
# Génération et validation des méthodes de Runge-Kutta

## Overview of four years of study

Alexandre Chapoutot, Julien Alexandre dit Sandretto

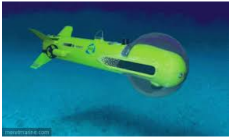
Department U2IS

ENSTA ParisTech

RAIM 2018 - Gif-sur-Yvette

# 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

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

## Arithmetic

Extension of operators $(+, -, *, /, sin, cos, ...)$, 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: $\int_{[x]} f(x')dx' \in [f]([x]).w([x])$

# 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 $(+, -, *, /, sin, cos, ...)$, 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: $\int_{[x]} f(x')dx' \in [f]([x]).w([x])$

# Interval Analysis: the suitable tool

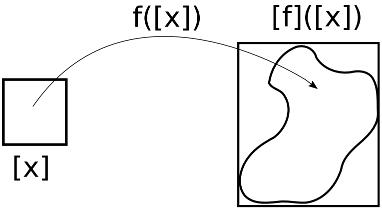
$[x] = [\underline{x}, \overline{x}]$ stands for the set of reals $x$ s.t. $\underline{x} \leq x \leq \overline{x}$

## Arithmetic

Extension of operators $(+, -, *, /, sin, cos, ...)$, 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: $\int_{[x]} f(x')dx' \in [f]([x]).w([x])$

# Interval Analysis: the suitable tool

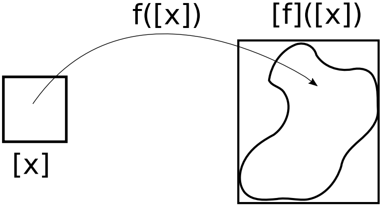
$[x] = [\underline{x}, \overline{x}]$ stands for the set of reals $x$ s.t. $\underline{x} \leq x \leq \overline{x}$

## Arithmetic

Extension of operators $(+, -, *, /, sin, cos, ...)$, e.g. $[-1, 1] + [1, 3] = [0, 4]$
Rounding error handled $(1/3 \in 0.33333333[3, 4])$

## Extension of function

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

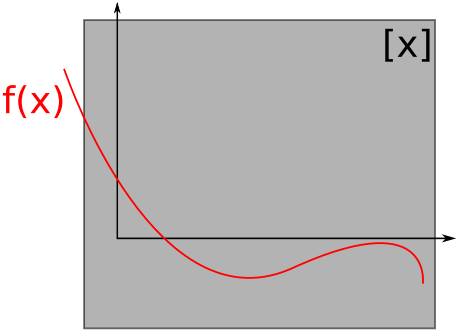



## Interval Integral

Rectangle rule: $\int_{[x]} f(x')dx' \in [f]([x]).w([x])$

# Constraint Satisfaction Problem (CSP)

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




## A classical problem

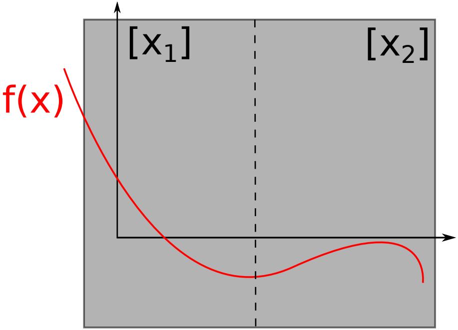
find $x$ s.t. $f(x) = 0$ assuming $x \in [x]$.

## More generally, a CSP is

- a set of variables $\mathcal{V}$
- a set of domains $\mathcal{D}$
- a set of constraints $\mathcal{C}$.

