# Lie symmetries applied to guaranteed integration: application to mobile robotics localisation

Julien DAMERS

L. Jaulin, S. Rohou

Hanover, 20th July 2022

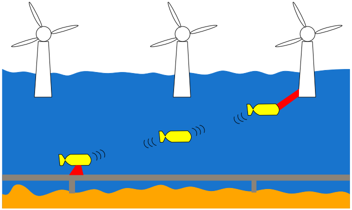
Section 1

# Introduction

# Context of this research

- Applications:
  - Offshore wind farms
  - Underwater mining
  - Underwater sensor fields
- Constraint:
  - No possibility to return to the surface before the end of the mission
  - Cheaper sensors (swarms)

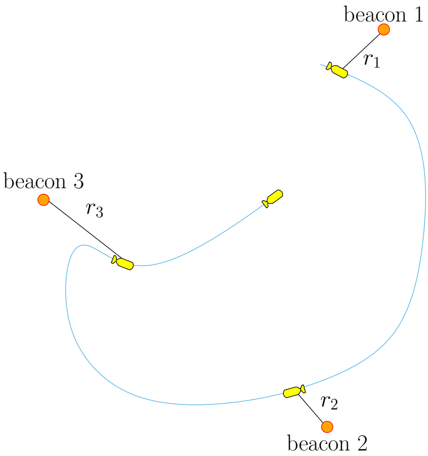
⟶ Problem to localise our robot



Autonomous Underwater Vehicles (AUV) used as data mules and for monitoring

# The localisation problem

- Aim:
  - Locate the robot offline to replace data on map
- Data available
  - Behaviour of the robot (evolution function)
  - Range-only measurements
  - Completely unknown initial condition



# Outline

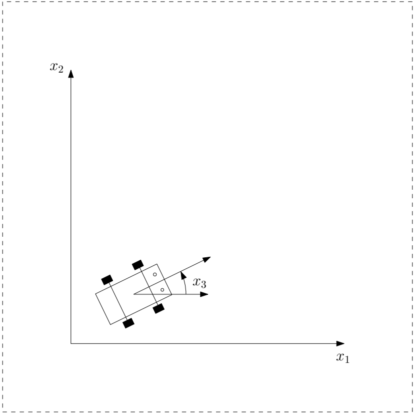
Section 2

# Modelling a robot

# Modelling a robot

The robot state is represented by a vector. For instance:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$



# Differential equation

- Behaviour modeled by the evolution function

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

# Differential equation

- Behaviour modeled by the evolution function

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

- In the examples we will consider in this presentation, $\mathbf{u}(t)$ is known.

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

# Differential equation

- Behaviour modeled by the evolution function

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

- In the examples we will consider in this presentation, $\mathbf{u}(t)$ is known.

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

- Finding solution of an Initial Value Problem (IVP)

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), t \in T \\ \mathbf{x}(t=0) = \mathbf{x}_0 \in \mathbb{R}^n \end{cases}$$

# Dynamical systems

## Definition (Flow function)

A dynamical system can be represented by a function $\Phi : T \times \mathcal{S} \to \mathcal{S}$ which follows the properties below:

# Dynamical systems

## Definition (Flow function)

A dynamical system can be represented by a function $\Phi : T \times \mathcal{S} \rightarrow \mathcal{S}$ which follows the properties below:

1. $t \in T$ will be the evolution parameter and $T$ the time set;

# Dynamical systems

## Definition (Flow function)

A dynamical system can be represented by a function $\Phi : T \times \mathcal{S} \to \mathcal{S}$ which follows the properties below:

1. $t \in T$ will be the evolution parameter and $T$ the time set;
2. $\mathcal{S}$ is the state space;

# Dynamical systems

## Definition (Flow function)

A dynamical system can be represented by a function $\Phi : T \times \mathcal{S} \to \mathcal{S}$ which follows the properties below:

1. $t \in T$ will be the evolution parameter and $T$ the time set;
2. $\mathcal{S}$ is the state space;
3. $\Phi(0, .)$ is the identity function, *i.e.* $\forall \mathbf{x} \in \mathcal{S}, \Phi(0, \mathbf{x}) = \mathbf{x}$ ;

# Dynamical systems

## Definition (Flow function)

A dynamical system can be represented by a function $\Phi : T \times \mathcal{S} \to \mathcal{S}$ which follows the properties below:

1. $t \in T$ will be the evolution parameter and $T$ the time set;
2. $\mathcal{S}$ is the state space;
3. $\Phi(0, .)$ is the identity function, *i.e.* $\forall \mathbf{x} \in \mathcal{S}, \Phi(0, \mathbf{x}) = \mathbf{x}$ ;
4. For any $\mathbf{x} \in \mathcal{S}$, and $t, \tau \in T, \Phi(t, \Phi(\tau, \mathbf{x})) = \Phi(t + \tau, \mathbf{x})$.

# Dynamical systems

## Definition (Flow function)

A dynamical system can be represented by a function $\Phi : T \times \mathcal{S} \to \mathcal{S}$ which follows the properties below:

1. $t \in T$ will be the evolution parameter and $T$ the time set;
2. $\mathcal{S}$ is the state space;
3. $\Phi(0, .)$ is the identity function, *i.e.* $\forall \mathbf{x} \in \mathcal{S}, \Phi(0, \mathbf{x}) = \mathbf{x}$ ;
4. For any $\mathbf{x} \in \mathcal{S}$, and $t, \tau \in T, \Phi(t, \Phi(\tau, \mathbf{x})) = \Phi(t + \tau, \mathbf{x})$.

