Recent Advances in Safety, Optimization, and Control

Ricardo Sanfelice
Department Electrical and Computer Engineering
University of California

CoE Review @ UT Austin - October 12, 2022

Outline of Recent Results

1. **Safety**
   - Safety Certificates
     ACC 22a, TAC 22, ESAIM 22,
     Letters/ACC23a (submitted), TAC (submitted) + CoE collab
   - Applications of Safety
     ACC 22b, CCTA 22a, and CCTA 22b
     security via barrier functions ACC23b (submitted)
2. **Optimization**
   - Dynamical systems approach
     ACC2023a (submitted), Optimization journal (almost ready)
     Automatica (submitted), ACC23c (submitted) w/ Matt Hale
   - Optimization with Computational Constraints
     CPSWeek-IoT 22 Workshop w/ Matt Hale and AFRL/RV
3. **Feedback Control**
   - Hybrid Control and Hybrid MPC
     ACC 22c (best student paper finalist), ACC 22d, CDC 22bc, ACC23c

On the Feasibility and Continuity of Feedback Controllers Defined by Multiple Control Barrier Functions

Axton Isaly, Masoumeh Ghanbarpour, Ricardo G. Sanfelice, Warren E. Dixon

***Abstract*—Control barrier functions are a popular method for encoding safety specifications for dynamical systems. In this paper, a notion of control barrier function is defined that permits vector-valued barrier functions and flow constraints involving both the state and the control input. Control barrier functions induce constraints on the control input that, when satisfied, guarantee the forward invariance of a safe set of states. The constraints are enforced using a pointwise-optimal feedback controller, and sufficient conditions for the continuity of the controller are given. The existence of a control barrier function is defined to be equivalent to the feasibility of the optimal feedback controller. Polynomial optimization problems based on sums of squares are formulated that can be used to certify that a given function is a control barrier function. Examples of the control barrier function design procedure are presented illustrating the process of formulation, verification, and synthesis.**

I. Introduction

The use of control barrier functions (CBF) to synthesize feedback controllers that render sets of states forward invariant has recently gained significant interest because of the tight relationship between forward invariance and safety [1]–[4]. The CBF literature has focused on complex safety specifications defined by multiple, sometimes conflicting requirements such as obstacle avoidance, shifting goal locations, dynamic constraints, and control input limitations. In practice, these control objectives are often described using multiple CBFs (e.g., [3] and [1, Sec. V]), whereas the majority of theoretical

CBF candidates is a promising way to obtain control laws that are continuous functions of the state.

A. Feasibility

Traditionally, a CBF is defined to guarantee that a safety-ensuring controller exists, meaning that all objectives in the safety specification can be met simultaneously. However, tools for verifying that a given function is a CBF are not fully developed. While analytical conditions exist to determine whether a scalar-valued function is a CBF (cf. [8], [9, Prop. 1]), the problem is significantly more challenging in the presence of multiple CBFs. In general, a CBF candidate defines a set of constraints in the decision variable (control input) that vary with an external parameter (the state of the dynamical system), and it must be verified that control inputs satisfying the constraints exist for all states in a given set. This verification should be done during the design phase, so that controllers are certified as feasible before deployment. The authors of [10] leverage a tool for checking that multiple constraints have at least one feasible solution at a particular point in the state space, but it it not clear how to verify this property on a given (uncountable) set of states. One method to ensure feasibility is by adding slack variables or similar relaxations to the optimization problem at the cost of losing safety guarantees [2]. The authors in [11] use slack variables to ensure feasibility only on the interior of the safe set, while still enforcing

Basic Setting

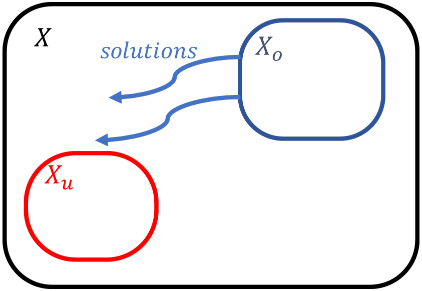
Consider the system

$$\dot{x} = f(x) \qquad x \in X \subset \mathbb{R}^n$$

and the sets

$X_o \subset X$ the initial set,

$X_u \subset X \backslash X_o$ the unsafe set.



$$\boxed{\text{Safety with respect to } (X_o, X_u) \quad \Leftrightarrow \quad \mathrm{reach}(X_o) \cap X_u = \emptyset}$$

