Guaranteed Computation of Robots Trajectories

Simon Rohou[1], Luc Jaulin[1], Lyudmila Mihaylova[2],
Fabrice Le Bars[1], Sandor M. Veres[2]

[1] ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, France
[2] University of Sheffield, Western Bank Sheffield S10 2TN, UK
simon.rohou@ensta-bretagne.org

Dependability of Complex Robotic Systems – June 2016

Section 1

Introduction

The University Of Sheffield.

ENSTA Bretagne

Introduction

Guaranteed integration

Problem:
We consider the problem of guaranteed integration of differential state equations defined as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t)$$

Where:

- $t \in \mathbb{R}$ is the time
- $\mathbf{x}(t)$ is the state vector
- $\mathbf{n}(t)$ is the noise vector, assumed to belong to a known box $[\mathbf{n}]$
- $\mathbf{f} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is the *evolution* function

Introduction

Guaranteed integration

Problem:
We consider the problem of guaranteed integration of differential state equations defined as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t)$$

Existing methods:

- initial box $[\mathbf{x}](0)$ known
- methods based on Euler, Runge-Kutta or Taylor integration
- validation using the Picard Theorem
- available libraries: Cosy, Vnode, DynIbex, Capd

Introduction

Guaranteed integration

Problem:
We consider the problem of guaranteed integration of differential state equations defined as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t)$$

Proposed method:

- **constraint-based** approach for guaranteed interval integration
- application to a set of **trajectories** representing feasible solutions
- trajectories handled within **tubes**

Introduction

Constraint Network based on trajectories

The problem is classically presented with a **Constraint Network**:

$$\begin{cases} \textbf{Variables: } \mathbf{x} \end{cases}$$

[Mackworth1977], [Chabert2009]

Introduction

Constraint Network based on trajectories

The problem is classically presented with a **Constraint Network**:

[Mackworth1977], [Chabert2009]

$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad - \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t) \end{cases}$$

Introduction

Constraint Network based on trajectories

The problem is classically presented with a **Constraint Network**:

[Mackworth1977], [Chabert2009]

$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad - \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t) \\ \quad - \quad \dots \end{cases}$$

Introduction

Constraint Network based on trajectories

The problem is classically presented with a **Constraint Network**:

[Mackworth1977], [Chabert2009]

$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad - \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t) \\ \quad - \quad \dots \\ \textbf{Domains: } [\mathbf{x}] \end{cases}$$

Introduction

Constraint Network based on trajectories

The problem is classically presented with a **Constraint Network**:

[Mackworth1977], [Chabert2009]

$$\begin{cases} \textbf{Variables: } \mathbf{x} \\ \textbf{Constraints:} \\ \quad - \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t) \\ \quad - \quad \dots \\ \textbf{Domains: } [\mathbf{x}] \end{cases}$$

Our approach consists in using:

- **trajectories** as variables: $\mathbf{x}(\cdot)$
- **tubes** as domains: $\mathbf{x}(\cdot) \in [\mathbf{x}](\cdot)$

Section 2

Tube Programming

The University Of Sheffield.

ENSTA Bretagne

Tube Programming

Tubes: definition

Tube $[x](\cdot)$**:** interval of functions $[x^-, x^+]$ such that $\forall t \in \mathbb{R},\ x^-(t) \leq x^+(t)$

[LeBars2012]

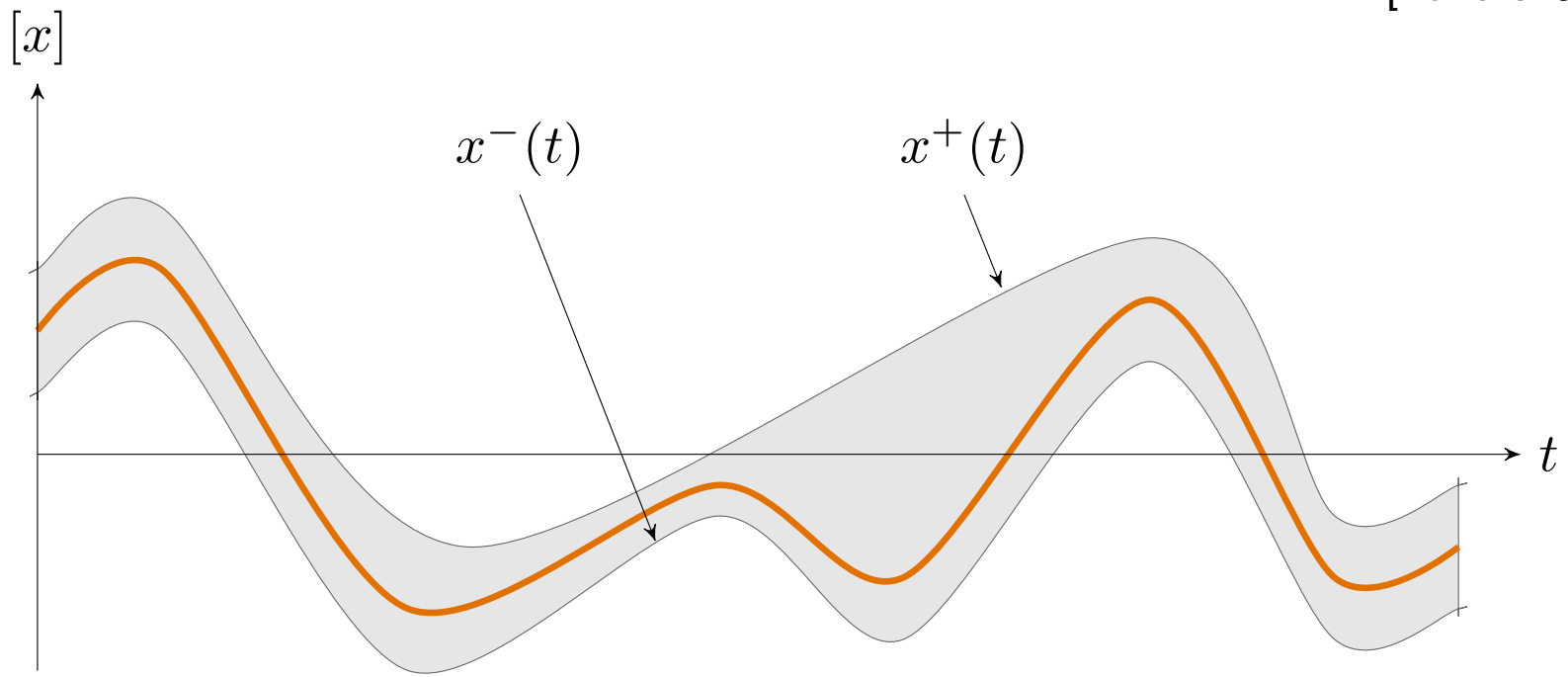


Figure: tube $[x](\cdot)$ enclosing an uncertain trajectory $x^*(\cdot)$

Tube Programming

Tubes: implementation

Tube $[x](\cdot)$**:** interval of functions $[x^-, x^+]$ such that $\forall t \in \mathbb{R},\ x^-(t) \leq x^+(t)$

[LeBars2012]

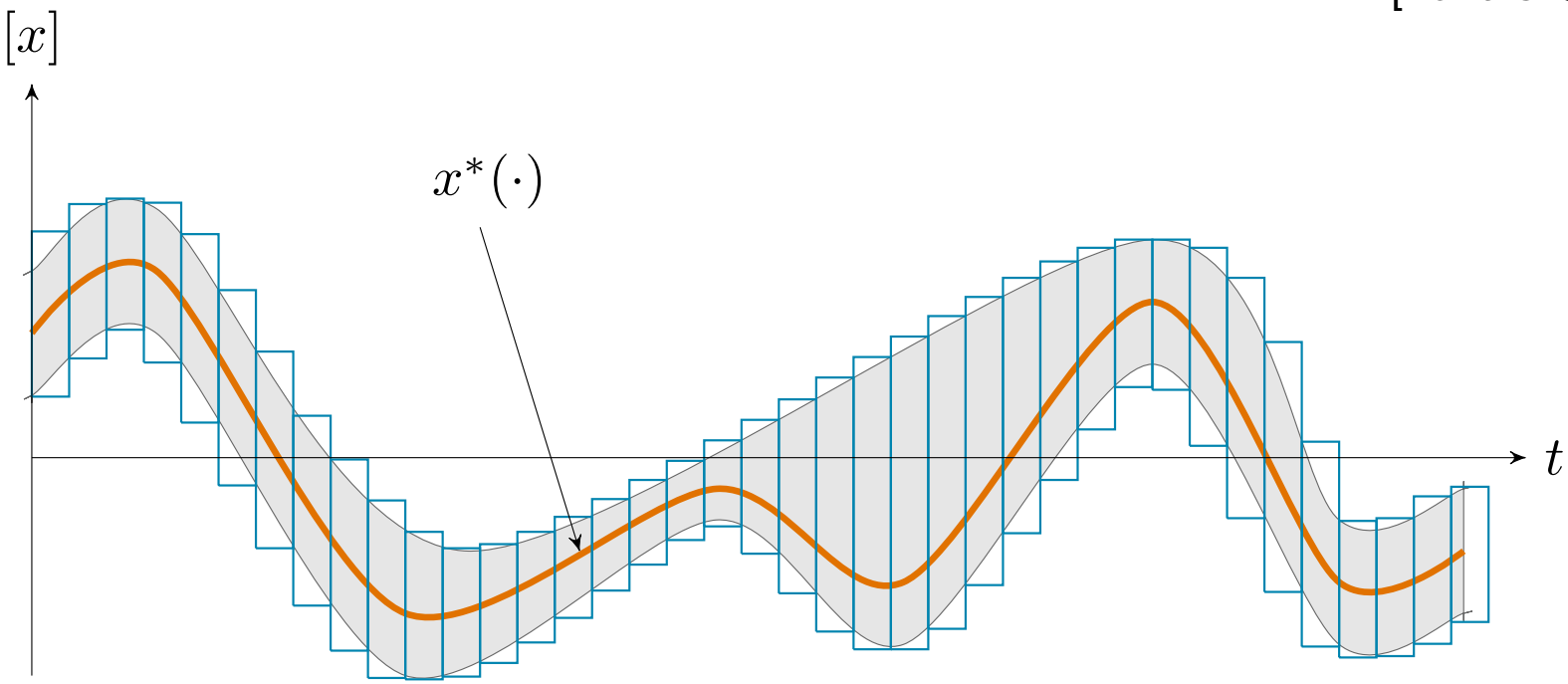


Figure: tube implementation with a set of boxes

Tube Programming

Tubes: integral

Definition: the integral of a tube $[x](\cdot) = [x^-, x^+]$ is an interval:

$$\int_a^b [x](\tau)d\tau = \left\{ \int_a^b x(\tau)d\tau \mid x \in [x] \right\} = \left[\int_a^b x^-(\tau)d\tau, \int_a^b x^+(\tau)d\tau \right]$$

[Aubry2013]

