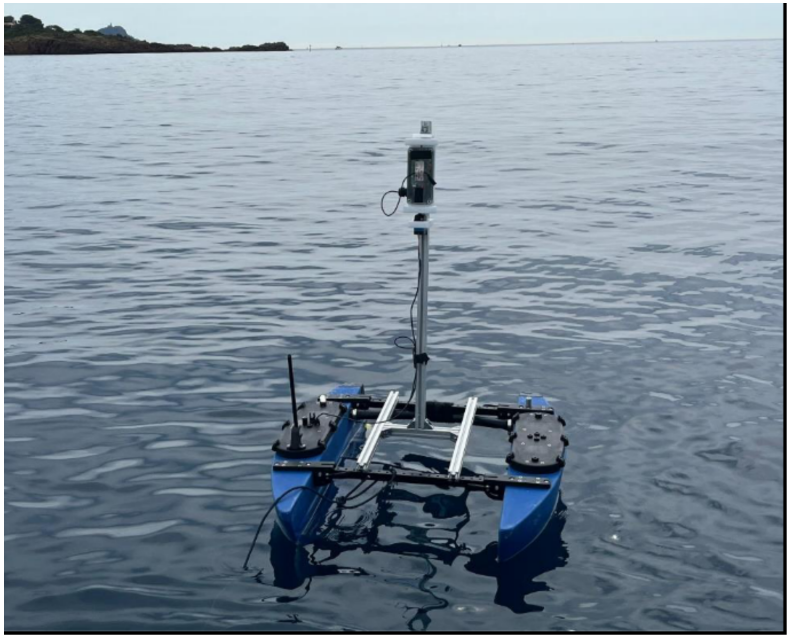
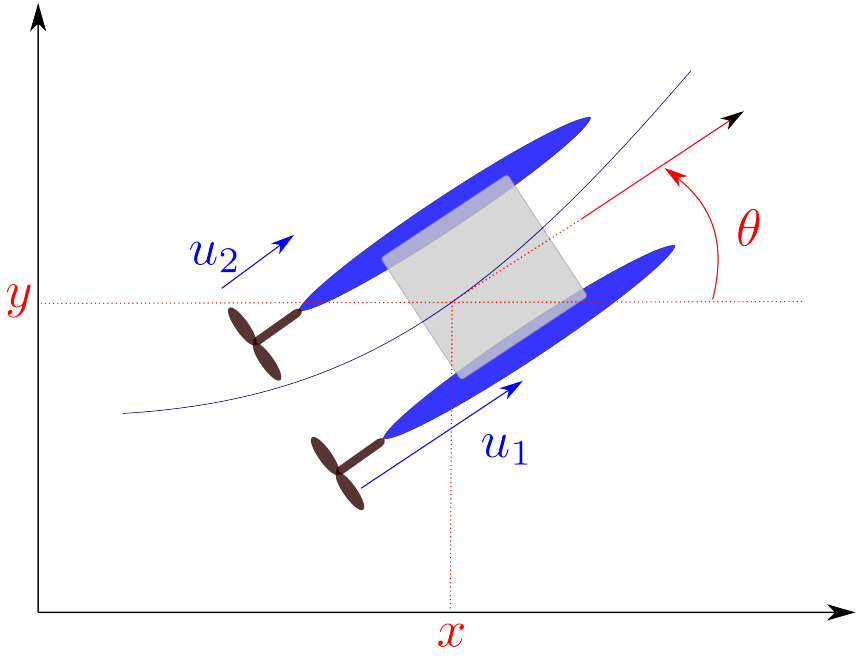
Blue Carousel Challenge : Learn a blue boat to rotate around a buoy

