Sets and B-series

Formalism and abstraction

Julien Alexandre dit Sandretto

Department U2IS

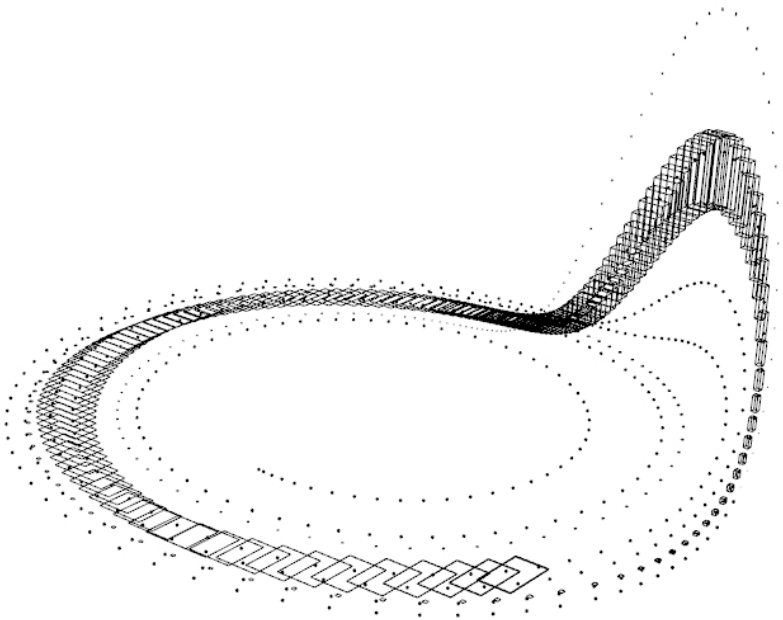
ENSTA Paris - IP Paris

GO20 2023 - Gozo

Our past contributions: part 1

Validated simulation for ODEs

- Interval arithmetic (guaranteed rounding)
- Validated Runge-Kutta (truncation error bounded with Picard-Lindelöf)
- Box based reachable tube



Our past contributions: part 2

Constraint programming and RK

- Order condition translated to constraint satisfaction problem and solve with interval based algorithms
- Prove some known results of RK theory
- 2-norm of $p+1$ order conditions as cost function $\Rightarrow$ new optimal order p methods (closest to order $p+1$)

Optimal explicit RK 3-3 with interval coefficient:

[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]
	$0.19590[599, 600]$	$0.42961[399, 400]$	$0.3744800[0, 1]$

A review

- Several applications in robotics (collision detection, optimal control, motion planning, etc)
- Thousands downloads of DynIbex
- Collaborations, invitations and discussions

Few open issues

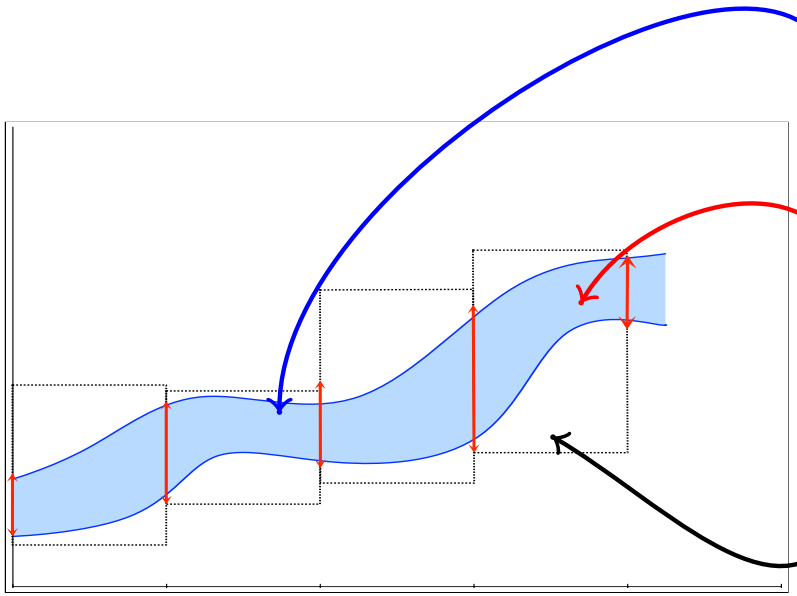
- Boxes is a - sometimes - too rough abstraction (e.g. for collision detection $\Rightarrow$ false positive)
- Expensive computations (mainly due to truncation error complexity)

Set abstraction

How represent sets of reals in a computer ?

With boxes - a brief recall

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 Runge-Kutta (DynIbex) validated methods

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

Main tool: Interval analysis

$[x] = [\underline{x}, \overline{x}]$ 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: $\int_{[x]} f(x')dx' \in [f]([x]).w([x])$

Main tool: Interval analysis

$[x] = [\underline{x}, \overline{x}]$ 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: $\int_{[x]} f(x')dx' \in [f]([x]).w([x])$

Main tool: Interval analysis

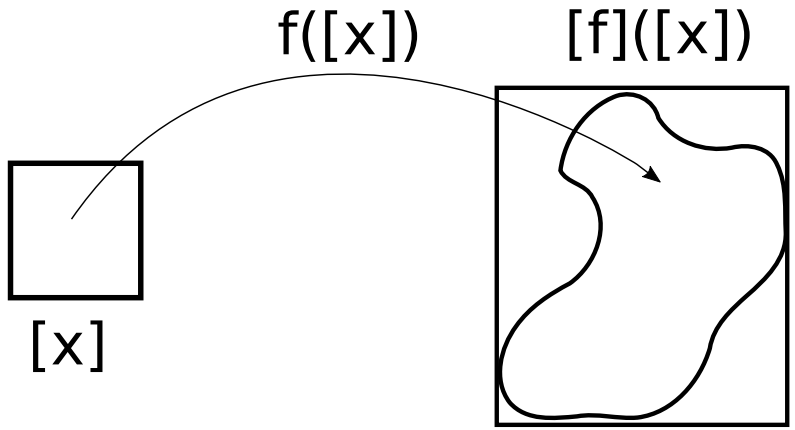
$[x] = [\underline{x}, \overline{x}]$ stands for the set of reals x s.t. $\underline{x} \leq x \leq \overline{x}$

