Lie Groups Applied to mobile robotics

Application to Reachability analysis

Julien DAMERS

Kopadia / ENSTA Bretagne

16 May 2022

Outline

Section 1

Context

Context of this research

- development of offshore wind farms

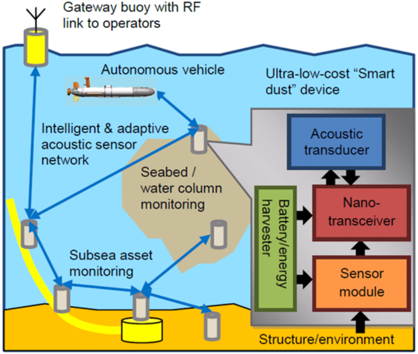


Figure: AUV used as an autonomous data mule

Source: USMART project

Context of this research

- development of offshore wind farms
- development of underwater mining

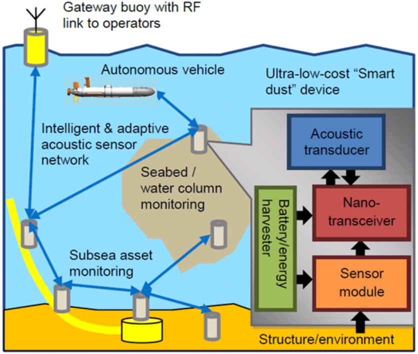


Figure: AUV used as an autonomous data mule

Source: USMART project

Context of this research

- development of offshore wind farms
- development of underwater mining
- development of underwater sensor fields

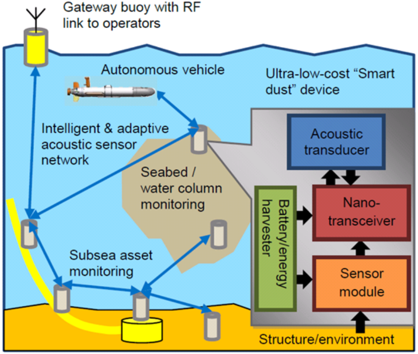


Figure: AUV used as an autonomous data mule

Source: USMART project

Context of this research

- development of offshore wind farms
- development of underwater mining
- development of underwater sensor fields
- Increasing need for data of maritime environment

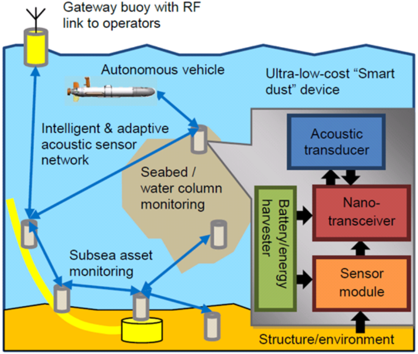


Figure: AUV used as an autonomous data mule

Source: USMART project

Context of this research

- development of offshore wind farms
- development of underwater mining
- development of underwater sensor fields
- Increasing need for data of maritime environment
- No possibility to return to the surface before the end of the mission (military)

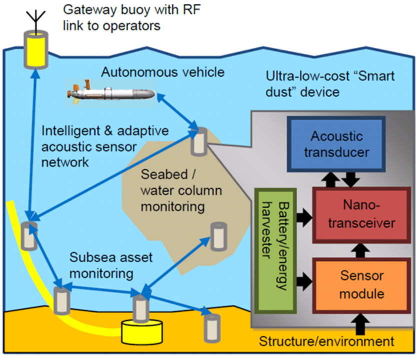


Figure: AUV used as an autonomous data mule

Source: USMART project

Context of this research

- development of offshore wind farms
- development of underwater mining
- development of underwater sensor fields
- Increasing need for data of maritime environment
- No possibility to return to the surface before the end of the mission (military)

Development of swarms of AUVs

↓

Need for cheaper units with no expensive sensors

↓

Improve algorithms of state estimation

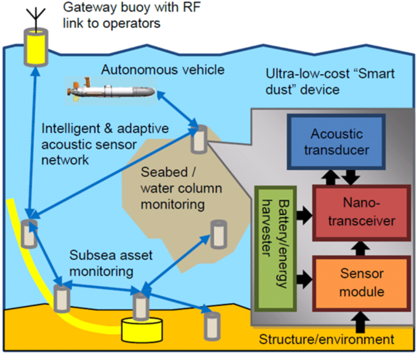


Figure: AUV used as an autonomous data mule

Source: USMART project

Section 2

Guaranteed integration method using Lie symmetries

Why a new guaranteed integration method ?

Why a new guaranteed integration method ?

- Need for guarantee as we are working with complex systems

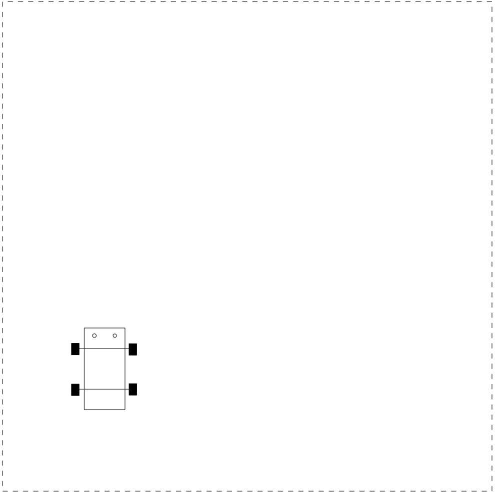
Why a new guaranteed integration method ?

