Reachability of Non-Linear Continuous Systems

Colin Chen

August 15, 2019

Reachability of Non-Linear Continuous Systems

Eric Goubault and Sylvie Putot (French Alternative Energies and Atomic Energy Commission (CEA)) *Forward Inner-Approximated Reachability of Non-Linear Continuous Systems* (HSCC 2017)

Problem

Problem specification:
a system of first order autonomous ODEs $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ where $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$, a time interval $[t_0, t_n]$ and initial conditions $\left[\boldsymbol{x}^{(0)}\right] \in \mathbb{IR}$.
We assume $\boldsymbol{f}$ is infinitely differentiable.

Definitions:
Suppose $\boldsymbol{x}(t, t_0, \boldsymbol{x}^{(0)})$ represents a solution to the ODE.
Fix $t \in [t_0, t_n]$.
$\mathbf{range}_t(\boldsymbol{x}, \left[\boldsymbol{x}^{(0)}\right]) = \boldsymbol{x}(t, t_0, \left[\boldsymbol{x}^{(0)}\right])$ is the set of all possible values $\boldsymbol{x}(t, t_0, \boldsymbol{x}^{(0)})$ at t given an initial conditions $\boldsymbol{x}^{(0)} \in \left[\boldsymbol{x}^{(0)}\right]$.

Problem

$\bigcup_{t \in [t_0, t_n]} \mathbf{range}_t(\boldsymbol{x}, \left[\boldsymbol{x}^{(0)}\right])$ is our <u>reachable state</u> or <u>flowpipe</u>.

At each slice of time...

- Outer approximations $[\boldsymbol{x}]$:
  - $\mathbf{range}_t(\boldsymbol{x}, \left[\boldsymbol{x}^{(0)}\right]) \subseteq [\boldsymbol{x}](t, t_0, x_0)$
- Inner approximations $]\boldsymbol{x}[$:
  - $]\boldsymbol{x}[(t, t_0, x_0) \subseteq \mathbf{range}_t(\boldsymbol{x}, \left[\boldsymbol{x}^{(0)}\right])$

Problem

Goal: Find tight inner and outer approximations of reachable states for all initial conditions in $\left[\boldsymbol{x}^{(0)}\right]$ over time interval $[t_0, t_n]$.

Outcome

Results can be used to solve other kinds of problems.

Dynamical systems with inputs $t \in \mathbb{R}_0^+, u \in \{\phi : \mathbb{R}_0^+ \to \mathbb{R}^m\}$:

$$\dot{x}(t) = \begin{cases} f(x(t), u(t)) & \text{if } t \geq 0 \\ z_0 & \text{if } t = 0 \end{cases}$$

Delayed Differential Equations with inputs $t \in \mathbb{R}_0^+$
and fixed parameters $\beta \in \mathbb{R}^m$

$$\begin{cases} \dot{x}(t) = f(x(t), x(t-\tau), \beta) & \text{if } t \geq t_0 + \tau \\ x(t) = x_0(t, \beta) & \text{if } t \in [t_0, t_0 + \tau] \end{cases}$$

Solution

What I made:

1. Flowpipes.jl
2. AffineArithmetic.jl (better intervals)
3. ModalIntervalArithmetic.jl

Bonus: hacked ForwardDiff.jl

Based on:
RINO `https://github.com/cosynus-lix/RINO`
aaflib `http://aaflib.sourceforge.net`
Miguel A. Sainz et al. - *Modal Interval Analysis: New Tools for Numerical Information* (Lecture Notes in Mathematics, Springer 2014)