https://webperso.ensta.fr/jaulin/carousel.html

29-30 Juin 2026
Séminaire IA&Océan, Moulin Mer

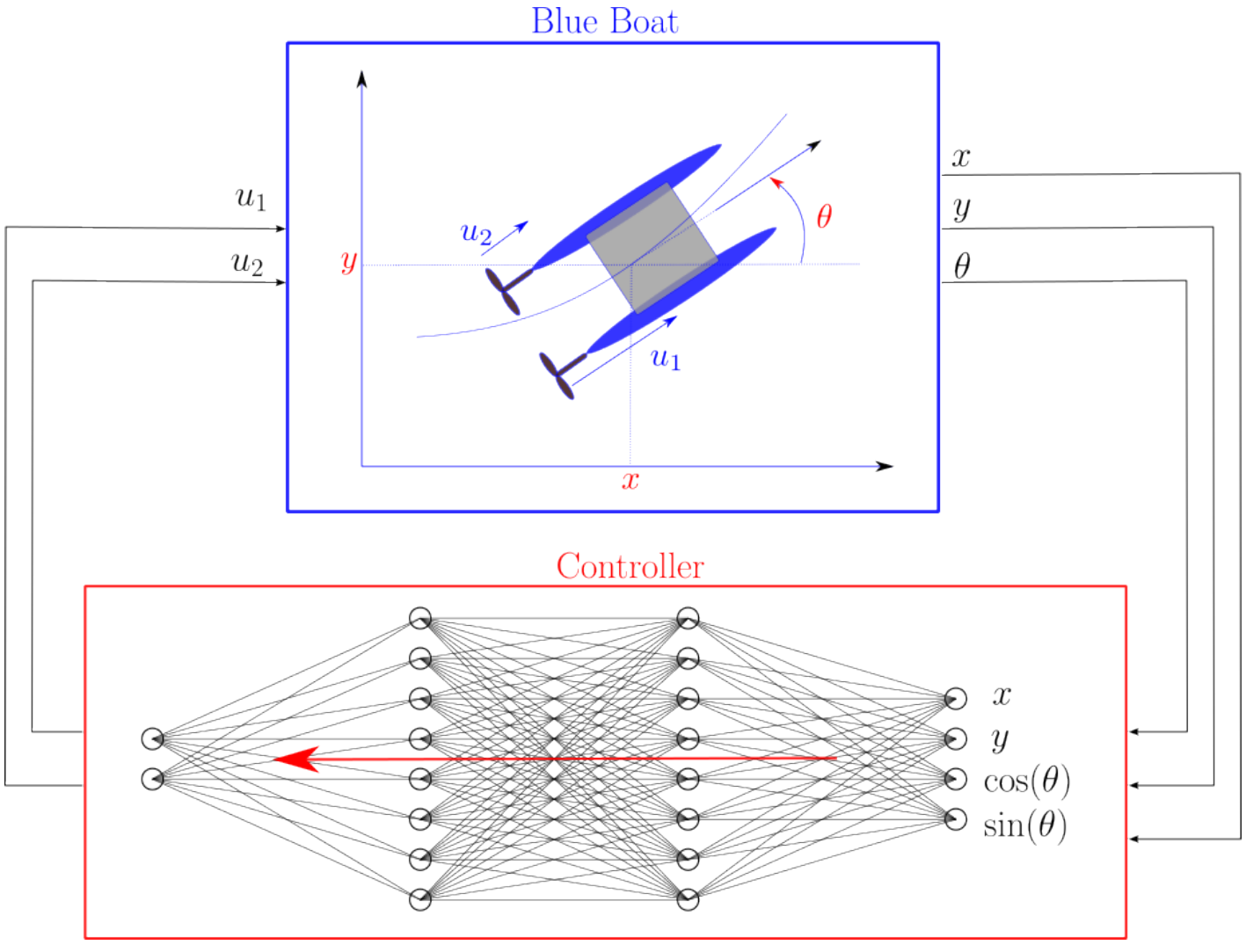
1. Principle

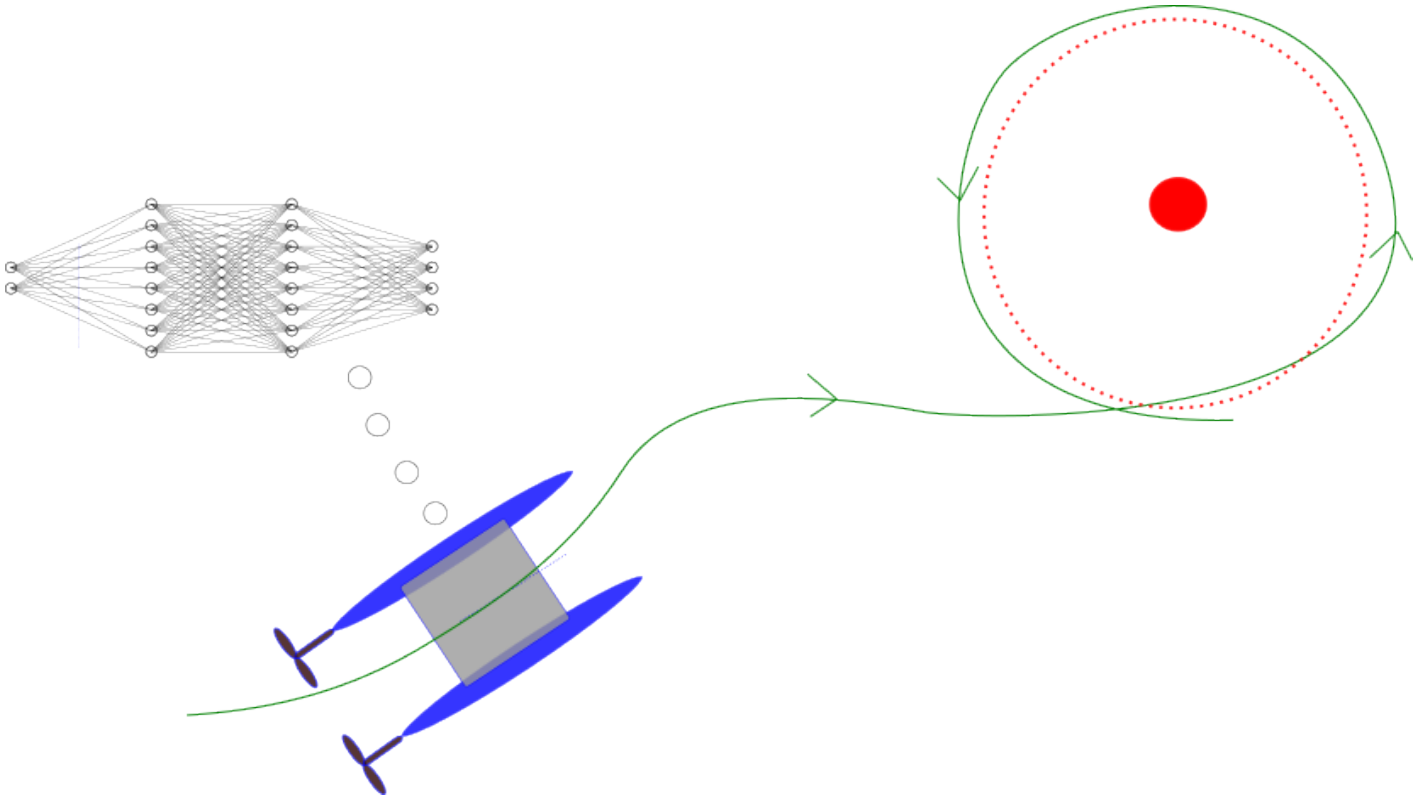



u_2
θ
y
u_1
x


Rte du Centre Nautique
Rte du Centre Nautique
Rte du Centre Nautique
Parking Centre
Moulin Mer
Parking
Centre Nautique
de Moulin Mer
Port de Moulin Mer

- The goal of the *Blue Carousel Challenge* is to use reinforcement learning to train a controller for the blue boat to rotate around a buoy, autonomously.
- The buoy is located at the origin (0,0).
- The real robot is equipped with: Two propellers (u_1: left and u_2 right)
- We measure the position (x, y) and heading θ


Blue Boat
u_1
u_2
y
u_2
θ
u_1
x
x
y
θ
Controller
x
y
$\cos(\theta)$
$\sin(\theta)$



The blue boat turns around the buoy forever

Controller

- The controller is a neural network.
- The input layer has 4 nodes: $x, y, \cos\theta, \sin\theta$
- It has 2 hidden layers with 8 neurons each,
- The output layer has 2 nodes: u_1, u_2