- We will focus on continuous time systems where $T = \mathbb{R}$.
- An analytic expression of the flow function is rarely available

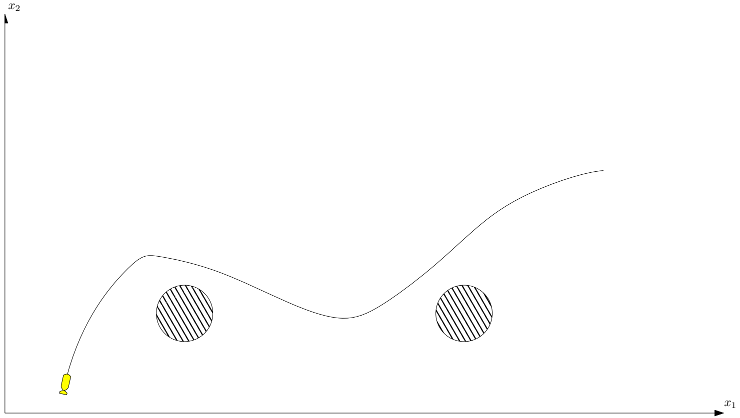
Section 3

# Towards a new guaranteed integration method

# Why a new guaranteed integration method ?

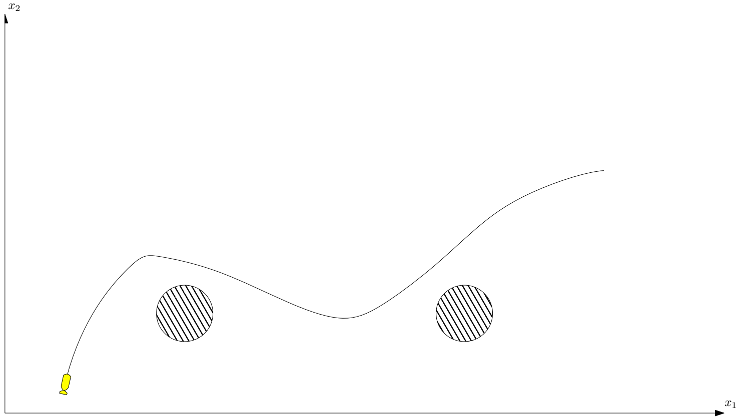
# Why a new guaranteed integration method ?

- Need for guarantee as we are working with complex systems



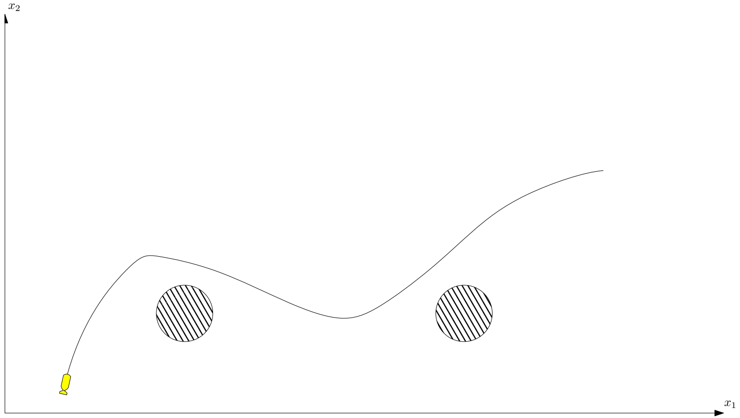
# Why a new guaranteed integration method ?

- Need for guarantee as we are working with complex systems
- Conventional tools can be quite slow when performing numerous integrations



# Why a new guaranteed integration method ?

- Need for guarantee as we are working with complex systems
- Conventional tools can be quite slow when performing numerous integrations
- Conventional tools cannot deal with large initial conditions



# Guaranteed integration tools, state of the art

2 main methods:

# Guaranteed integration tools, state of the art

2 main methods:

- Taylor expansion method

# Guaranteed integration tools, state of the art

2 main methods:

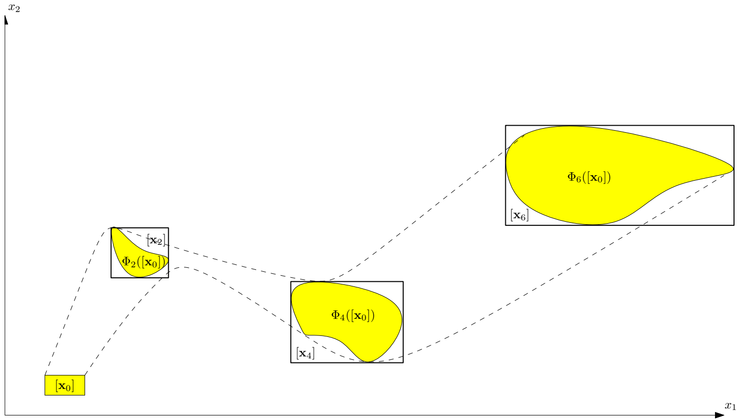
- Taylor expansion method
- Hermite-Obreshkov

# Guaranteed integration tools, state of the art

2 main methods:

- Taylor expansion method
- Hermite-Obreshkov

Many solvers: CAPD, VNODE, COSY ...

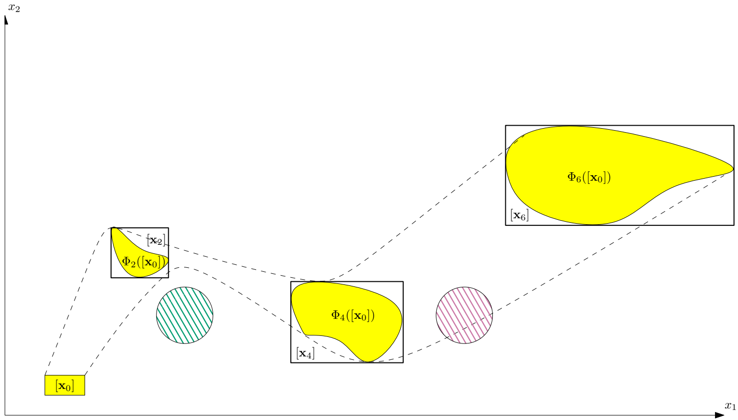


# Guaranteed integration tools, state of the art

2 main methods:

- Taylor expansion method
- Hermite-Obreshkov

Many solvers: CAPD, VNODE, COSY ...