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

Main tool: Interval analysis

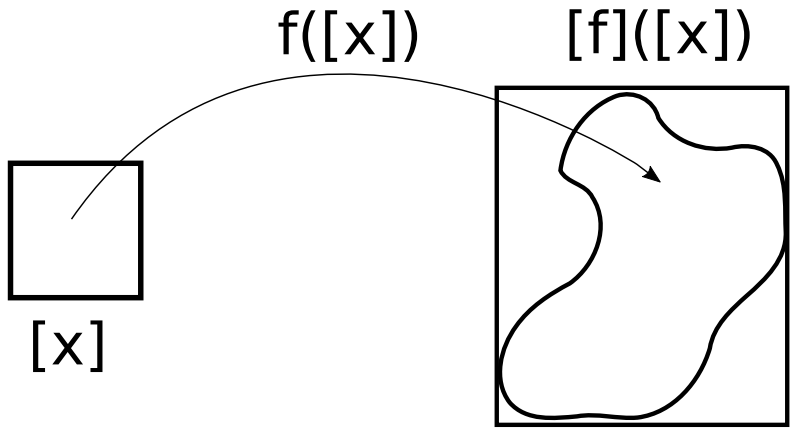
$[x] = [\underline{x}, \overline{x}]$ stands for the set of reals x s.t. $\underline{x} \leq x \leq \overline{x}$

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

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 $[\tilde{\mathbf{y}_0}], [\tilde{\mathbf{y}_1}], \ldots, [\tilde{\mathbf{y}_{n-1}}]$ such that

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

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

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

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

Validated Runge-Kutta methods

A validated algorithm

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

$[\Phi]$

- Interval evaluation of explicit RKs
- Contraction based approach for implicit RKs

Error of $[\Phi]$

From Theorem 1 and Theorem 2 (Butcher 1963), if a Runge-Kutta has order p then

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

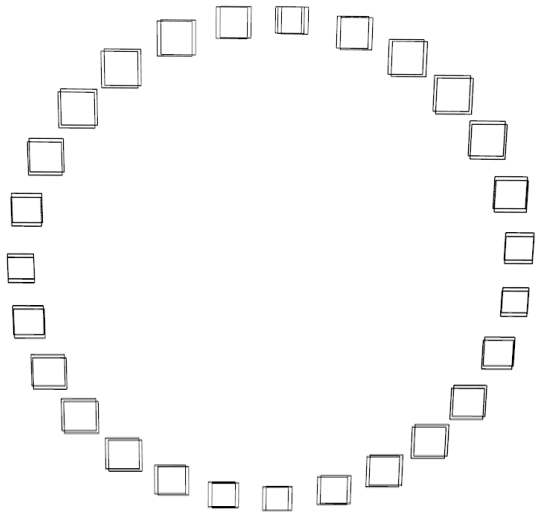
By Picard, $\mathbf{y}(\xi) \in [\tilde{\mathbf{y}}]$, thus LTE can be bounded.

Validated RK: Results

Circle 100s with RK4

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

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



Wrapping effect must be counteracted $\Rightarrow$ affine arithmetic

With zonotopes

Affine arithmetic extends numerical operations

- $\hat{x} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \varepsilon_i$
- $\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i \varepsilon_i$

then

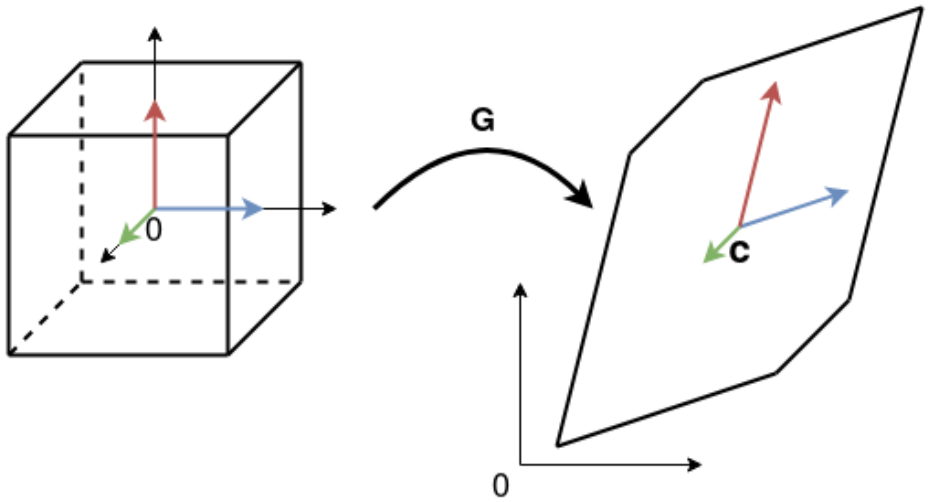
- $a\hat{x} + b\hat{y} + c = (a\alpha_0 + b\beta_0 + c) + \sum_{i=1}^{n}(a\alpha_i + b\beta_i)\varepsilon_i$

Most operations create new noise symbols, e.g. multiplication:
$\hat{x} \times \hat{y} = \alpha_0\alpha_1 + \sum_{i=1}^{n}(\alpha_i\beta_0 + \alpha_0\beta_i)\varepsilon_i + \nu\varepsilon_{n+1}$, where
$\nu = \left(\sum_{i=1}^{n} |\alpha_i|\right) \times \left(\sum_{i=1}^{n} |\beta_i|\right)$ over-approximates non linearities.

Arithmetic operations done with intervals to preserve guarantees!

