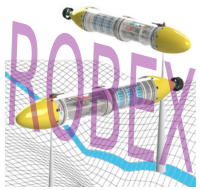
Set-membership Terrain Based Navigation

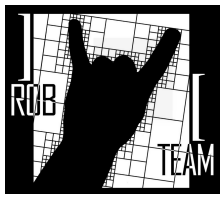
Example of use of the Codac library

Simon Rohou

ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, Brest, France

AID 2022
8th March 2022





Section 1

Codac in a nutshel

Codac in a nutshel

Domains (wrappers)

- for reals $x \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$: intervals $[x]$ and boxes $[\mathbf{x}]$
- for trajectories $x(\cdot) : \mathbb{R} \to \mathbb{R}$: tubes $[x](\cdot)$

Codac in a nutshel

Domains (wrappers)

- for reals $x \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$: intervals $[x]$ and boxes $[\mathbf{x}]$
- for trajectories $x(\cdot) : \mathbb{R} \to \mathbb{R}$: tubes $[x](\cdot)$
- for subsets $\mathbb{X} \subset \mathbb{R}^n$: thicksets $\mathbb{X} \in [\mathbb{X}] = [\mathbb{X}^-, \mathbb{X}^+]$

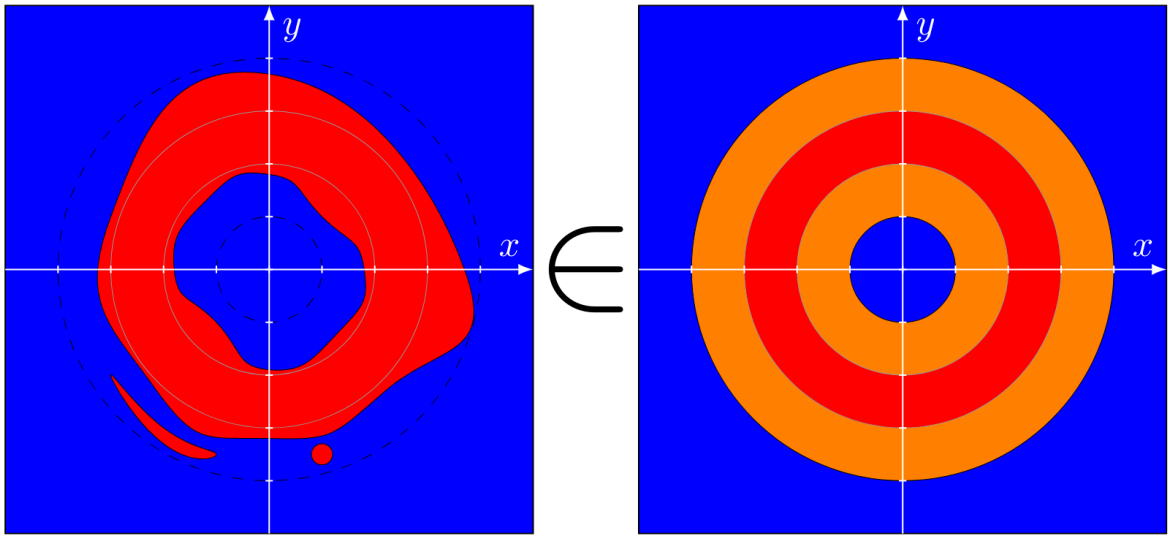


Illustration of a thickset (right-hand side)
for enclosing and uncertain red set (left-hand side)

Thick set inversion
Desrochers, Jaulin. *Artificial Intelligence. Volume 249, Issue C, Pages 1-18*, 2017

Codac in a nutshel

Domains (wrappers)

- for reals $x \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$: intervals $[x]$ and boxes $[\mathbf{x}]$
- for trajectories $x(\cdot) : \mathbb{R} \to \mathbb{R}$: tubes $[x](\cdot)$
- for subsets $\mathbb{X} \subset \mathbb{R}^n$: thicksets $\mathbb{X} \in [\mathbb{X}] = [\mathbb{X}^-, \mathbb{X}^+]$
- *etc.*

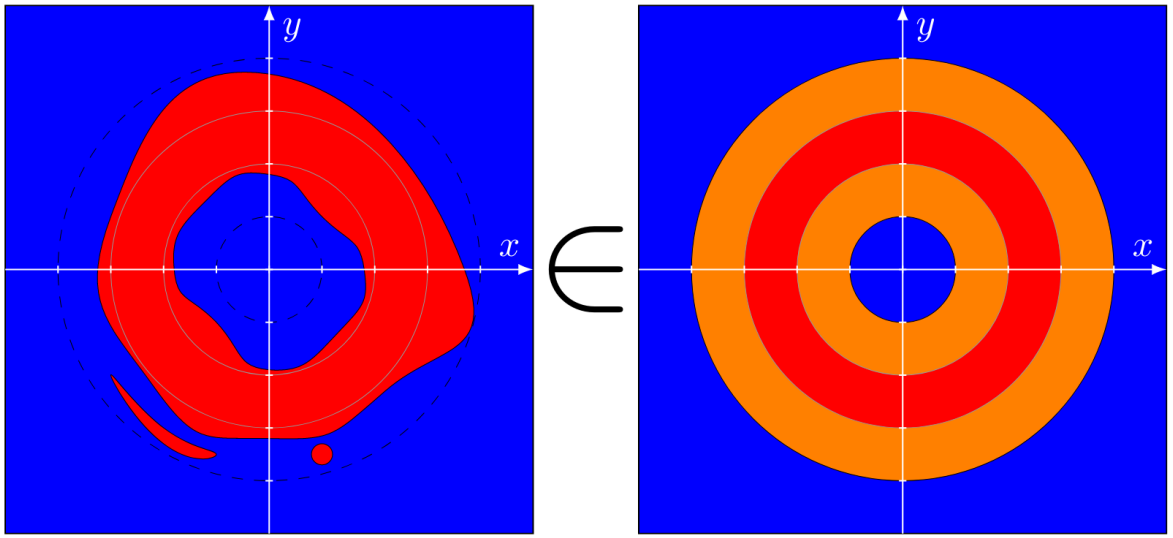


Illustration of a thickset (right-hand side)
for enclosing and uncertain red set (left-hand side)

Thick set inversion
Desrochers, Jaulin. *Artificial Intelligence. Volume 249, Issue C, Pages 1-18*, 2017

Codac in a nutshel

Catalog Of Domains And Contractors

Several types of **domains**:

- `Interval`, `IntervalVector`, `IntervalMatrix` (from IBEX)
- `Tube`, `TubeVector`, `Slice`
- `Thickset`
- ...

Codac in a nutshel

Catalog Of Domains And Contractors

Several types of **domains**:

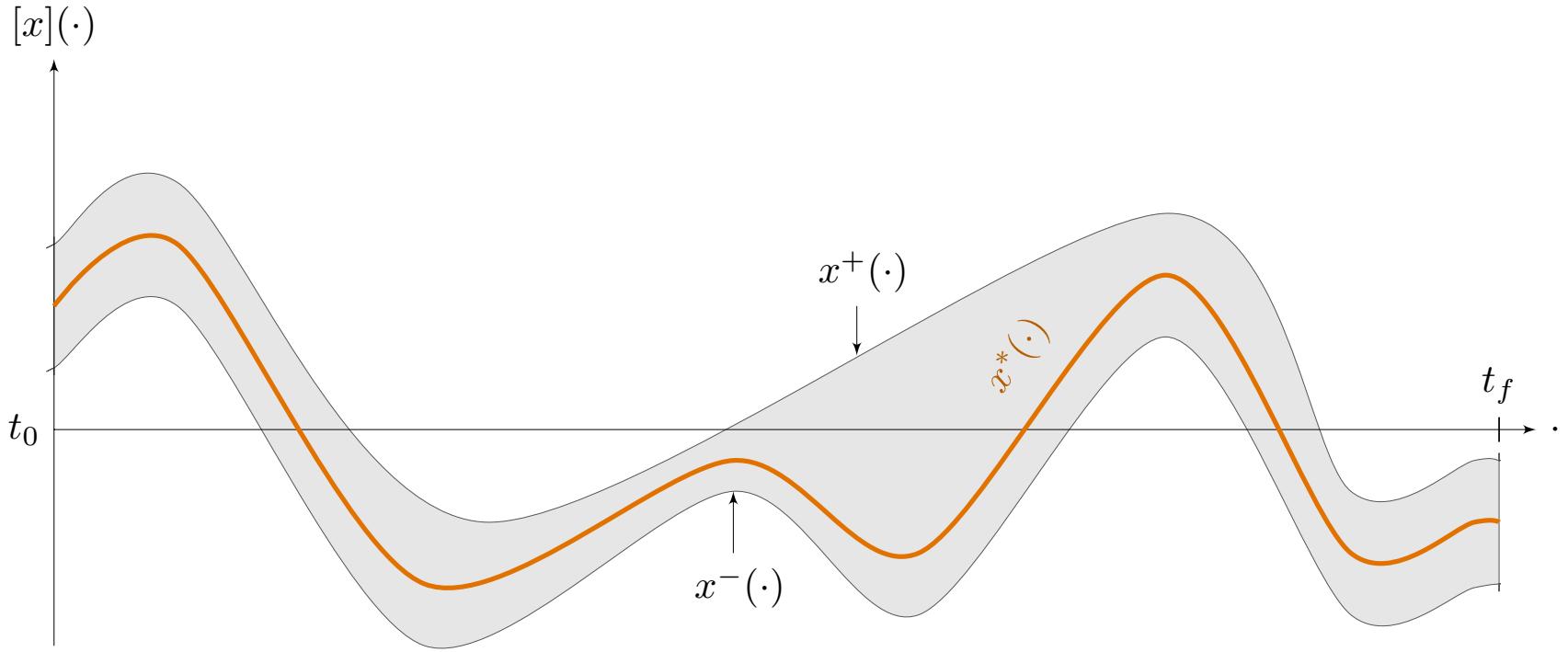
- `Interval`, `IntervalVector`, `IntervalMatrix` (from IBEX)
- `Tube`, `TubeVector`, `Slice`
- `Thickset`
- ...

Contractors for various constraints:

- non-linear constraints $\mathbf{f}(\mathbf{x}) = \mathbf{0}$
- geometric constraints: distance, polar equation, circles, ...
- differential equations: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$
- time uncertainties: $\mathbf{y} = \mathbf{x}(t), \quad \text{with } t \in [t]$
- delays: $x(t) = y(t - \tau)$
- ...

Codac in a nutshel

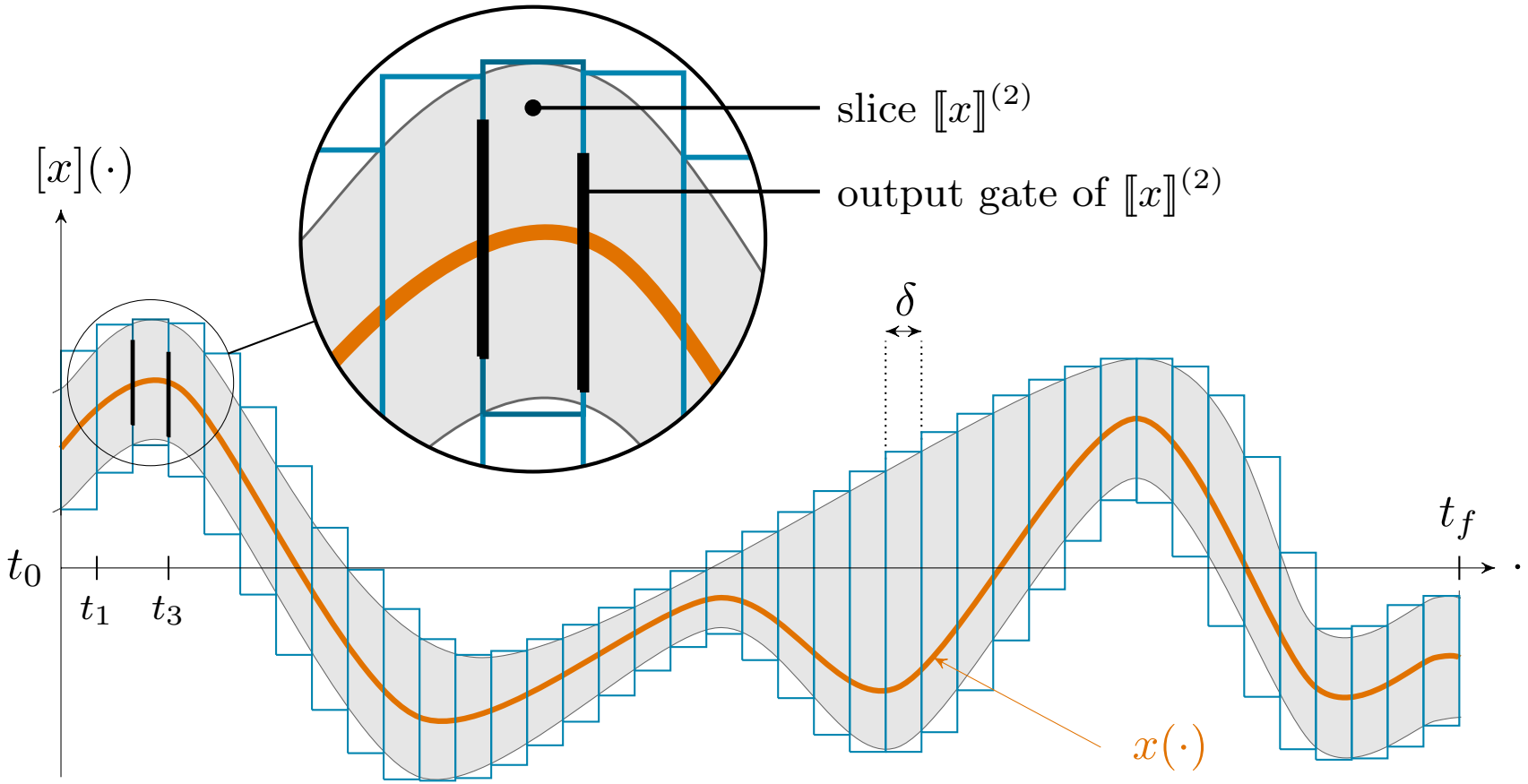
Domains for trajectories: tubes



Example of scalar tube: interval of two trajectories

Codac in a nutshel

Domains for trajectories: tubes



Computer implementation (http://codac.io)

