Eulerian and Lagrangian Approaches for Filtering and Smoothing

L. Jaulin, T. Le Mézo, S. Rohou, F. Le Bars, B. Zerr
ENSTA Bretagne, LabSTICC
9 octobre 2017 à Nancy

Lagrangian approach

Contractors

The operator $\mathscr{C} : \mathbb{IR}^n \to \mathbb{IR}^n$ is a *contractor* [4] for the equation $f(\mathbf{x}) = 0$, if

$$\begin{cases} \mathscr{C}([\mathbf{x}]) \subset [\mathbf{x}] & \text{(contractance)} \\ \mathbf{x} \in [\mathbf{x}] \text{ and } f(\mathbf{x}) = 0 \Rightarrow \mathbf{x} \in \mathscr{C}([\mathbf{x}]) & \text{(consistence)} \end{cases}$$

Building contractors

Consider the primitive equation

$$x_1 + x_2 = x_3$$

with $x_1 \in [x_1]$, $x_2 \in [x_2]$, $x_3 \in [x_3]$.

We have

$$
\begin{aligned}
x_3 = x_1 + x_2 \Rightarrow \quad x_3 \in & \quad [x_3] \cap ([x_1] + [x_2]) \\
x_1 = x_3 - x_2 \Rightarrow \quad x_1 \in & \quad [x_1] \cap ([x_3] - [x_2]) \\
x_2 = x_3 - x_1 \Rightarrow \quad x_2 \in & \quad [x_2] \cap ([x_3] - [x_1])
\end{aligned}
$$

The contractor associated with $x_1 + x_2 = x_3$ is thus

$$\mathscr{C}\begin{pmatrix} [x_1] \\ [x_2] \\ [x_3] \end{pmatrix} = \begin{pmatrix} [x_1] \cap ([x_3] - [x_2]) \\ [x_2] \cap ([x_3] - [x_1]) \\ [x_3] \cap ([x_1] + [x_2]) \end{pmatrix}$$

Tubes

A trajectory is a function $\mathbf{f} : \mathbb{R} \to \mathbb{R}^n$. For instance

$$\mathbf{f}(t) = \begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$$

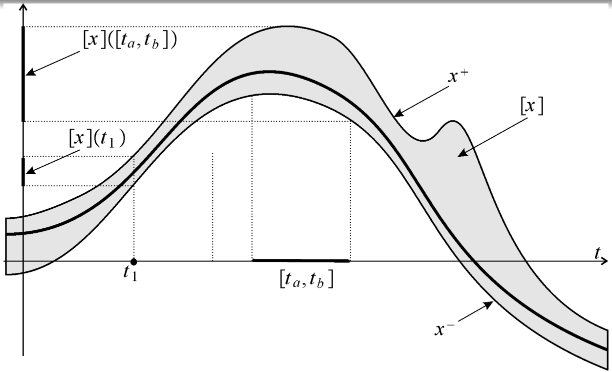
is a trajectory.

Order relation

$$\mathbf{f} \leq \mathbf{g} \Leftrightarrow \forall t, \forall i, f_i(t) \leq g_i(t).$$

We have

$$\begin{aligned}\mathbf{h} &= \mathbf{f} \wedge \mathbf{g} \Leftrightarrow \forall t, \forall i, h_i(t) = \min(f_i(t), g_i(t)),\\ \mathbf{h} &= \mathbf{f} \vee \mathbf{g} \Leftrightarrow \forall t, \forall i, h_i(t) = \max(f_i(t), g_i(t)).\end{aligned}$$



The set of trajectories is a lattice. Interval of trajectories (tubes) can be defined.

Example.

$$[\mathbf{f}](t)=\begin{pmatrix} \cos t+\left[0,t^2\right] \\ \sin t+[-1,1] \end{pmatrix}$$

is an interval trajectory (or tube).

