Constraint Programming (CP) coupled with Interval Analysis (IA) for mobile robotics

(a very short introduction)

Simon Rohou, Quentin Brateau

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

Journées Nationales de la Recherche en Robotique (JNRR 2023)
Moliets 16–20th October 2023

Planning de cet après-midi

- **14h00 – 15h00**
  Présentation
- **15h00 – 16h00**
  TD intervals .. static range-only
- **16h00 – 16h30**
  Pause café
- **16h30 – 17h00**
  Présentation
- **17h00 – 18h00**
  TD dynamic localization .. SLAM

Section 1

Introduction

Mobile robotics

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

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

Where:

- $\mathbf{x} \in \mathbb{R}^n$ is the state vector (position, bearing, ...)
- $\mathbf{u} \in \mathbb{R}^m$ is the input vector (command)
- $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the *evolution* function

Mobile robotics

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

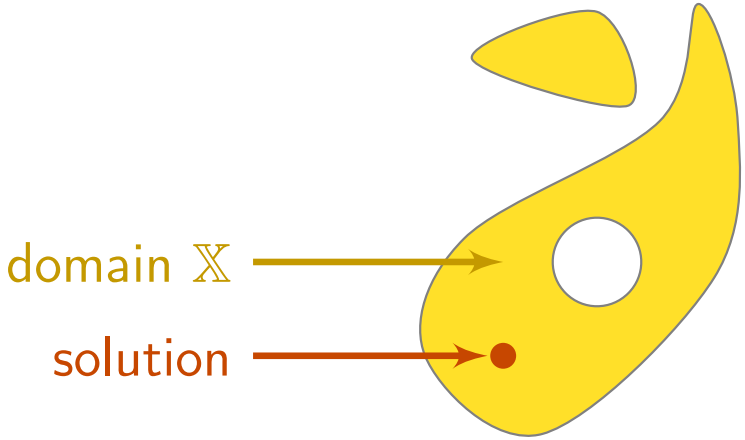
$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(evolution)} \\ \mathbf{z}(t) = \mathbf{g}\big(\mathbf{x}(t)\big) & \text{(observation)} \end{cases}$$

Where:

- $\mathbf{x} \in \mathbb{R}^n$ is the state vector (position, bearing, ...)
- $\mathbf{u} \in \mathbb{R}^m$ is the input vector (command)
- $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the *evolution* function
- $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^p$ is the *observation* function
- $\mathbf{z} \in \mathbb{R}^p$ is some exteroceptive measurement (camera, sonar...)

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$



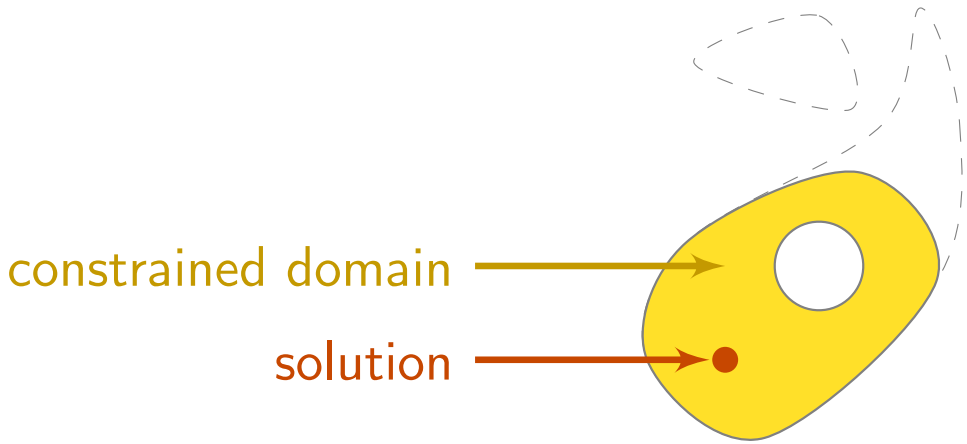
Constraint network:

$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \\ \textbf{Domains: } \mathbb{X} \end{cases}$$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



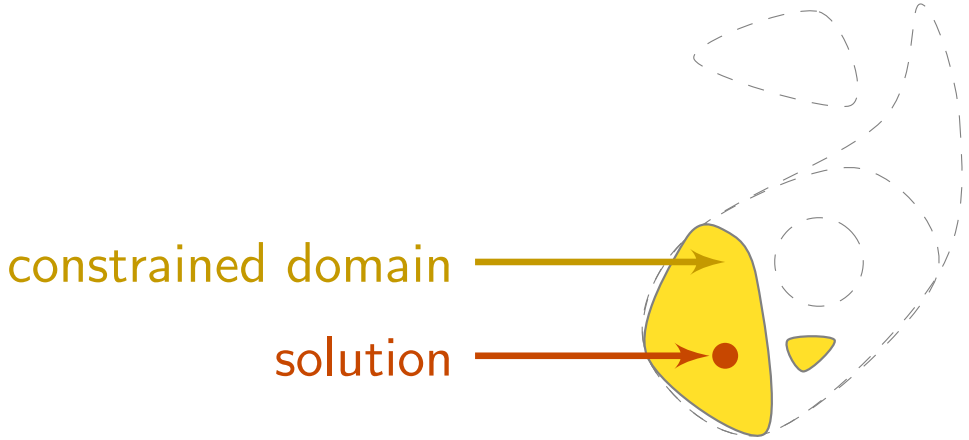
Constraint network:

$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_1(\mathbf{x}) \\ \\ \textbf{Domains: } \mathbb{X} \end{cases}$$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Constraints:

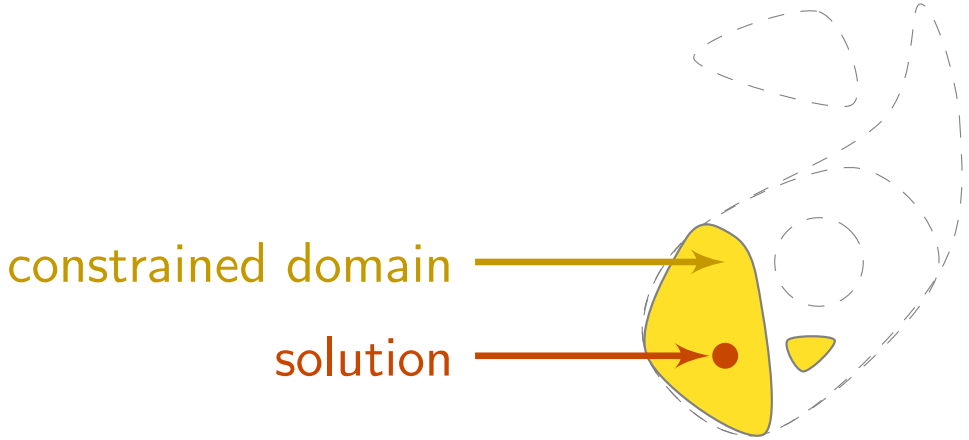
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$

Domains: $\mathbb{X}$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Constraints:

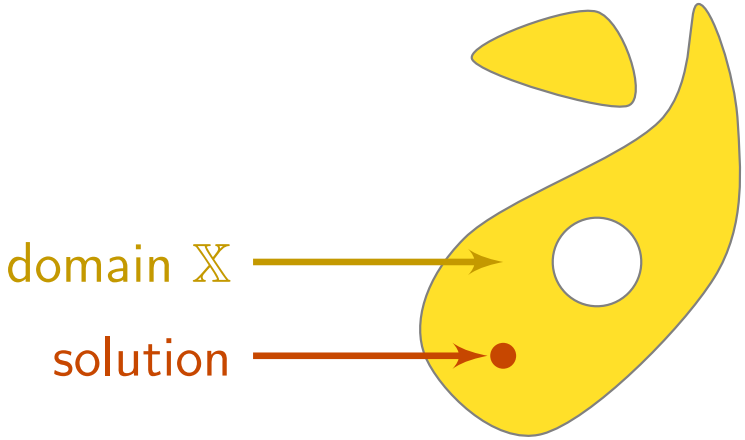
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Domains: $\mathbb{X}$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...



Constraint network:

Variables: $\mathbf{x}$

Constraints:

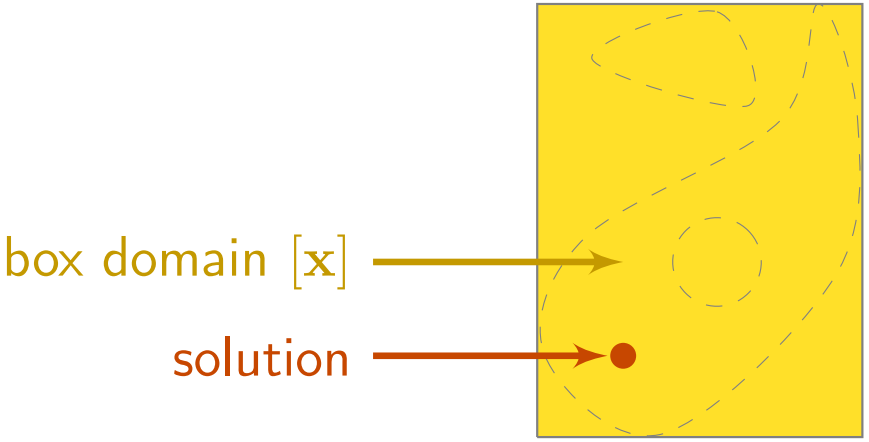
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Domains: $\mathbb{X}$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$



Constraint network:

Variables: $\mathbf{x}$

Constraints:

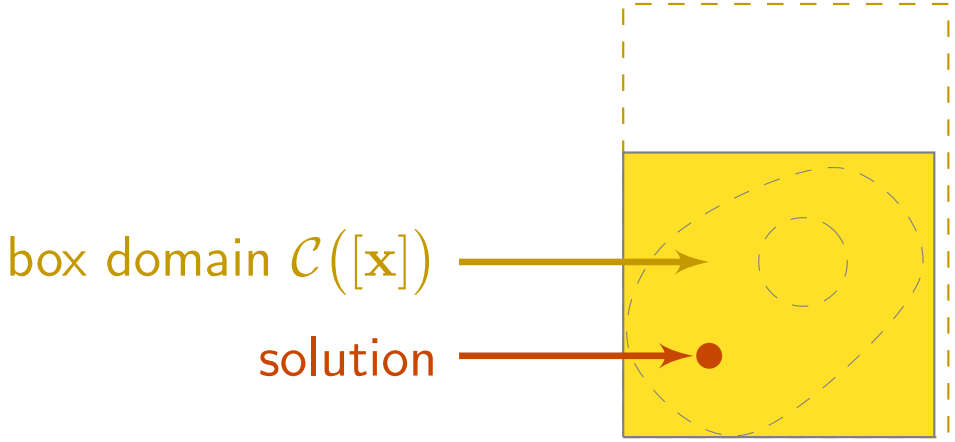
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Domains: $[\mathbf{x}]$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

Variables: $\mathbf{x}$

Constraints:

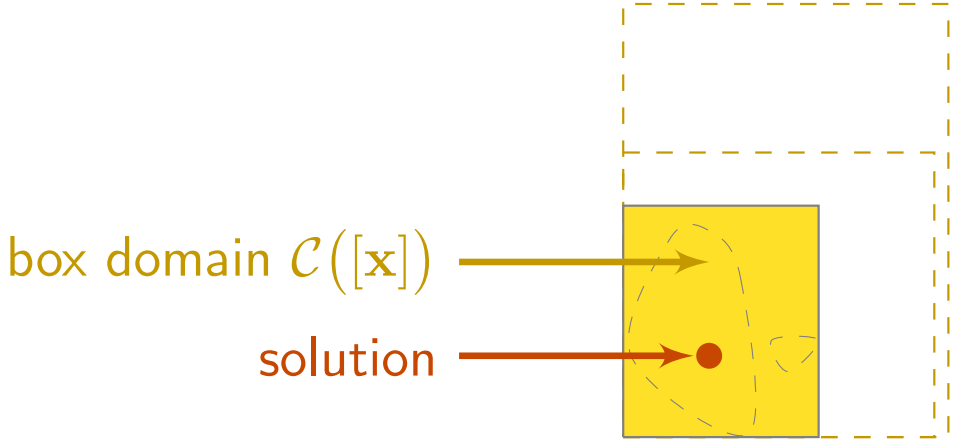
1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Domains: $[\mathbf{x}]$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

Variables: $\mathbf{x}$

Constraints:

1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Domains: $[\mathbf{x}]$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Section 2

Interval analysis

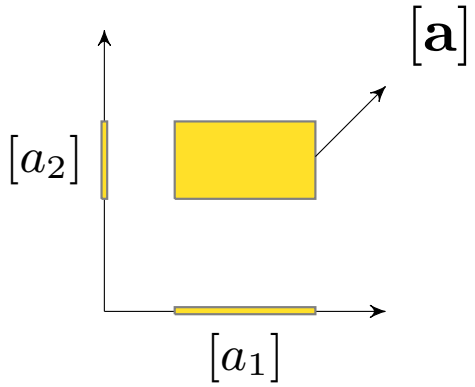
What is an interval?

An interval $[x]$:

- a closed and connected subset of $\mathbb{R}$ delimited by two bounds
- $[x] = [x^-, x^+] = \{x \in \mathbb{R} \mid x^- \leqslant x \leqslant x^+\}$
- $[x] \in \mathbb{IR}$

A box $[\mathbf{x}]$ (an interval vector):

- a cartesian product of n intervals
- $[\mathbf{x}] \in \mathbb{IR}^n$



a box $[\mathbf{a}] \in \mathbb{IR}^2$

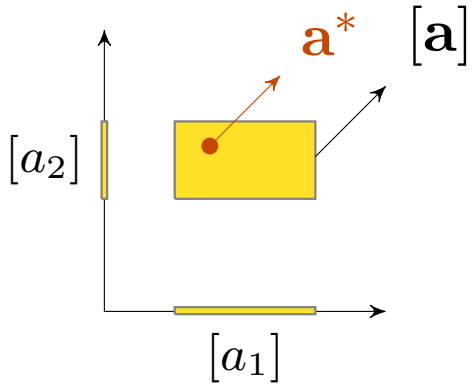
What is an interval?

An interval $[x]$:

- a closed and connected subset of $\mathbb{R}$ delimited by two bounds
- $[x] = [x^-, x^+] = \{x \in \mathbb{R} \mid x^- \leqslant x \leqslant x^+\}$
- $[x] \in \mathbb{IR}$

A box $[\mathbf{x}]$ (an interval vector):

- a cartesian product of n intervals
- $[\mathbf{x}] \in \mathbb{IR}^n$



a box $[\mathbf{a}] \in \mathbb{IR}^2$

Interval Analysis

IA is based on the extension of all classical **real arithmetic operators**:

- $+, -, \times, \div$

  ex: $[x] + [y] = [x^- + y^-, x^+ + y^+]$

  ex: $[x] - [y] = [x^- - y^+, x^+ - y^-]$

Interval Analysis

IA is based on the extension of all classical **real arithmetic operators**:

- $+$, $-$, $\times$, $\div$
  ex: $[x] + [y] = [x^- + y^-, x^+ + y^+]$
  ex: $[x] - [y] = [x^- - y^+, x^+ - y^-]$

Adaptation of **elementary functions** such as:

- $\cos$, $\exp$, $\tan$, etc.
- output is the smallest interval containing all the images of all defined inputs through the function

Example: $\exp([x])$

Example of a **forward evaluation** of $\exp([x])$:

Natural inclusion functions

Example of the previous function g.
Let us compute the **inclusion function** of the distance function g:
(distance between a position $\mathbf{x}$ and a landmark $\mathbf{b}$)

$$\begin{array}{rcl} g : \mathbb{R}^2 & \rightarrow & \mathbb{R}, \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \mapsto & \sqrt{(x_1 - b_1)^2 + (x_2 - b_2)^2}. \end{array} \tag{1}$$

Replacing items by their interval counterpart, $[g]$ is given by:

$$\begin{array}{rcl} [g] : \mathbb{IR}^2 & \rightarrow & \mathbb{IR}, \\ \begin{pmatrix} [x_1] \\ [x_2] \end{pmatrix} & \mapsto & \sqrt{([x_1] - [b_1])^2 + ([x_2] - [b_2])^2}. \end{array} \tag{2}$$

Forward/Backward with interval analysis

Information can be propagated in a forward and a backward way in the equation.

Recall the example of $\exp$:

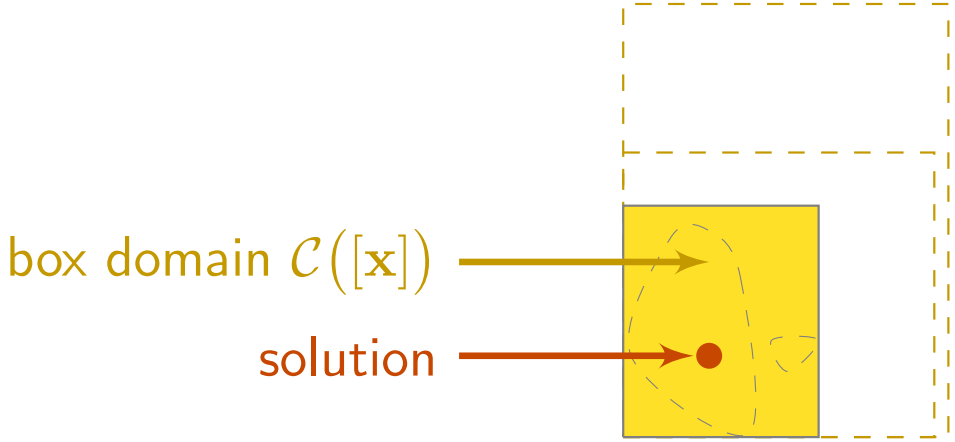
- $y = \exp(x)$
- $x \in [x]$, $y \in [y]$

Section 3

Constraint Propagation (CP) coupled with Interval Analysis(IA)

Constraint programming: overall concept

- system described by a *constraint network*
- **variables** belonging to **domains** $\mathbb{X}$
- continuous **constraints** $\mathcal{L}$: non-linear equations, inequalities, ...
- representable domains: *e.g.* boxes $[\mathbf{x}]$
- resolution by **contractors**, $\mathcal{C}_{\mathcal{L}}([\mathbf{x}])$



Constraint network:

Variables: $\mathbf{x}$

Constraints:

1. $\mathcal{L}_1(\mathbf{x})$
2. $\mathcal{L}_2(\mathbf{x})$
3. ...

Domains: $[\mathbf{x}]$

Contractor Programming
Chabert, Jaulin *Artifical Intelligence*, 2009

Contractors

Considering a constraint (*e.g.*, an equation) $\mathcal{L}$,

Definition

A contractor on a box is an operator $\mathcal{C}_{\mathcal{L}}$ from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that:

$$(i) \quad \forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}_{\mathcal{L}}([\mathbf{x}]) \subseteq [\mathbf{x}], \qquad \text{(contraction)}$$

$$(ii) \quad \begin{pmatrix} \mathcal{L}(\mathbf{x}) \\ \mathbf{x} \in [\mathbf{x}] \end{pmatrix} \Longrightarrow \mathbf{x} \in \mathcal{C}_{\mathcal{L}}([\mathbf{x}]). \qquad \text{(consistency)}$$

Contractors

Considering a constraint (*e.g.*, an equation) $\mathcal{L}$,

Definition

A contractor on a box is an operator $\mathcal{C}_{\mathcal{L}}$ from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that:

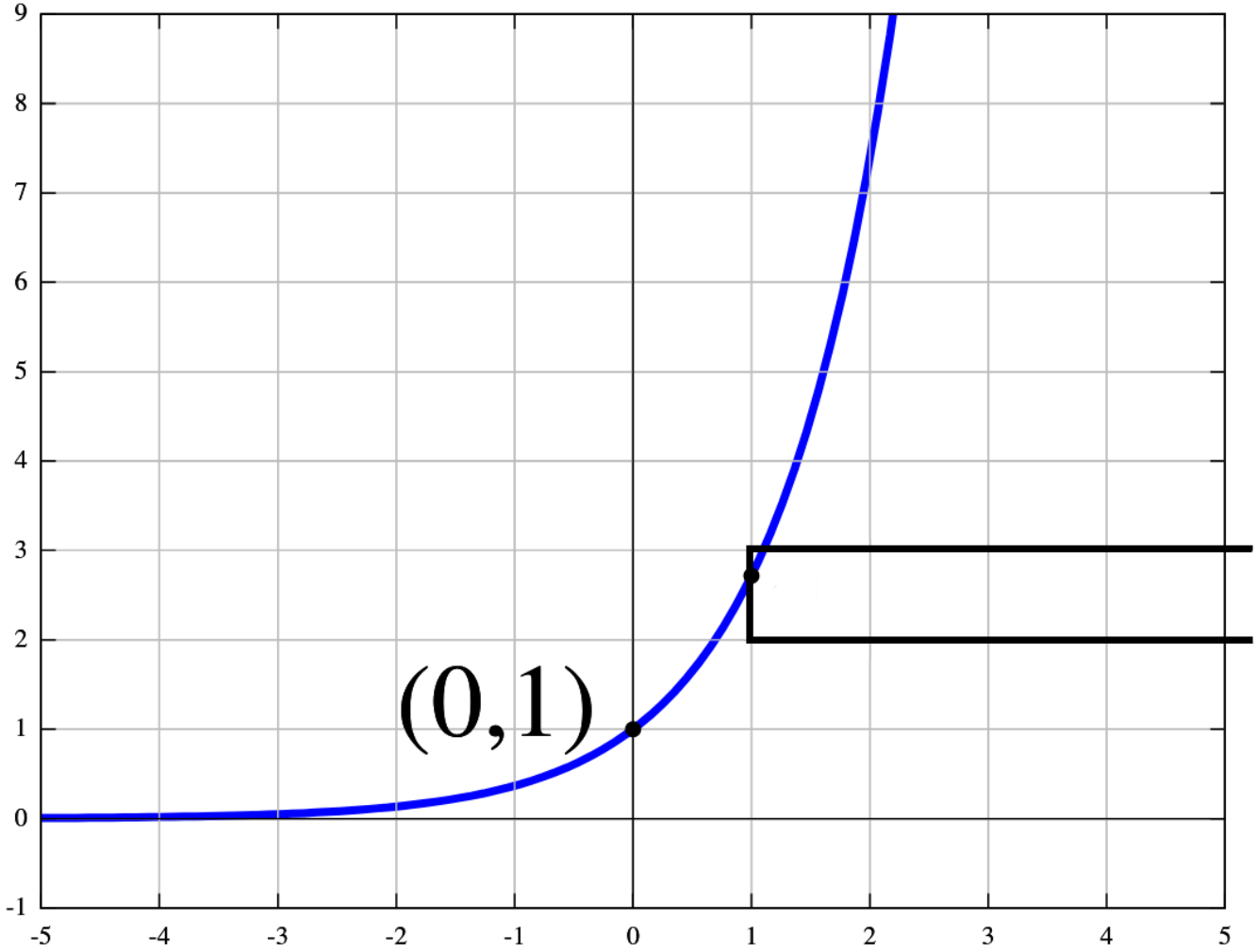
$$(i) \quad \forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}_{\mathcal{L}}([\mathbf{x}]) \subseteq [\mathbf{x}], \qquad \text{(contraction)}$$

$$(ii) \quad \begin{pmatrix} \mathcal{L}(\mathbf{x}) \\ \mathbf{x} \in [\mathbf{x}] \end{pmatrix} \Longrightarrow \mathbf{x} \in \mathcal{C}_{\mathcal{L}}([\mathbf{x}]). \qquad \text{(consistency)}$$

Example:

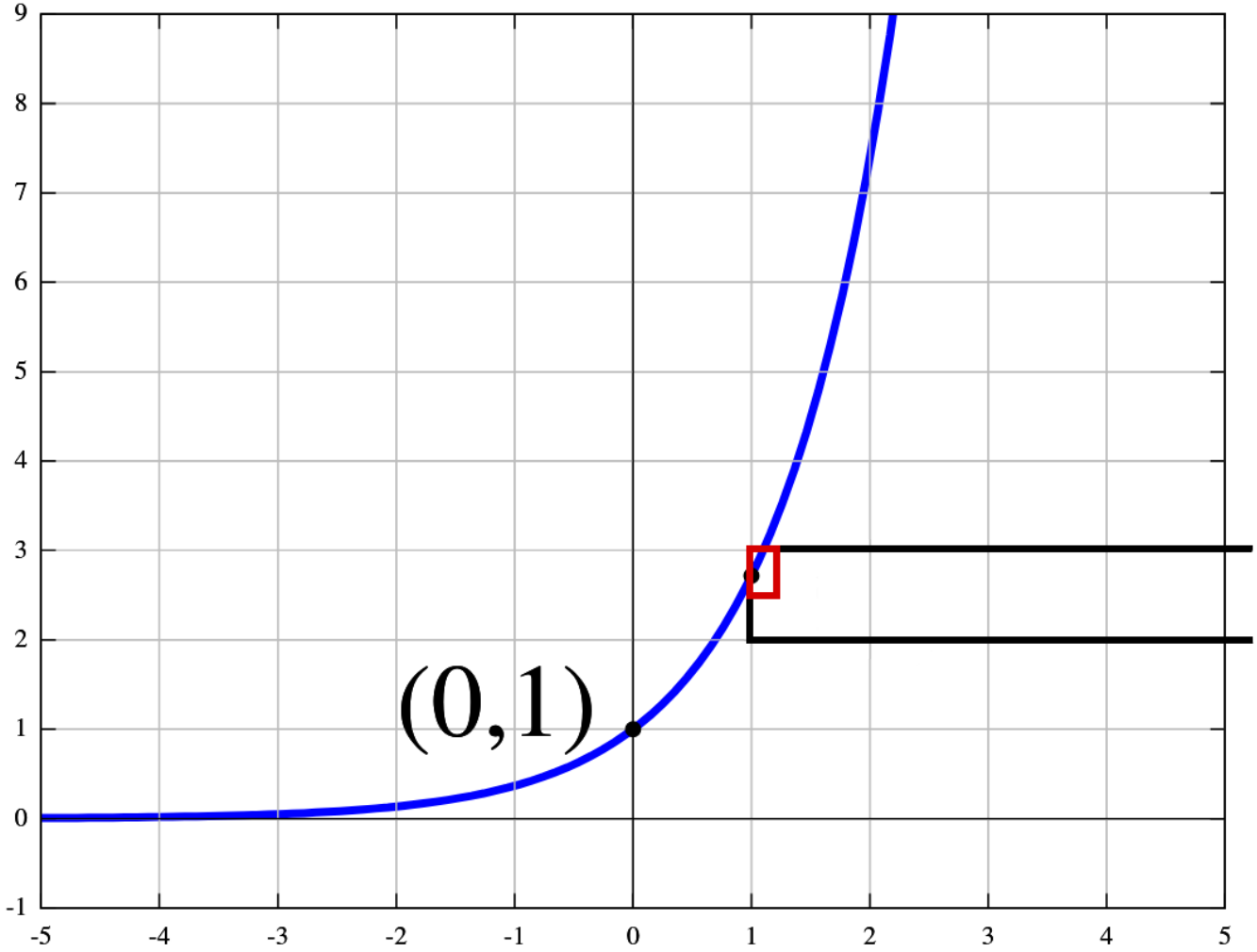
- $[\mathbf{x}] = [1, \infty] \times [2, 3]$
- $\mathcal{C}_{\exp}([\mathbf{x}])$ associated with $\exp(x_1) - x_2 = 0$
- after applying $\mathcal{C}_{\exp}$, $[\mathbf{x}] = [1, 1.099] \times [2.72, 3]$

Contractors



Black: initial box $[1, \infty] \times [2, 3]$.

Contractors



Black: initial box $[1, \infty] \times [2, 3]$. Red: contracted box.

Example of elementary contractors

Example 1: consider the constraint $a(\cdot) = x(\cdot) + y(\cdot)$
A minimal **contractor** to apply this constraint is:

$$\begin{pmatrix} [a] \\ [x] \\ [y] \end{pmatrix} \xmapsto{\mathcal{C}_+} \begin{pmatrix} [a] \cap ([x] + [y]) \\ [x] \cap ([a] - [y]) \\ [y] \cap ([a] - [x]) \end{pmatrix}$$

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

Example of elementary contractors

Example 1: consider the constraint $a(\cdot) = x(\cdot) + y(\cdot)$
A minimal **contractor** to apply this constraint is:

$$\begin{pmatrix} [a] \\ [x] \\ [y] \end{pmatrix} \xmapsto{\mathcal{C}_+} \begin{pmatrix} [a] \cap ([x] + [y]) \\ [x] \cap ([a] - [y]) \\ [y] \cap ([a] - [x]) \end{pmatrix}$$

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

Example 2: consider the constraint $y - \exp(x) = 0$
A **contractor** to apply this constraint is:

$$\begin{pmatrix} [x] \\ [y] \end{pmatrix} \xmapsto{\mathcal{C}_{\exp}} \begin{pmatrix} [x] \cap \log([y]) \\ [y] \cap \exp([x]) \end{pmatrix}$$

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

Example of composition of contractors

Example: **decomposition** of the observation constraint:

$$\mathcal{L}_{\text{dist}} : \rho = \sqrt{(x_1 - b_1)^2 + (x_2 - b_2)^2}$$

Example of composition of contractors

Example: **decomposition** of the observation constraint:

$$\mathcal{L}_{\text{dist}} : \rho = \sqrt{(x_1 - b_1)^2 + (x_2 - b_2)^2}$$

$$\Leftrightarrow \begin{cases} a & = & x_1 - b_1 \\ b & = & x_2 - b_2 \\ c & = & a^2 \\ d & = & b^2 \\ e & = & c + d \\ \rho & = & \sqrt{e} \end{cases}$$

Example of composition of contractors

Example: **decomposition** of the observation constraint:

$$\mathcal{L}_{\text{dist}} : \rho = \sqrt{(x_1 - b_1)^2 + (x_2 - b_2)^2}$$

$$\Leftrightarrow \begin{cases} a & = & x_1 - b_1 \\ b & = & x_2 - b_2 \\ c & = & a^2 \\ d & = & b^2 \\ e & = & c + d \\ \rho & = & \sqrt{e} \end{cases}$$

$$\begin{pmatrix} [e] \\ [c] \\ [d] \end{pmatrix} \xrightarrow{\mathcal{C}_+} \begin{pmatrix} [e] \cap ([c] + [d]) \\ [c] \cap ([e] - [d]) \\ [d] \cap ([e] - [c]) \end{pmatrix}$$

Example of composition of contractors

Example: **decomposition** of the observation constraint:

$$\mathcal{L}_{\text{dist}} : \rho = \sqrt{(x_1 - b_1)^2 + (x_2 - b_2)^2}$$

$$\Leftrightarrow \begin{cases} a & = & x_1 - b_1 & \mathcal{C}_-([a],[x_1],[b_1]) \\ b & = & x_2 - b_2 & \mathcal{C}_-([b],[x_2],[b_2]) \\ c & = & a^2 & \mathcal{C}_{.2}([c],[a]) \\ d & = & b^2 & \mathcal{C}_{.2}([d],[b]) \\ e & = & c + d & \mathcal{C}_+([e],[c],[d]) \\ \rho & = & \sqrt{e} & \mathcal{C}_{\sqrt{.}}([\rho],[e]) \end{cases}$$

$$\begin{pmatrix} [e] \\ [c] \\ [d] \end{pmatrix} \xrightarrow{\mathcal{C}_+} \begin{pmatrix} [e] \cap ([c] + [d]) \\ [c] \cap ([e] - [d]) \\ [d] \cap ([e] - [c]) \end{pmatrix}$$

Example of composition of contractors

Example: **decomposition** of the observation constraint:

$$\mathcal{L}_{\text{dist}} : \rho = \sqrt{(x_1 - b_1)^2 + (x_2 - b_2)^2}$$

$$\Leftrightarrow \left\{ \begin{array}{lcll} a & = & x_1 - b_1 & \mathcal{C}_{-}([a],[x_1],[b_1]) \\ b & = & x_2 - b_2 & \mathcal{C}_{-}([b],[x_2],[b_2]) \\ c & = & a^2 & \mathcal{C}_{.2}([c],[a]) \\ d & = & b^2 & \mathcal{C}_{.2}([d],[b]) \\ e & = & c + d & \mathcal{C}_{+}([e],[c],[d]) \\ \rho & = & \sqrt{e} & \mathcal{C}_{\sqrt{.}}([\rho],[e]) \end{array} \right\} \Longrightarrow \mathcal{C}_{\text{dist}}$$

$$\begin{pmatrix} [e] \\ [c] \\ [d] \end{pmatrix} \xrightarrow{\mathcal{C}_{+}} \begin{pmatrix} [e] \cap ([c] + [d]) \\ [c] \cap ([e] - [d]) \\ [d] \cap ([e] - [c]) \end{pmatrix}$$

Example: the $\mathcal{C}_{\text{dist}}$ contractor

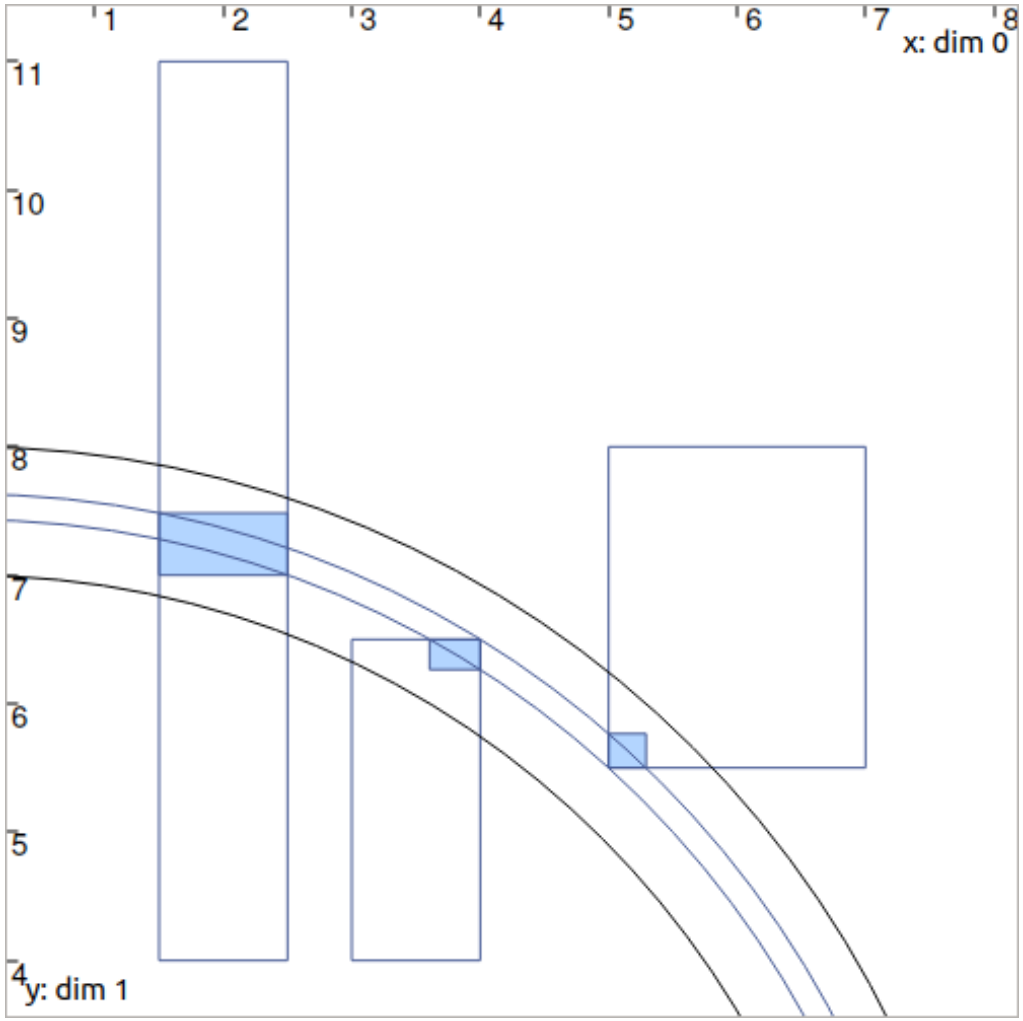


Illustration of several contracted boxes with $\mathcal{C}_{\text{dist}}$ contractor. The blue boxes have been contracted as well as the ring.