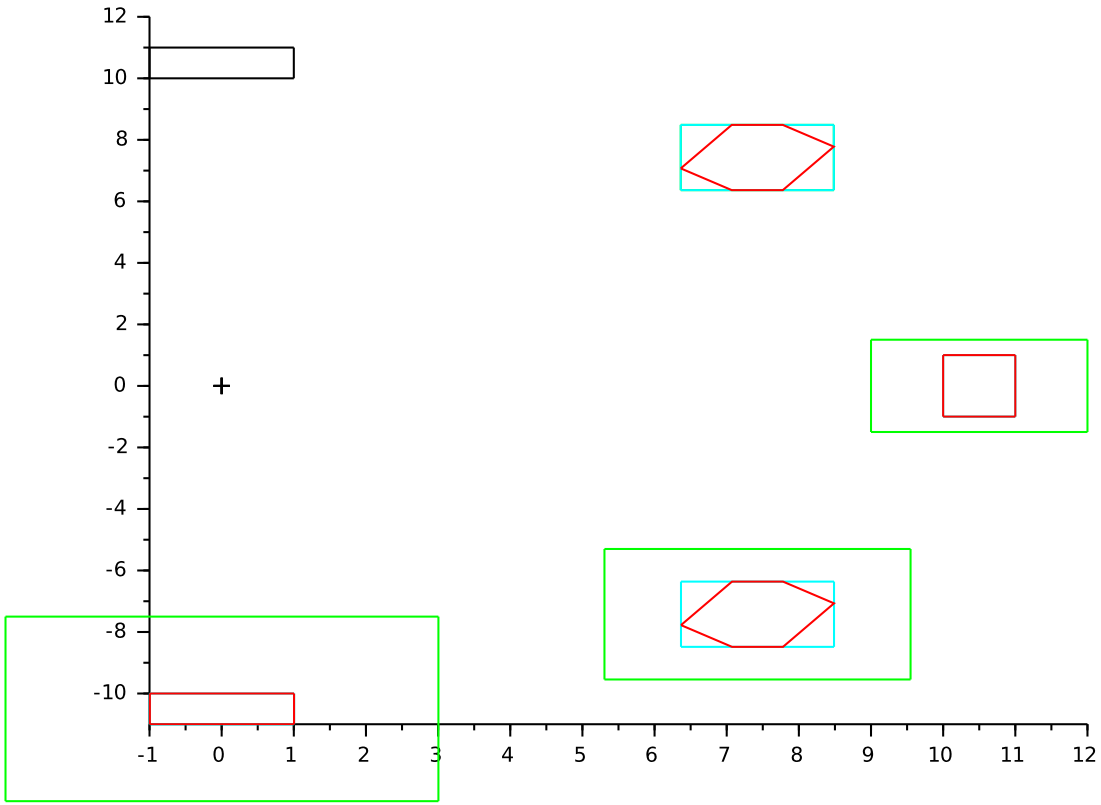
With zonotopes

Zonotopes are geometrical concretization of affine arithmetic



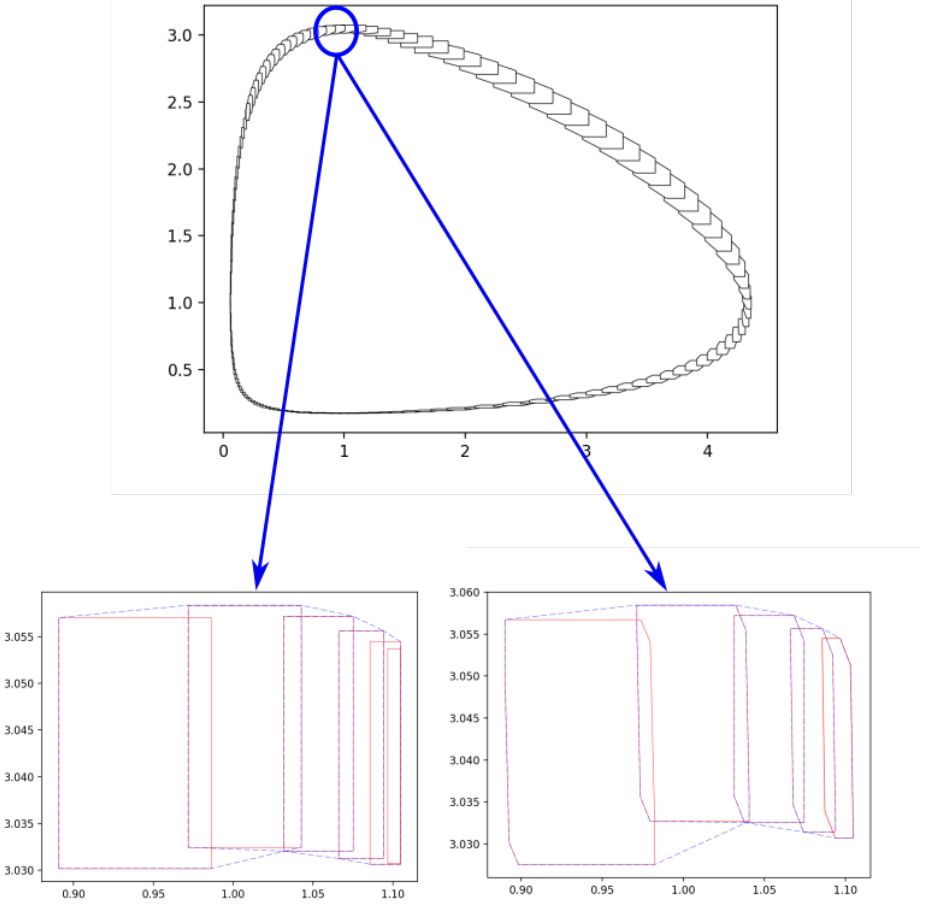
With zonotopes

Main property (that interests us): centrally symmetric polytope
$\Rightarrow$ robust to rotation.



With zonotopes

Volterra boxes vs zonotopes



With polytopes

- Zonotopes have a restrictive shape (need symmetry)
- Polytopes have better expressivity, come from linear constraints, etc
- But no associated arithmetic

