Differential inclusion using Matrix Exponentiation

Damien Massé
joint work with Luc Jaulin

LabSTICC - Robex
Université de Bretagne Occidentale
Brest, France

AID/FARO, nov. 2022

Outline

1. From differential inclusion to matrix exponentiation.
2. Extending scaling and squaring.
3. Abstract domain for differential inclusion contractor.
4. Examples.

The differential inclusion problem

Let's consider the differential inclusion problem on $\mathbb{R}^n$:

$$\dot{x} = f(x, u)$$

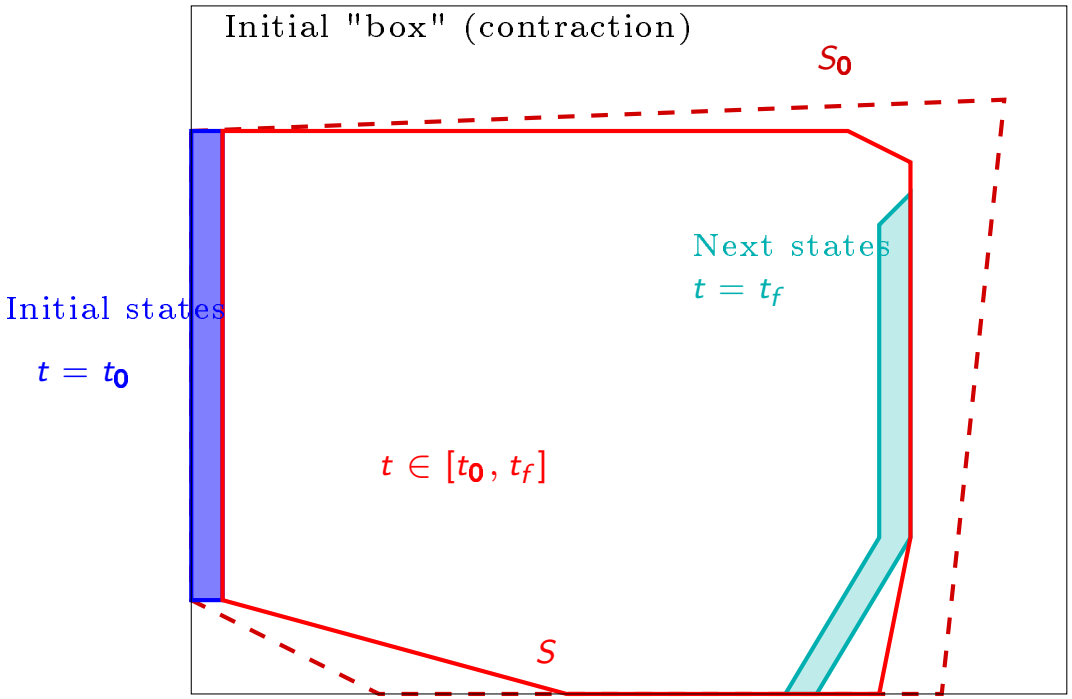
- f is in a set F of differentiable functions
- u can take any value in a tube $[u](t)$ at any time

We want to compute, from a set of initial states, an overapproximation of the possible states at time t, *inside an existing set* (contractor).

Approach

Classical approach:

- assume the states stay in a set S_0 to bound f between (t_0, t_f);
- deduce from the bounding of f an overapproximation S of the possible states;
- check that $S \subseteq S_0$ to prove the validity of the assumption.
- compute (more precisely) the set at time t_f.



First-order approximation of f

We consider an first-order approximation of f on S_0:

$$f(x,u) = C + A(x - x_m) + \psi(x,u)$$

- $x_m \in S_0$, C the center of $F(x_m,[u])$;
- A the center of $J_F(S_0,[u])$ defined as:

$$J_F(S_0,[u]) = \{J_f(S_0,u) \mid f \in F, u \in [u]\}$$

- $\psi(x,u)$ takes any value in a (zero-centered) box $[\Phi]$.

Solution of the diffential inclusion

Assuming the states stay in S_0, the solution of the differential equation is, from (t_0, x_0) (*variation-of-constant* formula):

$$\begin{aligned} x(t) - x_m &= e^{(t-t_0)A} \cdot (x_0 - x_m) && \text{(1. transformation)} \\ &+ \int_{t_0}^{t} e^{(t-\tau)A} d\tau \cdot C && \text{(2. translation)} \\ &+ \int_{t_0}^{t} e^{(t-\tau)A} \psi(x, u(\tau)) d\tau && \text{(3. other)} \end{aligned}$$

Approximations needed:

$$\begin{aligned} &\text{1st term} && e^{\delta t A} = \sum_0^\infty \frac{(\delta t A)^n}{n!} \\ &\text{2nd term} && \int_0^{\delta t} e^{\tau A} d\tau = \delta t \sum_0^\infty \frac{(\delta t A)^n}{(n+1)!} \\ &\text{3rd term} && \int_0^{\delta t} |e^{\tau A}| d\tau \qquad (\ \cdot [\Psi]) \end{aligned}$$

Matrix exponentiation

Rewriting the goal (first and second term):

1. for final states, approximations of:

$$\varphi_0^*(A) = e^A - \mathrm{Id} = \sum_{k=1}^{\infty} \frac{A^k}{k!}$$

$$\varphi_1^*(A) = \int_0^1 e^{\tau A} - \mathrm{Id}\, d\tau = \sum_{k=1}^{\infty} \frac{A^k}{(k+1)!}$$

   ("narrow" approximations)

2. for intermediate states:

$$\Phi_0^*(A) = \{\varphi_0^*(\alpha A) | \alpha \in [0,1]\} \text{ and } \Phi_1^*(A) = \{\varphi_1^*(\alpha A) | \alpha \in [0,1]\}$$

   ("wide" approximations)

φ_0^* and φ_1^* are *exponential integrator functions* (here, without the constant term).