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

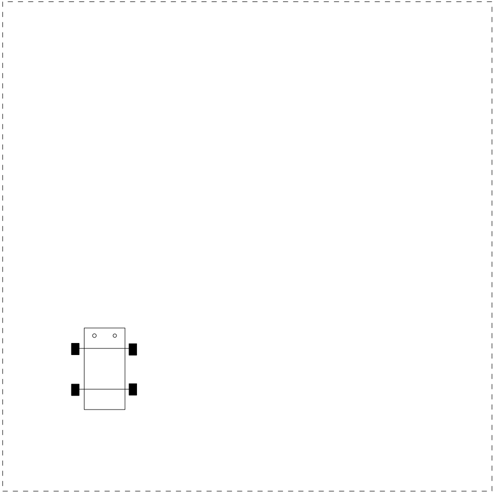
Why a new guaranteed integration method ?

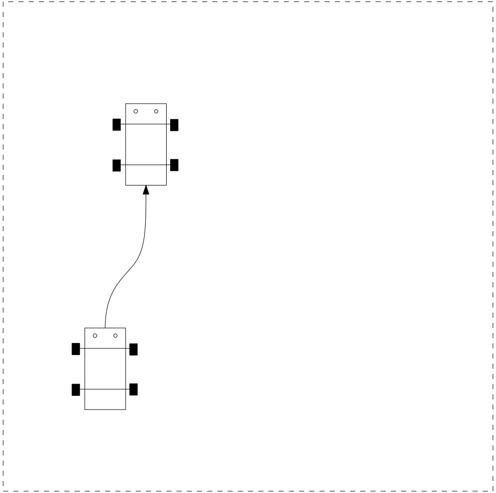
- Need for guarantee as we are working with complex systems
- Conventional tools can be quite slow when performing numerous integrations
- Conventional tools cannot deal with large initial conditions

