Guranteed numerical integration based on explicit methods of Runge-Kutta

Alexandre Chapoutot

U2IS, ENSTA ParisTech

May 14, 2014

Context

Scientific context

Fact: Simulink is a *de facto* standard in industry for model-based design of control-command systems because:

- it can model and numerically simulate hybrid systems, i.e. it mixes **ordinary differential equations** (ODE) and (very generally) **state transition systems**.

Goal

Defining and applying **formal verification methods** on Simulink models.

Main challenges

- **Solve ODEs for sets of initial values and bounded parameters**.
- Handle interactions between continuous-time and discrete-time components.

Numerical solution of IVP

Goal of numerical integration

Recall, we consider IVP (initial value problem):

$$\dot{\mathbf{x}} = f(t, \mathbf{x}) \quad \text{with} \quad x(0) = \mathbf{x}_0 \ . \tag{1}$$

This problem (Cauchy problem) admits a unique solution $x(t; \mathbf{x}_0)$ on $\mathbb{R}$, if $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n$ is continuous in t and Lipschitz in $\mathbf{x}$ that is:

$$\forall t, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n, \exists L > 0, \quad \| f(t, \mathbf{x}_1) - f(t, \mathbf{x}_2) \| \leq L \| \mathbf{x}_1 - \mathbf{x}_2 \| \ .$$

Goal

- Compute a sequence of time instants $t_0 \leq t_1 \leq \cdots \leq t_n$
- Compute a sequence of values $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$ such that

$$\forall i \in [0, n], \quad x_i \approx x(t_i; \mathbf{x}_0) \ .$$

Remark:

$$\dot{\mathbf{x}} = f(t, \mathbf{x}) \Leftrightarrow \dot{\mathbf{z}} = \begin{pmatrix} \dot{\mathbf{x}} \\ \dot{t} \end{pmatrix} = \begin{pmatrix} f(t, \mathbf{x}) \\ 1 \end{pmatrix} = g(\mathbf{z})$$

Example of numerical integration method: Euler's method

Consider a simple IVP:

$$\dot{x} = -\frac{x^3}{2} \quad \text{with} \quad x(0) = 1 \ .$$

The exact solution is $x(t) = \frac{1}{\sqrt{1+t}}$.

We consider two fixed-step methods (Euler and Heun) that is they compute the sequence of time instants such that $t_{i+1} = t_i + h$.

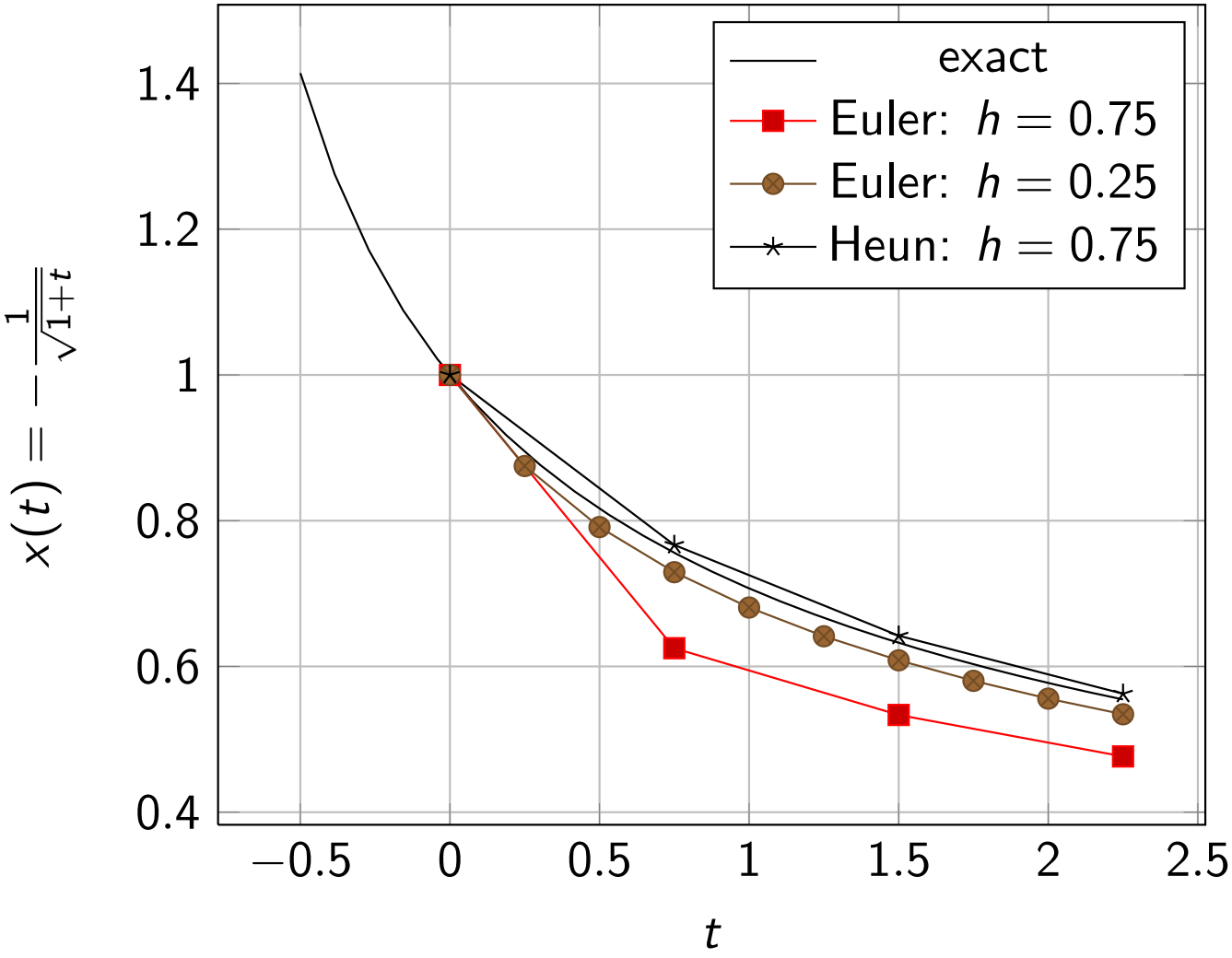
- The Euler's method computes the sequence of values

$$x_{i+1} = x_i + h \times -\frac{x_i^3}{2}$$

- The Heun's method computes the sequence of values:

$$k_1 = x_i + h \times -\frac{x_i^3}{2}$$
$$x_{i+1} = x_i + \frac{h}{2} \times \left(\left(-\frac{x_i^3}{2} \right) + \left(-\frac{k_1^3}{2} \right) \right)$$

Example of numerical integration method: Euler's method



Remark: precision vs performance in the application of Euler's method.