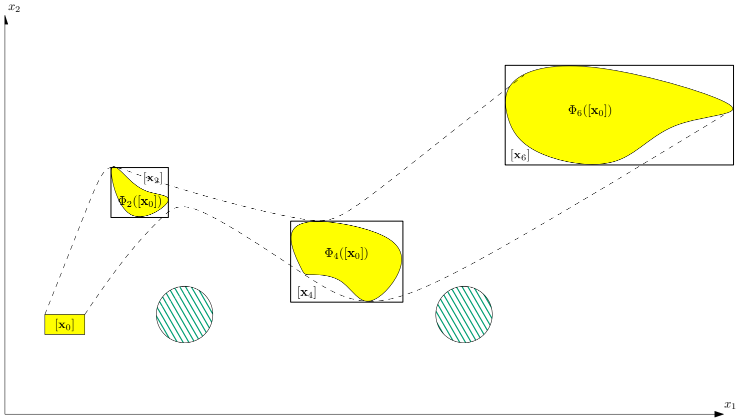


# Guaranteed integration tools, state of the art

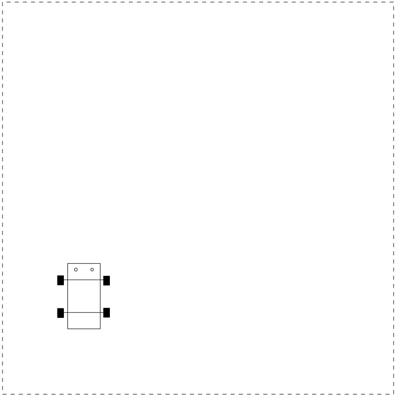
2 main methods:

- Taylor expansion method
- Hermite-Obreshkov

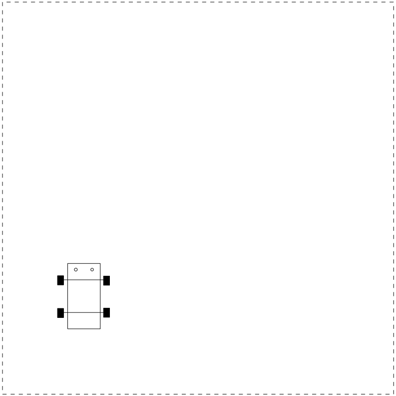
Many solvers: CAPD, VNODE, COSY ...