Principle

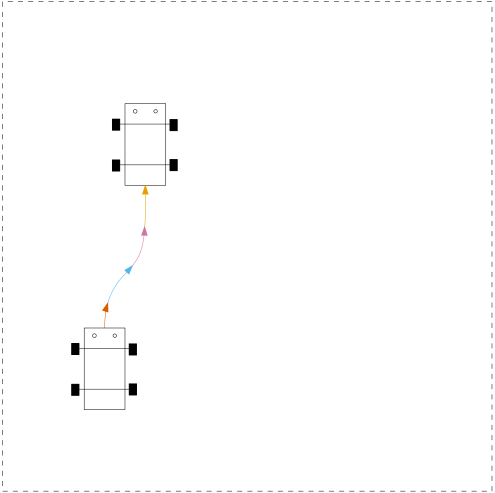


Principle

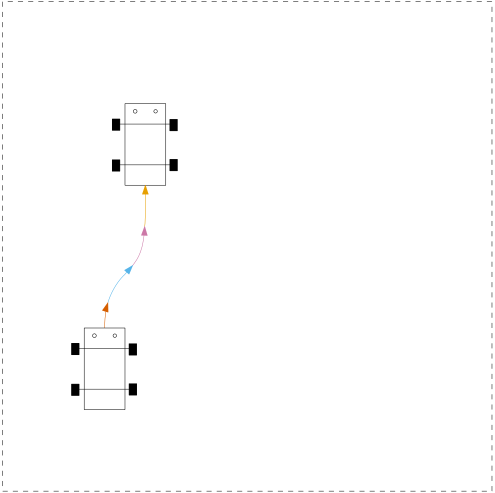


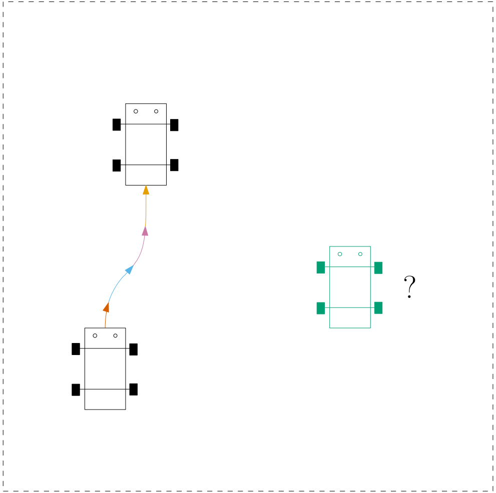


Principle

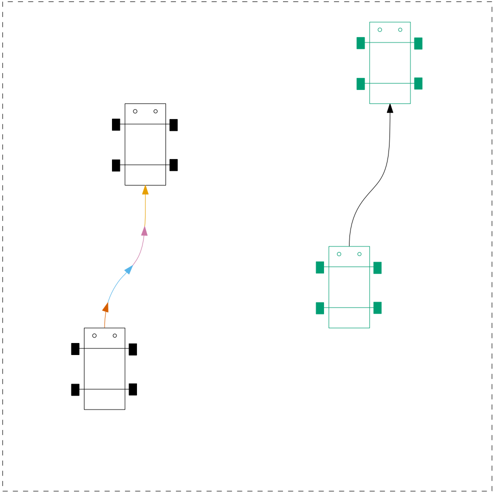


Principle

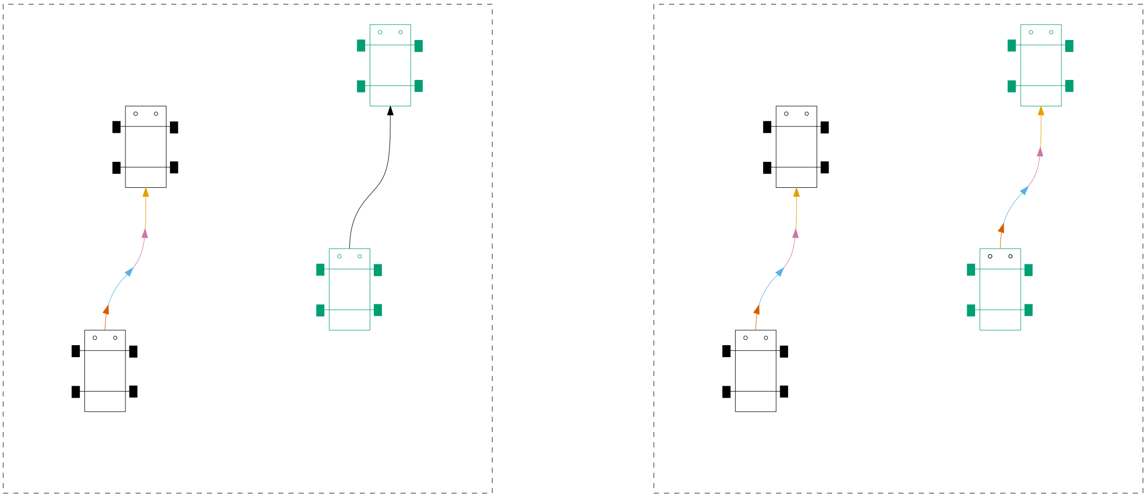




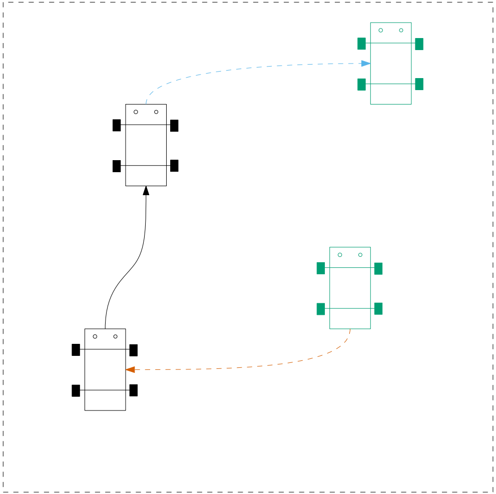
Principle



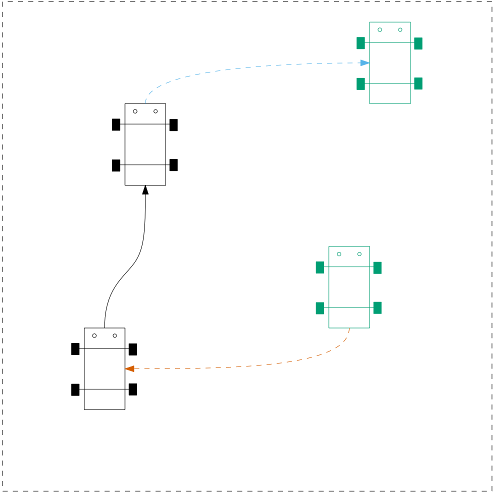
Principle

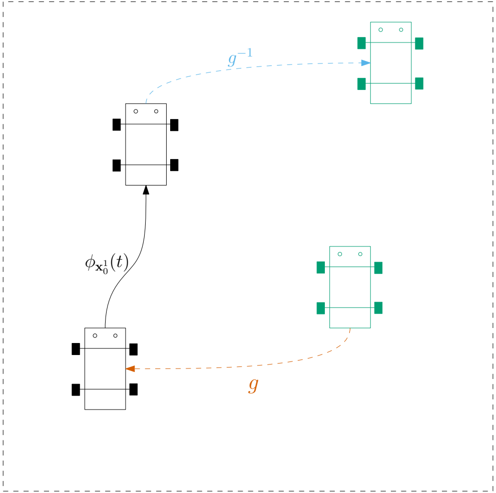


Principle



Principle



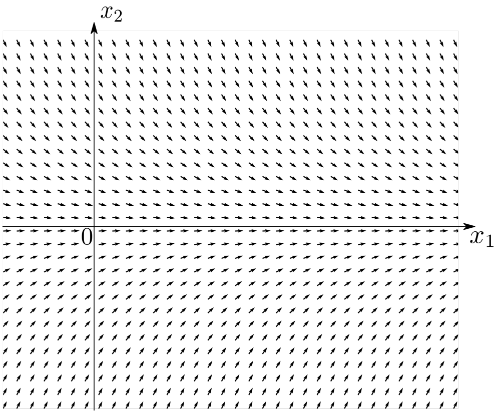


Hints from the vector field

Let us consider the system defined by:

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

From the associated vector field we can deduce a translation symmetry along Ox_1 and mirror symmetry over Ox_1 that does not affect the vector field. These symmetries are called **Lie** symmetries.