Goal of the guaranteed numerical integration

Set of solutions of IVP

We consider $\dot{x} = f(x), \quad x(0) \in \boldsymbol{x_0}$ with $\boldsymbol{x_0} \in \mathbb{IR}^n$.

Solutions:

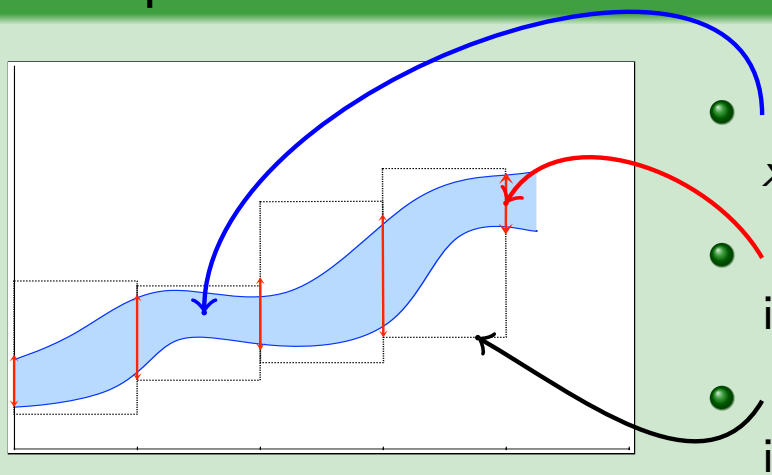
$\mathcal{X} = \{x \mid x_0 \in \boldsymbol{x_0}$ such that x is solution of $\dot{x} = f(x), x(0) = x_0\}$

Problem

Find a sequence of values $(t_n, \boldsymbol{x_n})$ such that

$$\forall x \in \mathcal{X},\ \forall n,\ x(t_n) \in \boldsymbol{x_n}$$

Example



- Exact solution of $\dot{x} = f(x)$ with $x(0) \in \mathcal{X}$.
- Safe approximation at discrete time instants.
- Safe approximation between time instants.

Guaranteed numerical integration: Taylor methods

One slide on interval analysis

- Interval analysis extends arithmetic operations and elementary functions to intervals.
- **Inclusion function:**

$$\forall \mathbf{x} \in [\mathbf{x}], \quad f(\mathbf{x}) \in [f]([\mathbf{x}]).$$

Operation	Interval arithmetic
$[a, b] + [c, d]$	$[a + c, b + d]$
$[a, b] - [c, d]$	$[a - d, b - c]$
$[a, b] \times [c, d]$	$[\min(E), \max(E)]$ with $E = (ac, ad, bc, bd)$
$[a, b] \div [c, d]$	$\left[\frac{a}{d}, \frac{b}{c}\right]$ if $0 \neq [c, d]$

Remark: substituting all operations in the definition of a function by their interval counter-part generates an inclusion function

Guaranteed integration with Taylor method

Basics

- Assume $x(t_n) \in [\boldsymbol{x}_n]$ and $h_{n+1} = t_{n+1} - t_n$.
- The Taylor expansion of x :

$$
\begin{aligned}
x(t_{n+1}) &= x(t_n) + \sum_{i=1}^{N-1} \frac{h_{n+1}^i}{i!} \frac{d^i x}{dt^i}(t_n) + \frac{h_{n+1}^N}{N!} \frac{d^N x}{dt^N}(t') \\
&\in [\boldsymbol{x}_n] + \sum_{i=1}^{N-1} h_{n+1}^i f^{[i-1]}\big(x(t_n)\big) + h_{n+1}^N f^{[N-1]}\big(x(\xi)\big) \\
&\in [\boldsymbol{x}_n] + \sum_{i=1}^{N-1} h_{n+1}^i [f^{[i-1]}]([\boldsymbol{x}_n]) + h_{n+1}^N [f^{[N-1]}]([\tilde{\boldsymbol{x}}_n]) \triangleq \boldsymbol{x}_{n+1}
\end{aligned}
$$

Challenges

- Compute $[\tilde{\boldsymbol{x}}_n]$ such that $\forall t \in [t_n, t_{n+1}],\ x(t) \in [\tilde{\boldsymbol{x}}_n]$. (**Solution:** Picard-Lindelöf operator and Banach fixpoint theorem)
- Note that, $w([\boldsymbol{x}_{n+1}]) \geq w([\boldsymbol{x}_n)]$.

Banach fixpoint theorem and Picard-Lindelöf operator

To bound the term $f^{(N+1)}(x(\xi))$ requires to know a bound of the solution $x(t; x_0)$ on $[t_n, t_{n+1}]$.

Banach fixed-point theorem

Let (K, d) a complete metric space and let $g : K \to K$ a contraction that is for all x, x in K there exists $c \in]0, 1[$ such that $d(g(x), g(x)) \leq c \cdot d(x, x)$, then g has a unique fixed-point in K.

Picard-Lindelöf operator

$$P(f; t_n; x_n)(t) = x_n + \int_{t_n}^{t} f(x(s))ds$$

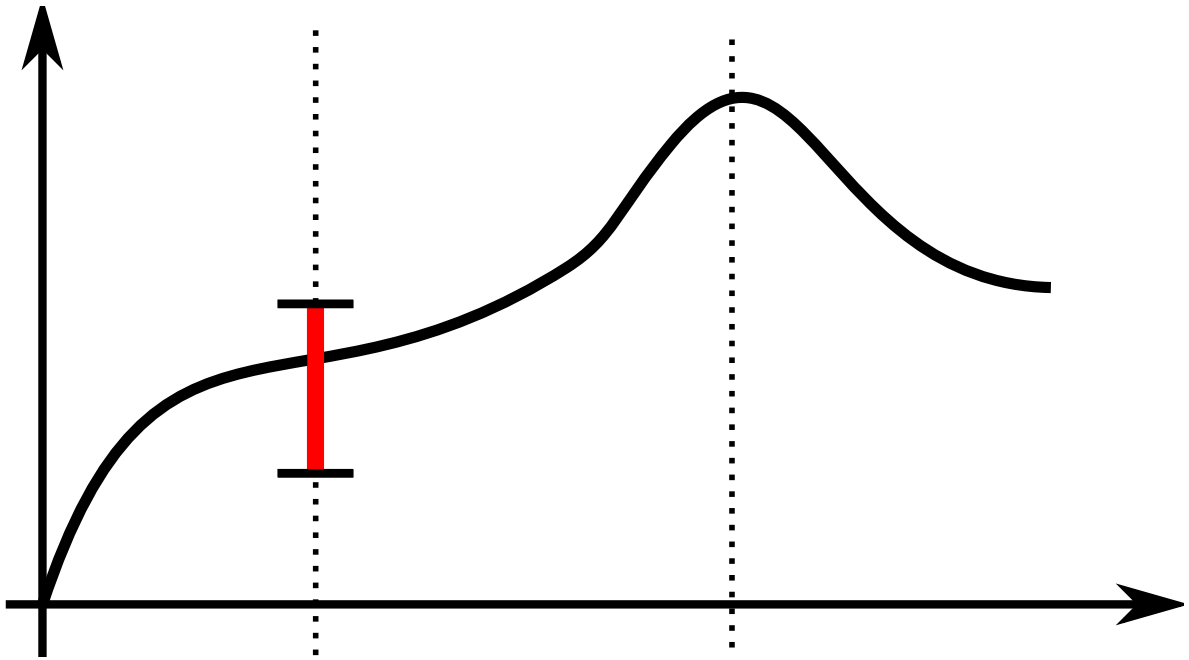
Remark: using interval arithmetic we can compute by iteration a solution of the Picard-Lindelöf operator.