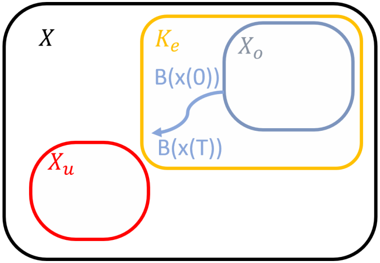
$\mathrm{reach}(X_o) := \{x \in \mathbb{R}^n : x = \phi(t; x_o)$, with ϕ a solution from $x_o \in X_o$ and $t \in \mathrm{dom}\,\phi\}$ $\leftarrow$ the infinite reach set

A solution to $\dot{x} = f(x)$ is denoted $t \mapsto \phi(t)$, and when starts at x_o as $t \mapsto \phi(t; x_o)$

Sufficient Conditions for Safety when $X = \mathbb{R}^n$

Consider $X = \mathbb{R}^n$ and let the function B satisfy

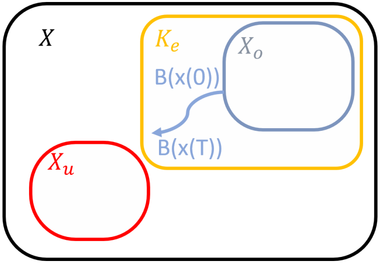
$$B(x) > 0 \qquad \forall x \in X_u$$
$$B(x) \leq 0 \qquad \forall x \in X_o$$



Sufficient Conditions for Safety when $X = \mathbb{R}^n$

Consider $X = \mathbb{R}^n$ and let the function B satisfy

$$B(x) > 0 \qquad \forall x \in X_u$$
$$B(x) \le 0 \qquad \forall x \in X_o$$



and

the set K_e is "forward invariant" for $\dot{x} = f(x)$

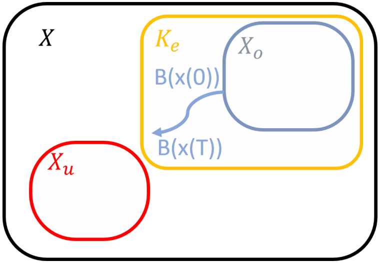
where

$K_e := \{x \in \mathbb{R}^n : B(x) \le 0\} \qquad \leftarrow \qquad$ the zero-sublevel set of B

Sufficient Conditions for Safety when $X = \mathbb{R}^n$

Consider $X = \mathbb{R}^n$ and let the function B satisfy

$$B(x) > 0 \qquad \forall x \in X_u$$
$$B(x) \leq 0 \qquad \forall x \in X_o$$



and

the set K_e is "forward invariant" for $\dot{x} = f(x)$

where

$K_e := \{x \in \mathbb{R}^n : B(x) \leq 0\}$ $\leftarrow$ the zero-sublevel set of B

It follows that the system $\dot{x} = f(x)$ is safe w.r.t. (X_o, X_u)

Motivation

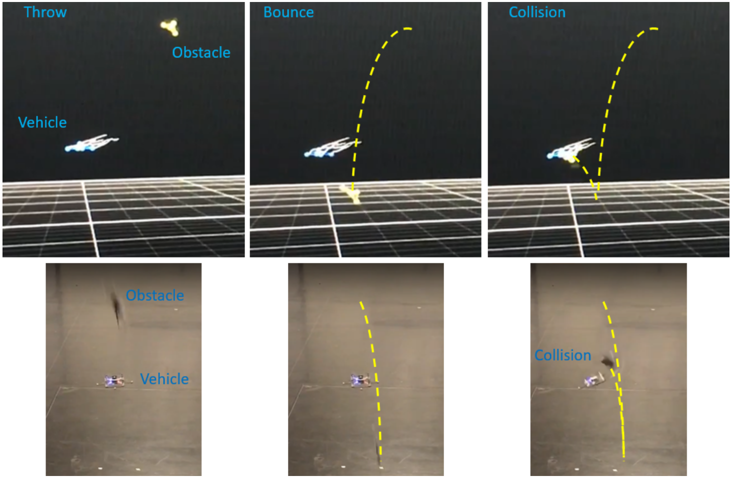


Figure: Motion planning for quadrotors to avoid obstacles without bounce prediction.