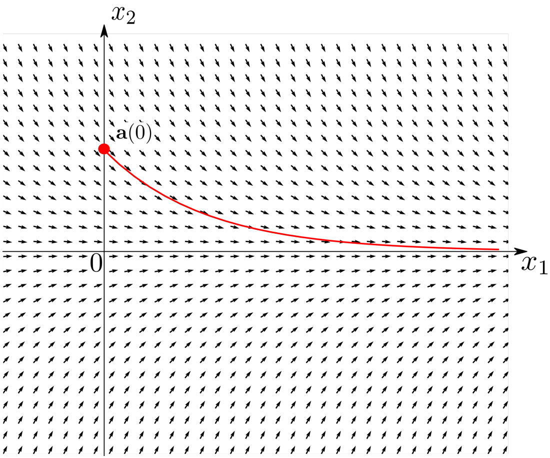
Determine the flow function

Objective: Determine the flow function $\Phi_t(\mathbf{x})$ i.e. the function that describes the evolution of the vector $\mathbf{x}$ at time t.

Determine the flow function

Objective: Determine the flow function $\Phi_t(\mathbf{x})$ i.e. the function that describes the evolution of the vector $\mathbf{x}$ at time t.

We already have a **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red) and a transformation function $\mathbf{g_p}$ which can send any point of the state space to any other following the symmetries of the system.



Determine the flow function

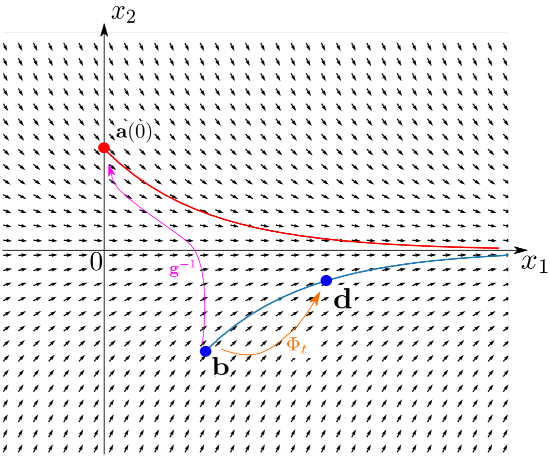
Objective: Determine the flow function $\Phi_t(\mathbf{x})$ i.e. the function that describes the evolution of the vector $\mathbf{x}$ at time t.

We already have a **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red) and a transformation function $\mathbf{g_p}$ which can send any point of the state space to any other following the symmetries of the system.

Therefore

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

where $\mathbf{p} = \mathbf{h}(\mathbf{x}, \mathbf{a}(0))$



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

Determine the flow function

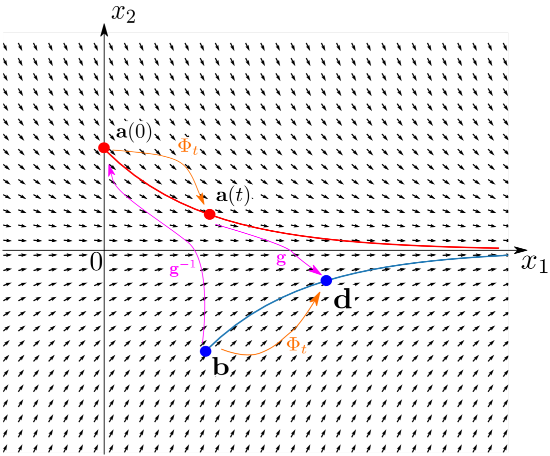
Objective: Determine the flow function $\Phi_t(\mathbf{x})$ i.e. the function that describes the evolution of the vector $\mathbf{x}$ at time t.

We already have a **reference** trajectory denoted $\mathbf{a}(\cdot)$ (painted red) and a transformation function $\mathbf{g_p}$ which can send any point of the state space to any other following the symmetries of the system.

Therefore

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

where $\mathbf{p} = \mathbf{h}(\mathbf{x}, \mathbf{a}(0))$



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

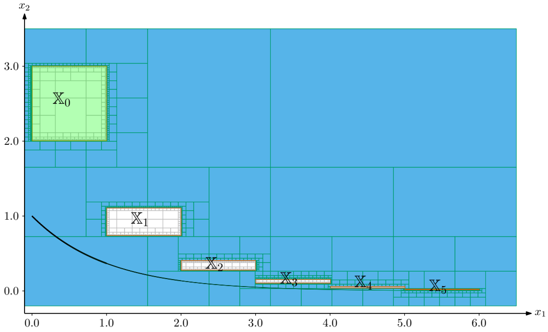
Guaranteed Integration, a set inversion problem

With the flow function Φ_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 Φ_t. We want to find the set $\mathbb{X}_t$ such that

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

Applying the SIVIA

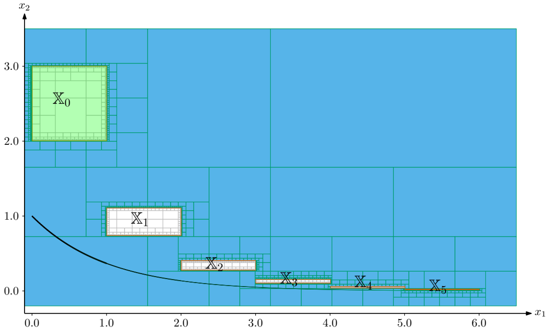
Initial condition: $[\mathbf{x}_0] = [0, 1] \times [2, 3]$