Computing $\tilde{\boldsymbol{x}}_n$.

- We use the interval Picard-Lindelöf operator:

$$\Psi(\boldsymbol{R}) = \boldsymbol{x}_n + [0, h].[f](\boldsymbol{R})$$

- If we can find $\boldsymbol{R}_1$ such that $\Psi(\boldsymbol{R}_1) \subseteq \boldsymbol{R}_1$, then the IVP admits a unique solution on $[t_n, t_{n+1}]$ and this solution is contained in $\boldsymbol{R}_1$.

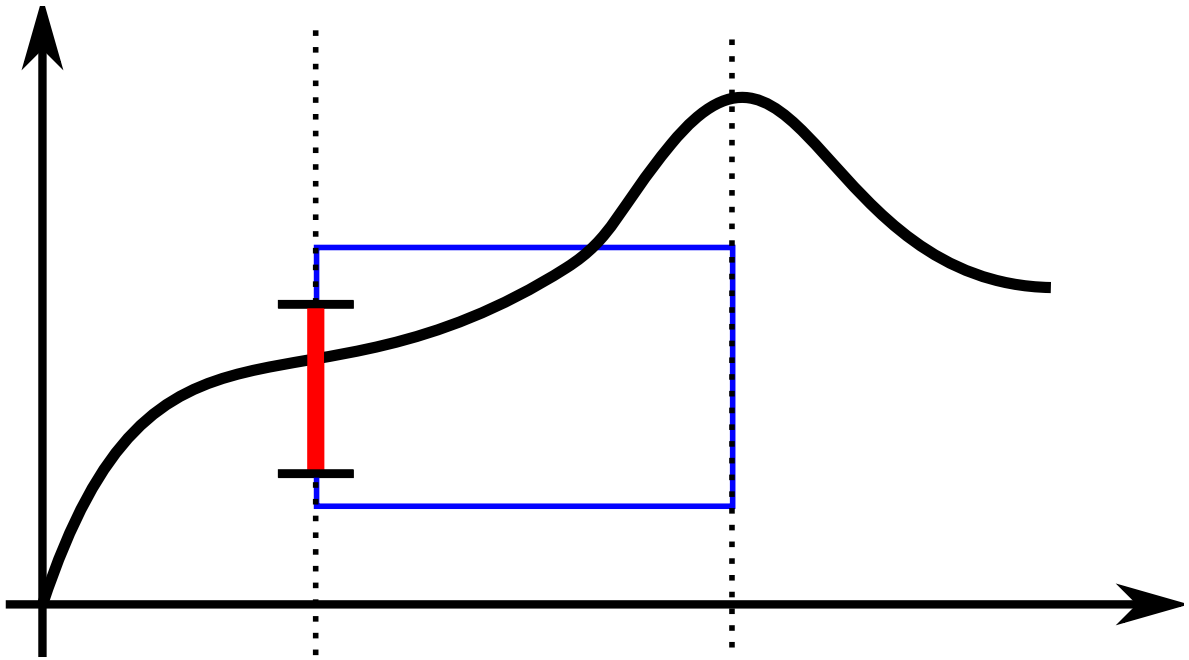


Computing $\tilde{\boldsymbol{x}}_n$.

- We use the interval Picard-Lindelöf operator:

$$\Psi(\boldsymbol{R}) = \boldsymbol{x}_n + [0, h].[f](\boldsymbol{R})$$

- If we can find $\boldsymbol{R}_1$ such that $\Psi(\boldsymbol{R}_1) \subseteq \boldsymbol{R}_1$, then the IVP admits a unique solution on $[t_n, t_{n+1}]$ and this solution is contained in $\boldsymbol{R}_1$.

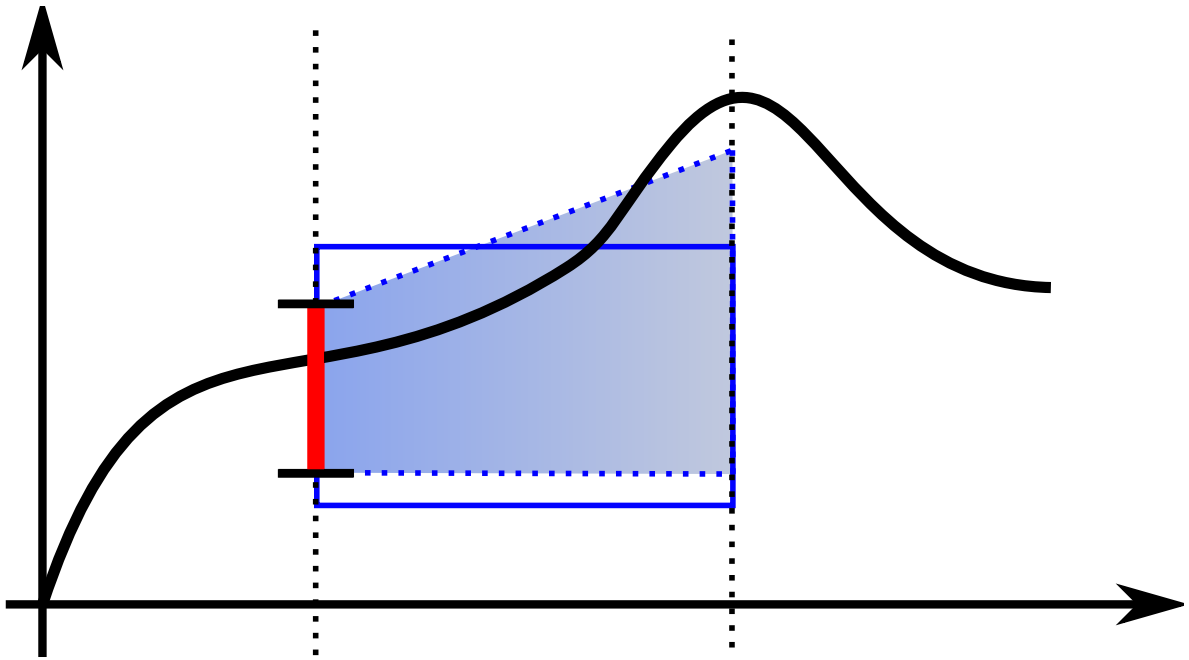


Computing $\tilde{\boldsymbol{x}}_n$.