Solution: AffineArithmetic.jl

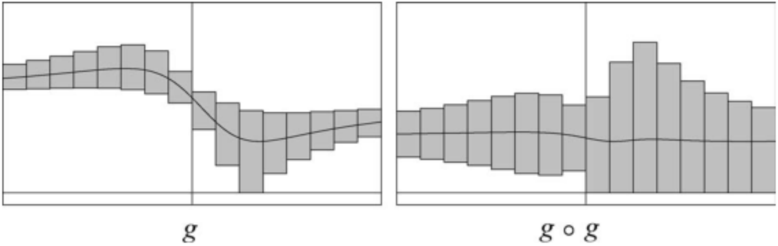
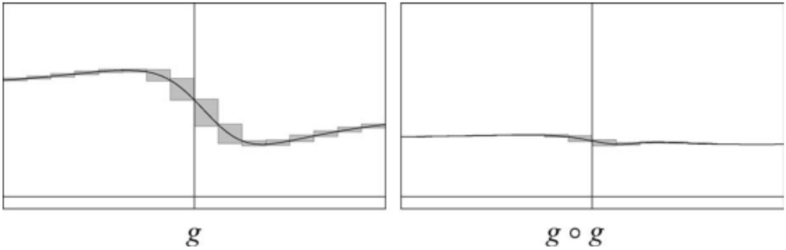


Figure 1. Catastrophic interval overestimation due to dependency.



Solution: Flowpipes.jl algorithm

Discretize $t_0 < t_1 < \cdots < t_j < t_{j+1} = t_j + \tau < \cdots < t_n$

Solution: Flowpipes.jl algorithm

Discretize $t_0 < t_1 < \cdots < t_j < t_{j+1} = t_j + \tau < \cdots < t_n$

For each $j = 0, \ldots, n$

1. get priori enclosure of
   - $\left[\boldsymbol{r}^{(j+1)}\right]$ of $\boldsymbol{x}(t, t_j, \left[\boldsymbol{x}^{(j)}\right])$
   - $\left[\tilde{\boldsymbol{r}}^{(j+1)}\right]$ of $\boldsymbol{x}(t, t_j, \left[\tilde{\boldsymbol{x}}^{(j)}\right])$
   - $\left[R^{(j+1)}\right]$ of $J(t, t_j, \left[\boldsymbol{x}^{(j)}\right]) = \mathrm{Jac}_{\boldsymbol{x}^{(j)}}\boldsymbol{x}(t, t_j, \left[\boldsymbol{x}^{(j)}\right])$

   over $t \in [t_j, t_{j+1}]$

2. get taylor approximations $[\boldsymbol{x}](t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right])$, $[\boldsymbol{x}](t_{j+1}, t_j, \left[\tilde{\boldsymbol{x}}^{(j)}\right])$ and $[J](t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right])$

3. deduce an inner approximation $]\boldsymbol{x}[(t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right])$

4. set $]\boldsymbol{x}^{(j+1)}[$, $\left[\boldsymbol{x}^{(j+1)}\right]$, $\left[\tilde{\boldsymbol{x}}^{(j+1)}\right]$ and $\left[J^{(j+1)}\right]$

Step 0: initialize data

Discretization $t_0 < t_1 < \cdots < t_j < t_{j+1} = t_j + \tau < \cdots < t_n$

Outer approximation $[\boldsymbol{x}^{(0)}]$ (given)

Outer approximation $[J^{(0)}] := I$

Outer approximate center of $[\boldsymbol{x}^{(0)}]$ as $[\tilde{\boldsymbol{x}}^{(0)}] := \mathrm{mid}\big[\boldsymbol{x}^{(0)}\big]$

Inner approximation $]\boldsymbol{x}^{(0)}[= [\boldsymbol{x}^{(0)}]$

Step 1: for each $j = 1, \ldots,$ priori enclosures

Get priori enclosure of

- $\left[\boldsymbol{r}^{(j+1)}\right]$ of $\quad \boldsymbol{x}(t, t_j, \left[\boldsymbol{x}^{(j)}\right])$
- $\left[\tilde{\boldsymbol{r}}^{(j+1)}\right]$ of $\quad \boldsymbol{x}(t, t_j, \left[\tilde{\boldsymbol{x}}^{(j)}\right])$
- $\left[R^{(j+1)}\right]$ of $\quad J(t, t_j, \left[\boldsymbol{x}^{(j)}\right]) = \mathrm{Jac}_{\boldsymbol{x}^{(j)}} \boldsymbol{x}(t, t_j, \left[\boldsymbol{x}^{(j)}\right])$

over $t \in [t_j, t_{j+1}]$

Step 1: for each $j = 1, \ldots,$ priori enclosures

Picard–Lindelöf Theorem

1. $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ has solution

$$\boldsymbol{x}(t) = \boldsymbol{x}^{(0)} + \int_{t_0}^{t} \boldsymbol{f}(\boldsymbol{x}(s))ds$$