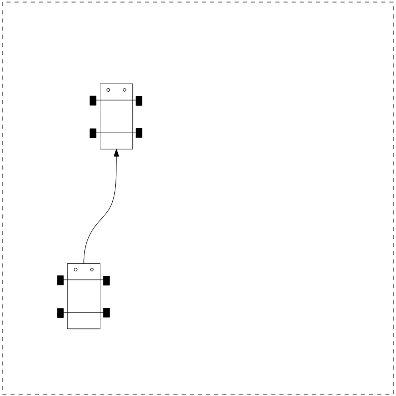


# Principle

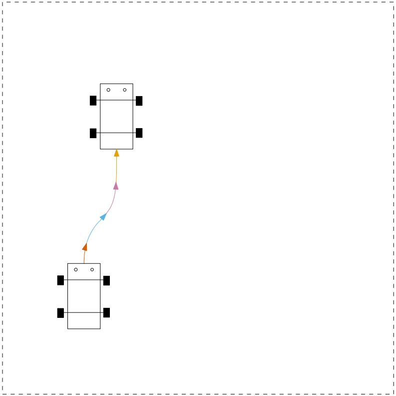


# Principle

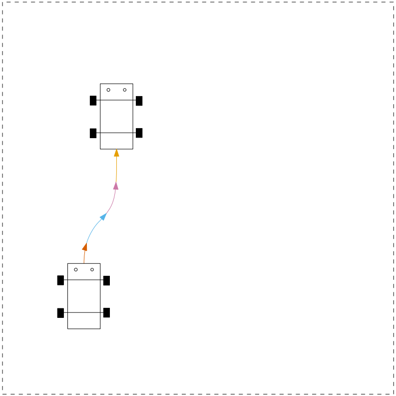


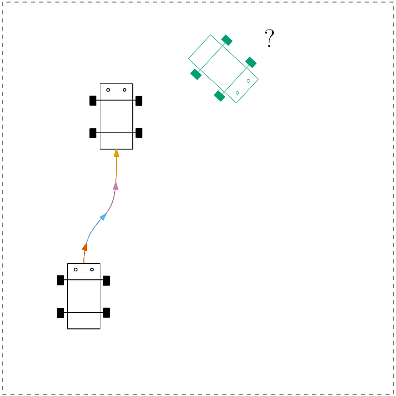


# Principle

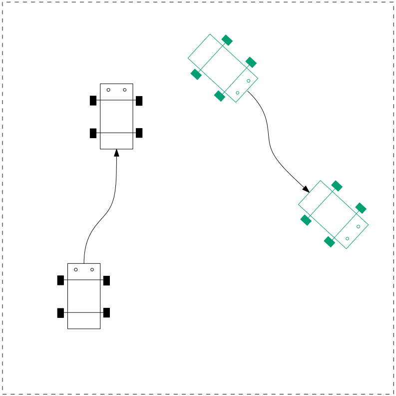


# Principle

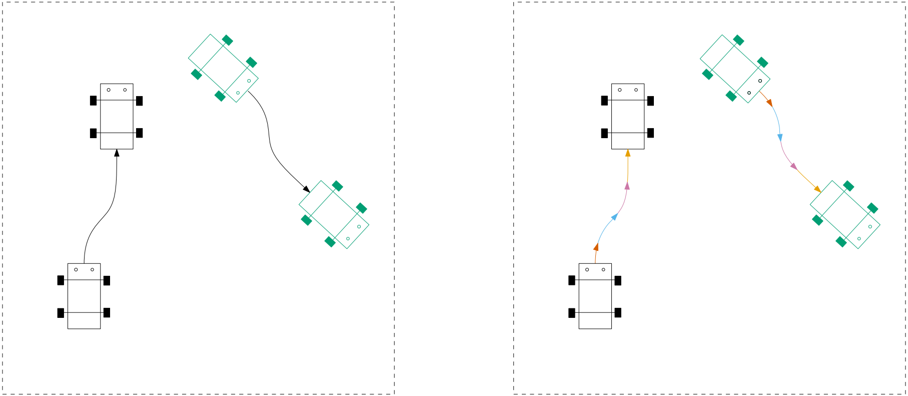




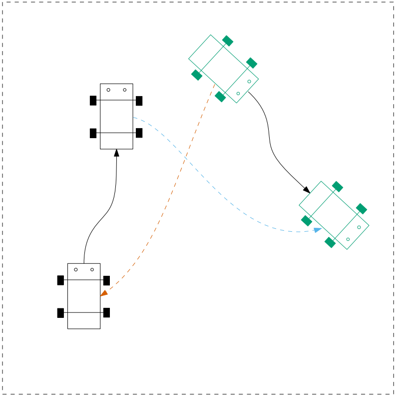
# Principle



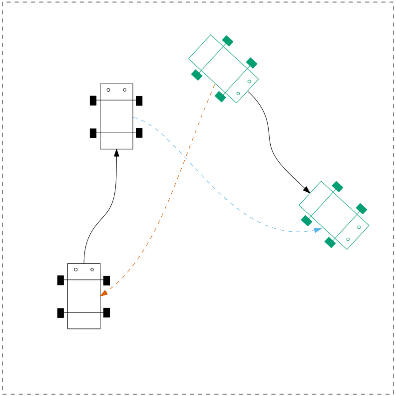
# Principle

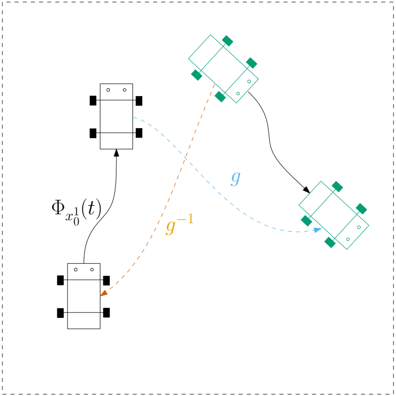


# Principle



# Principle

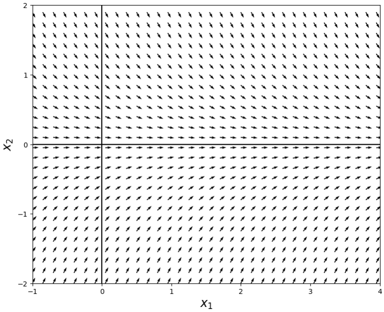




# Hints from the vector field

Let us consider the system defined by:

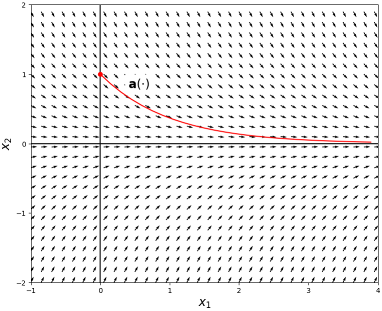
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$$



# Hints from the vector field

Let us consider the system defined by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$$

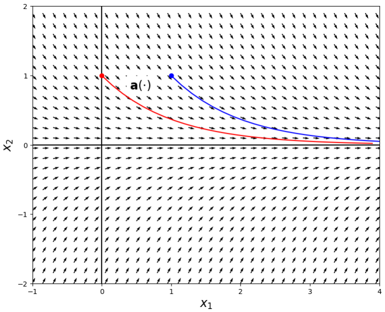


# Hints from the vector field

Let us consider the system defined by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$$

- A translation symmetry along $Ox_1$

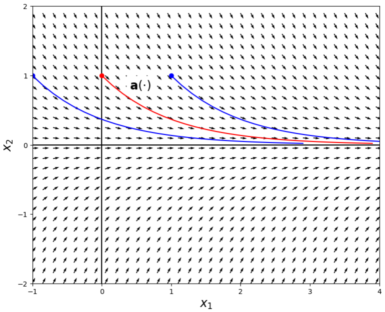


# Hints from the vector field

Let us consider the system defined by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$$

- A translation symmetry along $Ox_1$

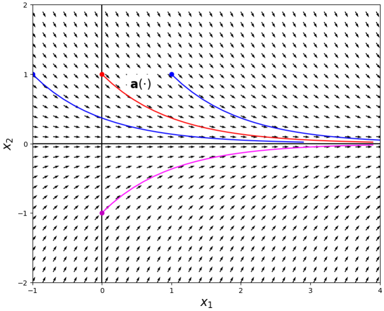


# Hints from the vector field

Let us consider the system defined by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$$

- A translation symmetry along $Ox_1$
- A mirror symmetry over $Ox_1$

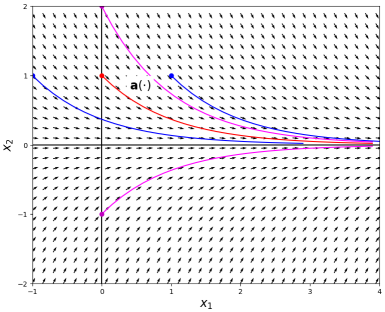


# Hints from the vector field

Let us consider the system defined by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$$

- A translation symmetry along $Ox_1$
- A mirror symmetry over $Ox_1$



# Action of a diffeomorphisms

**Definition (Action of diffeomorphisms)**

Consider a state equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{g} \in \text{diff}(\mathbb{R}^n)$. We define the action $\bullet$ of $\mathbf{g}$ on $\mathbf{f}$ as

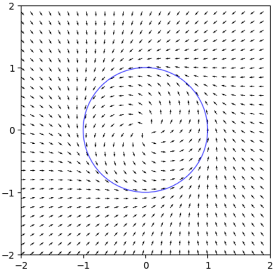
$$\mathbf{g} \bullet \mathbf{f} = \left( \frac{d\mathbf{g}}{d\mathbf{x}} \circ \mathbf{g}^{-1} \right) \cdot \left( \mathbf{f} \circ \mathbf{g}^{-1} \right)$$

# Action of a diffeomophisms

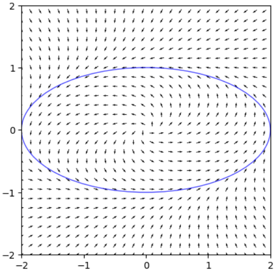
## Example

Consider:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} -x_1^3 - x_1 x_2^2 + x_1 - x_2 \\ -x_2^3 - x_1^2 x_2 + x_1 + x_2 \end{pmatrix} \text{ and } \mathbf{h}(\mathbf{x}) = \begin{pmatrix} 2x_1 \\ x_2 \end{pmatrix}$$