- We use the interval Picard-Lindelöf operator:

$$\Psi(\boldsymbol{R}) = \boldsymbol{x}_n + [0, h].[f](\boldsymbol{R})$$

- If we can find $\boldsymbol{R}_1$ such that $\Psi(\boldsymbol{R}_1) \subseteq \boldsymbol{R}_1$, then the IVP admits a unique solution on $[t_n, t_{n+1}]$ and this solution is contained in $\boldsymbol{R}_1$.

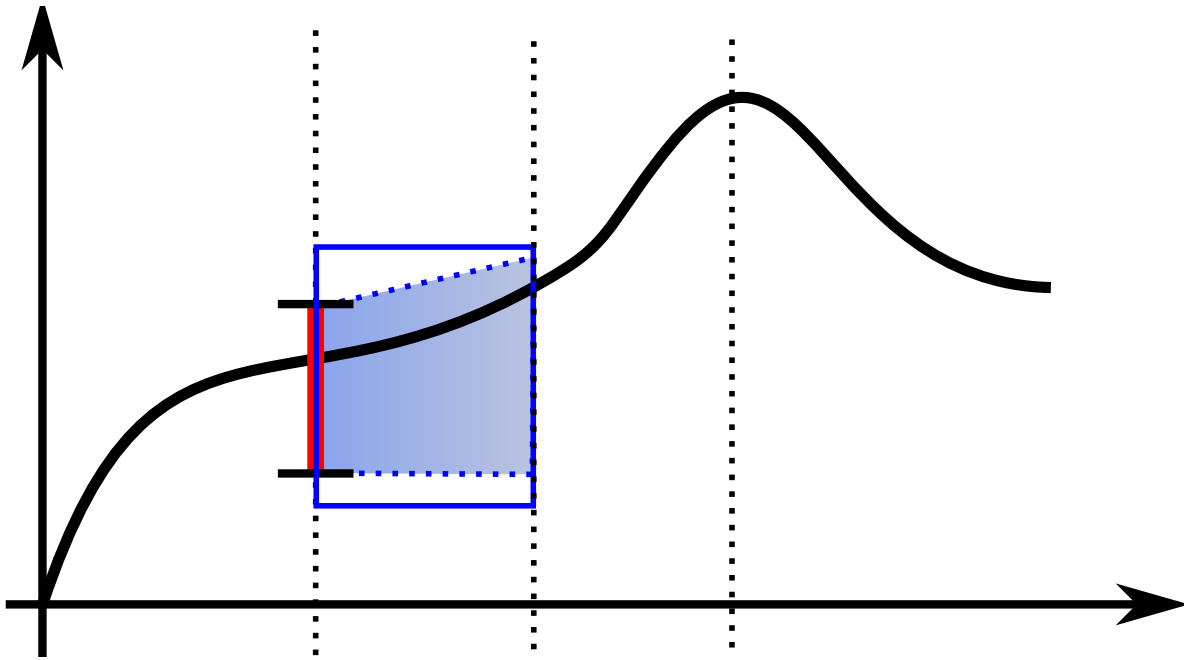


Computing $\tilde{\boldsymbol{x}}_n$.

- We use the interval Picard-Lindelöf operator:

$$\Psi(\boldsymbol{R}) = \boldsymbol{x}_n + [0, h].[f](\boldsymbol{R})$$

- If we can find $\boldsymbol{R}_1$ such that $\Psi(\boldsymbol{R}_1) \subseteq \boldsymbol{R}_1$, then the IVP admits a unique solution on $[t_n, t_{n+1}]$ and this solution is contained in $\boldsymbol{R}_1$.



Reduction of the width of $[\boldsymbol{x}_{n+1}]$.

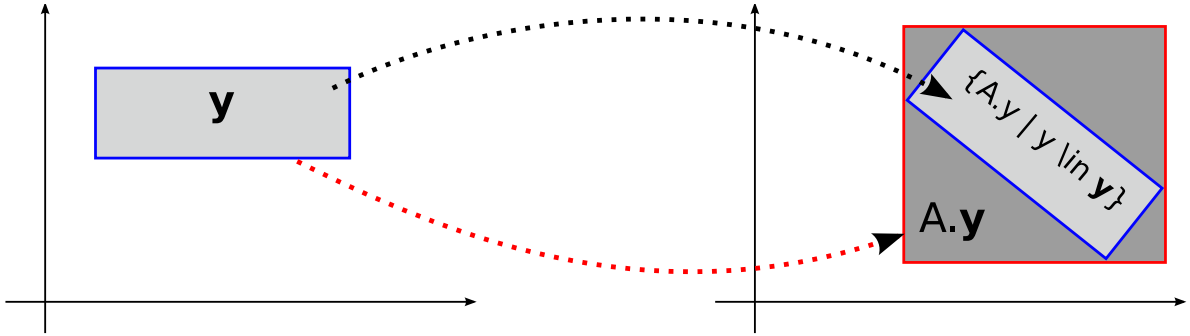
We want to evaluate:

$$[\boldsymbol{x}_{n+1}] \triangleq [\boldsymbol{x}_n] + \sum_{i=1}^{N-1} h_{n+1}^{i}[f^{[i-1]}]([\boldsymbol{x}_n]) + h_{n+1}^{N}[f^{[N-1]}]([\tilde{\boldsymbol{x}}_n]) \tag{2}$$

- To reduce the width of $[\boldsymbol{x}_{n+1}]$, we use centered form to compute $f^{[i]}\left(\boldsymbol{x}_n\right)$.

$$f\left([\boldsymbol{x}]\right) \in f\left(m([\boldsymbol{a}])\right) + J\left(f, [\boldsymbol{a}]\right).([\boldsymbol{a}] - m(\boldsymbol{a}))$$

  After rewriting we get a linear expressions of the form:
  $[\boldsymbol{x}_{n+1}] = \boldsymbol{v}_n + \boldsymbol{A}_n \boldsymbol{r}_n$
- Fighting the wrapping effect (Löhner's method):