Motivation

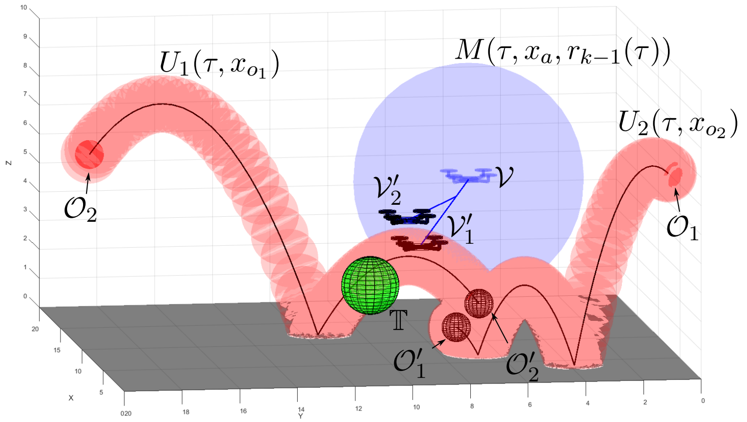


Figure: Motion planning for quadrotors to avoid obstacles with bounce prediction.

Obstacle Model

A bouncing ball obstacle with state

$$x := (p, v) \in \mathbb{R}^6$$

can be modeled as

$$\begin{aligned} \dot{x} &= (v, (0, 0, -\gamma)) \\ x^+ &= (p_x, p_y, 0, v_x, v_y, -\lambda v_z) \end{aligned}$$

where p is the position, v is the velocity, γ is the acceleration due to gravity, and λ is the restitution coefficient.

The Motion Planning Algorithm

Algorithm 0: Motion planning algorithm with input $(\mathbb{T}, \tau_p, \tau_e, \xi_a, \xi_o, \mathcal{H}_a, \mathcal{H}_{o_i})$

```
1:  $\kappa_0 \leftarrow 0$, $r_0 \leftarrow 0$, $x_a \leftarrow \xi_a$, $x_o \leftarrow \xi_o$
2:  for $k = 1, 2, \dots$ do
3:      $M \leftarrow \emptyset$, $U \leftarrow \emptyset$, $M_S \leftarrow \emptyset$
4:      for all $\hat{r} \in \Omega_p, \hat{r}(0) = r_{k-1}(\tau_e)$ do
5:          The solution $\phi_a$ of $\mathcal{H}_a$ is simulated from $x_a$ for reference $\hat{r}$ for $[0, \tau_p]$ seconds of flow
6:          $M \leftarrow M \cup \{(\hat{r}, \phi_a)\}$
7:      end for
8:      for $i \in \{1, 2, \dots, N\}$ do
9:          The solution $\phi_{o_i}$ of $\mathcal{H}_{o_i}$ is simulated from $x_{o_i}$ for $[0, \tau_p]$ seconds of flow
10:         $U \leftarrow U \cup \{\phi_{o_i}(t, j) : \forall (t, j) \in \mathrm{dom}\ \phi_{o_i} \cup ([0, \tau_p] \times \mathbb{N})\}$
11:     end for
12:     for all $(\hat{r}, \phi_a) \in M$ do
13:         for all $\phi_o \in U$ do
14:             if $\mathrm{dist}(p_{\phi_a}(t, j_a), p_{\phi_o}(t, j_o) \geq k_u,$ for all $(t, j_a) \in \mathrm{dom}\ \phi_a$ and all $(t, j_o) \in \mathrm{dom}\ \phi_o$ then
15:                 $M_S \leftarrow M_S \cup \{(\hat{r}, \phi_a)\}$
16:             end if
17:         end for
18:     end for
19:     $(r_k, \phi)$ is the lowest cost reference, solution pair in $M_S$
20:     $\kappa_k \leftarrow \kappa(r_k, \phi, r_{k-1}, \kappa_{k-1})$
21:     Execute $r_k$ for $\tau_e$ seconds.
22:     Update $x_a$ and $x_o$
23: end for
```

Simulation & Experiments: System Architecture

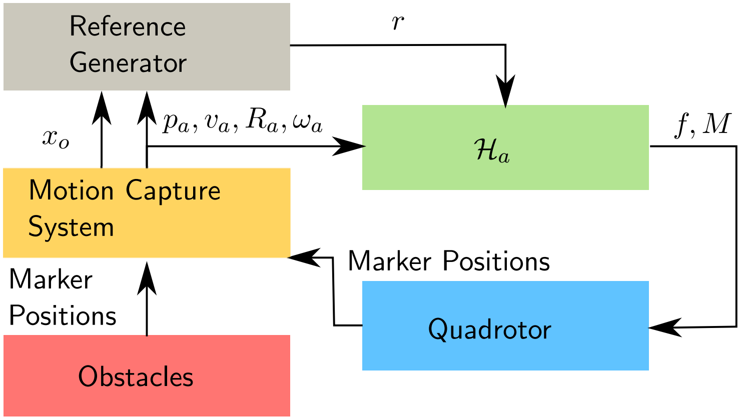


Figure: Diagram of the hybrid feedback loop.