Codac in a nutshel

Example of optimal contractors for the $\mathcal{L}_{\text{polar}}$ constraint

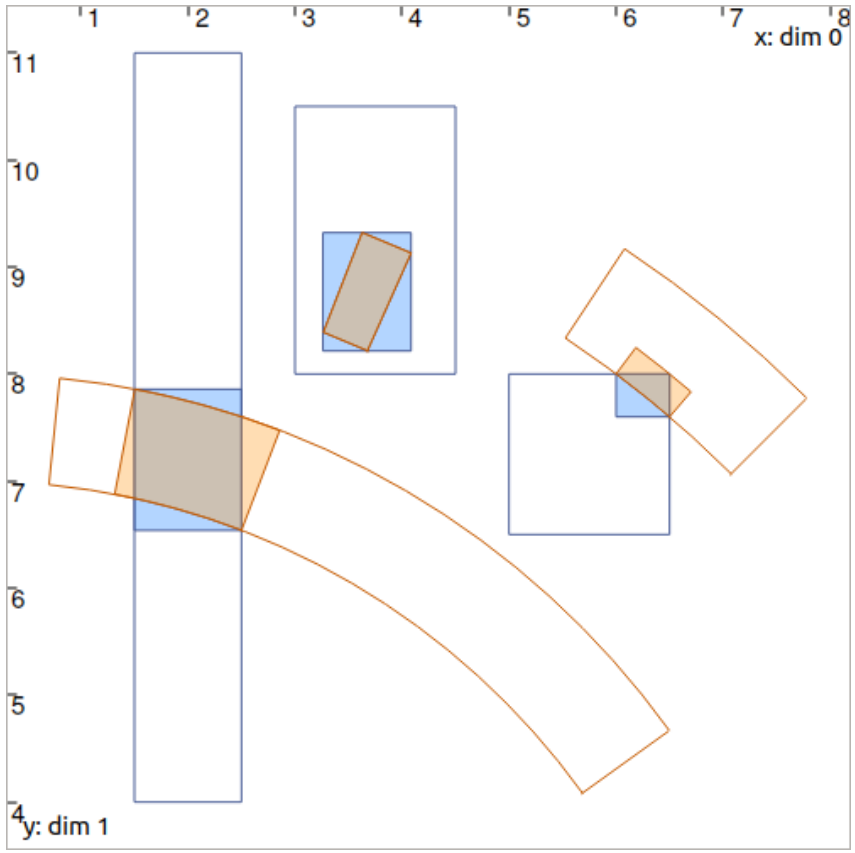
$$\mathcal{L}_{\text{polar}} : \begin{cases} x = \rho \cos\theta \\ y = \rho \sin\theta \end{cases}$$

Optimally dealt with by:

$$\mathcal{C}_{\text{polar}}\big([x],[y],[\rho],[\theta]\big)$$

Using Codac:

```
x=Interval(..)
y=Interval(..)
r=Interval(..)
theta=Interval(..)

ctc.polar.contract(x,y,r,theta)
```



A Minimal contractor for the Polar equation: application to robot localization
Desrochers, Jaulin. *Engineering Applications of Artificial Intelligence, 55(Supplement C):83–92*, 2016

Codac in a nutshel

Codac: make interval research available to the community

The library is open source and available:

- in Python and C++
- on Linux, Windows, MacOS systems
- from official packages:
  Python package: `pip install codac`
  Debian in progress..: `sudo apt install libcodac`

http://www.codac.io

Section 2

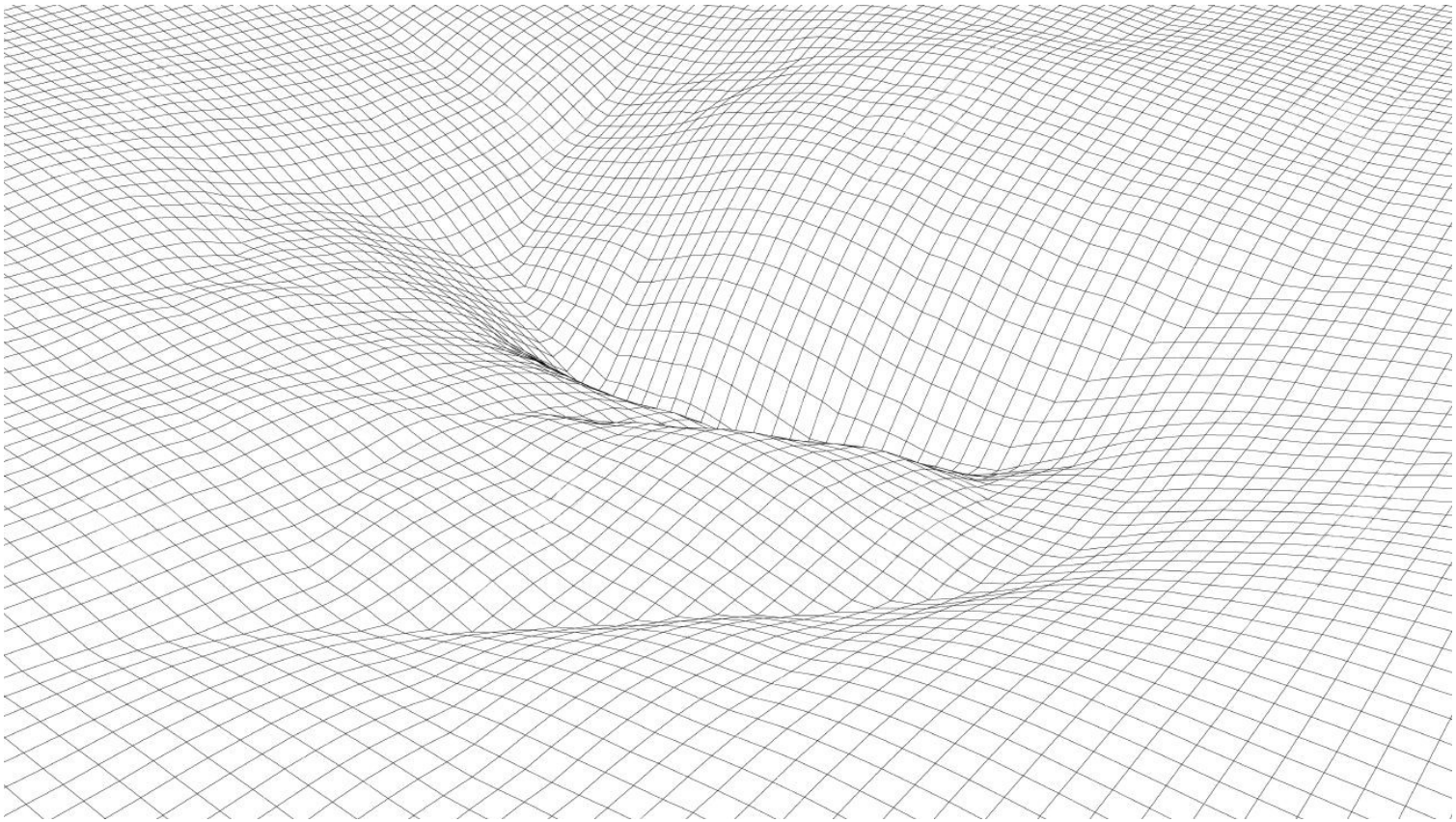
Motivation: robot localization

Motivation: robot localization

Problem: wide environment and poor observations

Under the surface:

- **no satellite** navigation systems available
- **no seamarks** or points of interest

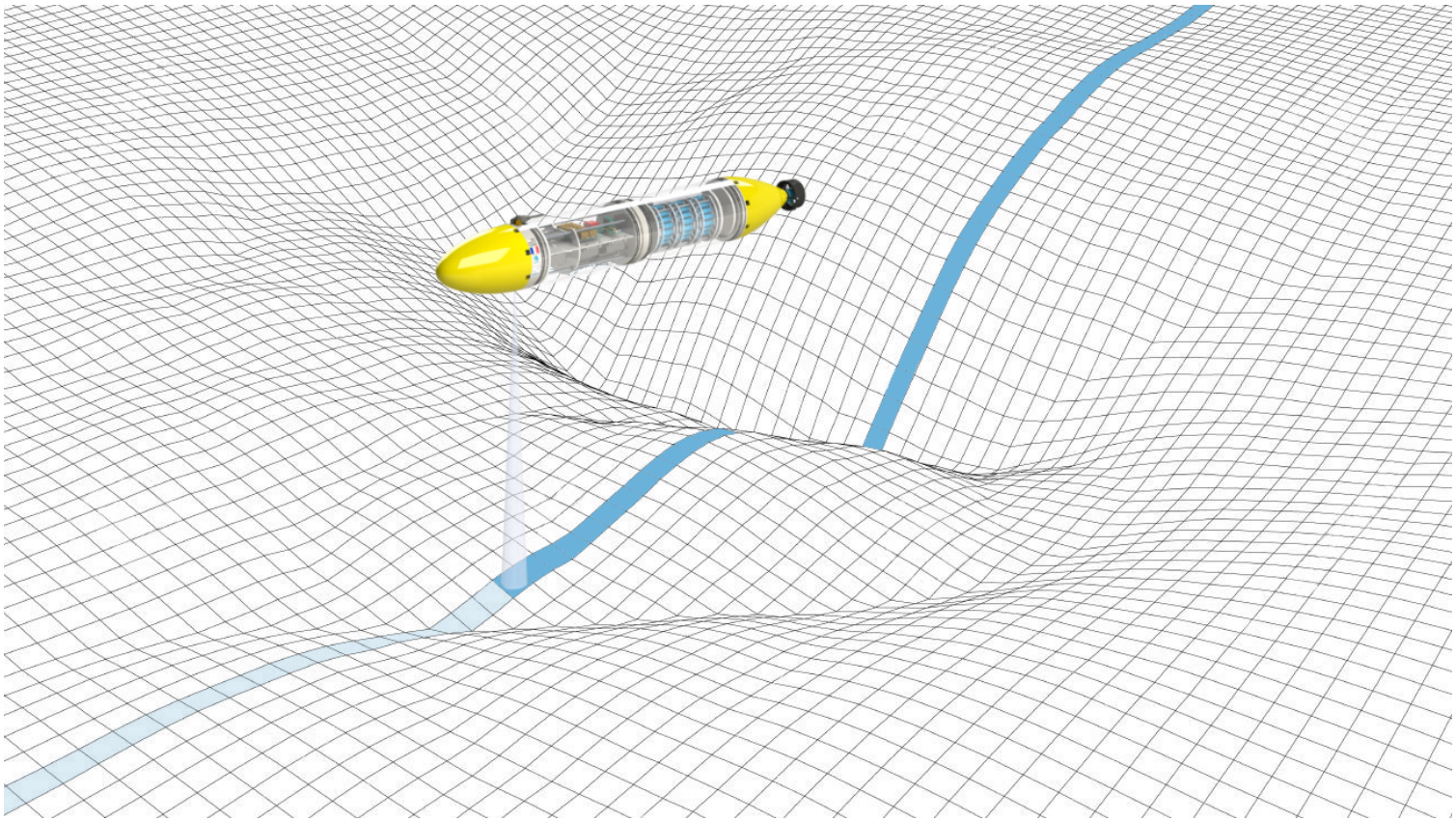


Motivation: robot localization

Problem: wide environment and poor observations

Available data:

- **bathymetric** measurements (scalar values)
- prior knowledge of the environment (embedded map)