Tube arithmetics

If $[x]$ and $[y]$ are two scalar tubes [1], we have

$$
\begin{array}{ll}
[z]=[x]+[y]\Rightarrow[z](t)=[x](t)+[y](t) & \text{(sum)}\\
[z]=\text{shift}_a([x])\Rightarrow[z](t)=[x](t+a) & \text{(shift)}\\
[z]=[x]\circ[y]\Rightarrow[z](t)=[x]([y](t)) & \text{(composition)}\\
[z]=\int[x]\Rightarrow[z](t)=\left[\int_0^t x^-(\tau)\,d\tau,\int_0^t x^+(\tau)\,d\tau\right] & \text{(integral)}
\end{array}
$$

Tube Contractors

Tube arithmetic allows us to build contractors [3] [5].

Consider for instance the differential constraint

$$\begin{array}{rcl} \dot{x}(t) & = & x(t+1)\cdot u(t), \\ x(t) & \in & [x](t), \dot{x}(t) \in [\dot{x}](t), u(t) \in [u](t) \end{array}$$

We decompose as follows

$$\left\{ \begin{array}{rcl} x(t) & = & x(0) + \int_0^t y(\tau)\, d\tau \\ y(t) & = & a(t)\cdot u(t). \\ a(t) & = & x(t+1) \end{array} \right.$$

Possible contractors are

$$\left\{\begin{array}{lll}[x](t) & = & [x](t)\cap\left([x](0)+\int_0^t[y](\tau)\,d\tau\right)\\ [y](t) & = & [y](t)\cap[a](t)\cdot[u](t)\\ [u](t) & = & [u](t)\cap\frac{[y](t)}{[a](t)}\\ [a](t) & = & [a](t)\cap\frac{[y](t)}{[u](t)}\\ [a](t) & = & [a](t)\cap[x](t+1)\\ [x](t) & = & [x](t)\cap[a](t-1)\end{array}\right.$$

Example. Consider $x(t) \in [x](t)$ with the constraint

$$\forall t,\ x(t) = x(t+1)$$

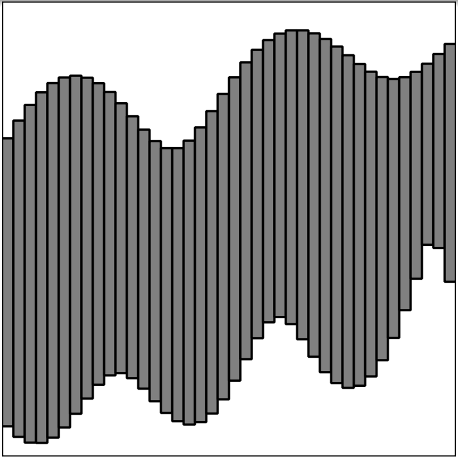
Contract the tube $[x](t)$.

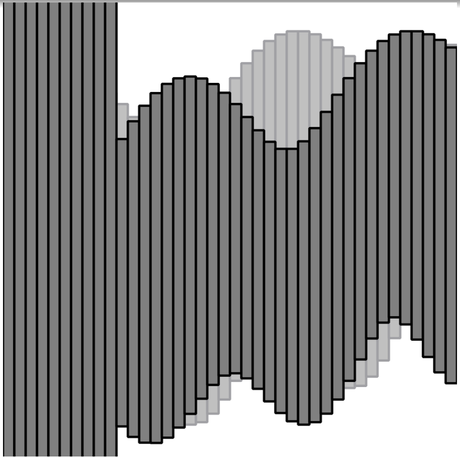
We first decompose into primitive trajectory constraints

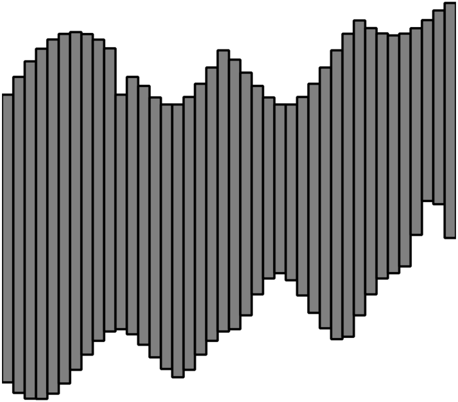
$$
\begin{aligned}
x(t) &= a(t+1) \\
x(t) &= a(t).
\end{aligned}
$$