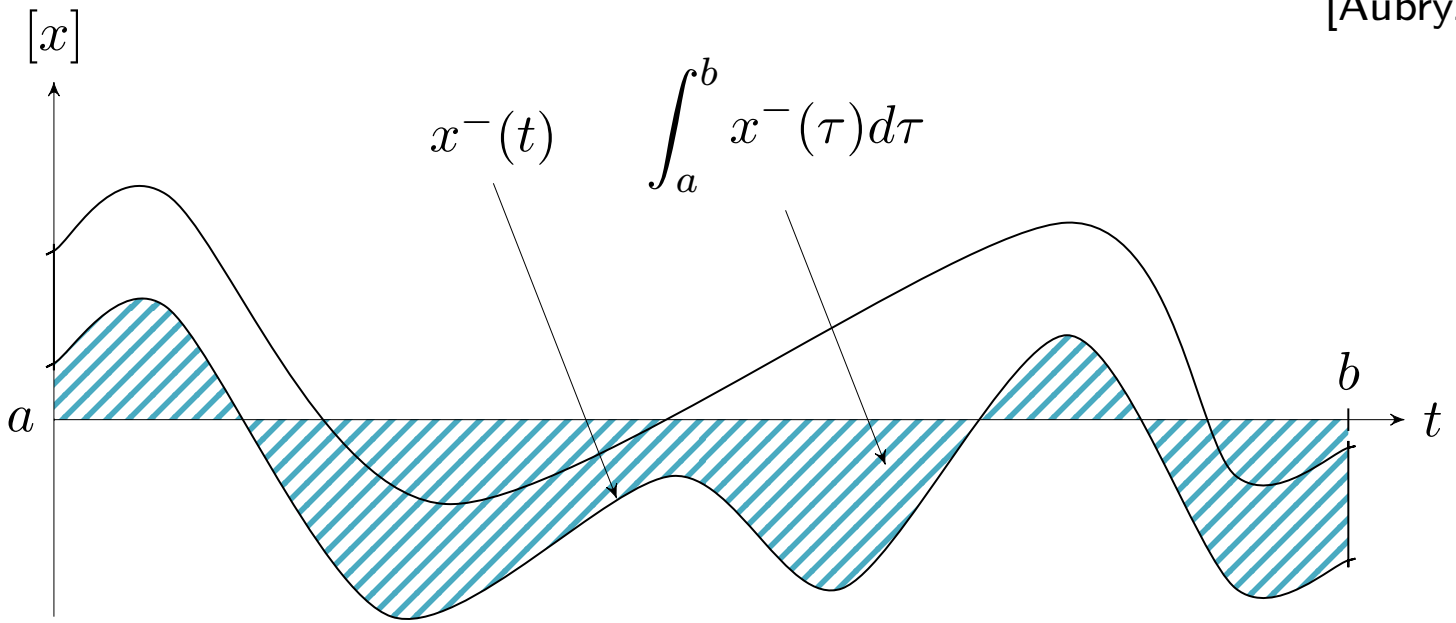


Figure: blue area: lower bound of tube's integral

Tube Programming

Tubes: integral

Definition: the integral of a tube $[x](\cdot) = [x^-, x^+]$ is an interval:

$$\int_a^b [x](\tau)d\tau = \left\{ \int_a^b x(\tau)d\tau \mid x \in [x] \right\} = \left[\int_a^b x^-(\tau)d\tau, \int_a^b x^+(\tau)d\tau \right]$$

[Aubry2013]

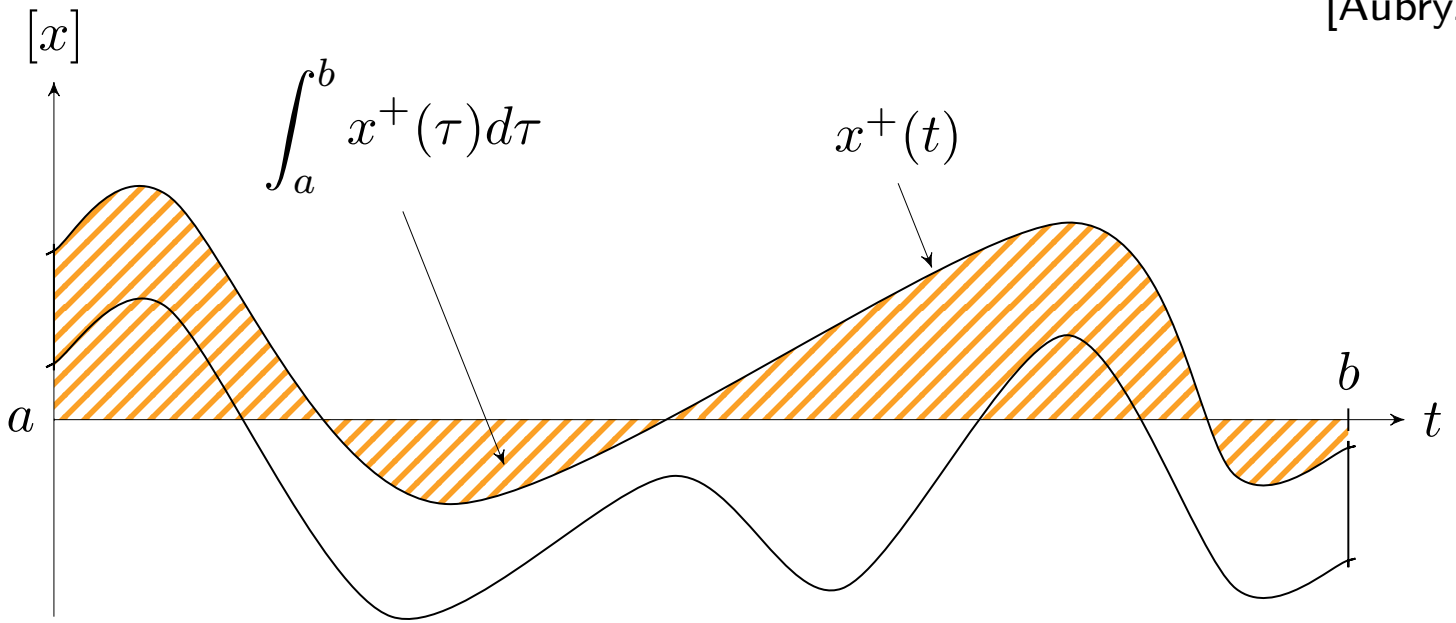


Figure: orange area: upper bound of tube's integral

Tube Programming

Tubes: arithmetic and contractors

Example:
Tube arithmetic makes it possible to compute the following tubes:

$$\begin{aligned} [a](\cdot) &= [x](\cdot) + [y](\cdot) \\ [b](\cdot) &= \sin([x](\cdot)) \\ [c](\cdot) &= \int_0 [x](\tau) d\tau \end{aligned}$$

Contractors for tubes:
To each primitive constraint on trajectories, tubes are contracted without removing any feasible solution.

Tube Programming

Tubes: minimal and non-minimal contractors

Example:

The minimal contractor associated to the constraint $[a](\cdot) = [x](\cdot) + [y](\cdot)$:

$$\begin{pmatrix} [a](\cdot) \\ [x](\cdot) \\ [y](\cdot) \end{pmatrix} \mapsto \begin{pmatrix} [a](\cdot) \cap ([x](\cdot) + [y](\cdot)) \\ [x](\cdot) \cap ([a](\cdot) - [y](\cdot)) \\ [y](\cdot) \cap ([a](\cdot) - [x](\cdot)) \end{pmatrix}$$

Tube Programming

Tubes: minimal and non-minimal contractors

Example:

The minimal contractor associated to the constraint $[a](\cdot) = [x](\cdot) + [y](\cdot)$:

$$\begin{pmatrix} [a](\cdot) \\ [x](\cdot) \\ [y](\cdot) \end{pmatrix} \mapsto \begin{pmatrix} [a](\cdot) \cap ([x](\cdot) + [y](\cdot)) \\ [x](\cdot) \cap ([a](\cdot) - [y](\cdot)) \\ [y](\cdot) \cap ([a](\cdot) - [x](\cdot)) \end{pmatrix}$$

Example:

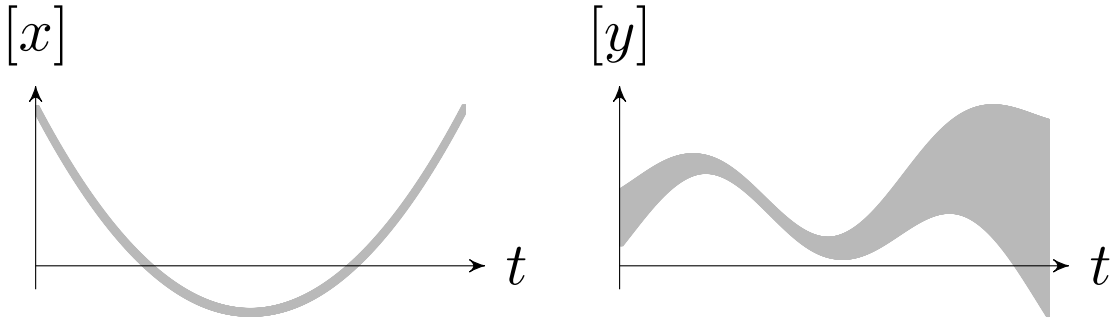
The non-minimal contractor associated to the constraint $[c](\cdot) = \int_0 [x](\tau) d\tau$:

$$\begin{pmatrix} [x](\cdot) \\ [c](\cdot) \end{pmatrix} \mapsto \begin{pmatrix} [x](\cdot) \\ [c](\cdot) \cap \int_0 [x](\tau)\, d\tau \end{pmatrix}$$

The University Of Sheffield.

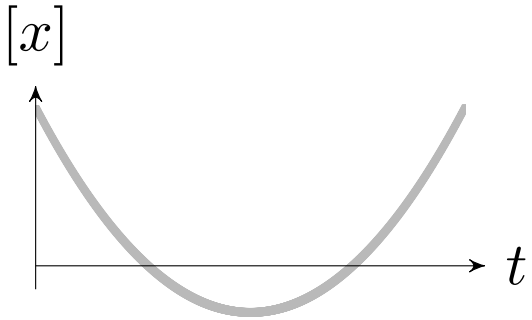
Tube Programming

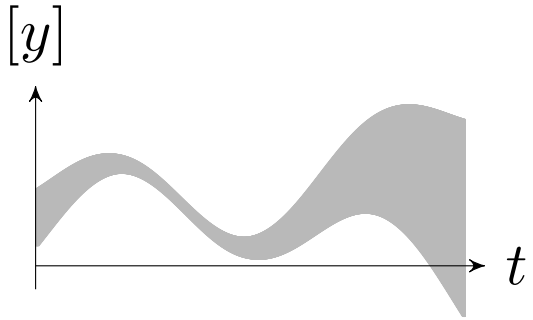
Tubes programming: example

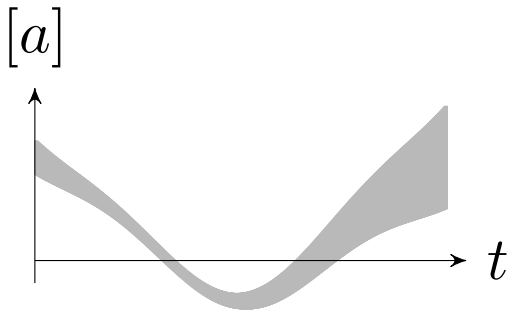


Tube Programming

Tubes programming: example



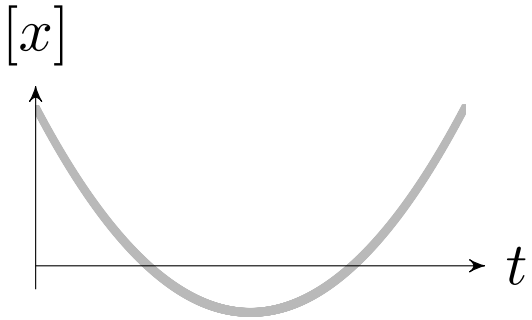


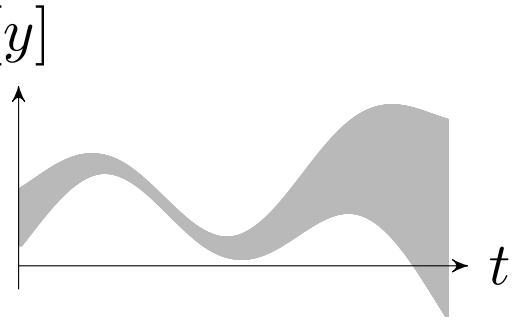


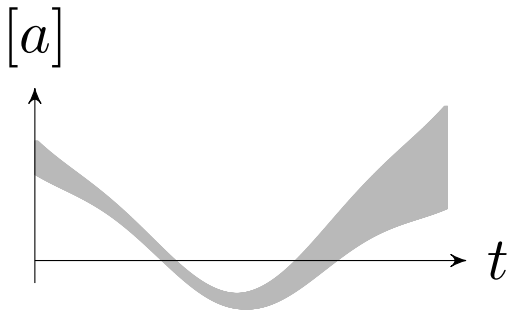
$$[a](\cdot) = [x](\cdot) + [y](\cdot)$$