- The neural network contains exactly $\underbrace{(4*8+8*8+8*2)}_{=a_{ij}} + \underbrace{(8+8+2)}_{b_j} = 130$ parameters.
- Activation rule: $Relu\left(\sum_i w_{ij} a_i + b_j\right)$

2. Simulator

Sim-to-Real (Simulation to Reality)

- A poor simulator is used to training the control policy.
- A transfer of the learned behavior will be done onto the unity model to improve the controller.
- The controller is put inside the real blue boat in the real world.

.

$$
\begin{aligned}
\dot{x} &= v_x \cos\theta - v_y \sin\theta \\
\dot{y} &= v_x \sin\theta + v_y \cos\theta \\
\dot{\theta} &= \omega \\
\dot{v}_x &= \omega v_y - p_5 v_x |v_x| + p_3 (\omega_1 |\omega_1| + \omega_2 |\omega_2|) \\
\dot{v}_y &= -\omega v_x - p_6 v_y |v_y| \\
\dot{\omega} &= -p_7 \omega |\omega| + p_4 (\omega_2 |\omega_2| - \omega_1 |\omega_1|) \\
\dot{\omega}_1 &= -p_1 \omega_1 |\omega_1| + p_2 u_1 \\
\dot{\omega}_2 &= -p_1 \omega_2 |\omega_2| + p_2 u_2
\end{aligned}
$$

Reinforcement Learning. The controller learns by trial and error, interacting with an environment to maximize some cumulative reward.