Exponential integrators

Exponential integrator functions φ_k are defined as:

$$\begin{aligned}\varphi_0(M) &= e^M = \sum_{i=0}^{\infty} \frac{M^i}{i!} \\ \varphi_k(M) &= \int_0^1 \tau^{k-1} e^{(1-\tau)M} d\tau = \sum_{i=0}^{\infty} \frac{M^i}{(i+k)!}\end{aligned}$$

For our needs:

- we want a guaranteed interval approximation of $\varphi_k(M)$;
- we want a guaranteed (and precise) interval approximation of $\{\varphi_k(\tau M) \mid \tau \in [0,1]\}$ ("wide" sets).

Note: non-autonomous equations require the computation of ϕ_2^*.

Common approaches

Commonly used approaches:

- Taylor approximations, i.e.:

$$e^A \approx \sum_{n=0}^{n=N} \frac{A^n}{n!}$$

  with a (simple) error bound in $||A||_\infty^{N+1}$

- Pade approximants. Issues: requires to compute matrix inversions, and has a complex error bound.

Also, matrix exponentiation methods use *scaling-and-squaring*, i.e. the fact that:

$$e^{2A} = (e^A)^2$$

Example

$$A = \begin{pmatrix} 0 & \pi \\ -\pi & 0 \end{pmatrix}$$

$\xrightarrow{\text{Taylor approx, order 14}}$ e^A, $\text{err} \approx 2.3 \cdot 10^{-5}$

$\downarrow A' = \frac{A}{2^8}$

$e^{A'}$
Taylor order 6:
$\text{err} \approx 7 \cdot 10^{-18}$

$\xrightarrow{(e^{A'})^{2^8}}$ e^A (8 squarings)
$\text{err} \approx 1.5 \cdot 10^{-13}$

Extension of scaling and squaring

Theorem

Let's consider:

$$\varphi_N^*(A) = \sum_{k \geq 1} \frac{A^k}{(k+N)!}$$

Then

$$\varphi_N^*(2A) = \frac{1}{2^N} \left(\varphi_N^*(A)\varphi_0^*(A) + 2\varphi_N^*(A) + \sum_{i=0}^{N-1} \frac{1}{(N-i)!} \varphi_i^*(A) \right)$$

Applying this result:

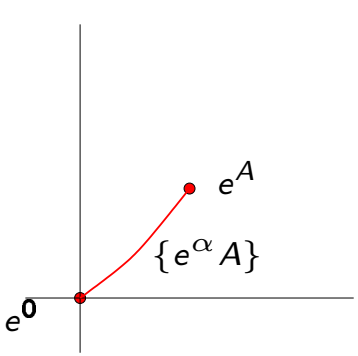
$$\begin{aligned} \varphi_0^*(2A) &= \varphi_0^*(A)(\varphi_0^*(A) + 2\mathrm{Id}) \\ \varphi_1^*(2A) &= \frac{1}{2}\varphi_1^*(A)\varphi_0^*(A) + \varphi_1^*(A) + \frac{1}{2}\varphi_0^*(A) \end{aligned}$$

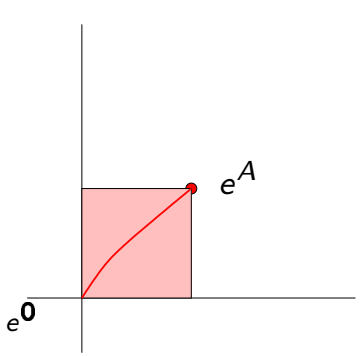
Intermediate states

To approximate $\Phi_0^*(A) = \{\varphi_0^*(\alpha A) \mid \alpha \in [0,1]\}$:

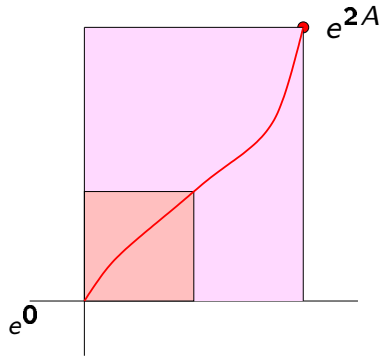
1. use a Taylor approximation (factorising α as much as possible);
2. naive application of *scaling and squaring* is too imprecise (by computing directly $\varphi_0^*(\alpha A)^2$). Instead:

$$\{e^{2\alpha A} \mid \alpha \in [0,1]\} = \{e^{\alpha A} \mid \alpha \in [0,1]\} \cup \{e^{A} e^{\alpha A} \mid \alpha \in [0,1]\}$$

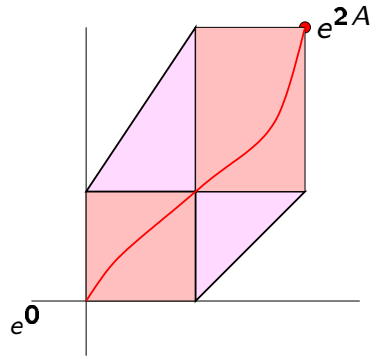




approximation of $\{e^\alpha A\}$



naive squaring



more precise squaring

Scaling and squaring

Theorem

Let's consider $\Phi^*_N(A) = \{\varphi^*_N(\alpha A) \mid \alpha \in [0,1]\}$. Then

$$\Phi^*_N(2A) = \Phi^*_N(A) \cup \left\{ \frac{\alpha^N}{(1+\alpha)^N} e^A \Phi^*_N(A) + \sum_{i=0}^{N} \frac{\alpha^{N-i}}{(1+\alpha)^N} \frac{\varphi^*_i(A)}{(N-i)!} \right\}$$

Application:

$$\Phi^*_0(2A) \subseteq \Phi^*_0(A)(A) \cup ([0,1]\,\Phi^*_0(A)(\varphi^*_0(A) + \mathrm{Id}) + \varphi^*_0(A))$$

$$\Phi^*_1(2A) \subseteq \Phi^*_1(A) \cup (\varphi^*_1(A) + [0, \tfrac{1}{2}](\varphi^*_0(A) - \varphi^*_1(A) + \Phi^*_1(A)(\varphi^*_0 + \mathrm{Id})))$$

These formulas avoid sums or products of "wide" approximations, and limit the use of interval coefficients.