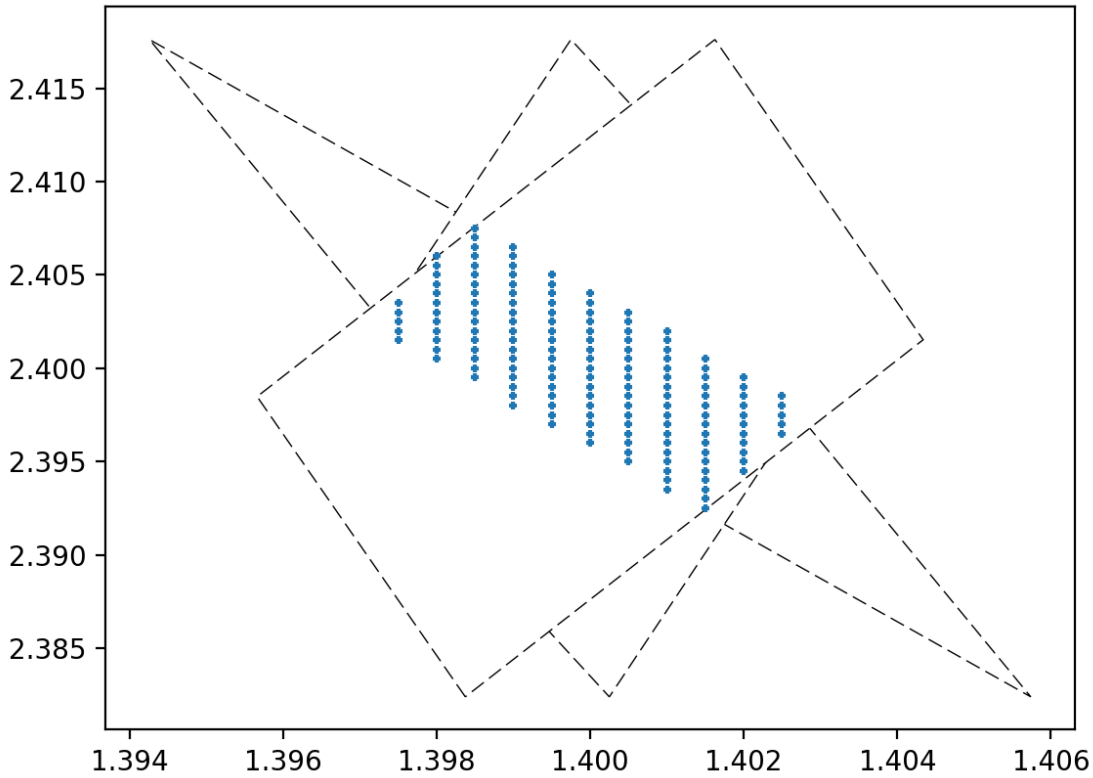
Geometrical trick

A polytope $\mathcal{P}$ can be represented exactly by the intersection of a finite number of zonotopes: $\mathcal{P} = Z_1 \cap \cdots \cap Z_n$
Image of a function f on polytope can be computed as $f(\mathcal{P}) = f(Z_1 \cap \cdots \cap Z_n) \subseteq f(Z_1) \cap \cdots \cap f(Z_n)$ where $f(Z_1), \cdots, f(Z_n)$ can be computed using affine arithmetic.

With polytopes

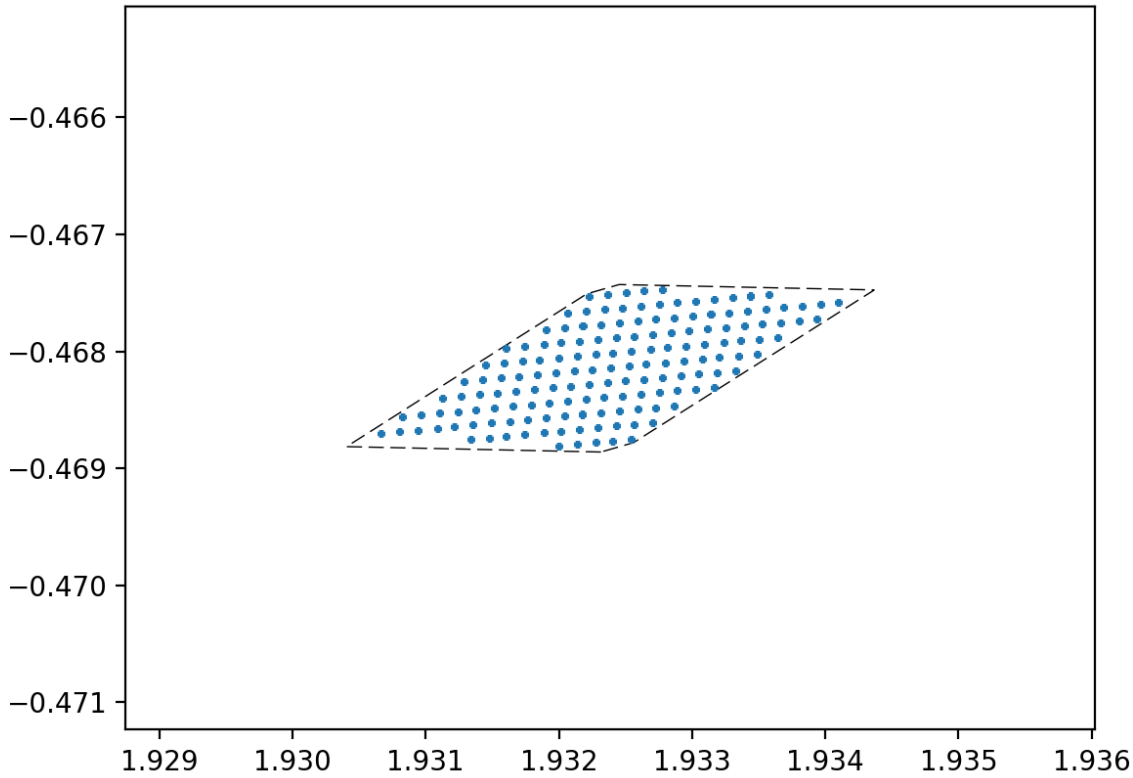
ENSTA IP PARIS

VanDerPol (initial condition)



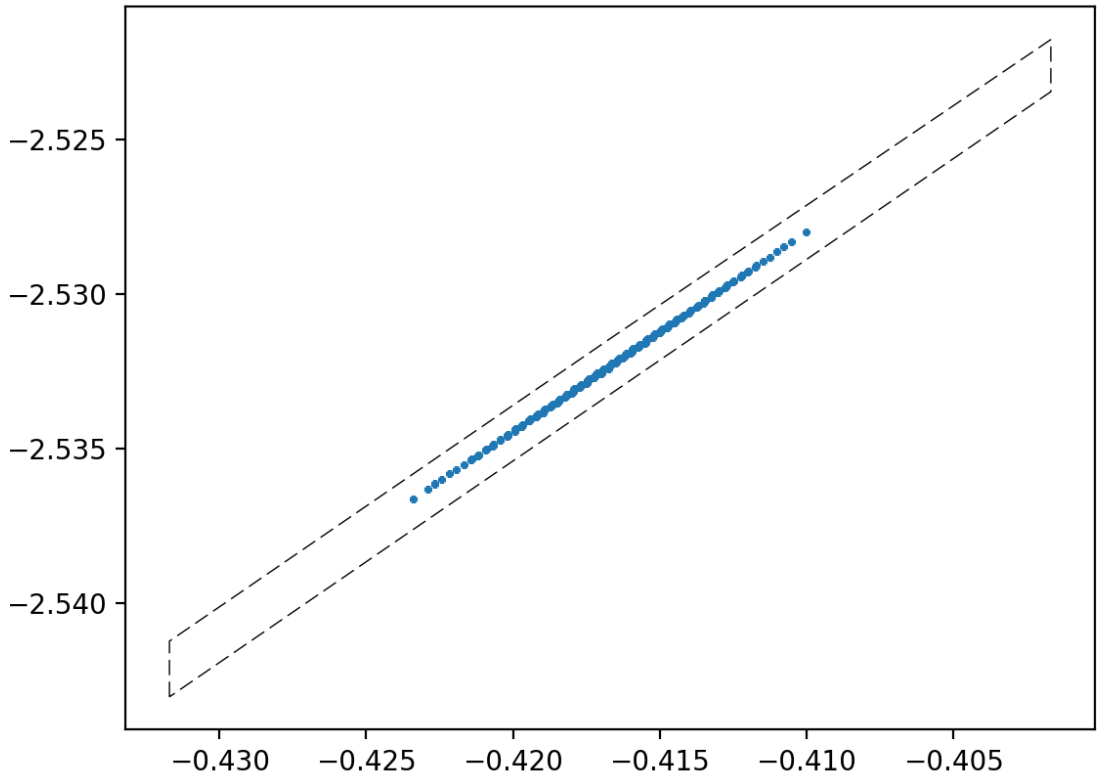
With polytopes

VanDerPol (after 1 seconds)