2. The sequence

$$\phi_k(t, \boldsymbol{x}^{(0)}) = \begin{cases} \boldsymbol{x}^{(0)} & \text{if } k = 0 \\ \boldsymbol{x}^{(0)} + \int_{t_0}^{t} \boldsymbol{f}(\phi_{k-1}(s, \boldsymbol{x}^{(0)}))ds & \text{if } k > 0 \end{cases}$$

$\boldsymbol{x}(t, t_0, \boldsymbol{x}^{(0)}) = \lim_{k \to \infty} \phi_k(t, \boldsymbol{x}^{(0)})$

Step 1: for each $j = 1, \ldots,$ priori enclosures

Picard–Lindelöf Theorem for interval boxes

$$\phi_k(t, [\boldsymbol{x}^{(j)}]) = \{\phi_k(t, \boldsymbol{x}^{(j)}) : \boldsymbol{x}^{(j)} \in [\boldsymbol{x}^{(j)}]\} \subseteq [\phi_k]([\boldsymbol{x}^{(j)}]) \text{ where}$$

$$[\phi_k]\left(\left[\boldsymbol{x}^{(j)}\right]\right) = \begin{cases} [\boldsymbol{x}^{(j)}] & \text{if } k = 0 \\ [\boldsymbol{x}^{(j)}] + [0, \tau][\boldsymbol{f}]\big([\phi_{k-1}]([\boldsymbol{x}^{(j)}])\big) & \text{if } k > 0 \end{cases}$$

So our enclosure for the solution is
$[\boldsymbol{r}^{(j+1)}] = \lim_{k\to\infty}[\phi_k]([\boldsymbol{x}^{(j)}])$

$$\{\boldsymbol{x}(t, t_j, [\boldsymbol{x}^{(j)}]) : t \in [t_j, t_{j+1}]\} \subseteq [\boldsymbol{r}^{(j+1)}]$$

Step 1: for each $j = 1, \ldots,$ priori enclosures

Picard–Lindelöf Theorem for jacobians

$J(t, t_0, \boldsymbol{x}^{(0)}) = \mathrm{Jac}_{\boldsymbol{x}^{(0)}}\ \boldsymbol{x}(t, t_0, \boldsymbol{x}^{(0)})$

1. $\dot{J} = \mathrm{Jac}_{\boldsymbol{x}} f(\boldsymbol{x}) J$ has the solution

$$J(t) = J^{(0)} + \int_{t_0}^{t} \mathrm{Jac}_{\boldsymbol{x}} f(\boldsymbol{x}(s)) J(s)\, ds$$

2. The sequence

$$\Phi_k(t, J^{(0)}) = \begin{cases} J^{(0)} & \text{if } k = 0 \\ J^{(0)} + \int_{t_0}^{t} \mathrm{Jac}_{\boldsymbol{x}} f(\boldsymbol{x}(s)) \Phi_{k-1}(s, J^{(0)})\, ds & \text{if } k > 0 \end{cases}$$

$\boldsymbol{x}(t, t_0, \boldsymbol{x}^{(0)}) = \lim_{k\to\infty} \phi_k(t, \boldsymbol{x}^{(0)})$