Uncertainties as sets

Example of **range-only** robot localization (three beacons):

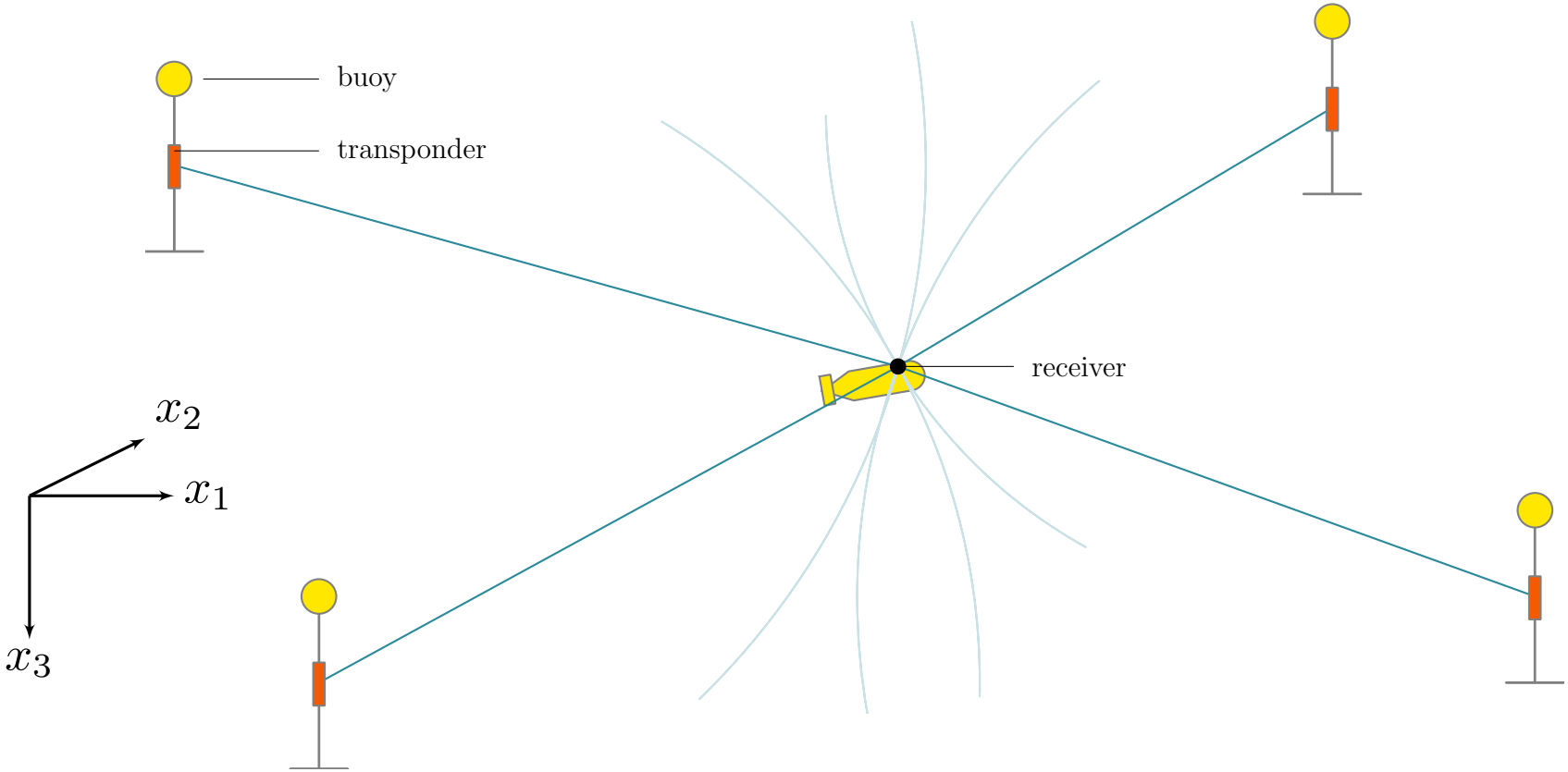
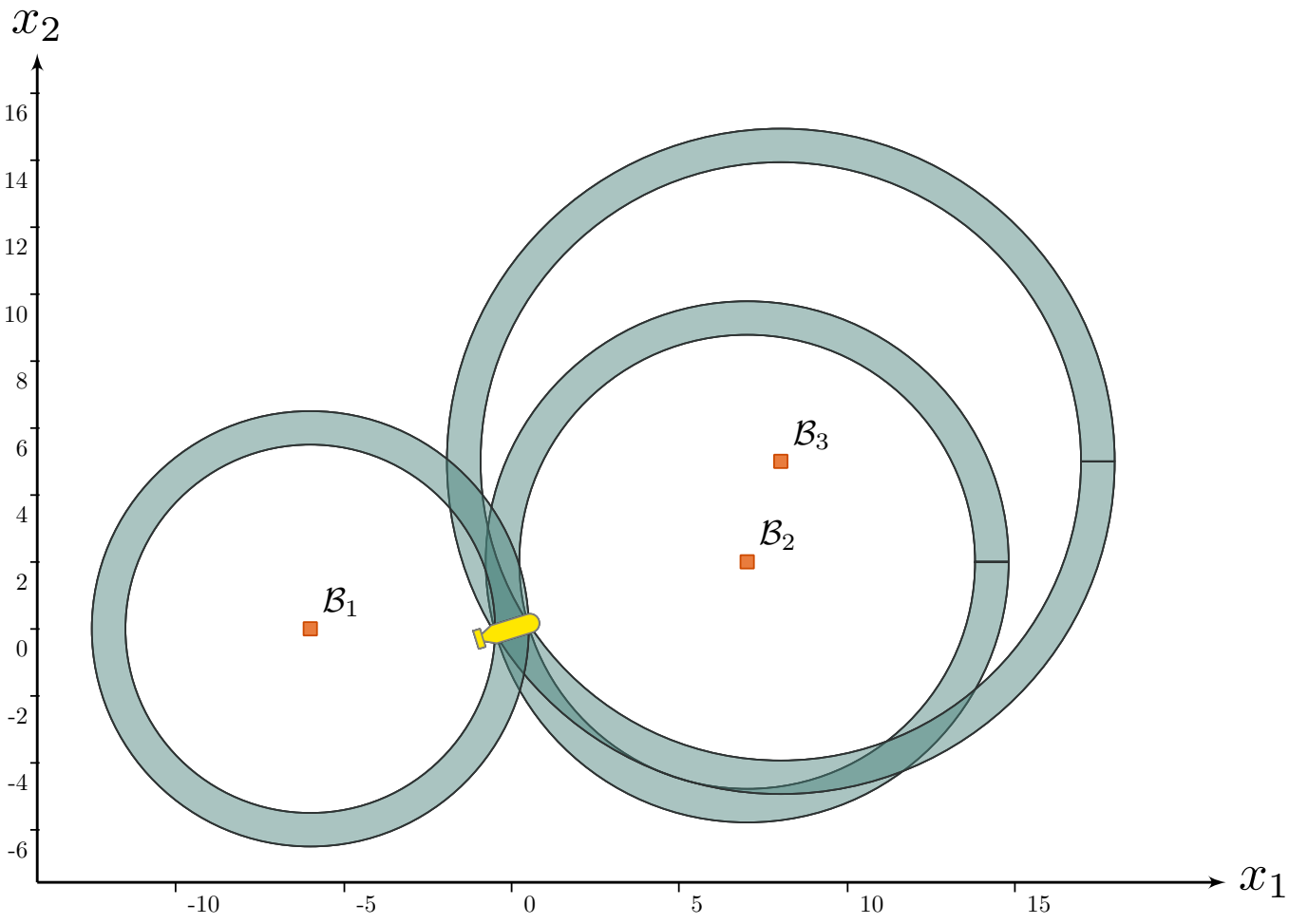


Illustration of Long BaseLine (LBL) positioning

Uncertainties as sets

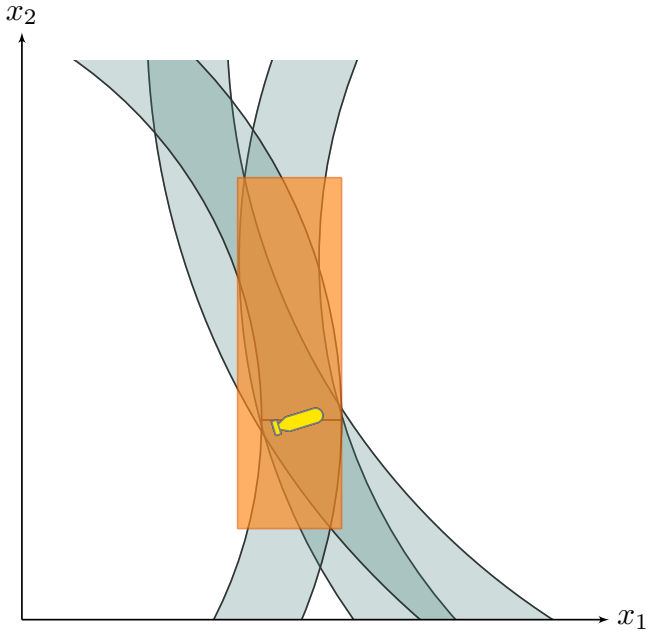
Example of **range-only** robot localization (three beacons):



LBL positioning with bounded uncertainties

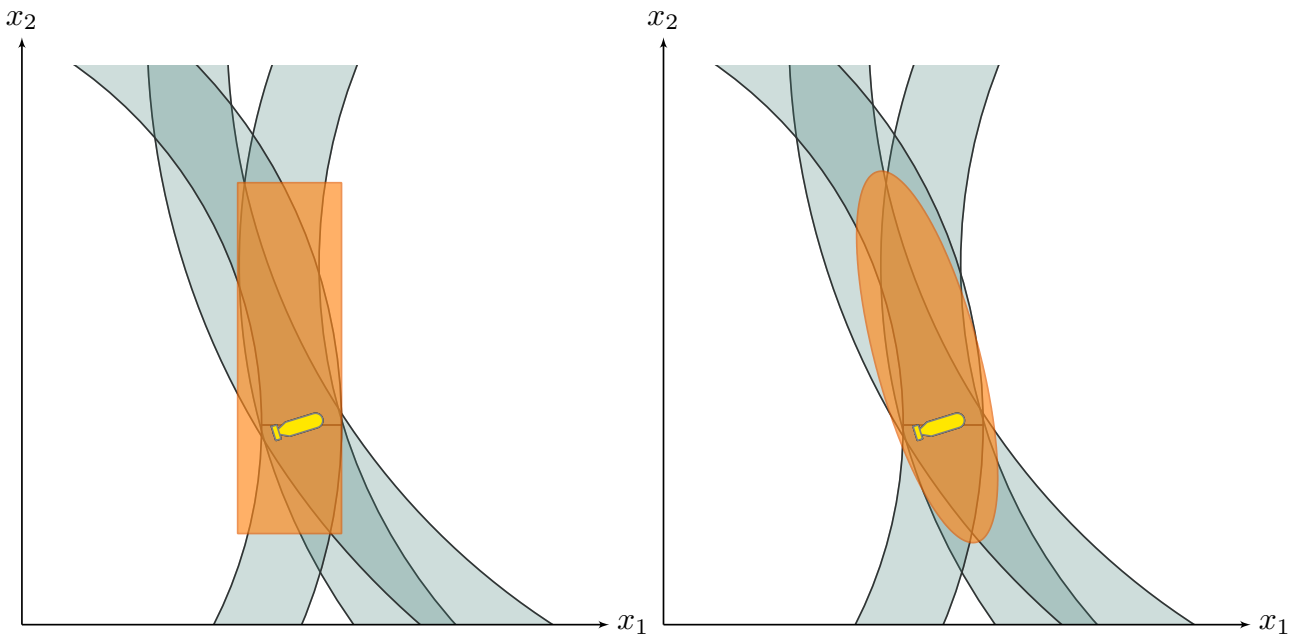
Wrappers

- box



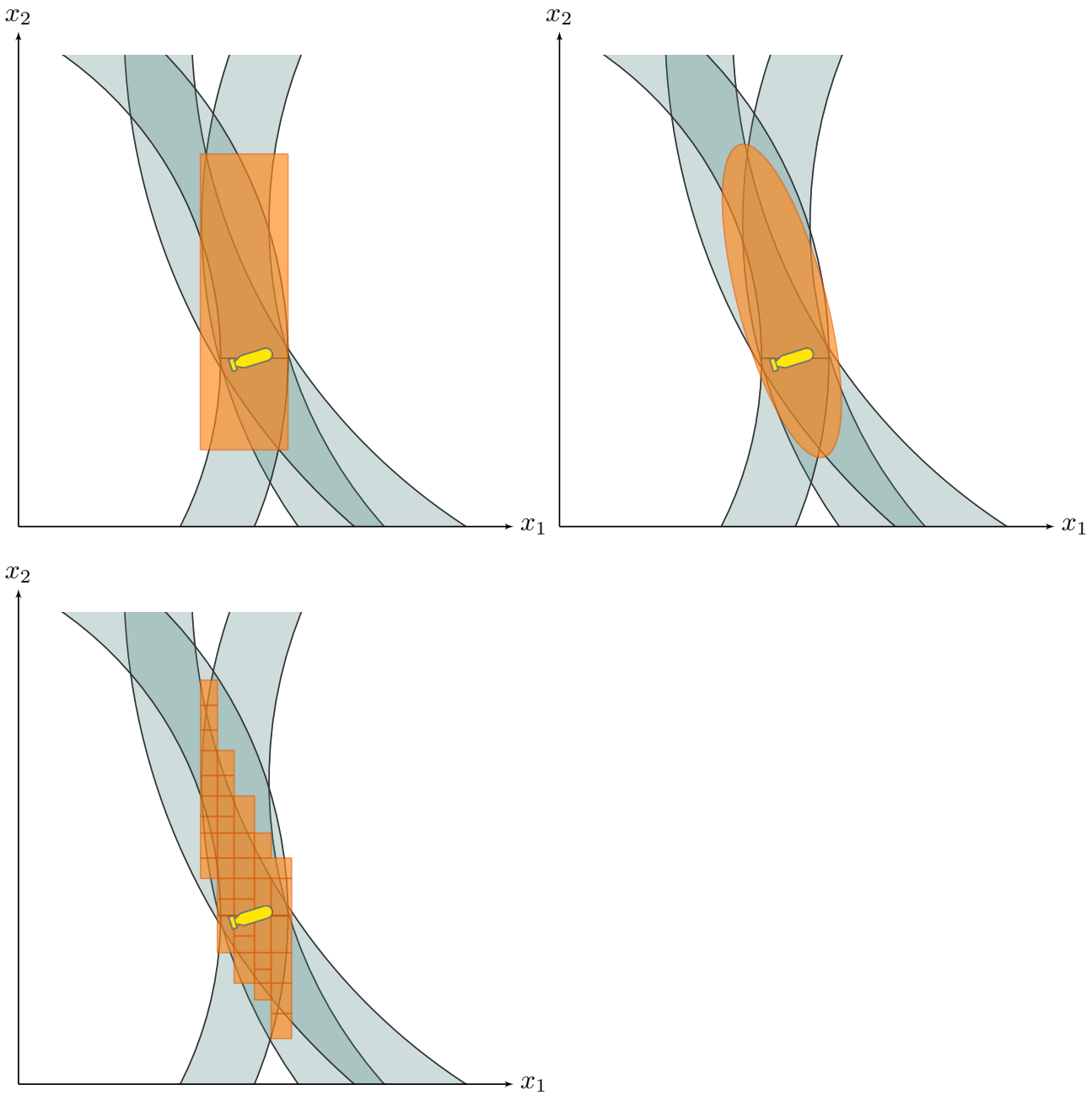
Wrappers

- box
- ellipsoid



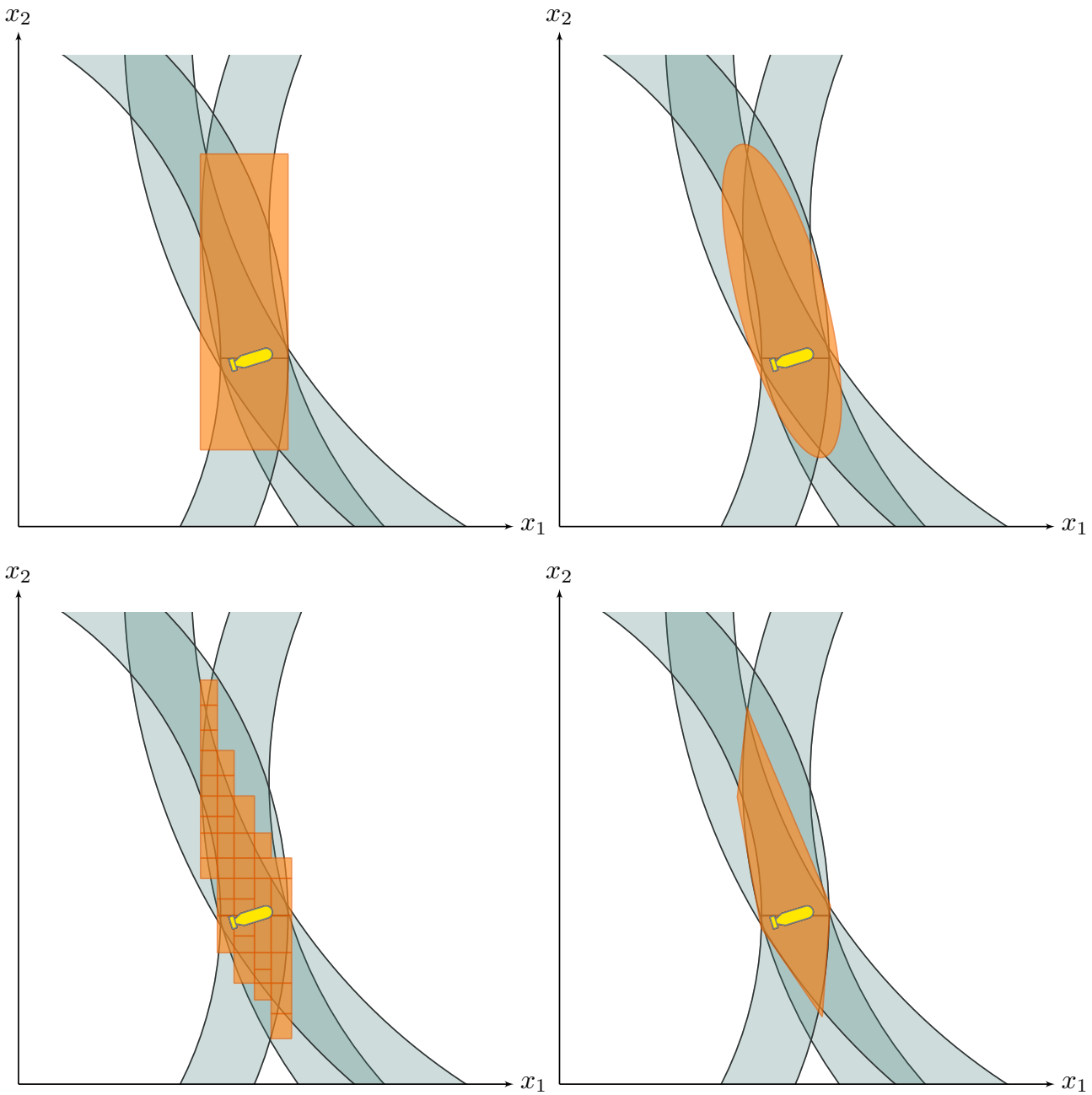
Wrappers

- box
- ellipsoid
- paving



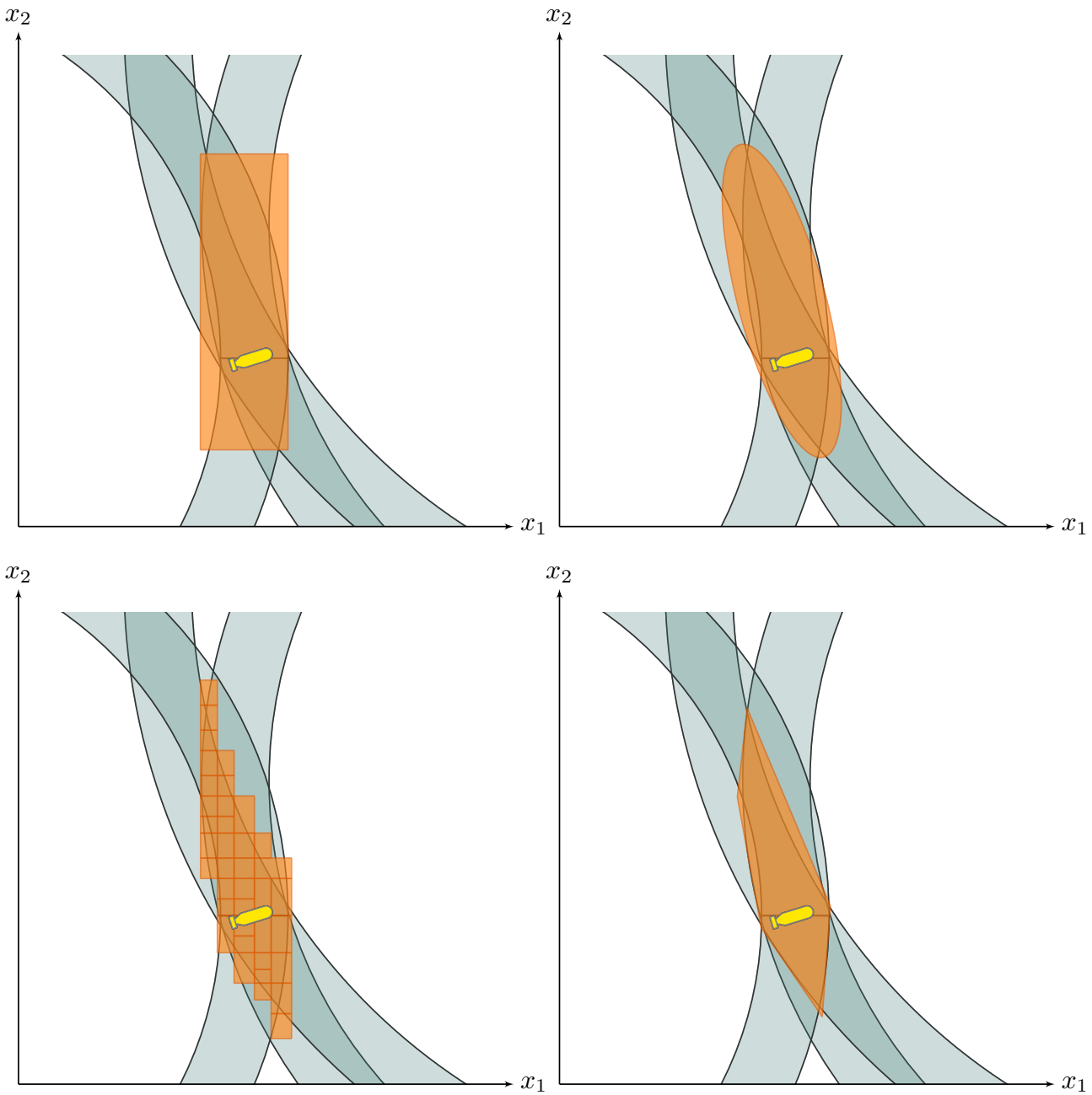
Wrappers

- box
- ellipsoid
- paving
- polytopes



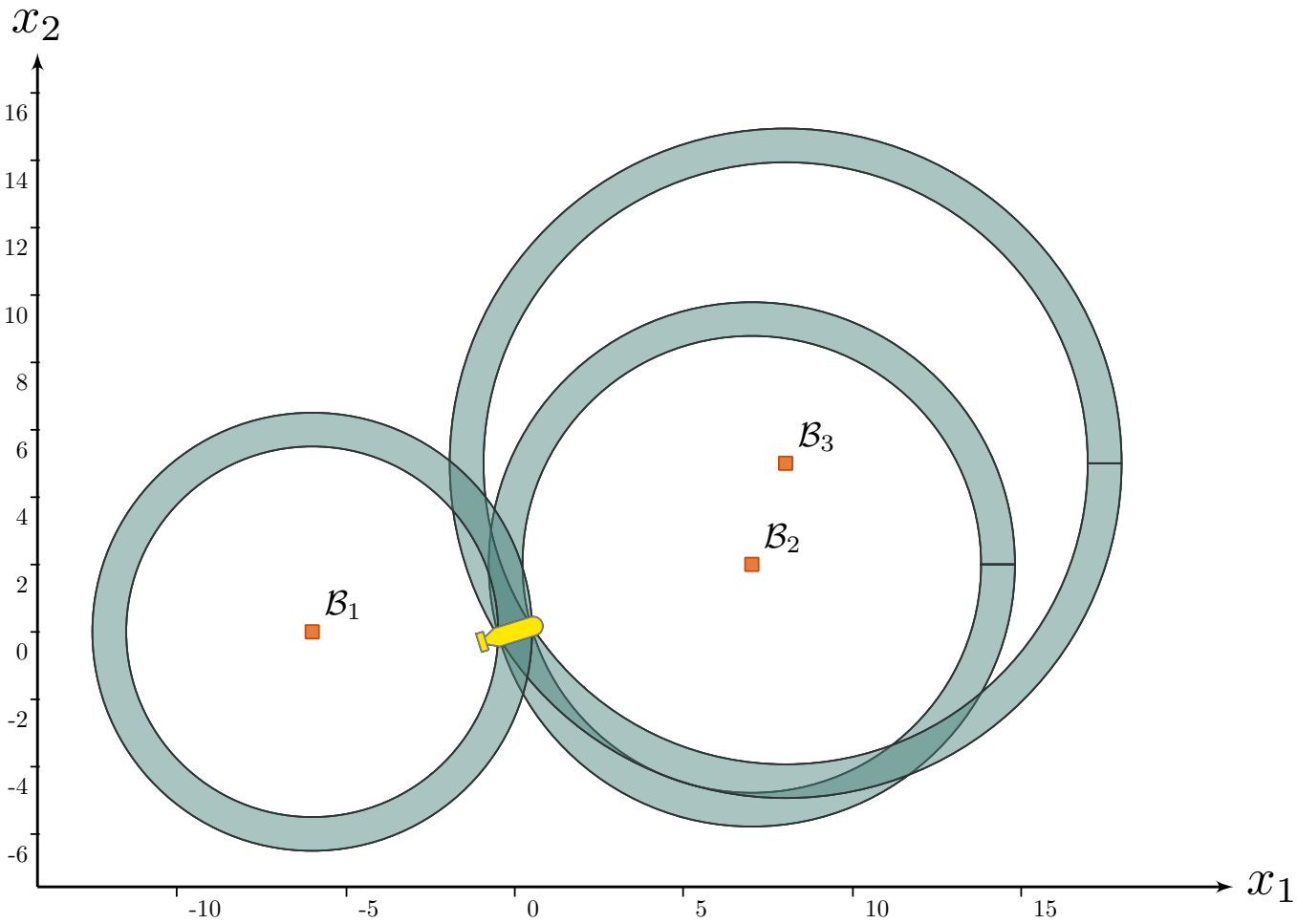
Wrappers

- box
- ellipsoid
- paving
- polytopes
- ...



Set-membership state estimation

Three observations $\rho^{(k)}$ from three beacons $\mathcal{B}^{(k)}$:



Constraints

Observation constraint, links a measurement $\rho^{(k)}$ to the state $\mathbf{x}$:

$$\rho^{(k)} = \sqrt{\left(x_1 - \mathcal{B}_1^{(k)}\right)^2 + \left(x_2 - \mathcal{B}_2^{(k)}\right)^2}.$$

Constraints

Observation constraint, links a measurement $\rho^{(k)}$ to the state $\mathbf{x}$:

$$\mathcal{L}_g^{(k)} : \rho^{(k)} = \sqrt{\left(x_1 - \mathcal{B}_1^{(k)}\right)^2 + \left(x_2 - \mathcal{B}_2^{(k)}\right)^2}.$$

Problem synthesized as a **constraint network**:

$$\begin{cases} \textbf{Variables: } \mathbf{x},\ \rho^{(1)},\ \rho^{(2)},\ \rho^{(3)} \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_g^{(1)}\left(\mathbf{x}, \rho^{(1)}\right) \\ \quad 2.\ \mathcal{L}_g^{(2)}\left(\mathbf{x}, \rho^{(2)}\right) \\ \quad 3.\ \mathcal{L}_g^{(3)}\left(\mathbf{x}, \rho^{(3)}\right) \\ \textbf{Domains: } [\mathbf{x}],\ [\rho^{(1)}],\ [\rho^{(2)}],\ [\rho^{(3)}] \end{cases}$$

Constraints

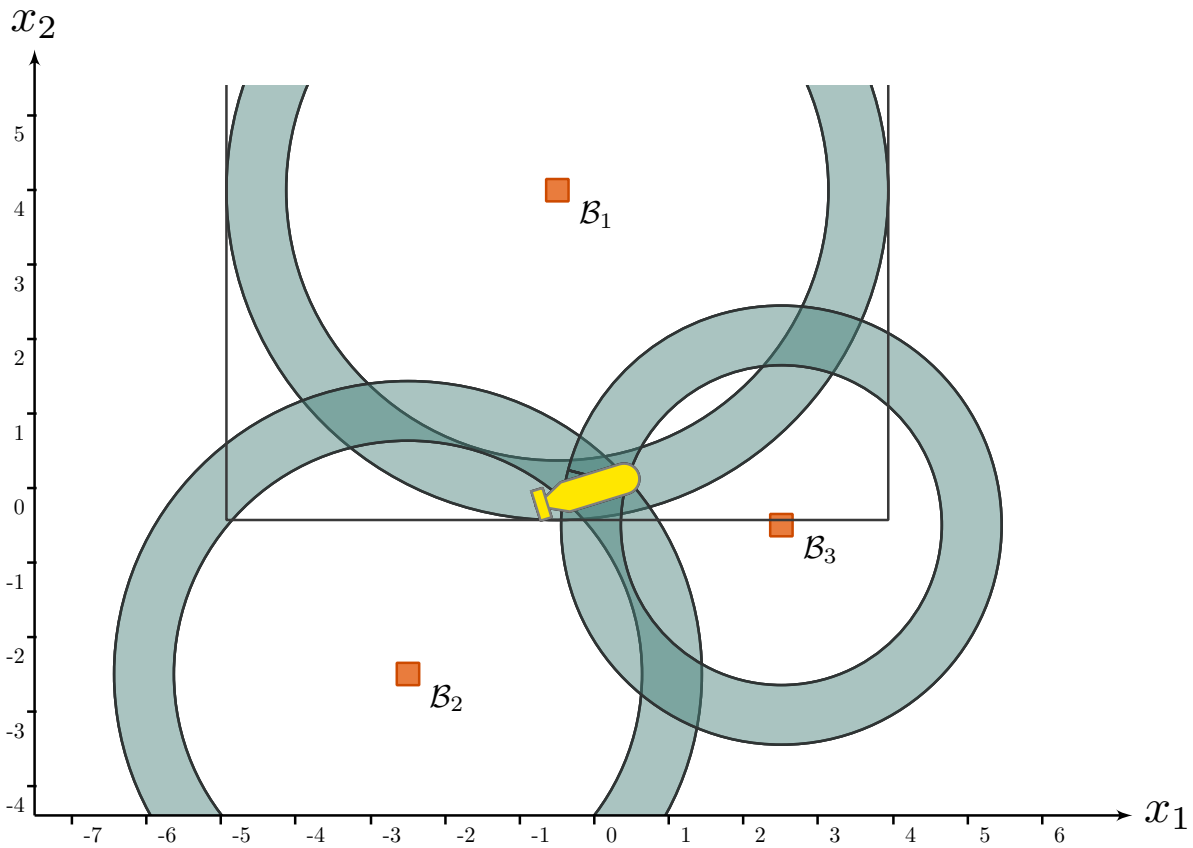
Observation constraint, links a measurement $\rho^{(k)}$ to the state $\mathbf{x}$:

$$\mathcal{L}_g^{(k)} : \rho^{(k)} = \sqrt{\left(x_1 - \mathcal{B}_1^{(k)}\right)^2 + \left(x_2 - \mathcal{B}_2^{(k)}\right)^2}.$$

Problem synthesized as a **constraint network**:

$$\begin{cases} \textbf{Variables: } \mathbf{x},\ \rho^{(1)},\ \rho^{(2)},\ \rho^{(3)} \\ \textbf{Constraints:} \\ \quad 1.\ \mathcal{L}_g^{(1)}\left(\mathbf{x}, \rho^{(1)}\right) \implies \mathcal{C}_g^{(1)}\left([\mathbf{x}], [\rho^{(1)}]\right) \\ \quad 2.\ \mathcal{L}_g^{(2)}\left(\mathbf{x}, \rho^{(2)}\right) \implies \mathcal{C}_g^{(2)}\left([\mathbf{x}], [\rho^{(2)}]\right) \\ \quad 3.\ \mathcal{L}_g^{(3)}\left(\mathbf{x}, \rho^{(3)}\right) \implies \mathcal{C}_g^{(3)}\left([\mathbf{x}], [\rho^{(3)}]\right) \\ \textbf{Domains: } [\mathbf{x}],\ [\rho^{(1)}],\ [\rho^{(2)}],\ [\rho^{(3)}] \end{cases}$$

Fixed point propagations

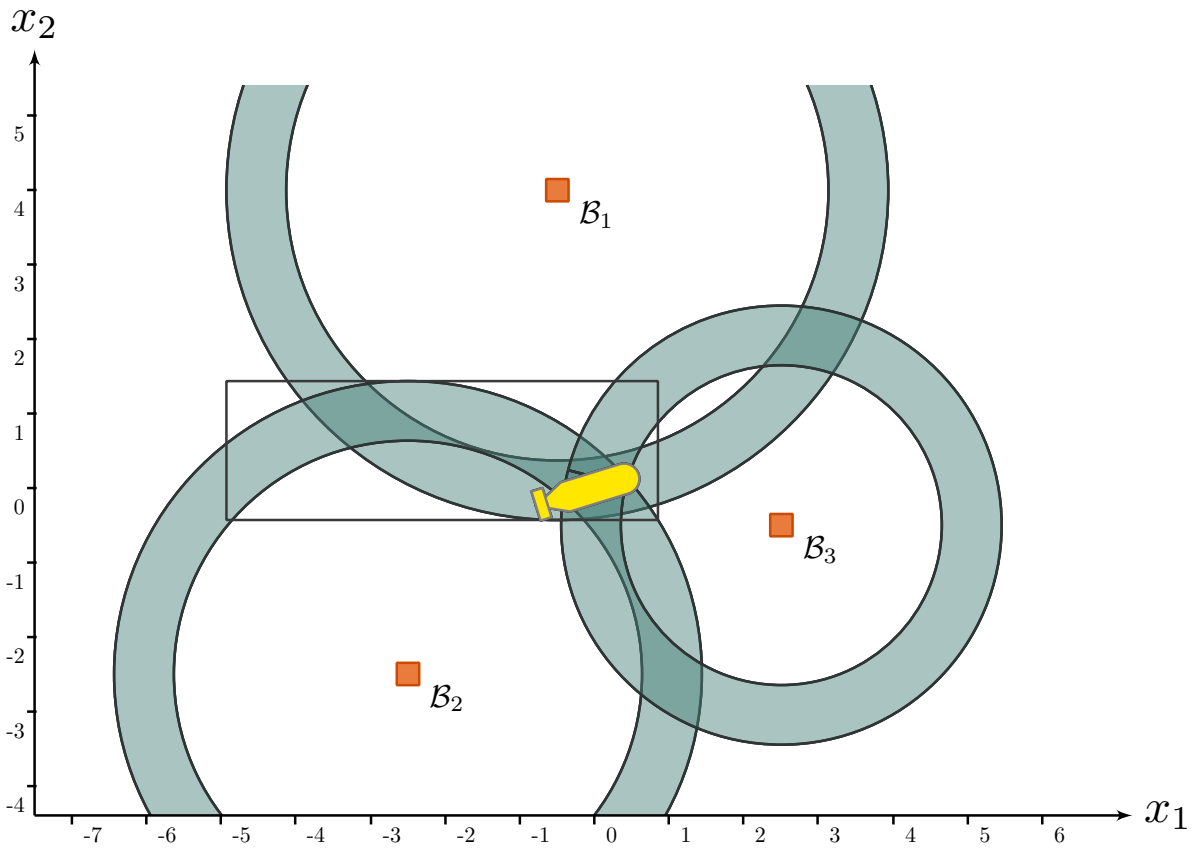


Applying $\mathcal{C}_g^{(1)}$

- Study of robust set estimation methods for a high integrity multi-sensor localization.