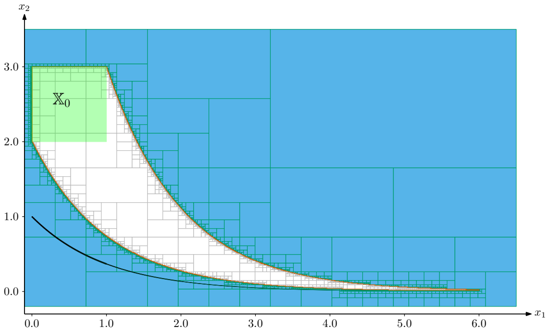
Discrete sets computation

Applying the SIVIA

Initial condition: $[\mathbf{x}_0] = [0,1] \times [2,3]$



Discrete sets computation

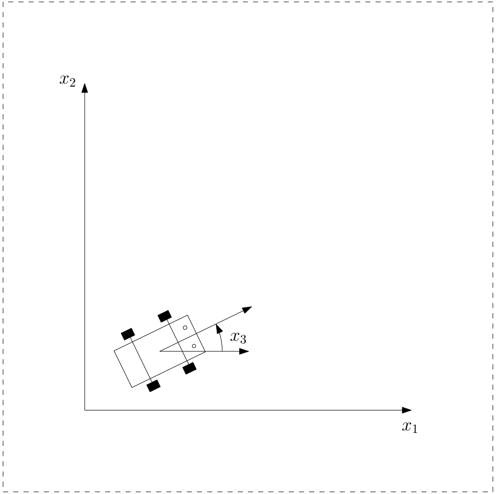


Continous set computation

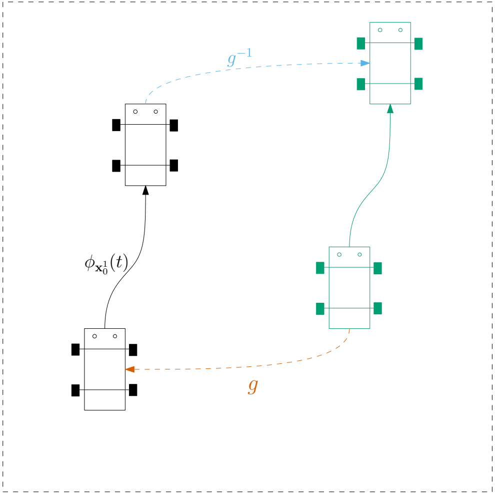
Example of the tank-like robot: Introduction

Let us consider the system defined by:

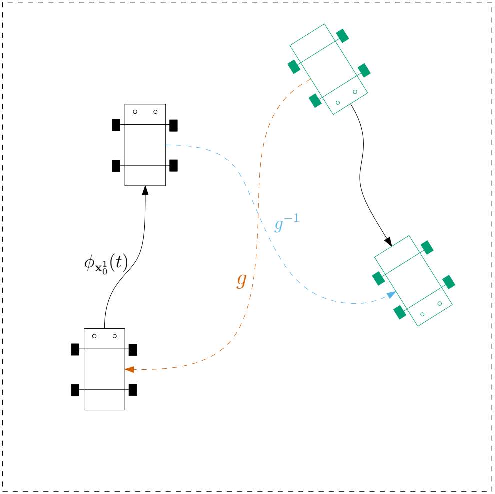
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}(t)) = \begin{pmatrix} u_1(t)\cos(x_3) \\ u_1(t)\sin(x_3) \\ u_2(t) \end{pmatrix}$$