Alternative computation

Theorem

Let's consider $\Phi_N^\sharp(A) = \{\frac{1}{\alpha}\varphi_N^*(\alpha A) \mid \alpha \in [0,1]\}$. Then

$$\begin{aligned}\Phi_N^\sharp(2A) = 2\Phi_N^\sharp(A) \cup \\ \Big\{\varphi_N^*(A) + \tfrac{1}{(1+\alpha)^{N+1}}(\alpha^{N+1}(e^A\Phi_N^\sharp(A) - \varphi_N^*(A)) + \\ \sum_{i=1}^{N} \tfrac{\alpha^{N-i}}{(N-i)!}(\varphi_i^*(A) - \tfrac{(N+1)!}{(i+1)!}\varphi_N^*(A))\Big\}\end{aligned}$$

Application:

$$\Phi_0^\sharp(2A) \subseteq 2\Phi_0^\sharp(A)(A) \cup (2\varphi_0^*(A) + [0,1](\Phi_0^\sharp(A)(\varphi_0^*(A) + \mathrm{Id}) - \varphi_0^*(A)))$$

Computing $\Phi_0^\sharp(2A)$ instead of $\Phi_0^*(A)$ is useful later to bound the uncertainty term. Furthermore this formula is more precise for $N \geq 2$.

Example

	$\Phi_0^\sharp(A)$
"Exact" result	$\begin{pmatrix} [-2.28;0] & [0;\pi] \\ [-\pi;0] & [-2.28;0] \end{pmatrix}$
Taylor (14)	$\begin{pmatrix} [-4.93;4.06] & [-2.03;3.15] \\ [-3.15;2.03] & [-4.93;4.06] \end{pmatrix}$
Taylor(6) + squaring(8)	$\begin{pmatrix} [-2.57;3\cdot 10^{-7}] & [-0.06;\pi] \\ [-\pi;0.06] & [-2.57;3\cdot 10^{-7}] \end{pmatrix}$

	$\Phi_1^*(A)$
"Exact" result	$\begin{pmatrix} [-1;0] & [0;0.73] \\ [-0.73;0] & [-1;0] \end{pmatrix}$
Taylor (14)	$\begin{pmatrix} [-1.65;0.81] & [0;1.57] \\ [-1.57;0] & [-1.65;0.81] \end{pmatrix}$
Taylor(6) + squaring(8)	$\begin{pmatrix} [-1.02;2\cdot 10^{-10}] & [0;0.82] \\ [-0.82;0] & [-1.02;2\cdot 10^{-10}] \end{pmatrix}$

Uncertainty term (third term)

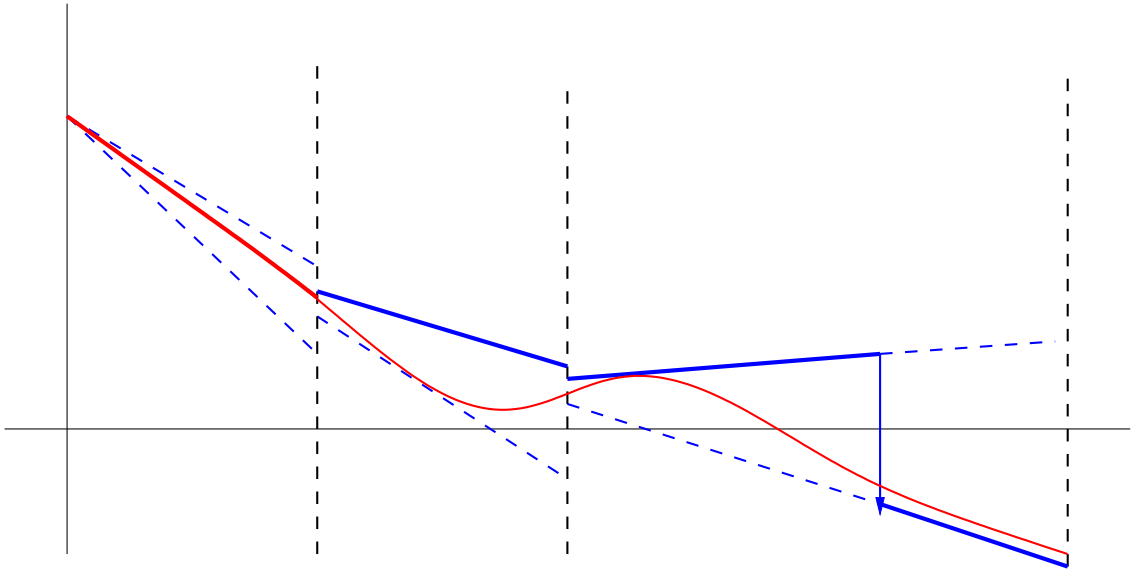
Bound for

$$\int_0^1 |e^{\tau A}| d\tau$$

as sums (during squaring) of

$$\int_0^{2^{-k}} |e^{2^{-k}A} e^{\tau A}| d\tau$$

Each part of $e^{2^{-k}A} e^{\tau A}$ is componentwise approximated by interval affine functions $I_1 + \tau I_2$.