# Constraint Satisfaction Problem (CSP)

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




## A simple method

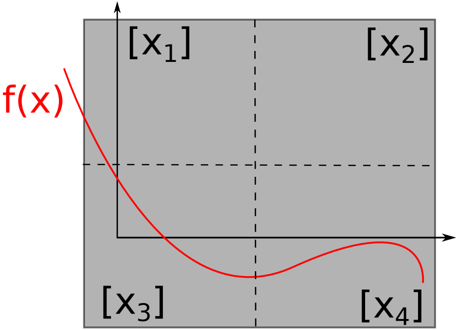
Interval arithmetic + bisection strategy

- 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

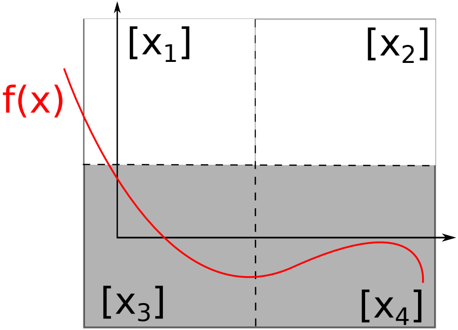
Interval arithmetic + bisection strategy

- if $0 \notin [f]([x])$ 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

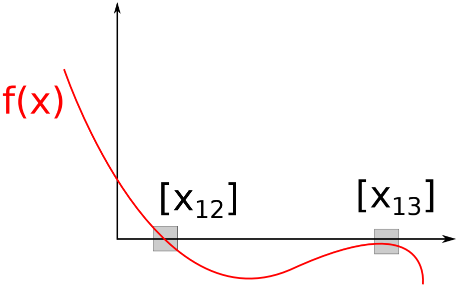
Interval arithmetic + bisection strategy

- if $0 \notin [f]([x])$ 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

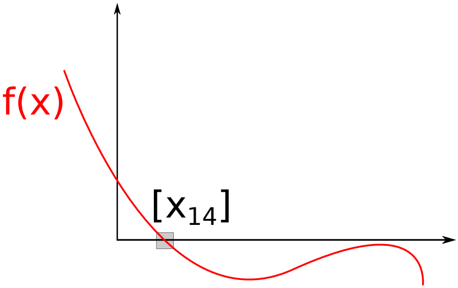
Interval arithmetic + bisection strategy

- if $0 \notin [f]([x])$ 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

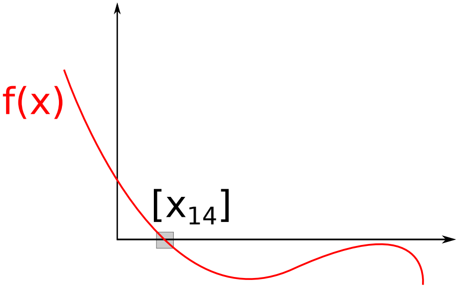
Interval arithmetic + bisection strategy

- if $0 \notin [f]([x])$ 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 $0 \notin [f]([x])$ 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]

[1] Jaulin et al., "Applied Interval Analysis", Springer, 2001

# Simulation of IVP

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

$$\dot{\mathbf{y}} = f(\mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0$$

IVP has a unique solution $\mathbf{y}(t; \mathbf{y}_0)$ if $f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz in $\mathbf{y}$ 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: $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n$ such that

$$\forall i \in \{0, \ldots, n\}, \quad \mathbf{y}_i \approx \mathbf{y}(t_i; \mathbf{y}_0) \ .$$

# Validated Simulation

## Goal of validated numerical integration

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

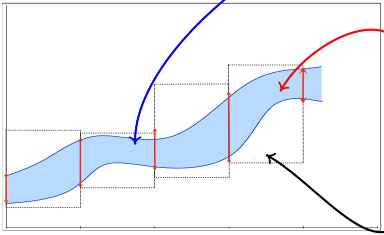
$$\forall i \in \{0, \dots, n\}, \quad \mathbf{y}(t; \mathbf{y}_0) \in [\tilde{\mathbf{y}}_i], \forall t \in [t_i, t_{i+1}] \ ,$$

- and a sequence of values: $[\mathbf{y}_0], [\mathbf{y}_1], \dots, [\mathbf{y}_n]$ such that

$$\forall i \in \{0, \dots, n\}, \quad \mathbf{y}(t_i; \mathbf{y}_0) \in [\mathbf{y}_i] \ .$$

# Validated Simulation

## A two-step approach



Exact solution of
$\dot{\mathbf{y}} = f(\mathbf{y}(t)), \mathbf{y}(0) \in [\mathbf{y}_0]$