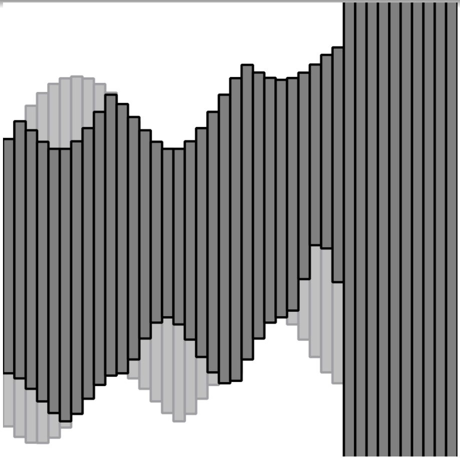
Contractors

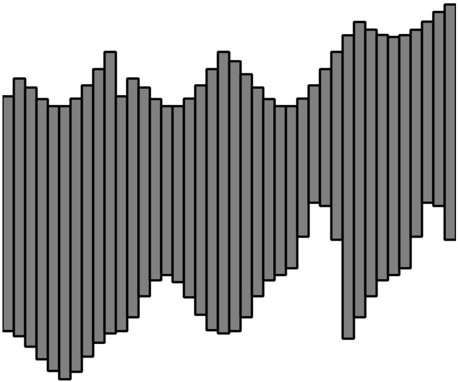
$$
\begin{aligned}
[x](t) &:= [x](t) \cap [a](t+1) \\
[a](t) &:= [a](t) \cap [x](t-1) \\
[x](t) &:= [x](t) \cap [a](t) \\
[a](t) &:= [a](t) \cap [x](t)
\end{aligned}
$$

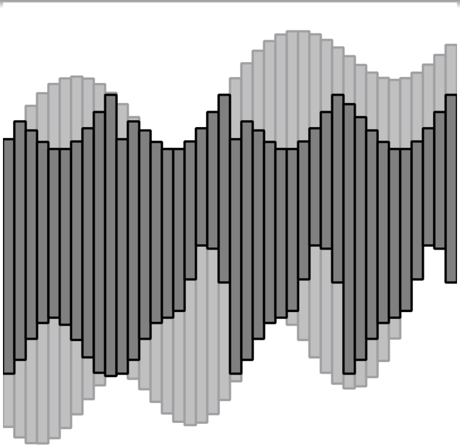


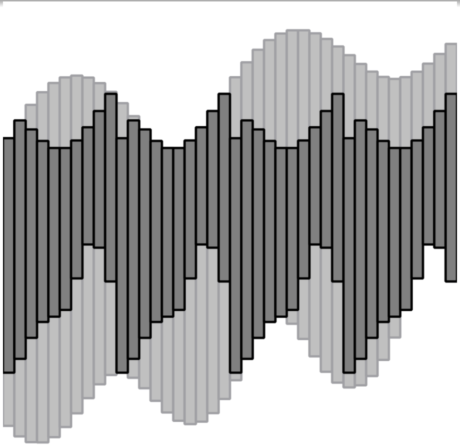


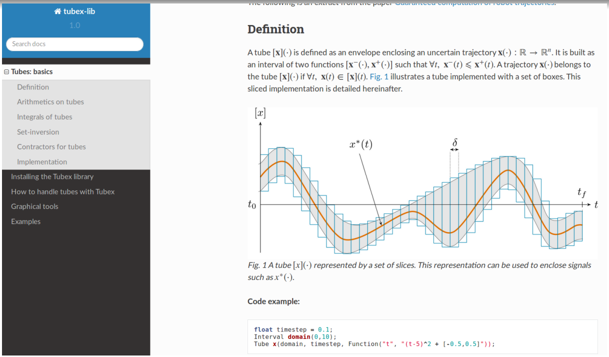












tubex-lib
1.0
Search docs

Tubes: basics
- Definition
- Arithmetics on tubes
- Integrals of tubes
- Set-inversion
- Contractors for tubes
- Implementation

Installing the Tubex library
How to handle tubes with Tubex
Graphical tools
Examples

The following is an extract from the paper Guaranteed computation of robot trajectories.