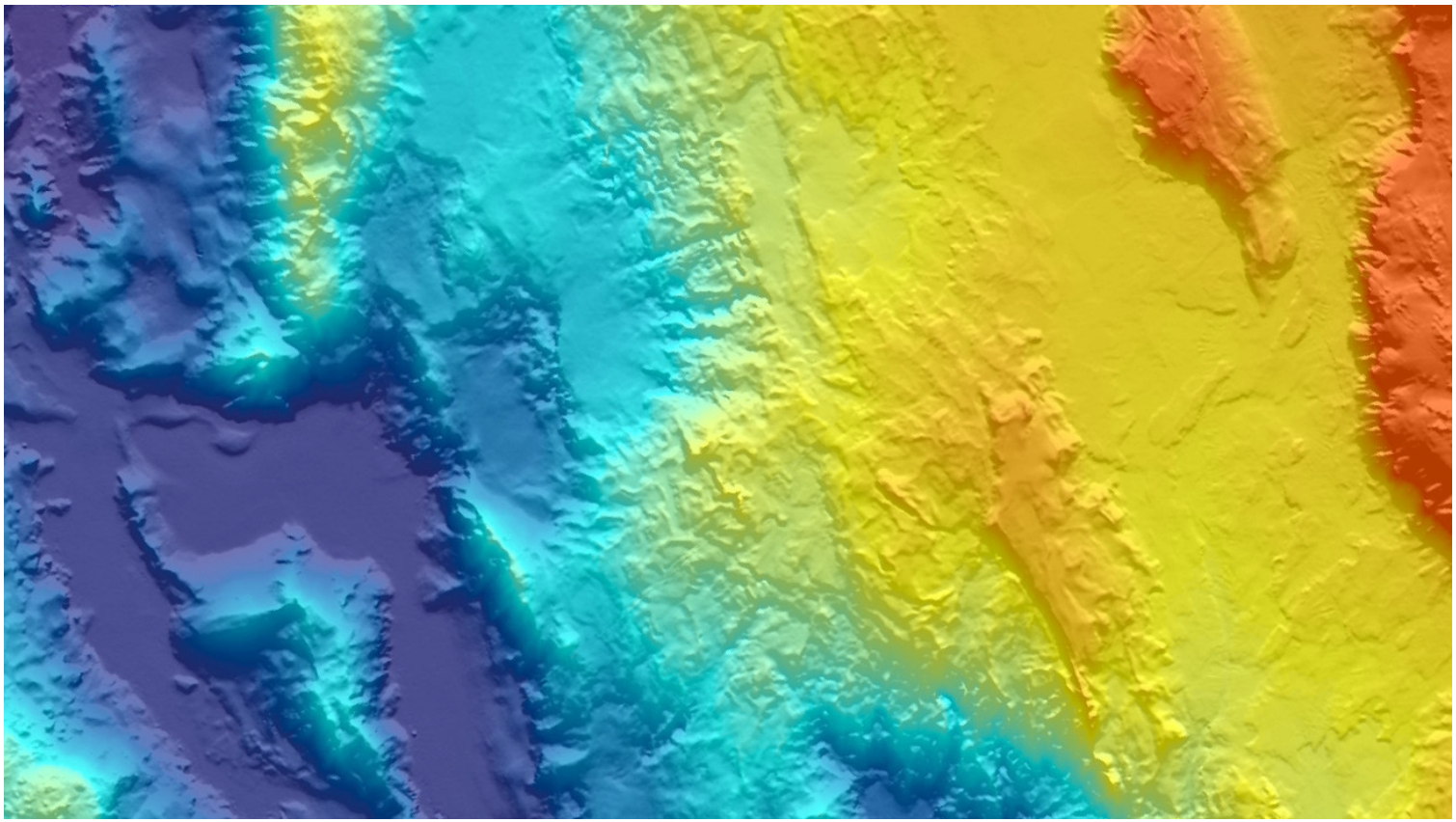


Motivation: robot localization

Problem: wide environment and poor observations

Available data:

- **bathymetric** measurements (scalar values)
- prior knowledge of the environment (embedded map)

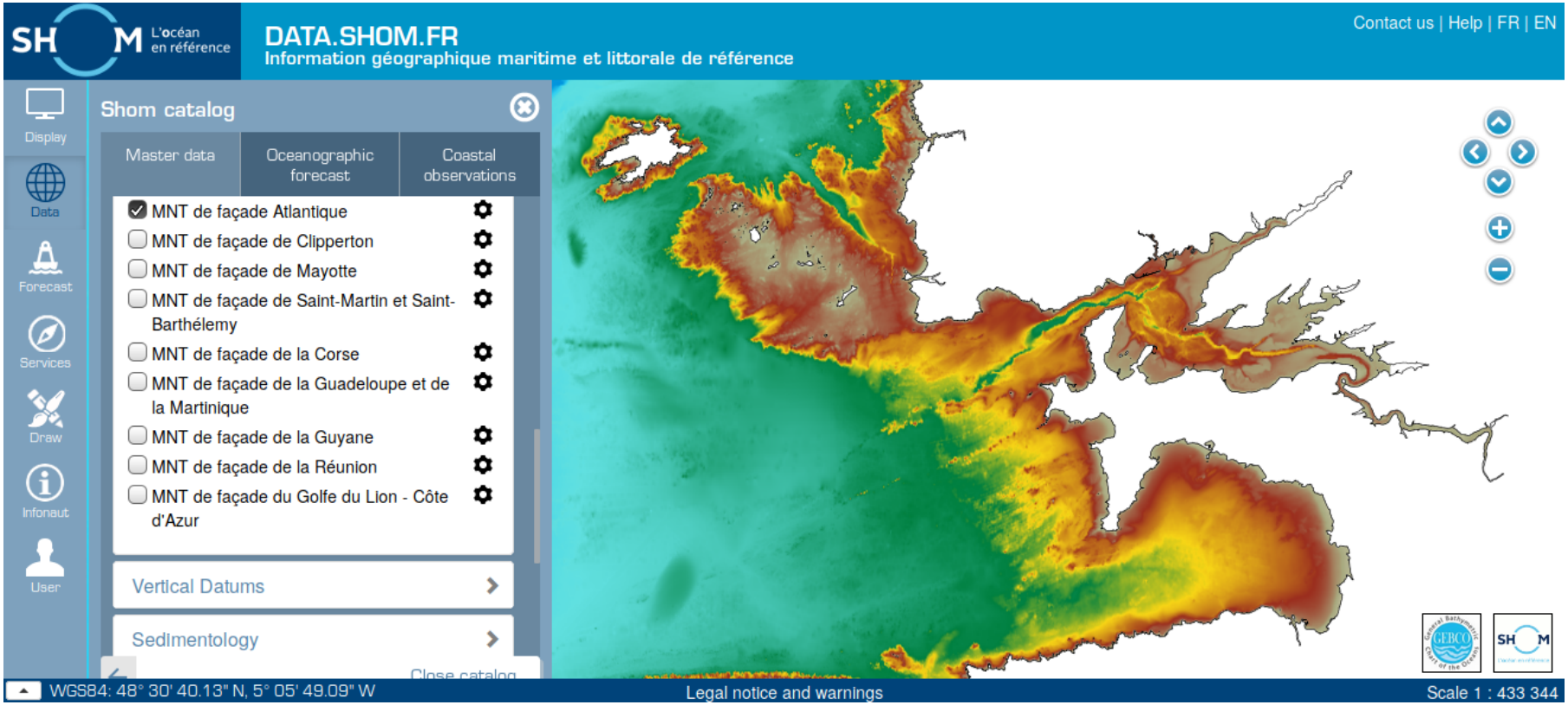


Motivation: robot localization

Available Digital Elevation Models (DEM)

Portail data.shom.fr

- Shom: Service Hydrographique et Océanographique de la Marine
- many marine DEM available



Section 3

Set-membership Terrain Based Navigation

Set-membership Terrain Based Navigation

Formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & (\text{navigation}) \\ \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function

Set-membership Terrain Based Navigation

Formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & (\text{navigation}) \\ z_i = g\big(\mathbf{x}(t_i), \mathbb{M}\big) & (\text{measurements}) \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function
- z_i: scalar measurements (at t_i)
- $\mathbb{M}$: terrain knowledge

Set-membership Terrain Based Navigation

Formalization

Robot localization = state estimation problem.
Classically, we have:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(navigation)} \\ z_i = g\big(\mathbf{x}(t_i), \mathbb{M}\big) & \text{(measurements)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function
- z_i: scalar measurements (at t_i) (ex: scalar bathymetric values)
- $\mathbb{M}$: terrain knowledge (ex: digital elevation map)

Set-membership Terrain Based Navigation

Formalization

Robot localization = state estimation problem.
Classically, we have:

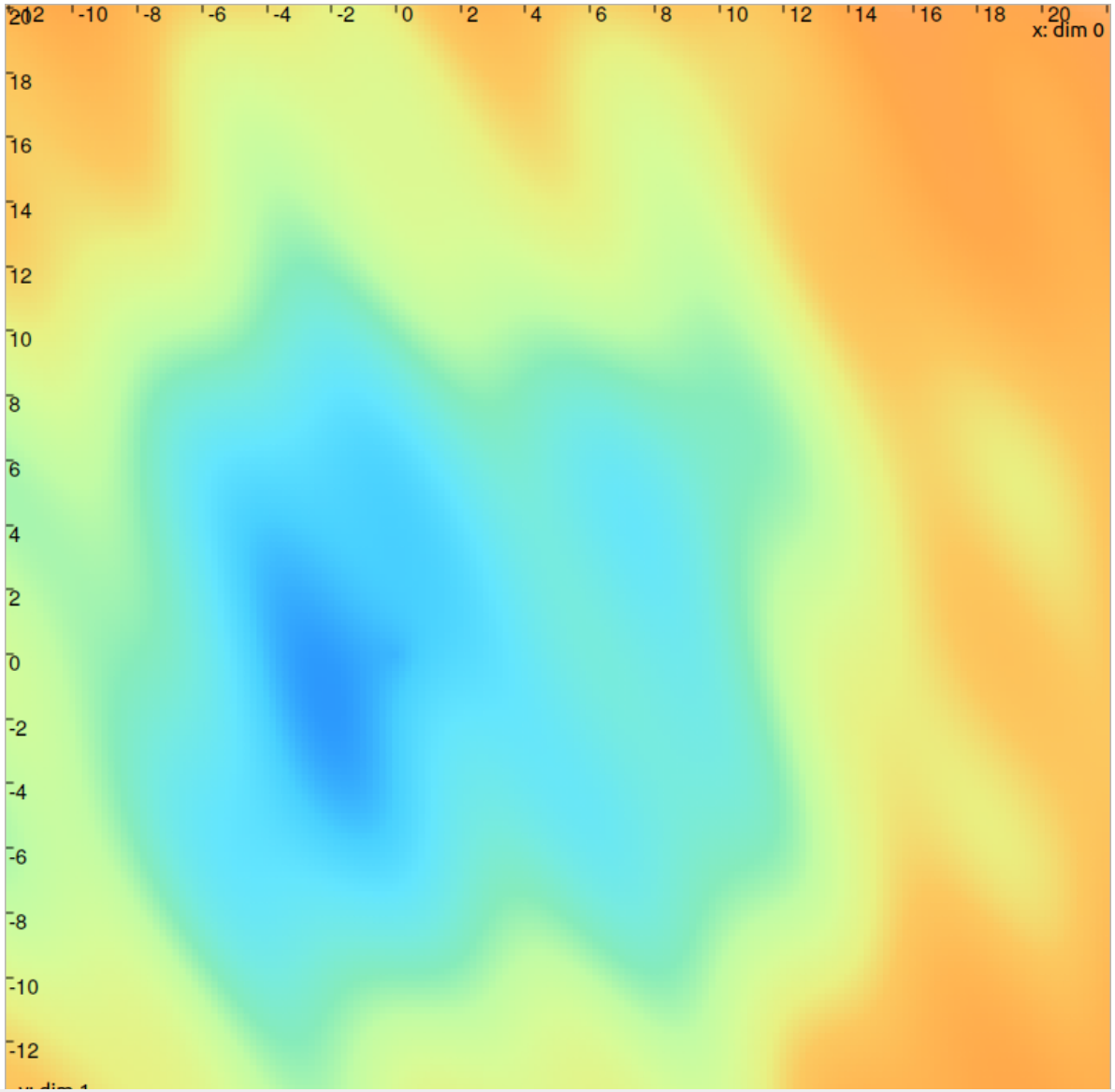
$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(navigation)} \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) & \text{(measurements)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, bearing, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function
- z_i: scalar measurements (at t_i) (ex: scalar bathymetric values)
- $\mathbb{M}$: terrain knowledge (ex: digital elevation map)