Safe approximation at discrete time instants $[\mathbf{y}_i]$, obtained with Taylor (Capd, Vnode) or Runge-Kutta (DynIbex) validated methods

Safe approximation between time instants $[\tilde{\mathbf{y}}_i]$, obtained with a Picard-Lindelöf operator

# Validated Simulation

## Picard-Lindelöf operator

Formal solution of ODE: $\mathbf{y}_{n+1} = \mathbf{y}_n + \int_0^h f(s)ds$
Following the rectangle rule, based on Brouwer's theorem, the Picard-Lindelöf operator is defined such that:

$$P([\mathbf{x}]) = \mathbf{y_n} + [0, h][f]([\mathbf{x}])$$

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

**Remarks:** the rectangle rule can be replaced by any validated scheme (Taylor series [1] or RK)

[1] Nedialkov et al., "Validated solutions of initial value problems for ordinary differential equations", Appl. Math. and Comp., 1999

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

$$[\mathbf{y}_{\ell+1}] = [\Phi]\left(h, [\mathbf{y}_\ell]\right) + \text{Error of } \Phi \; .$$

# Validated Runge-Kutta Methods

**How validate a RK method ?**

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

$$LTE = \mathbf{y}(t_\ell; \mathbf{y}_{\ell-1}) - \mathbf{y}_\ell = C \cdot h^{p+1} \quad \text{with} \quad C \in \mathbb{R}.$$

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 $q$th derivative of the **exact solution** is given by

$$\mathbf{y}^{(q)} = \sum_{r(\tau)=q} \alpha(\tau) F(\tau)(\mathbf{y}_0) \quad \text{with} \quad \begin{array}{l} r(\tau) \text{ the order of the rooted tree } \tau \\ \alpha(\tau) \text{ a positive integer} \\ F(\tau)(.) \text{ elementary differential for } \tau \end{array}$$

We can do the same for the numerical solution:

## Theorem 2 (Butcher, 1963)

The $q$th derivative of the **numerical solution** is given by

$$\mathbf{y}_1^{(q)} = \sum_{r(\tau)=q} \gamma(\tau)\phi(\tau)\alpha(\tau) F(\tau)(\mathbf{y}_0) \quad \text{with} \quad \begin{array}{l} \gamma(\tau) \text{ a positive integer} \\ \phi(\tau) \text{ depending on a Butcher tableau} \end{array}$$

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

A Runge-Kutta method has order $p$ iff $\phi(\tau) = \frac{1}{\gamma(\tau)} \quad \forall \tau, r(\tau) \leqslant p$

# LTE formula for **explicit and implicit** Runge-Kutta

From Theorem 1 and Theorem 2, if a Runge-Kutta has order $p$ then

$$\mathbf{y}(t_1; \mathbf{y}_0) - \mathbf{y}_1 = \frac{h^{p+1}}{(p+1)!} \sum_{r(\tau)=p+1} \alpha(\tau) \left[1 - \gamma(\tau)\phi(\tau)\right] F(\tau)(\mathbf{y}(\xi)), \quad \xi \in [t_1, t_0]$$

## Remark

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

- for each method the Butcher tableau and the order available
- $\mathbf{y}(\xi)$ is enclosed by $[\tilde{\mathbf{y}}]$ 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

$\Rightarrow$ LTE can be bounded, but...

## It remains to compute the RK scheme itself:

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

$$[\mathbf{k}_1] = [f](t_n, [\mathbf{y}_n]) \ , \qquad [\mathbf{k}_2] = [f](t_n + 1h, [\mathbf{y}_n] + h1[\mathbf{k}_1])$$

$$[\mathbf{y}_{n+1}] = [\mathbf{y}_n] + h\left(\frac{1}{2}[\mathbf{k}_1] + \frac{1}{2}[\mathbf{k}_2]\right)$$