Vincent Drevelle *Thesis*, 2011

Fixed point propagations

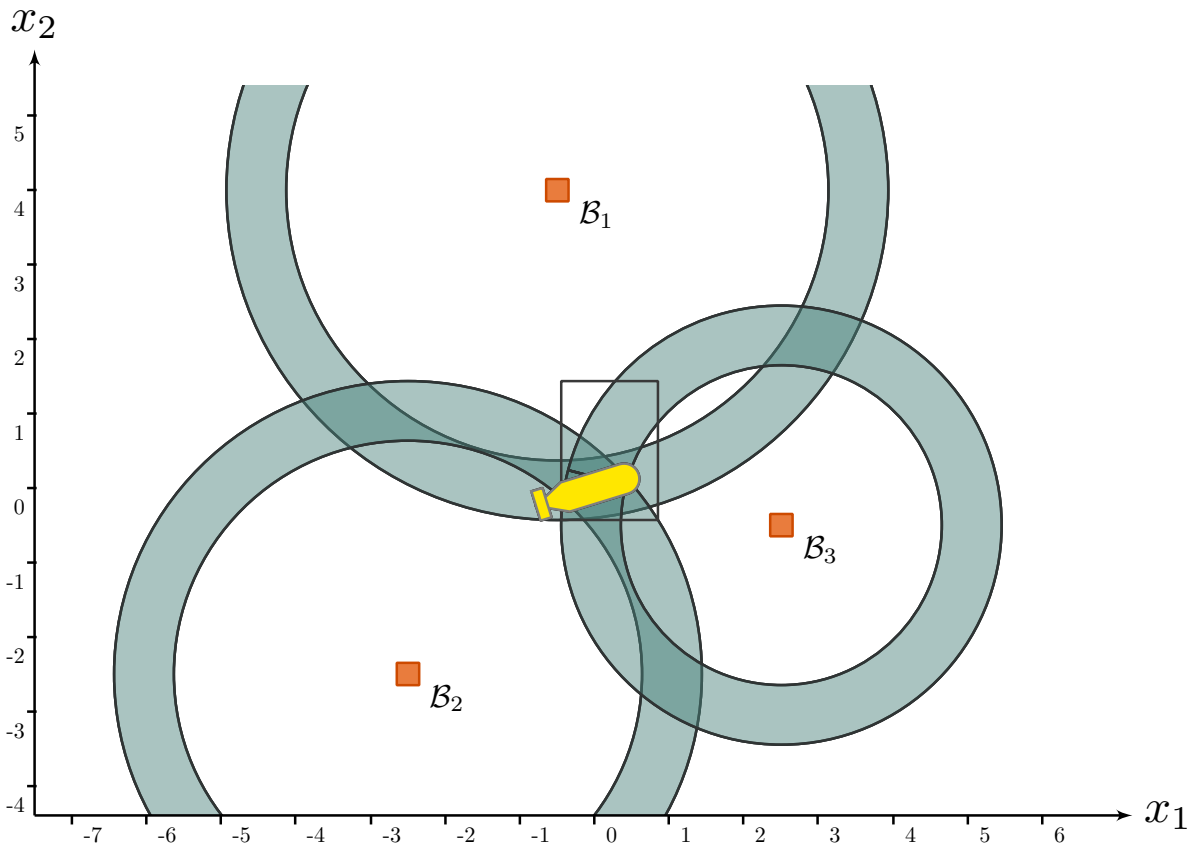


Applying $\mathcal{C}_g^{(2)}$

- Study of robust set estimation methods for a high integrity multi-sensor localization.

Vincent Drevelle *Thesis*, 2011

Fixed point propagations

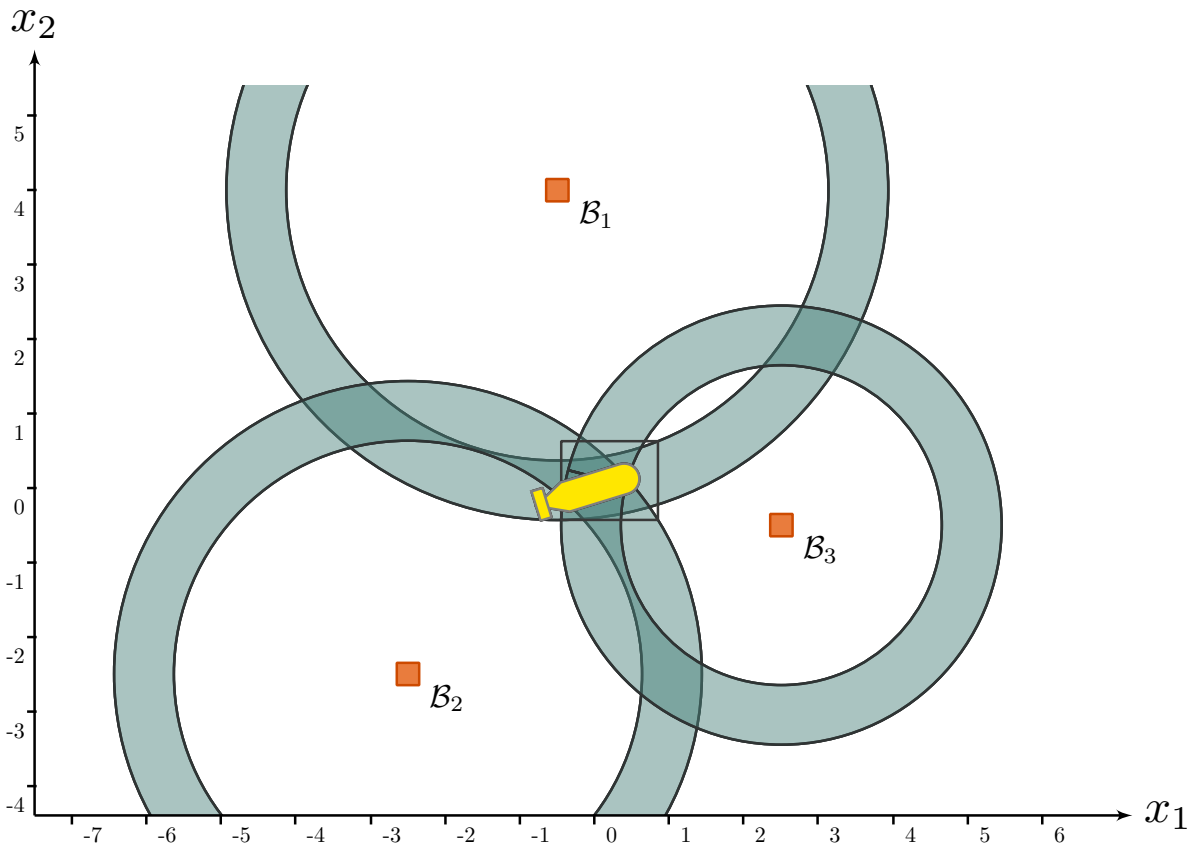


Applying $\mathcal{C}_g^{(3)}$

- Study of robust set estimation methods for a high integrity multi-sensor localization.

Vincent Drevelle *Thesis*, 2011

Fixed point propagations

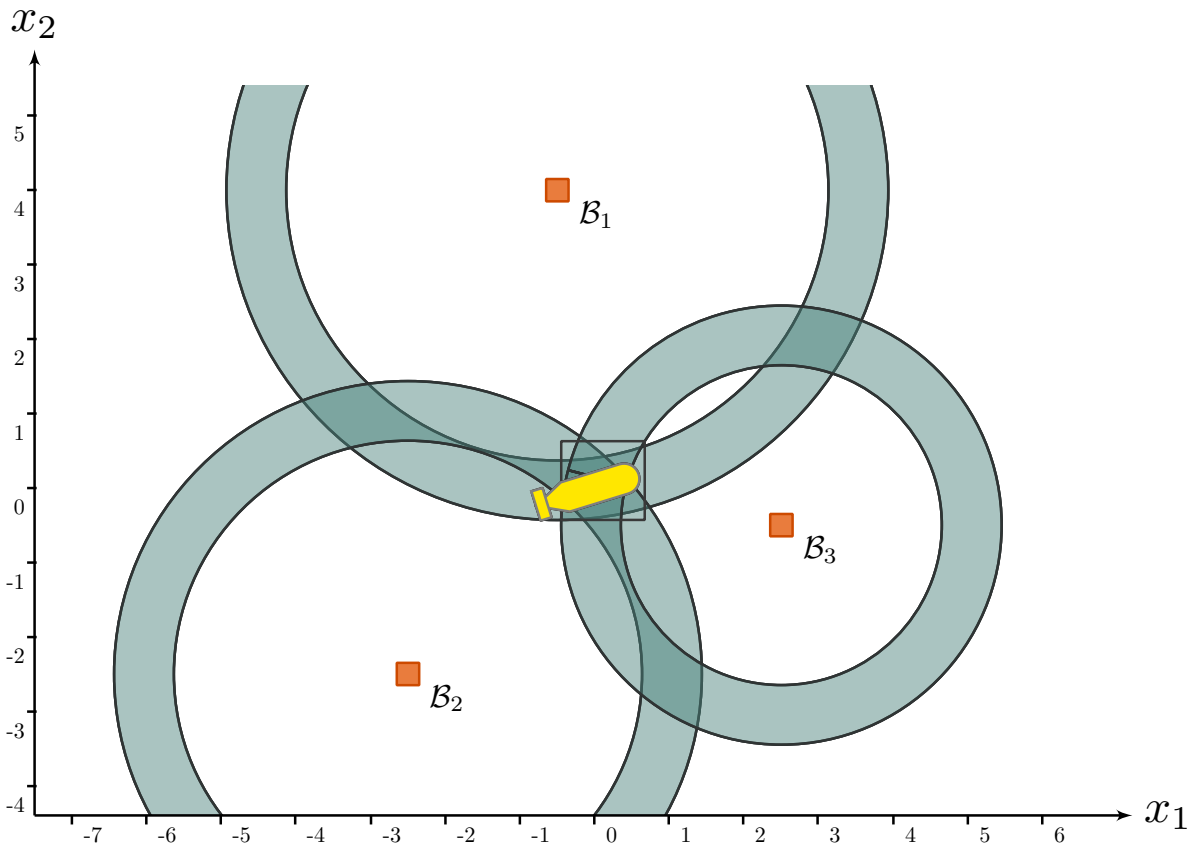


Applying $\mathcal{C}_g^{(1)}$

- Study of robust set estimation methods for a high integrity multi-sensor localization.

Vincent Drevelle *Thesis*, 2011

Fixed point propagations

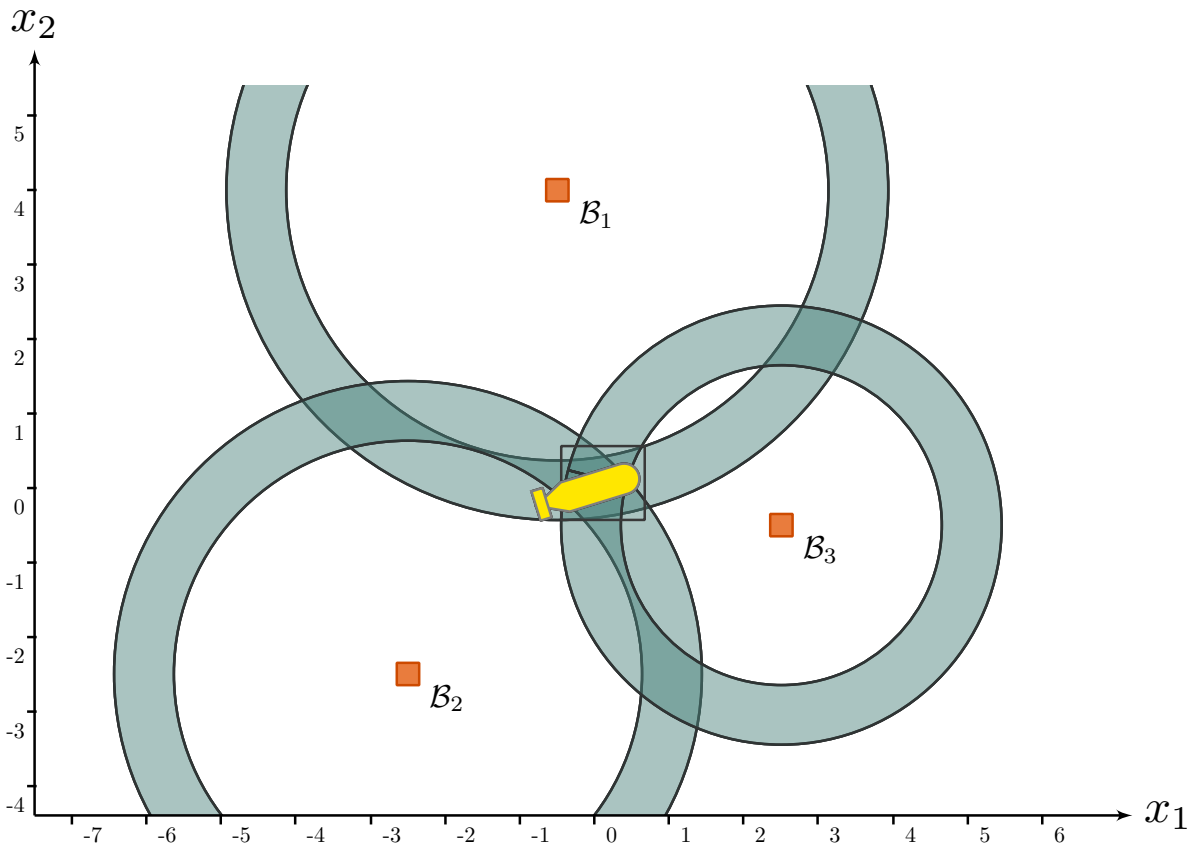


Applying $\mathcal{C}_g^{(2)}$

- Study of robust set estimation methods for a high integrity multi-sensor localization.

Vincent Drevelle *Thesis*, 2011

Fixed point propagations

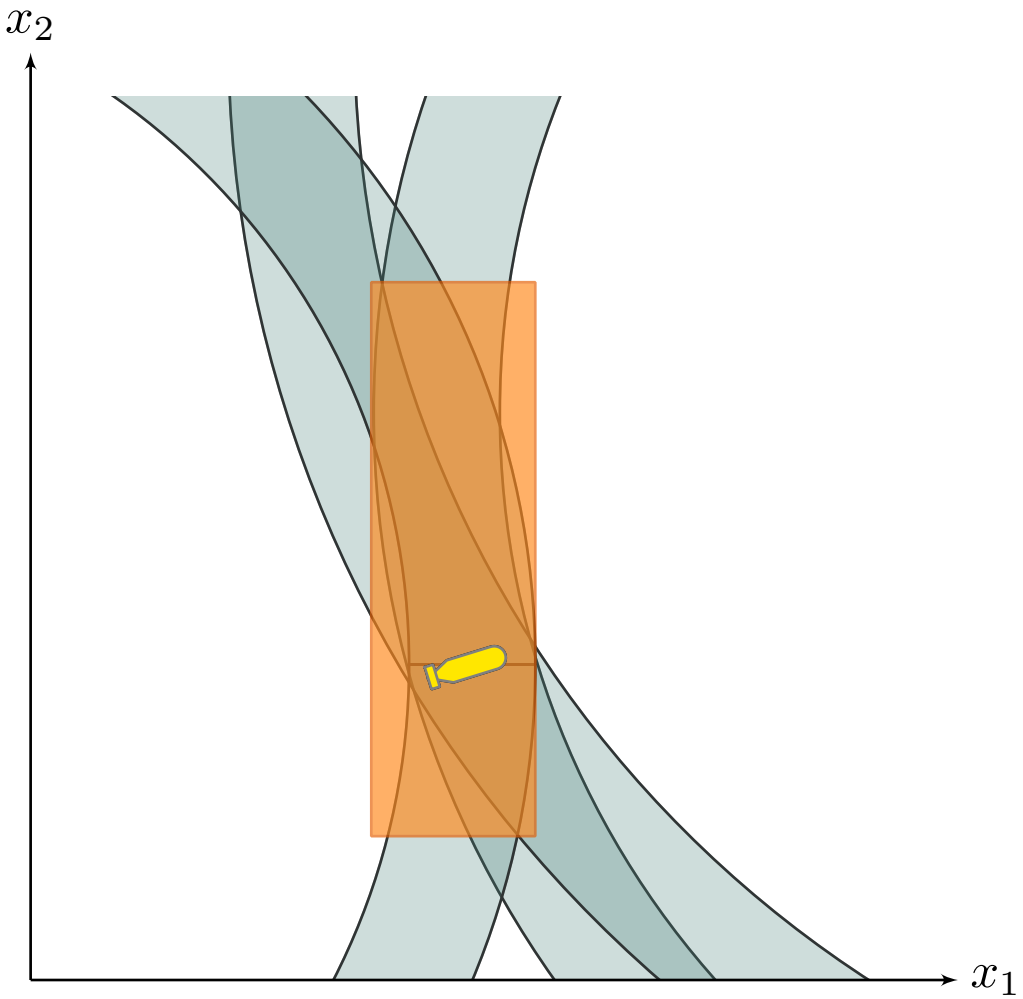


Fixed point reached.

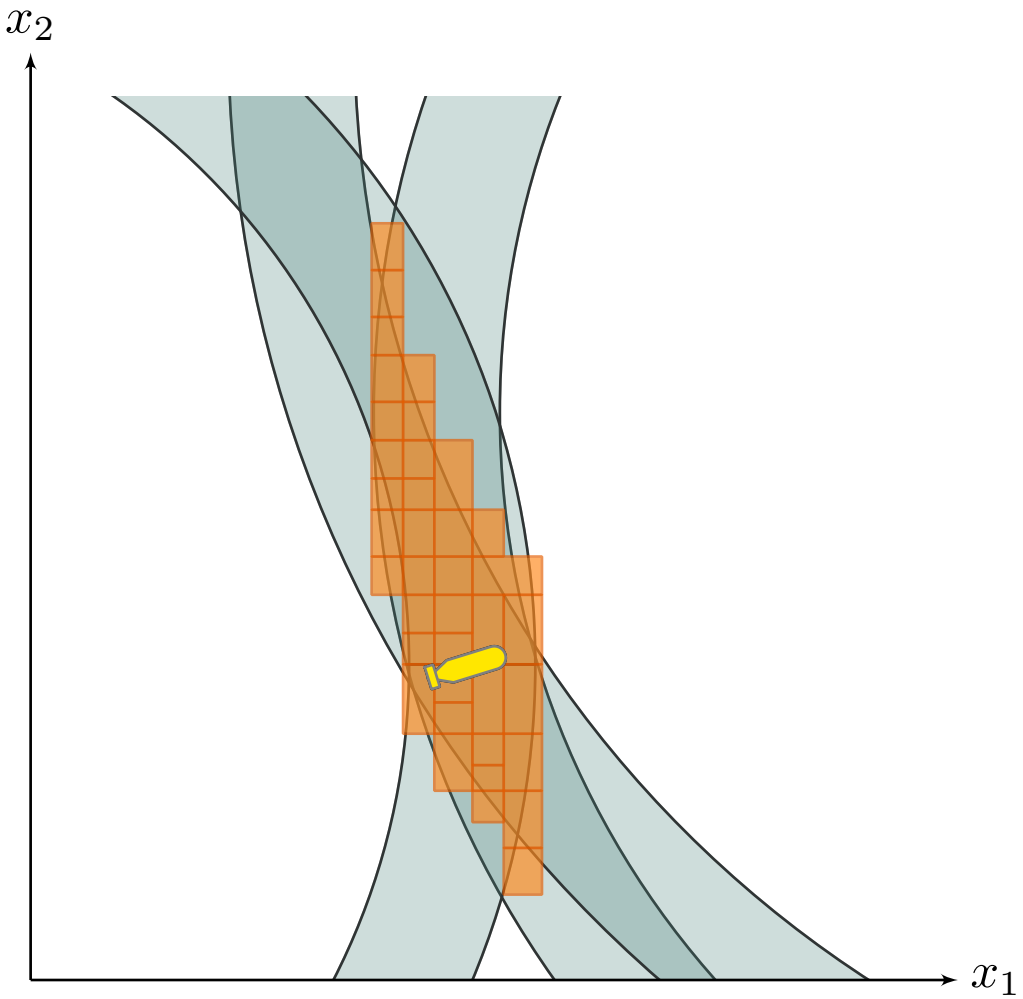
- Study of robust set estimation methods for a high integrity multi-sensor localization.

Vincent Drevelle *Thesis*, 2011

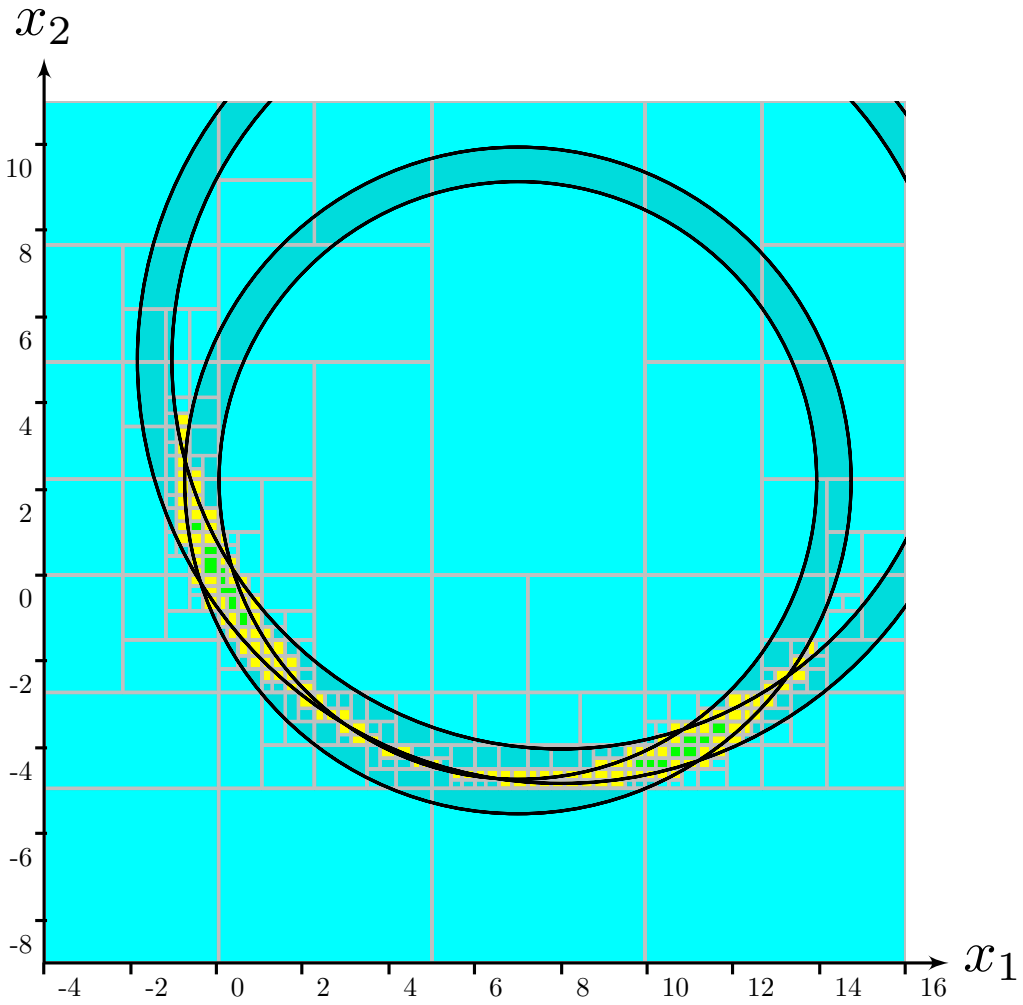
Sub-pavings: finer approximation of sets



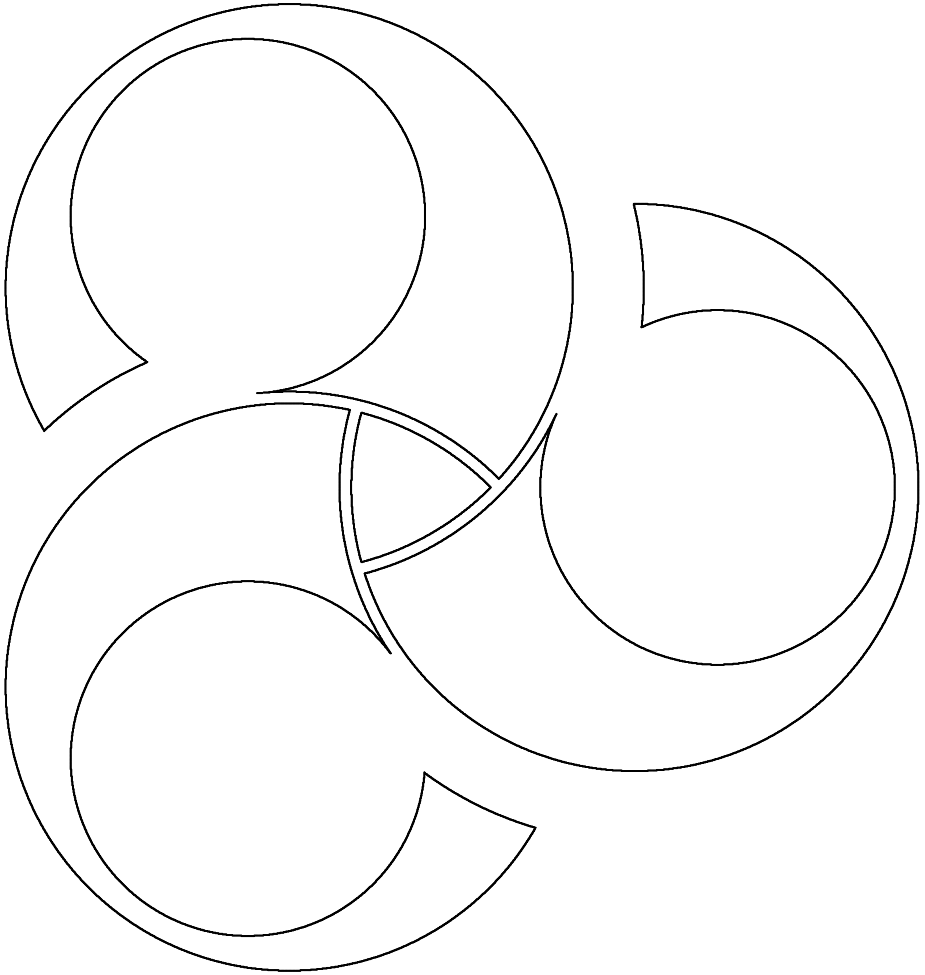
Sub-pavings: finer approximation of sets



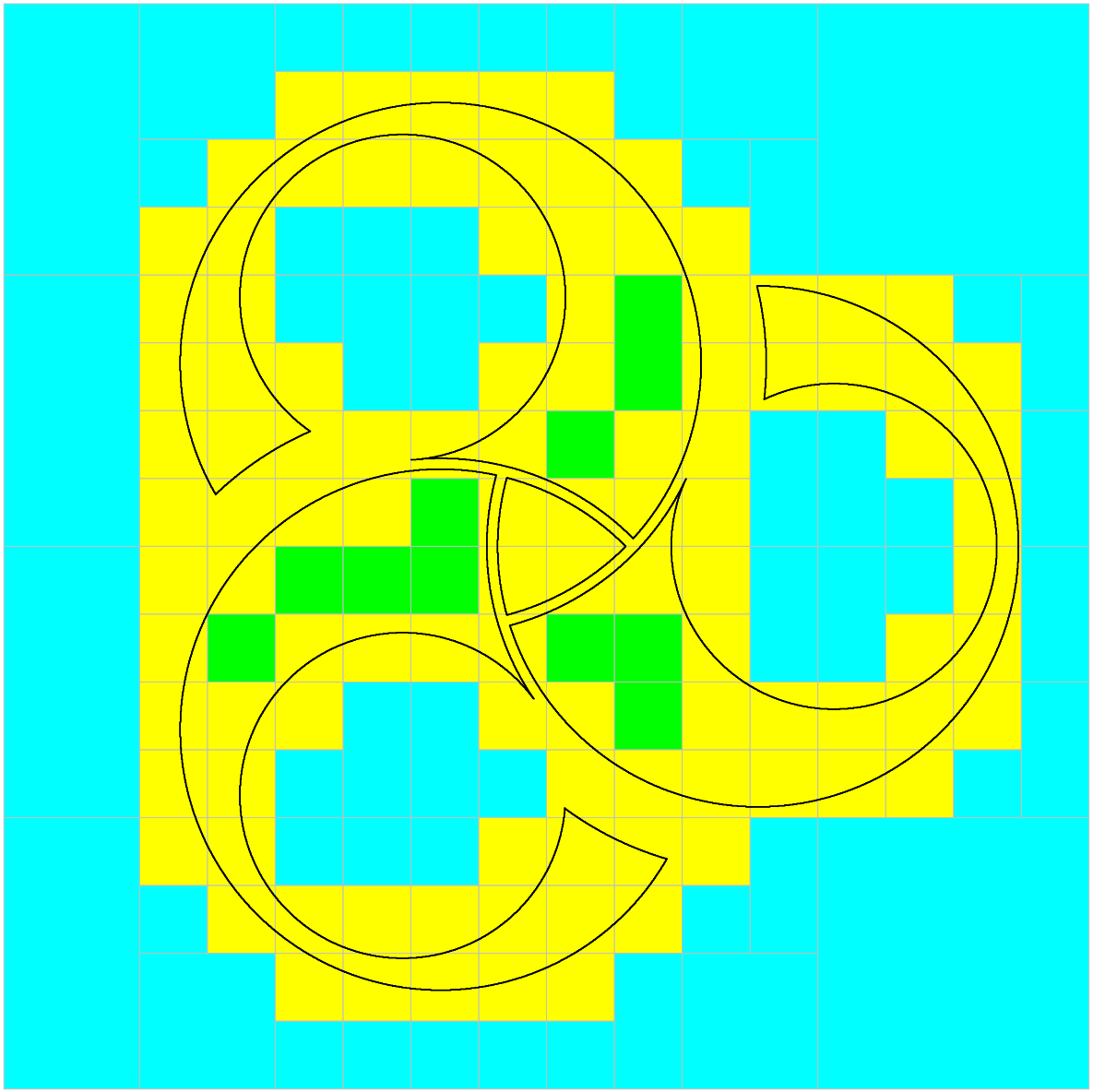
Sub-pavings: finer approximation of sets