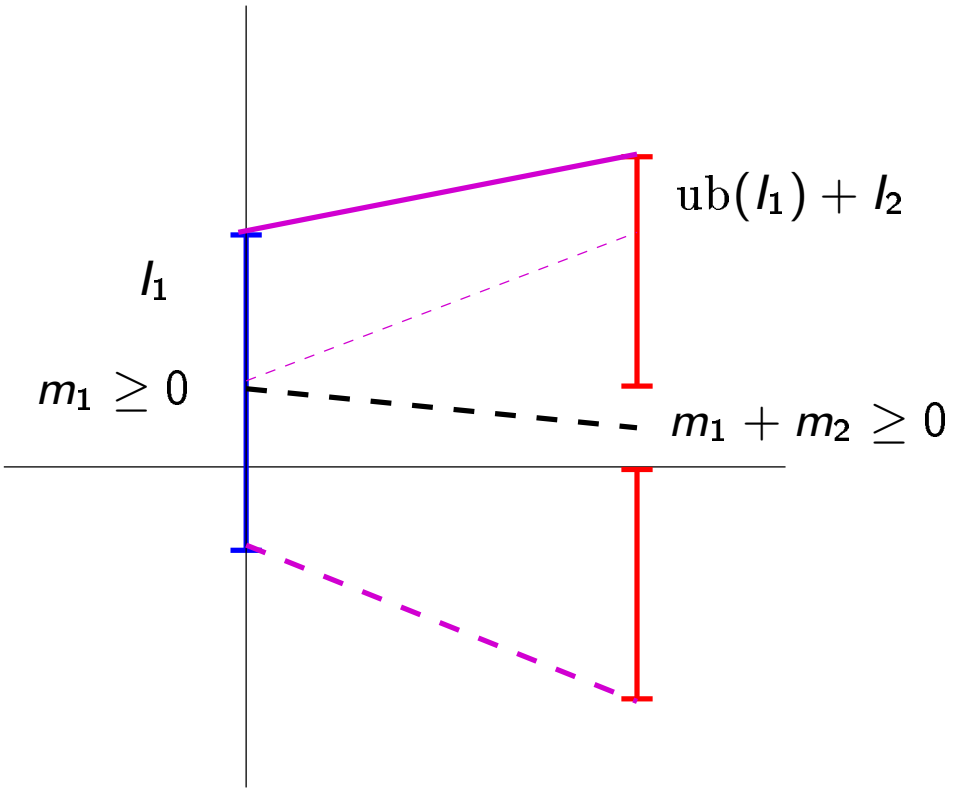
Componentwise computation

Componentwise, we search upper bounds for expressions of the form:

$$R = \int_0^1 |I_1 + \tau I_2| d\tau$$

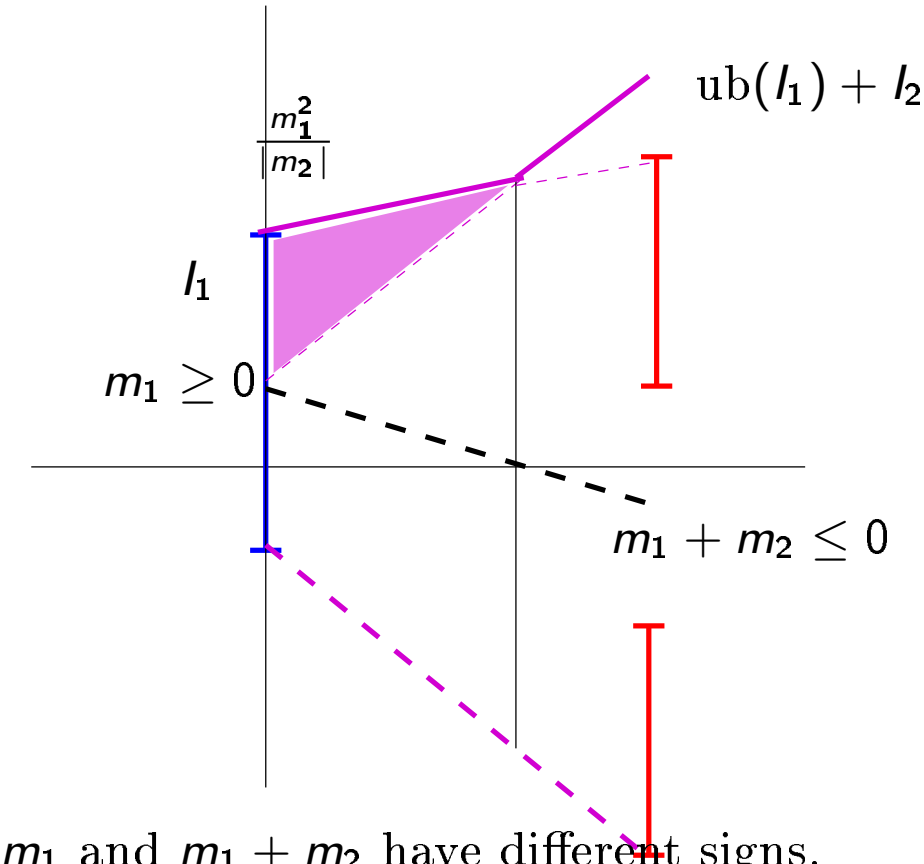
where I_1 and I_2 are both intervals.
The value of R is determined by m_1 and m_2, respective centers of I_1 and I_2:

m_1 and m_2 have		$R =$
the same sign	$m_1 \geq 0$	$\mathrm{ub}(I_1) + \mathrm{ub}(I_2)^2/2$
	$m_1 < 0$	- $\mathrm{lb}(I_1) - \mathrm{lb}(I_2)^2/2$
opposite signs	$m_2 > 0$	$\mathrm{ub}(I_1) + \mathrm{ub}(I_2)^2/2 + m_1^2/m_2$
	$m_2 < 0$	$-\mathrm{lb}(I_1) - \mathrm{lb}(I_2)^2/2 - m_1^2/m_2$



m_1 and $m_1 + m_2$ have the same sign.

$m_1 \geq 0 \;\Rightarrow\; R = \mathrm{ub}(I_1) + \frac{\mathrm{ub}(I_2)^2}{2}$



m_1 and $m_1 + m_2$ have different signs.