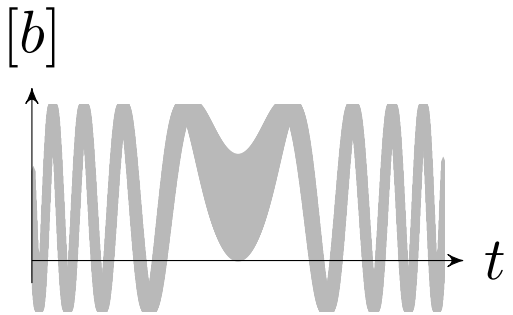
Tube Programming

Tubes programming: example





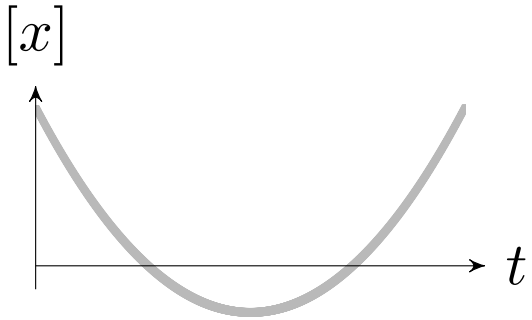


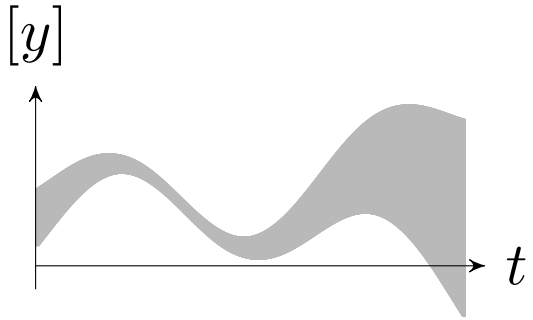


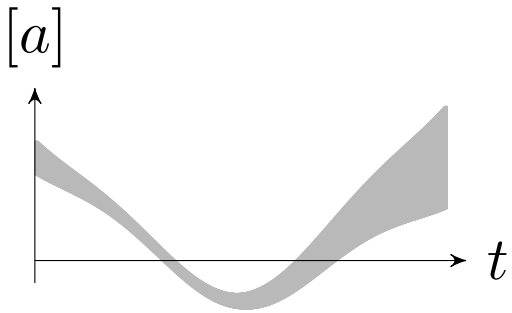
$$[a](\cdot) = [x](\cdot) + [y](\cdot) \qquad [b](\cdot) = \sin([x](\cdot))$$

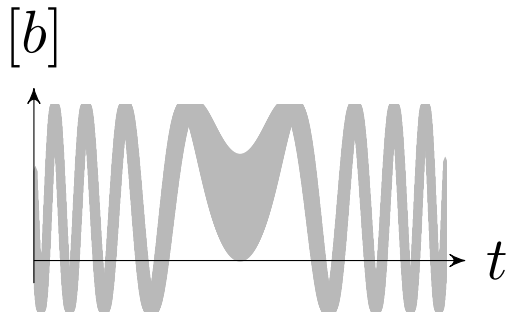
Tube Programming

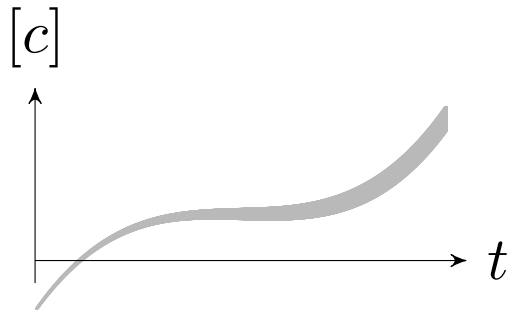
Tubes programming: example











$$[a](\cdot) = [x](\cdot) + [y](\cdot) \qquad [b](\cdot) = \sin([x](\cdot)) \qquad [c](\cdot) = \int_0 [x](\tau)d\tau$$

Section 3

Interval Integration

Interval Integration

Initial value problem

An interval integration corresponds to a **specific constraint network**:

- with only one constraint given by:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{n}(t)$$

- an initial condition, assumed known

To solve this integration, a **propagation method** should apply.

Interval Integration

Example: propagation method

Let us consider the following **initial value problem**:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

This problem is unstable.

Interval Integration

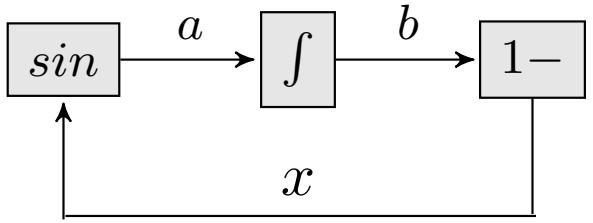
Example: propagation method

Let us consider the following **initial value problem**:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

This problem is unstable.
Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$



All trajectories $x(\cdot)$, $a(\cdot)$, $b(\cdot)$ belong to tubes.
Presence of one loop $\longrightarrow$ iterative resolution until a **fix-point**.

Interval Integration

Propagation method

Problem:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$

Domains:

$$\begin{cases} [x](\cdot) &= [-1,1] \\ t &\in [0,10] \end{cases}$$

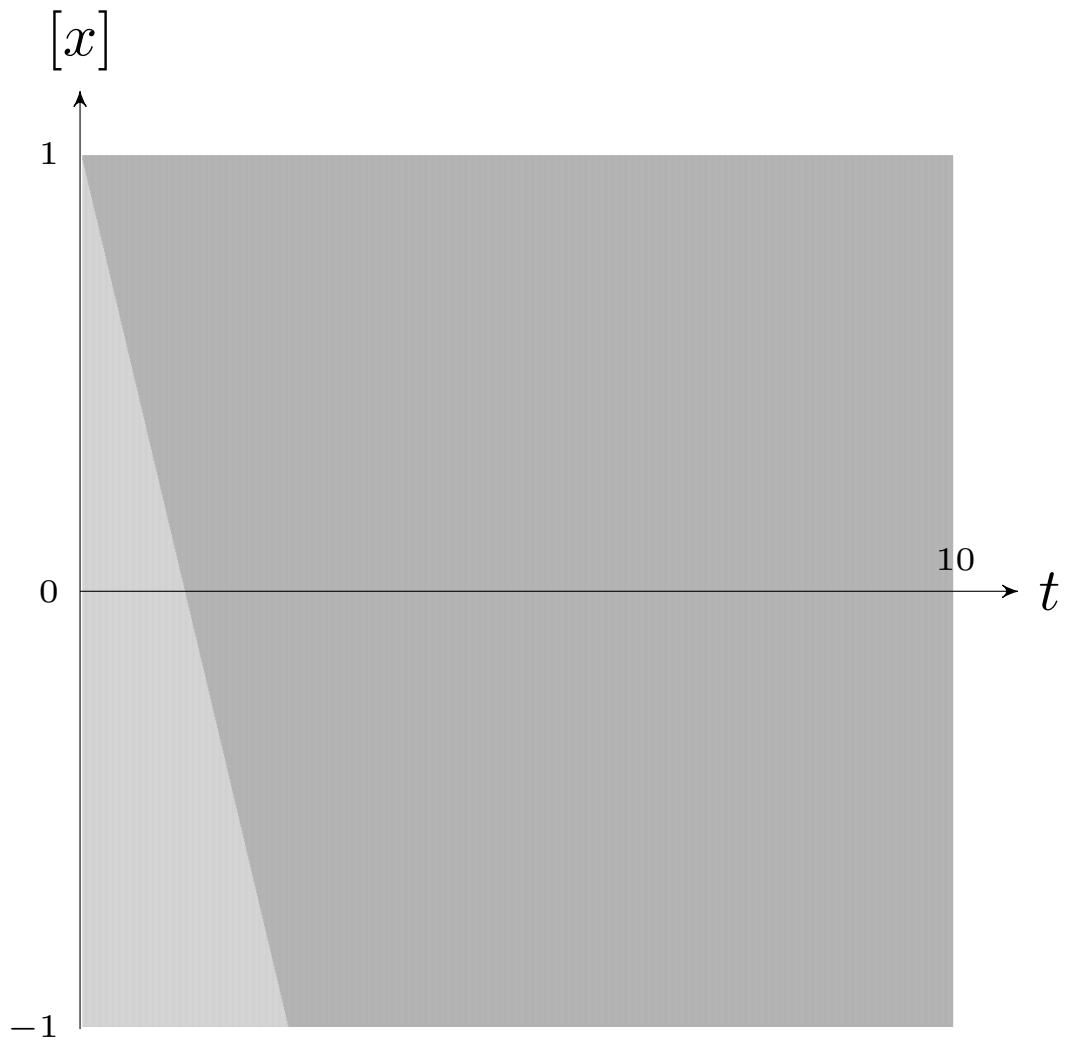


Figure: Successive contractions of tube $[x](\cdot)$

Step 1

Interval Integration

Propagation method

Problem:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$

Domains:

$$\begin{cases} [x](\cdot) &= [-1,1] \\ t &\in [0,10] \end{cases}$$

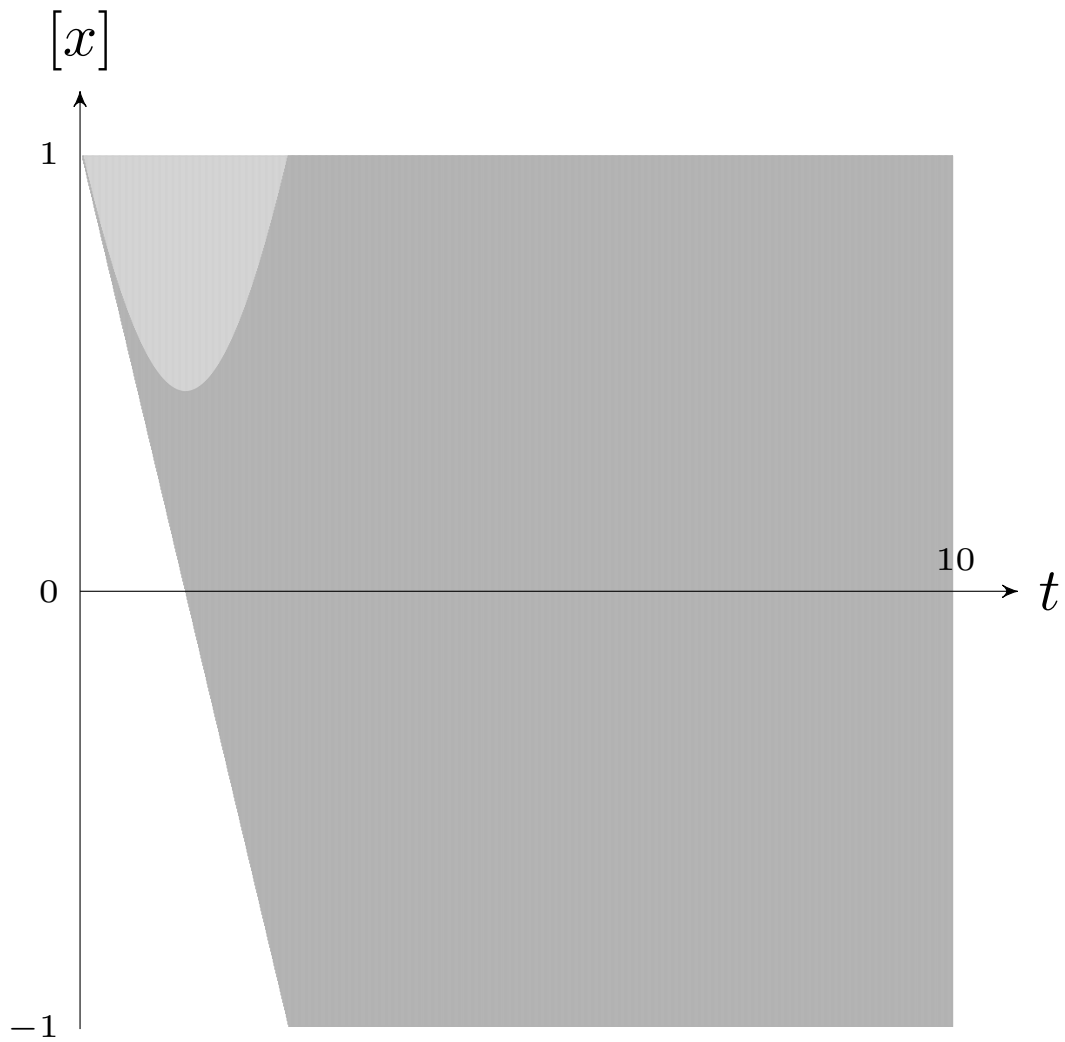


Figure: Successive contractions of tube $[x](\cdot)$

Step 2

Interval Integration

Propagation method

Problem:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$

Domains:

$$\begin{cases} [x](\cdot) &= [-1,1] \\ t &\in [0,10] \end{cases}$$

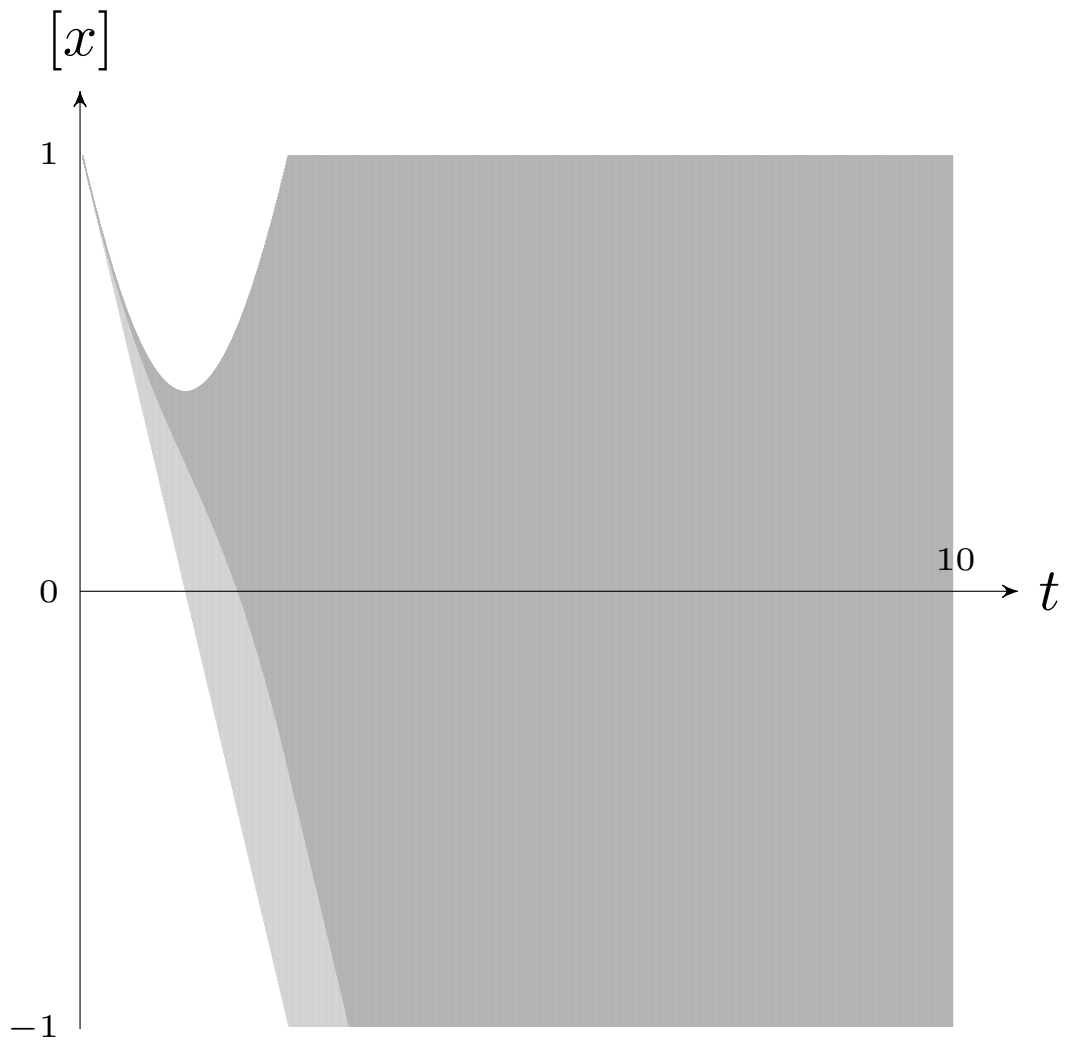


Figure: Successive contractions of tube $[x](\cdot)$

Step 3

Interval Integration

Propagation method

Problem:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$

Domains:

$$\begin{cases} [x](\cdot) &= [-1,1] \\ t &\in [0,10] \end{cases}$$

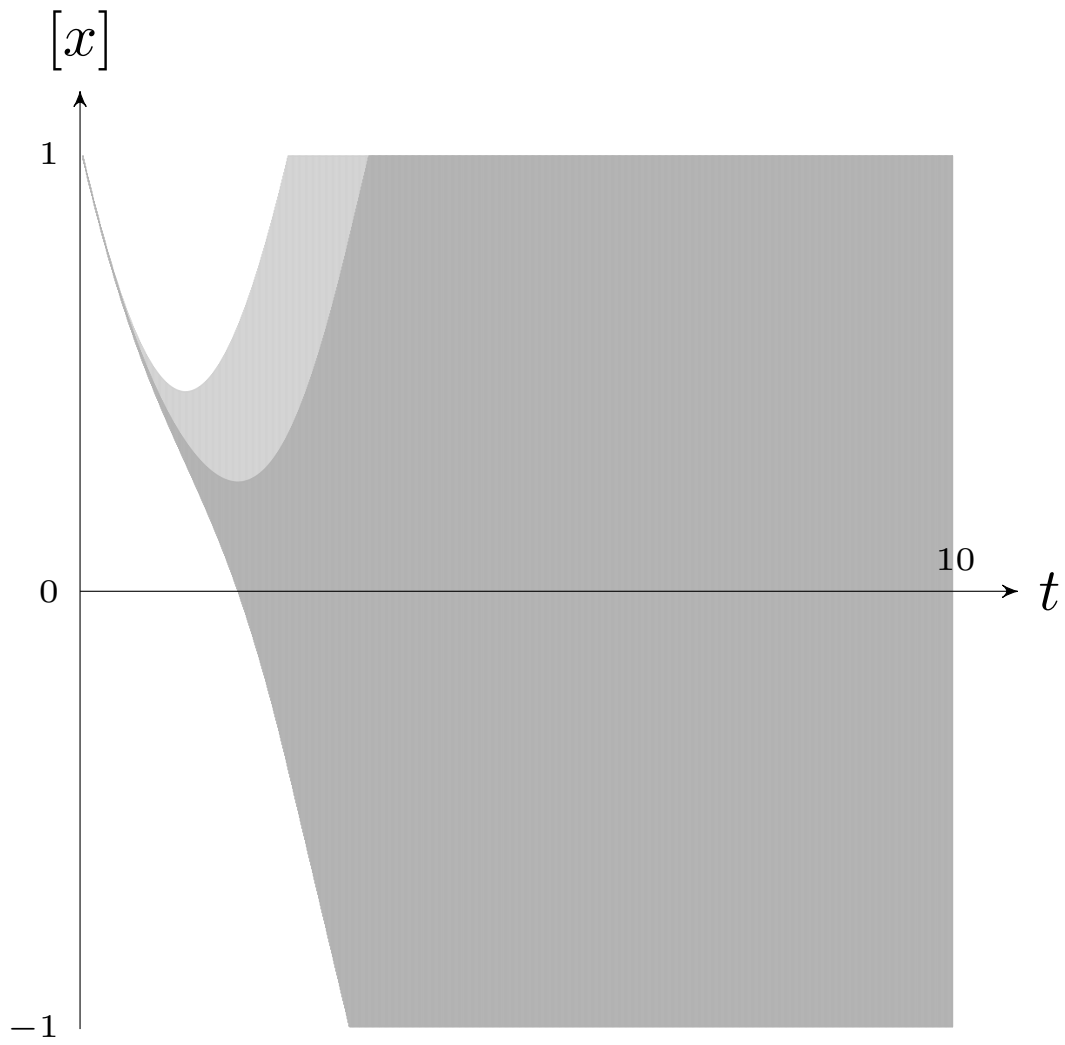


Figure: Successive contractions of tube $[x](\cdot)$

Step 4

Interval Integration

Propagation method

Problem:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$

Domains:

$$\begin{cases} [x](\cdot) &= [-1,1] \\ t &\in [0,10] \end{cases}$$

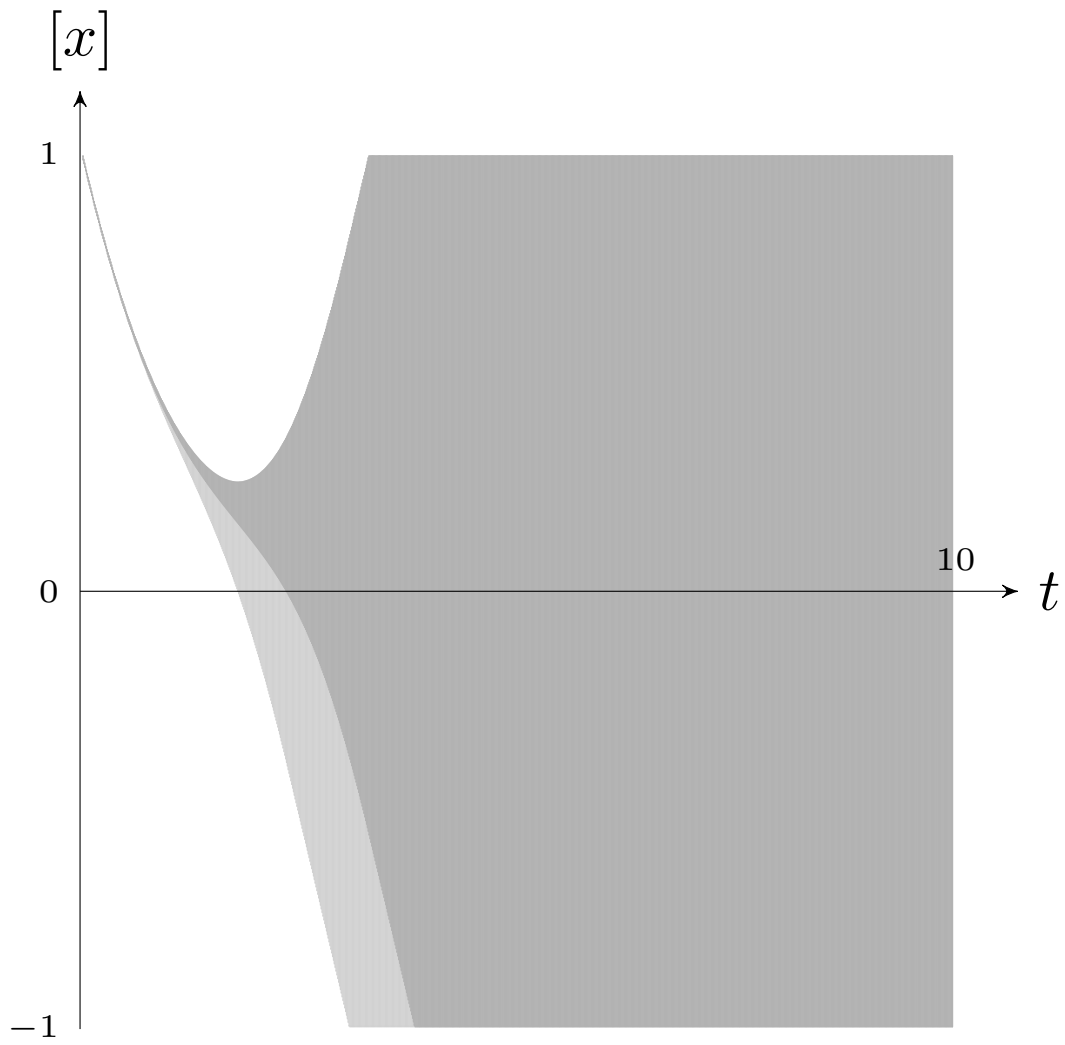


Figure: Successive contractions of tube $[x](\cdot)$

Step 5

Interval Integration

Propagation method

Problem:

$$\begin{cases} \dot{x} &= -\sin(x) \\ x_0 &= 1 \end{cases}$$

Decomposition:

$$\begin{cases} a(\cdot) &= \sin(x) \\ b(\cdot) &= \int_0 a(\tau)d\tau \\ x(\cdot) &= 1 - b(\cdot) \end{cases}$$