Definition

A tube $[\mathbf{x}](\cdot)$ is defined as an envelope enclosing an uncertain trajectory $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$. It is built as an interval of two functions $[\mathbf{x}^-(\cdot), \mathbf{x}^+(\cdot)]$ such that $\forall t,\ \mathbf{x}^-(t) \leqslant \mathbf{x}^+(t)$. A trajectory $\mathbf{x}(\cdot)$ belongs to the tube $[\mathbf{x}](\cdot)$ if $\forall t,\ \mathbf{x}(t) \in [\mathbf{x}](t)$. Fig. 1 illustrates a tube implemented with a set of boxes. This sliced implementation is detailed hereinafter.

Fig. 1 A tube $[x](\cdot)$ represented by a set of slices. This representation can be used to enclose signals such as $x^*(\cdot)$.

Code example:

```
float timestep = 0.1;
Interval domain(0,10);
Tube x(domain, timestep, Function("t", "(t-5)^2 + [-0.5,0.5]"));
```

http://www.simon-rohou.fr/research/tubex-lib/ [8]

Time-space estimation

Classical state estimation

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) & t \in \mathbb{R} \\ \mathbf{0} = \mathbf{g}(\mathbf{x}(t), t) & t \in \mathbb{T} \subset \mathbb{R}. \end{cases}$$

Space constraint $\mathbf{g}(\mathbf{x}(t), t) = 0$.

Example.

$$\begin{cases} \dot{x}_1 = x_3 \cos x_4 \\ \dot{x}_2 = x_3 \cos x_4 \\ \dot{x}_3 = u_1 \\ \dot{x}_4 = u_2 \\ (x_1(5) - 1)^2 + (x_2(5) - 2)^2 - 4 = 0 \\ (x_1(7) - 1)^2 + (x_2(7) - 2)^2 - 9 = 0 \end{cases}$$

With time-space constraints

$$\begin{cases} \dot{\mathbf{x}}(t) & = & \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) & t \in \mathbb{R} \\ \mathbf{0} & = & \mathbf{g}(\mathbf{x}(t), \mathbf{x}(t'), t, t') & (t, t') \in \mathbb{T} \subset \mathbb{R} \times \mathbb{R}. \end{cases}$$

Example. An ultrasonic underwater robot with state

$$\mathbf{x} = (x_1, x_2, \dots) = (x, y, \theta, v, \dots)$$

At time t the robot emits an onmidirectional sound. At time t' it receives it

$$\left(x_1 - x_1^{'}\right)^2 + \left(x_2 - x_2^{'}\right)^2 - c\left(t - t'\right)^2 = 0.$$

Mass spring problem

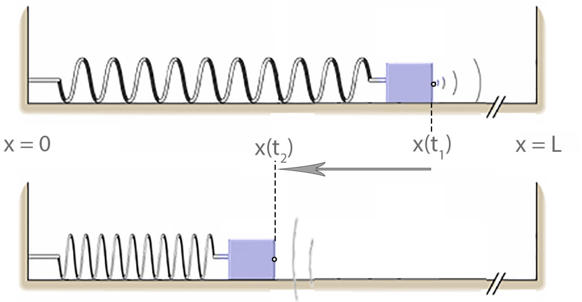
The mass spring satisfies

$$\ddot{x} + \dot{x} + x - x^3 = 0$$

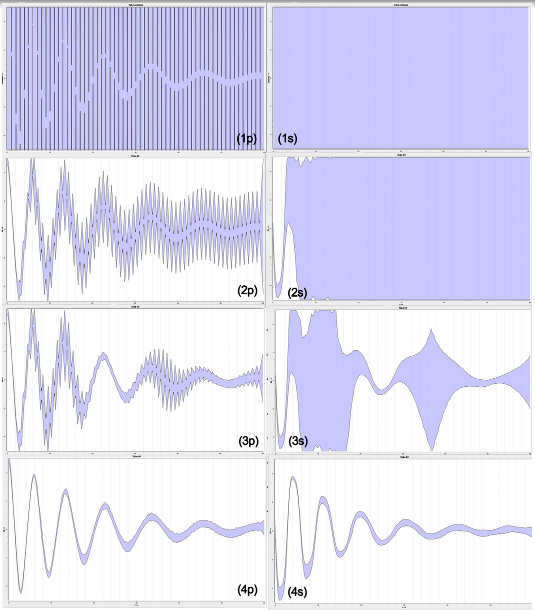
i.e.