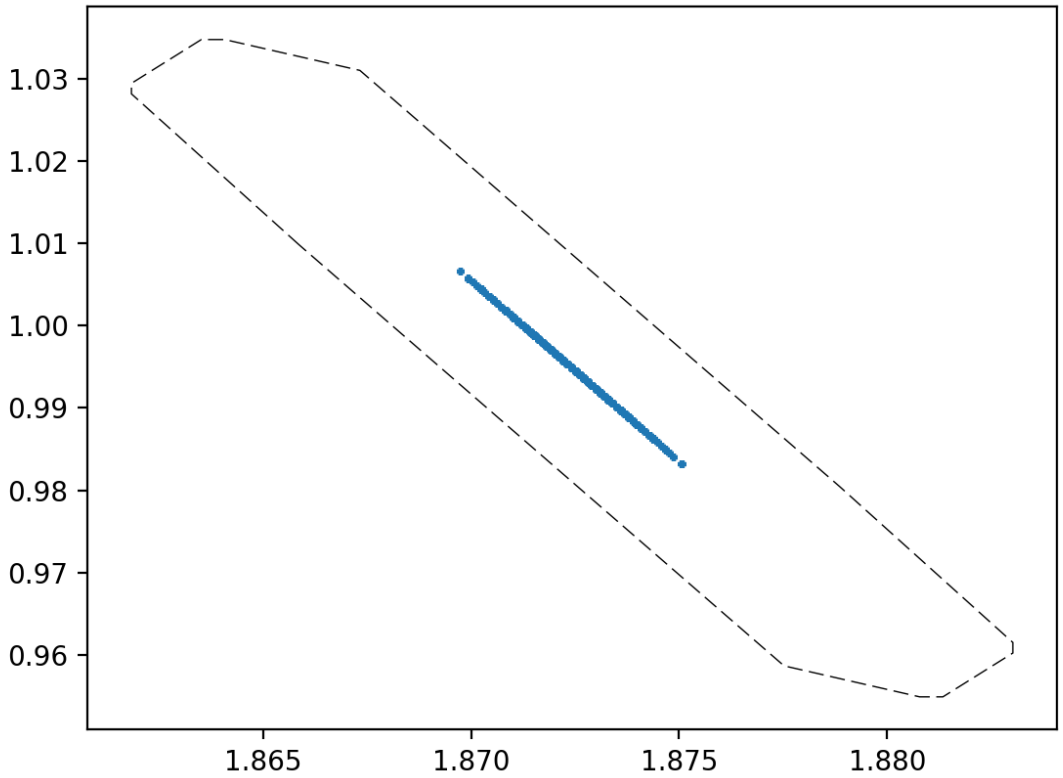
With polytopes

VanDerPol (after 3 seconds)



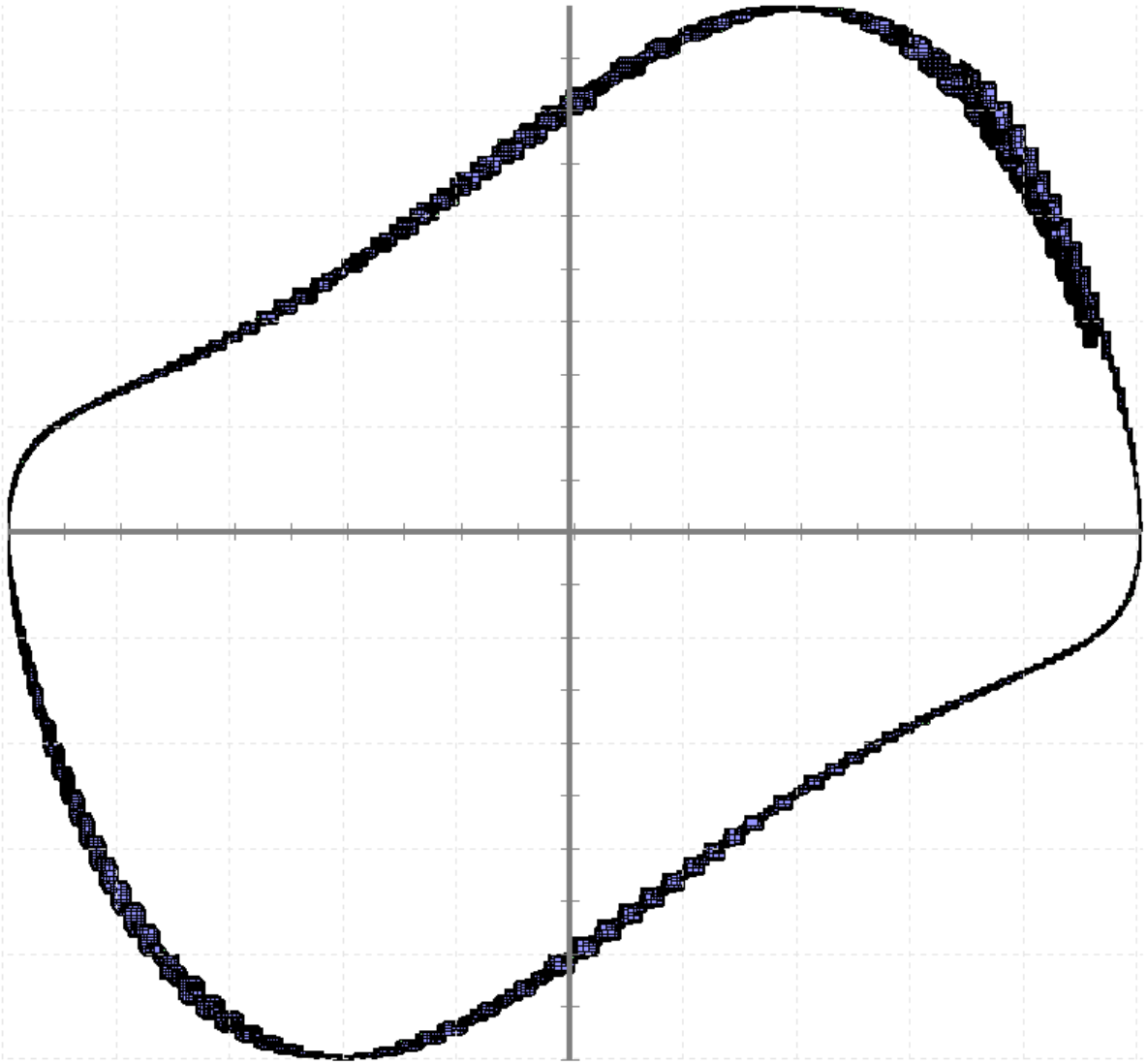
With polytopes

VanDerPol (after 7 seconds)