Step 1: for each $j = 1, \ldots,$ priori enclosures

Picard–Lindelöf Theorem for jacobian interval boxes

$\Phi_k(t, [J^{(j)}]) = \{\Phi_k(t, J^{(j)}) : J^{(j)} \in [J^{(j)}]\} \subseteq [\Phi_k]([J^{(j)}])$ where

we obtain $[\Omega] = [\boldsymbol{f}]([\boldsymbol{r}^{(j+1)}])$ and let

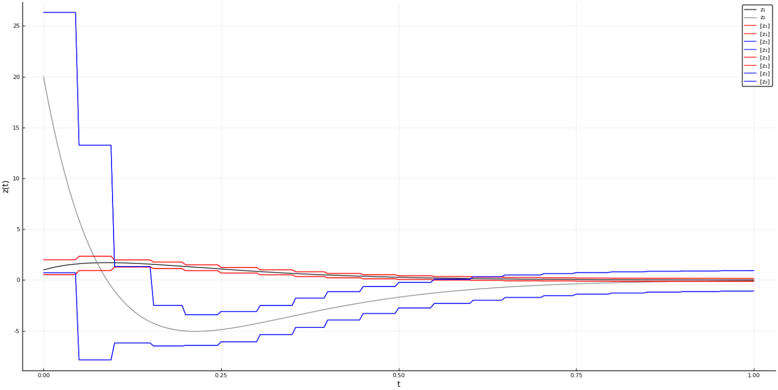
$$[\Phi_k]\left(\left[J^{(j)}\right]\right) = \begin{cases} [J^{(j)}] & \text{if } k = 0 \\ [J^{(j)}] + [0, \tau]\,[\Omega]\,[\Phi_{k-1}]([J^{(j)}]) & \text{if } k > 0 \end{cases}$$

So our enclosure is
$[R^{(j+1)}] = \lim_{k \to \infty} [\Phi_k]([J^{(j)}])$
$\{J(t, t_j, \boldsymbol{x}^{(j)}) : t \in [t_j, t_{j+1}]\} \subseteq [R^{(j+1)}]$

Step 1: for each $j = 1, \ldots,$ priori enclosures

example: damped oscillator

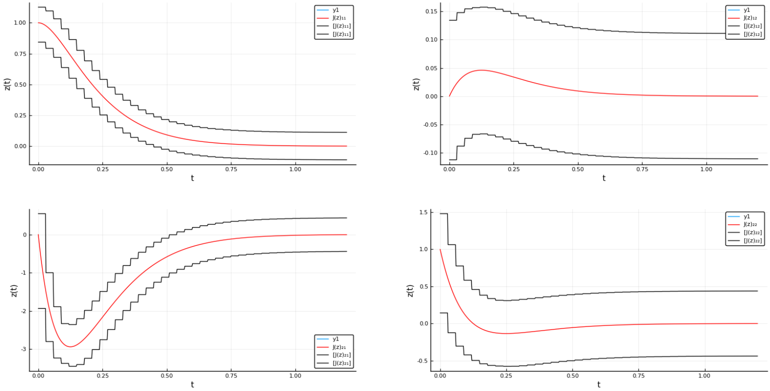
```
f(z::Vector) = [0 1; -64 -16] * z
z(t::Real) = exp(-8*t)*[1 + 28*t; 20 - 224*t]
```



Step 1: for each $j = 1, \ldots,$ priori enclosures

example: damped oscillator

```
f(z::Vector) = [0 1; -64 -16] * z
Jz(t::Real) = [exp(-8*t) + 8*t*exp(-8*t)   t*exp(-8*t);
               -64*t*exp(-8*t)   exp(-8*t) - 8*t*exp(-8*t)]
```



Step 2: for each $j = 1, \ldots,$ taylor approximations

Get taylor outer approximations $[\boldsymbol{x}](t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right])$, $[\boldsymbol{x}](t_{j+1}, t_j, \left[\tilde{\boldsymbol{x}}^{(j)}\right])$ and $[J](t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right])$

Step 2: for each $j = 1, \ldots,$ taylor approximations

Automatic generation of taylor coefficients

$$J(\boldsymbol{f}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{1}$$

$$\boldsymbol{f}^{[1]} = \boldsymbol{f}$$
$$\boldsymbol{f}^{[i+1]} = J(\boldsymbol{f}^{[i]}) \cdot \boldsymbol{f}$$