$$\begin{cases} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & -x_2 - x_1 + x_1^3 \end{cases}$$

The initial state is unknown.


x = 0
x(t_2)
x(t_1)
x = L

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -x_2 - x_1 + x_1^3 \\ L - x_1(t_1) + L - x_1(t_2) = c(t_2 - t_1). \end{cases}$$


(1p)
(1s)
(2p)
(2s)
(3p)
(3s)
(4p)
(4s)

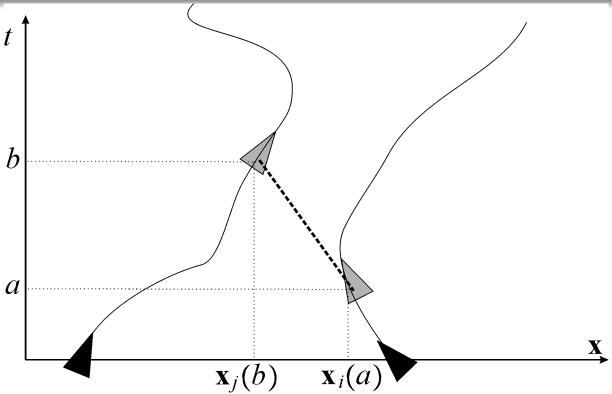
Swarm localization

Consider n robots $\mathscr{R}_1, \dots, \mathscr{R}_n$ described by

$$\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i), \mathbf{u}_i \in [\mathbf{u}_i].$$

Omnidirectional sounds are emitted and received.

A *ping* is a 4-uple (a, b, i, j) where a is the emission time, b is the reception time, i is the emitting robot and j the receiver.


t
b
a
x
x_j(b)
x_i(a)

With the time space constraint

$$\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i), \mathbf{u}_i \in [\mathbf{u}_i].$$
$$g\left(\mathbf{x}_{i(k)}(a(k)), \mathbf{x}_{j(k)}(b(k)), a(k), b(k)\right) = 0$$

where

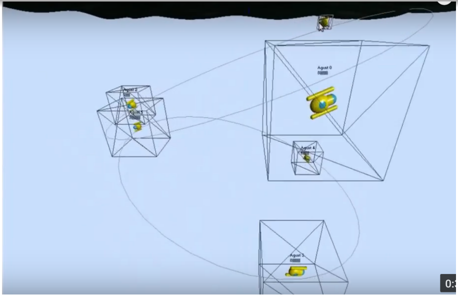
$$g(\mathbf{x}_i, \mathbf{x}_j, a, b) = \|x_1 - x_2\| - c(b - a).$$

Clocks are uncertain. We only have measurements $\tilde{a}(k), \tilde{b}(k)$ of $a(k), b(k)$ thanks to clocks h_i. Thus

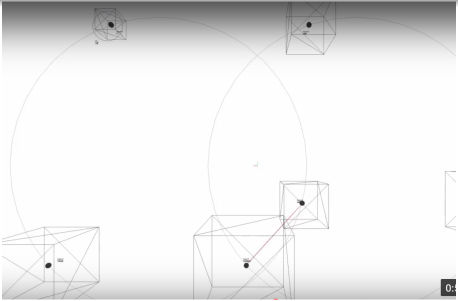
$$\begin{array}{l}
\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i), \mathbf{u}_i \in [\mathbf{u}_i]. \\
g\left(\mathbf{x}_{i(k)}(a(k)), \mathbf{x}_{j(k)}(b(k)), a(k), b(k)\right) = 0 \\
\tilde{a}(k) = h_{i(k)}(a(k)) \\
\tilde{b}(k) = h_{j(k)}(b(k))
\end{array}$$