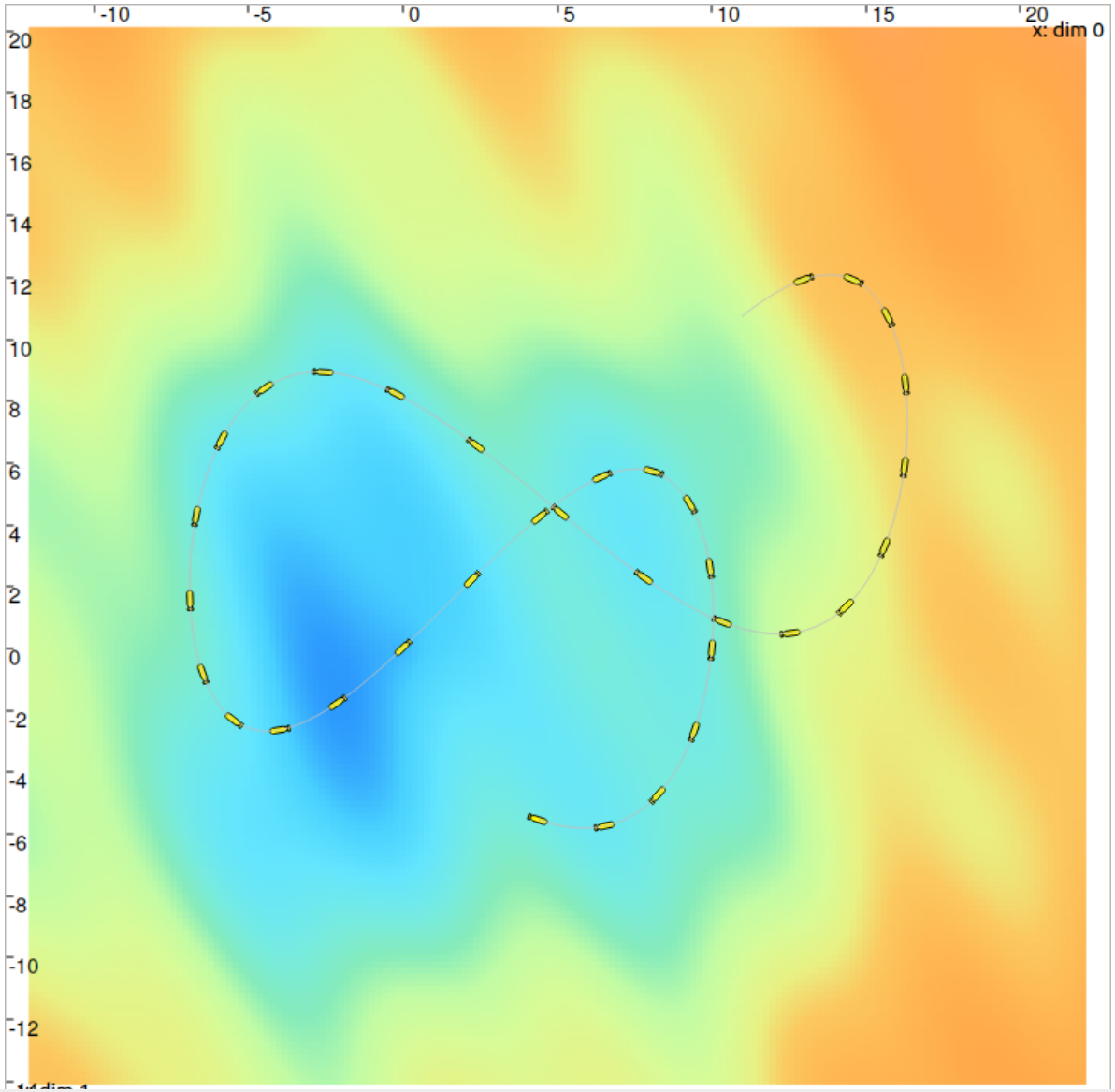
Set-membership Terrain Based Navigation

Simulated dataset



Set-membership Terrain Based Navigation

Simulated dataset



Set-membership Terrain Based Navigation

Simulated dataset

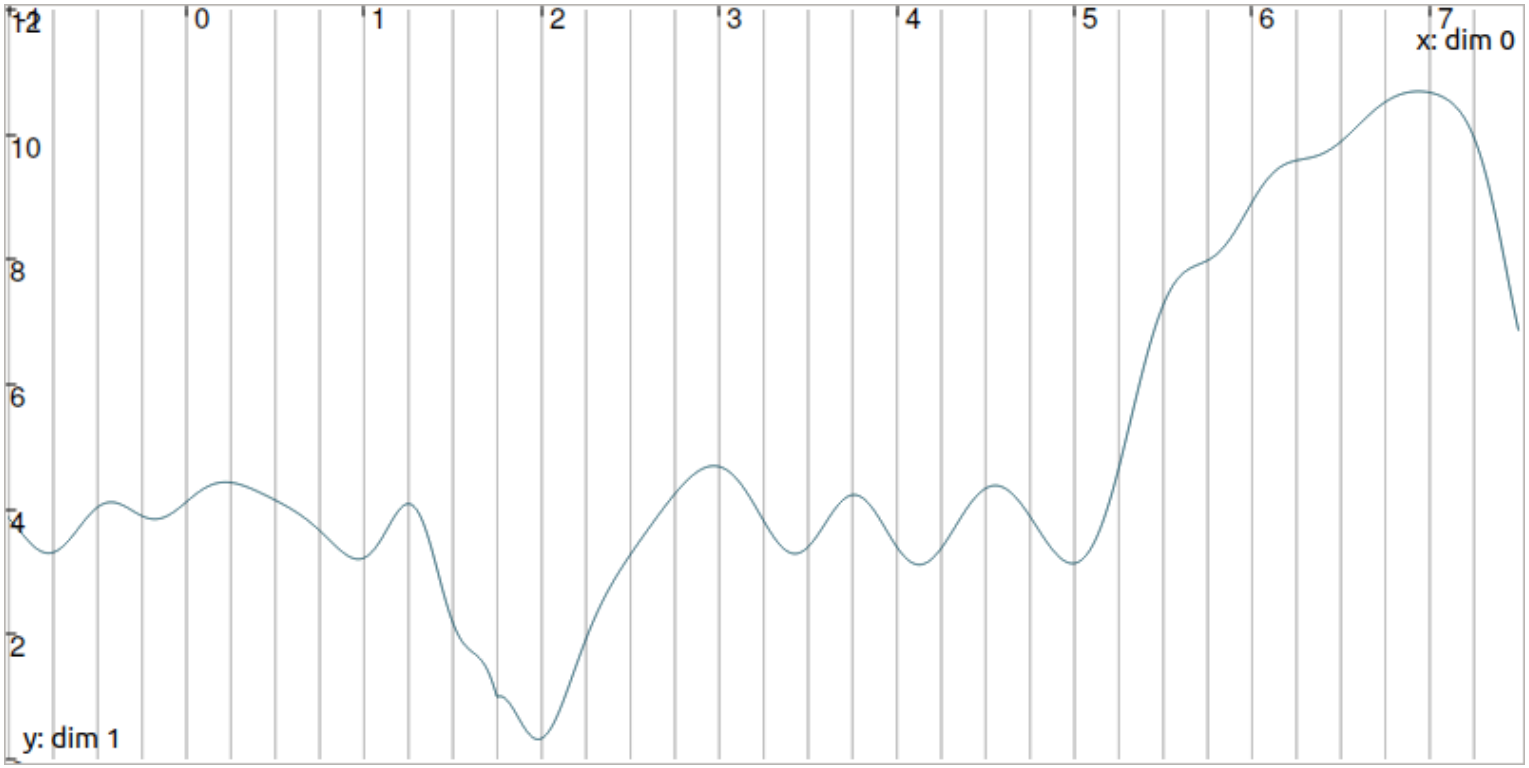


Illustration of a bathymetric signal $z(t)$, observed by the robot.
Only discrete measurements $z_i = z(t_i)$ are considered.

Section 4

Constraint propagation method

Constraint propagation method

Involved variables and domains

Variables:

- reals: $z_i \in \mathbb{R}$
- trajectories: $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$

Constraint propagation method

Involved variables and domains

Variables:

- reals: $z_i \in \mathbb{R}$
- trajectories: $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$

Numerical domains of the variables:

- intervals: $[z_i] \in \mathbb{IR}$
- tubes: $[\mathbf{x}](\cdot) : \mathbb{R} \to \mathbb{IR}^n$

Constraint propagation method

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Constraint propagation method

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Elementary constraints:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint $\rightarrow \mathcal{L}_{\mathbf{f}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$

Constraint propagation method

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Elementary constraints:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint $\rightarrow \mathcal{L}_{\mathbf{f}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$

Constraint propagation method

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Elementary constraints:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint $\rightarrow \mathcal{L}_{\mathbf{f}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$

Constraint propagation method

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Elementary constraints:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint $\rightarrow \mathcal{L}_{\mathbf{f}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$
- $z_i = g_{\mathbb{M}}\big(\mathbf{p}_i\big) \rightarrow$ map constraint $\rightarrow \mathcal{L}_{g_{\mathbb{M}}}\big(\mathbf{p}_i, z_i\big)$

Constraint propagation method

Decomposition of the problem

System:

$\mathbf{v}(\cdot)$ and $\mathbf{p}_i$ are intermediate variables

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Elementary constraints:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint $\rightarrow \mathcal{L}_{\mathbf{f}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$
- $z_i = g_{\mathbb{M}}\big(\mathbf{p}_i\big) \rightarrow$ map constraint $\rightarrow \mathcal{L}_{g_{\mathbb{M}}}\big(\mathbf{p}_i, z_i\big)$

Constraint propagation method

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

$\mathbf{v}(\cdot)$ and $\mathbf{p}_i$ are intermediate variables

Note: some symbolic solver could break down such problem automatically.

Elementary constraints:

- $\mathbf{v}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \rightarrow$ algebraic constraint $\rightarrow \mathcal{L}_{\mathbf{f}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot) \rightarrow$ derivative constraint $\rightarrow \mathcal{L}_{\frac{d}{dt}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$
- $z_i = g_{\mathbb{M}}\big(\mathbf{p}_i\big) \rightarrow$ map constraint $\rightarrow \mathcal{L}_{g_{\mathbb{M}}}\big(\mathbf{p}_i, z_i\big)$

Constraint propagation method

Examples of already existing contractors

Example 1: consider the constraint $z = x + y$
A minimal **contractor** to apply this constraint is:

$$\begin{pmatrix} [z] \\ [x] \\ [y] \end{pmatrix} \overset{\mathcal{C}_+}{\longmapsto} \begin{pmatrix} [z] \cap ([x] + [y]) \\ [x] \cap ([z] - [y]) \\ [y] \cap ([z] - [x]) \end{pmatrix}$$

Contractor programming: $\mathcal{C}_+\big([z],[x],[y]\big)$

Constraint propagation method

Examples of already existing contractors

Example 1: consider the constraint $z = x + y$
A minimal **contractor** to apply this constraint is:

$$\begin{pmatrix} [z] \\ [x] \\ [y] \end{pmatrix} \overset{\mathcal{C}_+}{\longmapsto} \begin{pmatrix} [z] \cap ([x] + [y]) \\ [x] \cap ([z] - [y]) \\ [y] \cap ([z] - [x]) \end{pmatrix}$$

Contractor programming: $\mathcal{C}_+\big([z],[x],[y]\big)$

Example 2: consider the constraint $c(\cdot) = \int_0^{\cdot} x(\tau)d\tau$
A non-minimal **contractor** to apply this constraint is:

$$\begin{pmatrix} [x](\cdot) \\ [c](\cdot) \end{pmatrix} \overset{\mathcal{C}_\int}{\longmapsto} \begin{pmatrix} [x](\cdot) \\ [c](\cdot) \cap \int_0^{\cdot} [x](\tau)\, d\tau \end{pmatrix}$$

Contractor programming: $\mathcal{C}_\int\big([x](\cdot),[c](\cdot)\big)$

Constraint propagation method

Using Codac

1. Defining the domains:

```
# Creating the state tube:
dt = 0.01
tdomain = Interval(0,6)        # temporal domain
x = TubeVector(tdomain, dt, 4) # dim. 4

# At this point,
# forall t, [x](t)=[-oo,oo]x[-oo,oo]x[-oo,oo]x[-oo,oo]

# The heading x[2] and speed x[3] are measured, so:
x[2] &= heading_data
x[3] &= speed_data
# Considering uncertainties <=> inflating the tube:
x.inflate(0.01)

# We also set bounded bathymetric measurements:
v_z = [ Interval(...), Interval(...), ... ]
```

Constraint propagation method

Using Codac

2. Defining contractors:

```
# Some contractors are objects already defined in the library:
ctc.deriv
ctc.eval
ctc.polar

# Other contractors can be built from analytical expressions:
ctc_plus = CtcFunction(Function("x","y","z","x+y-z"))
ctc_minus = CtcFunction(Function("x","y","z","x-y-z"))
# CtcFunction is related to constraints under the form f(x,..)=0
```

The map contractor is given under the form of an algorithm.

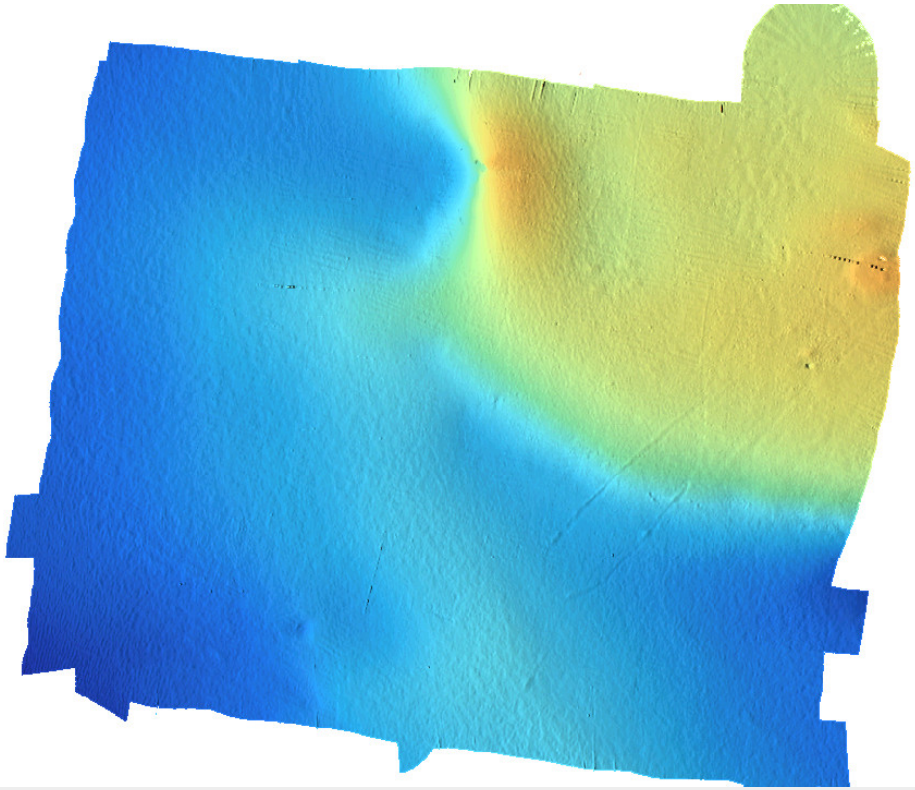
Constraint propagation method

The map constraint $\mathcal{L}_{g_{\mathbb{M}}}(z_i, \mathbf{p}_i) : z_i = g_{\mathbb{M}}(\mathbf{p}_i)$

With $z_i \in \mathbb{R}$, $\mathbf{p_i} \in \mathbb{R}^2$.