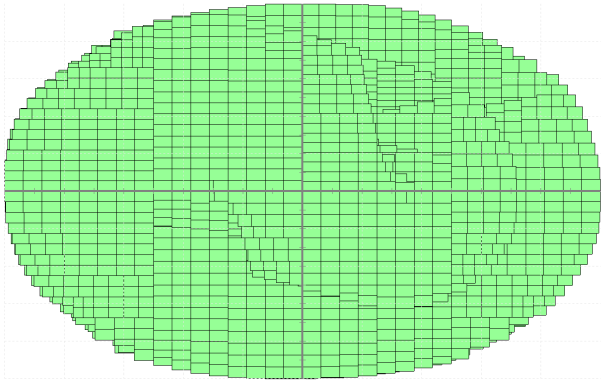
With polytopes

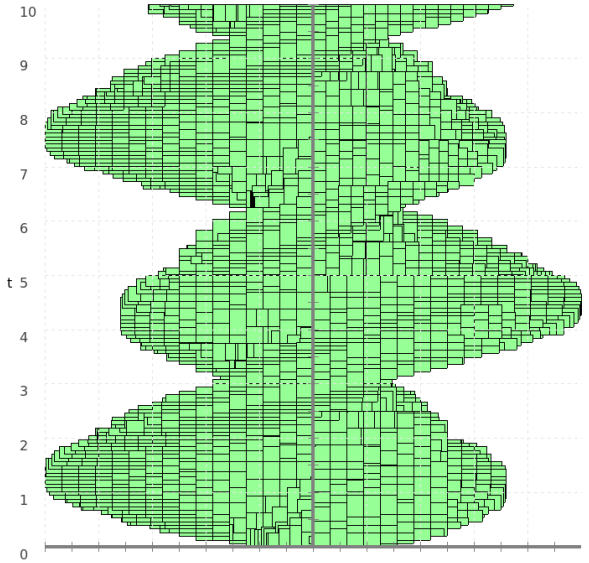
VanDerPol (tube)



With polytopes

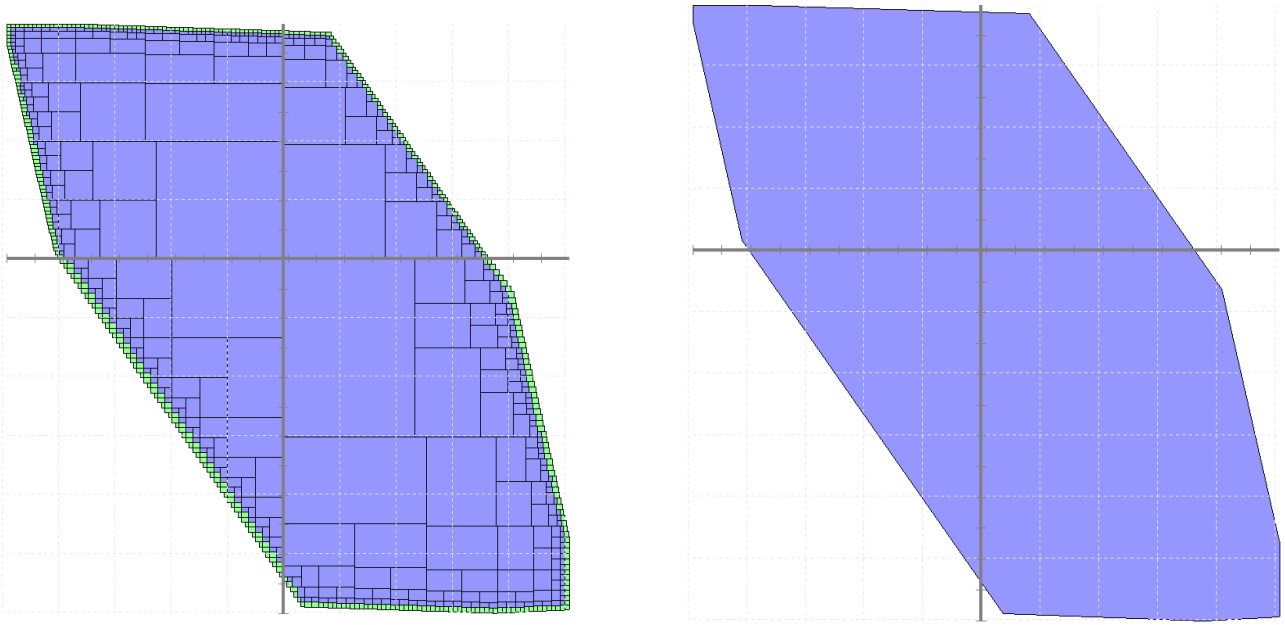
Another example, the circle





With polytopes

- Drawbacks: concretization very expensive (paving with boxes vs polytope)
- But perfect for detection collision (we hope no solution)



Set based B-series

Gathered in one formalism !

B-series

A B-series is a formal series of the form

$$B(a, hf, y) = a(\emptyset)y + \sum_{\tau \in \mathcal{T}} \frac{a(\tau)h^{|\tau|}}{\sigma(\tau)} F(\tau)(y),$$

with $F(.)(y)$ the elementary differentials.

Lemma (B-series and IVP)

The exact solution of an IVP is given by a B-series

$$y(h) = B(E, hf, y_0), \text{ with } E(\tau) = 1/\tau!$$

Different abstractions

To enclose the same set of trajectories!

One formalism: Set-based B-series

For differential inclusion:

$$\dot{y} \in \mathcal{F}(y), y(0) \in \mathcal{Y}_0$$

- Prop1: A set-based B-series $B(E, h\mathcal{F}, \mathcal{Y}_0)$ (with $E(\tau) = 1/\tau!$) is the exact reachable set of the differential inclusion.
  That is to say $\mathcal{Y}(h) \equiv B(E, h\mathcal{F}, \mathcal{Y}_0)$

Different abstractions

One formalism: Set-based B-series

- Prop2: $\bigcup_{f \in \mathcal{F}, y_0 \in \mathcal{Y}_0} B(E, hf, y_0) \equiv B(E, h\mathcal{F}, \mathcal{Y}_0)$.