Motivation for new guaranteed integration methods

- Benefit of the well-known properties of Runge-Kutta methods, e.g.
  - Stability: A-stable, L-Stable, Algebraic stable;
  - Avoid spurious fixed-point;
  - Preservation of algebraic invariant, e.g.,:

$$\dot{\mathbf{x}} = f(t, \mathtt{x}) \quad \text{with} \quad x(0) = \mathbf{x}_0 \quad \text{and such that} \quad g(\mathbf{x}(t); \mathbf{x}_0) = 0 \forall t \geq 0$$

- A better adoption of the guaranteed integration methods by the engineers

Representation of sets of values

Representation of sets

- Extension of **interval arithmetic** to reduce the dependency issue.

$$[a, b] + [c, d] \quad = \quad [a + c, b + d]$$

$$x = [0, 1] \quad \Rightarrow \quad x - x = [-1, 1]$$

- Main idea: parametric variables w.r.t. a set of **noise symbols** ε_i with $\varepsilon_i \in [-1, 1]$.

$$\begin{aligned} x &= x_0 + x_1\varepsilon_1 + x_3\varepsilon_3 \\ y &= y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 \end{aligned}$$

- This representation encodes the linear relation between noise symbols and variables and allows for precise linear transformation.
- What is a noise symbol ?
  - Initial uncertainty: $[a, b] \rightarrow \frac{a+b}{2} + \frac{b-a}{2}\varepsilon$
  - Non-linear operations and round-off errors.

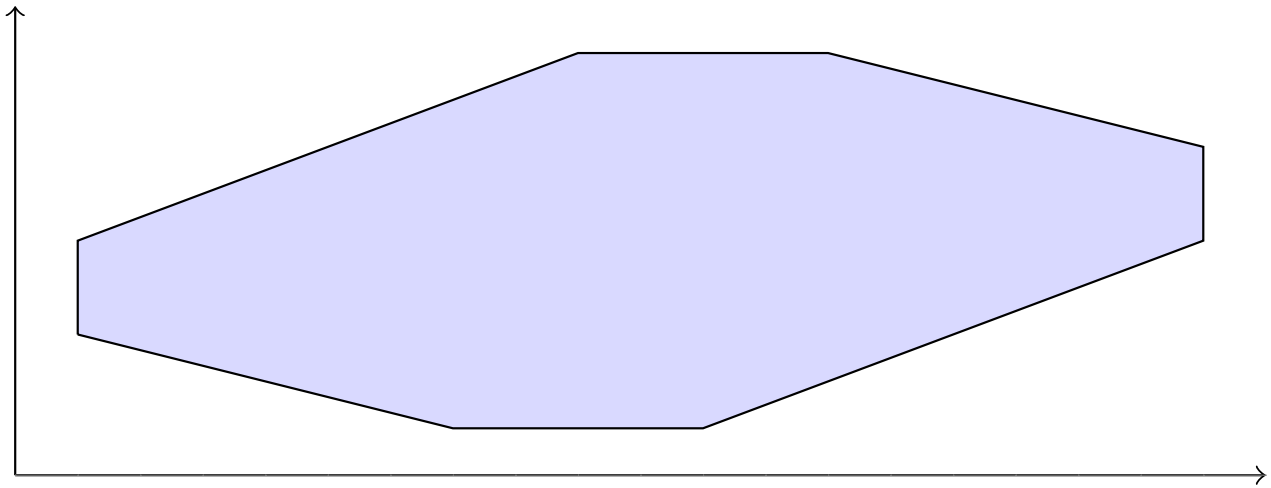
Geometric representation

Given m variables $x^1, x^2, \ldots, x^m$, an affine set

$$x^j = x_0^j + \sum_{i=1}^{n} x_i^j \varepsilon_i$$

is a **zonotope** $Z = \gamma(x^1, \ldots, x^m)$.

Example: $x = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4 \qquad y = 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4$



Affine sets arithmetic
Linear operations

Given an affine set

$$x = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i \qquad\qquad y = y_0 + \sum_{i=1}^{n} y_i \varepsilon_i$$

- Affine operations: α, β, γ are constants

$$\alpha x + \beta y + \gamma = \alpha x_0 + \beta y_0 + \gamma \sum_{i=1}^{n} (\alpha x_i + \beta y_i) \varepsilon_i$$

- Multiplication

$$\begin{aligned} x \times y &= x_0 y_0 + \sum_{i=1}^{n} (y_0 x_i + x_0 y_i) \varepsilon_i + \sum_{i=1}^{n} x_i \varepsilon_i \sum_{i=1}^{n} y_i \varepsilon_i \\ &= x_0 y_0 + \sum_{i=1}^{n} (y_0 x_i + x_0 y_i) \varepsilon_i + r_{n+1} \varepsilon_{n+1} \end{aligned}$$

- Other operations (sin, cos, ...) are computed using Taylor series.
- Union, intersection and inclusion test can also be defined.

Implementation

Goal of the implementation

1. Sound with respect to floating-point operations.
2. Efficient by limiting the number of noise symbols.

Floating-point operations:

- when we add two coefficients x_i and y_j, the result is approximated.
- we want to measure and collect this error
- **error free transformations**: error e attached to $a \oplus b$ is

$$e = (a \ominus (s \ominus (s \ominus a))) \oplus (b \ominus (s \ominus a)) \ .$$

- **collect the error**: each time $e \neq 0$, we create a new noise symbol.