Definition of the related operator $\mathcal{C}_{g_{\mathbb{M}}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} \left[[z_i] \cap \{ g_{\mathbb{M}}(\mathbf{p}_i) \mid \mathbf{p}_i \in [\mathbf{p}_i] \} \right] \\ \bigsqcup \{ \mathbf{p}_i \in [\mathbf{p}_i] \mid g_{\mathbb{M}}(\mathbf{p}_i) \in [z_i] \} \end{pmatrix} \tag{1}$$



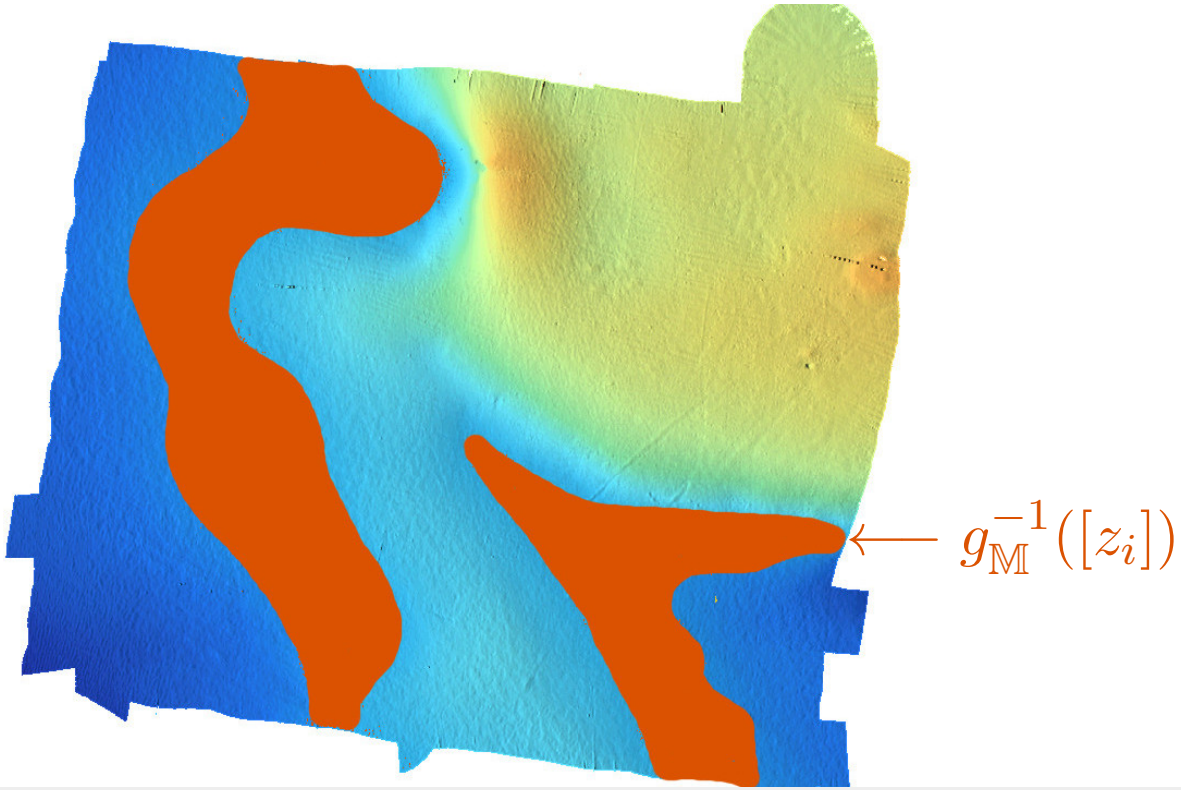
Constraint propagation method

The map constraint $\mathcal{L}_{g_{\mathbb{M}}}(z_i, \mathbf{p}_i) : z_i = g_{\mathbb{M}}(\mathbf{p}_i)$

With $z_i \in \mathbb{R}$, $\mathbf{p_i} \in \mathbb{R}^2$.

Definition of the related operator $\mathcal{C}_{g_{\mathbb{M}}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} \left[[z_i] \cap \{ g_{\mathbb{M}}(\mathbf{p}_i) \mid \mathbf{p}_i \in [\mathbf{p}_i] \} \right] \\ \bigsqcup \{ \mathbf{p}_i \in [\mathbf{p}_i] \mid g_{\mathbb{M}}(\mathbf{p}_i) \in [z_i] \} \end{pmatrix} \tag{1}$$



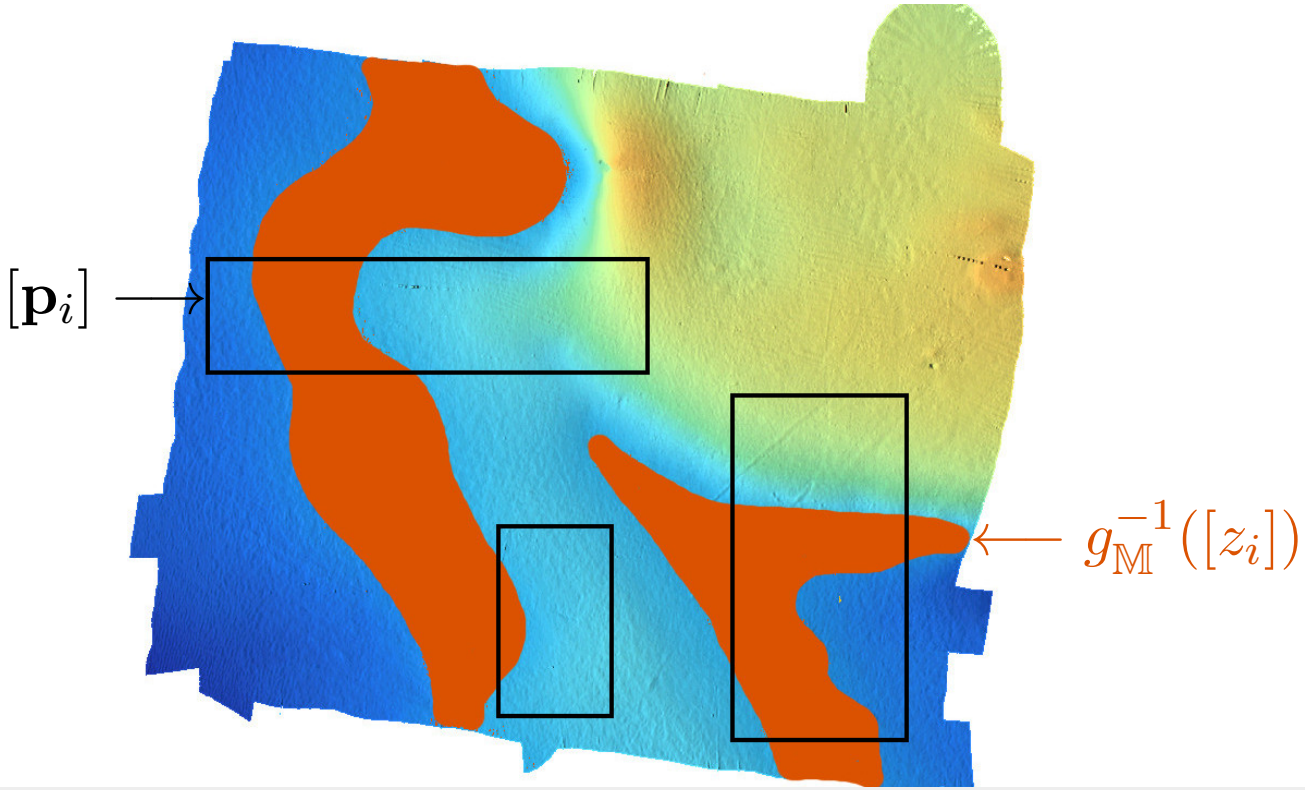
Constraint propagation method

The map constraint $\mathcal{L}_{g_{\mathbb{M}}}(z_i, \mathbf{p}_i) : z_i = g_{\mathbb{M}}(\mathbf{p}_i)$

With $z_i \in \mathbb{R}$, $\mathbf{p_i} \in \mathbb{R}^2$.

Definition of the related operator $\mathcal{C}_{g_{\mathbb{M}}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} \left[[z_i] \cap \{ g_{\mathbb{M}}(\mathbf{p}_i) \mid \mathbf{p}_i \in [\mathbf{p}_i] \} \right] \\ \bigsqcup \{ \mathbf{p}_i \in [\mathbf{p}_i] \mid g_{\mathbb{M}}(\mathbf{p}_i) \in [z_i] \} \end{pmatrix} \tag{1}$$



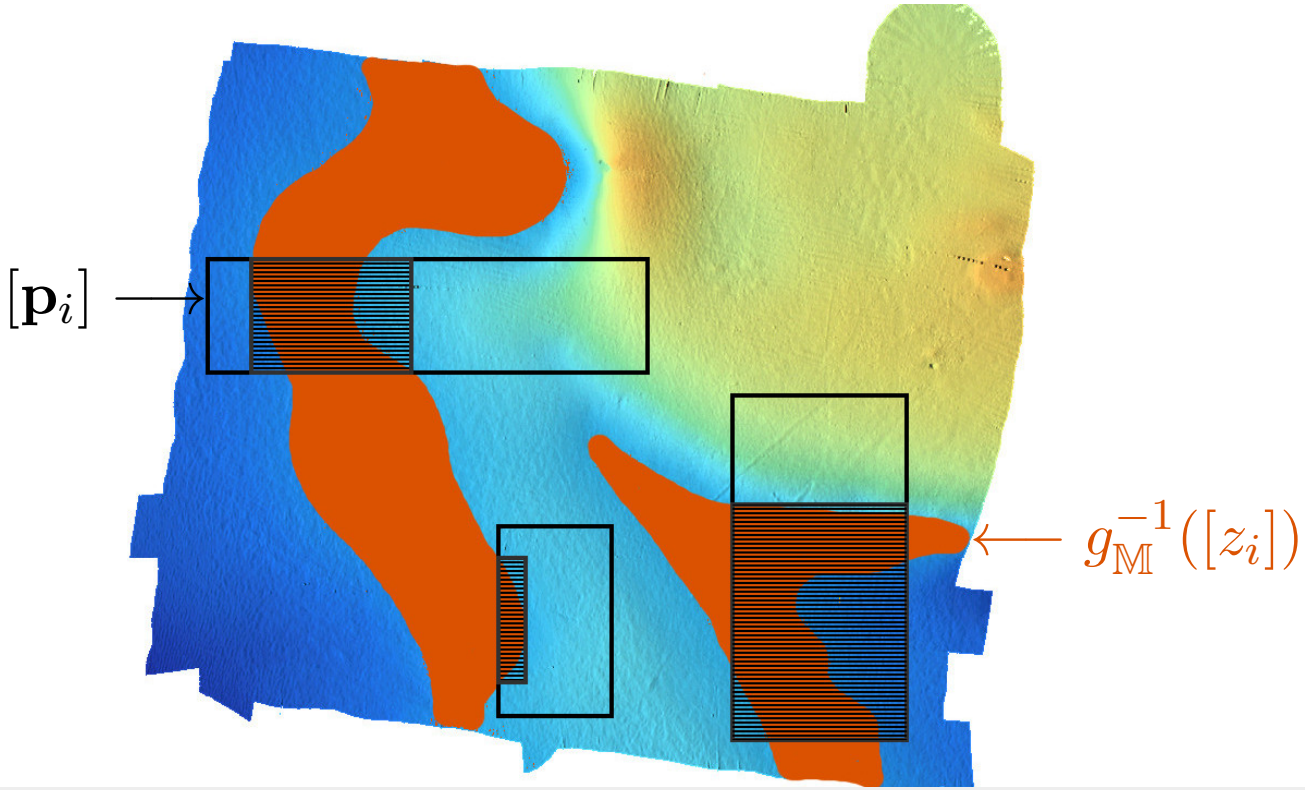
Constraint propagation method

The map constraint $\mathcal{L}_{g_{\mathbb{M}}}(z_i, \mathbf{p}_i) : z_i = g_{\mathbb{M}}(\mathbf{p}_i)$

With $z_i \in \mathbb{R}$, $\mathbf{p_i} \in \mathbb{R}^2$.