Domains:

$$\begin{cases} [x](\cdot) &= [-1,1] \\ t &\in [0,10] \end{cases}$$

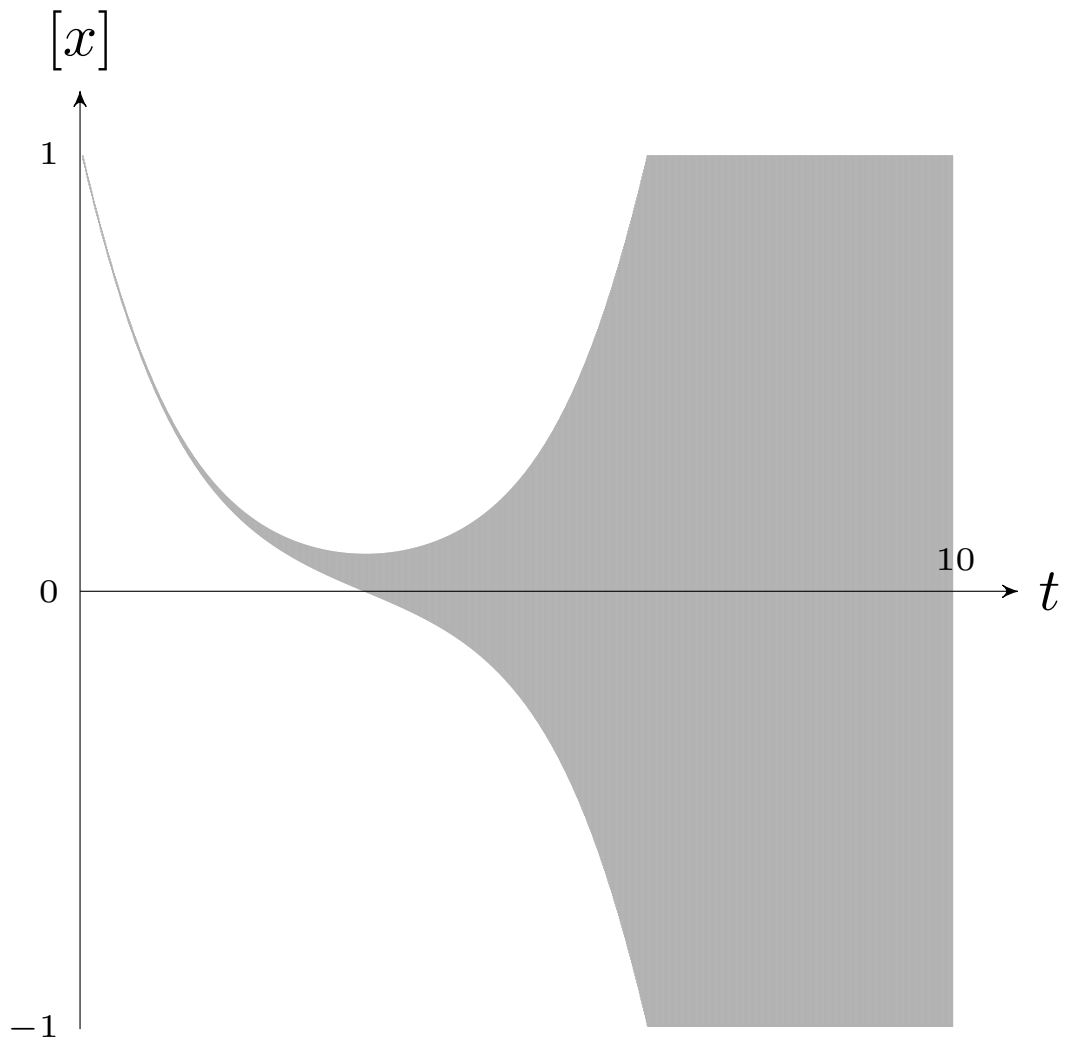


Figure: Successive contractions of tube $[x](\cdot)$

Step 41: fixpoint

Interval Integration

Example: comparing with CAPD

Comparing with the CAPD library: [CAPD]

- the obtained **envelop is poor** compared to the results produced with the CAPD library (not detailed here)
- CAPD is **typically devoted to** this type of problem where the information given on the initial state has to propagate forward
- to make the propagation working, we had to **assume an initial interval** for the solution which is not necessary with CAPD

Section 4

Simulation of a Mobile Robot

The University Of Sheffield.

ENSTA Bretagne

Simulation of a Mobile Robot

Dubin's car

For mobile robots, the differential equation often has a structure of a **causal kinematic chain** $\longrightarrow$ no propagation method needed.

Considering a Dubin's car $\mathcal{R}$ which state is $\mathbf{x} = (x, y, \theta)^\intercal$.

$$\mathcal{R} \begin{cases} \dot{x}(t) &= 10 \cdot \cos(\theta) \\ \dot{y}(t) &= 10 \cdot \sin(\theta) \\ \dot{\theta}(t) &= u(t) \end{cases}$$

Simulation of a Mobile Robot

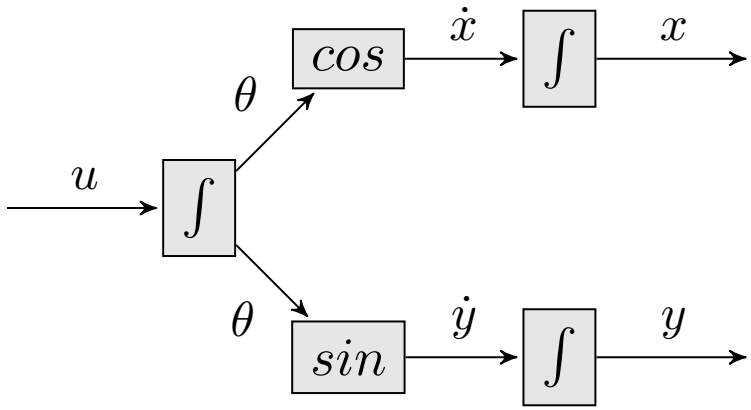
Dubin's car

For mobile robots, the differential equation often has a structure of a **causal kinematic chain** $\longrightarrow$ no propagation method needed.

Considering a Dubin's car $\mathcal{R}$ which state is $\mathbf{x} = (x, y, \theta)^\intercal$.

$$\mathcal{R} \begin{cases} \dot{x}(t) &= 10 \cdot \cos(\theta) \\ \dot{y}(t) &= 10 \cdot \sin(\theta) \\ \dot{\theta}(t) &= u(t) \end{cases}$$

We obtain the following causal chain:



Simulation of a Mobile Robot

Dubin's car in forward

In order to compare with the CAPD library, let us specify:

- car's speed: $v = 10m.s^{-1}$
- $\mathbf{x}_0 \in [-1,1] \times [-1,1] \times [\frac{-6}{5\pi} - 0.002, \frac{-6}{5\pi} + 0.002]$
- $u(t) \in -\cos\left(\frac{t+33}{5}\right) + [-0.02, 0.02]$

Simulation of a Mobile Robot

Dubin's car in forward

In order to compare with the CAPD library, let us specify:

- car's speed: $v = 10m.s^{-1}$
- $\mathbf{x}_0 \in [-1,1] \times [-1,1] \times [\frac{-6}{5\pi} - 0.002, \frac{-6}{5\pi} + 0.002]$
- $u(t) \in -\cos\left(\frac{t+33}{5}\right) + [-0.02, 0.02]$

Initial tube $[u](\cdot)$ is obtained with the analytical expression.
Other tubes $[\theta](\cdot)$, $[\dot{x}](\cdot)$, $[x](\cdot)$, ..., are initialized to $[-\infty, +\infty]$.

Simulation of a Mobile Robot

Dubin's car in forward

In order to compare with the CAPD library, let us specify:

- car's speed: $v = 10m.s^{-1}$
- $\mathbf{x}_0 \in [-1,1] \times [-1,1] \times [\frac{-6}{5\pi} - 0.002, \frac{-6}{5\pi} + 0.002]$
- $u(t) \in -\cos\left(\frac{t+33}{5}\right) + [-0.02, 0.02]$

Initial tube $[u](\cdot)$ is obtained with the analytical expression.
Other tubes $[\theta](\cdot)$, $[\dot{x}](\cdot)$, $[x](\cdot)$, ..., are initialized to $[-\infty, +\infty]$.

N.B.: with tubes, analytical expression or **real data** can be considered, which is useful for robotic purposes.

Simulation of a Mobile Robot

Dubin's car in forward

The following figure shows that our method is more accurate than CAPD on this example:

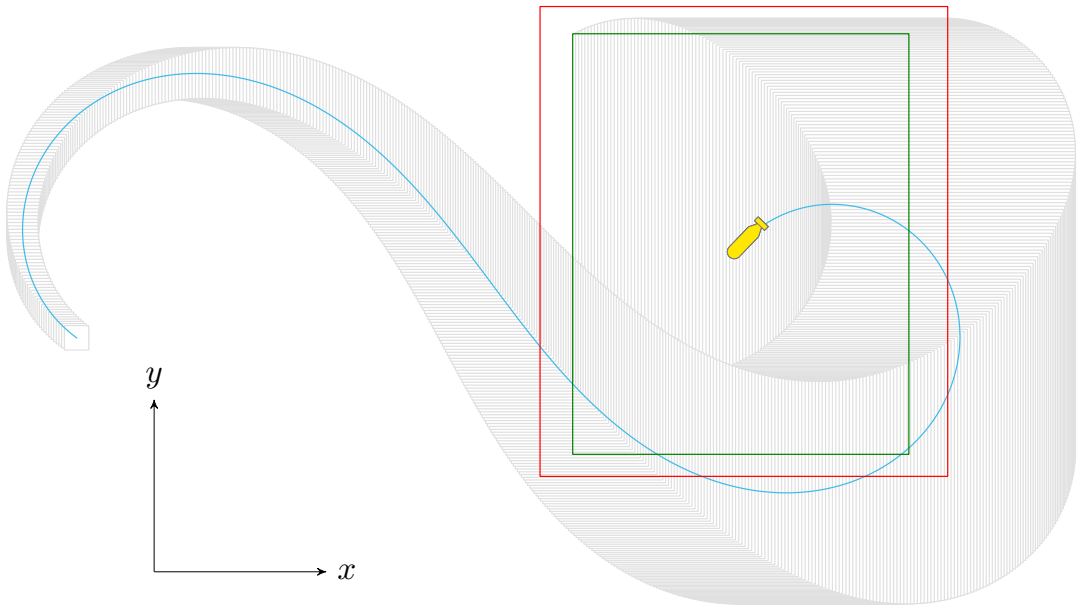


Figure: Interval simulation of the robot following Dubin's car equations. Blue line: true poses of the robot. Gray boxes: tubes $[x](\cdot) \times [y](\cdot)$ projected on the world frame. Green box: final box obtained for $t = 14$. Red box: result computed with CAPD.