Implementation

Goal of the implementation

1. Sound with respect to floating-point operations.
2. Efficient by limiting the number of noise symbols.

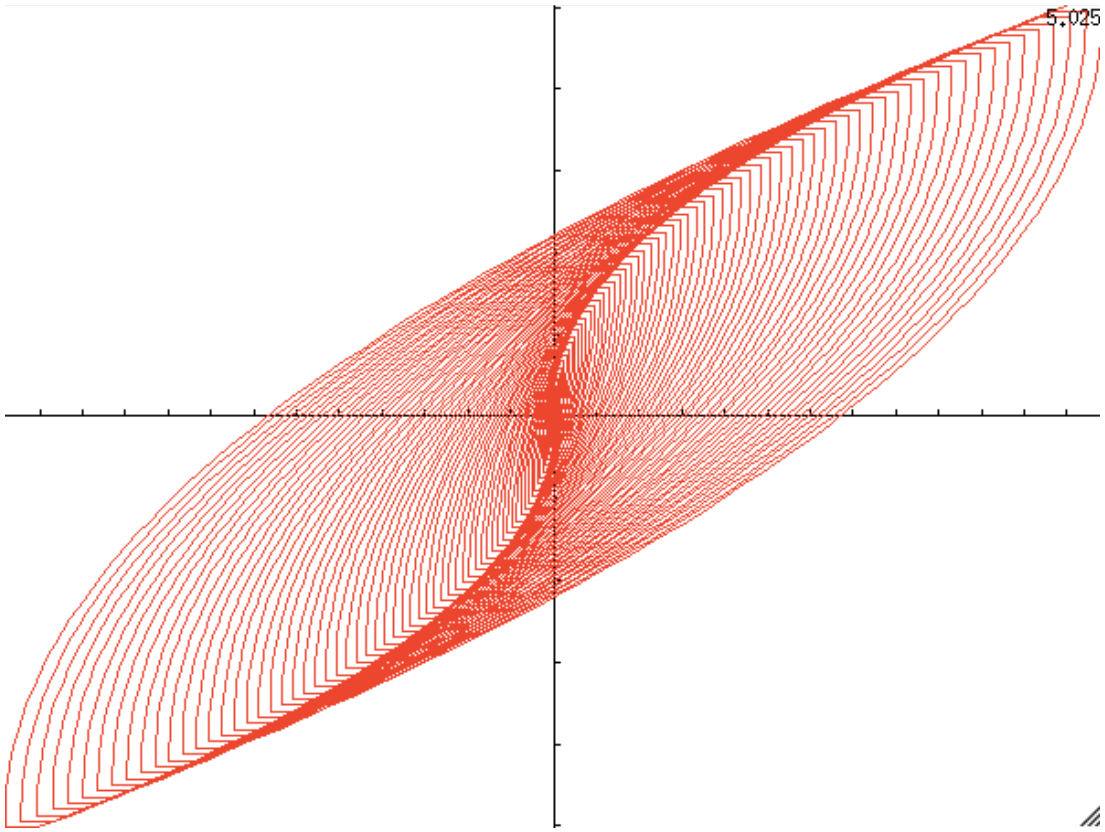
Efficiency:

- one new symbol per non-linear computation and error-prone floating-point operation.
- we chose a **sparse** representation of the coefficients attached to each variable.
- we only keep coefficients that are larger than a given threshold.
  $\rightarrow$ others are **accumulated** into a new noise symbol.

Examples: the strength of affine forms

Double integrators:

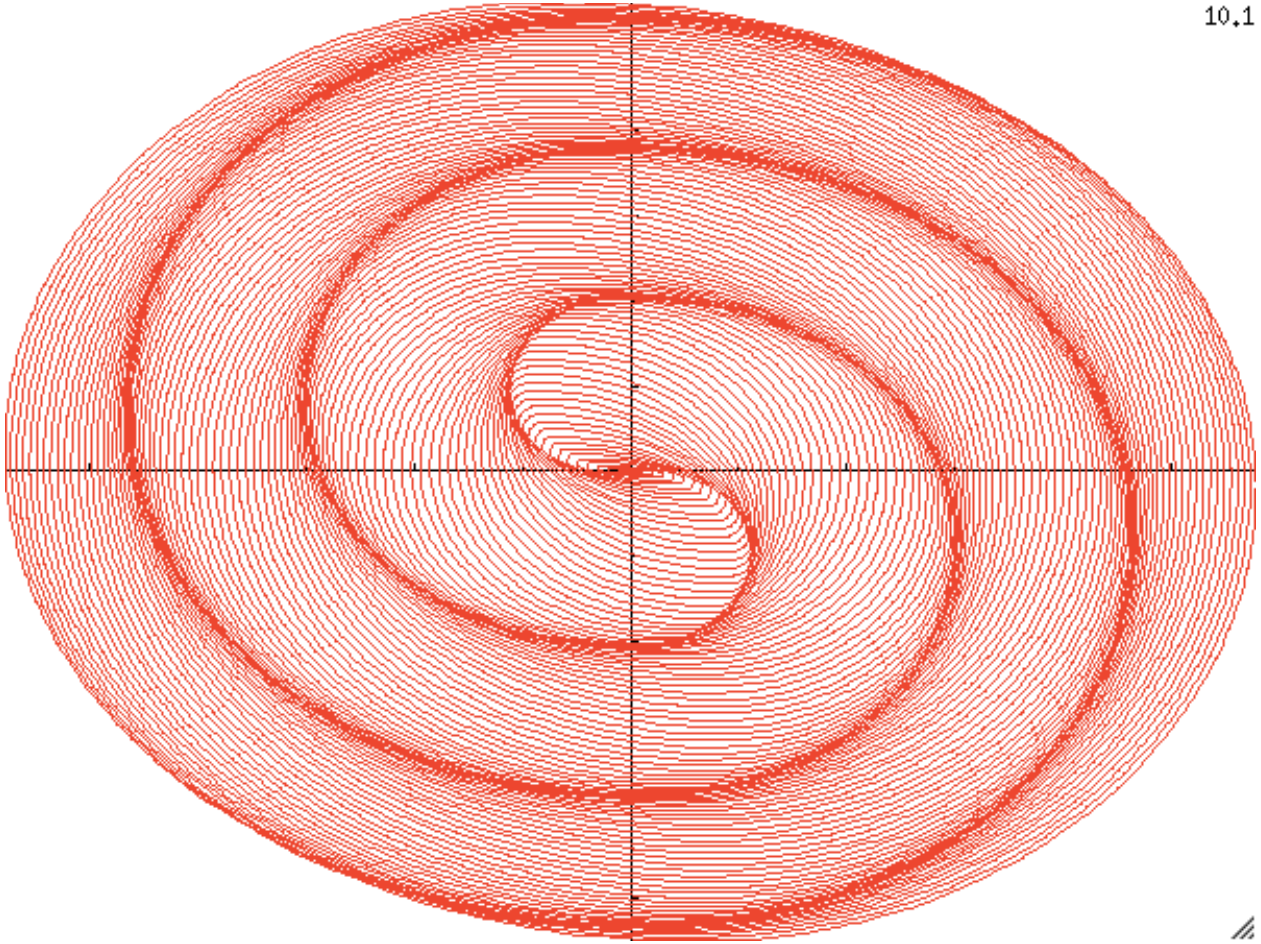
$$\ddot{x} = u \quad \text{with} \quad x(0) = 0, \quad \dot{x}(0) = 0, \quad u \in [-1, 1]$$



Examples: the strength of affine forms

Oscillator:

$\dot{x} = -v \quad \dot{v} = x - u$ with $x(0) = 0, \quad v(0) = 0, \quad u \in [-1, 1]$