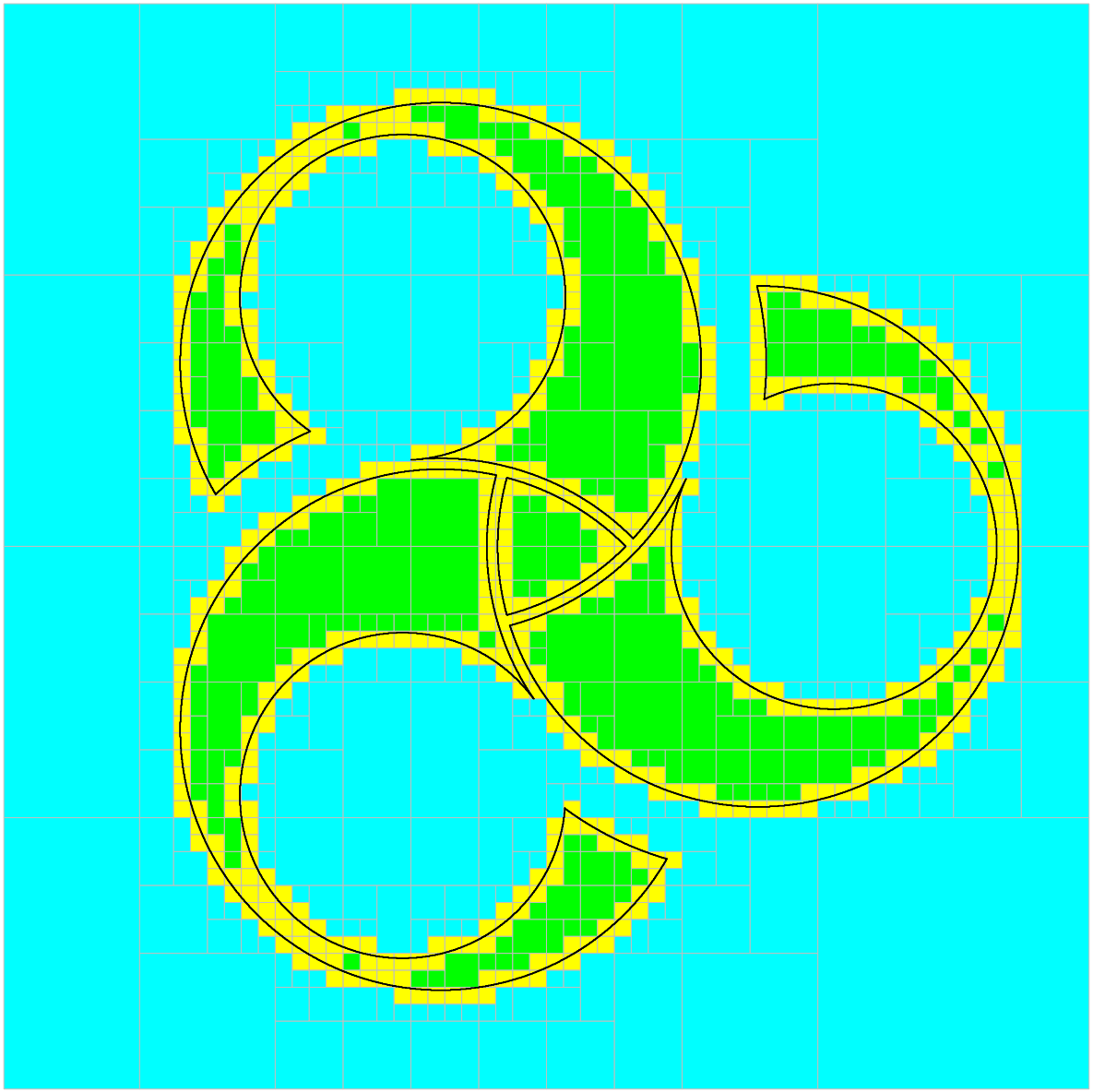
Sub-pavings: precision



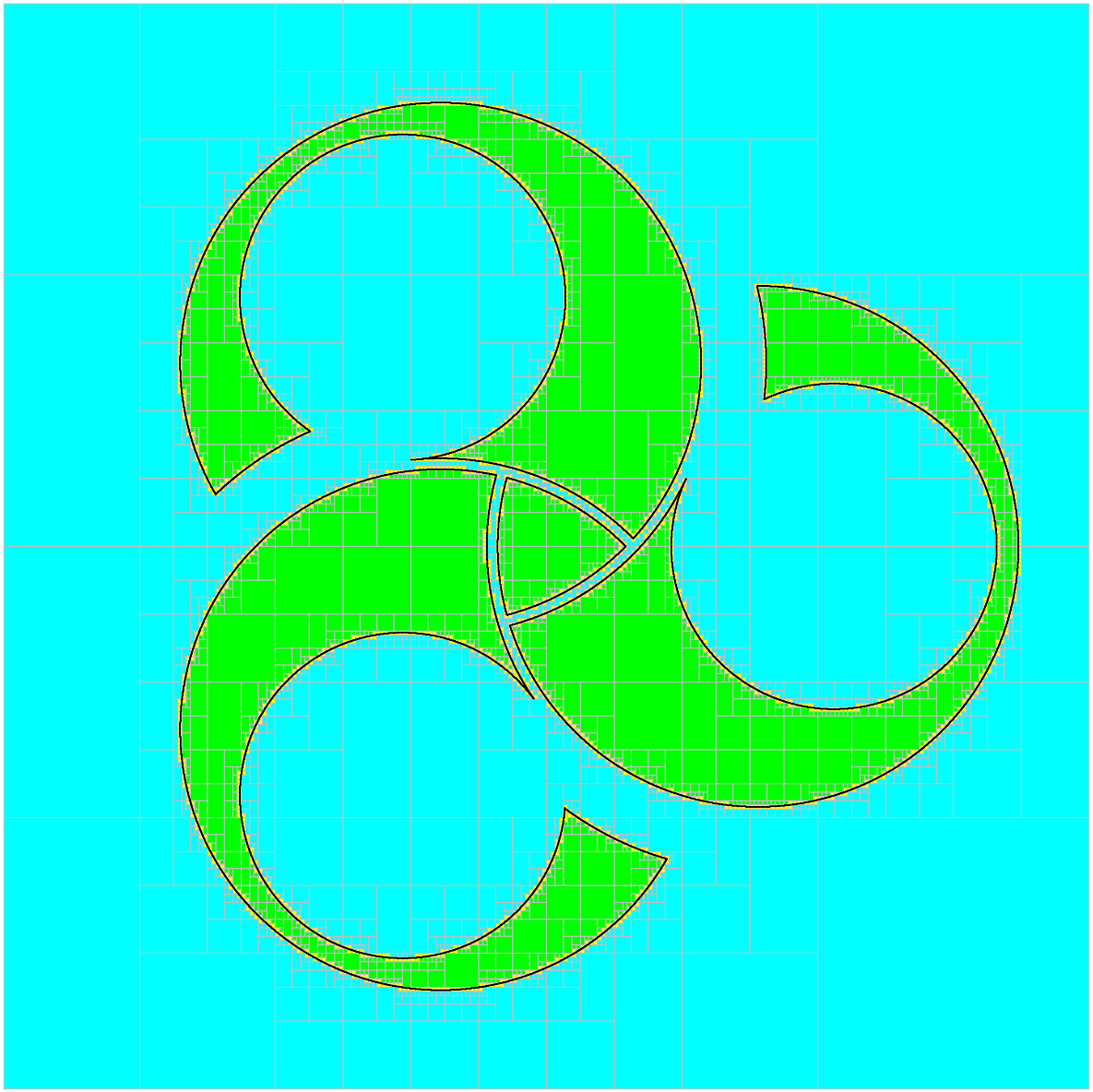
Sub-pavings: precision



Sub-pavings: precision



Sub-pavings: precision



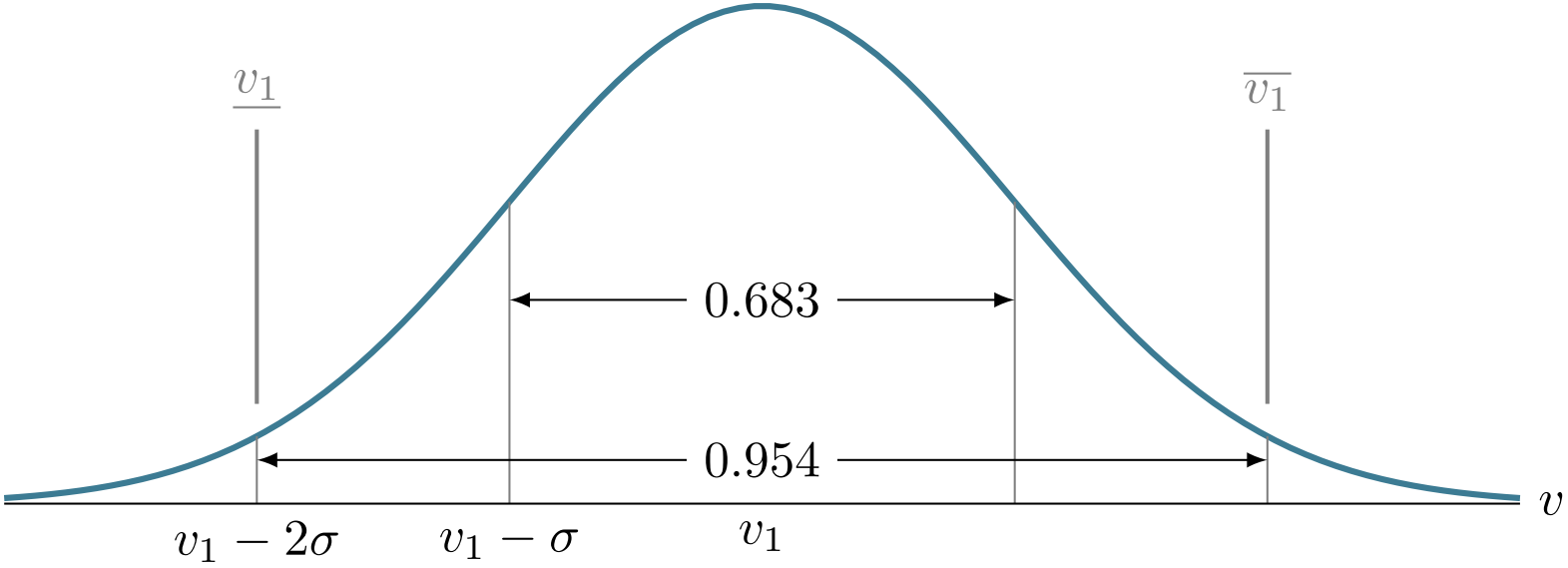
Intervals from sensor data

Video

Intervals from sensor data

The Gaussian case:

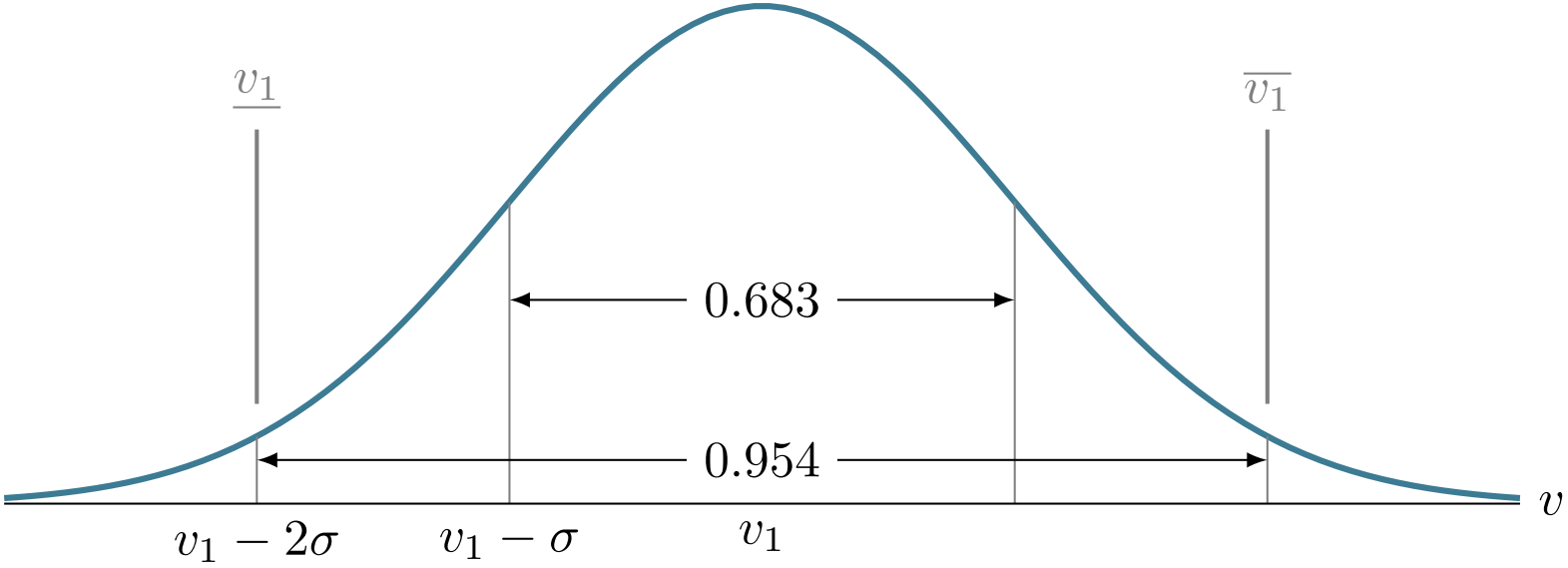
- datasheets $\implies$ standard deviation σ for each sensor
- 95% confidence rate: $v_1^* \in [v_1] = [v_1 - 2\sigma, v_1 + 2\sigma]$



Intervals from sensor data

The Gaussian case:

- datasheets $\implies$ standard deviation σ for each sensor
- 95% confidence rate: $v_1^* \in [v_1] = [v_1 - 2\sigma, v_1 + 2\sigma]$



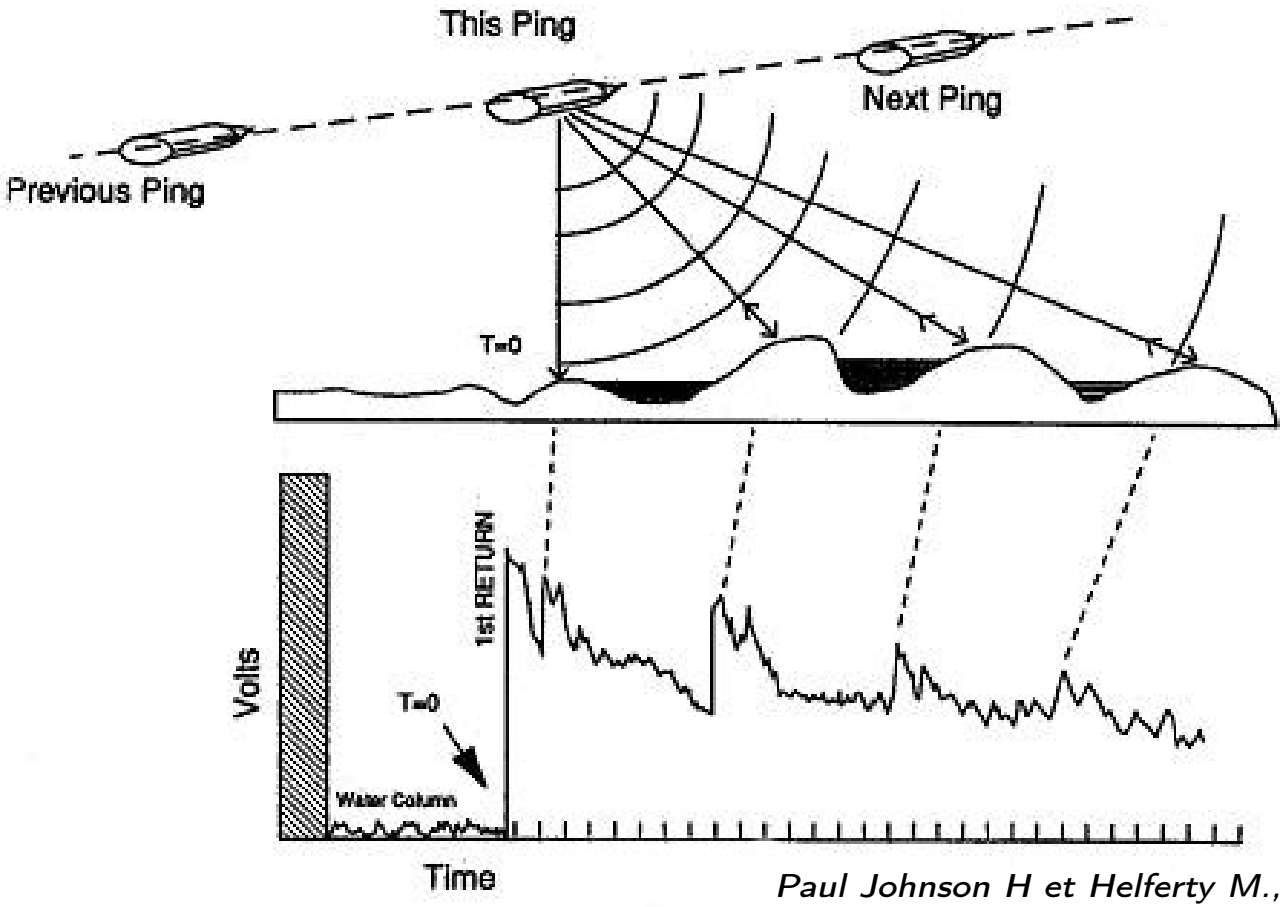
- uncertainties then reliably propagated in the system

  ex: $[x] + [y] = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right]$

Intervals from sensor data

The realistic case:

- the error is rarely a Gaussian distribution
- multimodal distribution, "no-signal" information, *etc.*

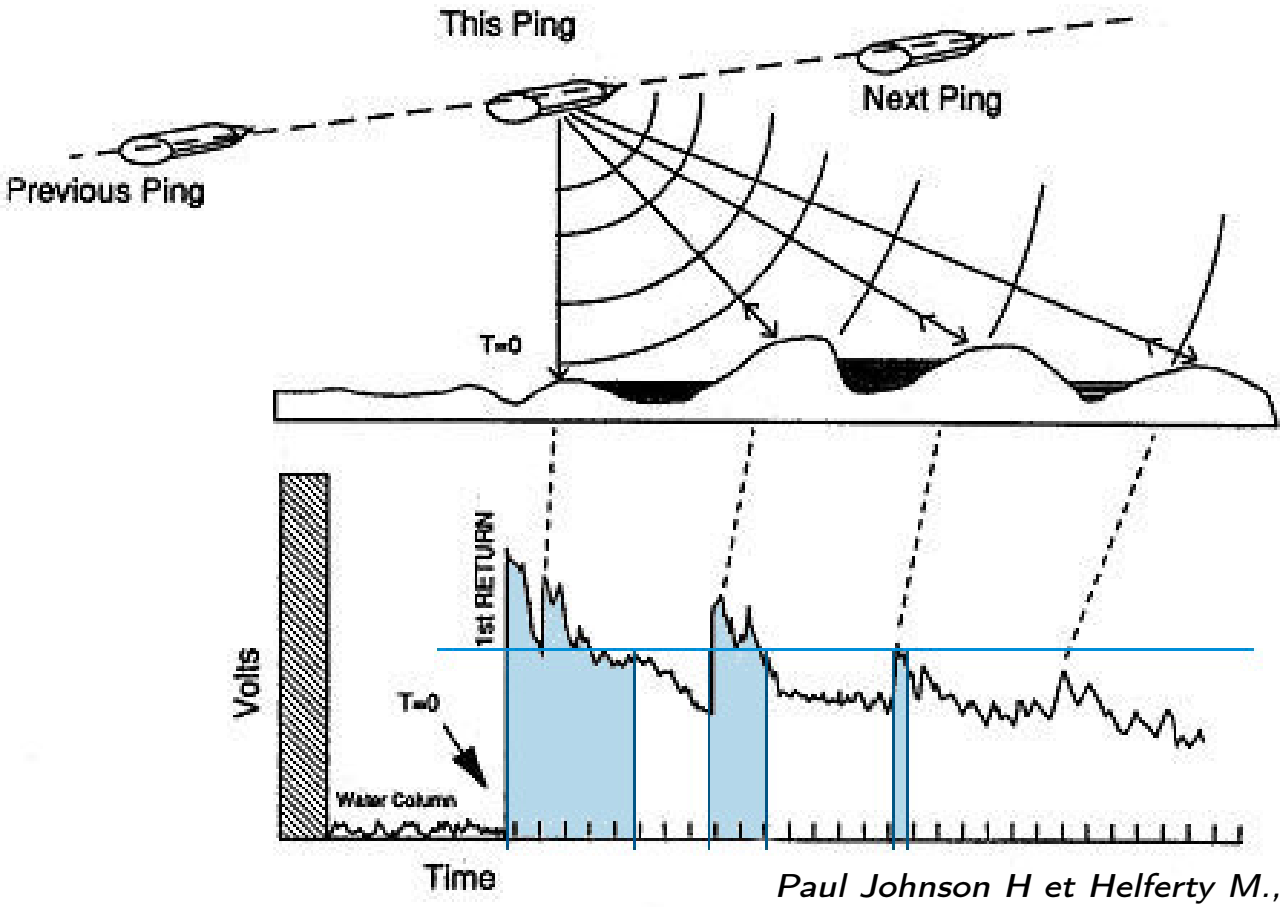


Paul Johnson H et Helferty M., 1990

Intervals from sensor data

The realistic case:

- the error is rarely a Gaussian distribution
- multimodal distribution, "no-signal" information, *etc.*

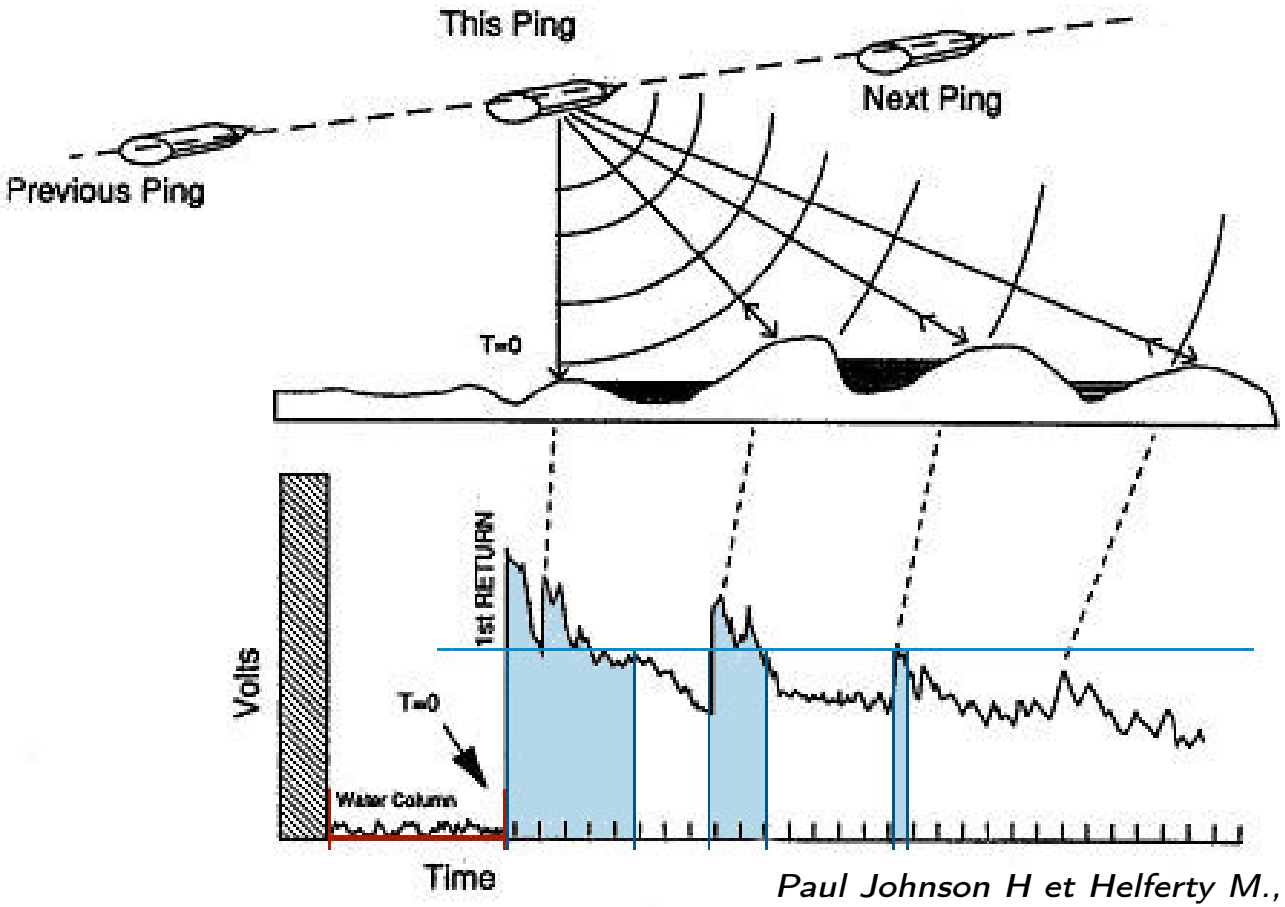


Paul Johnson H et Helferty M., 1990

Intervals from sensor data

The realistic case:

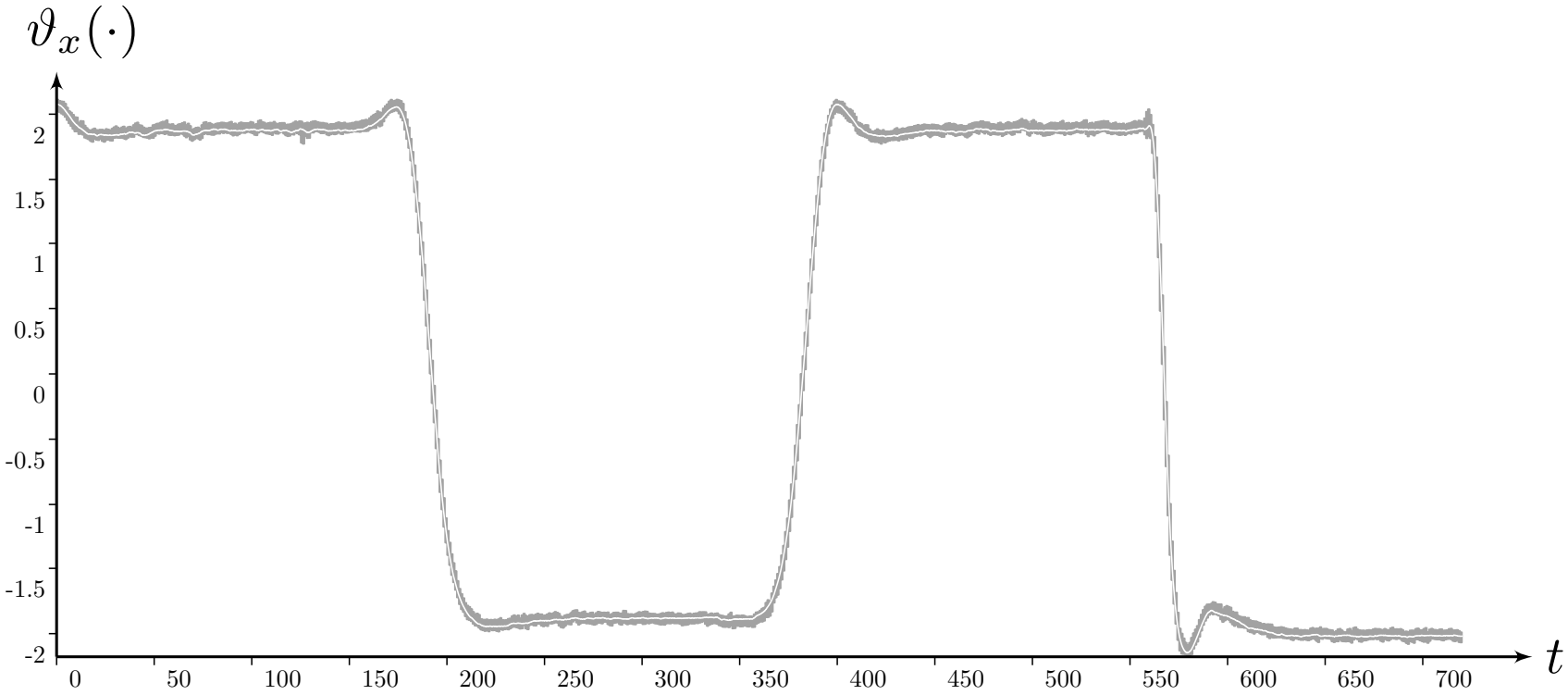
- the error is rarely a Gaussian distribution
- multimodal distribution, "no-signal" information, *etc.*



Paul Johnson H et Helferty M., 1990

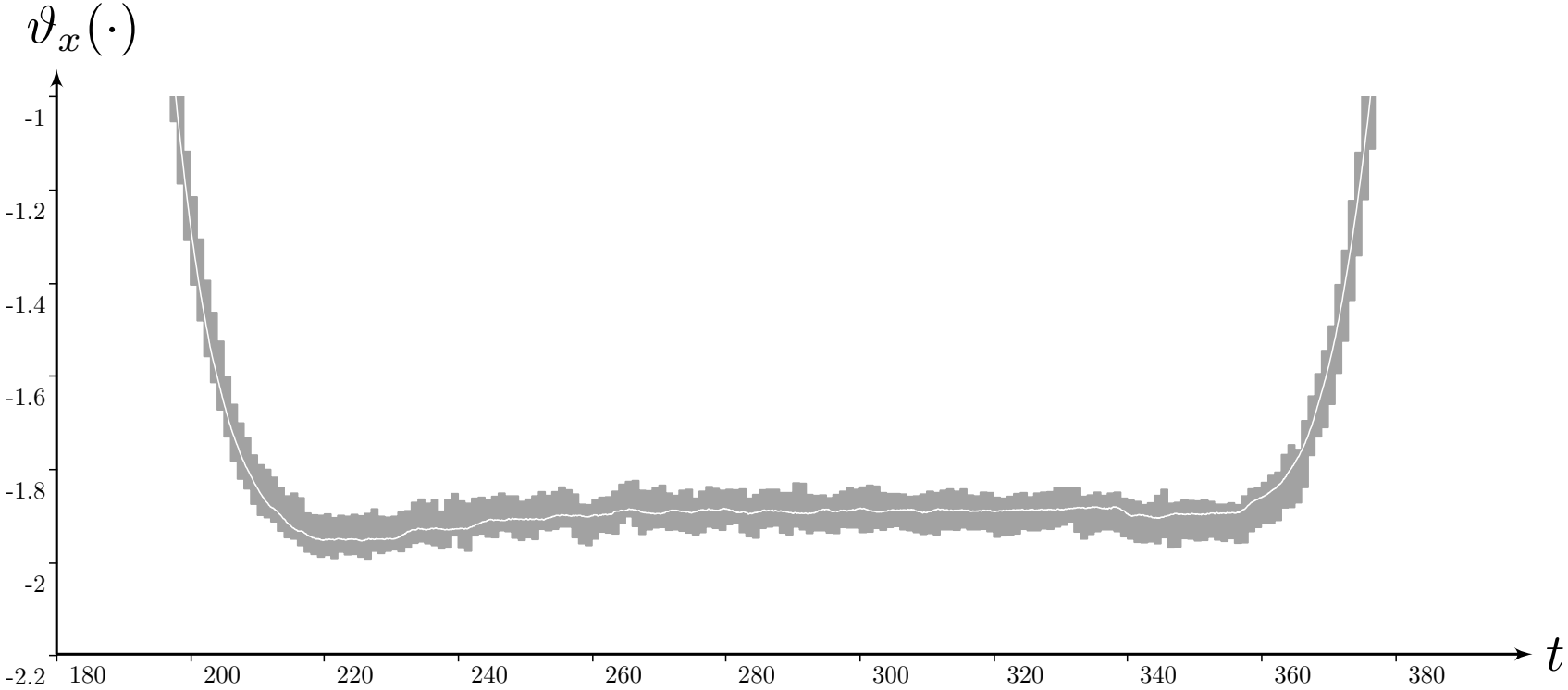
Tubes from sensor data

Example coming from underwater robotics.
East velocity given by DVL + IMU:



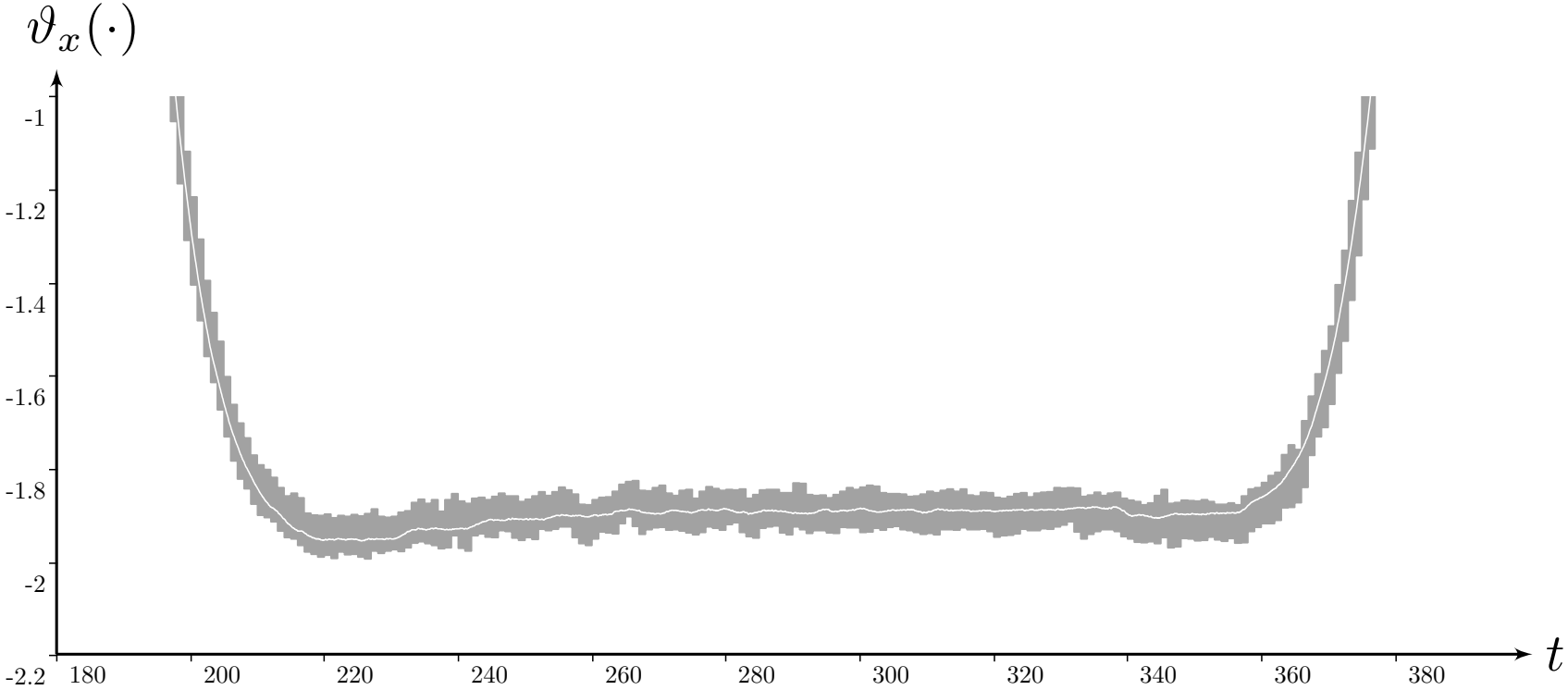
Tubes from sensor data

Example coming from underwater robotics.
East velocity given by DVL + IMU (zoom):