$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ has the solution for some $\zeta \in [t_0, t]$

$$\boldsymbol{x}(t, t_0, \boldsymbol{x}^{(0)}) = \boldsymbol{x}^{(0)} + \sum_{i=1}^{k-1} \frac{(t - t_0)^i}{i!} \boldsymbol{f}^{[i]}(\boldsymbol{x}^{(0)}) + \frac{(t - t_0)^k}{k!} \boldsymbol{f}^{[k]}(\boldsymbol{x}(\zeta))$$

Step 2: for each $j = 1, \ldots,$ taylor approximations

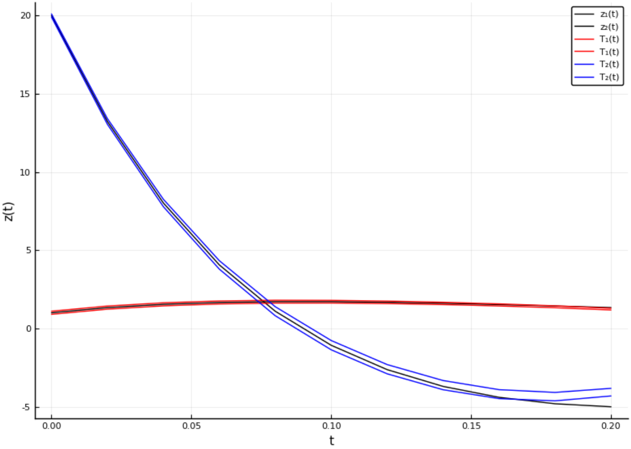
$$[\boldsymbol{x}](t, t_j, \left[\boldsymbol{x}^{(j)}\right]) = \left[\boldsymbol{x}^{(j)}\right] + \sum_{i=1}^{k-1} \frac{(t - t_j)^i}{i!} \boldsymbol{f}^{[i]}(\left[\boldsymbol{x}^{(j)}\right]) + \frac{(t - t_j)^k}{k!} \boldsymbol{f}^{[k]}(\left[\boldsymbol{r}^{(j+1)}\right])$$

$$[J](t, t_j, \left[J^{(j)}\right]) = \left[J^{(j)}\right] + \sum_{i=1}^{k-1} \frac{(t - t_j)^i}{i!} \mathrm{Jac}_{\boldsymbol{x}} \boldsymbol{f}^{[i]}(\left[\boldsymbol{x}^{(j)}\right]) \left[J^{(j)}\right] + \frac{(t - t_j)^k}{k!} \mathrm{Jac}_{\boldsymbol{x}} \boldsymbol{f}^{[k]}(\left[\boldsymbol{r}^{(j+1)}\right]) \left[R^{(j+1)}\right]$$

Step 2: for each $j = 1, \ldots,$ taylor approximations

example: damped oscillator

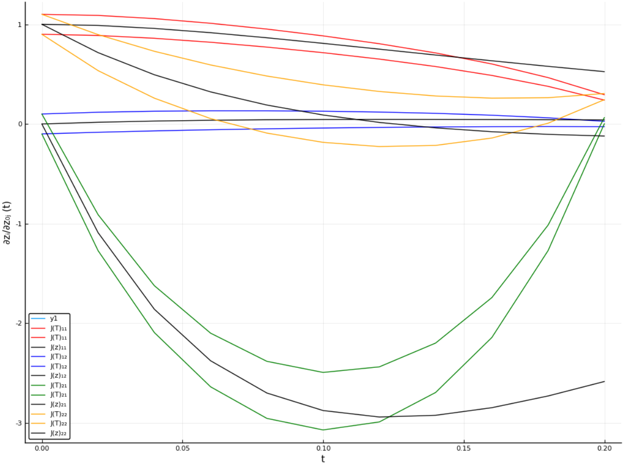
```
f(z::Vector) = [0 1; -64 -16] * z
z(t::Real) = exp(-8*t)*[1 + 28*t; 20 - 224*t]
```