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

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

$$[\mathbf{k}_1] = [f]\left(t_n + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)h, \quad [\mathbf{y}_n] + h\left(\frac{1}{4}[\mathbf{k}_1] + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)[\mathbf{k}_2]\right)\right)$$

$$[\mathbf{k}_2] = [f]\left(t_n + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)h, \quad [\mathbf{y}_n] + h\left(\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)[\mathbf{k}_1] + \frac{1}{4}[\mathbf{k}_2]\right)\right)$$

$$[\mathbf{y}_{n+1}] = [\mathbf{y}_n] + h\left(\frac{1}{2}[\mathbf{k}_1] + \frac{1}{2}[\mathbf{k}_2]\right)$$

# Validated Runge-Kutta Methods

Solve a problem with interval analysis: contraction technique** !
$[\mathbf{k}_1] = [\mathbf{k}_2] = [\mathbf{k}_3] = [\mathbf{k}_4] = [\tilde{\mathbf{y}}]$
Then, we repeat:

$$[\mathbf{k}_1] = [\mathbf{k}_1] \cap [f]\left([\mathbf{y}_n] + h\left(\frac{1}{4}[\mathbf{k}_1] + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)[\mathbf{k}_2]\right)\right)$$

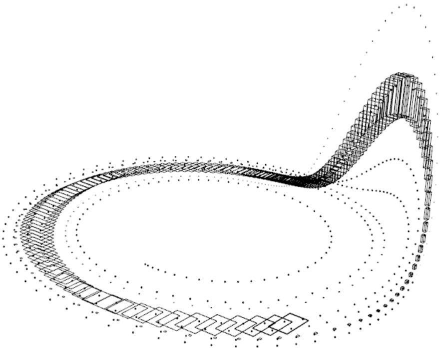
$$[\mathbf{k}_2] = [\mathbf{k}_2] \cap [f]\left([\mathbf{y}_n] + h\left(\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)[\mathbf{k}_1] + \frac{1}{4}[\mathbf{k}_2]\right)\right)$$

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

# Examples

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

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

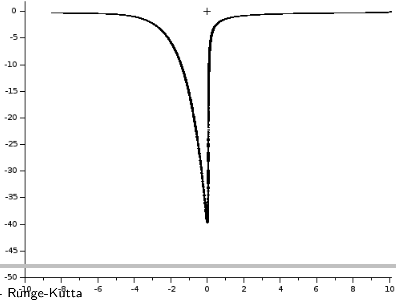


# Examples

## A chemical reaction simulated (stiff)

$$\begin{cases} \dot{y} = z \\ \dot{z} = z^2 - \dfrac{3}{0.0001 + y^2} \end{cases} \quad \text{with} \quad \begin{cases} y(0) = 10 \\ z(0) = 0 \end{cases} \quad \text{and} \quad t \in [0, 50]$$