$m_2 \geq 0 \;\Rightarrow\; R = -\mathrm{lb}(I_1) - \frac{\mathrm{lb}(I_2)^2}{2} - \frac{m_1^2}{m_2}$

Representation of states

Once the different terms are approximated, state evolution from t_0 to t appears as:

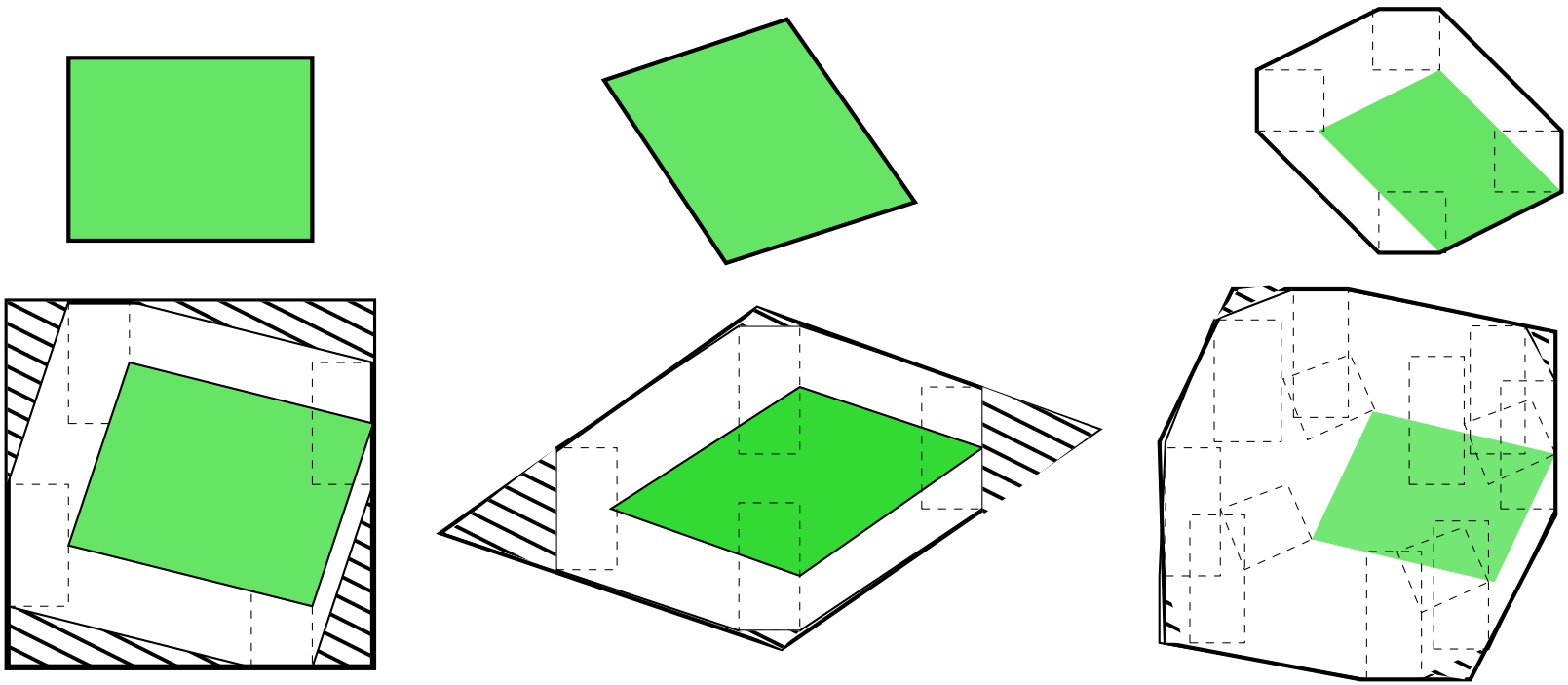
$$x(t) - x_m \in [M](x(t_0) - x_m) + [v]$$

where x_m is a vector, $[M]$ is a (small-diameter) matrix of intervals and $[v]$ is a vector of intervals.
Hence we need to consider an abstraction (state representations) $\langle\!|x|\!\rangle$ which efficiently handles the operation:

$$\langle\!|x'|\!\rangle = x_m + [M](\langle\!|x|\!\rangle - x_m) + [v]$$

Common abstractions

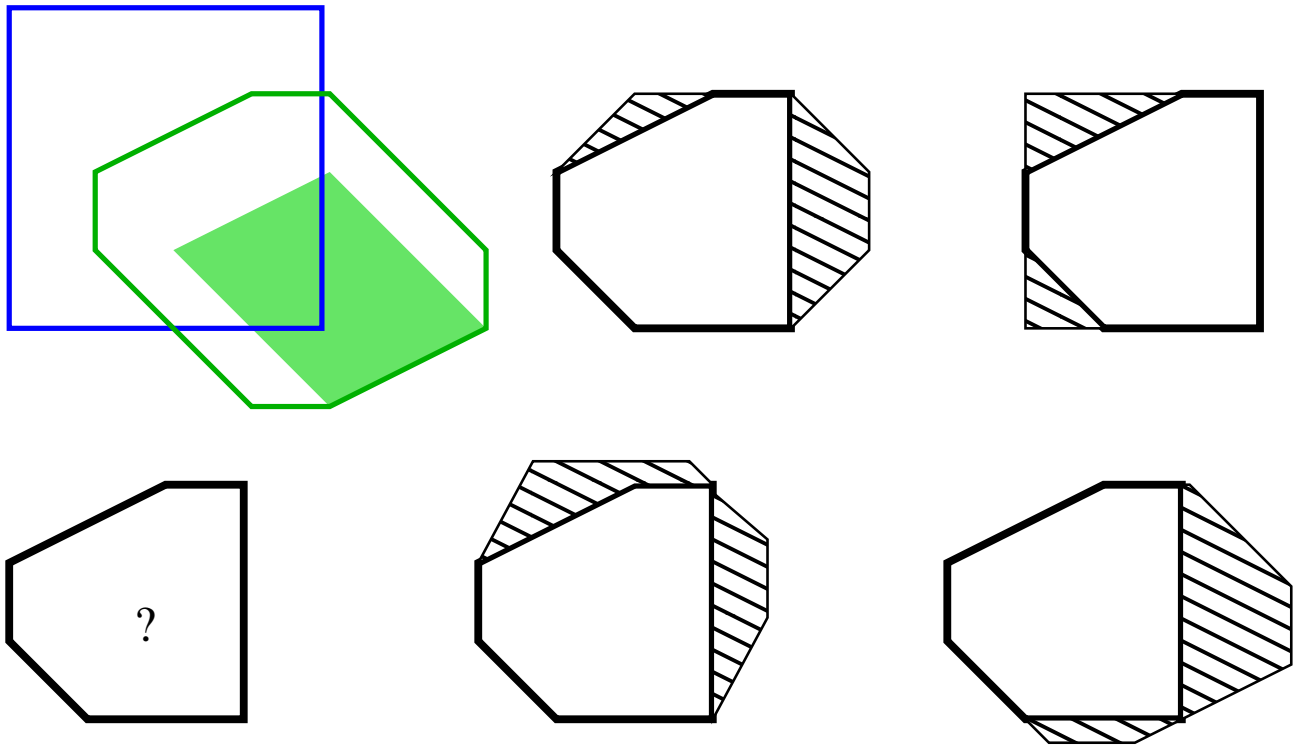
1. Boxes (vector of intervals): easy to use, not precise.
2. Parallelotopes $\{Mx \mid x \in [v]\}$: still lacks precision.
3. *Doubletons* (sums of a box and a parallelotope) or *zonotopes* (projections of higher-dimension parallelotopes) are more precise.



Boxes | Parallelotopes | Doubletons

Contractors

Doubletons, parallelotopes and zonotopes are not suited for intersection operations (needed for contractors).



$\longrightarrow$ we propose to use *finite intersections of parallelotopes*.

Intersections of parallelotopes

Abstract set $\langle\!|x|\!\rangle$ given by:

- a (small) number k of non-singular matrices $M_1, \ldots, M_k$;
- $k+1$ vectors of intervals $[v_0]$, $[v_1]$, ..., $[v_k]$.

Then $x \in \langle\!|x|\!\rangle$ iff:

$$x \in [v_0] \text{ and, for } 1 \leq i \leq k \ , x \in M_i \cdot [v_i]$$

$\langle\!|x|\!\rangle$ is a convex polyhedron. Its facets are defined by the matrices M_i^{-1}.
The set $\mathcal{M}$ of M_i is the $\mathcal{M}$-template of $\langle\!|x|\!\rangle$.

Properties

As a case of template polyhedral domain:

- when $\mathcal{M}$ is defined, given X, a *most precise* approximation of X exists in *compact form*;
- when $\langle\!|X|\!\rangle$ and $\langle\!|Y|\!\rangle$ shares the same template (in compact form), operations (unions, intersections, inclusion testing...) are fast and precise;
- any convex polyhedron can be represented as a $\langle\!|x|\!\rangle$ (if size of template not bounded).

Compared with the classical (constraint-based) polyhedral domain, non-singular matrices templates enable to design sound and (relatively) precise heuristics without using linear programming.

Issues

Let's consider the contractor $C_{\langle\!|y|\!\rangle} : \langle\!|x|\!\rangle \mapsto \langle\!|x|\!\rangle \cap \langle\!|y|\!\rangle$. If $\langle\!|x|\!\rangle$ and $\langle\!|y|\!\rangle$ do not share the same template:

- we may add new matrices in the template of $\langle\!|x|\!\rangle$, but but the final number of matrices may be unbounded;
- or we keep the template for $\langle\!|x|\!\rangle$ (if it's already large enough), but the result may be imprecise;
- of we change the template of $\langle\!|x|\!\rangle$ to achieve greater precision, but contractance may be lost.

Result: differential inclusion

$$(\dot{x};\dot{y}) = (y + [-10^{-4}, 10^{-4}]; (1 - x^2) * y - x + [-10^{-4}, 10^{-4}])$$

For $t = [0, 1]$, 128 slices.

Initial state	Our approach	CAPD (CW method)
(2;0)	$[1.507996, 1.508295]$ $\times[-0.780331, -0.780100]$	$[1.508005, 1.508283]$ $\times[-0.780311, -0.780126]$
(2;3)	$[2.300230, 2.300730]$ $\times[-0.479903, -0.479745]$	$[2.300371, 2.300625]$ $\times[-0.479863, -0.479778]$