Different abstractions

One formalism: Set-based B-series

- Prop3: Validated truncated set-based B-series
  $\mathcal{Y}(h) = B(E, h\mathcal{F}, \mathcal{Y}_0) \subset B(E_{0,\dots,p}, h\mathcal{F}, \mathcal{Y}_0) + B(E_{p+1}, h\mathcal{F}, \mathcal{Y}^*)$, with $E_{0,\dots,p} = 1/\tau!$ if $|\tau| \leq p$ and $E_{p+1} = 1/\tau!$ if $|\tau| = p+1$. $\mathcal{Y}^*$ such that $\mathcal{Y}(x) \subset \mathcal{Y}^*, x \in [0, h]$

Different abstractions

One formalism: Set-based B-series

- Prop4: Picard-Lindelöf $B(E, [0, h]\mathcal{F}, \mathcal{R}) \subset \mathcal{R}$, then $\mathcal{Y}^* \subset \mathcal{R}$

Different abstractions

One formalism: Set-based B-series

- Prop5: Let consider a Runge-Kutta scheme at order p and defined by the Butcher tableau (A, b):
  $\mathcal{Y}(h) = B(E, h\mathcal{F}, \mathcal{Y}_0) \subset B(\phi(A, b), h\mathcal{F}, \mathcal{Y}_0) + B(\psi(A, b), h\mathcal{F}, \mathcal{Y}^*)$,
  with $\psi(A, b)(\tau) = 1/\tau! - \phi(A, b)(\tau), \forall \tau \in \mathcal{T}$ if $|\tau| = p + 1$, or
  $\psi(A, b)(\tau) = 0$ if $|\tau| \neq p + 1$

Different abstractions

One formalism: Set-based B-series

- Prop6: if $\mathcal{Y}_0$ enclosed by a polytope $\mathcal{P}_0$ and if $\mathcal{P}_0 = \mathcal{Z}^0 \cap \mathcal{Z}^1 \cap \cdots \cap \mathcal{Z}^z$, then $\mathcal{Y}(h) = B(E, h\mathcal{F}, \mathcal{Y}_0) \subset B(E, h\mathcal{F}, \mathcal{P}_0) = B(E, h\mathcal{F}, \mathcal{Z}^0) \cap B(E, h\mathcal{F}, \mathcal{Z}^1) \cap \cdots \cap B(E, h\mathcal{F}, \mathcal{Z}^z)$

More details in *Alexandre dit Sandretto, Julien. 2022. "Set-Based B-Series" Mathematics 10, no. 17: 3165*

Truncation error

Complexity of the truncation error computation is an issue...

Truncation error complexity

Some ideas (and discussions)

- Optimal RK methods (smaller LTE $\Rightarrow$ larger steps $\Rightarrow$ fewer LTE computations)
- Condition on tableau to optimize tree complexity (and reduce LTE computation)
- Symbolic pre-computation of the LTE for a specific ODE
- RK pairs to estimate LTE

Symbolic pre-computation

A difficult problem: coupled VanDerPol:

$$\begin{aligned}
\dot{x}_1 &= y_1 \\
\dot{y}_1 &= \mu(1-x_1^2)y_1 + b(x_2 - x_1) - x_1 \\
\dot{x}_2 &= y_2 \\
\dot{y}_2 &= \mu(1-x_2^2)y_2 - b(x_2 - x_1) - x_2 \\
\dot{b} &= 0
\end{aligned}$$

with $\mu = 1$.
The initial condition is $x_{1,2}(0) \in [1.25, 1.55]$, $y_{1,2}(0) \in [2.35; 2.45]$ and $b(0) \in [1, 3]$, and the simulation horizon is $[0, 7]$.

Goal

Prove that $y_{1,2}([0, 7])$ stayed in the box $[-\infty, 2.75]$

Symbolic pre-computation

The local truncation error for RK4 (substraction of truncated B-series and exact B-series)[1]

$$\frac{-1}{120}h^5F_f\left(\right) + \frac{1}{480}h^5F_f\left(\right) + \frac{-1}{240}h^5F_f\left(\right) +$$

$$\frac{1}{120}h^5F_f\left(\right) + \frac{-1}{720}h^5F_f\left(\right) + \frac{-1}{480}h^5F_f\left(\right) + \frac{1}{160}h^5F_f\left(\right) + \frac{1}{480}h^5F_f\left(\right) + \frac{1}{2880}h^5F_f\left(\right)$$