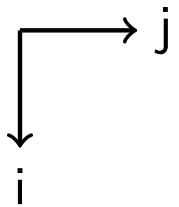


Guaranteed numerical integration: Runge-Kutta methods

Runge-Kutta methods

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

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \\ & b'_1 & b'_2 & \cdots & b'_s \end{array} \qquad \text{(optional)}$$



Which induces the following recurrence:

$$k_i = f\left(t_n + c_i h_n, x_n + h\sum_{j=1}^{s} a_{ij} k_j\right) \qquad x_{n+1} = x_n + h\sum_{i=1}^{s} b_i k_i \tag{3}$$

- **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 $a_{ii} \neq 0$
- **Singly Diagonal implicit** method (SDIRK) if $a_{ij} = 0$ is $i \leq j$ and all $a_{ii} = \gamma$ are identical.
- **Implicit** method (IRK) otherwise

Examples of Runge-Kutta methods: ERK

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

e.g. Euler's method is defined by:

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
x_{n+1} &= x_n + h1k_1
\end{aligned}
$$

$$
\begin{array}{c|c}
0 & \\
 & 1
\end{array}
$$

e.g. Heun's method is defined by:

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f(t_n + h_n, x_n + h1k_1) \\
x_{n+1} &= x_n + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)
\end{aligned}
$$

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

Examples of Runge-Kutta methods: ERK

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

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

$$k_1 = f(t_n, x_n)$$

$$k_2 = f(t_n + \frac{1}{2}h_n, x_n + \frac{1}{2}hk_1)$$

$$k_3 = f(t_n + \frac{3}{4}h_n, x_n + \frac{3}{4}hk_2)$$

$$x_{n+1} = x_n + h\left(\frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3\right)$$

$$k_4 = f(t_n + 1h_n, x_{n+1})$$

$$z_{n+1} = x_n + h\left(\frac{7}{24}k_1 + \frac{1}{4}k_2 + \frac{1}{3}k_3 + \frac{1}{8}k_4\right)$$

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{3}{4} & 0 & \frac{3}{4} & & \\
1 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
\hline
 & \frac{2}{9} & \frac{1}{3} & \frac{4}{9} & \\
 & \frac{7}{24} & \frac{1}{4} & \frac{1}{3} & \frac{1}{8}
\end{array}$$

Remark: the step-size h is adapted following $\| x_{n+1} - z_{n+1} \|$

Examples of Runge-Kutta methods: IRK

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

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

$$k_1 = f\left(t_n + \left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)h_n, \quad x_n + h\left(\frac{1}{4}k_1 + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)k_2\right)\right) \tag{4a}$$

$$k_2 = f\left(t_n + \left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)h_n, \quad x_n + h\left(\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)k_1 + \frac{1}{4}k_2\right)\right) \tag{4b}$$

$$x_{n+1} = x_n + h\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right) \tag{4c}$$

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

Remark 2: this kind of methods is not considered here.

Bounding the truncation error[1]

Goal: to bound the truncation error $\| x(t_i; x_0) - x_i \|$

Order condition of Runge-Kutta method

A method is of order p iff the $p+1$ first coefficients of the Taylor expansion of the solution and the Taylor expansion of the numerical methods are equal.

The truncation error is defined by:

$$x(t_n; x_0) - x_n = \frac{h_n^{p+1}}{(p+1)!} \left(f^{(p)}(\xi, x(\xi)) - \frac{d^{p+1}\phi}{dt^{p+1}}(\eta) \right)$$
$$\xi \in]t_k, t_{k+1}[\text{ and } \eta \in]t_n, t_{n+1}[\ . \quad (5)$$

with $\phi(t) = x_n + (t - t_n) \sum_{i=1}^{s} b_i k_i(t)$.

Problem: bounding the term $f^{(p)}(\xi, x(\xi))$ (**Solution**: Picard-Lindelöf operator)

[1] "Enclosing Temporal Evolution of Dynamical Systems Using Numerical Methods", NFM'13

Main algorithm

- Input:
  - a Butcher tableau of the explicit Runge-Kutta methods
  - the order p of the method
  - the problem f to solve s.t. $\dot{\mathbf{x}} = f(\mathbf{x})$
  - the initial conditions $[\mathbf{x}_0]$
- Preliminaries:
  - Compute the p-th derivatives of f
  - Compute the $p+1$-th derivatives of ϕ
- The main steps are (in a loop):
  1. From $[\mathbf{x}_n]$ apply explicit Runge-Kutta methods ϕ to compute $[\hat{\mathbf{x}}_{n+1}]$
  2. Apply Picard-Lindelöf to compute $[\tilde{\mathbf{x}}]$
  3. If truncation error E below of a tolerance ε then goto 5
  4. Else reduce step-size $h/2$ and goto 1
  5. $[\mathbf{x}_{n+1}] = [\hat{\mathbf{x}}_{n+1}] + E$ and
  6. update step-size h in function to E and goto 1

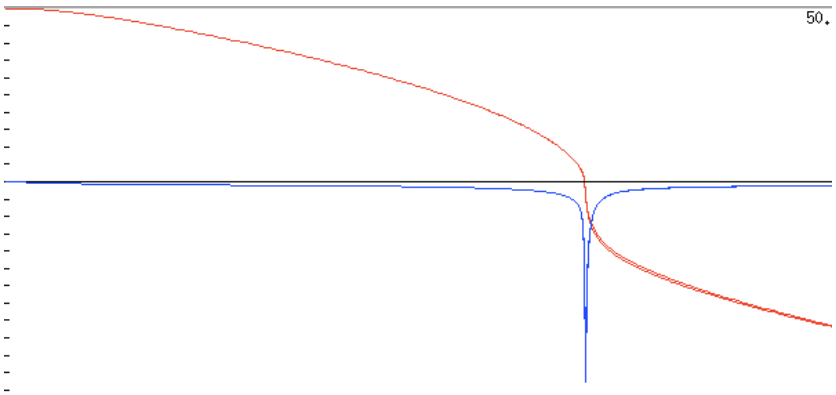
Remark: even fixed-step explicit Runge-Kutta methods are transformer into variable-step guaranteed methods.