The drift of the clocks is bounded

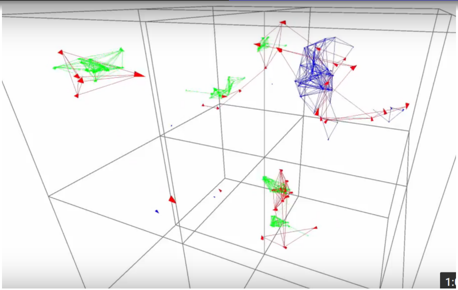
$$
\begin{aligned}
&\dot{\mathbf{x}}_i = \mathbf{f}\left(\mathbf{x}_i, \mathbf{u}_i\right), \mathbf{u}_i \in [\mathbf{u}_i]. \\
&g\left(\mathbf{x}_{i(k)}\left(a(k)\right), \mathbf{x}_{j(k)}\left(b(k)\right), a(k), b(k)\right) = 0 \\
&\tilde{a}(k) = h_{i(k)}\left(a(k)\right) \\
&\tilde{b}(k) = h_{j(k)}\left(b(k)\right) \\
&\dot{h}_i = 1 + n_h,\ n_h \in [n_h]
\end{aligned}
$$



https://youtu.be/j-ERcoXF1Ks [2]

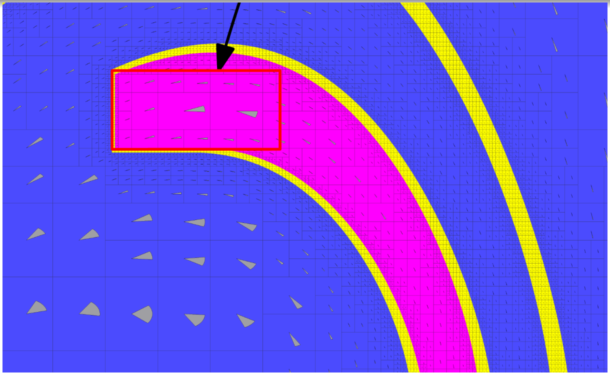


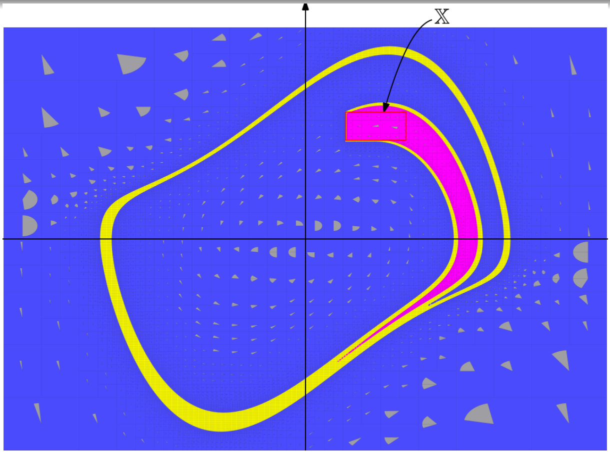
https://youtu.be/jr8xKIe0Nds



https://youtu.be/GycJxGFvYE8

Eulerian approach




X

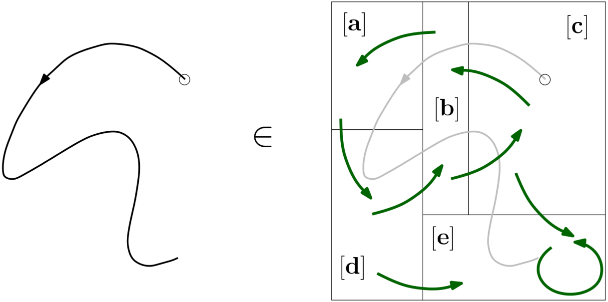
Maze

An ***interval*** is *a domain* which encloses a real number.

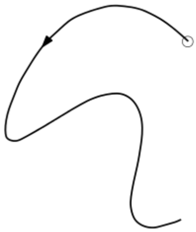
A ***polygon*** is *a domain* which encloses a vector of $\mathbb{R}^n$.

A ***maze*** is *a domain* which encloses a path. [7]

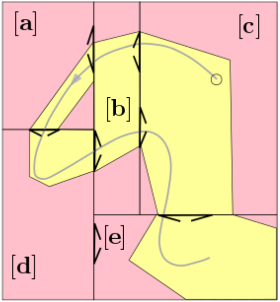
A maze is a set of paths.



