; 0.7886713459[5,6];
[0.25, 0.25]; -0.038675134594[9,8]
0.53867513459[5,6]; [0.25, 0.25])
Re-discover the theory

Only one 2-stage method of order 4

Variables
b[2] in [-1,1];
c[2] in [0,1];
a[2][2] in [-1,1];

Constraints
b(1) +b(2) -1.0=0;
b(1)*c(1) +b(2)*c(2) -1.0/2.0=0;
b(1)*(c(1))^2 +b(2)*(c(2))^2 -1.0/3.0=0;
b(1)*a(1)(1)*c(1) +b(1)*a(1)(2)*c(2) +
  b(2)*a(2)(1)*c(1) +b(2)*a(2)(2)*c(2) -1.0/6.0=0;
b(1)*(c(1))^3 +b(2)*(c(2))^3 -1.0/4.0=0;
b(1)*a(1)(1)*(c(1))^2 +b(1)*a(1)(2)*(c(2))^2 +
  b(2)*a(2)(1)*(c(1))^2 +b(2)*a(2)(2)*(c(2))^2 -1.0/12.0=0;
b(1)*a(1)(1)*(c(1))^2 +b(1)*a(1)(2)*(c(2))^2 +
  b(2)*a(2)(1)*(c(1))^2 +b(2)*a(2)(2)*(c(2))^2 -1.0/24.0=0;
a(1)(1)+a(1)(2)-c(1) = 0; a(2)(1)+a(2)(2)-c(2) = 0;
c(1) < c(2);

Solved with Ibex

number of solutions=1
cpu time used=0.013073s.
([0.5, 0.5] ; [0.5, 0.5] ;
0.21132486540[5,6] ; 0.78867513459[5,6];
[0.25, 0.25] ; -0.038675134594[9,8]
0.53867513459[5,6] ; [0.25, 0.25])

⇒ Validated Gauss-Legendre!
Re-discover the theory and ...

No 2-stage method of order 5
Proof in 0.04s!

...find new methods
Remark: it is hard to be sure that a method is new...
A method order 4, 3 stages, singly, stiffly accurate

This method is promising: capabilities wanted for a stiff problem, singly to optimize the Newton solving and stiffly accurate to be more efficient w.r.t. stiff problems (and DAEs).

<table>
<thead>
<tr>
<th></th>
<th>0.1610979566[59, 62]</th>
<th>0.105662432[67, 71]</th>
<th>0.172855006[54, 67]</th>
<th>-0.117419482[69, 58]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[44, 50]</td>
<td>0.655889341[44, 50]</td>
<td>0.48209622[04, 10]</td>
<td>0.105662432[67, 71]</td>
<td>0.068127286[68, 74]</td>
</tr>
<tr>
<td>[1, 1]</td>
<td>0.3885453883[37, 75]</td>
<td>0.3885453883[37, 75]</td>
<td>0.5057921789[56, 65]</td>
<td>0.105662432[67, 71]</td>
</tr>
</tbody>
</table>

**Table:** New method S3O4
Integration with the new schemes

Implemented in DynIbex (a tool for validated simulation)
Norm of diameter of final solution bounds the global error

<table>
<thead>
<tr>
<th>Methods</th>
<th>time (s)</th>
<th>nb of steps</th>
<th>norm of diameter of final solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3O4</td>
<td>39</td>
<td>1821</td>
<td>$5.9 \times 10^{-5}$</td>
</tr>
<tr>
<td>Radau3</td>
<td>52</td>
<td>7509</td>
<td>$2 \times 10^{-4}$</td>
</tr>
<tr>
<td>Radau5</td>
<td>81</td>
<td>954</td>
<td>$7.6 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

Table: S3O4 on a stiff problem (oil problem)

⇒ As efficient than Radau at order 5, but faster than order 3!
Cost function to define optimal schemes

Problem: continuum of solutions
CSP can be under constrained (e.g., \( p \leq s \))

Example of countless methods
Countless number of 2-stage; order 2; stiffly accurate; fully implicit

Optimization
▶ We could find the best one!
▶ How choose the cost function?
Cost function to define optimal schemes

Problem: continuum of solutions
CSP can be under constrained (e.g., $p \leq s$)

Example of countless methods
Countless number of 2-stage; order 2; stiffly accurate; fully implicit

Optimization
▶ We could find the best one!
▶ How choose the cost function?
Cost function to define optimal schemes

Problem: continuum of solutions
CSP can be under constrained (e.g., $p \leq s$)

Example of countless methods
Countless number of 2-stage; order 2; stiffly accurate; fully implicit

Optimization
- We could find the best one!
- How choose the cost function?
Cost function

Minimizing local truncation error

- Method with lower error for the same order
- Example of general form of ERK with 2 stages and order 2

\[
\begin{array}{ccc}
0 & 0 & 0 \\
\alpha & \alpha & 0 \\
1-1/(2\alpha) & 1/(2\alpha) & \\
\end{array}
\]