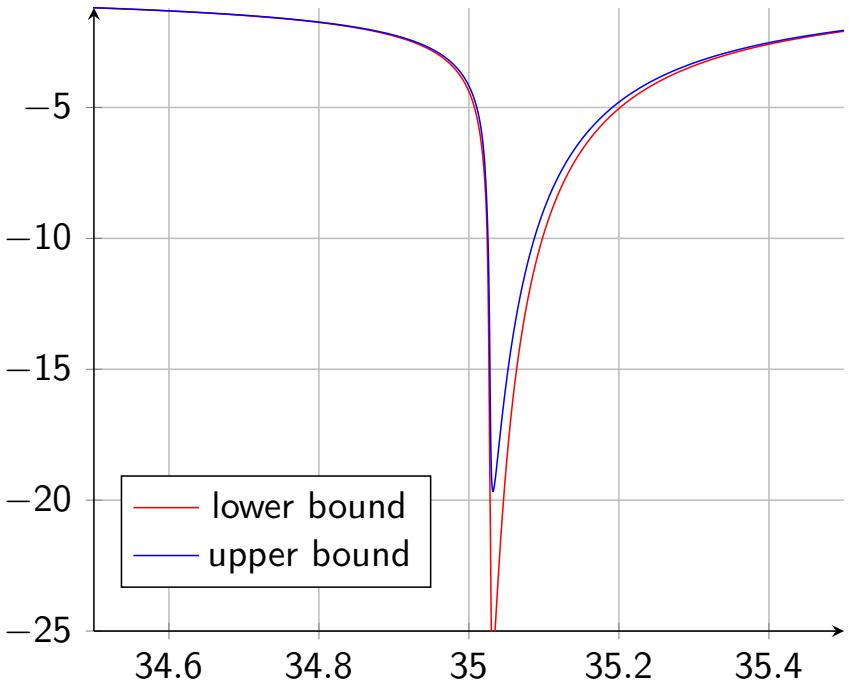
Example: a chemical reaction

Consider the IVP:

$$\begin{cases} \dot{y} = z \\ \dot{z} = z^2 - \dfrac{3}{0.001 + y^2} \end{cases} \quad \text{with} \quad \begin{cases} y(0) = 10 \\ z(0) = 0 \end{cases}$$

It admits a discontinuity in the solution around $t = 35$.

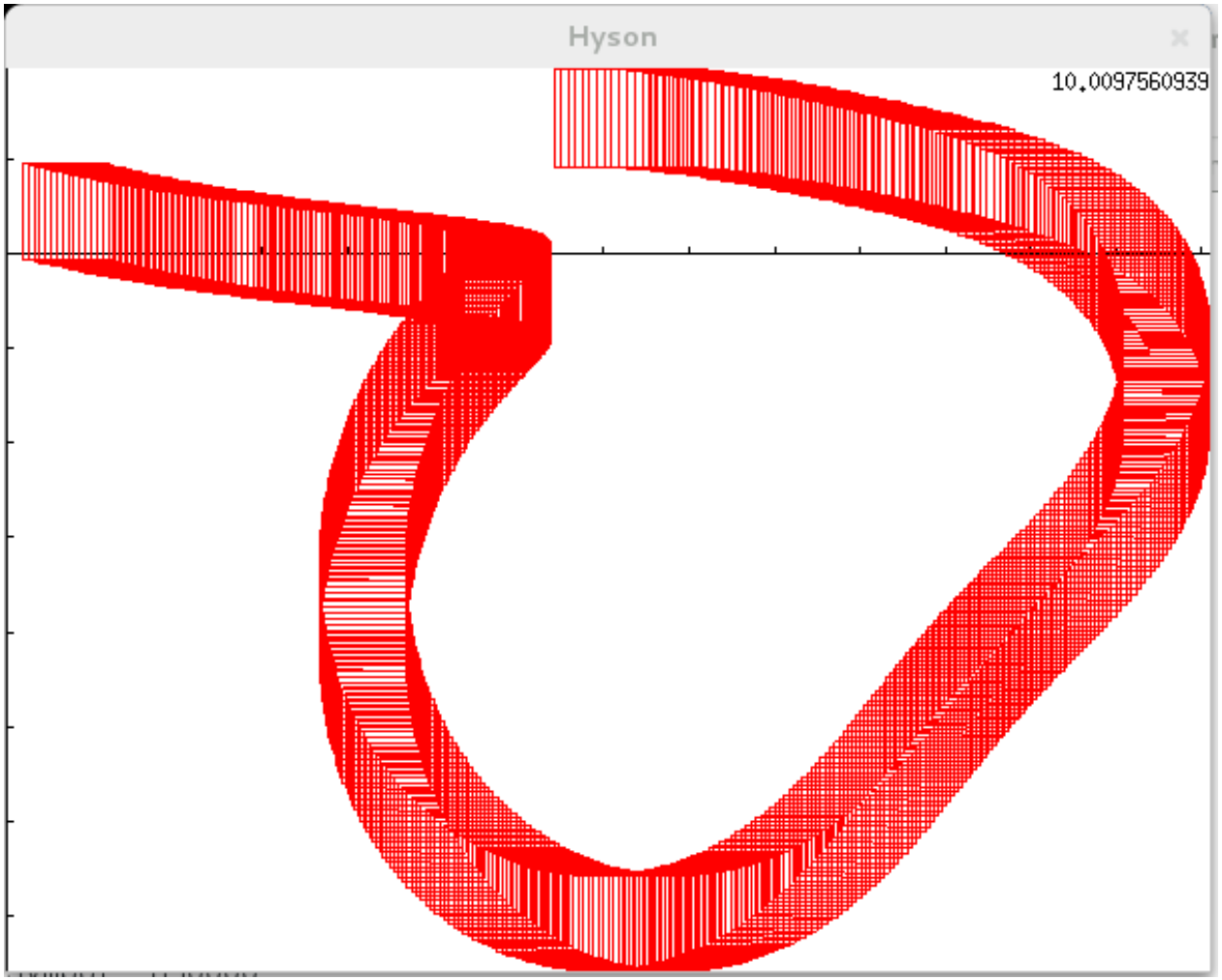




Example: sail boat

```
init x = [0,1];
init y = [0,1]; # initial position
init theta = 0;
init deltav = 0.5;
init deltag = 0;
init v = 2;
init omega = 0;

u1 = -0.5;
u2 = 0;

# Physical model
x' = v * cos(theta);
y' = v * sin(theta) - beta * vent;
theta' = omega;
deltav' = u1;
deltag' = u2;
v' = (fv * sin(deltav) - fg * sin(deltag) - alphaf * v) / m;
omega' = (fv * (long - rv * cos(deltav)) -
             rg * fg * cos(deltag) - alphatheta * omega) / j;

fv=alphav*vent*cos(theta+deltav)-alphav*v*sin(deltav);
fg=alphag*vitesse*sin(deltag);

output(x,y);
```

Example: sail boat



Conclusion

Conclusion

Guaranteed numerical solution of IVP based on:

- affine arithmetic
- well known numerical scheme: explicit Runge-Kutta methods

Future work

- Consider implicit Runge-Kutta methods.
- Consider multi-step methods as Adams-Bashworth, BDF, etc.
- Consider DAE e.g., $F(x, \dot{x}, y) = 0$.
- Consider delayed differential equations.