Simulation of a Mobile Robot

Dubin's car in forward/backward

Same simulation considering the initial and final states known:

$$[\mathbf{x}](14) = [53.9, 55.9] \times [6.9, 8.9] \times [-2.36, -2.32]$$

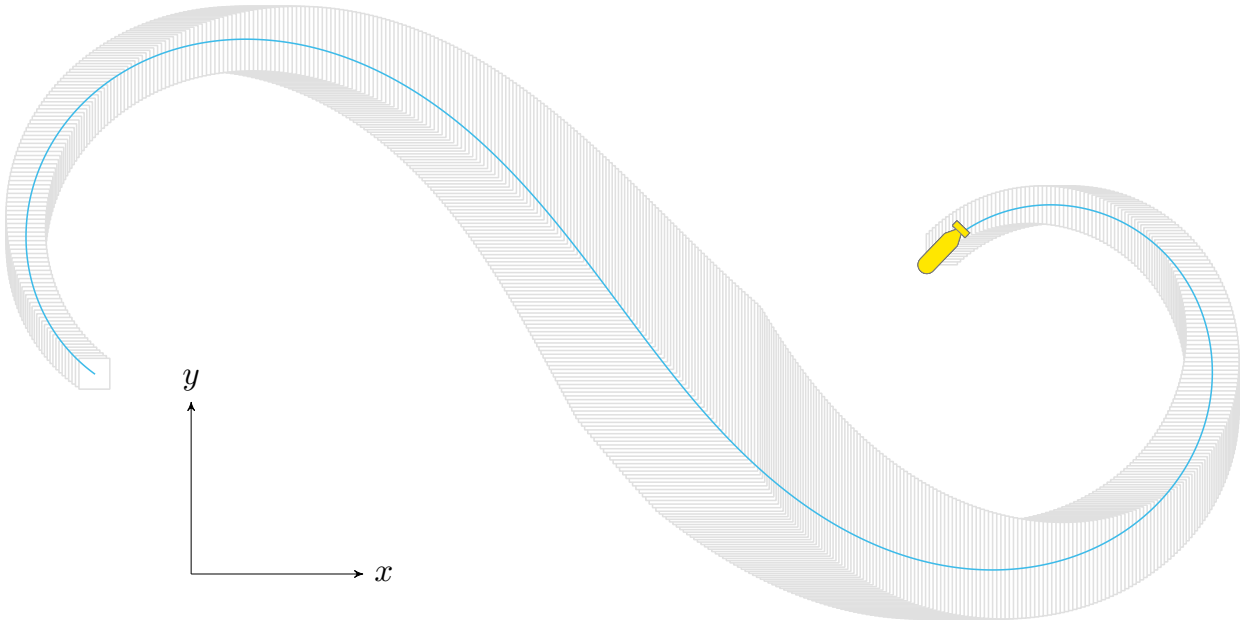


Figure: In forward/backward, uncertainties are maximal in the middle of the mission. No possible comparison with CAPD.

Section 5

Dead Reckoning of an Underwater Robot

The University Of Sheffield.

ENSTA Bretagne

Dead Reckoning of an Underwater Robot

Application on real data

Figure: DAURADE AUV managed by DGA Techniques Navales Brest and the Service hydrographique et océanographique de la Marine, during an experiment in the Rade de Brest, October 2015.

Dead Reckoning of an Underwater Robot

Application on real data

Classical kinematic model of an underwater robot:

$$\begin{cases} \dot{\mathbf{p}} &= \mathbf{R}(\psi,\theta,\varphi) \cdot \mathbf{v}_r \\ \dot{\mathbf{v}}_r &= \mathbf{a}_r - \boldsymbol{\omega}_r \wedge \mathbf{v}_r \end{cases}$$

Where:

- $\mathcal{R}_0$ is the absolute inertial coordinate system
- $\mathcal{R}_1$ is robot's own coordinate system
- (ψ,θ,φ) and $\mathbf{R}(\psi,\theta,\varphi)$ are Euler angles and matrix
- $\mathbf{p} = (p_x, p_y, p_z)$ gives the center of the robot in $\mathcal{R}_0$
- $\mathbf{v}_r$ is the speed vector in $\mathcal{R}_1$
- $\mathbf{a}_r$ is the acceleration vector in $\mathcal{R}_1$
- $\boldsymbol{\omega}_r = (\omega_x, \omega_y, \omega_z)$ is the rotation vector of the robot relative to $\mathcal{R}_0$ expressed in $\mathcal{R}_1$

Dead Reckoning of an Underwater Robot

Application on real data

Classical kinematic model of an underwater robot:

$$\begin{cases} \dot{\mathbf{p}} &= \mathbf{R}(\psi,\theta,\varphi) \cdot \mathbf{v}_r \\ \dot{\mathbf{v}}_r &= \mathbf{a}_r - \boldsymbol{\omega}_r \wedge \mathbf{v}_r \end{cases}$$

Sensed values:
Speed vector $\mathbf{v}_r$, acceleration vector $\mathbf{a}_r$, Euler angles (ψ,θ,φ)

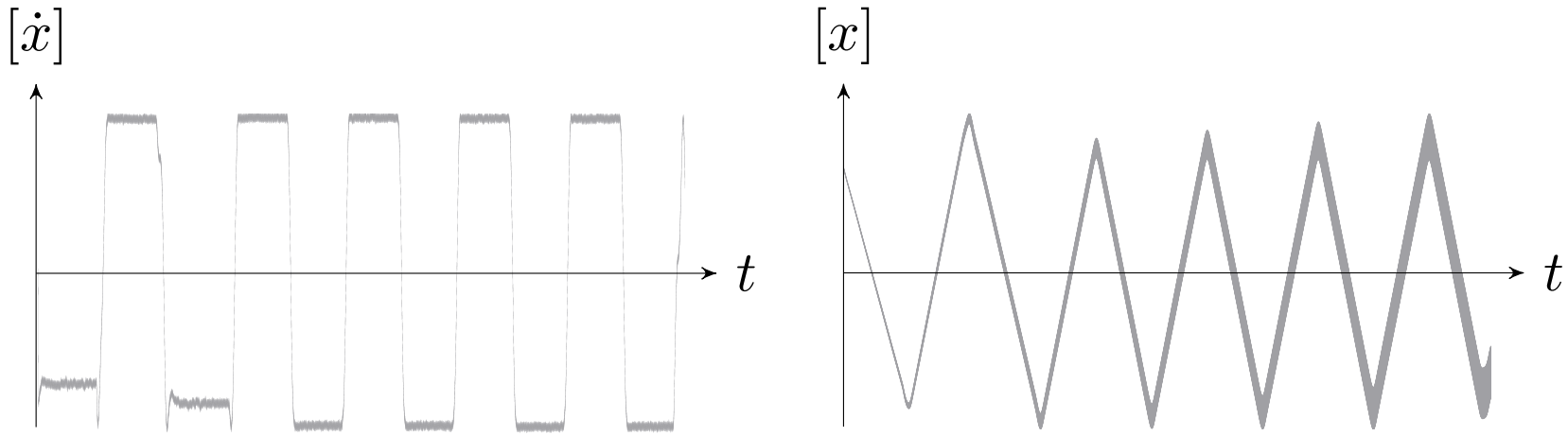
Corresponding tubes:

- feeded with sensor data:
  $[\mathbf{v}_r](\cdot)$, $[\mathbf{a}_r](\cdot)$, $[\psi](\cdot)$, $[\theta](\cdot)$, $[\varphi](\cdot)$
- to compute:
  $[\mathbf{p}](\cdot)$

Dead Reckoning of an Underwater Robot

Application on real data

Presenting Daurade's tubes $[\dot{x}](\cdot)$ (velocity along x in $\mathcal{R}_0$) and $[x](\cdot)$ (position along x in $\mathcal{R}_0$):



- $[\dot{x}](\cdot)$ – tube's thickness coming from measurements remains constant
- $[x](\cdot)$ – tube becomes thicker depicting cumulative errors due to the dead-reckoning method and $[\dot{x}](\cdot)$ non-zero thickness

Dead Reckoning of an Underwater Robot

Application on real data

Mission map after one hour:

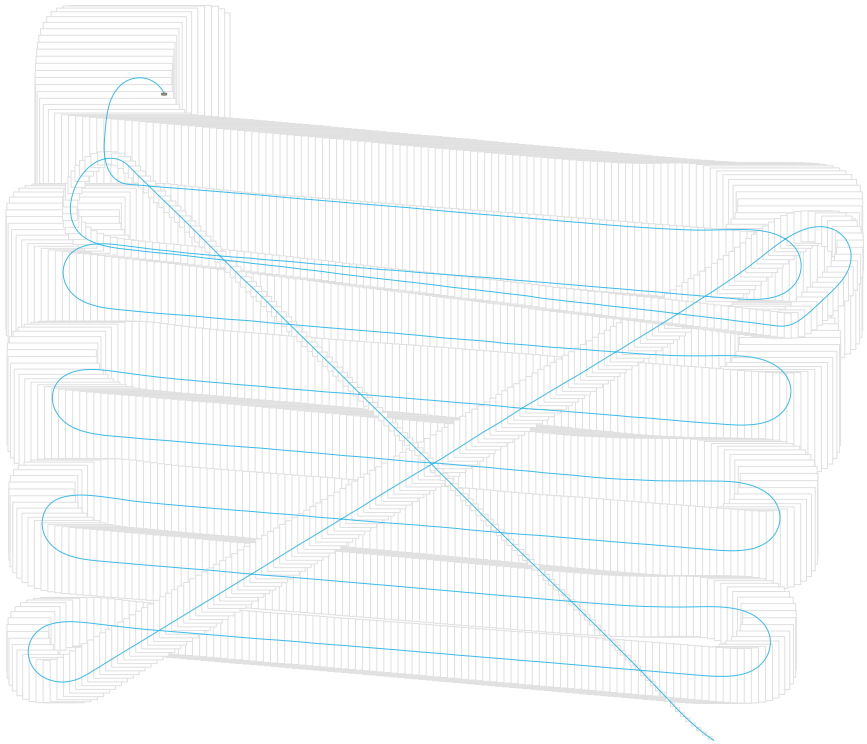


Figure: Blue line: Daurade's true trajectory, given by an ultra-short baseline (USBL), a system made of underwater acoustic sensors for positioning purposes. Gray boxes: tubes $[x](\cdot) \times [y](\cdot)$ projected on the world frame. Thanks to the DVL sensor, the drift is limited.

Conclusion

Tube programming provides a simple and efficient constraint-based method for the guaranteed interval computation of mobile robots trajectories.

Advantages:

- competitive method for robotics applications
- approach much simpler and more general than other existing methods

Drawbacks:

- pessimism added by implementation
- computation time?

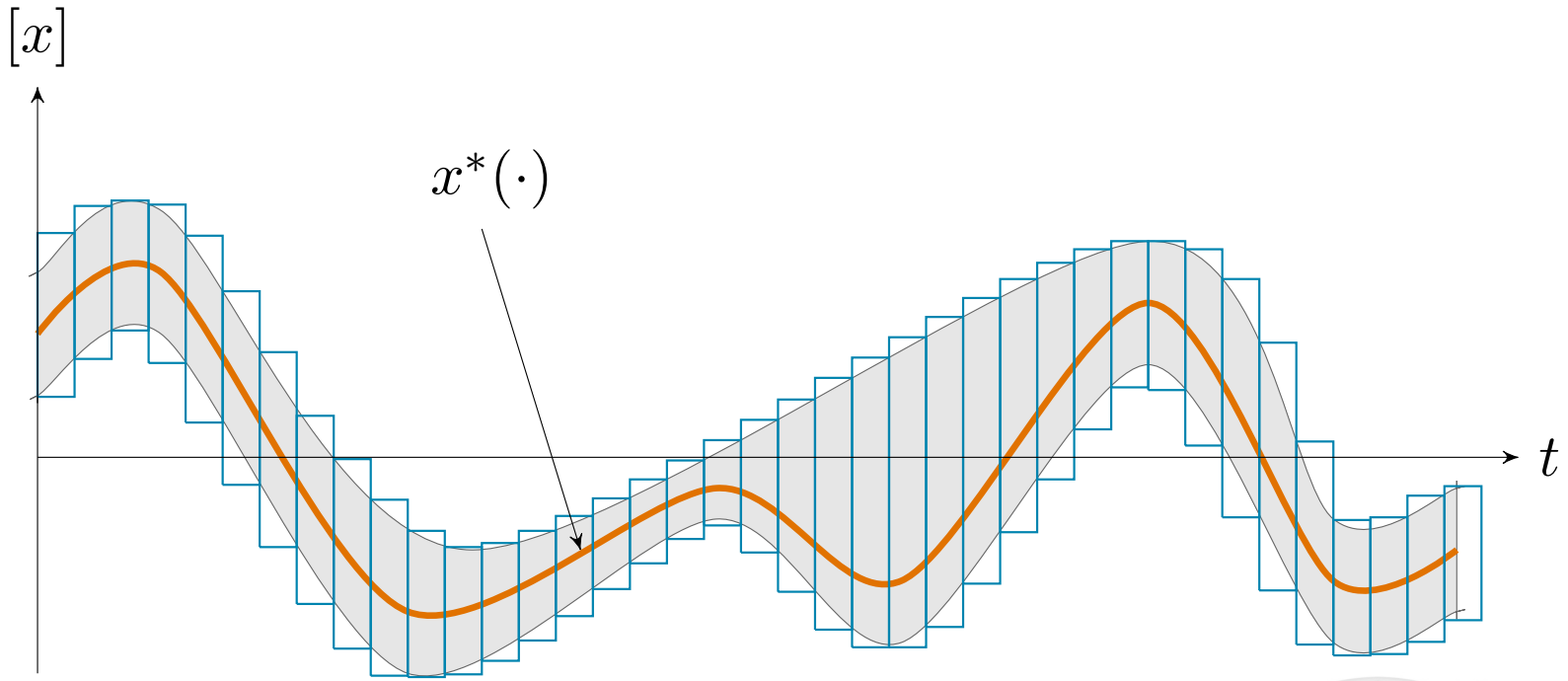
The University Of Sheffield.