(d) **f**



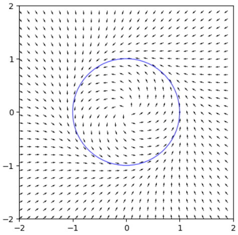

(e) **h • f**

# Action of a diffeomophisms

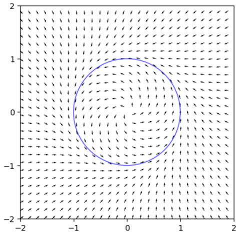
## Example

Consider the previous system and the following function:

$$\mathbf{r}(\mathbf{x}) = \begin{pmatrix} \cos\left(\frac{\pi}{4}\right) x_1 - \sin\left(\frac{\pi}{4}\right) x_2 \\ \cos\left(\frac{\pi}{4}\right) x_2 + \sin\left(\frac{\pi}{4}\right) x_1 \end{pmatrix}$$




(f) **f**




(g) **r • f**

The vector field stays the same !

# Lie symmetry

## Definition (Lie symmetry)

$\mathbf{g} \in \text{diff}(\mathbb{R}^n)$ is a symmetry of $\mathbf{f}$ if the action $\bullet$ of $\mathbf{g}$ on $\mathbf{f}$ leaves $\mathbf{f}$ unchanged i.e

$$\mathbf{g} \bullet \mathbf{f} = \mathbf{f}.$$

Lie symmetries are also called **stabilisers**.

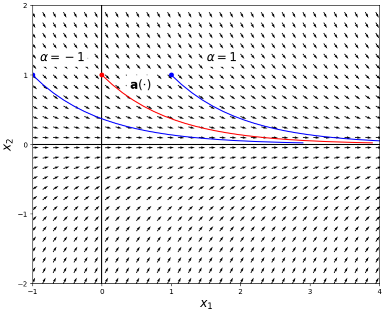
# Translation symmetry

$$\mathbf{g}_{\alpha} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + \alpha \\ x_2 \end{pmatrix}$$

# Translation symmetry

$$\mathbf{g}_{\alpha} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + \alpha \\ x_2 \end{pmatrix}$$

$$\begin{aligned}
\mathbf{g}_{\alpha} \bullet \mathbf{f}(\mathbf{x}) &= \left( \frac{d\mathbf{g}_{\alpha}}{d\mathbf{x}} \circ \mathbf{g}_{\alpha}^{-1} \right) \cdot \left( \mathbf{f} \circ \mathbf{g}_{\alpha}^{-1} \right) (\mathbf{x}) \\
&= \left( \frac{d\mathbf{g}_{\alpha}}{d\mathbf{x}} \cdot \mathbf{f} \right) \circ \mathbf{g}_{\alpha}^{-1}(\mathbf{x}) \\
&= \left( \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -x_2 \end{pmatrix} \right) \circ \begin{pmatrix} x_1 - \alpha \\ x_2 \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ -x_2 \end{pmatrix} \\
&= \mathbf{f}(\mathbf{x})
\end{aligned}$$



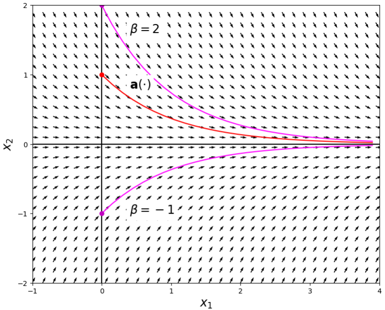
# Mirror-symmetry

$$\mathbf{g}_{\beta} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \\ \beta x_2 \end{pmatrix}$$

# Mirror-symmetry

$$\mathbf{g}_\beta : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \\ \beta x_2 \end{pmatrix}$$

$$\begin{aligned} \mathbf{g}_\beta \bullet \mathbf{f}(\mathbf{x}) &= \left( \frac{d\mathbf{g}_\beta}{d\mathbf{x}} \circ \mathbf{g}_\beta^{-1} \right) \cdot \left( \mathbf{f} \circ \mathbf{g}_\beta^{-1} \right) (\mathbf{x}) \\ &= \left( \frac{d\mathbf{g}_\beta}{d\mathbf{x}} \cdot \mathbf{f} \right) \circ \mathbf{g}_\beta^{-1}(\mathbf{x}) \\ &= \left( \begin{pmatrix} 1 & 0 \\ 0 & \beta \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -x_2 \end{pmatrix} \right) \circ \begin{pmatrix} x_1 \\ \frac{x_2}{\beta} \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ -x_2 \end{pmatrix} \\ &= \mathbf{f}(\mathbf{x}) \end{aligned}$$



# Complete symmetry

$$\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$$

# Complete symmetry

$$\mathbf{g}_\mathbf{p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$$

## Definition (Lie group of symmetry)

Consider a state equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ and a manifold $\mathbb{P}$. A Lie group $G_\mathbf{p}$ of symmetries is a family of diffeomorphisms $\mathbf{g}_\mathbf{p} \in \text{diff}(\mathbb{R}^n)$ parameterised by $\mathbf{p} \in \mathbb{P}$ such that:

- $G_\mathbf{p}$ is a Lie group with respect to the composition $\circ$,
- $\forall \mathbf{p} \in \mathbb{P}, \mathbf{g}_\mathbf{p} \bullet \mathbf{f} = \mathbf{f}$.

# Determine the flow function

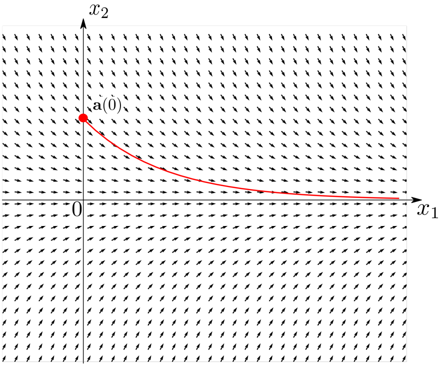
**Objective**: Determine the flow function $\Phi_t(\mathbf{x})$.