**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

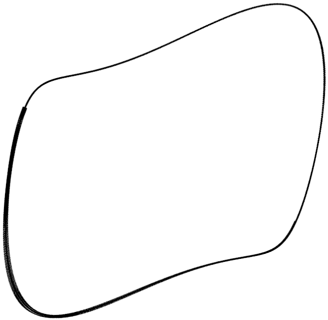


# Examples

## Van Der Pol 50s

Initial states: $y(0) = (2,0)$, One parameter: $\mu = 1.0$ *or* $2.0$

$$\dot{y} = \begin{cases} y_1 \\ \mu * (1 - y_0^2) * y_1 - y_0) \end{cases}$$

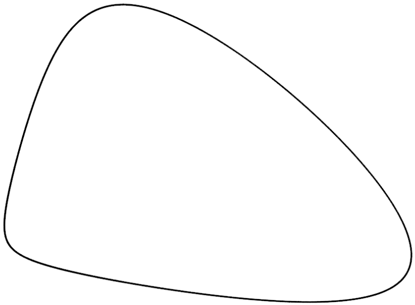


# Examples

## Volterra 6s

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

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

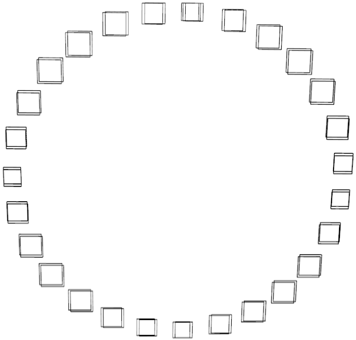


# Examples

## Circle 100s

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

The differential system: $\dot{y} = \begin{cases} -y_1 \\ y_0 \end{cases}$



# Dynamical systems

A general settings of dynamical systems

$$S \equiv \begin{cases} \dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \mathbf{x}(t), \mathbf{p}), \\ 0 = \mathbf{g}(t, \mathbf{y}(t), \mathbf{x}(t)) \\ 0 = \mathbf{h}(\mathbf{y}(t), \mathbf{x}(t)) \end{cases} .$$

we denote by

$$\mathcal{Y}(\mathcal{T}, \mathcal{Y}_0, \mathcal{P}) = \{\mathbf{y}(t; \mathbf{y}_0, \mathbf{p}) : t \in \mathcal{T}, \mathbf{y}_0 \in \mathcal{Y}_0, \mathbf{p} \in \mathcal{P}\} \ .$$

the set of solutions

# Example of ODEs with constraints

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

$$\begin{pmatrix} \dot{y}_0 \\ \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} \dfrac{-y_0 y_1}{1 + y_0} \\ \dfrac{y_0 y_1}{1 + y_0} - a y_1 \\ a y_1 \end{pmatrix}$$

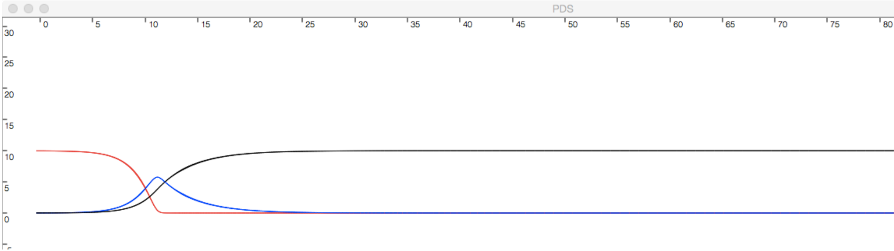
and associated to constraints:

$$\begin{aligned} y_0 + y_1 + y_2 &= 10.0 \\ y_0 &\geqslant 0 \\ y_1 &\geqslant 0 \\ y_2 &\geqslant 0 \end{aligned}$$

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.*, $(\mathbf{y}_0, \mathbf{p})$, a time variable $t$ and some other algebraic variables $\mathbf{q}$;
- a domain $\mathcal{D}$ made of the domain of parameters $\mathbf{p} : \mathcal{D}_p$, of initial values $\mathbf{y}_0 : \mathcal{D}_{y_0}$, 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 $\mathcal{Y}_i(\mathcal{D}_t, \mathcal{D}_{y_0}, \mathcal{D}_p)$ representing the set of the solution of $S_i$ in $\mathcal{S}$.

with set-based constraints considered:

$$\mathbf{g}(\mathcal{A}) \subseteq \mathcal{B} \qquad \mathbf{g}(\mathcal{A}) \supseteq \mathcal{B}$$

$$\mathbf{g}(\mathcal{A}) \cap \mathcal{B} = \emptyset \qquad \mathbf{g}(\mathcal{A}) \cap \mathcal{B} \neq \emptyset$$