ENSTA Bretagne

Conclusion

An open-source and IBEX-based tube library will be soon available:

IBEX library
`https://github.com/ibex-team/ibex-lib`

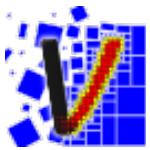


Support:

DGA
Direction Générale de l'Armement

Tools:

IBEX library
used for interval arithmetic, contractor programming



VIBES
used for rendering

References:

[Mackworth1977] A. K. Mackworth,
Consistency in networks of relations,
Artificial Intelligence, 8(1):99–118, 1977.

[Chabert2009] G. Chabert, L. Jaulin,
Contractor Programming,
Artificial Intelligence, 173:1079–1100, 2009.

[LeBars2012] F. Le Bars, J. Sliwka, O. Reynet, L. Jaulin,
State estimation with fleeting data,
Automatica, 48(2):381–387, 2012.

[Aubry2013] C. Aubry, R. Desmare, L. Jaulin,
Loop detection of mobile robots using interval analysis,
Automatica, 2013.

[CAPD] Capd, Computer Assisted Proofs in Dynamics group, a C++ package for rigorous numerics,
`http://capd.ii.uj.edu.pl`

Section 8

Appendix

Appendix

Tubes: contractors

Contractor based on the observation $[y_1]$ made at time $[t_1]$.

$$[x](t) = [x](t) \cap \left([y_1] + \int_{t_1^-}^{t} [\dot{x}](\tau) d\tau \right) \ , \ t \in [t_1^-, t_1^+]$$

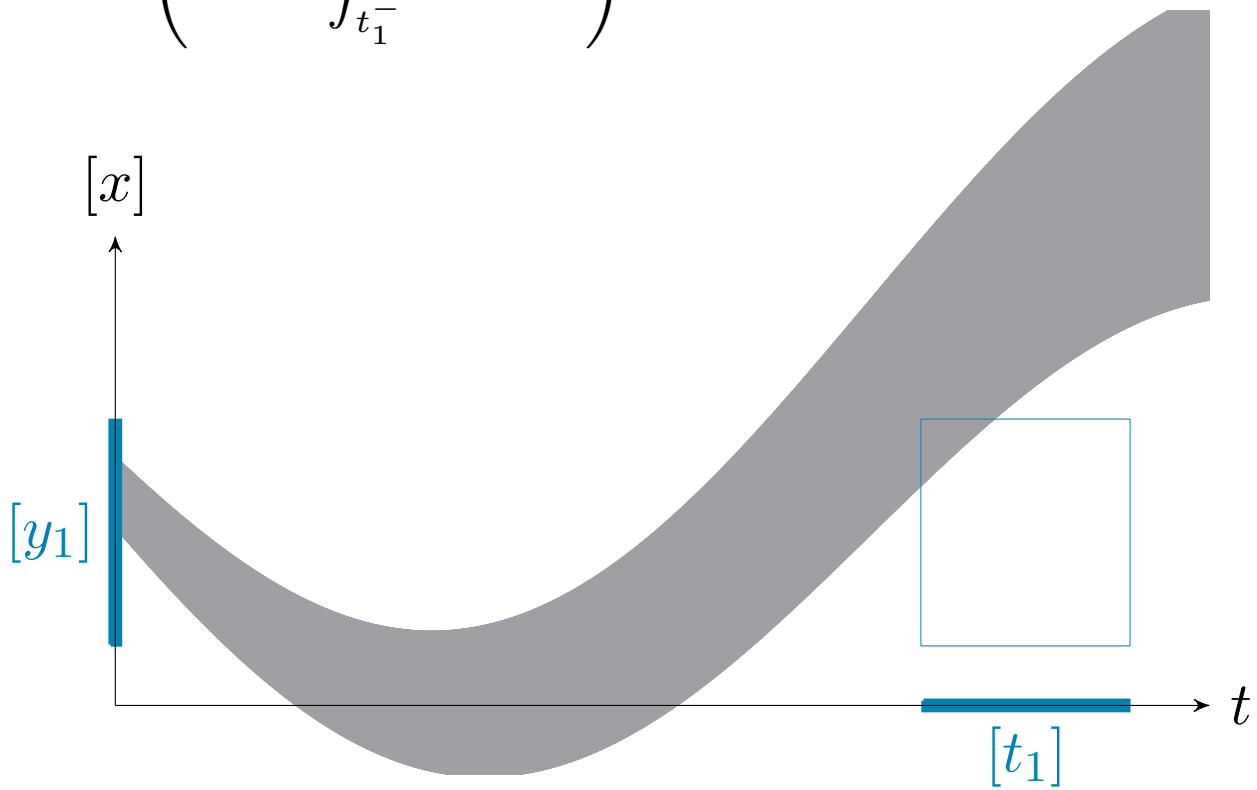


Figure: tube $[x](t)$ before contraction

Appendix

Tubes: contractors

Contractor based on the observation $[y_1]$ made at time $[t_1]$.

$$[x](t) = [x](t) \cap \left([y_1] + \int_{t_1^-}^{t} [\dot{x}](\tau) d\tau \right) \ , \ t \in [t_1^-, t_1^+]$$

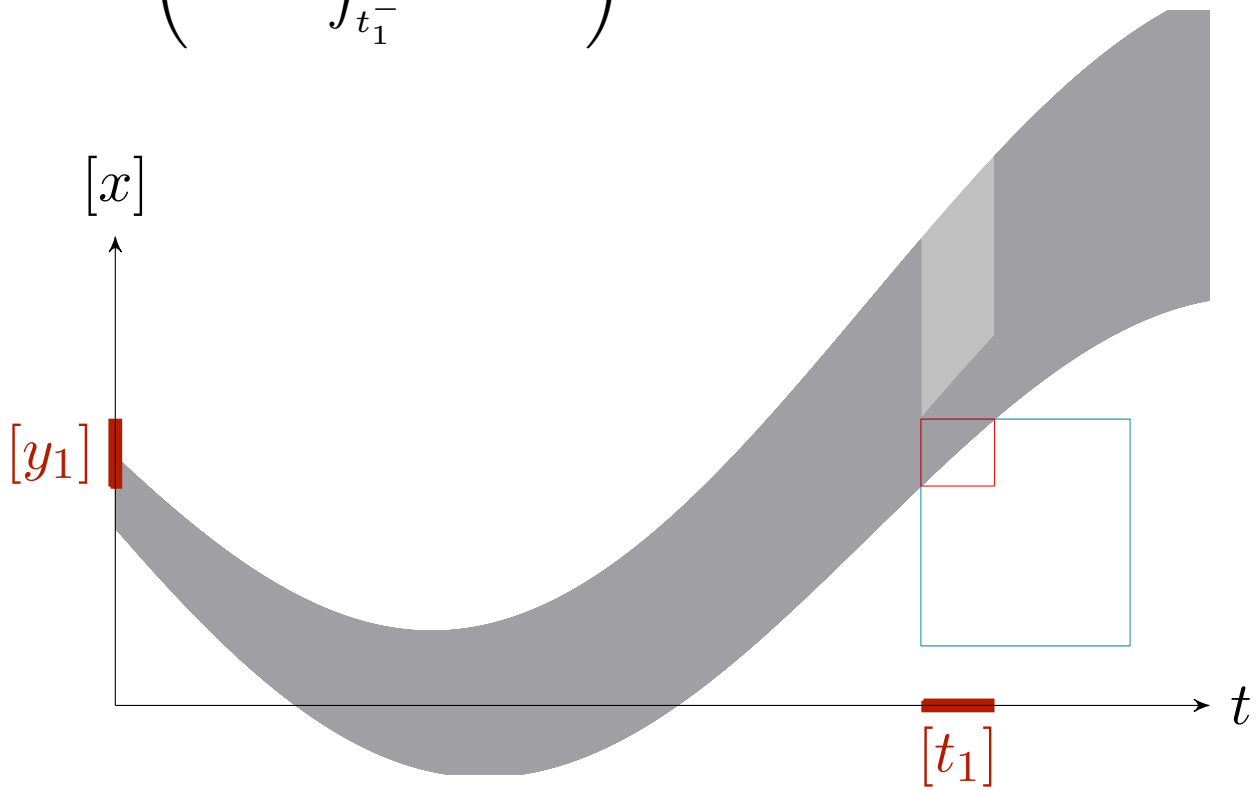


Figure: contraction of tube $[x](t)$ and both $[y_1]$ and $[t_1]$

Appendix

Tubes: contractors

Contractor based on the observation $[y_1]$ made at time $[t_1]$.

$$[x](t) = [x](t) \cap \left([y_1] + \int_{t_1^-}^{t} [\dot{x}](\tau) d\tau \right) \ , \ t \in [t_1^-, t_1^+]$$

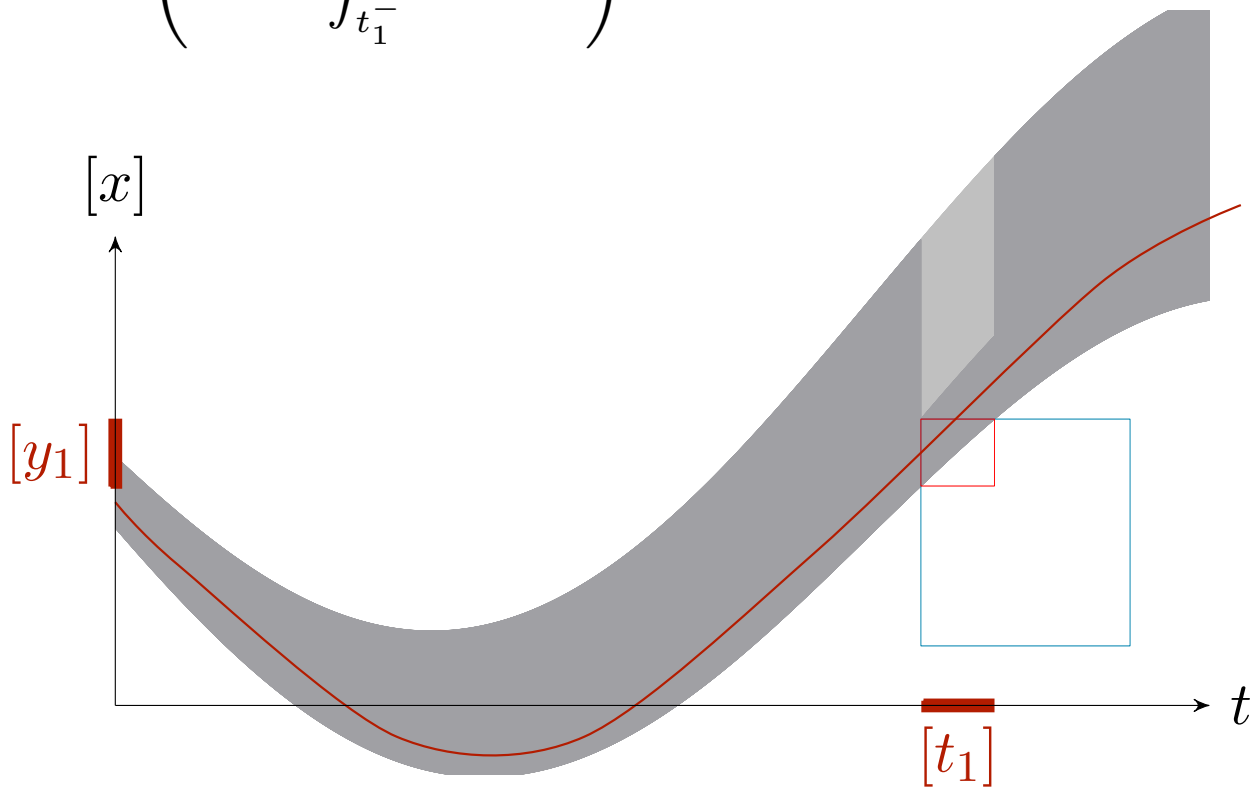


Figure: contraction of tube $[x](t)$ and both $[y_1]$ and $[t_1]$

Appendix

Tubes: contractors

Contractor based on the evolution function: $[\dot{x}] = \mathbf{f}([x], t)$

$$[x](k) = [x](k) \cap \left([x](k-1) + \int_0^{\delta t} [\dot{x}](k) d\tau \right)$$