Simulation & Experiments: Simulation Results

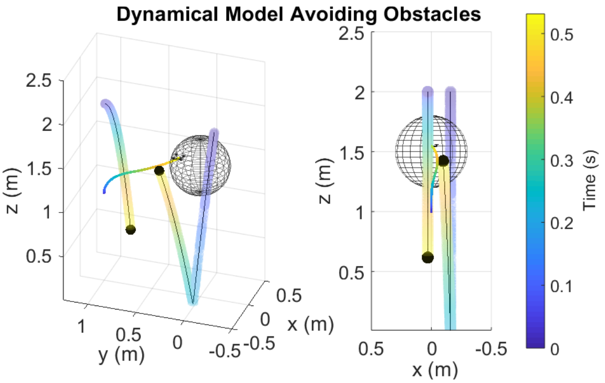
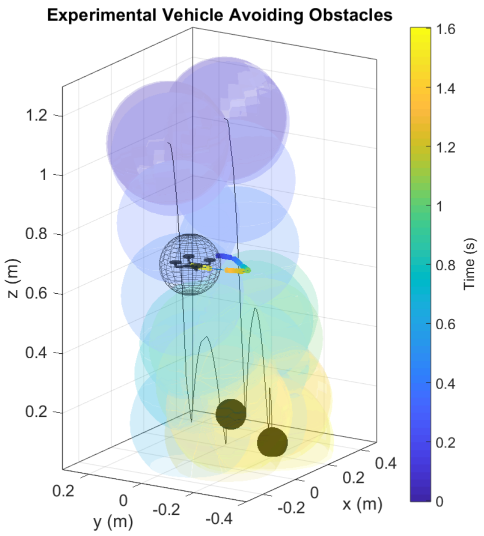


Figure: Simulation avoiding two obstacles using a dynamical quadrotor model. The target set is the sphere with radius 0.3m centered at $(0,0,2)$. Initial quadrotor state is $(0,1,2,0,-0.5,0,I_{3\times3},0,0,0)$ with initial obstacle states $(-0.16,-0.2,2,0,1,-8)$ and $(0.03,1,2,0,-0.5,0)$. Planning period was 0.3 seconds, execute period 0.05s, and obstacle radius of 0.05m.

Simulation & Experiments: Experiment Set-ups

- Windows 10 computer with a 3.20GHz dual core processor with 8GB of memory
- Eight motion capture cameras running at 120Hz
- Crazyflie 2.0 controlled over 2.4GHz radio though the Crazyflie client
- Wiffle ball with 0.08m diameter wrapped in retro-reflective tape with $\lambda = 0.65$ and $\Sigma = [-0.02, 0.02]$m/s.
- The motion planner and controller are implemented in Matlab.
- The motion planner using $\tau_p = 0.3$s, $\tau_e = 0.28$s, and $k_u = 0.2$m.

Simulation & Experiments: Experiment Results



Simulation & Experiments: Experiment Results

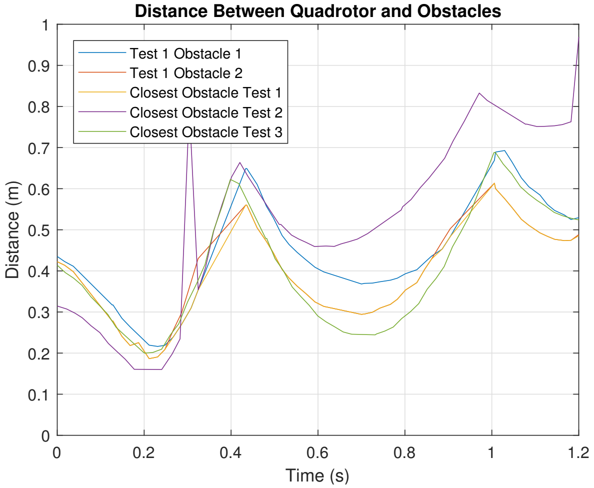


Figure: Plot of distance between quadrotor and the closest of two obstacles over time for three experiment runs.