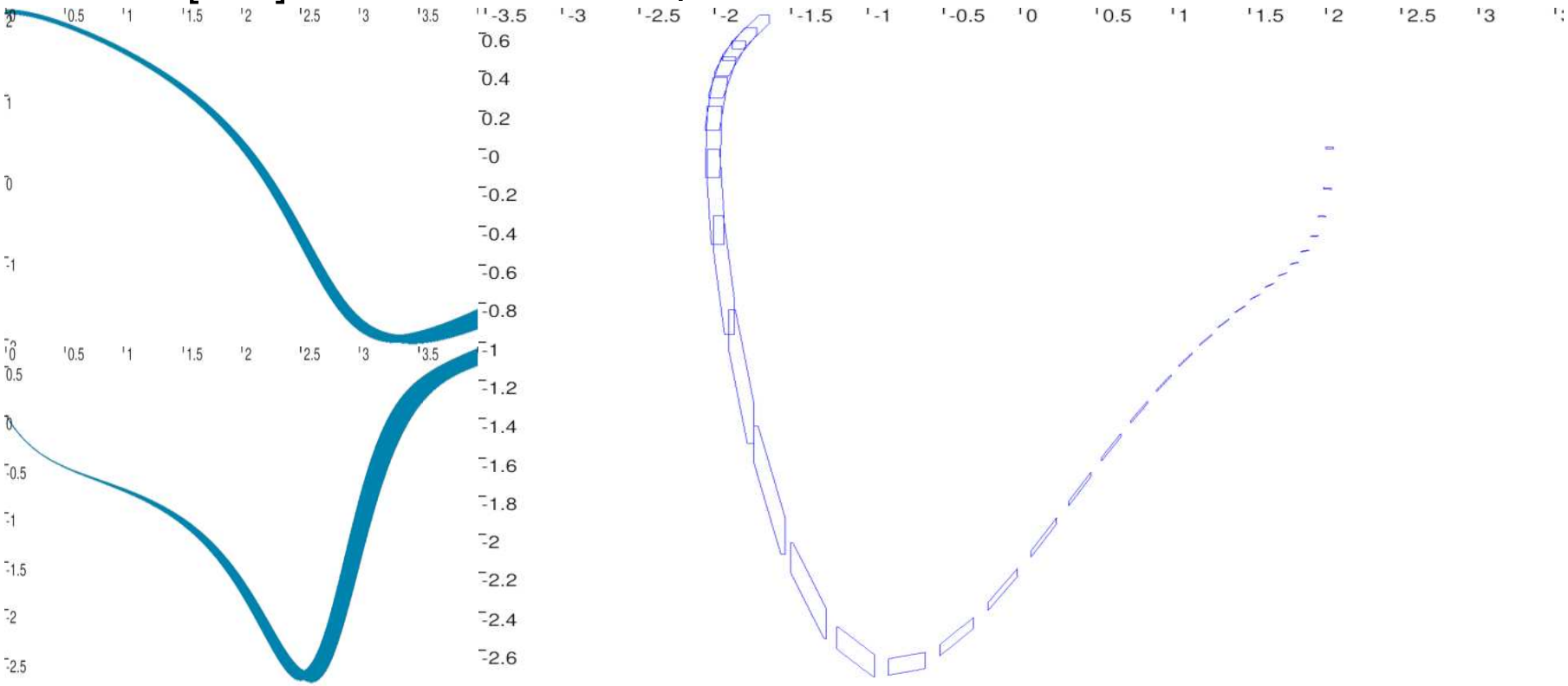
Increasing the number of slices eliminate the gap between both approaches.

DynInex gives similar results.

Result: longer time

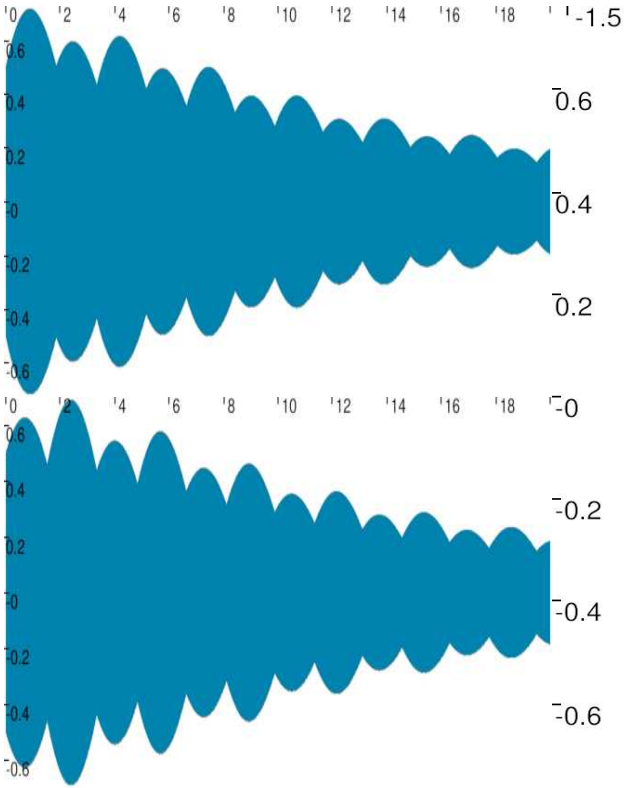
$$(\dot{x};\dot{y}) = (y;(1-x^2)*y-x) \quad \text{initial: } [2;2.05]\times[0;0.01]$$

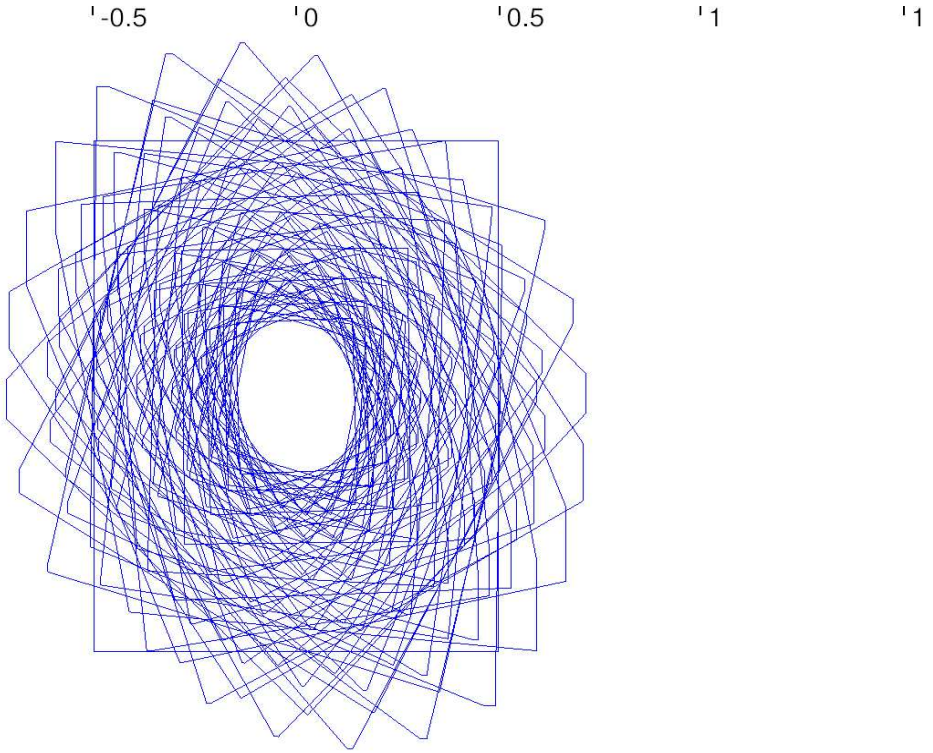
For $t=[0;4]$, 2048 slices, computation time: <1s.