[1] Julia's package for B-series

Symbolic pre-computation

First and second dimensions of LTE specific for RK4 and coupled VDP:

```
h^5*(1.0 - y1^2)*(-4*y1*y2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)) - 2*y2^3)/480 - h^5*(-4*y2^2*(-y1 + y2*(1.0 - y1^2) +
y5*(-y1 + y3)) + y2^2*(2*y1 - 2*y2*(1.0 - y1^2) - 2*y5*(-y1 + y3)))/720 - h^5*(y5*(y2*y5 + y4*(-2*y3*y4 - y5 - 1) +
(1.0 - y3^2)*(-y3 + y4*(1.0 - y3^2) + y5*(y1 - y3))) + (1.0 - y1^2)*(y5*(-y3 + y4*(1.0 - y3^2) + y5*(y1 - y3)) +
(1.0 - y1^2)*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) + (-y1 +
y2*(1.0 - y1^2) + y5*(-y1 + y3))*(-2*y1*y2 - y5 - 1)) + (-2*y1*y2 - y5 - 1)*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 +
(1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))))/120 - h^5*(-2*y1*y2*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 +
(1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) - 2*y1*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))^2 -
2*y2^2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)))/240

h^5*y1*y2*(-4*y1*y2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)) - 2*y2^3)/240 - h^5*(1.0 - y1^2)*(-4*y2^2*(-y1 +
y2*(1.0 - y1^2) + y5*(-y1 + y3)) + y2^2*(2*y1 - 2*y2*(1.0 - y1^2) - 2*y5*(-y1 + y3)))/720 - h^5*(1.0 -
y1^2)*(-2*y1*y2*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) -
2*y1*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))^2 - 2*y2^2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)))/240 +
h^5*(y2^2*(-2*y2*(-2*y1*y2 - y5 - 1) - 2*y4*y5 - 2*(1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) +
2*y2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))*(2*y1 - 2*y2*(1.0 - y1^2) - 2*y5*(-y1 + y3)))/480 +
h^5*(-4*y1*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 -
y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) - 2*y2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))^2)/160 +
h^5*(y5*(-4*y3*y4*(-y3 + y4*(1.0 - y3^2) + y5*(y1 - y3)) - 2*y4^3) + (1.0 - y1^2)^2*(-4*y1*y2*(-y1 +
y2*(1.0 - y1^2) + y5*(-y1 + y3)) - 2*y2^3) + (-4*y1*y2*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)) -
2*y2^3)*(-2*y1*y2 - y5 - 1))/480 - h^5*(y5*(y5*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)) + (1.0 - y3^2)*(y2*y5 +
y4*(-2*y3*y4 - y5 - 1) + (1.0 - y3^2)*(-y3 + y4*(1.0 - y3^2) + y5*(y1 - y3))) + (-y3 + y4*(1.0 - y3^2) +
y5*(y1 - y3))*(-2*y3*y4 - y5 - 1)) + (1.0 - y1^2)*(y5*(y2*y5 + y4*(-2*y3*y4 - y5 - 1) + (1.0 - y3^2)*(-y3 +
y4*(1.0 - y3^2) + y5*(y1 - y3))) + (1.0 - y1^2)*(y5*(-y3 + y4*(1.0 - y3^2) + y5*(y1 - y3)) + (1.0 -
y1^2)*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) + (-y1 +
y2*(1.0 - y1^2) + y5*(-y1 + y3))*(-2*y1*y2 - y5 - 1)) + (-2*y1*y2 - y5 - 1)*(y2*(-2*y1*y2 - y5 - 1) +
y4*y5 + (1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3)))) + (-2*y1*y2 - y5 - 1)*(y5*(-y3 + y4*(1.0 -
y3^2) + y5*(y1 - y3)) + (1.0 - y1^2)*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 - y1^2)*(-y1 + y2*(1.0 -
y1^2) + y5*(-y1 + y3))) + (-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))*(-2*y1*y2 - y5 - 1)))/120 +
h^5*(-2*y1*y2*(y5*(-y3 + y4*(1.0 - y3^2) + y5*(y1 - y3)) + (1.0 - y1^2)*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 +
(1.0 - y1^2)*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))) + (-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))*(-2*y1*y2 -
y5 - 1)) - 2*y1*(-y1 + y2*(1.0 - y1^2) + y5*(-y1 + y3))*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 - y1^2)*(-y1 +
y2*(1.0 - y1^2) + y5*(-y1 + y3))) - 2*y2^2*(y2*(-2*y1*y2 - y5 - 1) + y4*y5 + (1.0 - y1^2)*(-y1 + y2*(1.0 -
y1^2) + y5*(-y1 + y3))))/120
```

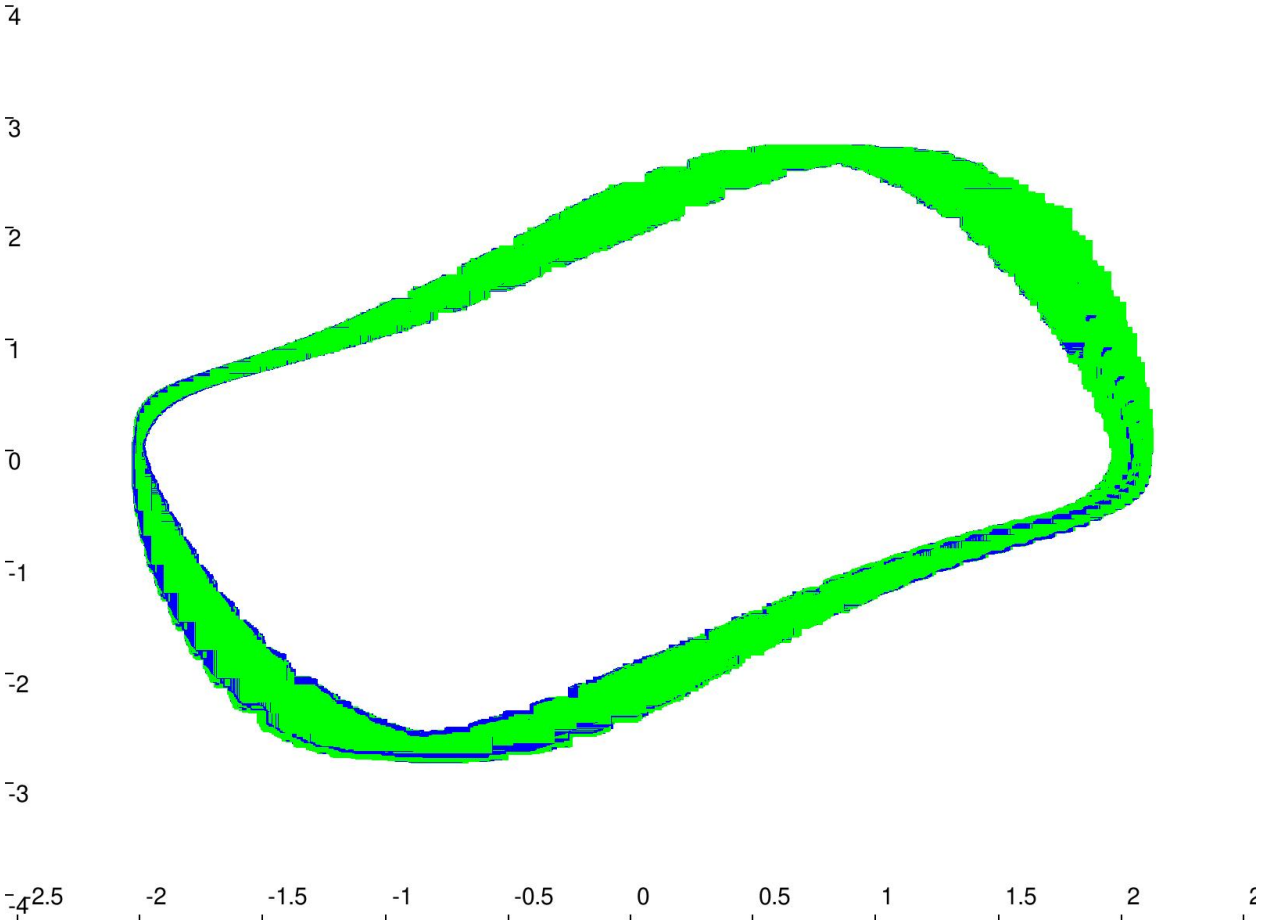
Symbolic pre-computation

Huge...but can be simplified, factorized, etc...and optimized by C++ compiler.

Results:

- Frechet derivatives (Faà di bruno): 395 bisections, 215m10s
- Automatic differentiation: 349 bisections, 35m52s
- Compiled LTE: 363 bisections, 18m53s

Symbolic pre-computation



Conclusion

Some advancements on a unified formalism and set abstraction

Ideas but no general results for LTE complexity...

Great! Many things to work on!

Questions?