Ralston[1]: \(\alpha = 2/3\) minimizes the sum of square of coefficients of rooted trees in the lte computation

Our approach: maximizing the order

- Minimizing the sum of squares of order constraints
- Cost easy to compute: direct from constraints
- Same result \(\alpha \in [0.666...6, 0.666...7]\)

Cost function

Minimizing local truncation error

- Method with lower error for the same order
- Example of general form of ERK with 2 stages and order 2

\[
\begin{array}{c|cc}
0 & 0 & 0 \\
\alpha & \alpha & 0 \\
\hline
1-1/(2 \alpha) & 1/(2 \alpha) & \\
\end{array}
\]

Ralston[1]: $\alpha = 2/3$ minimizes the sum of square of coefficients of rooted trees in the lte computation

Our approach: maximizing the order

- Minimizing the sum of squares of order constraints
- Cost easy to compute: direct from constraints
- Same result $\alpha \in [0.666...6, 0.666...7]$!

Re-discover the theory

Theory

Countless 2-stage order 2 stiffly accurate fully implicit. But there is only one method at order 3: RadauIIA.

Optimization of (2,2)

- best feasible point $(0.749999939992 ; 0.250000060009 ; 0.333333280449 ; 0.999999998633 ; 0.416655823215 ; -0.0833225527662 ; 0.749999932909 ; 0.250000055725)$
- cpu time used 0.3879s.

with a cost of $[-\infty, 2.89787805696 \cdot 10^{-11}]$: there is an order 3!

Verification with solver

We add constraints $b_1 = 0.75$ and $c_2 = 1$, then we find RadauIIA
Explicit 3 stages 3 order

Theory (again)

There is countless explicit (3,3)-methods, but there is no order 4 method with 3 stages.

With optimizer: Erk33

<table>
<thead>
<tr>
<th></th>
<th>[0, 0]</th>
<th>[0, 0]</th>
<th>[0, 0]</th>
<th>[0, 0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.4659048[706, 929]</td>
<td>0.4659048[706, 929]</td>
<td>0.9552627[48, 86]</td>
<td>[0, 0]</td>
</tr>
<tr>
<td></td>
<td>0.8006855[74, 83]</td>
<td>−0.154577[20, 17]</td>
<td>0.42961[399, 400]</td>
<td>0.3744800[0, 1]</td>
</tr>
<tr>
<td></td>
<td>0.19590[599, 600]</td>
<td>0.42961[399, 400]</td>
<td>0.3744800[0, 1]</td>
<td></td>
</tr>
</tbody>
</table>

Comparison to Kutta (known to be efficient)

Norm of order constraints at order 4:

- ERK33: 0.04522[277, 304]
- Kutta: 0.058925

⇒ Our method is then closer to fourth order than Kutta.
Integration with Erk33, on VanDerPol

<table>
<thead>
<tr>
<th>Methods</th>
<th>time</th>
<th>nb of steps</th>
<th>norm of diameter of final solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERK33</td>
<td>3.7</td>
<td>647</td>
<td>$2.2 \cdot 10^{-5}$</td>
</tr>
<tr>
<td>Kutta (3,3)</td>
<td>3.55</td>
<td>663</td>
<td>$3.4 \cdot 10^{-5}$</td>
</tr>
<tr>
<td>RK4 (4,4)</td>
<td>4.3</td>
<td>280</td>
<td>$1.9 \cdot 10^{-5}$</td>
</tr>
</tbody>
</table>

⇒ Equivalent to Kutta in term of time, but performance closer to RK4
Linear Stability

Example of explicit methods (s=p) [Hairer]

\[ R(z) = 1 + z \sum_j b_j + z^2 \sum_{j,k} b_j a_{jk} + z^3 \sum_{j,k,l} b_j a_{jk} a_{kl} + \ldots \]

Stability domain given by \( S = \{ z \in \mathbb{C} : |R(z)| \leq 1 \} \)

For RK4: \( R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} \)

After \( z = x + iy \), and some processing:

\[ |R(x, y)| = \sqrt{((((((0.166667 \times x^3) \times y) + ((0.5 \times x^2) \times y)) - ((0.166667 \times x) \times y^3)) + ((1 \times x) \times y)) - (0.166667 \times y^3)) + y)^2 + (((((((0.041667 \times x^4) + (0.166667 \times x^3)) - ((0.25 \times x^2) \times y^2)) + (0.5 \times x^2)) - ((0.5 \times x) \times y^2)) + x) + (0.041667 \times y^4)) - (0.5 \times y^2)) + 1)^2)) \leq 1 \]
Linear Stability

Paving of stability domain for RK4 method with high precision coefficients (blue) and with error ($10^{-8}$ and $10^{-2}$) on coefficients (red).
Algebraically stable

Algebraically stable if:
- \( b_i \geq 0 \), for all \( i = 1, \ldots, s \)
- \( M = (m_{ij}) = (b_ia_{ij} + b_ja_{ji} - b_ib_j)i,j=1^s \) is non-negative definite

Problem to solve

Solving the eigenvalue problem \( \det(A - \lambda I) = 0 \) \( (1) \) and proving \( \lambda > 0 \).