Example of the tank-like robot: Symmetries



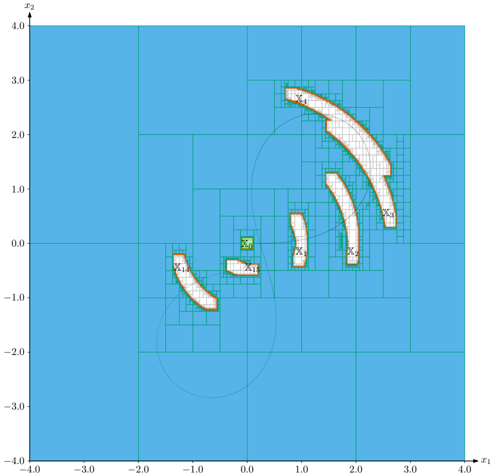
Translation symmetries



Rotation symmetry

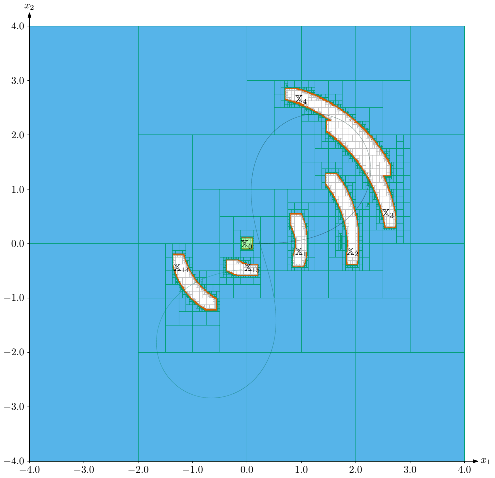
Applying the SIVIA

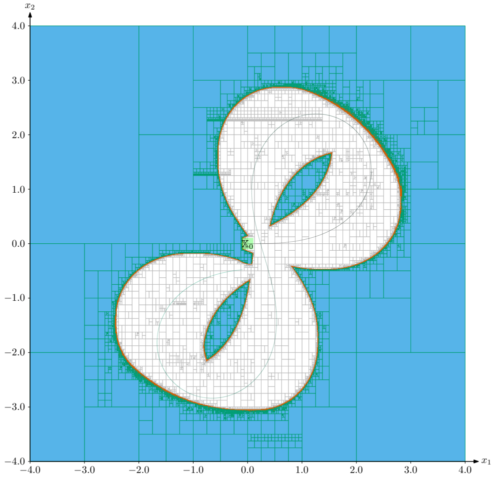
Initial condition: $[\mathbf{x}_0] = [-0.1, 0.1]^2 \times [-0.4, 0.4]$



Applying the SIVIA

Initial condition: $[\mathbf{x}_0] = [-0.1, 0.1]^2 \times [-0.4, 0.4]$





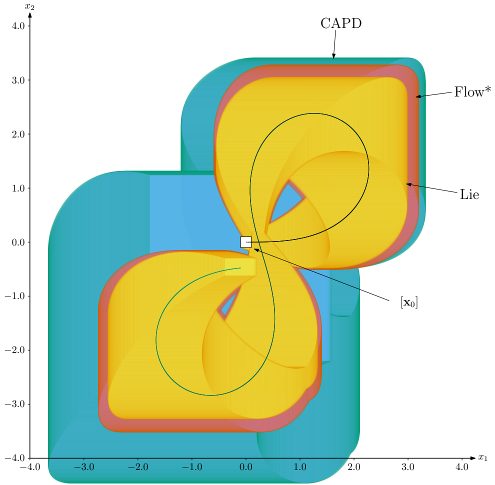
Pros and limits of the method

Pros:

Pros and limits of the method

Pros:

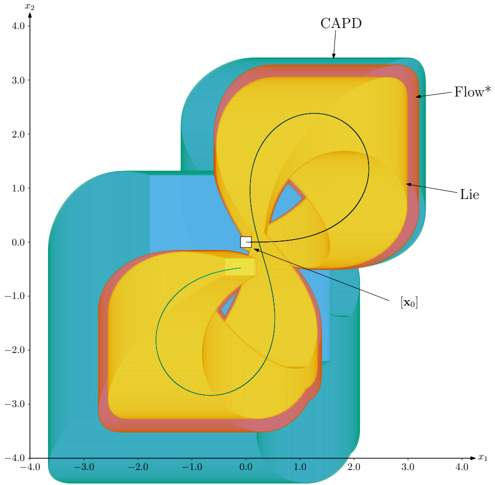
- able to deal with large initial condition (no bloating effect)



Pros and limits of the method

Pros:

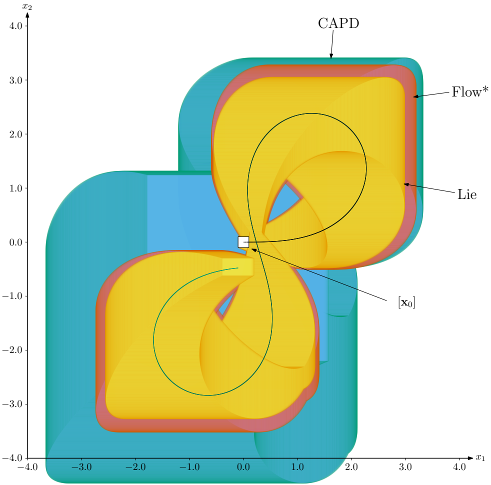
- able to deal with large initial condition (no bloating effect)
- less computation time (less steps of operations)



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

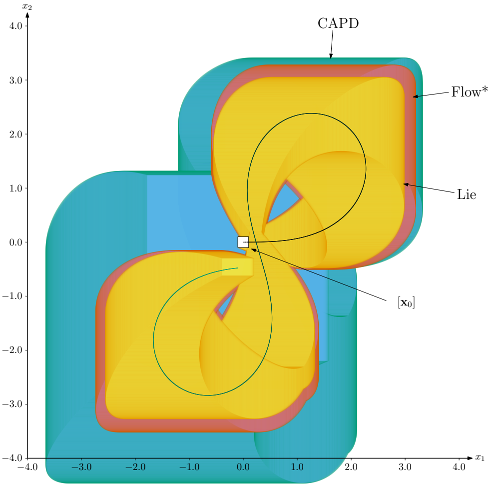


Pros and limits of the method

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:



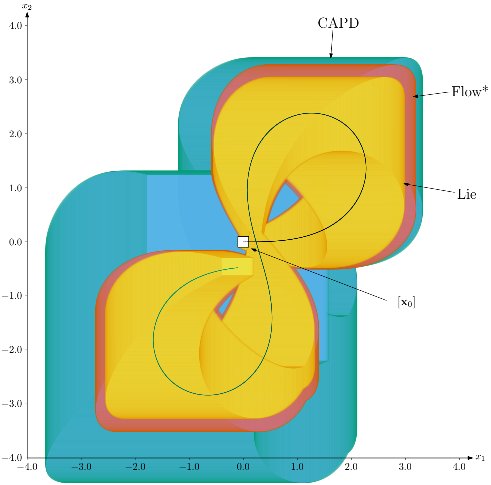
Pros and limits of the method

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



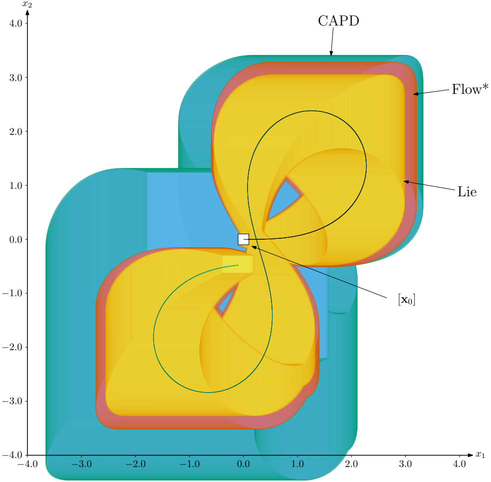
Pros and limits of the method

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



Section 3

Application to reachability

Reachability analysis

One of the applications of guaranteed integration is reachability analysis. It is used for various purposes:

Reachability analysis

One of the applications of guaranteed integration is reachability analysis. It is used for various purposes:

- Performance assessment

Reachability analysis

One of the applications of guaranteed integration is reachability analysis. It is used for various purposes:

- Performance assessment
- Scheduling

Reachability analysis

One of the applications of guaranteed integration is reachability analysis. It is used for various purposes:

- Performance assessment
- Scheduling
- Controller design

Reachability analysis

One of the applications of guaranteed integration is reachability analysis. It is used for various purposes:

- Performance assessment
- Scheduling
- Controller design
- Deadlock

Reachability analysis

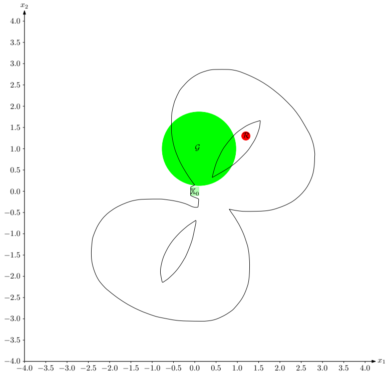
One of the applications of guaranteed integration is reachability analysis. It is used for various purposes:

- Performance assessment
- Scheduling
- Controller design
- Deadlock

The following slides will focus on performance assessment in a mobile robotic context

Reachable area problem

Common robotic problem: Ensure that a vehicle will either certainly reach a mandatory area or avoid (a dangerous) one.



Reachable area problem

Checking that we reach a mandatory area:

$$\forall t \in [t], \begin{cases} 1.\ \Phi_{-t}(\mathbf{x}(t)) \in [\mathbf{x}_0] & \text{(evolution)} \\ 2.\ \mathbf{x}(t) \notin \mathcal{G} & \text{(area to reach)} \end{cases}$$

These two constraints cannot be both satisfied $\Rightarrow$The system will go through the mandatory area

Reachable area problem

Checking that we avoid a forbidden area:

$$\begin{cases} 1.\ \Phi_{-t}(\mathbf{x}(t)) \in [\mathbf{x}_0] & \text{(evolution function)} \\ 2.\ \exists t \in [t], \mathbf{x}(t) \in \mathcal{R} & \text{(area to reach)} \end{cases}$$

These two constraints cannot be both satisfied $\Rightarrow$The system will avoid the forbidden area

Reachable area problem

Section 4

Conclusion

Conclusion

- A new guaranteed integration method based on Lie symmetries
- adapted to large initial conditions
- Not suitable for every kind of dynamical systems but suitable for robotics

Thank you for your attention

Proof $\mathbb{X}_t = \Phi_{-t}^{-1}([\mathbf{x}_0])$:

$$\begin{aligned}
\mathbf{x} \in \mathbb{X}_t \quad & \Longleftrightarrow \quad \exists \mathbf{x}_0 \in [\mathbf{x}_0], \mathbf{x} = \phi_t(\mathbf{x}_0) \\
& \Longleftrightarrow \quad \exists \mathbf{x}_0 \in [\mathbf{x}_0], \mathbf{x}_0 = \phi_{-t}(\mathbf{x}) \\
& \Longleftrightarrow \quad \phi_{-t}(\mathbf{x}) \in [\mathbf{x}_0] \\
& \Longleftrightarrow \quad \mathbf{x} \in \phi_{-t}^{-1}([\mathbf{x}_0])
\end{aligned}$$

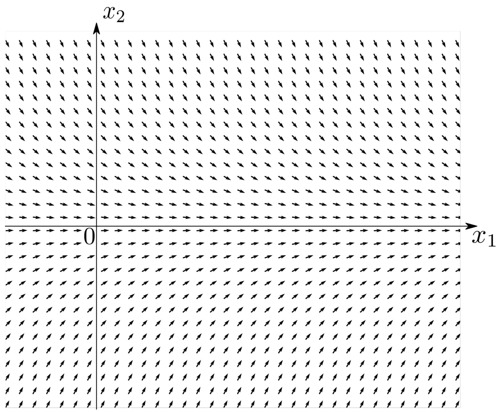
Defintion (Lie symmetry) : A function $\mathbf{g_p}$ is a symmetry of $\mathbf{f}$ if the action $\bullet$ of $\mathbf{g_p}$ on $\mathbf{f}$ leaves $\mathbf{f}$ unchanged i.e

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

It is also called a **stabiliser**.