Definition of the related operator $\mathcal{C}_{g_{\mathbb{M}}}([z_i], [\mathbf{p}_i])$:

$$\begin{pmatrix} [z_i] \\ [\mathbf{p}_i] \end{pmatrix} \mapsto \begin{pmatrix} \left[[z_i] \cap \{ g_{\mathbb{M}}(\mathbf{p}_i) \mid \mathbf{p}_i \in [\mathbf{p}_i] \} \right] \\ \sqcup \{ \mathbf{p}_i \in [\mathbf{p}_i] \mid g_{\mathbb{M}}(\mathbf{p}_i) \in [z_i] \} \end{pmatrix} \tag{1}$$



Constraint propagation method

TBN: resolution method

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Contractor programming:

1. $\mathcal{C}_{\mathbf{f}}\big([\mathbf{x}](\cdot),[\mathbf{v}](\cdot)\big)$
2. $\mathcal{C}_{\frac{d}{dt}}\big([\mathbf{x}](\cdot),[\mathbf{v}](\cdot)\big)$
3. $\mathcal{C}_{\text{eval}}\big([t_i],[\mathbf{p}_i],[\mathbf{x}_{1,2}](\cdot)\big)$
4. $\mathcal{C}_{g_{\mathbb{M}}}\big([z_i],[\mathbf{p}_i]\big)$

Constraint propagation method

TBN: resolution method

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ z_i = g_{\mathbb{M}}\big(\mathbf{x}_{1,2}(t_i)\big) \end{cases}$$

Contractor programming:

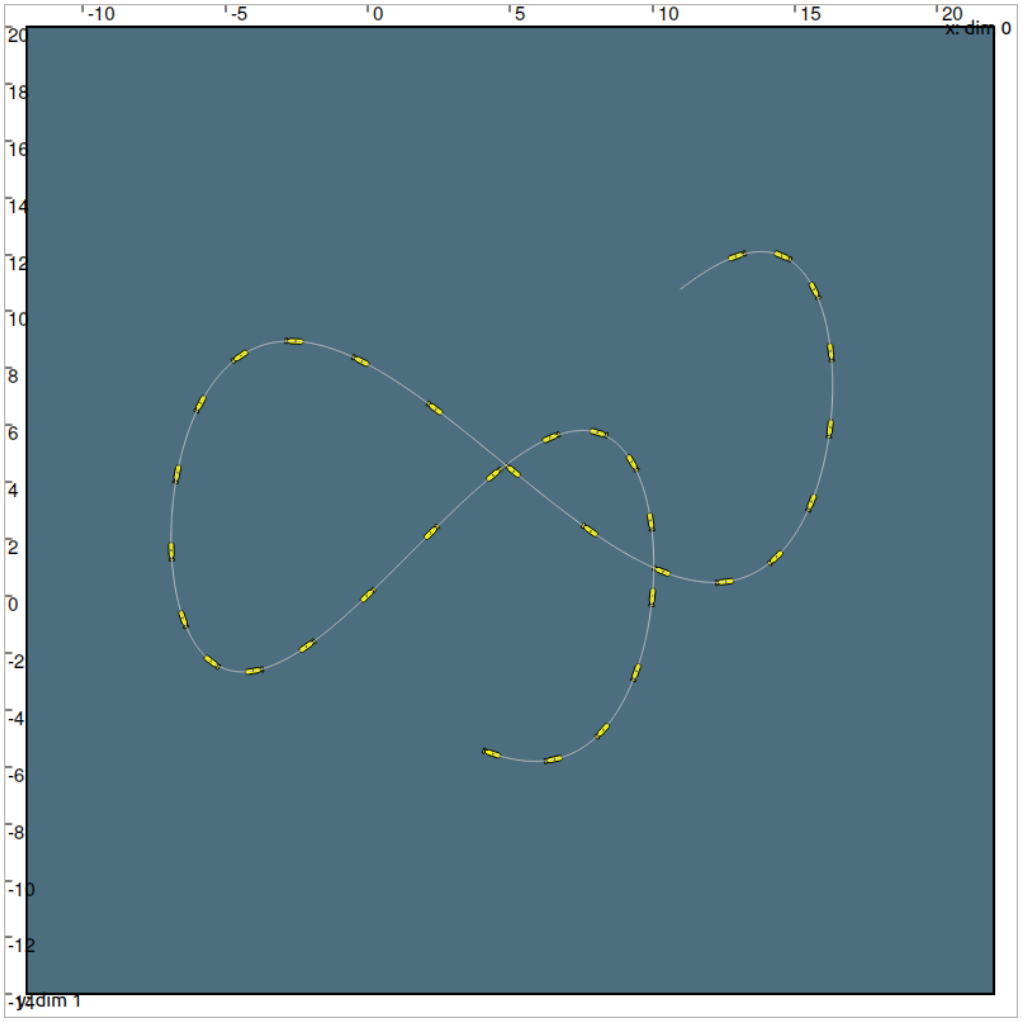
1. $\mathcal{C}_{\mathbf{f}}\big([\mathbf{x}](\cdot),[\mathbf{v}](\cdot)\big)$
2. $\mathcal{C}_{\frac{d}{dt}}\big([\mathbf{x}](\cdot),[\mathbf{v}](\cdot)\big)$
3. $\mathcal{C}_{\text{eval}}\big([t_i],[\mathbf{p}_i],[\mathbf{x}_{1,2}](\cdot)\big)$
4. $\mathcal{C}_{g_{\mathbb{M}}}\big([z_i],[\mathbf{p}_i]\big)$

Using Codac:

```
cn = ContractorNetwork()

cn.add(ctc.polar,
  [v[0],v[1],x[3],x[2]])                    #1
cn.add(ctc.deriv, [x,v])                    #2

for i in range(len(v_t)):
  pi = IntervalVector(4)
  cn.add(ctc.eval, [t[i],pi,x]              #3
  cn.add(ctc_map, [z[i],pi[0],pi[1]])       #4

cn.contract()                               -
```

Constraint propagation method

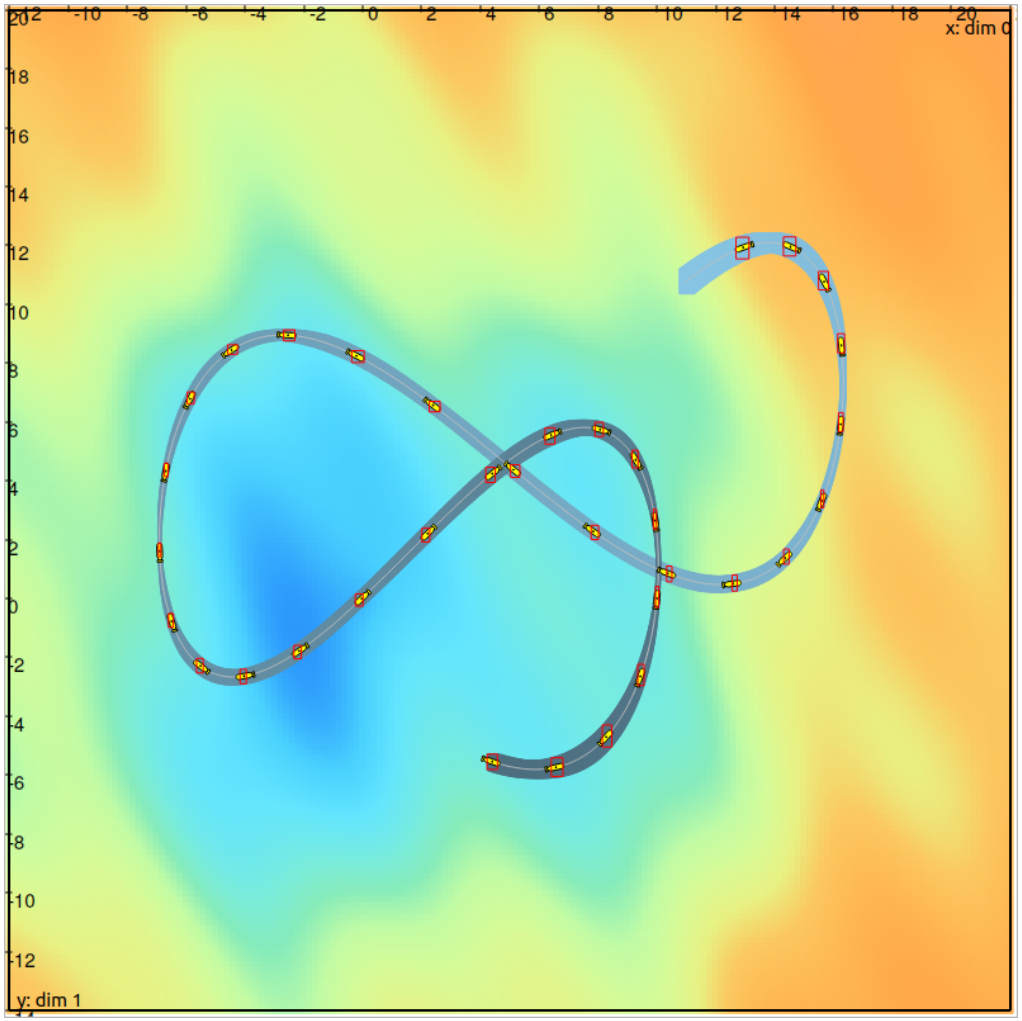
TBN: resolution method




```
cn = ContractorNetwork()

cn.add(ctc.polar, ..
cn.add(ctc.deriv, ..

for i in range(len(v_t)):
  pi = IntervalVector(4)
  cn.add(ctc.eval, ..
  cn.add(ctc_map, ..
```

Constraint propagation method

TBN: resolution method

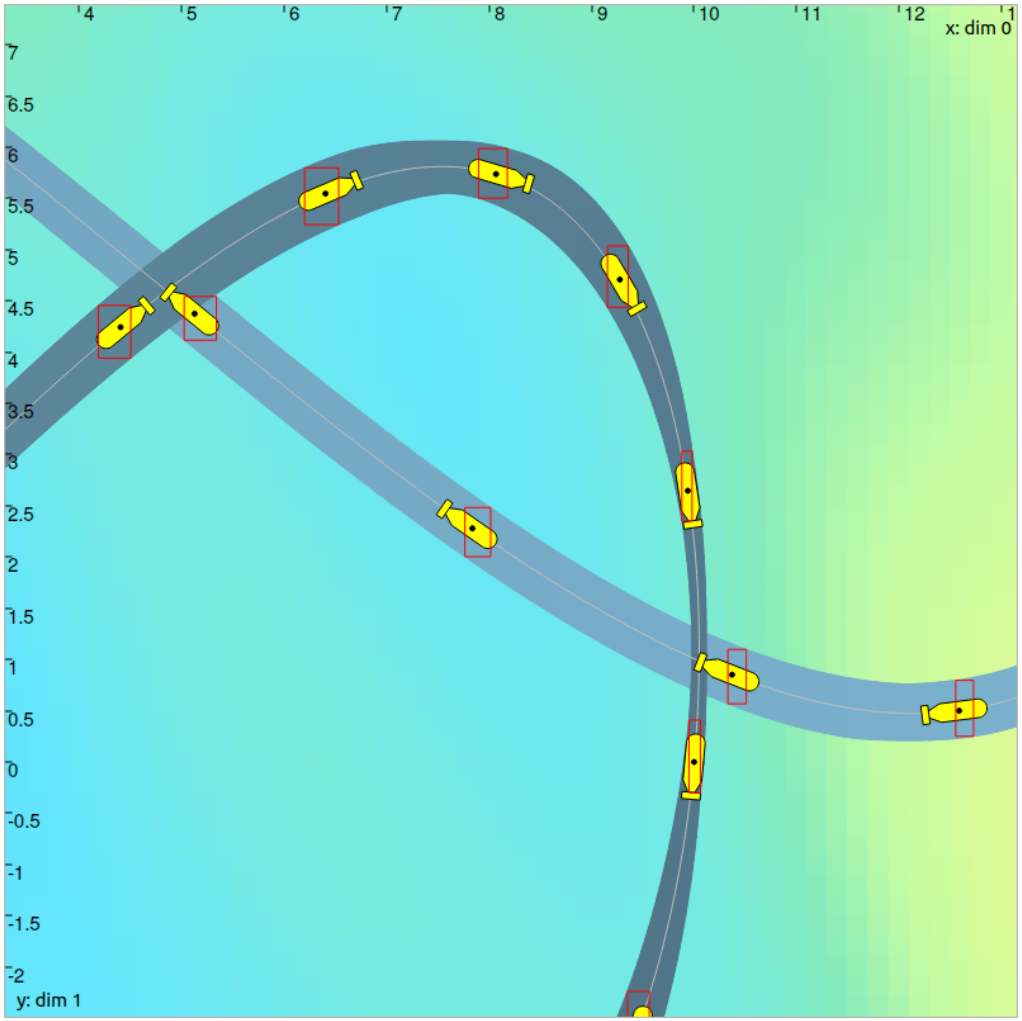



```
cn = ContractorNetwork()

cn.add(ctc.polar, ..
cn.add(ctc.deriv, ..

for i in range(len(v_t)):
  pi = IntervalVector(4)
  cn.add(ctc.eval, ..
  cn.add(ctc_map, ..

cn.contract()
```