Mazes can be made more accurate:



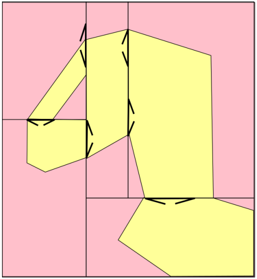
$\in$



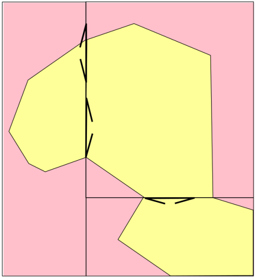
Here, a **maze** $\mathscr{L}$ is composed of [7][6].

- A paving $\mathscr{P}$
- Doors between adjacent boxes

The set of mazes forms a lattice with respect to $\subset$.



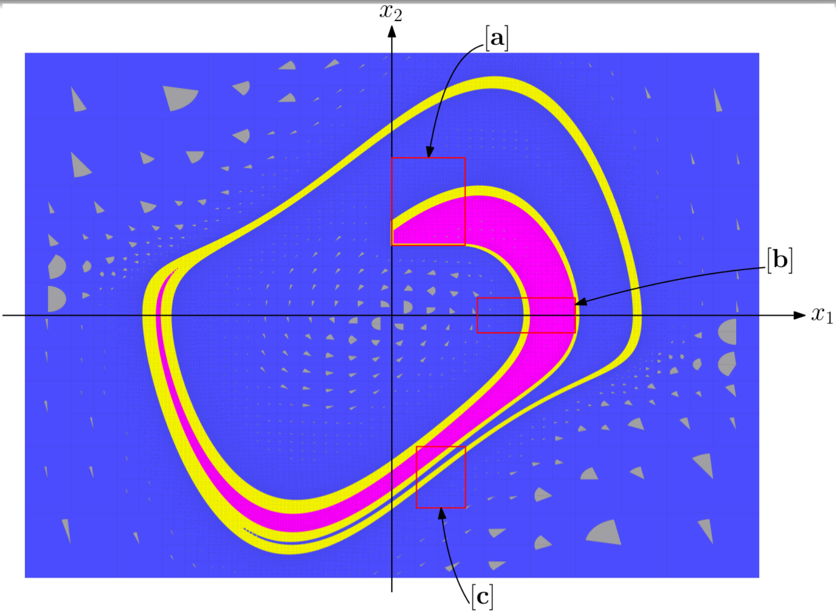
$\subset$

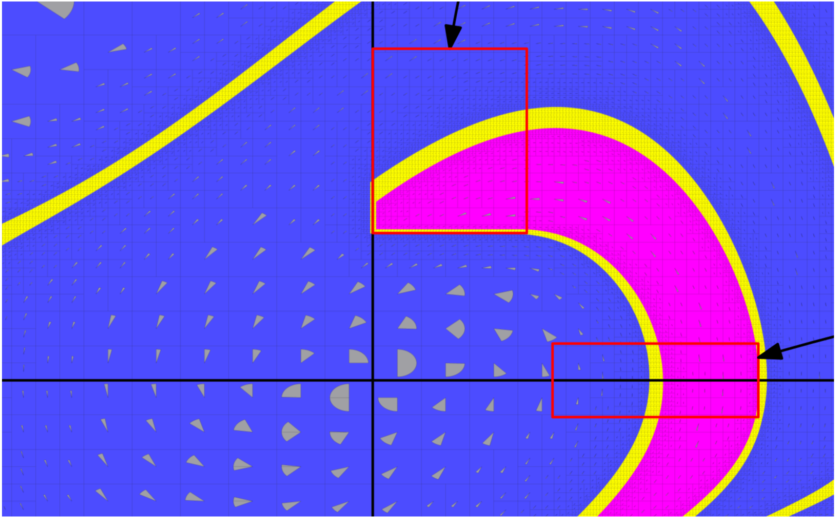


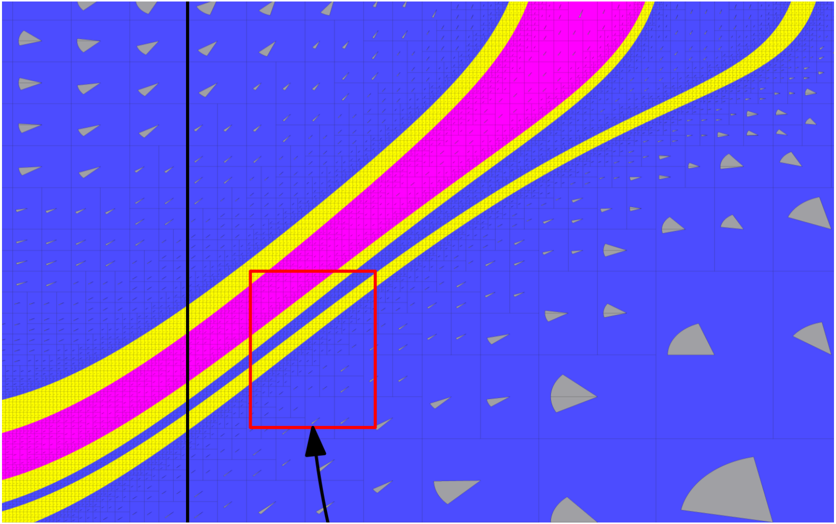
Eulerian smoother

Example. Take the Van der Pol system with

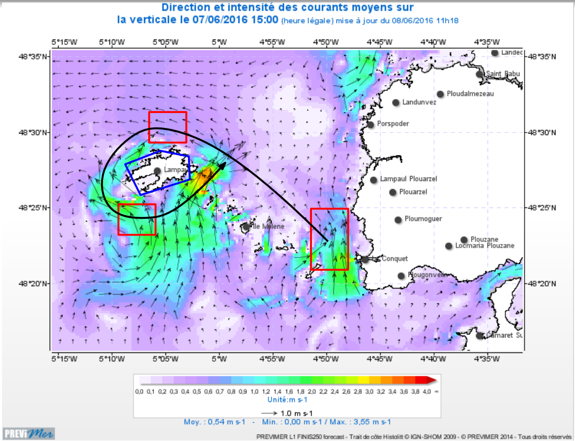
$$\begin{aligned}
\mathbb{X}_0 &= [\mathbf{a}] = [0, 0.6] \times [0.8, 1.8] \\
\mathbb{X}_1 &= [\mathbf{b}] = [0.7, 1.5] \times [-0.2, 0.2] \\
\mathbb{X}_2 &= [\mathbf{c}] = [0.2, 0.6] \times [-2.2, -1.5]
\end{aligned}$$


x_2
x_1
[a]
[b]
[c]





An application of Eulerian state estimation moving taking advantage of ocean currents.



Visiting the three red boxes using a buoy that follows the currents is an Eulerian state estimation problem

F. Le Bars, J. Sliwka, O. Reynet, and L. Jaulin.
State estimation with fleeting data.