For 3-stage Runge-Kutta methods:
\[
(m_{11} - \lambda) * ((m_{22} - \lambda) * (m_{33} - \lambda) - m_{23} * m_{32}) - m_{12} * (m_{21} * (m_{33} - \lambda) - m_{23} * m_{31}) + m_{13} * (m_{21} * m_{32} - (m_{22} - \lambda) * m_{31}) = 0
\]

With contractor programming (Fwd/Bwd + Newton)

Eq.(1) has no solution in \( ] - \infty, 0[ \equiv M \) is non-negative definite.
Algebraically stable

Verification of theory

- Lobatto IIIC: contraction to empty set $\Rightarrow$ algebraically stable
- Lobatto IIIA: solution found ($-0.0481125$) $\Rightarrow$ not algebraically stable

With floating number

Lobatto IIIC with error of $10^{-9}$ on $a_{ij}$: solution found ($-1.03041 \cdot 10^{-05}$) $\Rightarrow$ not algebraically stable
Algebraically stable

Verification of theory

- Lobatto IIIC: contraction to empty set $\Rightarrow$ algebraically stable
- Lobatto IIIA: solution found $(-0.0481125) \Rightarrow$ not algebraically stable

With floating number

Lobatto IIIC with error of $10^{-9}$ on $a_{ij}$: solution found $(-1.03041 \cdot 10^{-05}) \Rightarrow$ not algebraically stable
Symplectic

Symplectic if $M = 0$, with $M = (m_{ij}) = (b_i a_{ij} + b_j a_{ji} - b_i b_j)^s_{i,j=1}$

Problem to solve

$0 \in [M]$ with interval arithmetic

Verification of theory with Gauss-Legendre:

$$M = 10^{-17} \cdot \begin{pmatrix} [-1.38, 1.38] & [-2.77, 2.77] & [-2.77, 1.38] \\ [-2.77, 2.77] & [-2.77, 2.77] & [-1.38, 4.16] \\ [-2.77, 1.38] & [-1.38, 4.16] & [-1.38, 1.38] \end{pmatrix}$$

With $a_{1,2} = 2.0/9.0 - \sqrt{15.0}/15.0$ computed with float

$$M = \begin{pmatrix} [-1.38e^{-17}, 1.38e^{-17}] & [-1.91e^{-09}, -1.91e^{-09}] & [-2.77e^{-17}, 1.38e^{-17}] \\ [-1.91e^{-09}, -1.91e^{-09}] & [-2.77e^{-17}, 2.77e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] \\ [-2.77e^{-17}, 1.38e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] & [-1.38e^{-17}, 1.38e^{-17}] \end{pmatrix}$$
Symplectic

Symplectic if \( M = 0 \), with \( M = (m_{ij}) = (b_ia_{ij} + b_ja_{ji} - b_ib_j)^s_{i,j=1} \)

Problem to solve

\( 0 \in [M] \) with interval arithmetic

Verification of theory with Gauss-Legendre:

\[
M = 10^{-17} \cdot \begin{pmatrix}
[-1.38, 1.38] & [-2.77, 2.77] & [-2.77, 1.38] \\
[-2.77, 2.77] & [-2.77, 2.77] & [-1.38, 4.16] \\
[-2.77, 1.38] & [-1.38, 4.16] & [-1.38, 1.38]
\end{pmatrix}
\]

With \( a_{1,2} = 2.0/9.0 - \sqrt{15.0}/15.0 \) computed with float

\[
M = \begin{pmatrix}
[-1.38e^{-17}, 1.38e^{-17}] & [-1.91e^{-09}, -1.91e^{-09}] & [-2.77e^{-17}, 1.38e^{-17}] \\
[-1.91e^{-09}, -1.91e^{-09}] & [-2.77e^{-17}, 2.77e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] \\
[-2.77e^{-17}, 1.38e^{-17}] & [-1.38e^{-17}, 4.16e^{-17}] & [-1.38e^{-17}, 1.38e^{-17}]
\end{pmatrix}
\]
Conclusion

Validated simulation with RK
► Method to bound the LTE
► Contractor based approach to solve Implicit RK
► Good results, even for stiff problems
► Library DynIbex (DAEs, constrained ODEs)

Constraint programming for RK
► Tool to re-discover the theory on RK methods…
► …and able to define new (optimal) schemes !

Future works
► DynIbex in continuous development
► New RK schemes with higher order
► Solve some open problems
Questions ?
Appendices