Tubes from sensor data

Example coming from underwater robotics.
East velocity given by DVL + IMU (zoom):

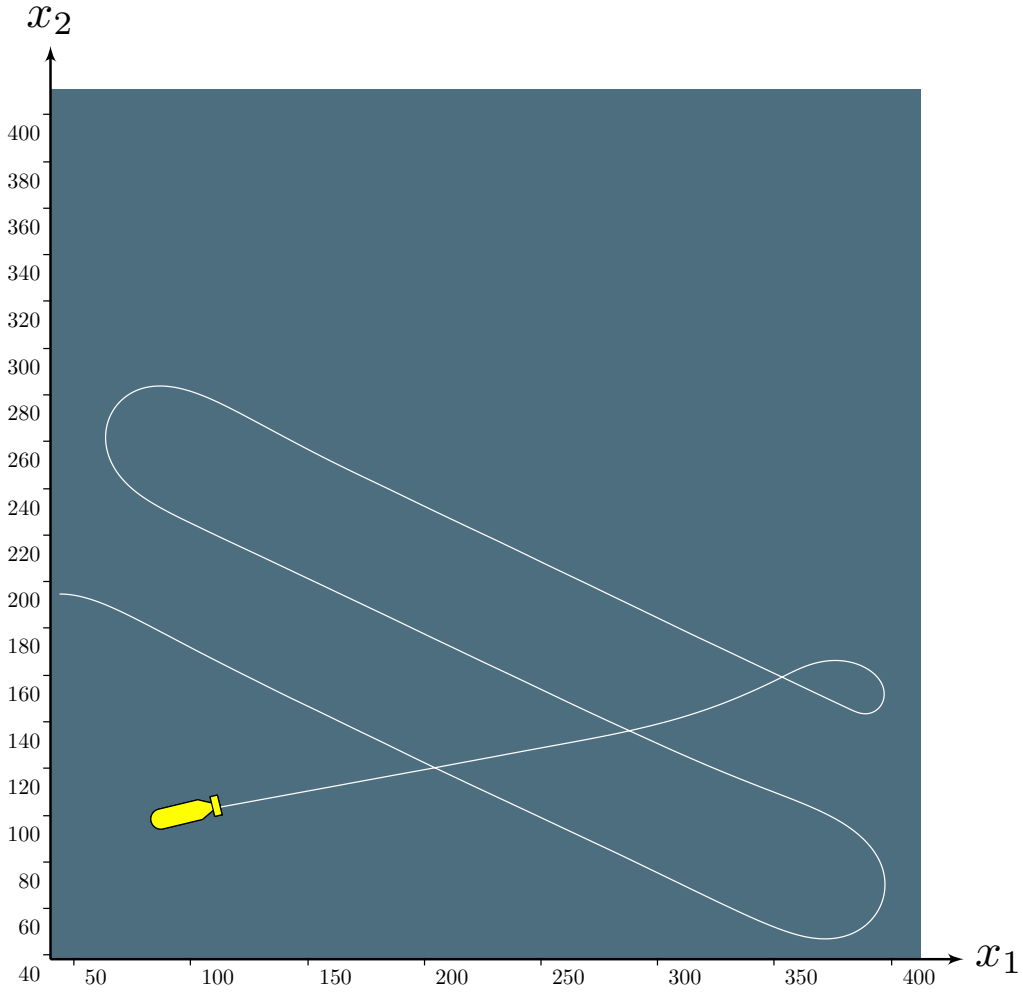


- new variable: **trajectory** $x(\cdot)$
- new domain (set): **tube** $[x](\cdot)$, interval of trajectories

Dead reckoning with actual data

Video

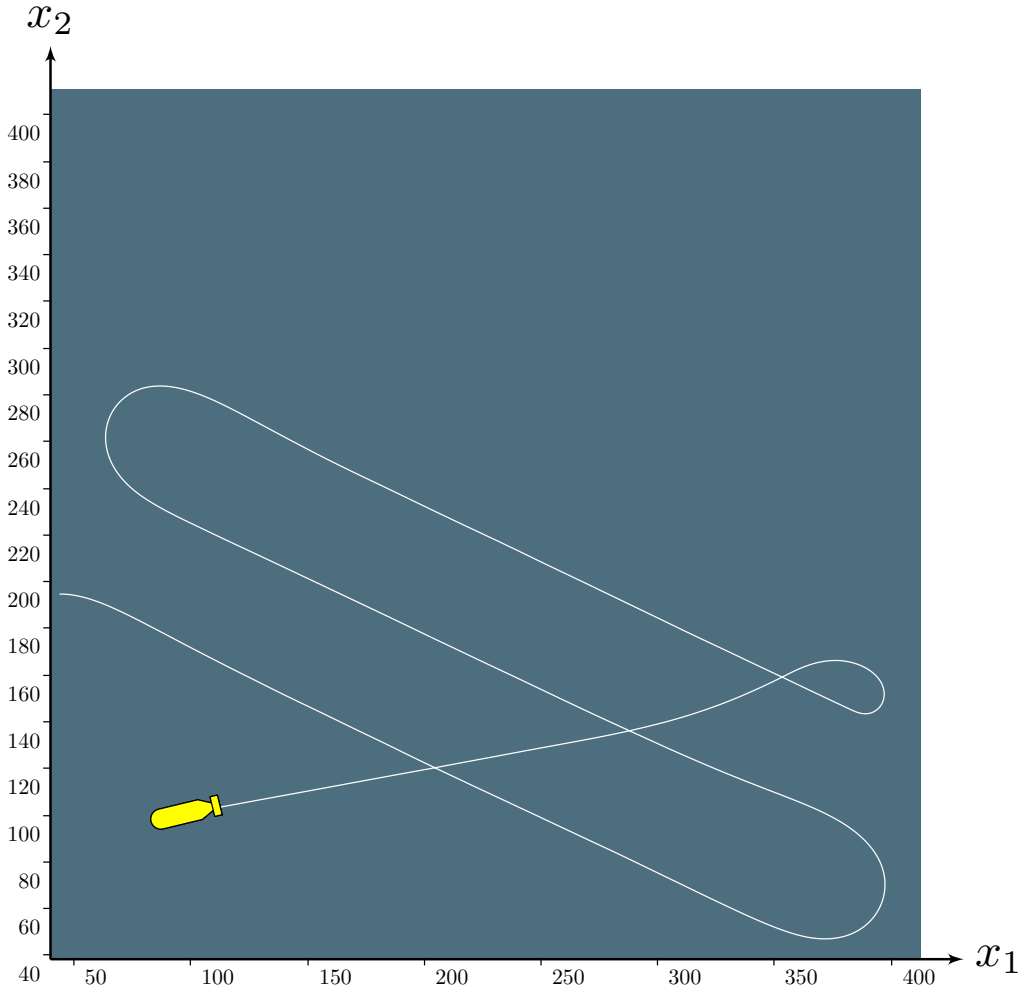
Dynamic state estimation



State estimation:

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

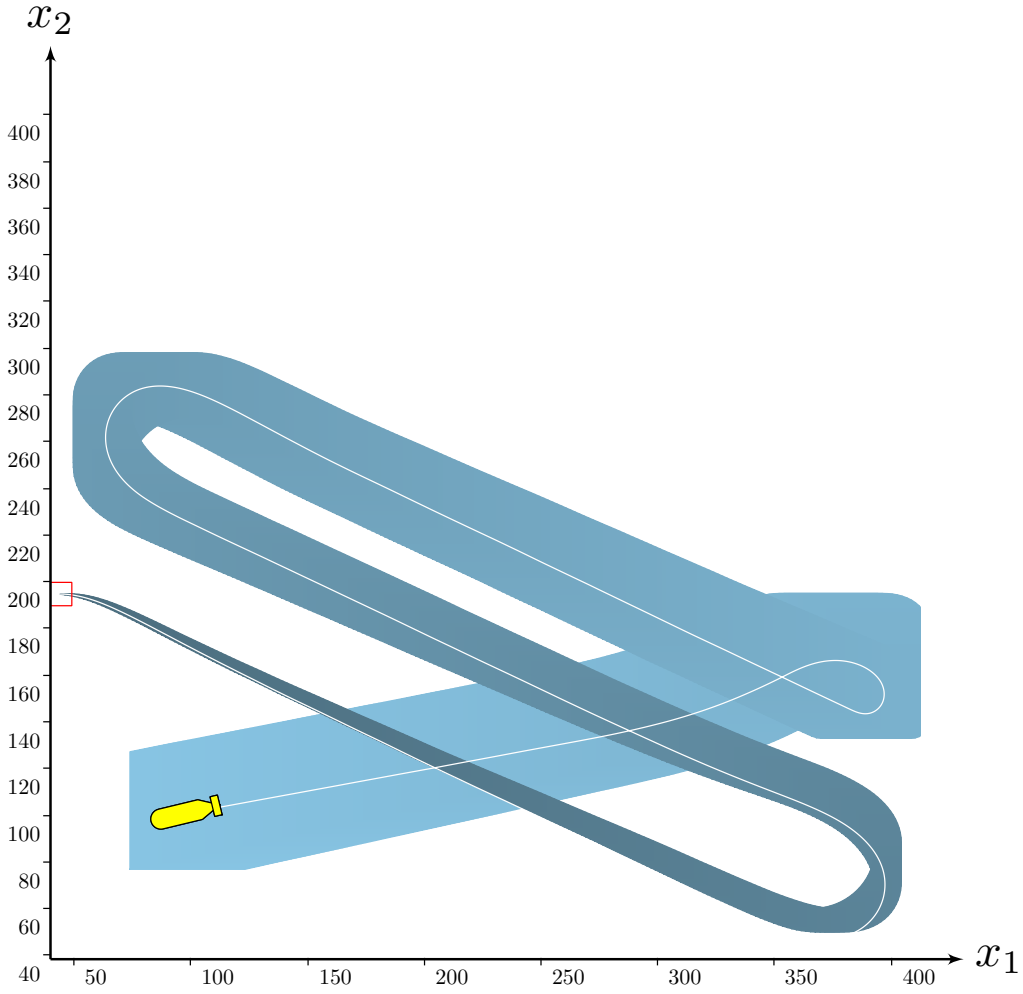
Dynamic state estimation



State estimation:

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

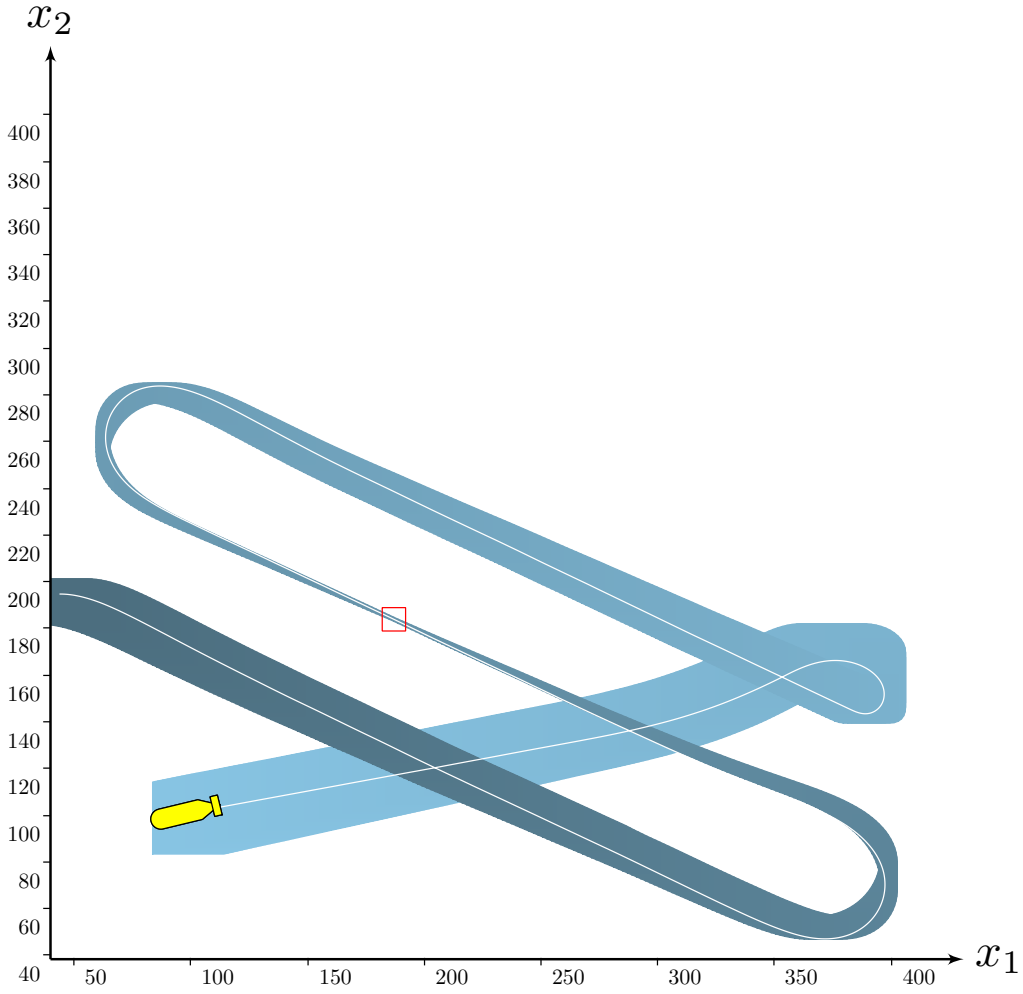
Dynamic state estimation



State estimation:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \\ \dot{\mathbf{x}}(t) = \mathbf{v}(t) \\ \mathbf{x}(t_0) \in [\mathbf{x}_0] \end{cases}$$

Dynamic state estimation



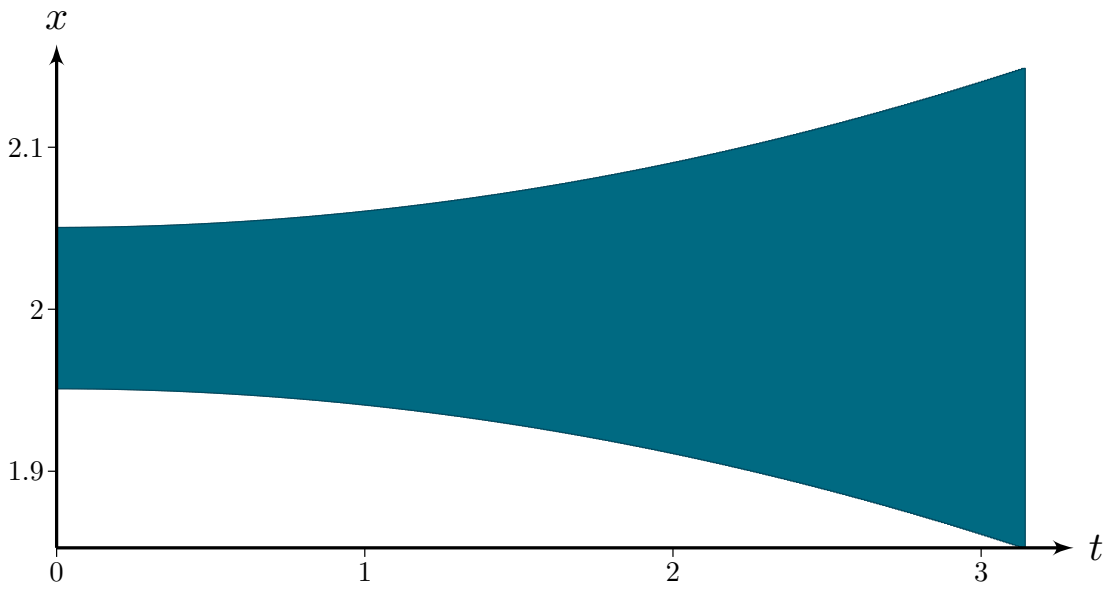
State estimation:

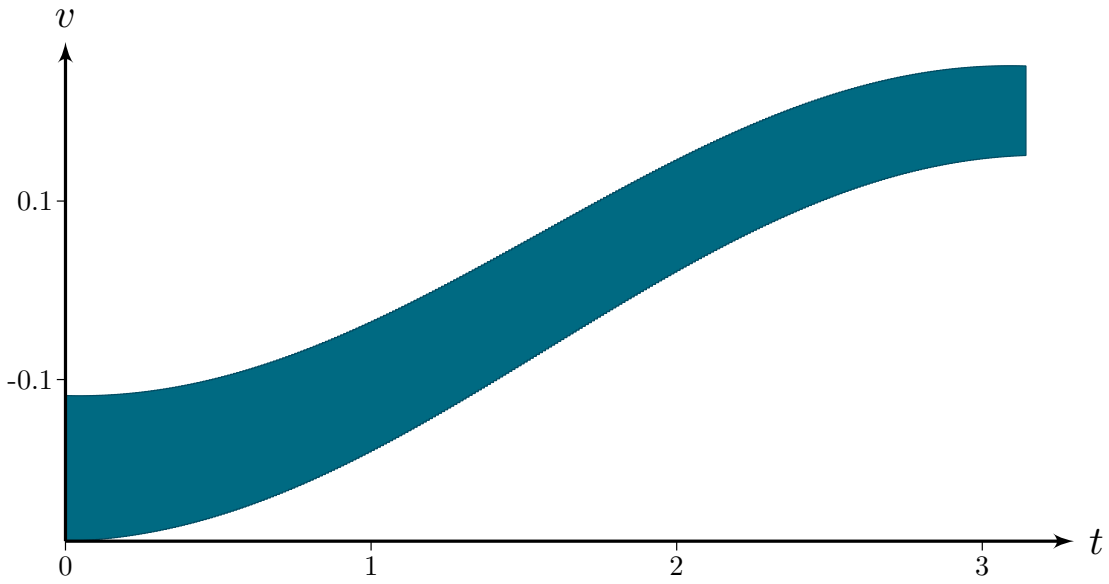
$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \\ \dot{\mathbf{x}}(t) = \mathbf{v}(t) \\ \mathbf{x}(t_1) \in [\mathbf{x}_1] \end{cases}$$

Derivative constraint

Differential constraint:

- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- one trajectory and its derivative





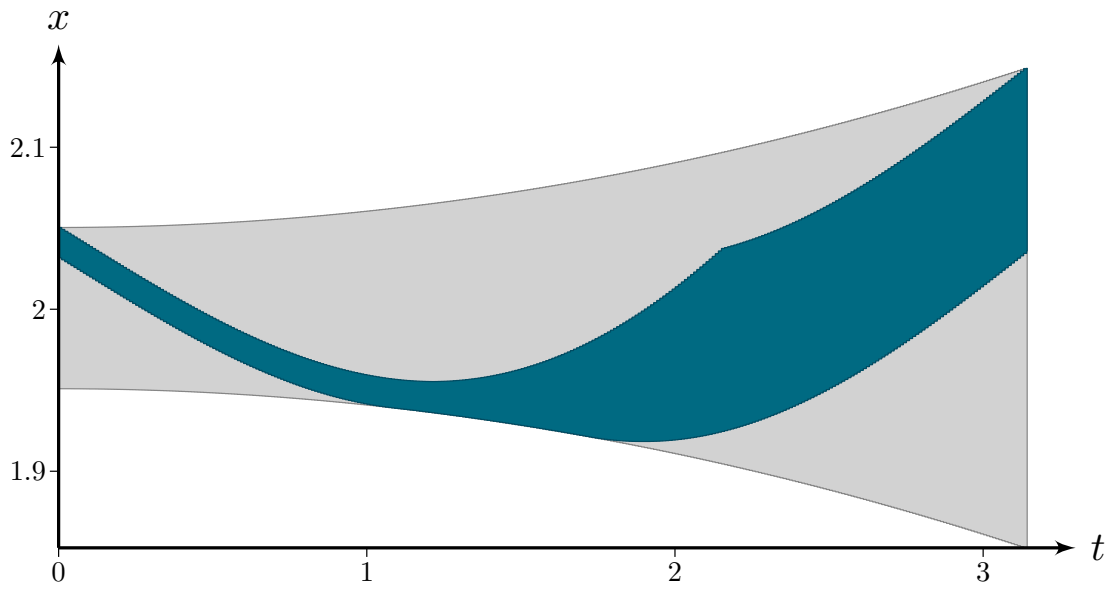
Derivative constraint

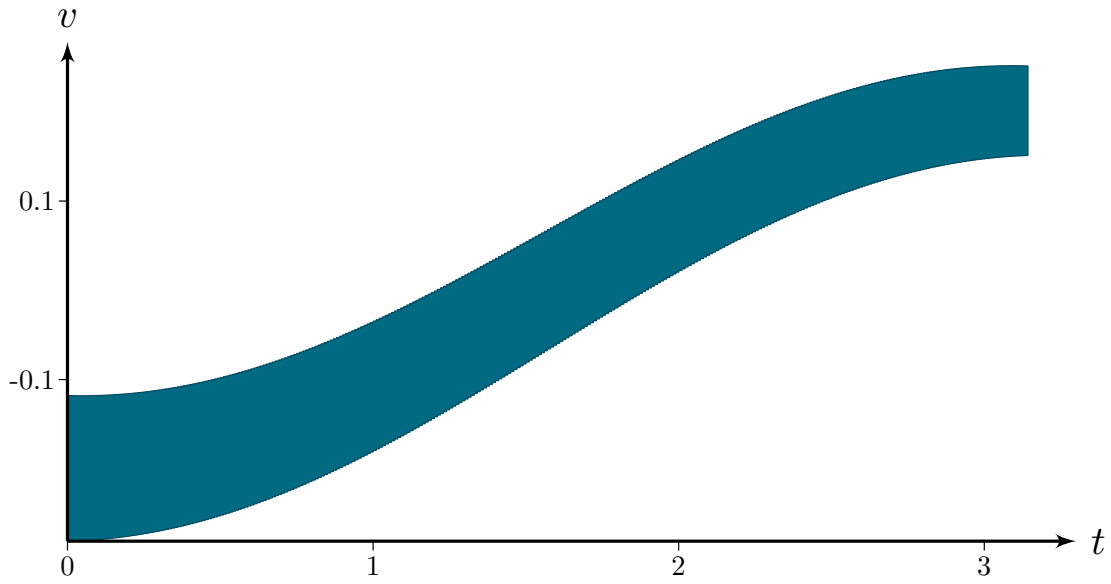
Differential constraint:

- $\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$
- one trajectory and its derivative

Related contractor:

- $\mathcal{C}_{\texttt{deriv}}\big([\mathbf{x}](\cdot), [\mathbf{v}](\cdot)\big)$





Guaranteed computation of robot trajectories

Rohou, Jaulin, Mihaylova, Le Bars, Veres

Robotics and Autonomous Systems, 2017

Trajectory evaluation constraint

Trajectory evaluation $\begin{cases} \mathbf{z} = \mathbf{y}(t) \\ \\ \end{cases}$

Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

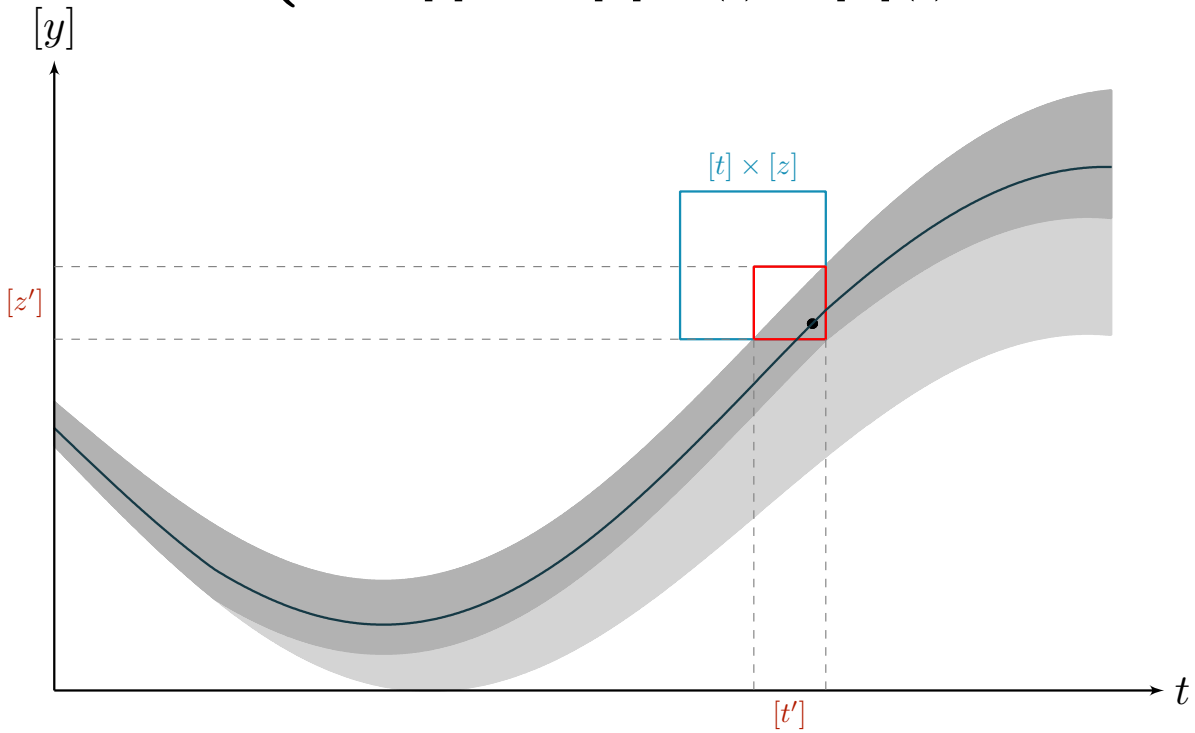
Trajectory evaluation constraint

$$\text{Trajectory evaluation} \begin{cases} \mathbf{z} = \mathbf{y}(t) \\ \\ t \in [t], \mathbf{z} \in [\mathbf{z}], \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot) \end{cases}$$

Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Trajectory evaluation constraint

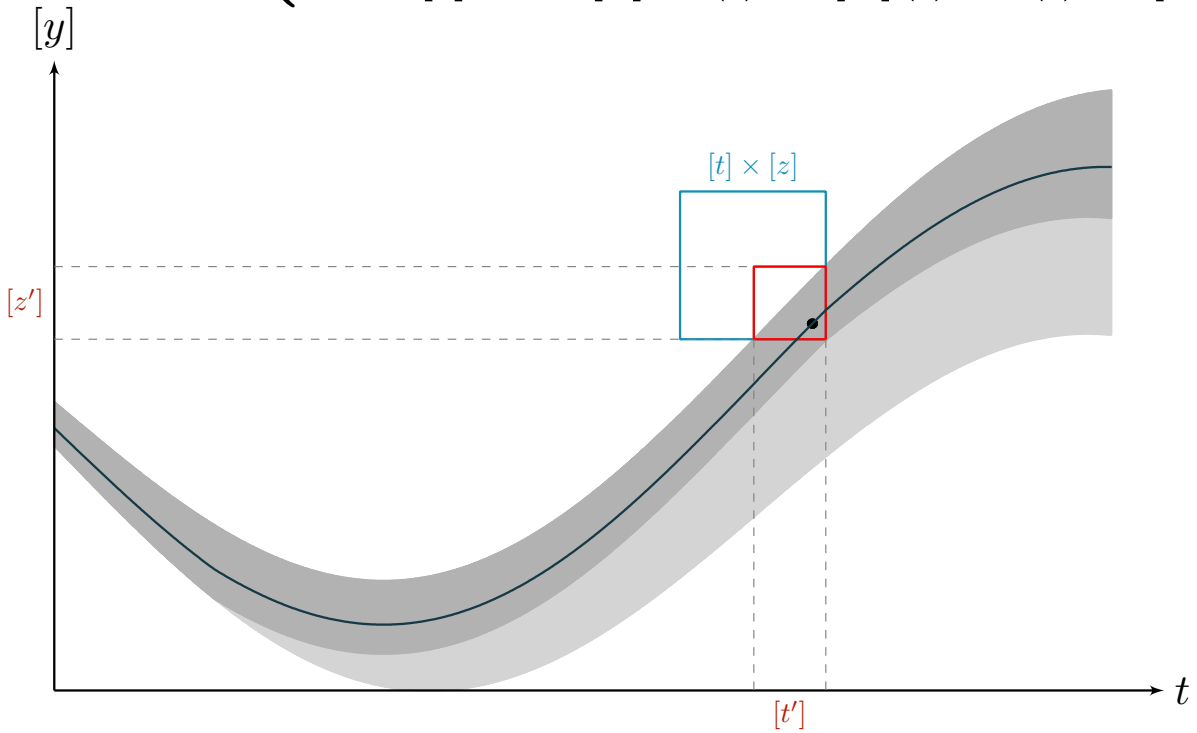
Trajectory evaluation $\begin{cases} \mathbf{z} = \mathbf{y}(t) \\ t \in [t], \mathbf{z} \in [\mathbf{z}], \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot) \end{cases}$



Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Trajectory evaluation constraint

Trajectory evaluation $\begin{cases} \mathbf{z} = \mathbf{y}(t) \\ \dot{\mathbf{y}}(\cdot) = \mathbf{w}(\cdot) \\ t \in [t], \mathbf{z} \in [\mathbf{z}], \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot), \mathbf{w}(\cdot) \in [\mathbf{w}](\cdot) \end{cases}$

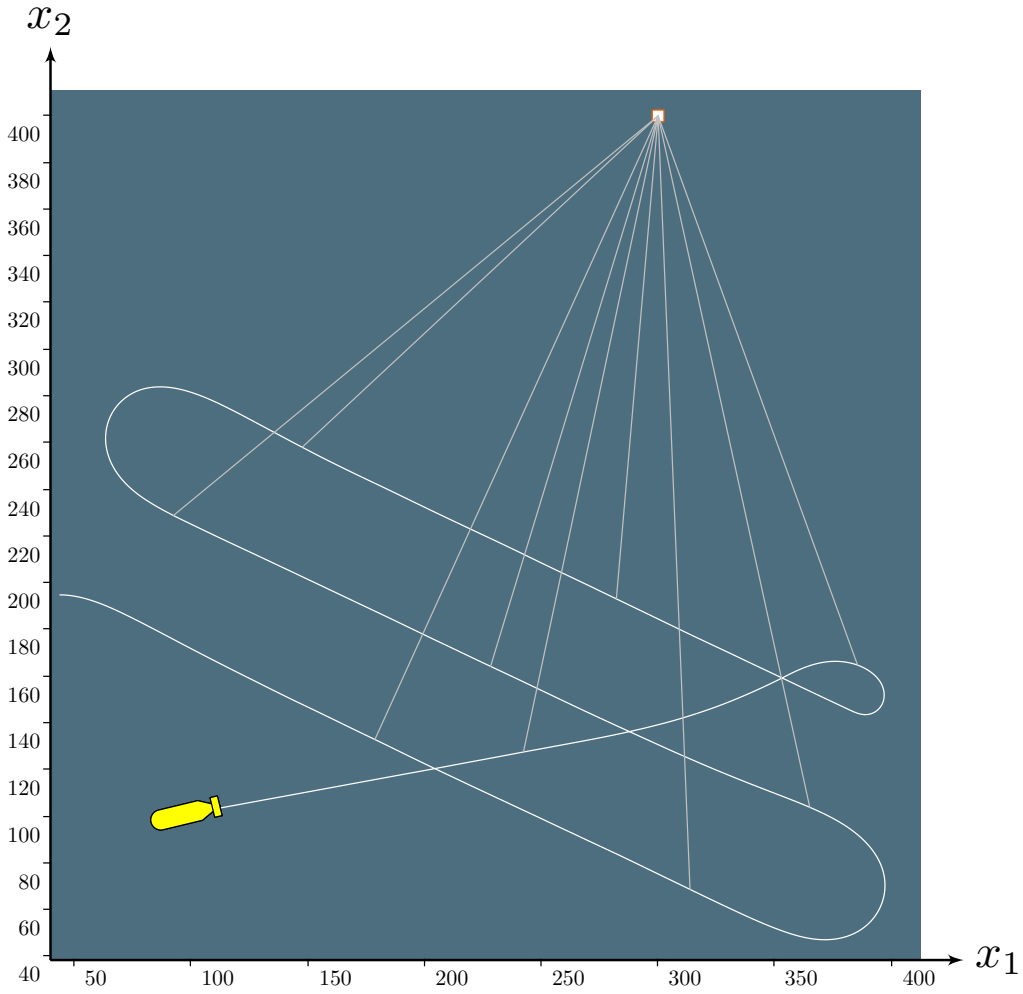


Contractor programming: $\mathcal{C}_{\text{eval}}\big([t], [\mathbf{z}], [\mathbf{y}](\cdot), [\mathbf{w}](\cdot)\big)$

Reliable non-linear state estimation involving time uncertainties
Rohou, Jaulin, Mihaylova, Le Bars, Veres *Automatica*, 2018

Dynamic state estimation

Considering **range-only measurements** from a known beacon.