Automatica, 48(2):381–387, 2012.

A. Bethencourt and L. Jaulin.
Cooperative localization of underwater robots with unsynchronized clocks.
Journal of Behavioral Robotics, 4(4):233–244, 2013.

A. Bethencourt and L. Jaulin.
Solving non-linear constraint satisfaction problems involving time-dependant functions.
Mathematics in Computer Science, 8(3), 2014.

G. Chabert and L. Jaulin.
Contractor Programming.
Artificial Intelligence, 173:1079–1100, 2009.

A. Chapoutot, J. Alexandre dit Sandretto, and O. Mullier.

Dynibex, available at ,
`http://perso.ensta-paristech.fr/˜chapoutot/dynibex/`.

ENSTA, 2015.

T. Le Mézo L. Jaulin and B. Zerr.
Bracketing the solutions of an ordinary differential equation with uncertain initial conditions.
Applied Mathematics and Computation, 2017.

T. Le Mézo, L. Jaulin, and B. Zerr.
An interval approach to compute invariant sets.
IEEE Transaction on Automatic Control, 2017.

S. Rohou, L. Jaulin, M. Mihaylova, F. Le Bars, and S. Veres.
Guaranteed Computation of Robots Trajectories.
Robotics and Autonomous Systems, 93:76–84, 2017.