Outline of Recent Results

1. **Safety**
   - Safety Certificates
     ACC 22a, TAC 22, ESAIM 22,
     Letters/ACC23a (submitted), TAC (submitted) + CoE collab
   - Applications of Safety
     ACC 22b, CCTA 22a, and CCTA 22b
     security via barrier functions ACC23b (submitted)
2. **Optimization**
   - Dynamical systems approach
     ACC2023a (submitted), Optimization journal (almost ready)
     Automatica (submitted), ACC23c (submitted) w/ Matt Hale
   - Optimization with Computational Constraints
     CPSWeek-IoT 22 Workshop w/ Matt Hale and AFRL/RV
3. **Feedback Control**
   - Hybrid Control and Hybrid MPC
     ACC 22c (best student paper finalist), ACC 22d, CDC 22bc, ACC23c

Outline of Recent Results

1. **Safety**
   - Safety Certificates

     ACC 22a, TAC 22, ESAIM 22,

     Letters/ACC23a (submitted), TAC (submitted) + CoE collab
   - Applications of Safety

     ACC 22b, CCTA 22a, and CCTA 22b

     security via barrier functions ACC23b (submitted)
2. **Optimization**
   - Dynamical systems approach

     ACC2023a (submitted), Optimization journal (almost ready)

     Automatica (submitted), ACC23c (submitted) w/ Matt Hale
   - Optimization with Computational Constraints

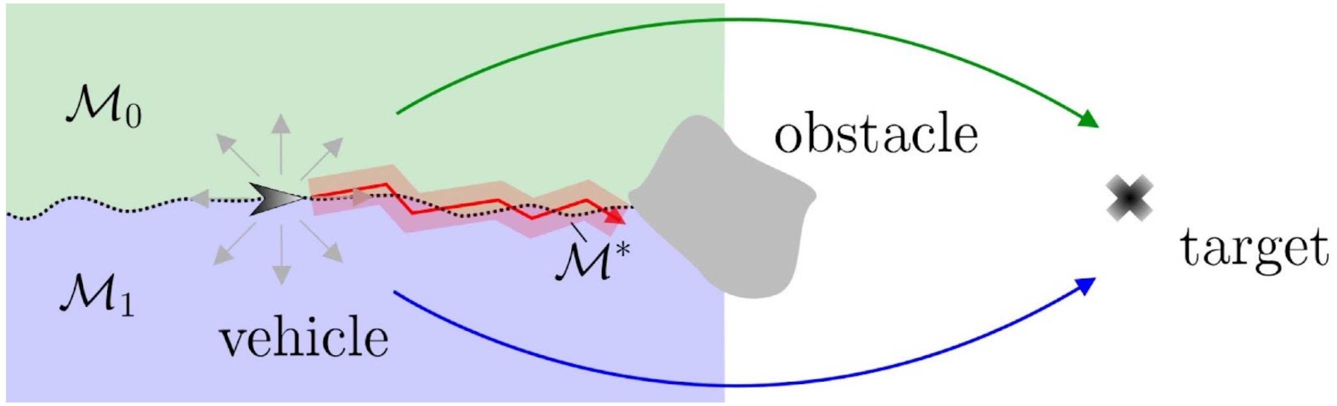
     CPSWeek-IoT 22 Workshop w/ Matt Hale and AFRL/RV
3. **Feedback Control**

PhD students Dawn Hustig-Schultz and Katherine Hendrickson graduated

Visited S. Phillips at AFRL/RV, the P. Ganash at REEF, and Z. Bell at AFRL/RW

Released new version of Hybrid Equations Toolbox (v3.0) for Matlab

Hysteresis-Based RL: Robustifying Reinforcement Learning-based Control Policies via Hybrid Control



Jan de Priester[1], Ricardo G. Sanfelice[1], Nathan van de Wouw[2]