# Determine the flow function

**Objective**: Determine the flow function $\Phi_t(\mathbf{x})$.

We have:

- A **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red)
- A transformation function $\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$

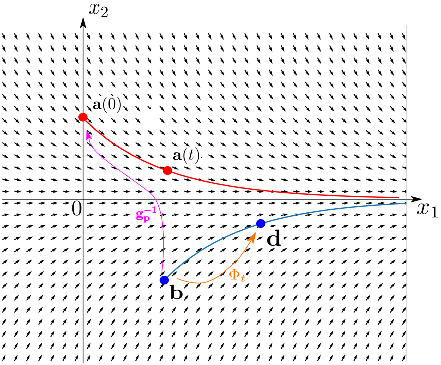


# Determine the flow function

**Objective**: Determine the flow function $\Phi_t(\mathbf{x})$.

We have:

- A **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red)
- A transformation function $\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$



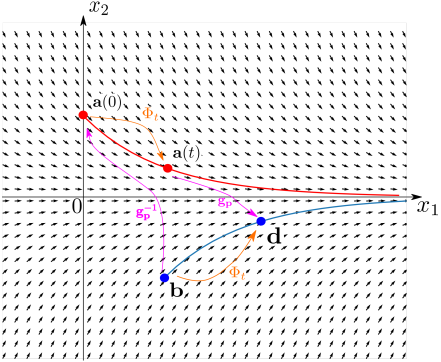

$$\mathbf{d} = \Phi_t(\mathbf{b})$$

# Determine the flow function

**Objective**: Determine the flow function $\Phi_t(\mathbf{x})$.

We have:

- A **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red)
- A transformation function $\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$




$$\mathbf{d} = \Phi_t(\mathbf{b}) = \mathbf{g_p}(\mathbf{a}(t)) = \mathbf{g_p} \circ \mathbf{a}(t)$$

$$\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$$

# Determine the flow function

**Objective**: Determine the flow function $\Phi_t(\mathbf{x})$.

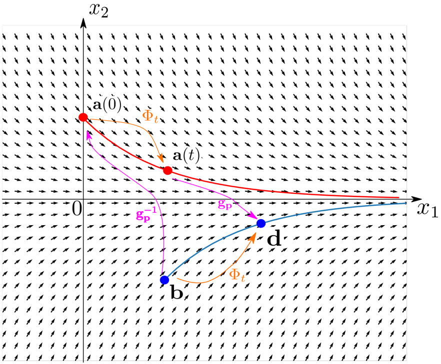
We have:

- A **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red)
- A transformation function

$$\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$$

Therefore

$$\Phi_t(x) = \mathbf{g_p} \circ \mathbf{a}(t)$$




$$\mathbf{d} = \Phi_t(\mathbf{b}) = \mathbf{g_p}(\mathbf{a}(t)) = \mathbf{g_p} \circ \mathbf{a}(t)$$

$$\mathbf{g_p} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + p_1 \\ p_2 x_2 \end{pmatrix}$$

# The transport function

To find the right value of $\mathbf{p}$, we must solve

$$\mathbf{g_p}(\mathbf{a}(0)) = \mathbf{b},$$

in order to express $\mathbf{p}$ using only $\mathbf{a}(0)$ and $\mathbf{b}$.

# The transport function

To find the right value of $\mathbf{p}$, we must solve

$$\mathbf{g_p}(\mathbf{a}(0)) = \mathbf{b},$$

in order to express $\mathbf{p}$ using only $\mathbf{a}(0)$ and $\mathbf{b}$.

Using the previous example :

$$\mathbf{g_p}(\mathbf{a}(0)) = \mathbf{b} \iff \begin{pmatrix} a_1 + p_1 \\ p_2 a_2 \end{pmatrix} = \mathbf{b}$$

$$\iff \mathbf{p} = \begin{pmatrix} b_1 - a_1 \\ \frac{b_2}{a_2} \end{pmatrix}$$

# The transport function

To find the right value of $\mathbf{p}$, we must solve

$$\mathbf{g_p}(\mathbf{a}(0)) = \mathbf{b},$$

in order to express $\mathbf{p}$ using only $\mathbf{a}(0)$ and $\mathbf{b}$.

Using the previous example :

$$\mathbf{g_p}(\mathbf{a}(0)) = \mathbf{b} \iff \begin{pmatrix} a_1 + p_1 \\ p_2 a_2 \end{pmatrix} = \mathbf{b}$$

$$\iff \mathbf{p} = \begin{pmatrix} b_1 - a_1 \\ \frac{b_2}{a_2} \end{pmatrix}$$

We introduce a new tool, the **transport function** denoted $\mathbf{h}(\mathbf{x}, \mathbf{a})$ such that:

$$\mathbf{p} = \mathbf{h}(\mathbf{b}, \mathbf{a}) = \begin{pmatrix} b_1 - a_1 \\ \frac{b_2}{a_2} \end{pmatrix}.$$

# The transport function

## Definition (Transport function)

Consider a transitive Lie group of symmetries $G_{\mathbf{p}}$(i.e it only has one orbit). In this case, there exists a function $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{P}$ such that $\mathbf{h}(\mathbf{x}, \mathbf{a})$ corresponds to the displacement $\mathbf{p}$ to be chosen so that the point $\mathbf{a}$ is moved to $\mathbf{x}$ by $\mathbf{g_p}$ , which means:

$$\mathbf{g}_{\mathbf{h}(\mathbf{x},\mathbf{a})}(\mathbf{a}) = \mathbf{x}$$

.