Step 1: for each $j = 1, \ldots,$ priori enclosures

example: damped oscillator

```
f(z::Vector) = [0 1; -64 -16] * z
Jz(t::Real) = [exp(-8*t) + 8*t*exp(-8*t)   t*exp(-8*t);
               -64*t*exp(-8*t)   exp(-8*t) - 8*t*exp(-8*t)]
```



Step 3: for each $j = 1, \ldots$, inner approximation

Deduce an inner approximation $]\boldsymbol{x}[(t_{j+1}, t_j)$

Step 3: for each $j = 1, \ldots,$ inner approximation

Mean-value AE-extension

Let $f : \mathbb{R}^n \to \mathbb{R}$, $[\boldsymbol{x}] \in \mathbb{IR}^n$, $\tilde{x} \in [\boldsymbol{x}]$
and $\boldsymbol{\Delta} \in \mathbb{IR}^n$ such that $\left\{ \frac{\partial f}{\partial x_i}(x) : x \in [\boldsymbol{x}] \right\} \subseteq \Delta_i$.

If $]\boldsymbol{z}[= f(\tilde{x}) + \boldsymbol{\Delta}(\text{dual } [\boldsymbol{x}] - \tilde{x})$ is an improper integral vector, then

$(\forall z \in]z[)(\exists \boldsymbol{x} \in [\boldsymbol{x}])(f(\boldsymbol{x}) = z)$ making dual $]z[$ an inner approximation

Step 3: for each $j = 1, \ldots$, inner approximation

Define:

$$]\boldsymbol{x}[(t, t_j) := [\boldsymbol{x}](t, t_j, [\tilde{\boldsymbol{x}}^{(j)}]) + [\boldsymbol{J}](t, t_j, [\boldsymbol{x}^{(j)}])(\text{dual}\,[\boldsymbol{x}^{(0)}] - \tilde{\boldsymbol{x}}^{(0)})$$

If it is improper for given t, then dual $]\boldsymbol{x}[(t, t_j)$ is an inner approximation.

Step 4: for each $j = 1, \ldots,$ set up next step

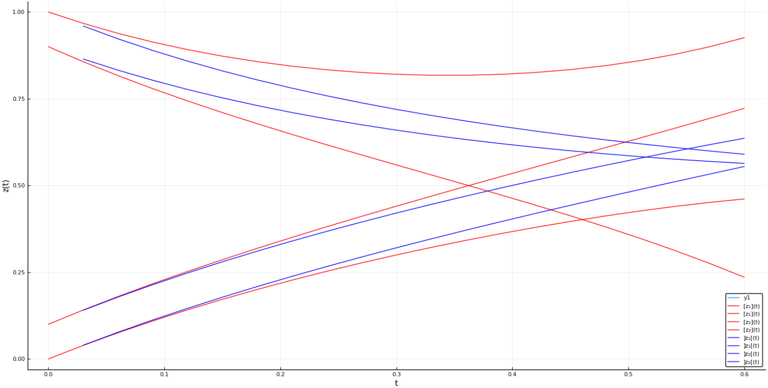
Set $]\boldsymbol{x}^{(j+1)}[$, $\left[\boldsymbol{x}^{(j+1)}\right]$, $\left[\tilde{\boldsymbol{x}}^{(j+1)}\right]$ and $\left[J^{(j+1)}\right]$

$$
\begin{aligned}
]\boldsymbol{x}^{(j+1)}[&=]\boldsymbol{x}[(t_{j+1}, t_j) \\
\left[\boldsymbol{x}^{(j+1)}\right] &= [\boldsymbol{x}](t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right]) \\
\left[\tilde{\boldsymbol{x}}^{(j+1)}\right] &= [\boldsymbol{x}](t_{j+1}, t_j, \left[\tilde{\boldsymbol{x}}^{(j)}\right]) \\
\left[J^{(j+1)}\right] &= [J](t_{j+1}, t_j, \left[\boldsymbol{x}^{(j)}\right])
\end{aligned}
$$