# Particular problems considered and temporal properties

We focus on particular problems of robotics involving quantifiers

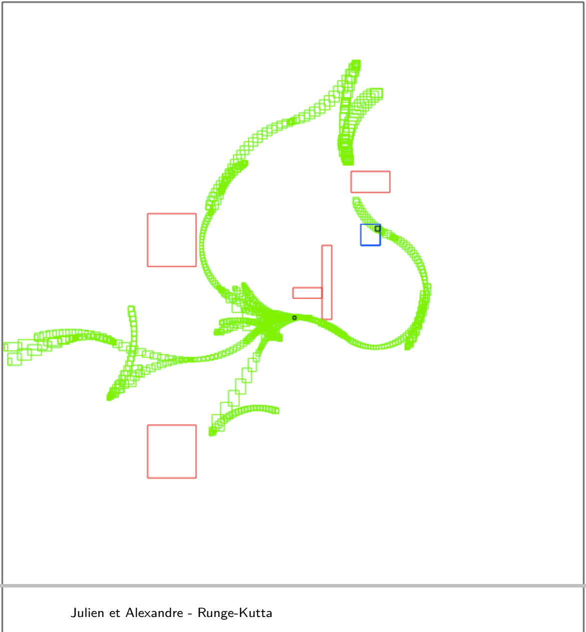
- Robust controller synthesis: $\exists \mathbf{u}, \forall \mathbf{p}, \forall \mathbf{y}_0$ + temporal constraints
- Parameter synthesis: $\exists \mathbf{p}, \forall \mathbf{u}, \forall \mathbf{y}_0$ + temporal constraints
- etc.

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

| Verbal property | QCSDP translation |
|---|---|
| Stay in $\mathcal{A}$ | $\forall t \in [0, t_{\text{end}}],\ [\mathbf{y}](t, \mathbf{v}') \subseteq \text{Int}(\mathcal{A})$ |
| In $\mathcal{A}$ at $\tau$ | $\exists t \in [0, t_{\text{end}}],\ [\mathbf{y}](t, \mathbf{v}') \subseteq \text{Int}(\mathcal{A})$ |
| Has crossed $\mathcal{A}$* | $\exists t \in [0, t_{\text{end}}],\ [\mathbf{y}](t, \mathbf{v}') \cap \text{Hull}(\mathcal{A}) \neq \emptyset$ |
| Go out $\mathcal{A}$ | $\exists t \in [0, t_{\text{end}}],\ [\mathbf{y}](t, \mathbf{v}') \cap \text{Hull}(\mathcal{A}) = \emptyset$ |
| Has reached $\mathcal{A}$* | $[\mathbf{y}](t_{\text{end}}, \mathbf{v}') \cap \text{Hull}(\mathcal{A}) \neq \emptyset$ |
| Finished in $\mathcal{A}$ | $[\mathbf{y}](t_{\text{end}}, \mathbf{v}') \subseteq \text{Int}(\mathcal{A})$ |

*: shall be used in negative form

# One application: validated path planning



# Second part

# Constraint Programming and Runge-Kutta

## 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.$\sum_1^s b_i c_i^2 = 1/3 \quad \sum_1^s \sum_1^s b_i a_{ij} c_j = 1/6$

## 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 ⇒ 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

# 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 ⇒ 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

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

**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)*c(1)*a(1)(1)*c(1) +b(1)*c(1)*a(1)(2)*c(2) +
    b(2)*c(2)*a(2)(1)*c(1) +b(2)*c(2)*a(2)(2)*c(2)
    -1.0/8.0=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);
end
```

## 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])
```

# 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)*c(1)*a(1)(1)*c(1) +b(1)*c(1)*a(1)(2)*c(2) +
    b(2)*c(2)*a(2)(1)*c(1) +b(2)*c(2)*a(2)(2)*c(2)
    -1.0/8.0=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);
end
```

### 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])
```

### $\Rightarrow$ 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).

| | | | |
|---|---|---|---|
| 0.1610979566[59, 62] | 0.105662432[67, 71] | 0.172855006[54, 67] | -0.117419482[69, 58] |
| 0.655889341[44, 50] | 0.482099622[04, 10] | 0.105662432[67, 71] | 0.068127286[68, 74] |
| [1, 1] | 0.3885453883[37, 75] | 0.5057921789[56, 65] | 0.105662432[67, 71] |
| | 0.3885453883[37, 75] | 0.5057921789[56, 65] | 0.105662432[67, 71] |

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

| Methods | time (s) | nb of steps | norm of diameter of final solution |
|---|---|---|---|
| S3O4 | 39 | 1821 | $5.9 \cdot 10^{-5}$ |
| Radau3 | 52 | 7509 | $2 \cdot 10^{-4}$ |
| Radau5 | 81 | 954 | $7.6 \cdot 10^{-5}$ |

Table: S3O4 on a stiff problem (oil problem)

$\Rightarrow$ 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

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

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 $\alpha \in [0.666...6, 0.666...7]$ !