# The flow function

- Reference:

$$\mathbf{a}(t) \in [\mathbf{a}](t), \mathbf{a}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

# The flow function

- Reference:

$$\mathbf{a}(t) \in [\mathbf{a}](t), \mathbf{a}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- Symmetry:

$$\mathbf{g}_{\mathbf{p}} : \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \mapsto \begin{pmatrix} u_1 + p_1 \\ p_2 u_2 \end{pmatrix},$$

# The flow function

- Reference:

$$\mathbf{a}(t) \in [\mathbf{a}](t), \mathbf{a}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- Symmetry:

$$\mathbf{g_p} : \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \mapsto \begin{pmatrix} u_1 + p_1 \\ p_2 u_2 \end{pmatrix},$$

- Transport function:

$$\mathbf{h}(\mathbf{x}, \mathbf{a}) = \begin{pmatrix} x_1 - a_1 \\ \frac{x_2}{a_2} \end{pmatrix},$$

# The flow function

- Reference:

$$\mathbf{a}(t) \in [\mathbf{a}](t), \mathbf{a}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- Symmetry:

$$\mathbf{g}_\mathbf{p} : \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \mapsto \begin{pmatrix} u_1 + p_1 \\ p_2 u_2 \end{pmatrix},$$

- Transport function:

$$\mathbf{h}(\mathbf{x}, \mathbf{a}) = \begin{pmatrix} x_1 - a_1 \\ \frac{x_2}{a_2} \end{pmatrix},$$

$$\begin{aligned} \Phi_t(\mathbf{x}) &= \mathbf{g}_\mathbf{p} \circ \mathbf{a}(t) \\ &= \mathbf{g}_{\mathbf{h}(\mathbf{x},\mathbf{a}_0)} \circ \mathbf{a}(t) \\ &= \mathbf{g}_{x_1,x_2} \circ \begin{pmatrix} a_1(t) \\ a_2(t) \end{pmatrix} \\ &= \begin{pmatrix} a_1(t) + x_1 \\ x_2 \cdot a_2(t) \end{pmatrix} \\ &= \begin{pmatrix} t + x_1 \\ x_2 \cdot e^{-t} \end{pmatrix} \end{aligned}$$

# The flow function

- Reference:

$$\mathbf{a}(t) \in [\mathbf{a}](t), \mathbf{a}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- Symmetry:

$$\mathbf{g_p} : \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \mapsto \begin{pmatrix} u_1 + p_1 \\ p_2 u_2 \end{pmatrix},$$

- Transport function:

$$\mathbf{h}(\mathbf{x}, \mathbf{a}) = \begin{pmatrix} x_1 - a_1 \\ \frac{x_2}{a_2} \end{pmatrix},$$

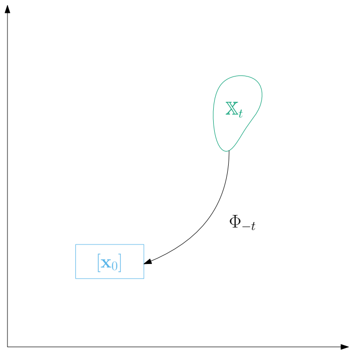
$$\begin{aligned} \Phi_t(\mathbf{x}) &= \mathbf{g_p} \circ \mathbf{a}(t) \\ &= \mathbf{g}_{\mathbf{h}(\mathbf{x},\mathbf{a}_0)} \circ \mathbf{a}(t) \\ &= \mathbf{g}_{x_1,x_2} \circ \begin{pmatrix} a_1(t) \\ a_2(t) \end{pmatrix} \\ &= \begin{pmatrix} a_1(t) + x_1 \\ x_2 \cdot a_2(t) \end{pmatrix} \\ &= \begin{pmatrix} t + x_1 \\ x_2 \cdot e^{-t} \end{pmatrix} \end{aligned}$$

We finally have a analytic expression for the flow !

# A set inversion problem

With the flow function $\mathbf{\Phi}_t$, performing a guaranteed integration for an uncertain initial condition is equivalent to solving a set inversion problem.
Consider a uncertain initial box $[\mathbf{x}_0]$ for which we want to find the image set by $\mathbf{\Phi}_t$. We want to find the set $\mathbb{X}_t$ such that

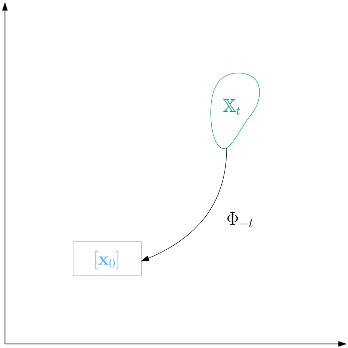
$$\mathbb{X}_t = \mathbf{\Phi}_{-t}^{-1}([\mathbf{x}_0]).$$

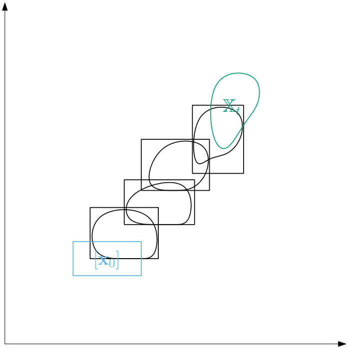


# A set inversion problem

With the flow function $\mathbf{\Phi}_t$, performing a guaranteed integration for an uncertain initial condition is equivalent to solving a set inversion problem.
Consider a uncertain initial box $[\mathbf{x}_0]$ for which we want to find the image set by $\mathbf{\Phi}_t$. We want to find the set $\mathbb{X}_t$ such that

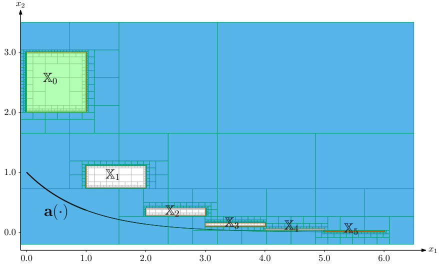
$$\mathbb{X}_t = \mathbf{\Phi}_{-t}^{-1}([\mathbf{x}_0]).$$



# Applying a SIVIA algorithm

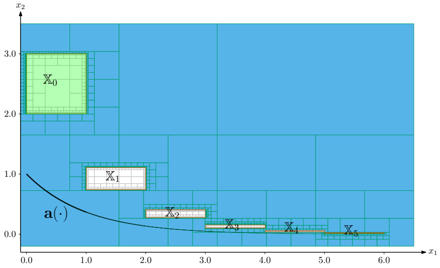
- $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$
- $[\mathbf{x}_0] = [0, 1] \times [2, 3]$