Solution: Flowpipes.jl algorithm

example: Brusselator

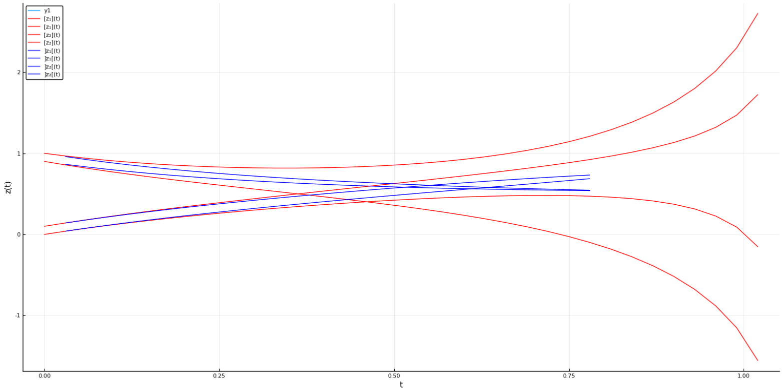
```
f(z::Vector) = [1.0 - 2.5*z[1] + z[2]*z[1]^2;
    1.5*z[1] - z[2]*z[1]^2]
```



Solution: Flowpipes.jl algorithm

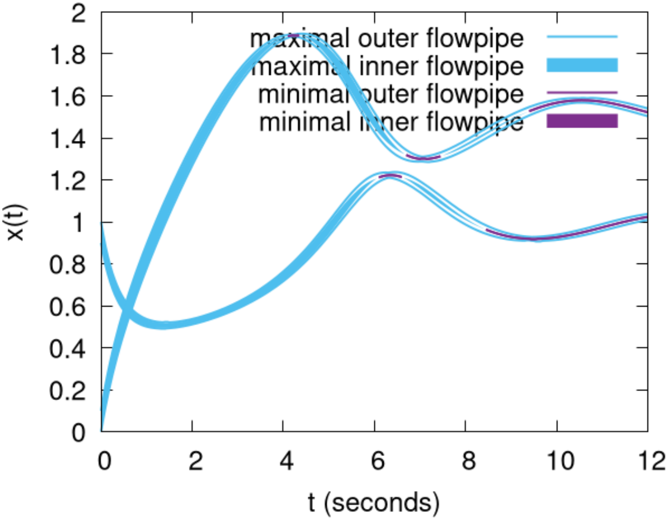
example: Brusselator

```
f(z::Vector) = [1.0 - 2.5*z[1] + z[2]*z[1]^2;
    1.5*z[1] - z[2]*z[1]^2]
```



Solution: Flowpipes.jl algorithm

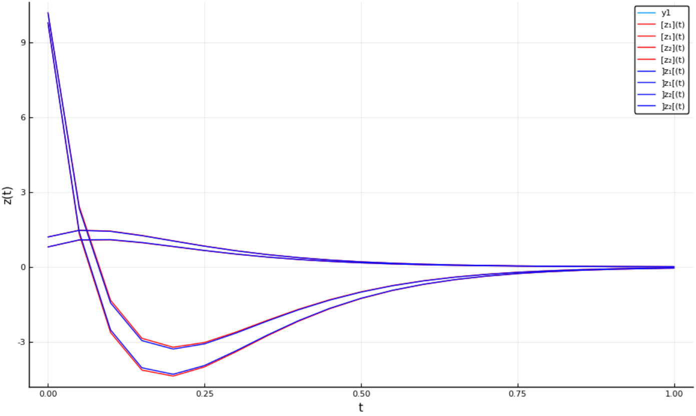
example: Brusselator



Solution: Flowpipes.jl algorithm

example: damped oscillator

```
f(z::Vector) = [0 1; -64 -16] * z
z(t::Real) = exp(-8*t)*[1 + 28*t; 20 - 224*t]
```



Solution: Flowpipes.jl

1. Flowpipes.jl
2. AffineArithmetic.jl (better intervals)
3. ModalIntervalArithmetic.jl
4. hacked ForwardDiff.jl

Things to do: do away with `ForwardDiff.jacobian(f, x)`, Zygote.jl, ???

`https://github.com/fireofearth/2019s-verification`