Computation time: 10s

Constraint propagation method

TBN: resolution method




```
cn = ContractorNetwork()

cn.add(ctc.polar, ..
cn.add(ctc.deriv, ..

for i in range(len(v_t)):
  pi = IntervalVector(4)
  cn.add(ctc.eval, ..
  cn.add(ctc_map, ..

cn.contract()
```

Computation time: 10s

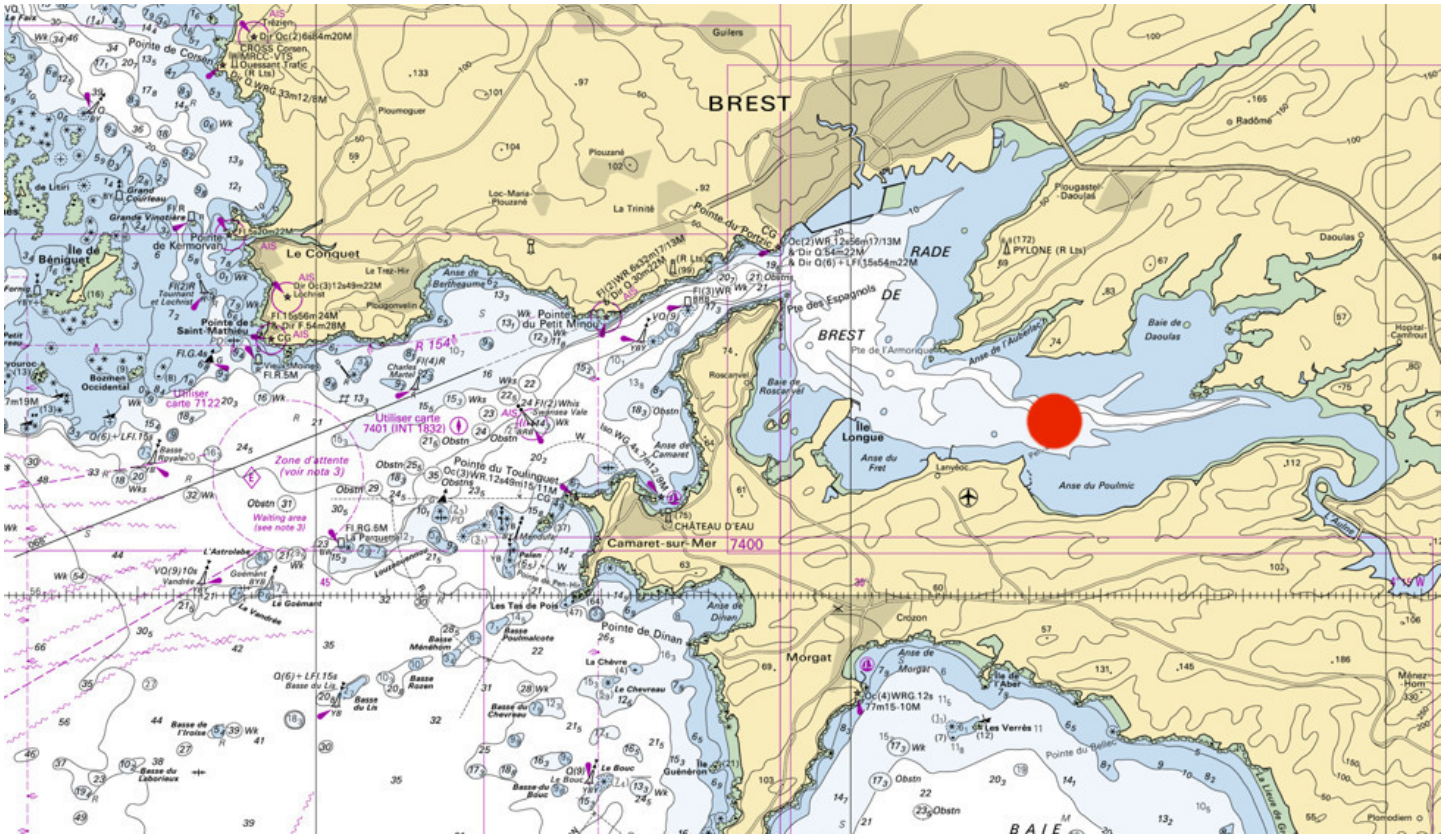
Section 5

Application

Application

Experimental mission with the Daurade AUV

- 2 hours experimental mission
- *Rade de Brest*, Brittany



Location: *Polygone de Rascas* – Credits: Shom

Application

Experimental mission with the Daurade AUV

- Daurade: Autonomous Underwater Vehicle (AUV)
- weight: 1010kg – length: 5m – max depth: 300m

Special thanks to DGA-TN Brest (formerly GESMA)

Application

Experimental mission with the Daurade AUV

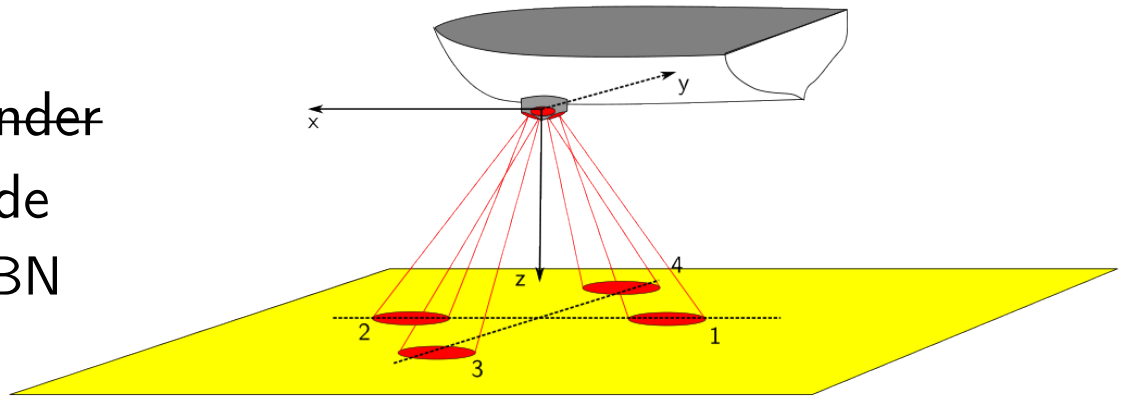
Mission data

Proprioceptive sensors:

- Inertial Navigation System (INS): Euler angles, accelerations
- Doppler Velocity Log (DVL): absolute velocities

Exteroceptive sensors:

- ~~single beam echo-sounder~~
- DVL $\rightarrow$ vehicle altitude
  - not suitable for TBN

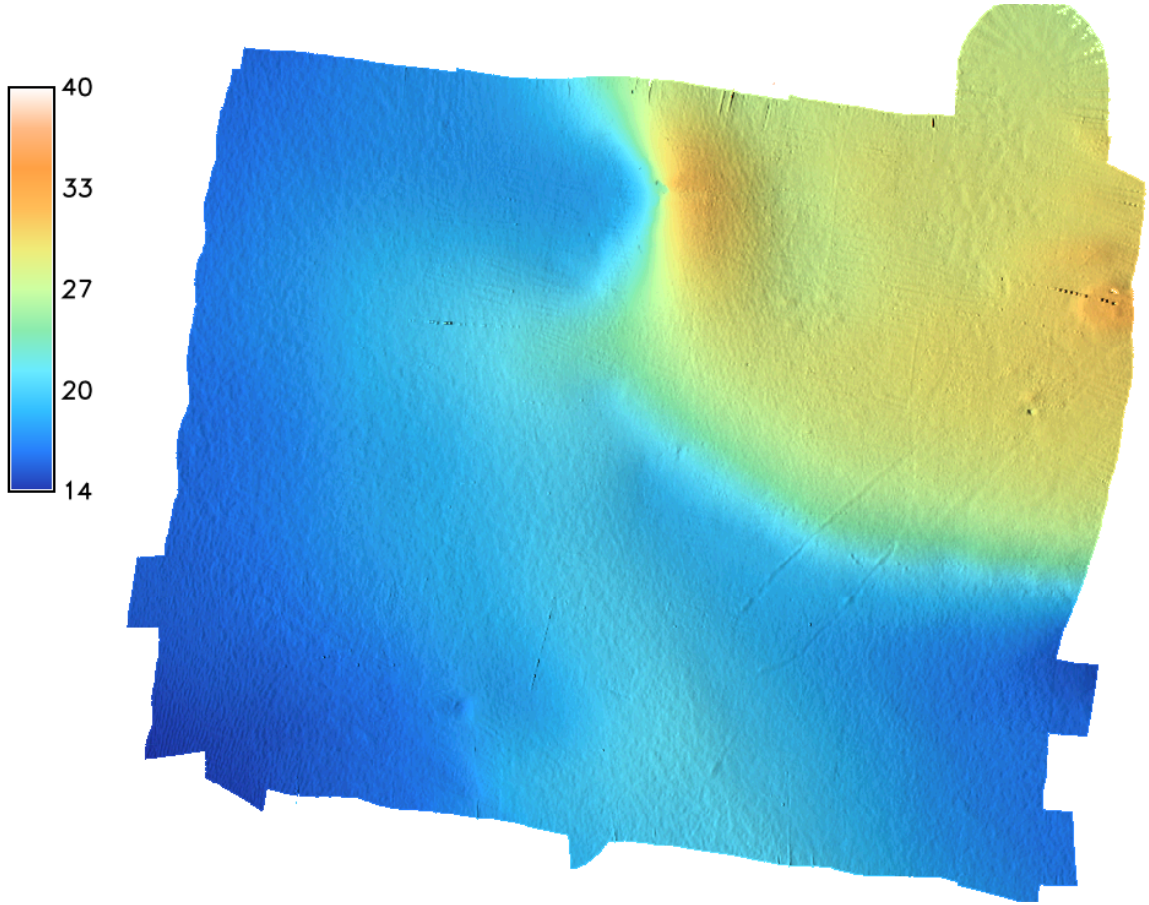


Double Janus DVL with 4 beams (image: Michel Legris)

Application

Map $\mathbb{M}$ of the environment

Prior knowledge: bathymetric map of Rascas, denoted $\mathbb{M}$

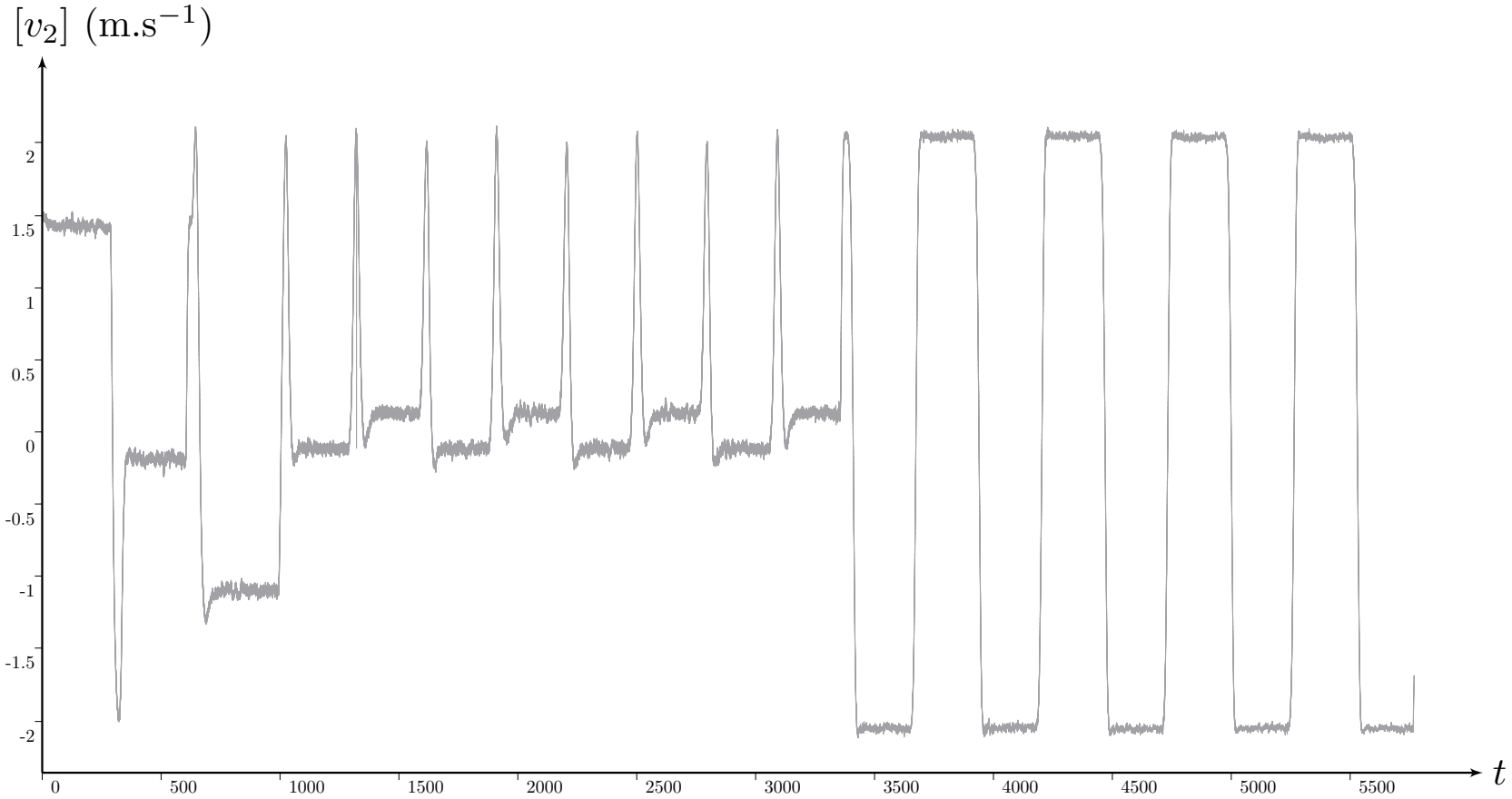


Thanks to Irène Mopin, Pierre Simon,
Romain Schwab, Michel Legris, Rodéric Moitié