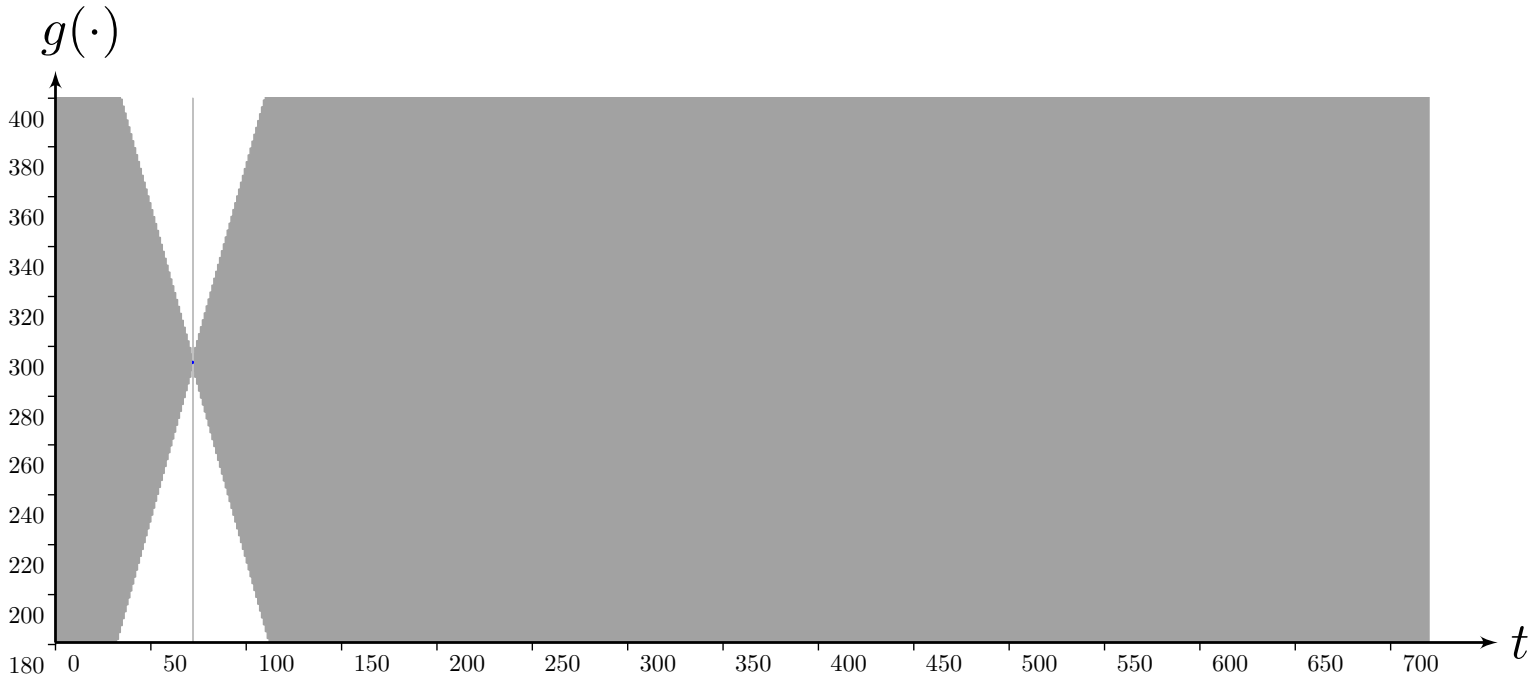


Nonlinear state estimation:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \\ y_i = g\big(\mathbf{x}(t_i), \mathbf{b}\big) \end{cases}$$

Exteroceptive measurements

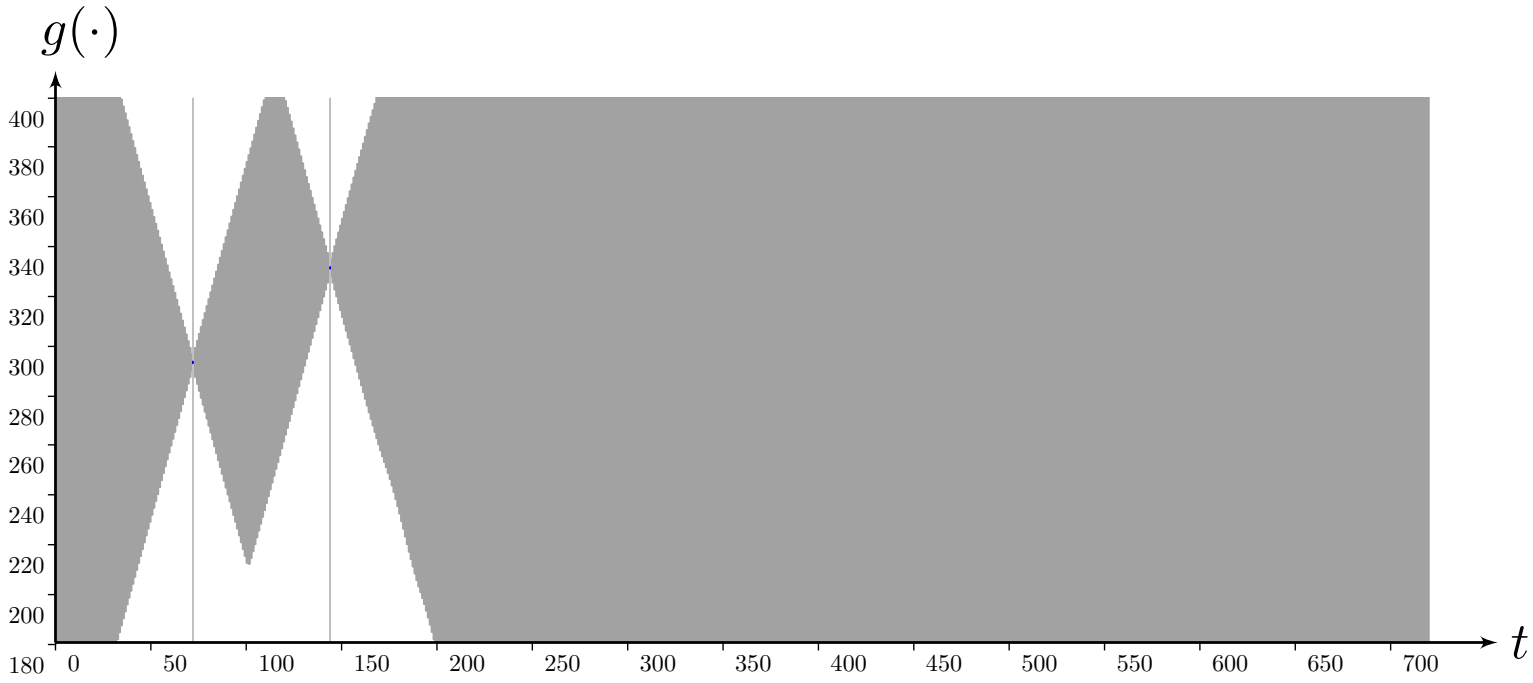
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 1 range-only measurement from the beacon.

Exteroceptive measurements

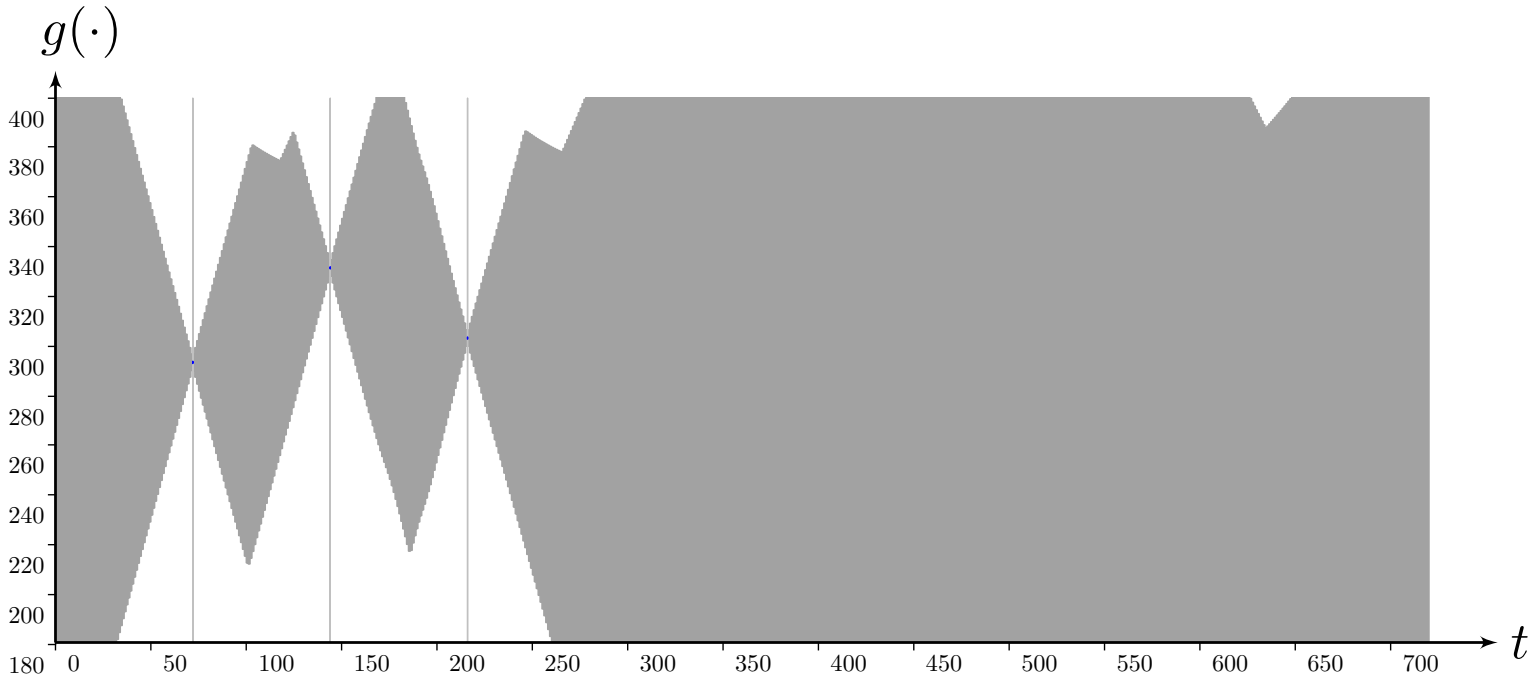
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 2 range-only measurements from the beacon.

Exteroceptive measurements

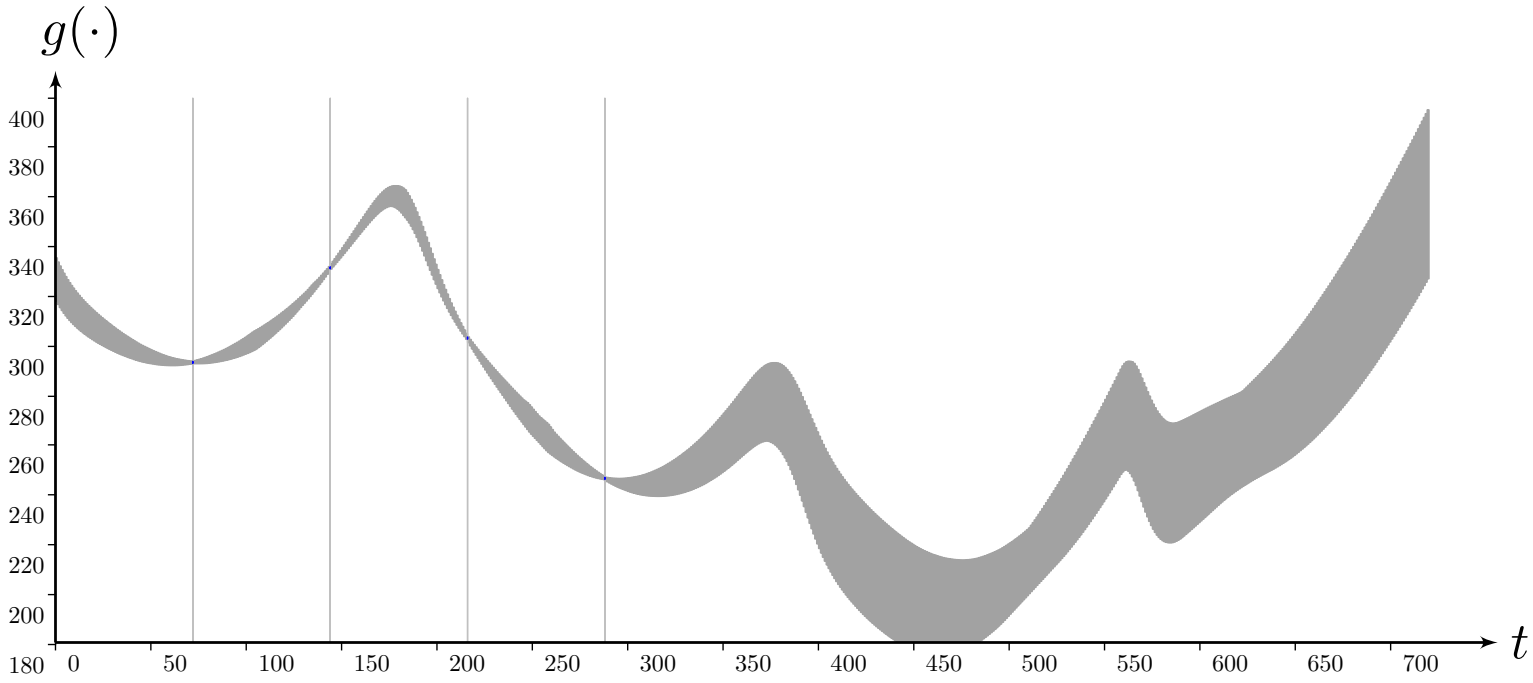
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 3 range-only measurements from the beacon.

Exteroceptive measurements

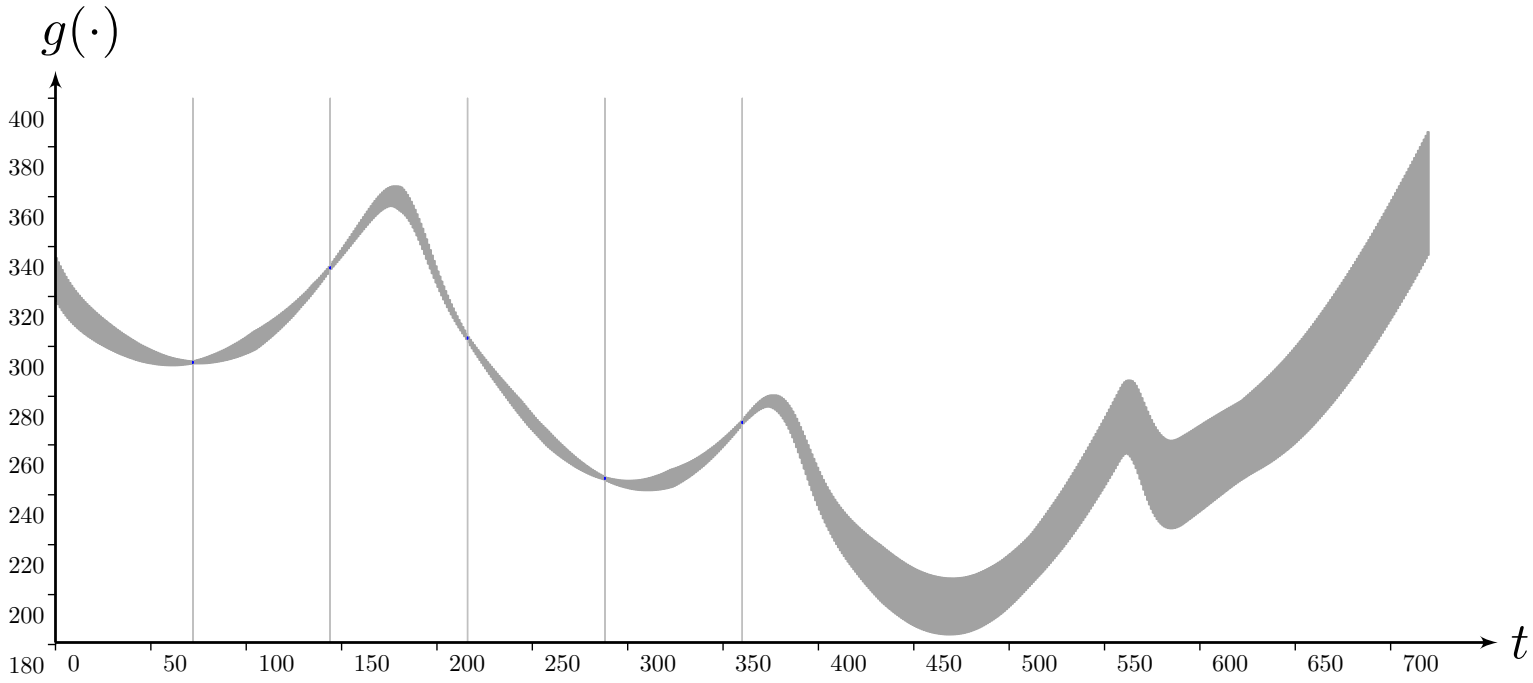
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 4 range-only measurements from the beacon.

Exteroceptive measurements

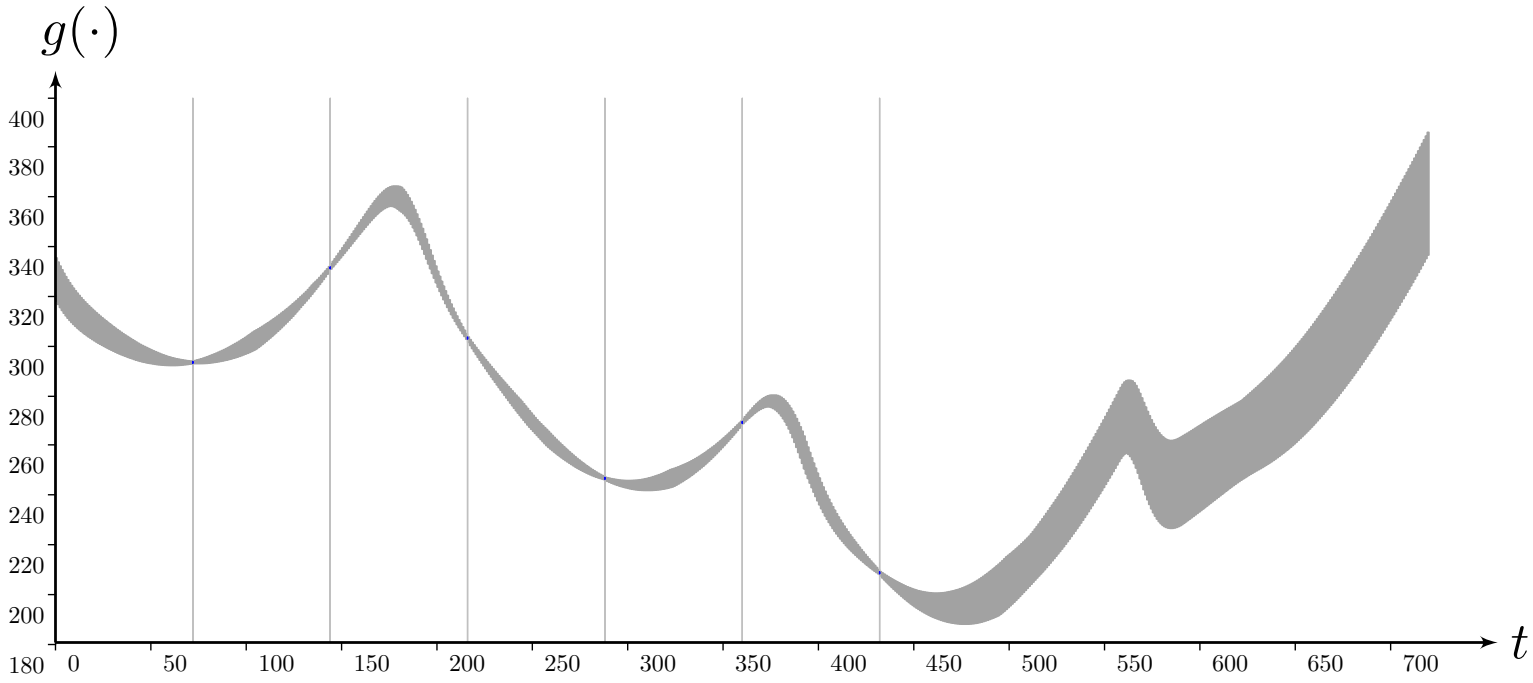
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 5 range-only measurements from the beacon.

Exteroceptive measurements

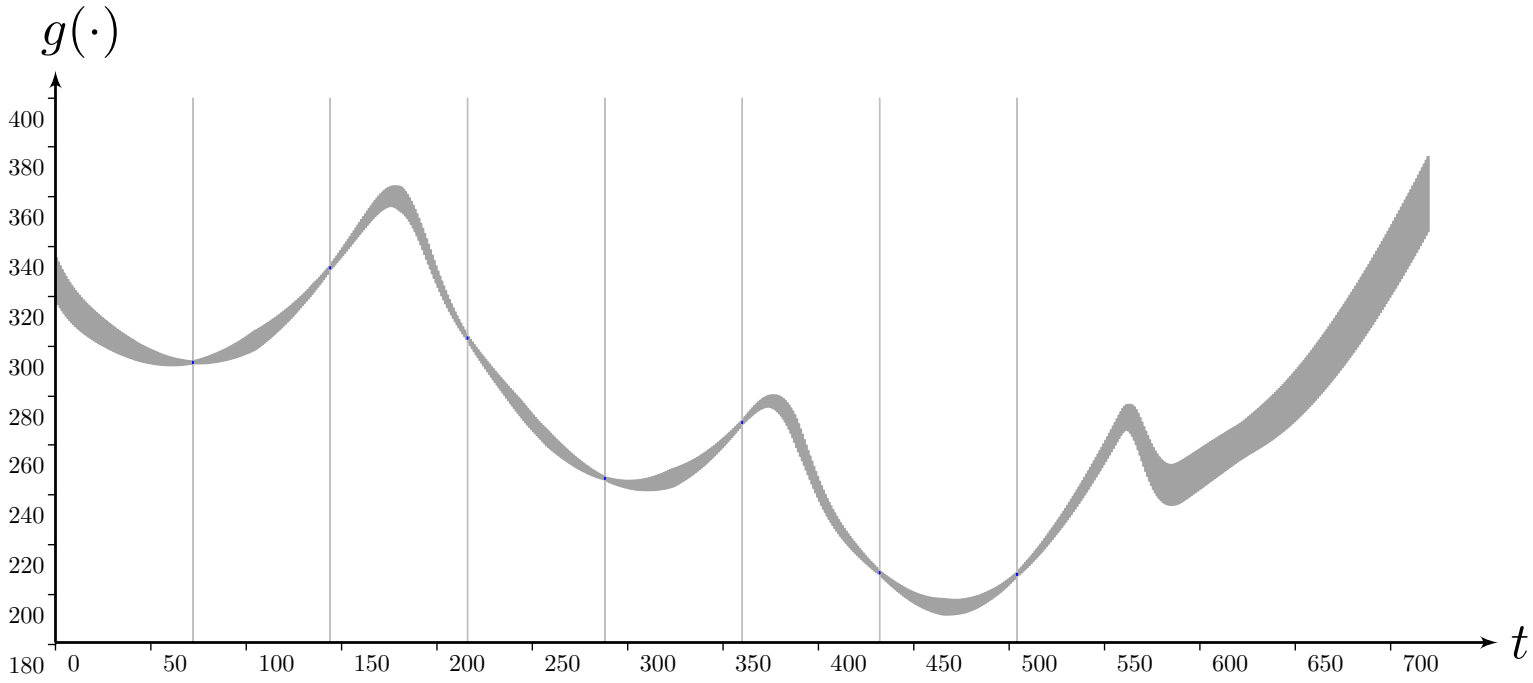
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 6 range-only measurements from the beacon.

Exteroceptive measurements

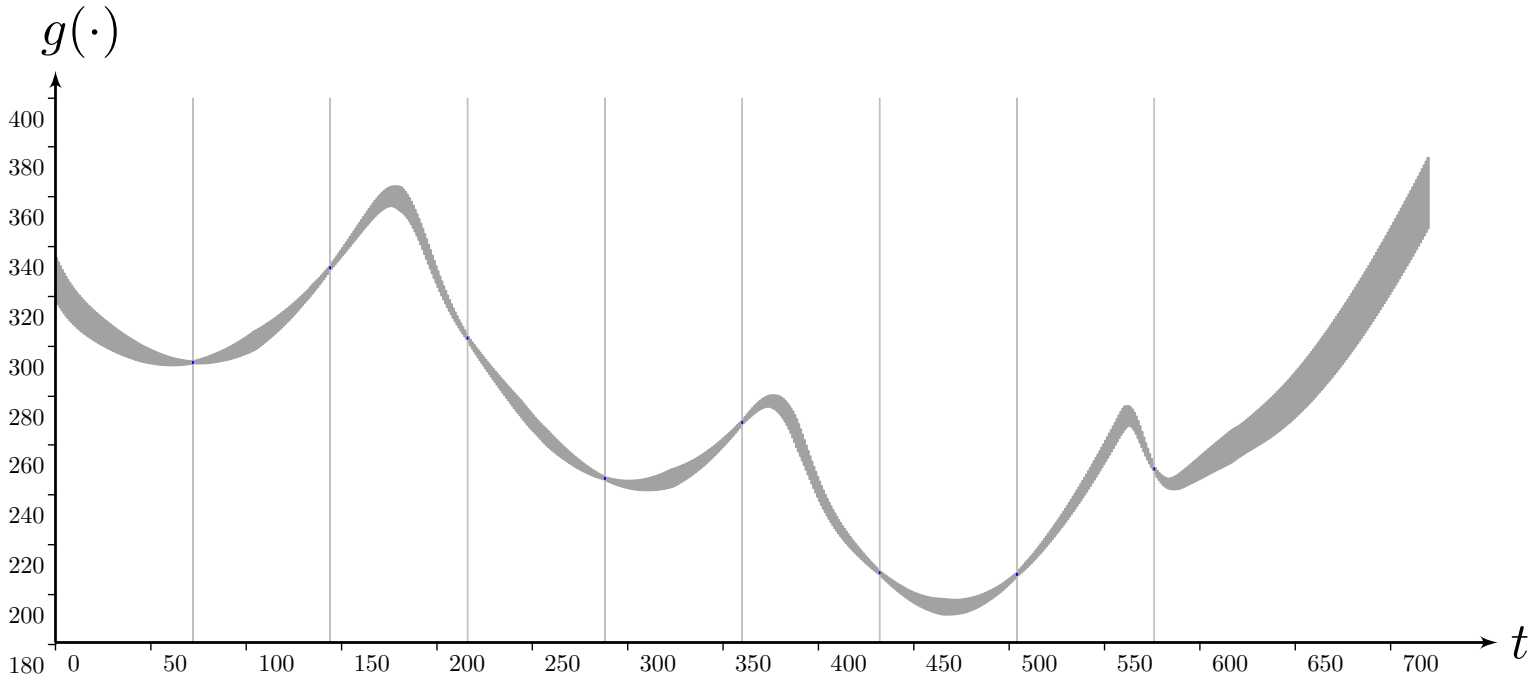
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 7 range-only measurements from the beacon.

Exteroceptive measurements

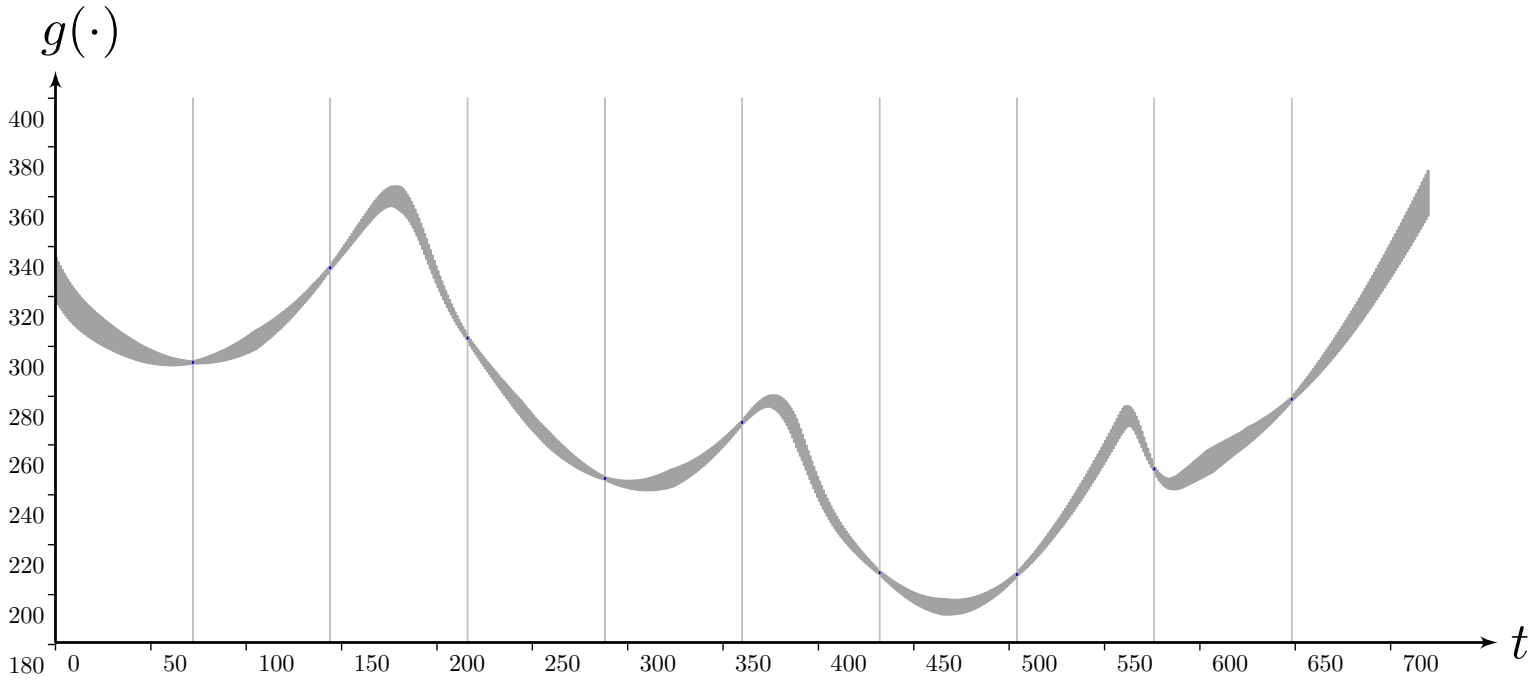
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 8 range-only measurements from the beacon.

Exteroceptive measurements

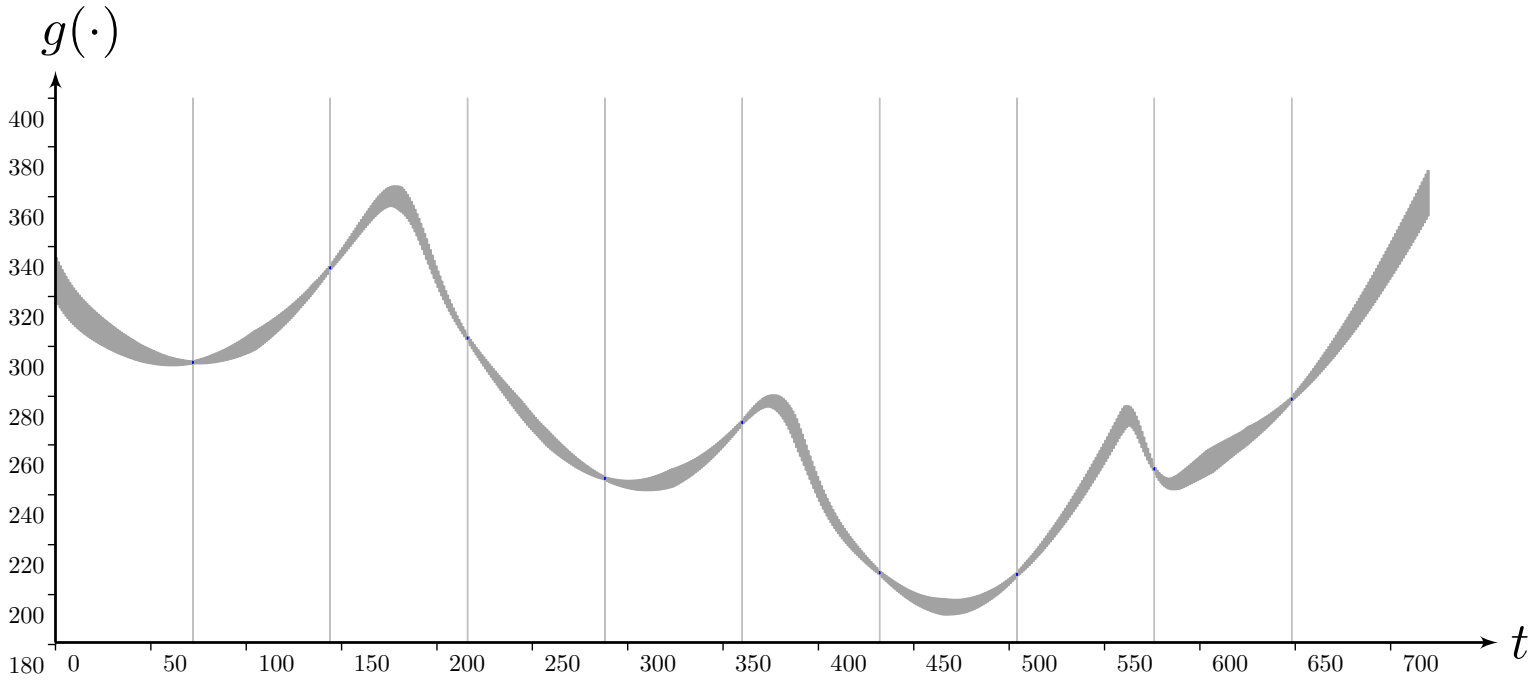
Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



Observation tube, considering 9 range-only measurements from the beacon.

Exteroceptive measurements

Creating another tube $[g](\cdot)$ that will be **constrained by measurements**.