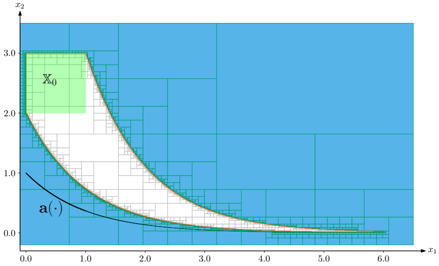

Discrete sets computation (Lie 70 ms, CAPD 300 ms)

# Applying a SIVIA algorithm

- $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ -x_2 \end{pmatrix}$
- $[\mathbf{x}_0] = [0, 1] \times [2, 3]$




Discrete sets computation (Lie 70 ms, CAPD 300 ms)




Continous set computation (229 ms)

# Pros and limits of the method

Pros:

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)
- Less computation time (less steps of operations)

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)
- Less computation time (less steps of operations)
- Easily get an outer **and** inner approximation

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)
- Less computation time (less steps of operations)
- Easily get an outer **and** inner approximation

Limits:

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)
- Less computation time (less steps of operations)
- Easily get an outer **and** inner approximation

Limits:

- Need for enough symmetries, method will not work for every systems

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)
- Less computation time (less steps of operations)
- Easily get an outer **and** inner approximation

Limits:

- Need for enough symmetries, method will not work for every systems
- Need for a reference

# Pros and limits of the method

Pros:

- Able to deal with large initial condition (no bloating effect)
- Less computation time (less steps of operations)
- Easily get an outer **and** inner approximation

Limits:

- Need for enough symmetries, method will not work for every systems
- Need for a reference

Julien Damers, Luc Jaulin, Simon Rohou. "Lie symmetries applied to interval integration". Accepted in: Automatica 2022

Section 4

## Solving the localisation problem for an unknown initial condition

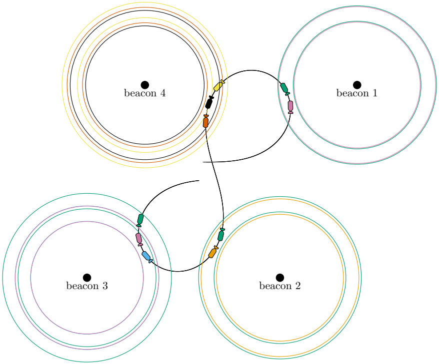
## Problem presentation

Hypotheses:

- 1 robot
- 4 beacons
- Completely unknown inital condition
- Range only measurements

Objectives:

- Estimate the initial condition
- Locate the robot



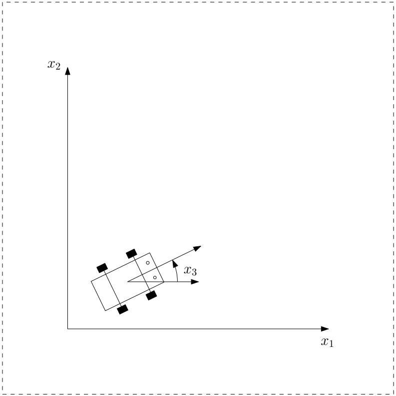
# The tank-like robot model

Let us consider the system defined by:

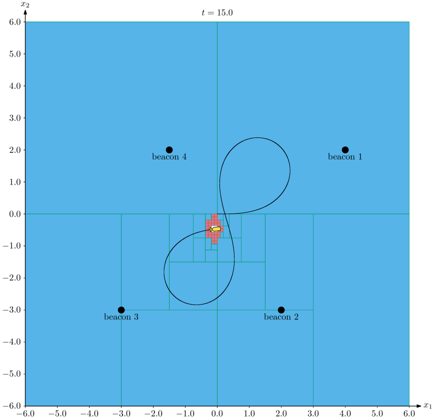
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t)) = \begin{pmatrix} u_1(t)\cos(x_3) \\ u_1(t)\sin(x_3) \\ u_2(t) \end{pmatrix}$$

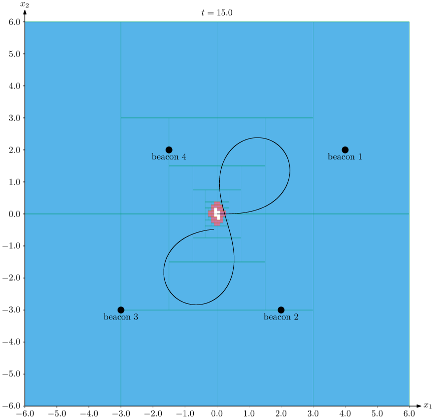
- General kinematic model
- Can be applied to a large group of robots

**In our example $\mathbf{u}(t)$ is known for all $t$**



# Result


$t = 15.0$
$x_2$
$x_1$
beacon 1
beacon 2
beacon 3
beacon 4


$t = 15.0$
$x_2$
$x_1$
beacon 1
beacon 2
beacon 3
beacon 4

Section 5

# Conclusion

# Conclusion

- Notion of transport function
- Development of a new guaranteed integration method
- Application to localisation with an unknown initial condition

# Conclusion

- Notion of transport function
- Development of a new guaranteed integration method
- Application to localisation with an unknown initial condition

Prospects:

- Solve differential inclusions $\dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}, \mathbf{u})$
- Handle both space and time displacement (sliding window)
- Apply symmetries in other context than interval analysis (particle filter)
- Compute the transport function automatically

Thank you for your attention

# Codac code

**Listing 5.1** Characterising $\mathbb{X}_3$ using the Lie integration method

```
// The uncertain initial condition
IntervalVector x_0({{0,1},{2,3}});

// The space to explore for the set inversion
IntervalVector m({{-0.1,6.5},{-0.2,3.5}});

double epsilon = 0.01; // define accuracy of paving

// define transformation function
Function phi("x1","x2","t","(x1+t;x2*exp(-t))");

// Create the general separator on phi_t with [x_0] as constraint
SepFwdBwd SepPhi(phi,x_0);

// Define the time for which we want to perform the integration
Interval t(-3,-3);

// Create the projected separator object
SepProj sepProj(SepPhi,t,epsilon);

// Perform the set inversion algorithm
vector<vector<IntervalVector>> pavings = sivia(m,sepProj,epsilon);
```