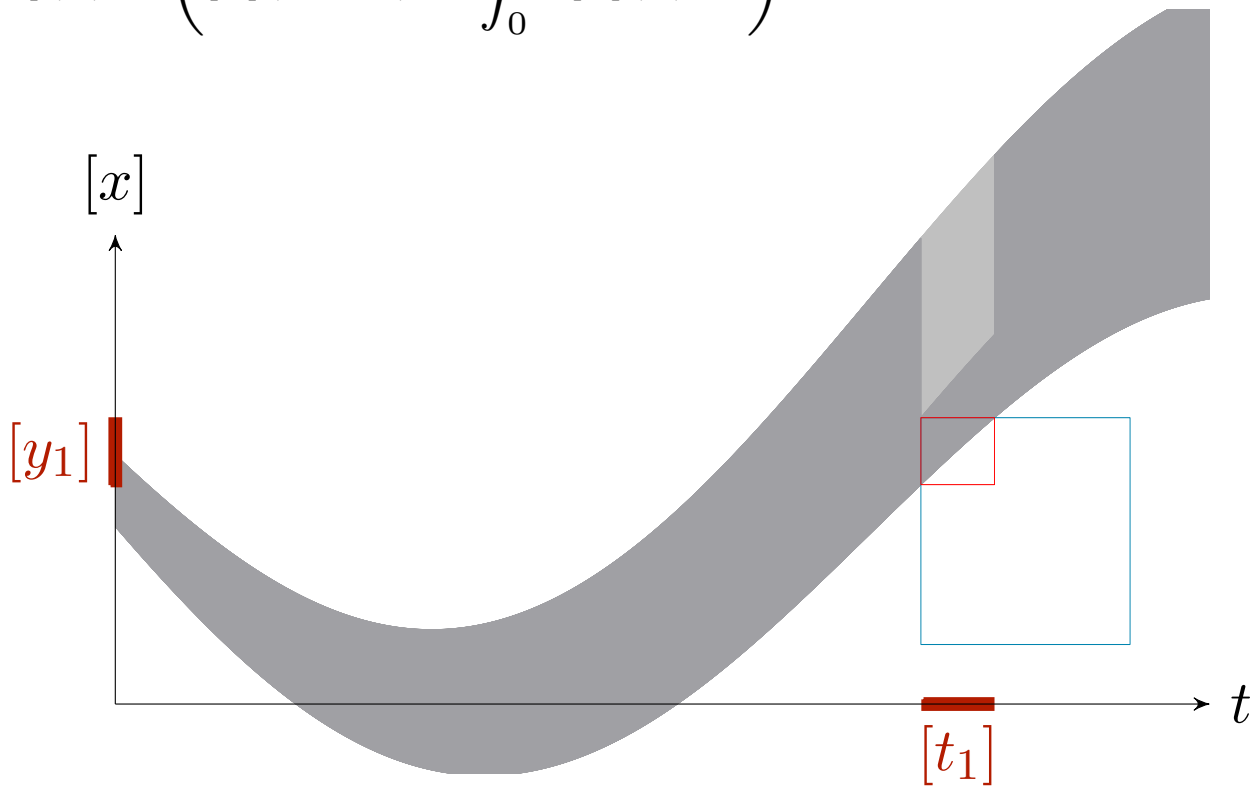


Figure: contraction of tube $[x](t)$ and both $[y_1]$ and $[t_1]$

Appendix

Tubes: contractors

Contractor based on the evolution function: $[\dot{x}] = \mathbf{f}([x], t)$

$$[x](k) = [x](k) \cap \left([x](k-1) + \int_0^{\delta t} [\dot{x}](k) d\tau \right)$$

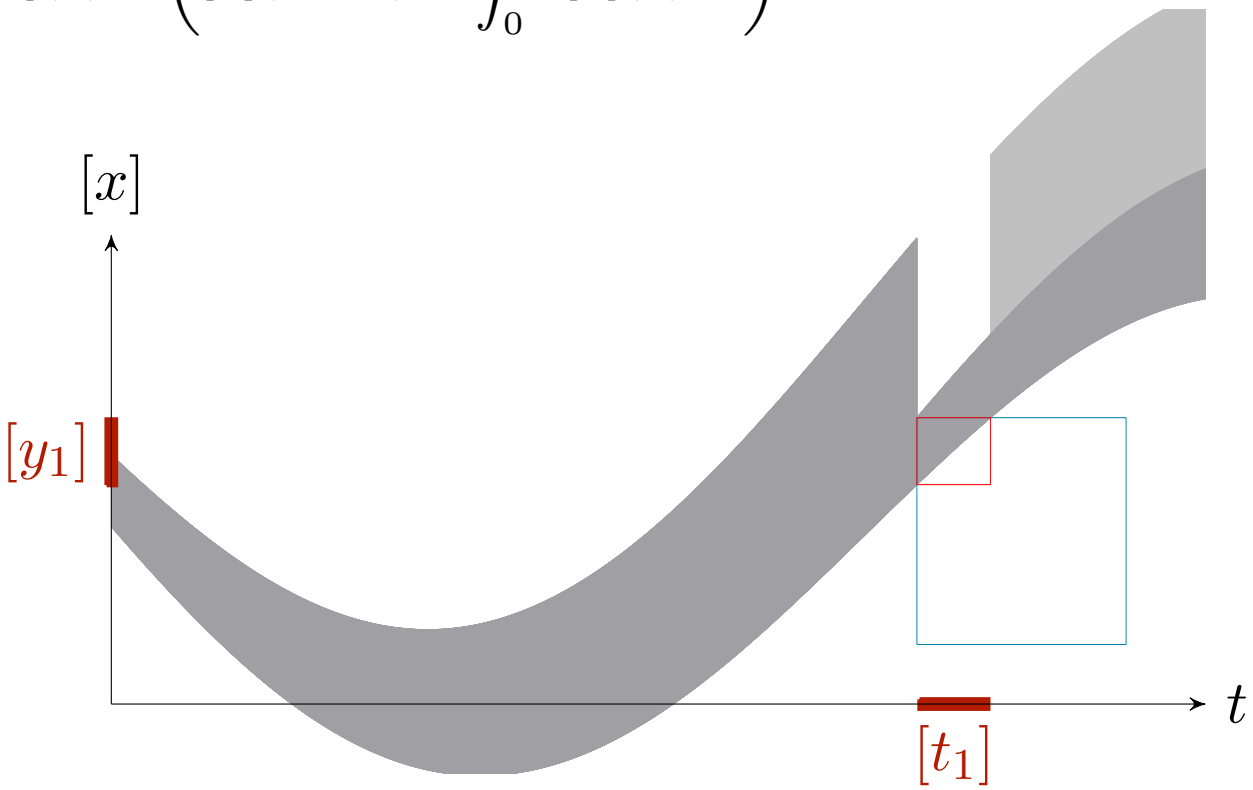


Figure: tube contraction in forward

Appendix

Tubes: contractors

Contractor based on the evolution function: $[\dot{x}] = \mathbf{f}([x], t)$

$$[x](k) = [x](k) \cap \left([x](k+1) - \int_0^{\delta t} [\dot{x}](k) d\tau \right)$$

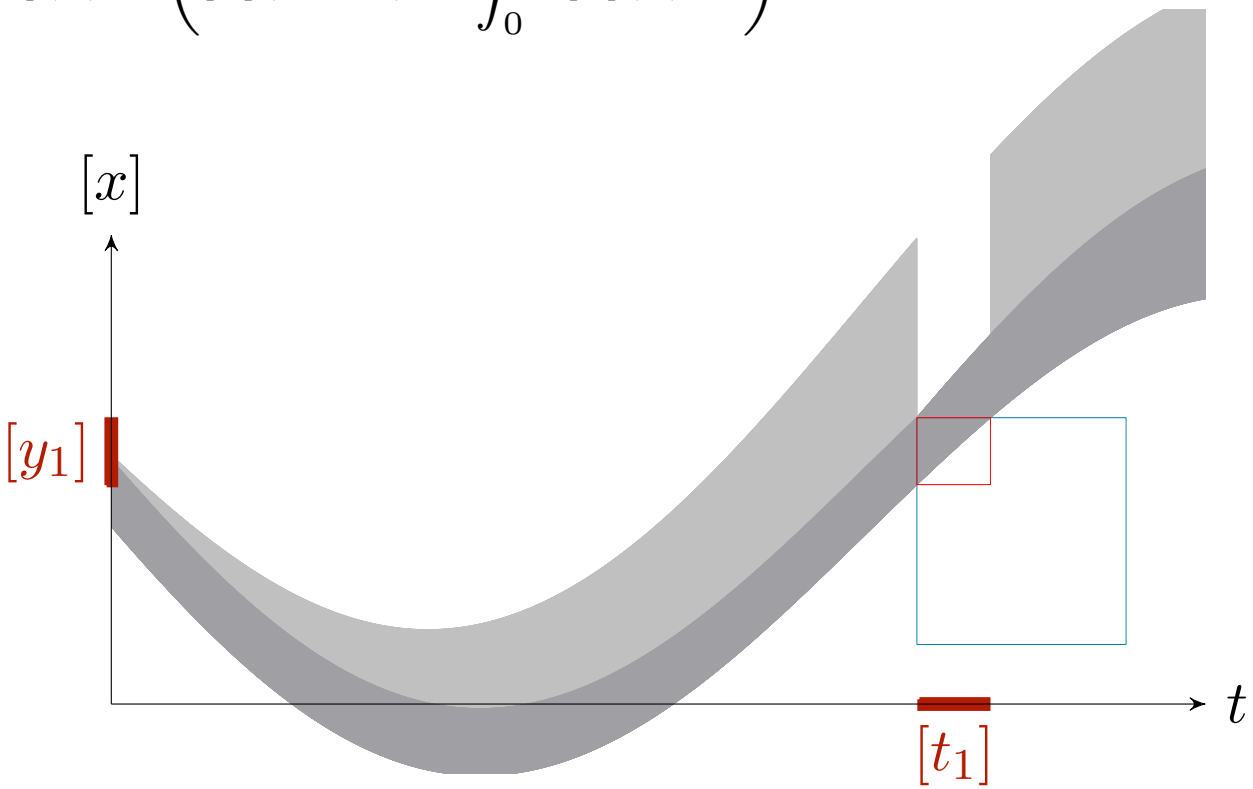


Figure: tube contraction in forward/backward