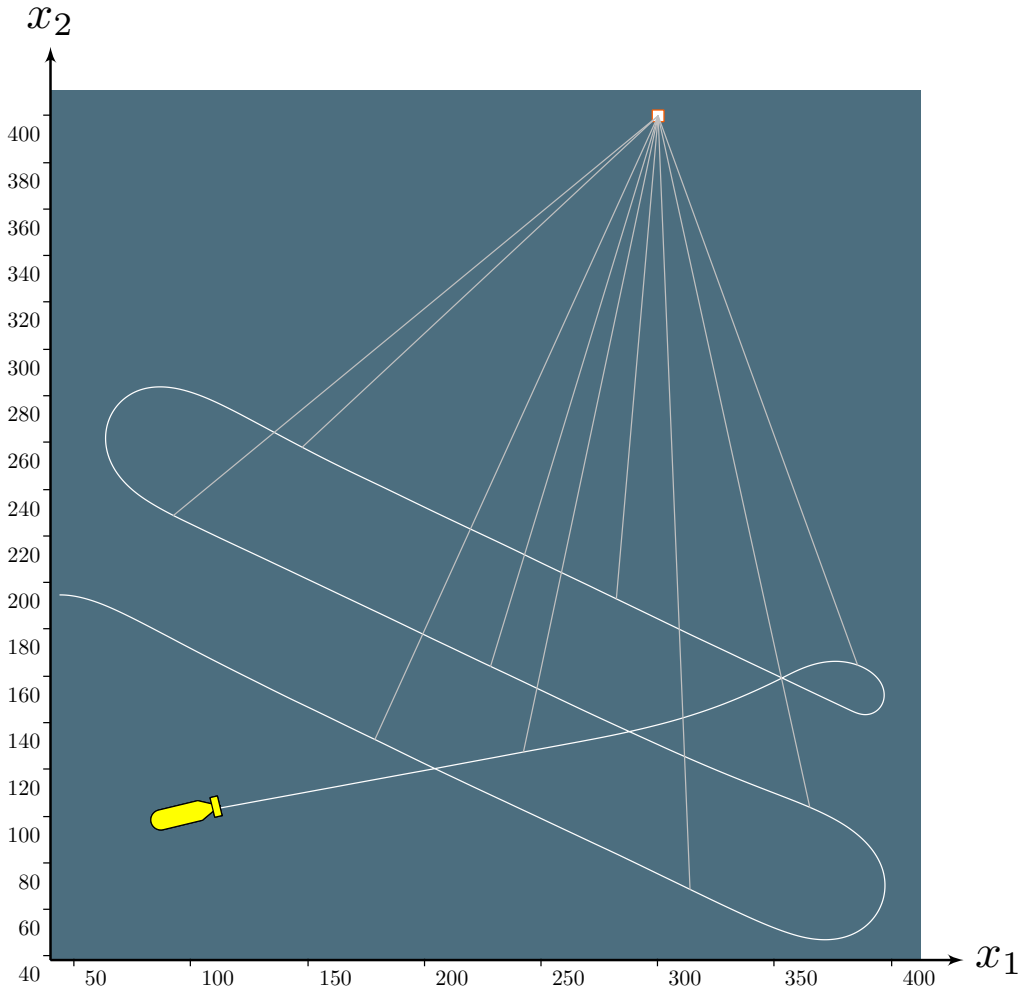
Observation tube, considering 9 range-only measurements from the beacon.

Then the state tube $[\mathbf{x}](\cdot)$ will be constrained by $[g](\cdot)$.

$$\mathcal{L}_g : \quad g(\cdot) = \sqrt{\left(x_1(\cdot) - \mathcal{B}_1\right)^2 + \left(x_2(\cdot) - \mathcal{B}_2\right)^2}.$$

Dynamic state estimation

Considering **range-only measurements** from a known beacon.

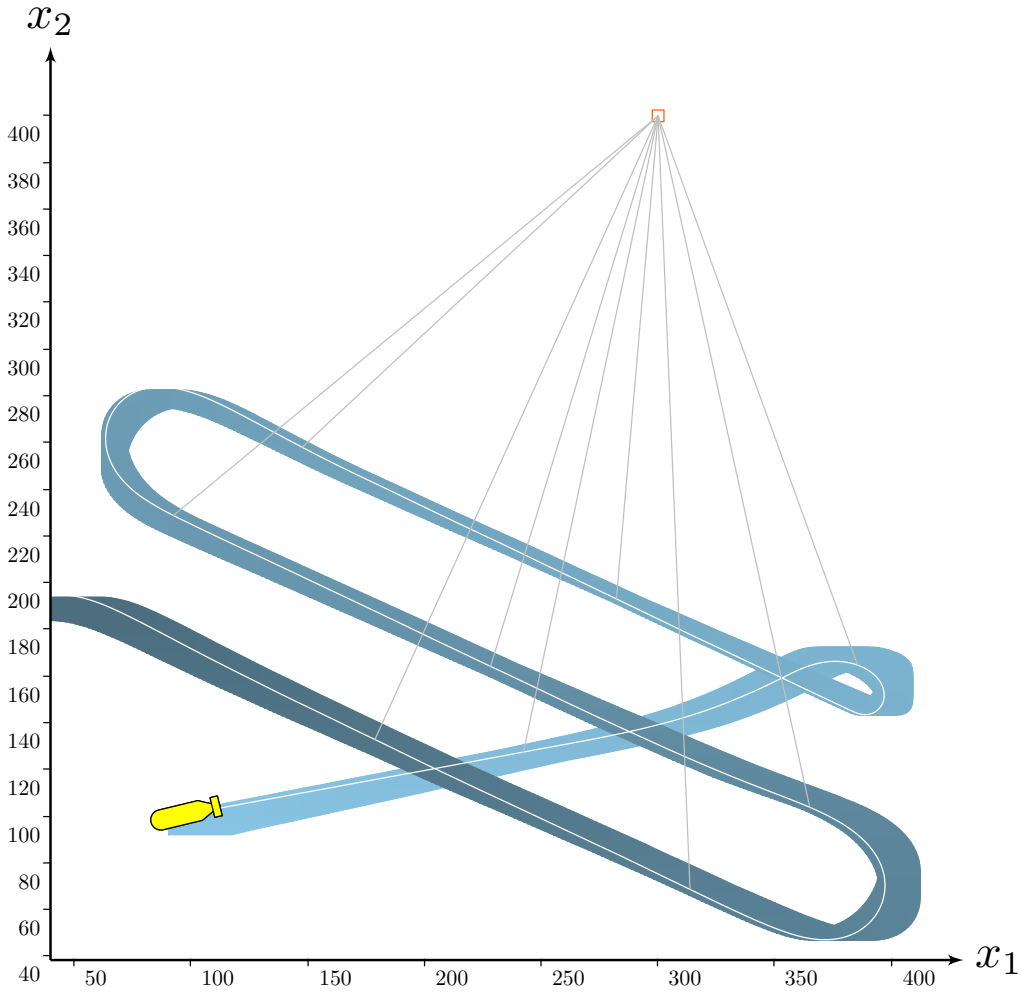


Nonlinear state estimation:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \\ y_i = g\big(\mathbf{x}(t_i), \mathbf{b}\big) \end{cases}$$

Dynamic state estimation

Considering **range-only measurements** from a known beacon.



Nonlinear state estimation:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) \\ y_i = g\big(\mathbf{x}(t_i), \mathbf{b}\big) \end{cases}$$

Other example: terrain based navigation

Video

Assets of constraint programming $\times$ interval analysis

Assets of constraint programming coupled with interval analysis:

- **simplicity** of the approach

Assets of constraint programming $\times$ interval analysis

Assets of constraint programming coupled with interval analysis:

- **simplicity** of the approach
- **reliability** of the results: no solution can be lost

Assets of constraint programming $\times$ interval analysis

Assets of constraint programming coupled with interval analysis:

- **simplicity** of the approach
- **reliability** of the results: no solution can be lost
- focus on **the *what* instead of the *how***

Assets of constraint programming $\times$ interval analysis

Assets of constraint programming coupled with interval analysis:

- **simplicity** of the approach
- **reliability** of the results: no solution can be lost
- focus on **the *what* instead of the *how***
- **complex systems** easily handled

Assets of constraint programming × interval analysis

Assets of constraint programming coupled with interval analysis:

- **simplicity** of the approach
- **reliability** of the results: no solution can be lost
- focus on **the *what* instead of the *how***
- **complex systems** easily handled

Video

Section 4

The Codac library

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 $[x](\cdot)$

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 $\mathbb{X} \in [\mathbb{X}] = [\mathbb{X}^-, \mathbb{X}^+]$

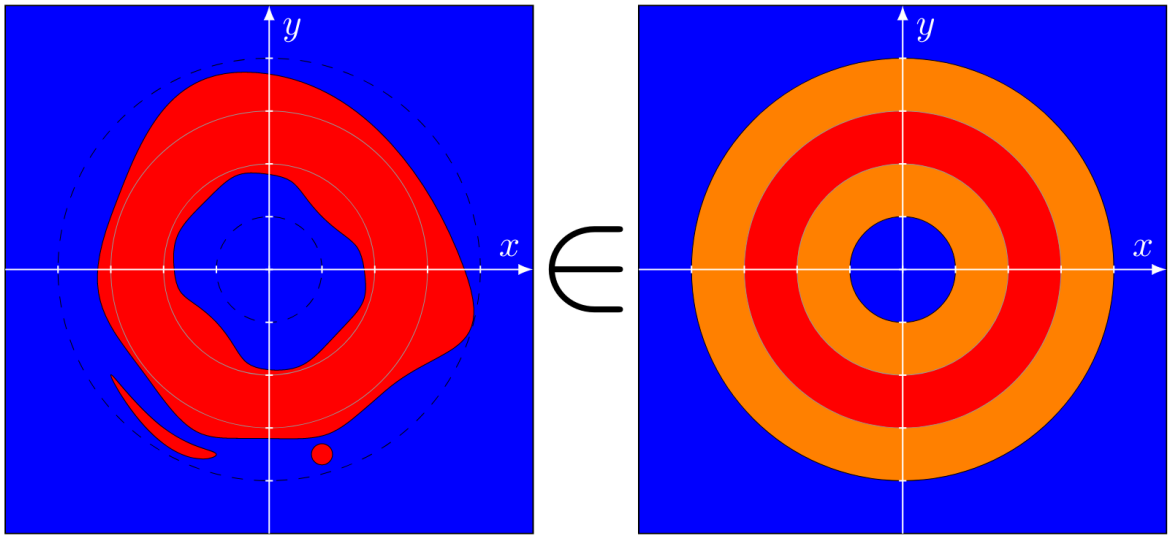


Illustration of a thickset (right-hand side)
for enclosing and uncertain red set (left-hand side)

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

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.*

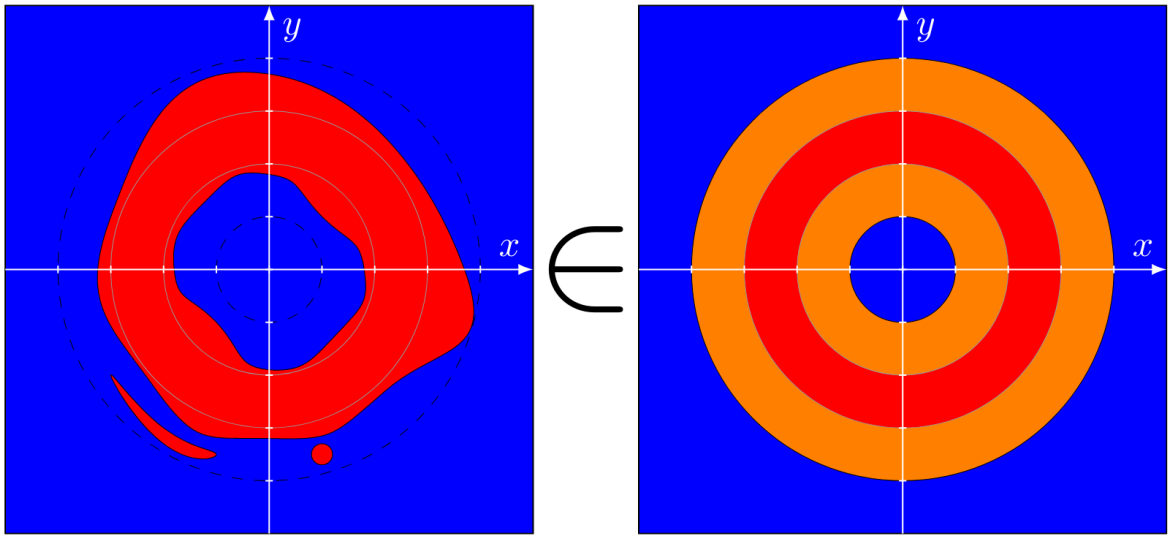
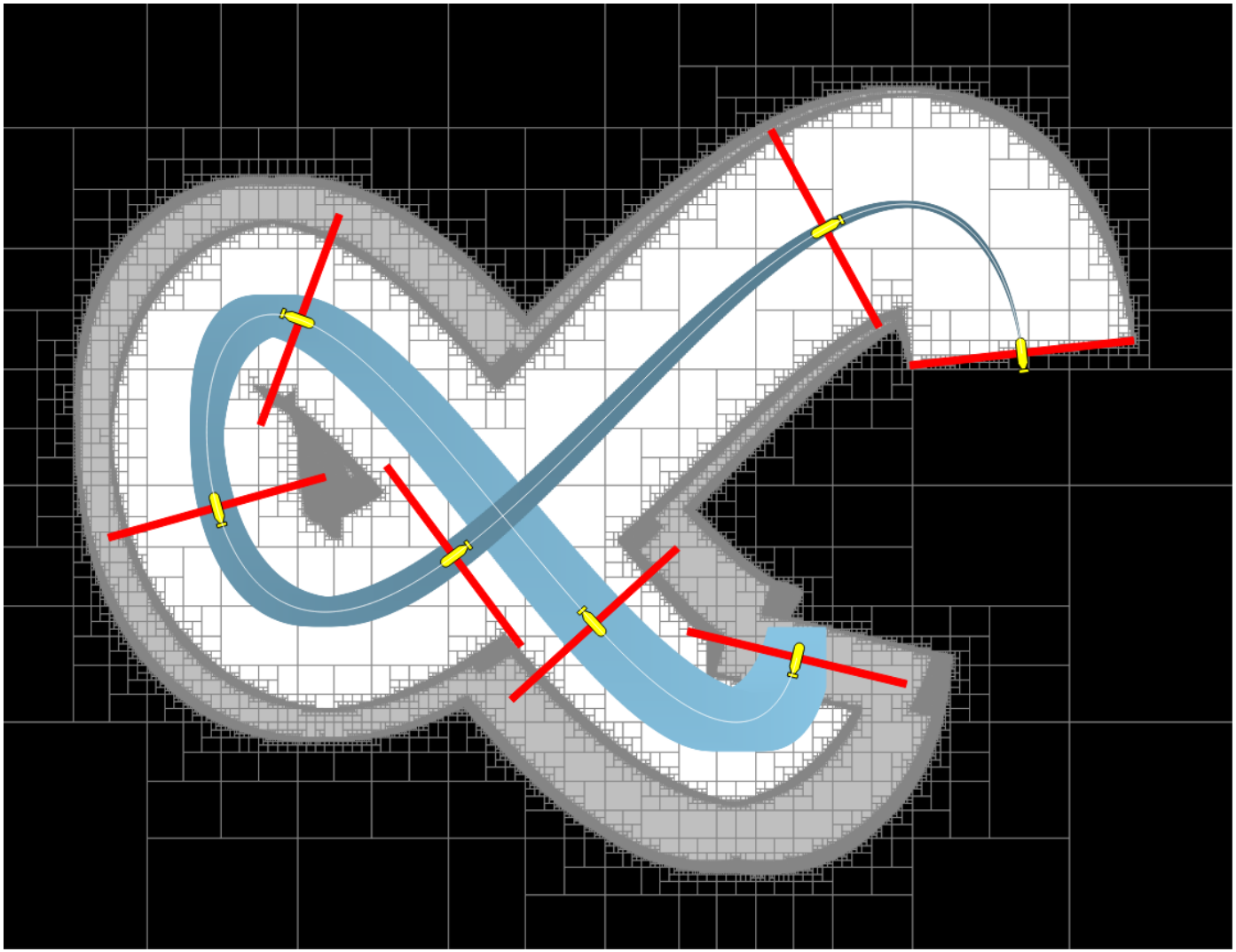


Illustration of a thickset (right-hand side)
for enclosing and uncertain red set (left-hand side)

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

Example of tubes and thicksets



Computing a Guaranteed Approximation of the Zone Explored by a Robot
Desrochers, Jaulin. *IEEE Transaction on Automatic Control. Volume 62, Issue 1, pages 425-430*, 2017

Codac: Catalog Of Domains And Contractors

Several types of **domains**:

- `Interval`, `IntervalVector`, `IntervalMatrix`
- `Tube`, `TubeVector`, `Slice`
- `Thickset`
- `Ellipsoid` (next release)
- ...

Codac: Catalog Of Domains And Contractors

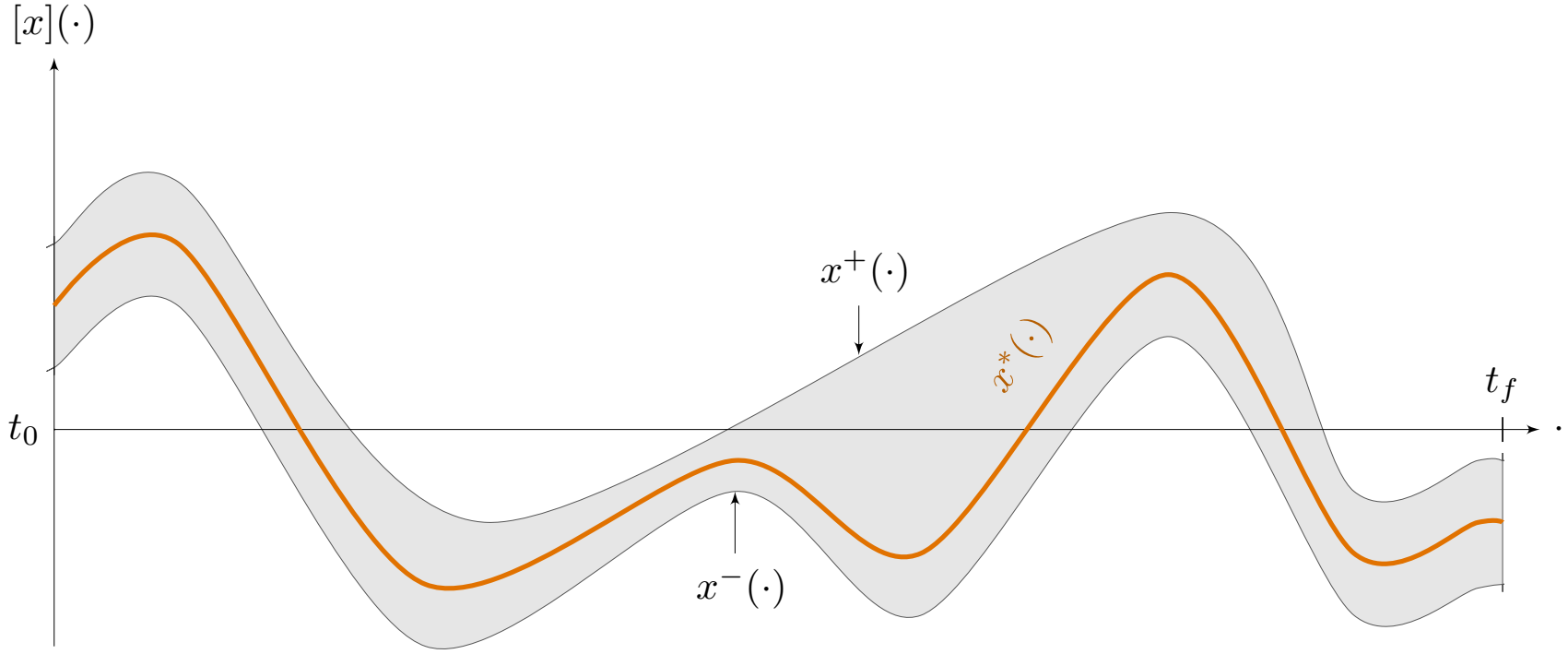
Several types of **domains**:

- `Interval`, `IntervalVector`, `IntervalMatrix`
- `Tube`, `TubeVector`, `Slice`
- `Thickset`
- `Ellipsoid` (next release)
- ...

Contractors for various constraints:

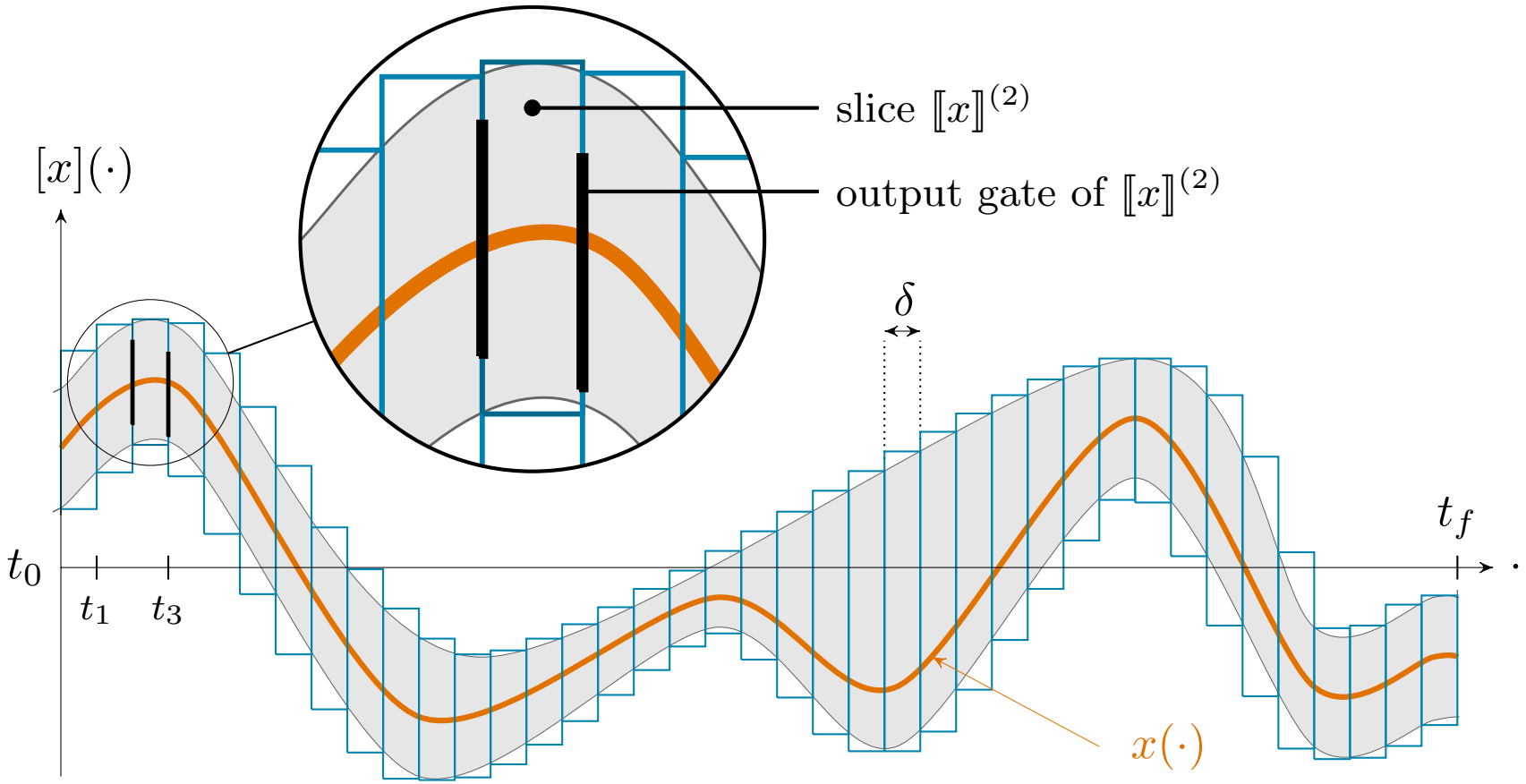
- non-linear constraints $\mathbf{f}(\mathbf{x}) = \mathbf{0}$
- geometric constraints: distance, polar equation, circles, ...
- differential equations: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$
- time uncertainties: $\mathbf{y} = \mathbf{x}(t),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

Domains for trajectories: tubes



Example of scalar tube: interval of two trajectories

Domains for trajectories: tubes



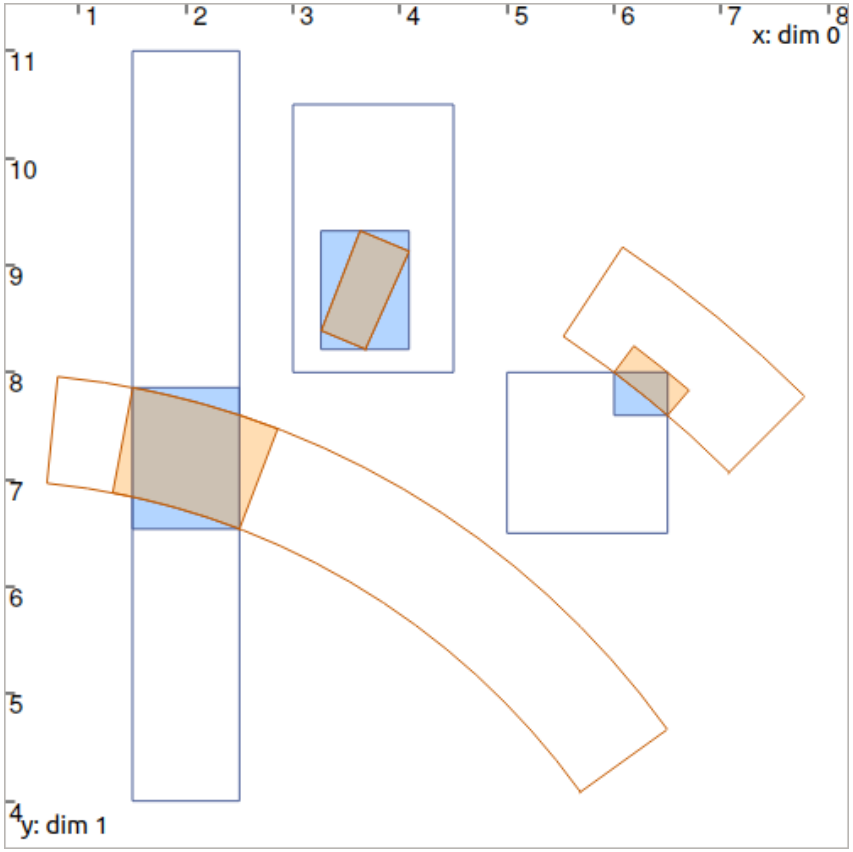
Computer implementation (http://codac.io)

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

The library is open source and available:

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

http://www.codac.io

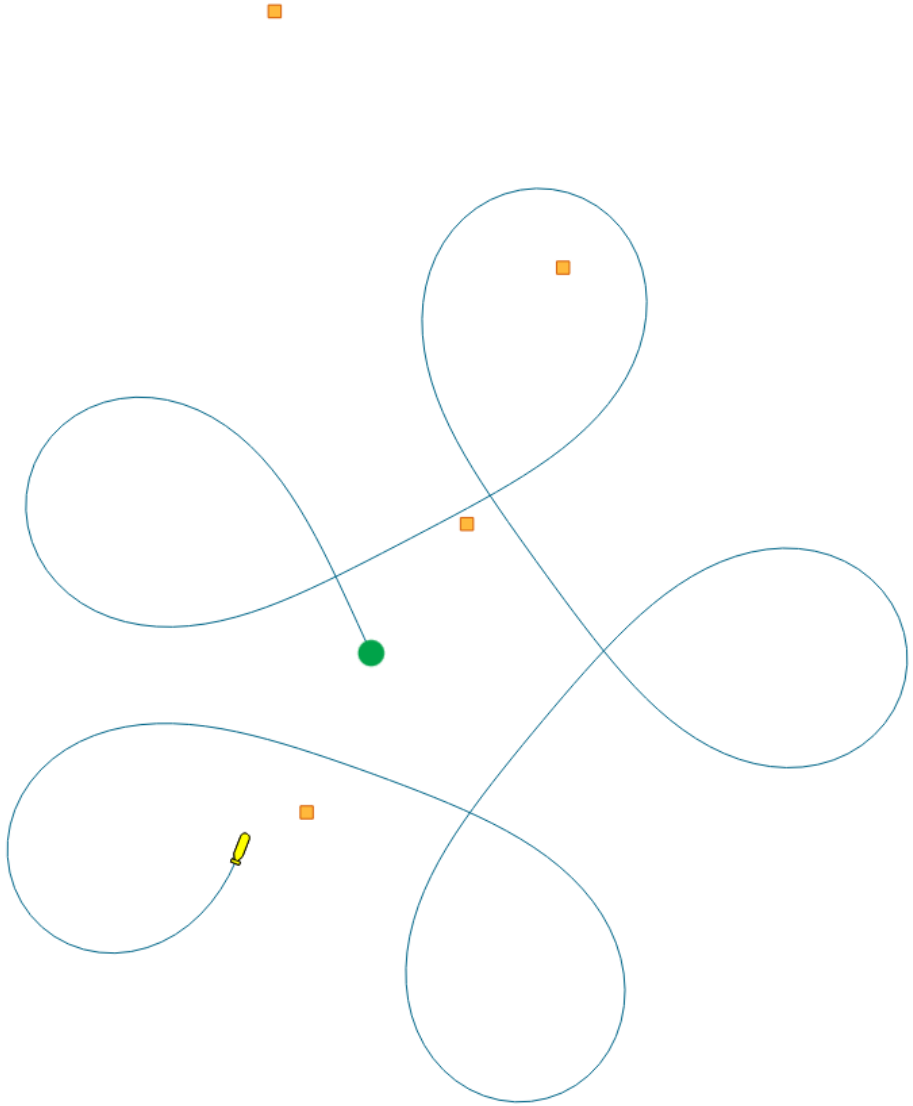
Current list of contributors

- Benoît Desrochers
- Luc Jaulin
- Gilles Chabert
- Auguste Bourgois
- Julien Damers
- Thomas Le Mézo
- Raphael Voges
- Cyril Bouvier
- Andreas Rauh
- Fabrice Le Bars
- Quentin Brateau
- Damien Massé
- Bertrand Neveu
- Peter Franek
- Gilles Trombettoni
- Aaronkumar Ehambram
- Verlein Radwan
- Mohamed Saad Ibn Seddik

Section 5

Application: range-only SLAM

Simultaneous Localization And Mapping



Formalization

SLAM: Simultaneous Localization And Mapping.
Classically, we have:

$$\begin{cases} \mathbf{x}(0) = \mathbf{0} & \text{(initial state)} \\ \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(evolution)} \end{cases}$$

With:

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

Formalization

SLAM: Simultaneous Localization And Mapping.
Classically, we have:

$$\begin{cases} \mathbf{x}(0) = \mathbf{0} & \text{(initial state)} \\ \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(evolution)} \\ y_i = g\big(\mathbf{x}(t_i), \mathbf{b}_j\big) & \text{(observations)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, heading, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function (scalar, distance equation)
- y_i: scalar measurements (at t_i) (distance values)
- $\mathbf{b}_j$: unknown position of a landmark

Formalization

SLAM: Simultaneous Localization And Mapping.
Classically, we have:

$$\begin{cases} \mathbf{x}(0) = \mathbf{0} & \text{(initial state)} \\ \dot{\mathbf{x}}(t) = \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big) & \text{(evolution)} \\ y_i = g\big(\mathbf{x}(t_i), \mathbf{b}_j\big) & \text{(observations)} \end{cases}$$

With:

- $\mathbf{x}$: state vector (position, heading, ...)
- $\mathbf{u}$: input vector (command)
- $\mathbf{f}$: *evolution* function
- g: *observation* function (scalar, distance equation)
- y_i: scalar measurements (at t_i) (distance values)
- $\mathbf{b}_j$: unknown position of a landmark

Involved variables and domains

Variables:

- reals: $y_i \in \mathbb{R}$
- vectors: $\mathbf{b}_j \in \mathbb{R}^2$
- trajectories: $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$

Involved variables and domains

Variables:

- reals: $y_i \in \mathbb{R}$
- vectors: $\mathbf{b}_j \in \mathbb{R}^2$
- trajectories: $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$

Domains (envelopes) of the variables:

- intervals: $[y_i] \in \mathbb{IR}$
- boxes: $[\mathbf{b}_j] \in \mathbb{IR}^2$
- tubes: $[\mathbf{x}](\cdot) : \mathbb{R} \to \mathbb{IR}^n$

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

Elementary constraints:

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

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

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 $\rightarrow \mathcal{L}_{\mathrm{deriv}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

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 $\rightarrow \mathcal{L}_{\text{deriv}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

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 $\rightarrow \mathcal{L}_{\text{deriv}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$
- $y_i = g\big(\mathbf{p}_i, \mathbf{b}_j\big) \rightarrow$ distance constraint $\rightarrow \mathcal{L}_{\text{dist}}\big(\mathbf{p}_i, \mathbf{b}_j, y_i\big)$

Decomposition of the problem

System:

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

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

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 $\rightarrow \mathcal{L}_{\text{deriv}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$
- $y_i = g\big(\mathbf{p}_i, \mathbf{b}_j\big) \rightarrow$ distance constraint $\rightarrow \mathcal{L}_{\text{dist}}\big(\mathbf{p}_i, \mathbf{b}_j, y_i\big)$

Decomposition of the problem

System:

$$\begin{cases} \dot{\mathbf{x}}(\cdot) = \mathbf{f}\big(\mathbf{x}(\cdot)\big) \\ y_i = g\big(\mathbf{x}_{1,2}(t_i), \mathbf{b}_j\big) \end{cases}$$

$\mathbf{v}(\cdot)$ 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 $\rightarrow \mathcal{L}_{\text{deriv}}\big(\mathbf{x}(\cdot), \mathbf{v}(\cdot)\big)$
- $\mathbf{p}_i = \mathbf{x}_{1,2}(t_i) \rightarrow$ evaluation constraint $\rightarrow \mathcal{L}_{\text{eval}}\big(t_i, \mathbf{p}_i, \mathbf{x}_{1,2}(\cdot)\big)$
- $y_i = g\big(\mathbf{p}_i, \mathbf{b}_j\big) \rightarrow$ distance constraint $\rightarrow \mathcal{L}_{\text{dist}}\big(\mathbf{p}_i, \mathbf{b}_j, y_i\big)$