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

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

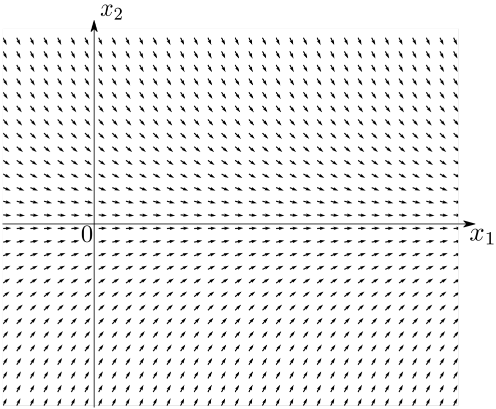


Proposition (Action of a diffeomorphism):

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

Objective: Finding $\mathbf{g}$ such that

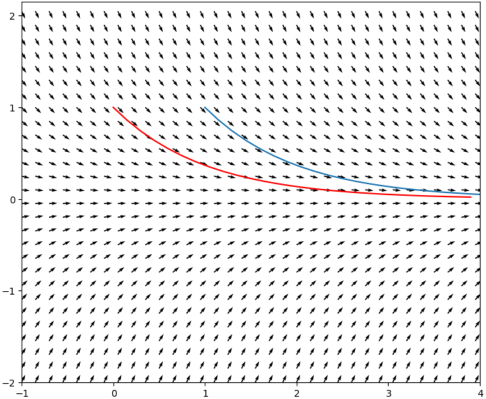
$$\mathbf{g} \bullet \mathbf{f} = \left(\frac{d\mathbf{g}}{d\mathbf{x}} \circ \mathbf{g}^{-1} \right) \cdot (\mathbf{f} \circ \mathbf{g}^{-1}) = \mathbf{f}$$



Translation symmetry:

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

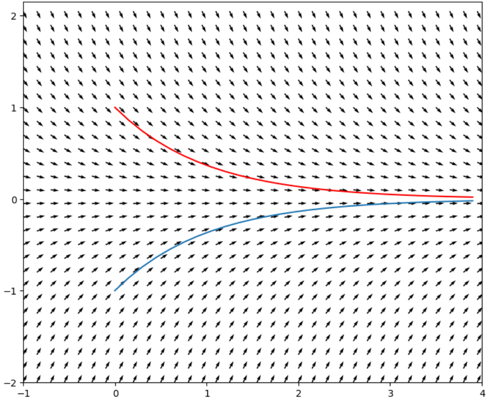
$$\begin{aligned} \mathbf{g}_\alpha \bullet \mathbf{f}(\mathbf{x}) &= \left(\frac{d\mathbf{g}_\alpha}{d\mathbf{x}} \circ \mathbf{g}_\alpha^{-1} \right) \cdot \left(\mathbf{f} \circ \mathbf{g}_\alpha^{-1} \right)(\mathbf{x}) \\ &= \left(\frac{d\mathbf{g}_\alpha}{d\mathbf{x}} \cdot \mathbf{f} \right) \circ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



Mirror symmetry:

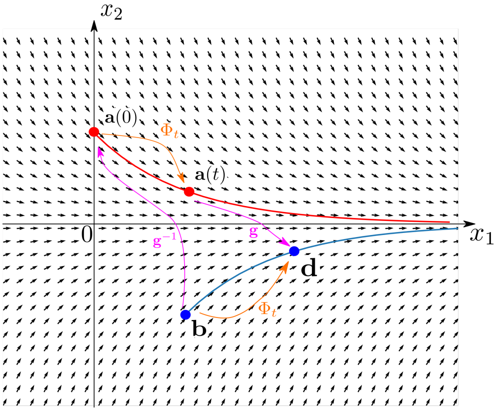
$$\mathbf{g}_\beta : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 \\ \beta x_2 \end{pmatrix}$$

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



Complete symmetry:

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



To find the transport function, we must solve

$$\mathbf{g_p}(\mathbf{a}) = \mathbf{x},$$

in order to express **p** using only **a** and **x**.
Using the previous example :

$$\mathbf{g_p}(\mathbf{a}) = \mathbf{x} \iff \begin{pmatrix} a_1 + p_1 \\ p_2 a_2 \end{pmatrix} = \mathbf{x}$$
$$\iff \mathbf{p} = \begin{pmatrix} x_1 - a_1 \\ \frac{x_2}{a_2} \end{pmatrix}$$

Therefore,

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

Flow function of the first example:

$$\mathbf{a}(t) = \begin{pmatrix} t \\ e^{-t} \end{pmatrix} \text{ and } \mathbf{a}(0) = (0, 1)$$

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

Flow function for the tank-like robot:

$$\mathbf{a}(0) = (0, 0, 0) \text{ and a(t) obtained with CAPD}$$

$$\Phi_{-t}(\mathbf{x}) = \begin{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \mathbf{R}_{x_3 - a_3(t)} \cdot \begin{pmatrix} -a_1(t) \\ -a_2(t) \end{pmatrix} \\ x_3 + a_3(t) \end{pmatrix}$$