Large initial states

$$(\dot{x};\dot{y}) = (y; -0.3y - sin(x) + [-0.02, 0.02]) \quad \text{initial: } [-0.5; 0.5] \times [-0.5; 0.5]$$

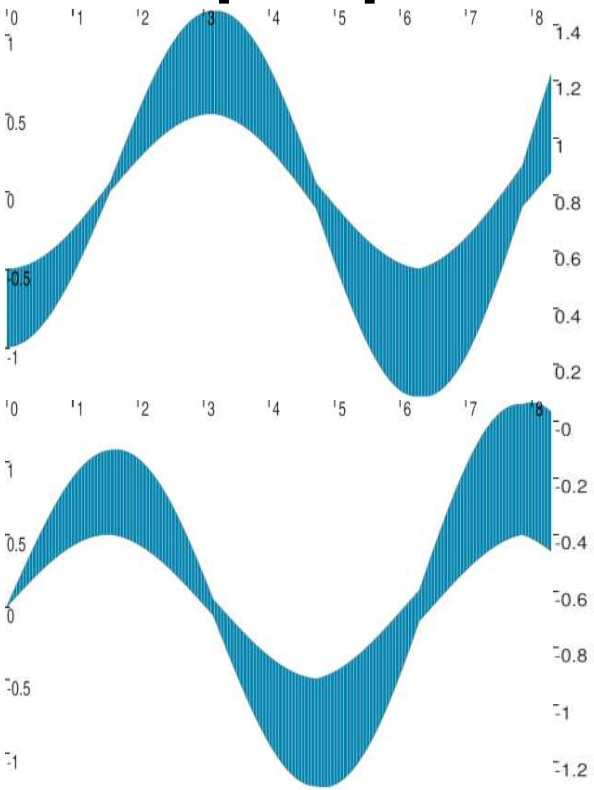


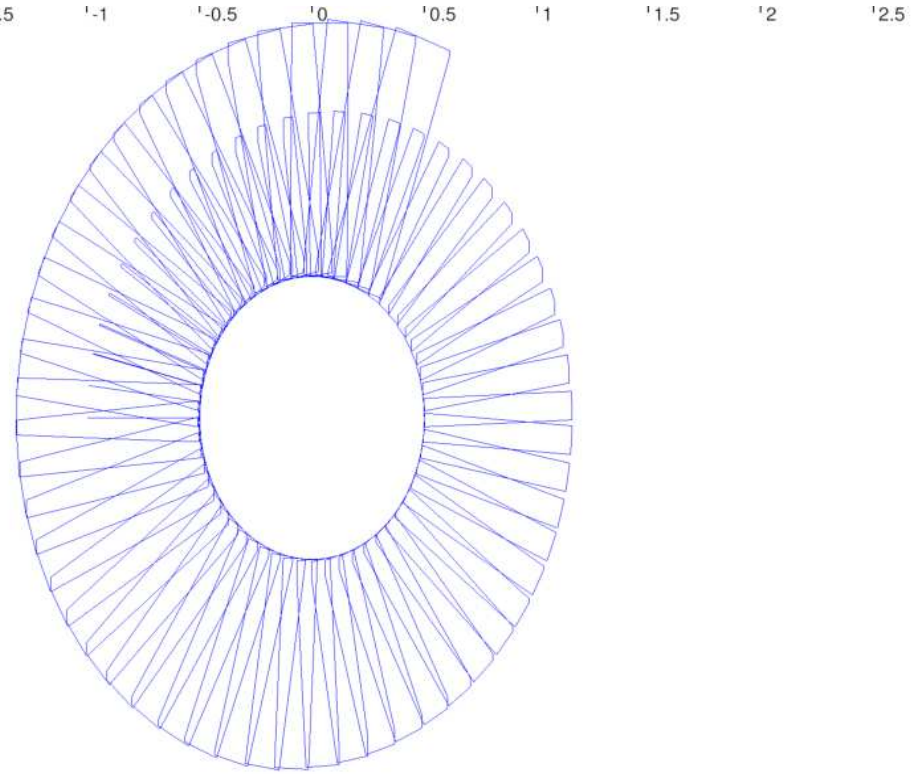


Non-autonomous equation (resonant forcing)

$$(\dot{x}; \dot{y}) = (y; -x + [0.0, 0.1] * sin(t)) \quad \text{initial: } [-1.0; -0.5] \times [0.0; 0.0]$$

For $t = [0; 8.3]$, 1024 slices, computation time: 2s.





Comparisons

With CAPD and DynIbex, not easy: several approaches, different ways to express uncertainties, different parameters to customize computation time and precision.
All tools diverge on some cases (e.g. Van Der Pol oscillator with non-ponctual box and long time).
Preliminary experiments seem to indicate that our approach may be more precise with large initial sets and perturbations (but may be worse worse on other problems).

Issue: confidence on the safety of (more precise) results (esp. when one tool is "much better" than the others).

Conclusions, future work

1. Design of exponential-based differential inclusion contractor.
2. Construction of a specific abstract domain (with a margin of improvement).
3. Integration in Codac (almost) done.

Directions for future work:

1. Different (more general) exponential integrations?
2. Tuning and test on "real" examples.
3. Integration with mazes.