[1] Ralston, Anthony. "Runge-Kutta methods with minimum error bounds." Mathematics of computation (1962).

# 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

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

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 $\alpha \in [0.666...6, 0.666...7]$ !

[1] Ralston, Anthony. "Runge-Kutta methods with minimum error bounds." Mathematics of computation (1962).

# 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

| | | | |
|---|---|---|---|
| [0, 0] | [0, 0] | [0, 0] | [0, 0] |
| 0.4659048[706, 929] | 0.4659048[706, 929] | [0, 0] | [0, 0] |
| 0.8006855[74, 83] | −0.154577[20, 17] | 0.9552627[48, 86] | [0, 0] |
| | 0.19590[599, 600] | 0.42961[399, 400] | 0.3744800[0, 1] |

## Comparison to Kutta (known to be efficient)

Norm of order constraints at order 4:

- ERK33: $0.045221[277, 304]$
- Kutta: 0.058925

$\Rightarrow$ Our method is then closer to fourth order than Kutta.

Kutta third order:

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1/2 | 1/2 | 0 | 0 |
| 1 | -1 | 2 | 0 |
| | 1/6 | 2/3 | 1/6 |

# Integration with Erk33, on VanDerPol

| Methods | time | nb of steps | norm of diameter of final solution |
|---|---|---|---|
| ERK33 | 3.7 | 647 | $2.2 \cdot 10^{-5}$ |
| Kutta (3,3) | 3.55 | 663 | $3.4 \cdot 10^{-5}$ |
| RK4 (4,4) | 4.3 | 280 | $1.9 \cdot 10^{-5}$ |

$\Rightarrow$ Equivalent to Kutta in term of time, but performance closer to RK4

# Linear Stability

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

$$R(z) = 1 + z\sum_{j} b_j + z^2 \sum_{j,k} b_j a_{jk} + z^3 \sum_{j,k,l} b_j a_{jk} a_{kl} + \dots$$

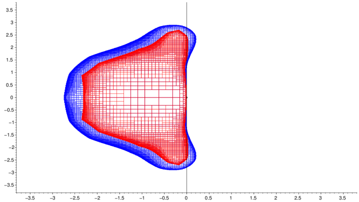
Stability domain given by $S = \{z \in \mathcal{C} : |R(z)| \leq 1\}$

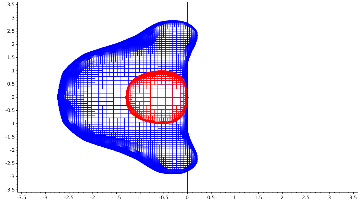
For RK4: $R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$

After $z = x + iy$, and some processing:

$$|R(x,y)| = \sqrt{(((((((((0.166667 * x^3) * y) + ((0.5 * x^2) * y)) - ((0.166667 * x) * y^3)) + ((1 * x) * y)) - (0.166667 * y^3)) + y)^2 + (((((((((0.0416667 * x^4) + (0.166667 * x^3)) - ((0.25 * x^2) * y^2)) + (0.5 * x^2)) - ((0.5 * x) * y^2)) + x) + (0.0416667 * y^4)) - (0.5 * y^2)) + 1)^2)} \leq 1$$

# 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 $i = 1, \ldots, s$
- $M = (m_{ij}) = (b_i a_{ij} + b_j a_{ji} - b_i b_j)_{i,j=1}^{s}$ is non-negative definite

## Problem to solve

Solving the eigenvalue problem $\det(A - \lambda I) = 0 \quad (1)$
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 ⇒ algebraically stable
- Lobatto IIIA: solution found ($-0.0481125$) ⇒ not algebraically stable

## With floating number

Lobatto IIIC with error of $10^{-9}$ on $a_{ij}$: solution found ($-1.03041 \cdot 10^{-05}$) ⇒ 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 $M = (m_{ij}) = (b_i a_{ij} + b_j a_{ji} - b_i b_j)_{i,j=1}^{s}$

## 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`

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

# Symplectic

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

## 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`

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

# 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