- The environment: The simulator (or the real world).
- The state: x, y, θ.
- The action: u_1, u_2.
- The reward: the score given to the controller based on the taken action.

3. Challenge

Teams

- 5 teams should be made
- Each team contains a robiticist, a connectionist, a Python expert
- Each team contains a Robex member, a OSE member, a M3 member.

Submission Format

- Teams must submit their controller as a CSV file containing the 130 parameters
- The robot will use these parameters directly onboard

What you should do

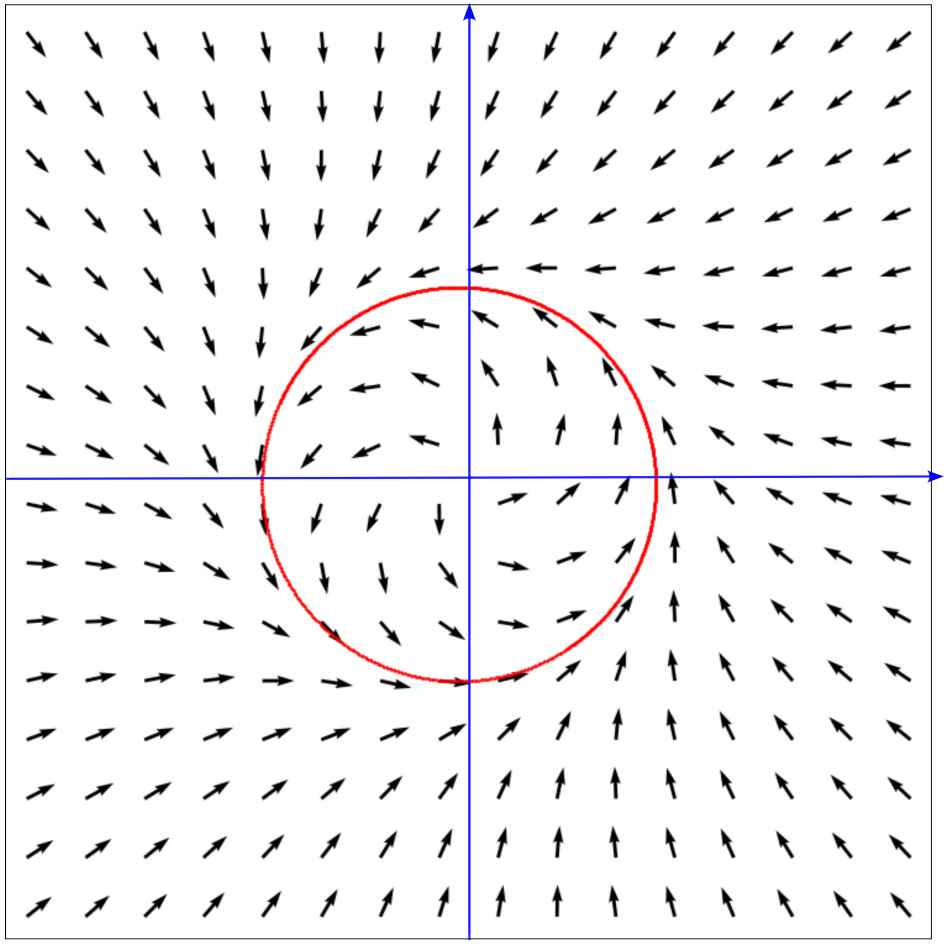
- Use the python simulator to train your controller (see Matheus)
- Validate your controller on the Unity simulator (see Toméo)
- Validate your controller on the real blue-boat (see Quentin)
- Bonus point : the one wheel journey (see Taddeo)

- All videos, photos, data will be shared and made public on the Web.
- They can be used by anybody for publications, communication, etc.

4. Imitation Learning approach

Poincaré oscillator:

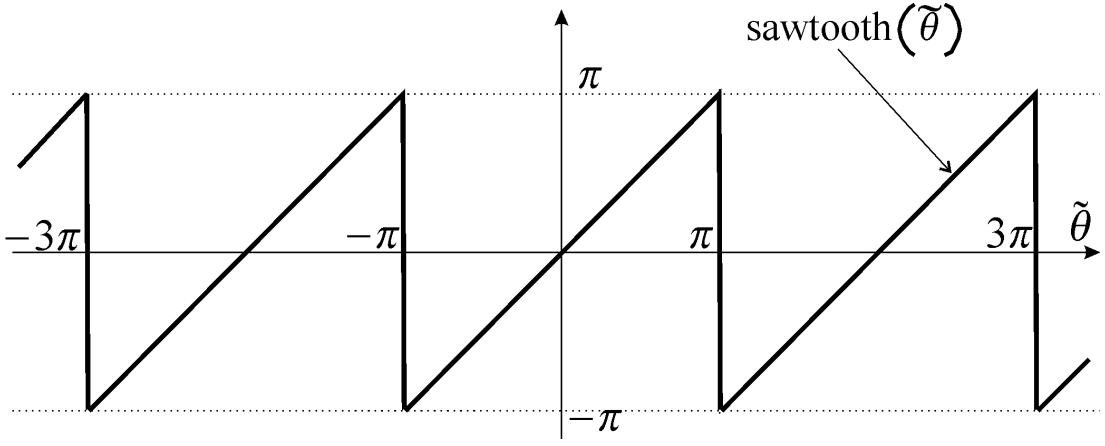
$$\dot{\mathbf{x}} = \mathbf{P}^0(\mathbf{x}) = \begin{pmatrix} -x_1^3 - x_1 x_2^2 + x_1 - x_2 \\ -x_2^3 - x_1^2 x_2 + x_1 + x_2 \end{pmatrix}$$



Zoom by 10 of the field, to have a distance of 10m :

$$P^{10}(\mathbf{x}) = P^{0}(\frac{\mathbf{x}}{10})$$

Sawtooth function



$$\text{sawtooth}(\widetilde{\theta}) = \text{mod}(\widetilde{\theta} + \pi, 2\pi) - \pi$$

Take

$$\tilde{\theta}(\mathbf{x}, \theta) = \text{angle}\left(\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, P^{10}(\mathbf{x})\right)$$

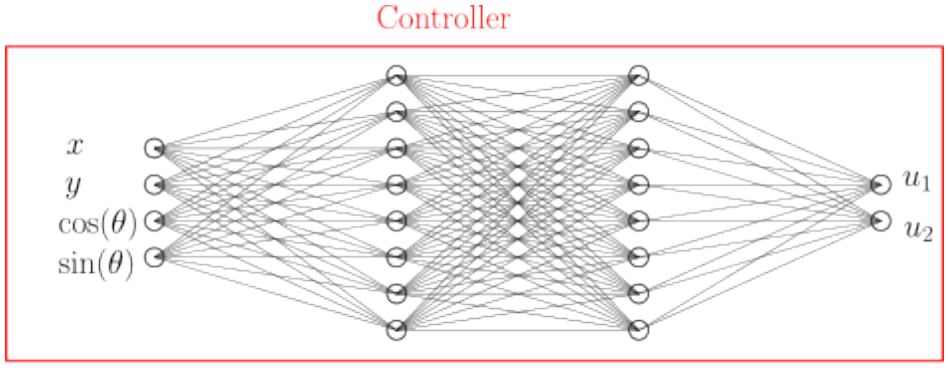
with

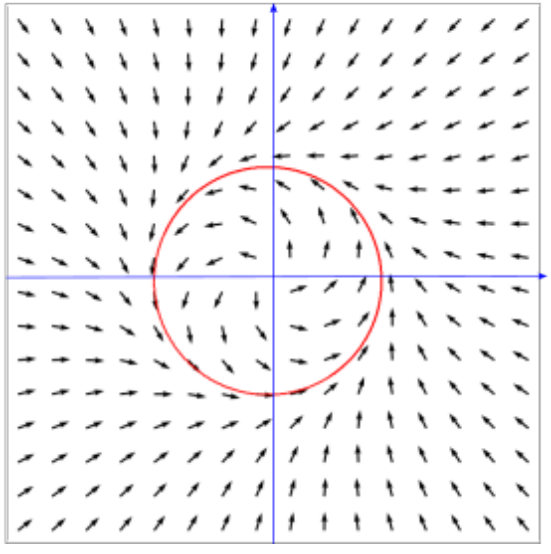
$$\text{angle}(\mathbf{u}, \mathbf{v}) = \text{atan2}(u_1 v_2 - u_2 v_1, u_1 v_1 + u_2 v_2)$$

We take

$$\begin{aligned} u_1 + u_2 &= 1.5 \\ u_2 - u_1 &= \tilde{\theta}(\mathbf{x}, \theta) \end{aligned}$$

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 0.7 \\ \frac{1}{\pi} \cdot \text{sawtooth}\left(\tilde{\theta}(\mathbf{x}, \theta)\right) \end{pmatrix}$$





Imitation learning: imitate the desired vector field