Application

Evolution measurements

- velocity measurements obtained with a DVL
- considering uncertainties, building a tube $[\mathbf{v}](\cdot)$

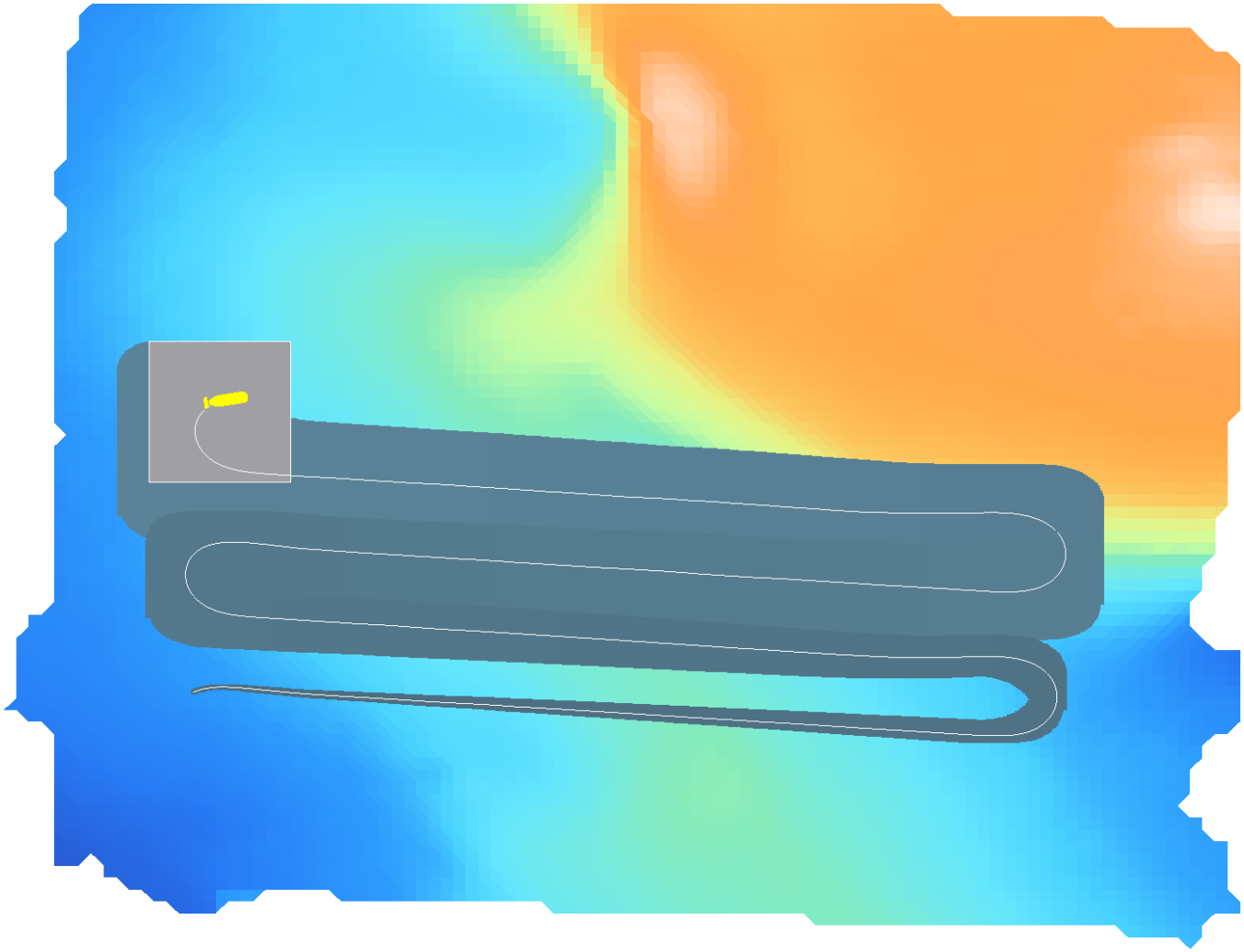


North speed velocity tube $[v_2](\cdot)$

Application

Dead reckoning from evolution measurements only

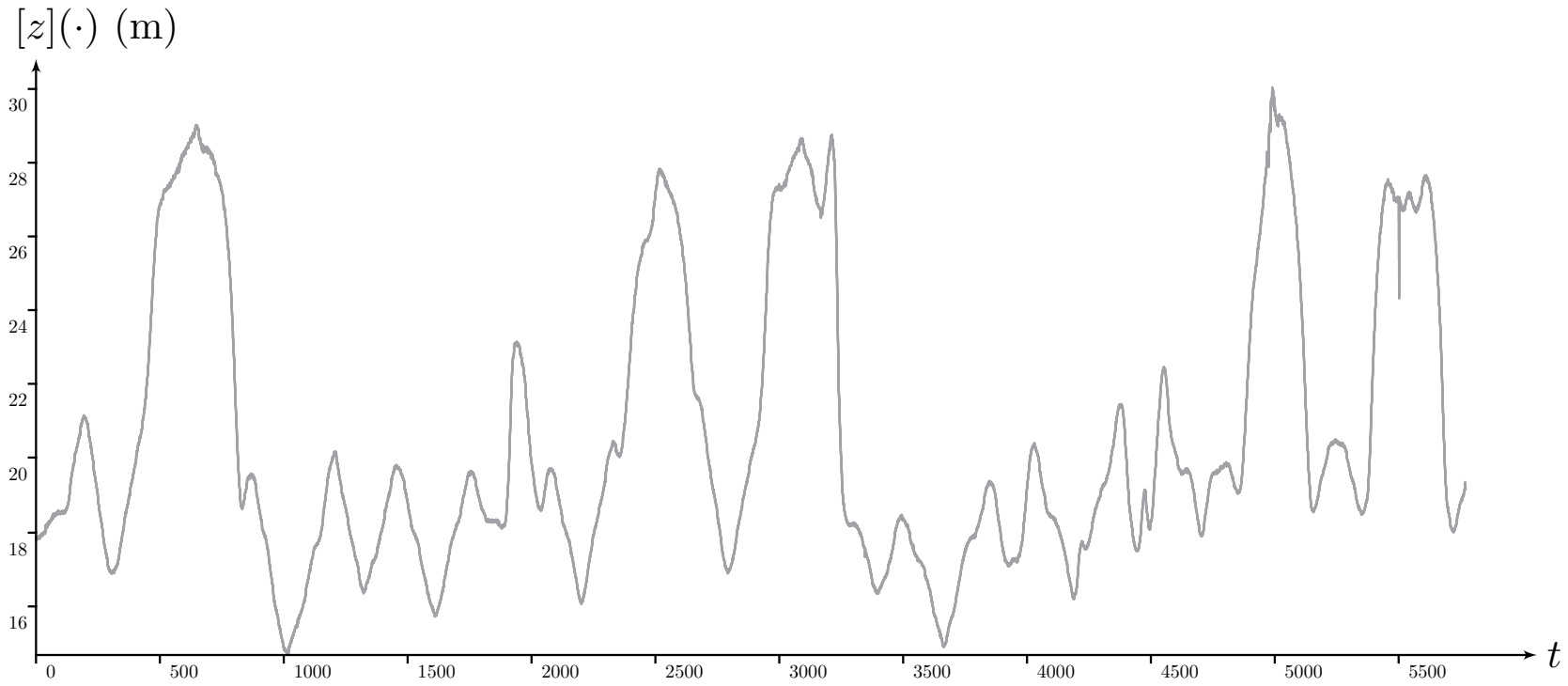
Video



Application

Observations measurements: bathymetric values

- DVL, same sensor, can provide **altitude measurements** z_{alt}
- pressure sensor: depth values z_{depth}
- time-dependent measurements, use of **tide models**
- $z = z_{\text{alt}} + z_{\text{depth}} + z_{\text{tide}}$

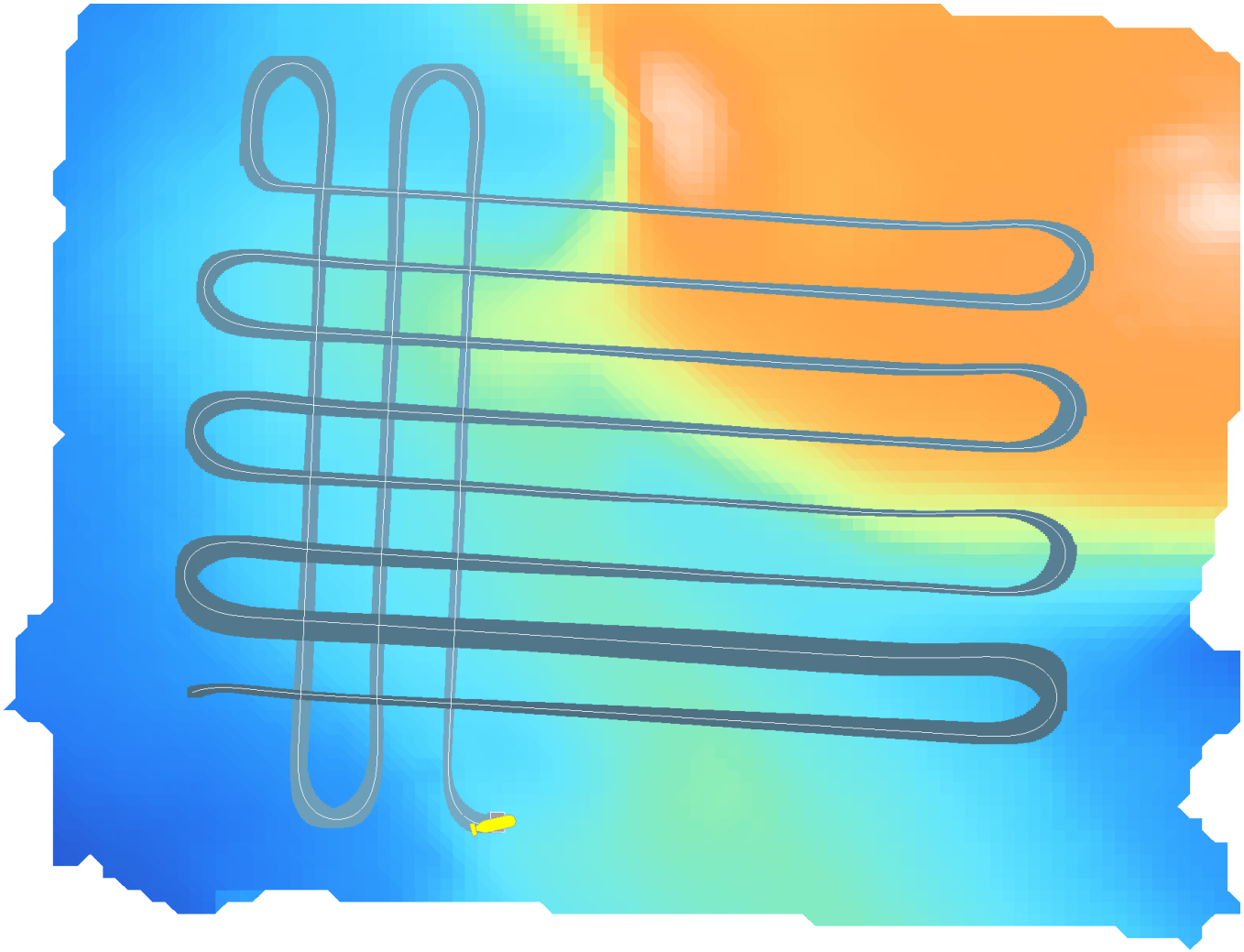


Observation tube $[z](\cdot)$

Application

Full example of TBN

Video



Section 6

Conclusions

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems
- focus on **the *what* instead of the *how***
  no expertise required on how to solve a problem

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems
- focus on **the *what* instead of the *how***
  no expertise required on how to solve a problem
- **complex systems** easily handled
  non-linearities, differential equations, values from datasets

Conclusions

Assets of constraint programming

- **simplicity** of the approach
  transparent application of contractors on elementary constraints
- **reliability** of the results: no solution can be lost
  useful for proof purposes and the safety of systems
- focus on **the *what* instead of the *how***
  no expertise required on how to solve a problem
- **complex systems** easily handled
  non-linearities, differential equations, values from datasets

Codac library: open-source C++/Python library providing tools for constraint programming over reals, trajectories and sets

http://www.codac.io