Graph of involved contractors and domains

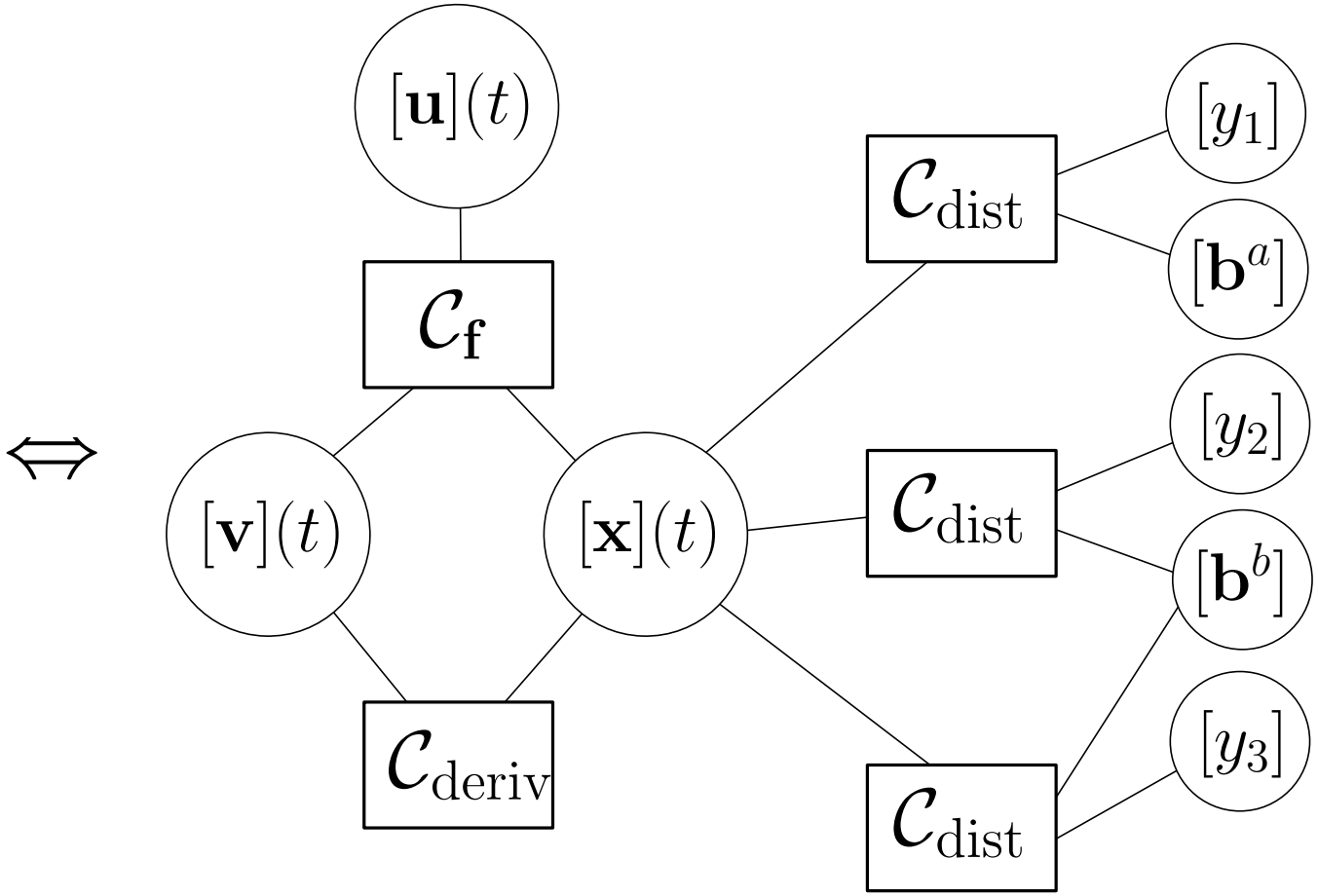
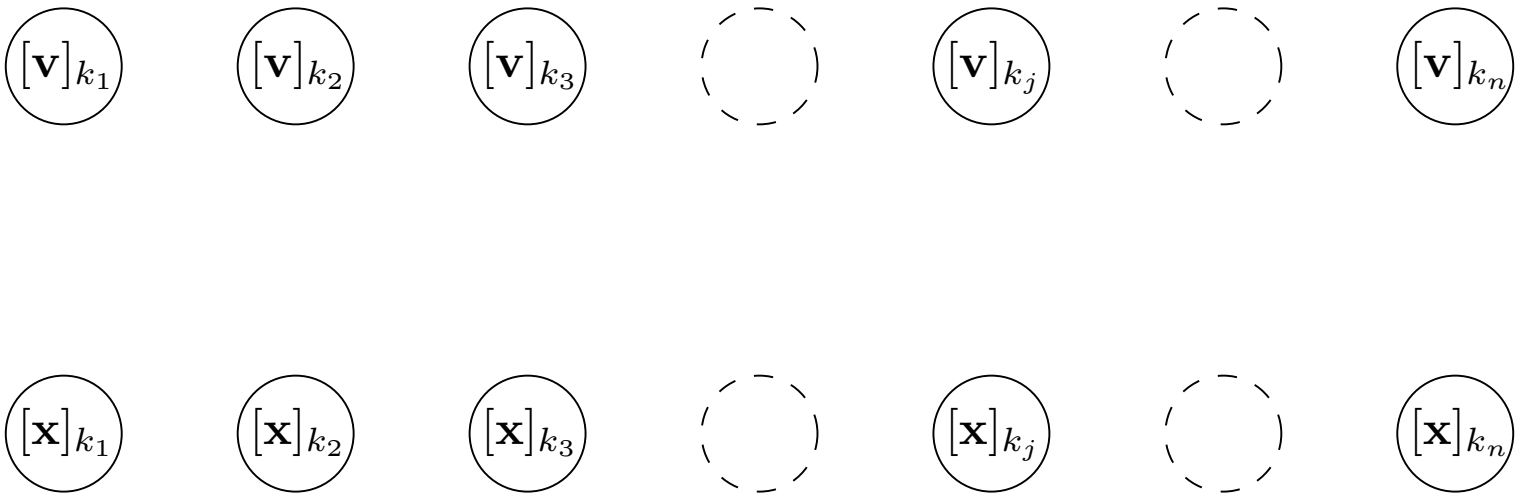
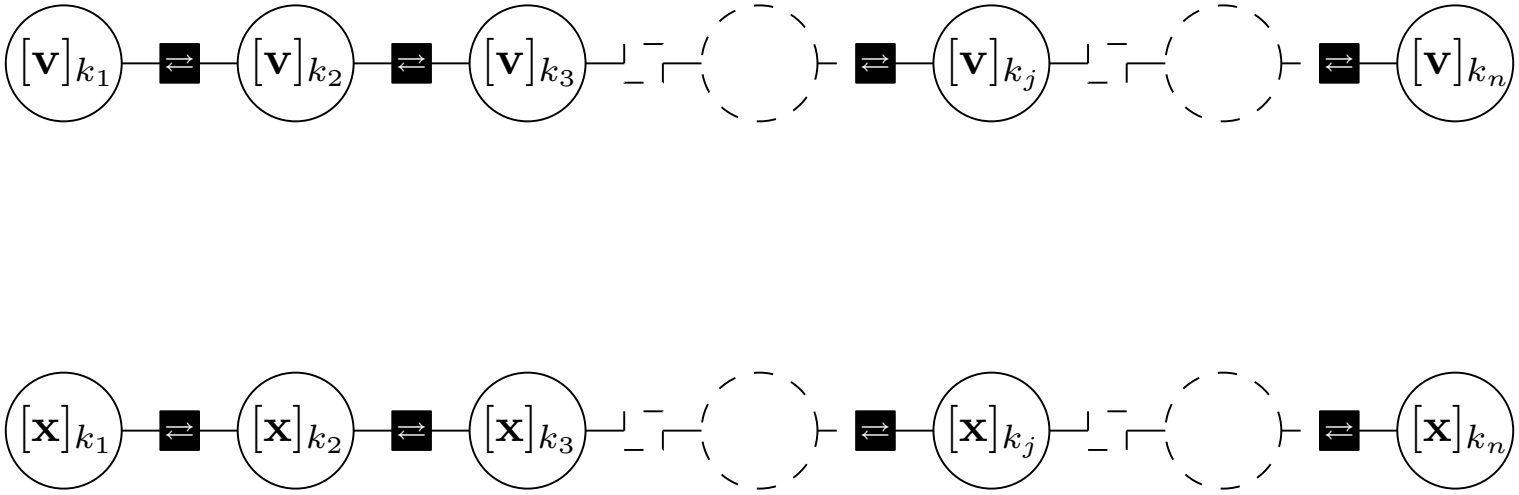


Illustration of the graph of contractors and domains corresponding to the SLAM problem: so-called **Contractor Network**.

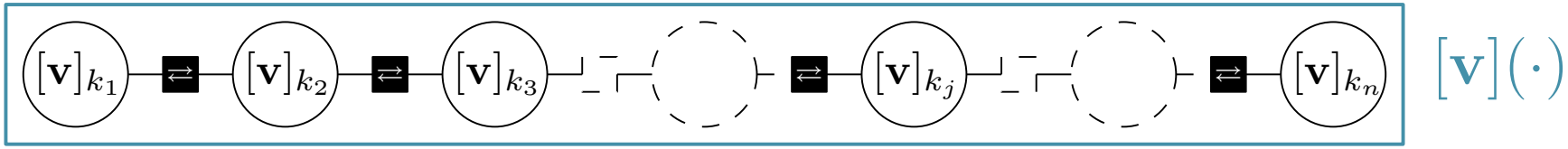
Graph of involved contractors and domains

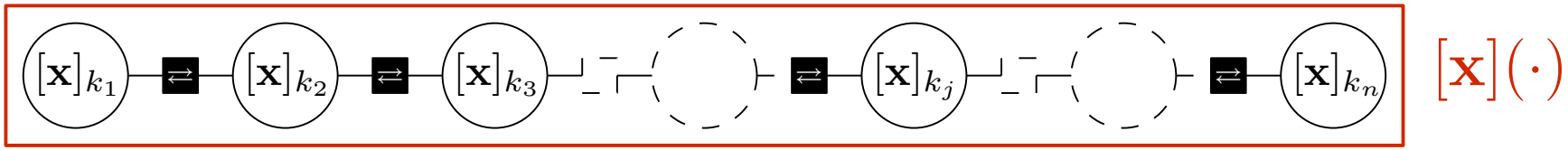


Graph of involved contractors and domains

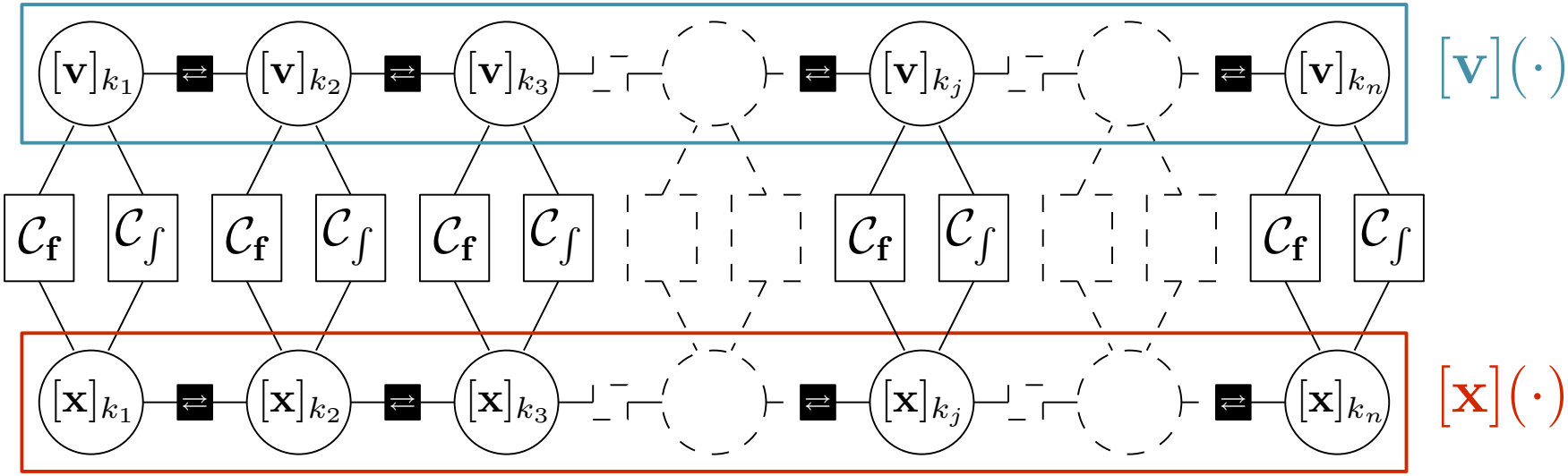


Graph of involved contractors and domains

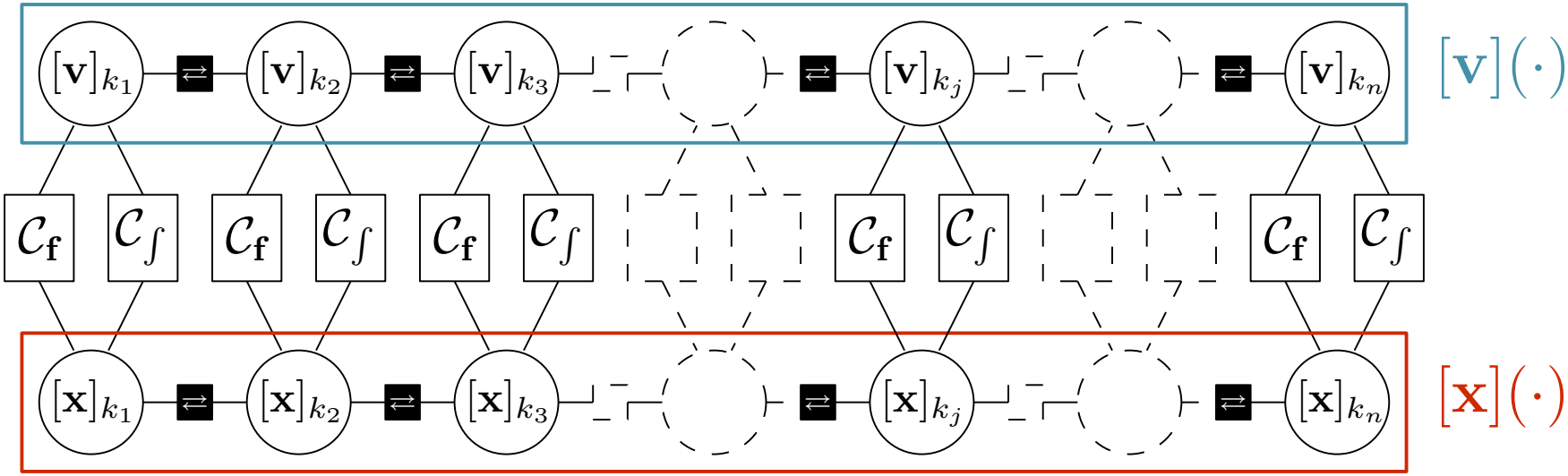




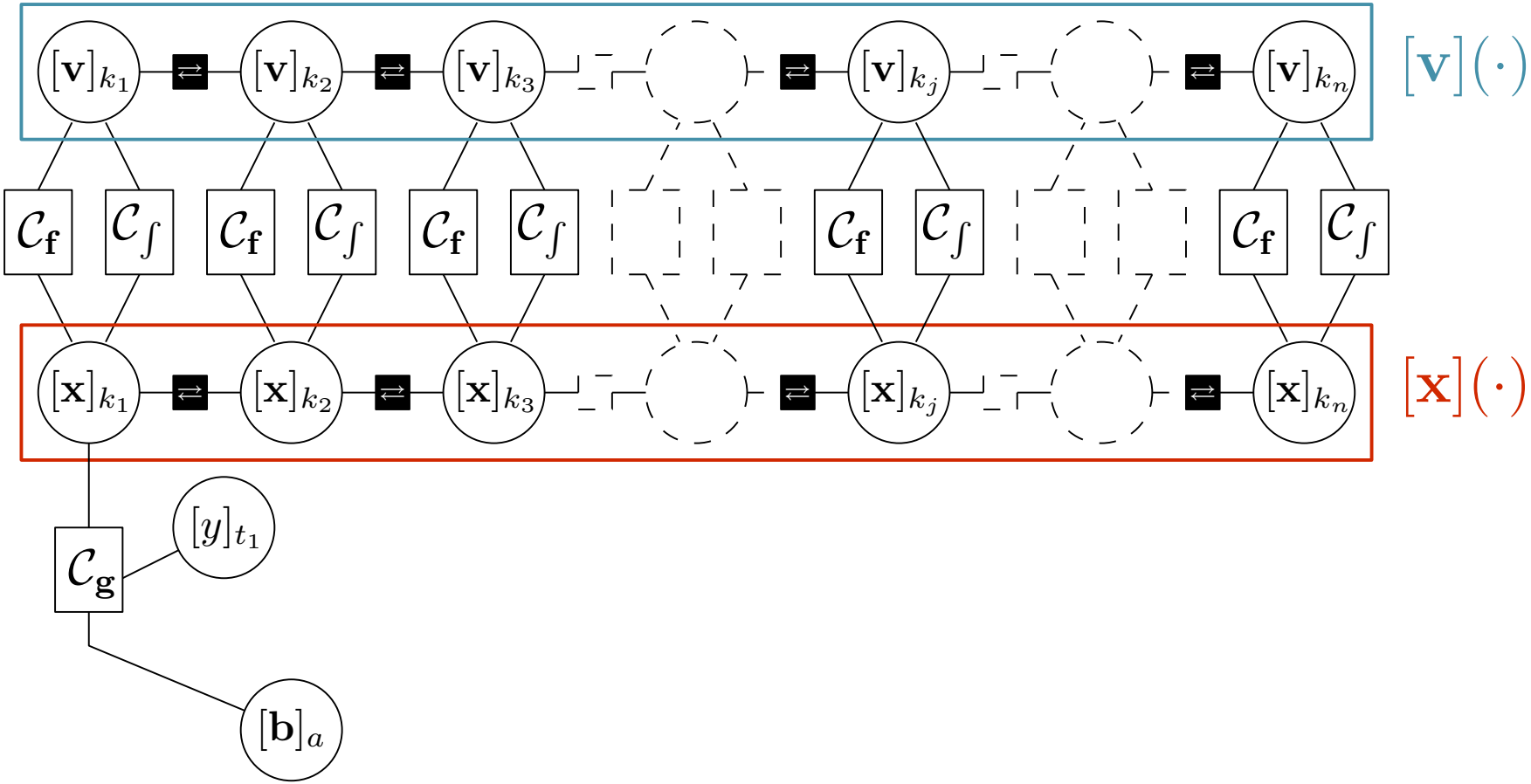
Graph of involved contractors and domains



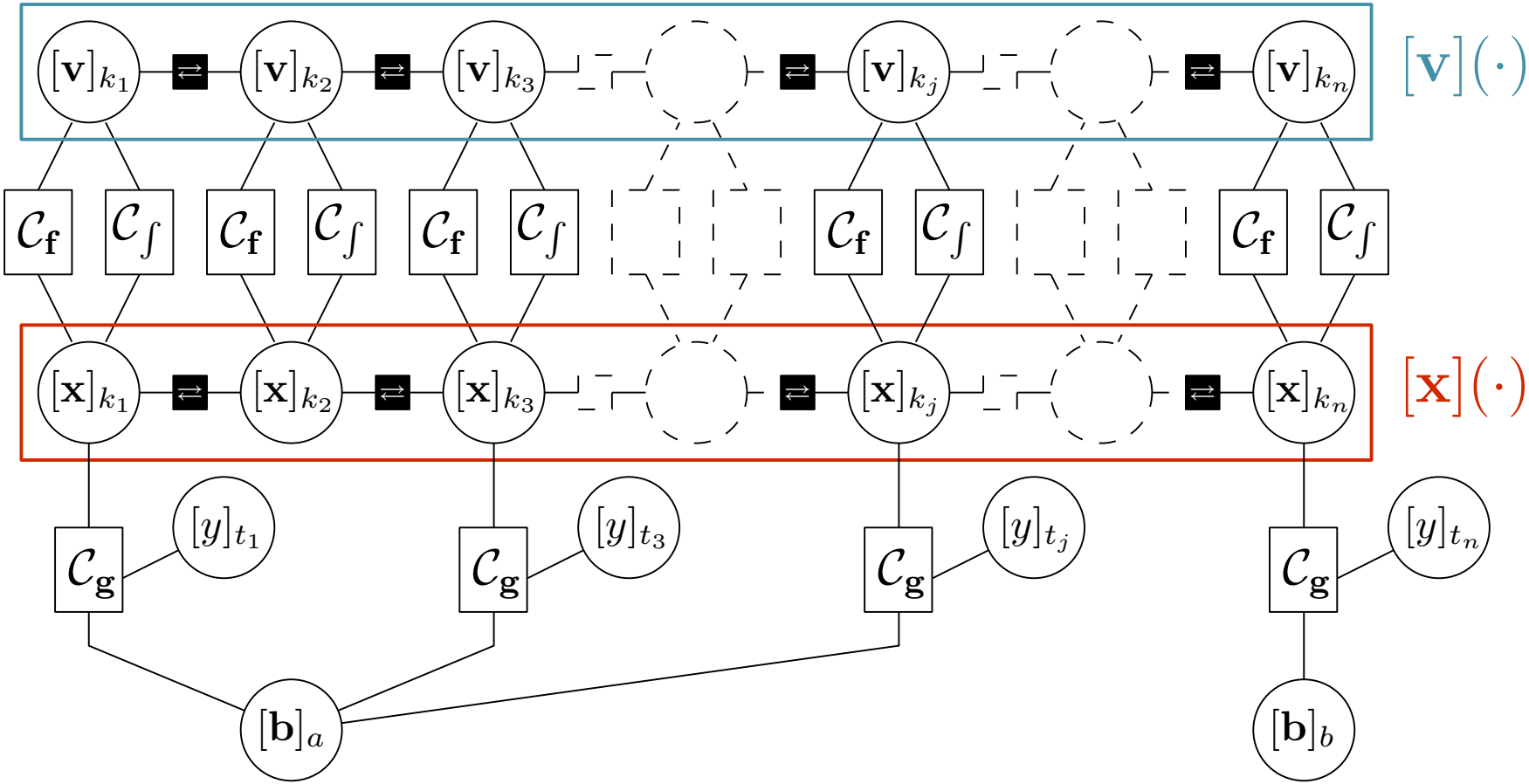
Graph of involved contractors and domains



Graph of involved contractors and domains



Graph of involved contractors and domains



Programming a SLAM-CN: methodology

1. Define domains:
   - intervals, boxes, tubes, ...
   - related to measurements or initialized as $[-\infty, \infty]$

..

Programming a SLAM-CN: methodology

1. Define domains:
   - intervals, boxes, tubes, ...
   - related to measurements or initialized as $[-\infty, \infty]$
2. Define contractors:
   - use/configure already existing contractors from the library
   - or build own contractors for specific constraints

..

Programming a SLAM-CN: methodology

1. Define domains:
   - intervals, boxes, tubes, ...
   - related to measurements or initialized as $[-\infty, \infty]$
2. Define contractors:
   - use/configure already existing contractors from the library
   - or build own contractors for specific constraints
3. Build the Contractor Network:

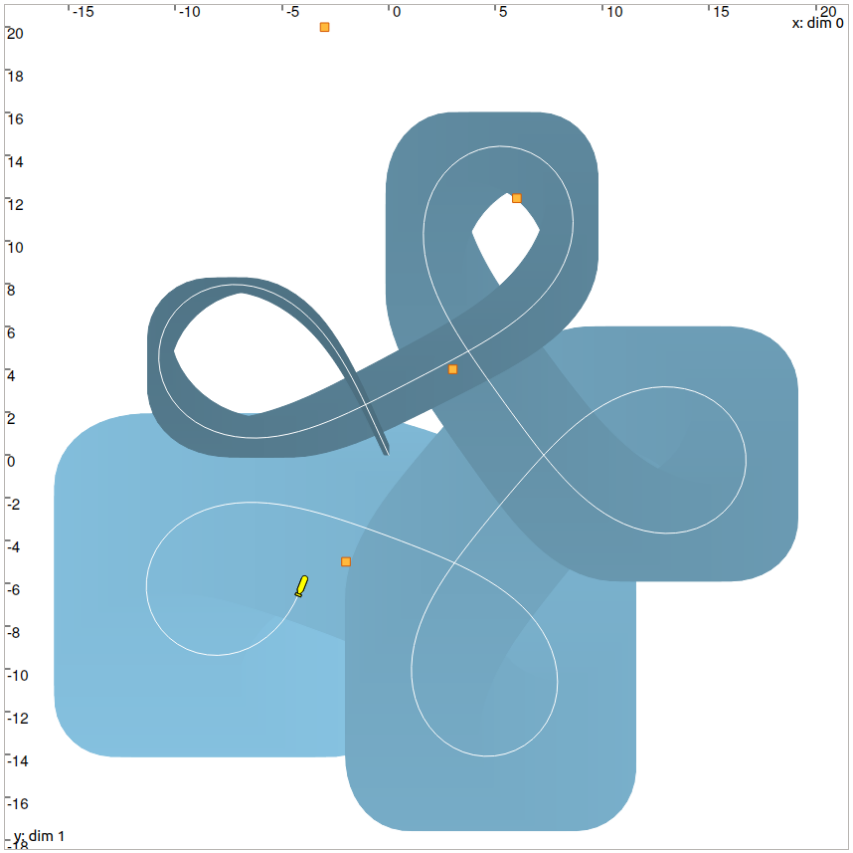
..

Programming a SLAM-CN: methodology

1. Define domains:
   - intervals, boxes, tubes, ...
   - related to measurements or initialized as $[-\infty, \infty]$
2. Define contractors:
   - use/configure already existing contractors from the library
   - or build own contractors for specific constraints
3. Build the Contractor Network:

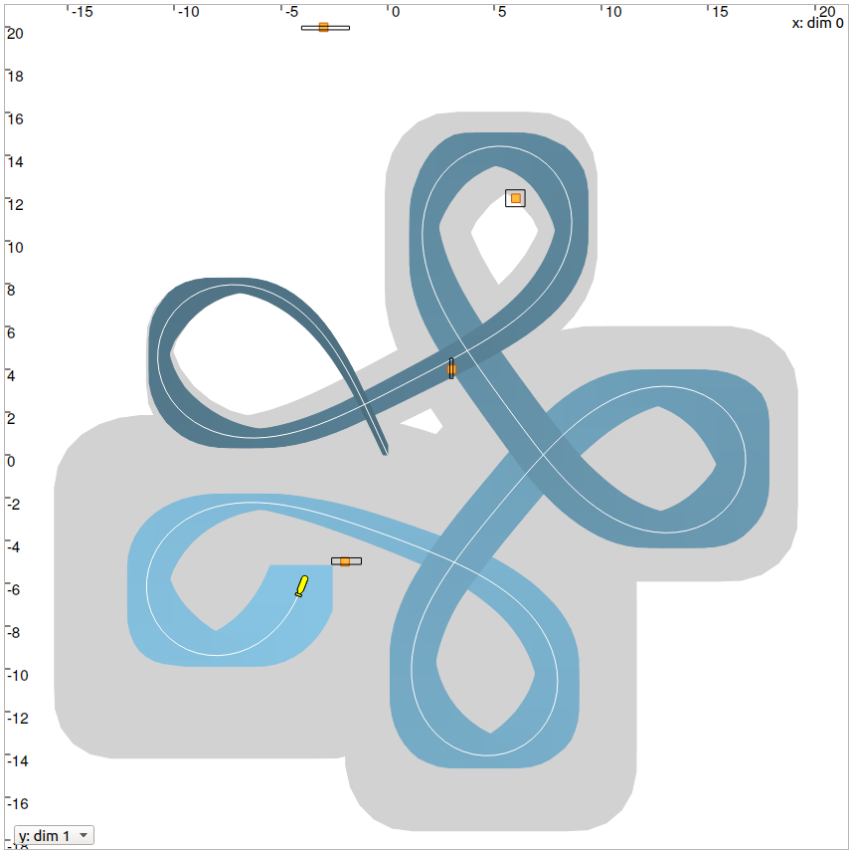
```
cn = ContractorNetwork()
cn.add(ctc_f, [x,v])
cn.add(ctc.deriv, [x,v])

for i in range(len(v_t)):
  pi = IntervalVector(4)
  cn.add(ctc.eval, [t[i],pi,x]
  cn.add(ctc.dist, [y[i],pi,b[i]])

cn.contract()
```

Programming a SLAM-CN: methodology




```
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.dist, ..
```

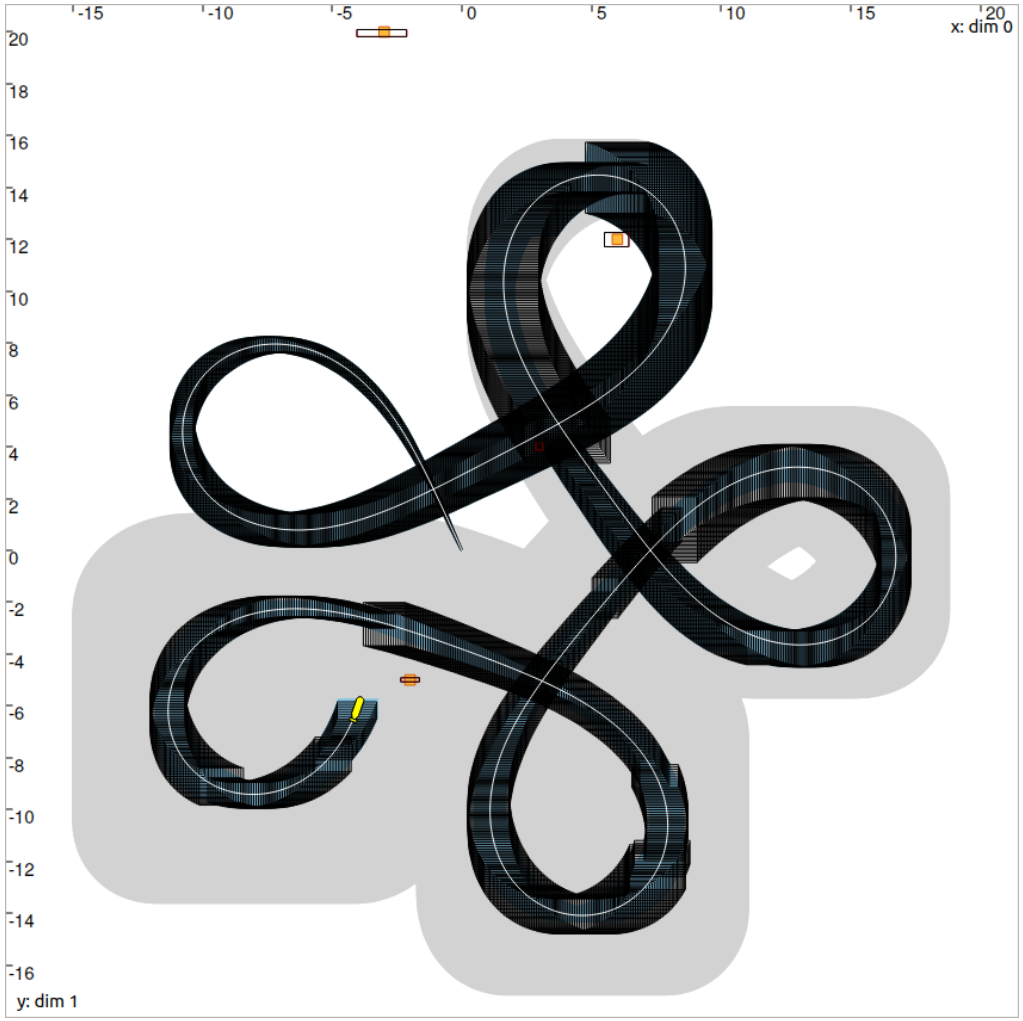
Programming a SLAM-CN: methodology



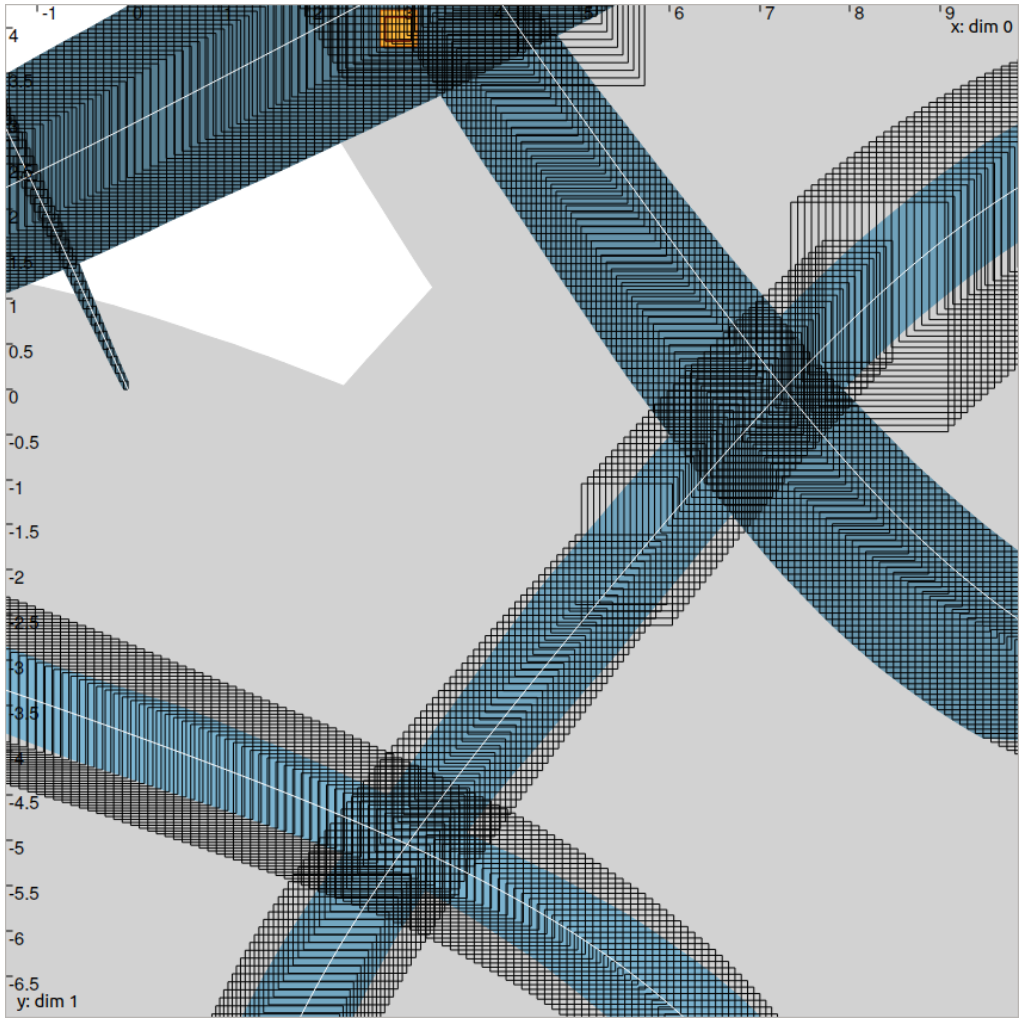

```
cn = ContractorNetwork()

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.dist, ..

cn.contract()
```

2399 contractors, 2410 dom.
Computation time: 0.25s

SLAM-CN: realtime application



SLAM-CN: realtime application



SLAM-CN: realtime application

Video