[1]University of California, Santa Cruz, USA

[2]Eindhoven University of Technology, The Netherlands

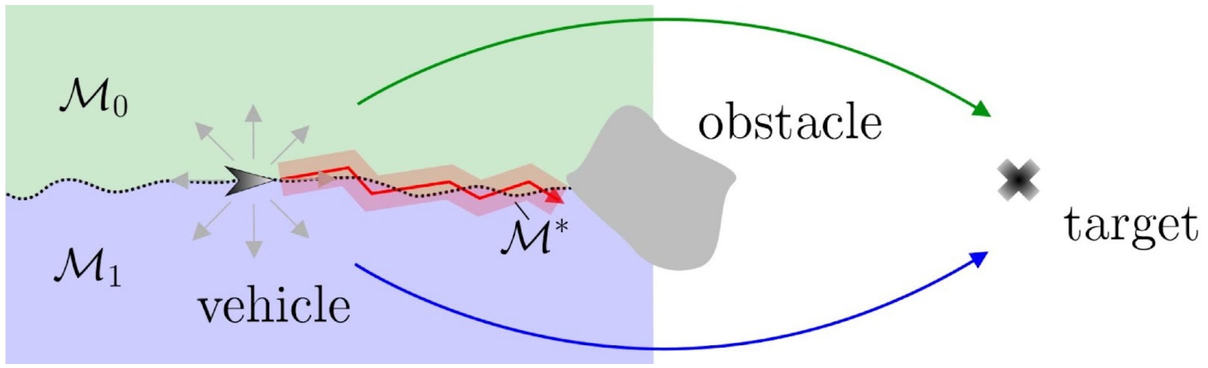
Jan de Priester[1], Ricardo G. Sanfelice[1], Nathan van de Wouw[2]

[1]University of California, Santa Cruz, USA

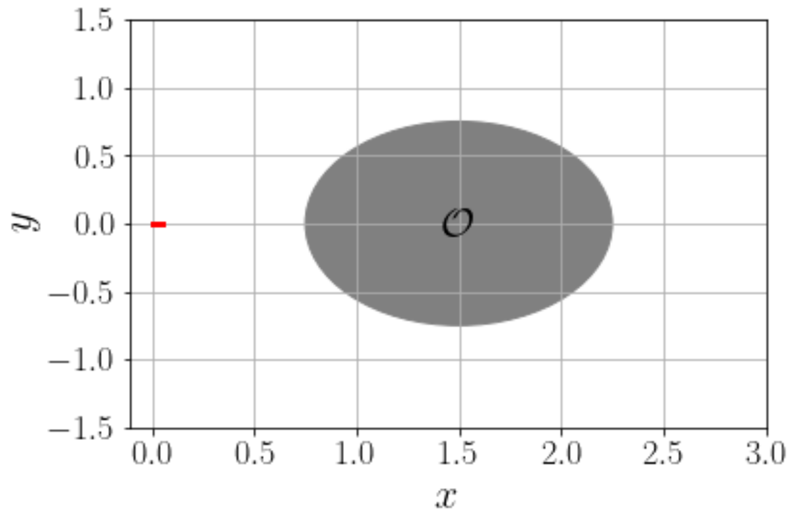
[2]Eindhoven University of Technology, The Netherlands

Motivation

- Policies obtained from RL methods may lack robustness guarantees.
  - Solutions evolve in opposite directions for a small change in the state.
  - A small amount of measurement noise can cause the system to fail.



Autonomous vehicle obstacle avoidance. The vehicle has to steer left or right past the obstacle. In red, the area for which, due to measurement nose, the vehicle can crash into the obstacle.



Simulation of the autonomous vehicle example. A small amount of noise causes the vehicle to crash into the obstacle.

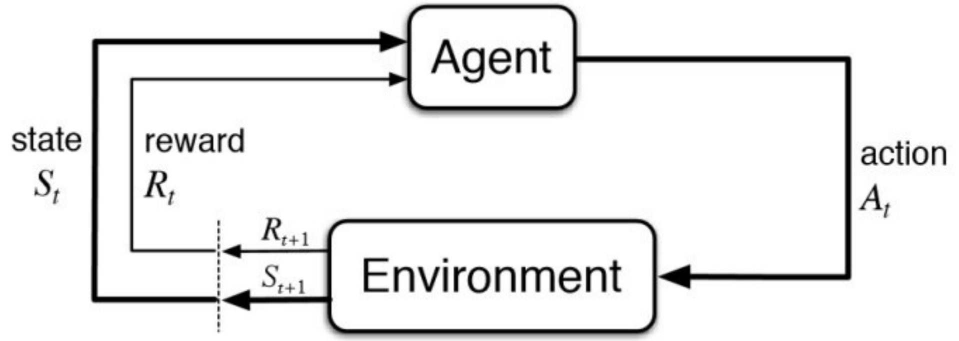
Reinforcement Learning (RL) Methods

The agent (controller) learns how to control the system via direct interactions:

The agent aims to maximize the expected discounted return

$$G_k \coloneqq \sum_{l=0}^{\infty} \gamma^l R_{k+l+1}.$$

Colloquially, the agent aims to find the policy that maximizes the episodic reward.



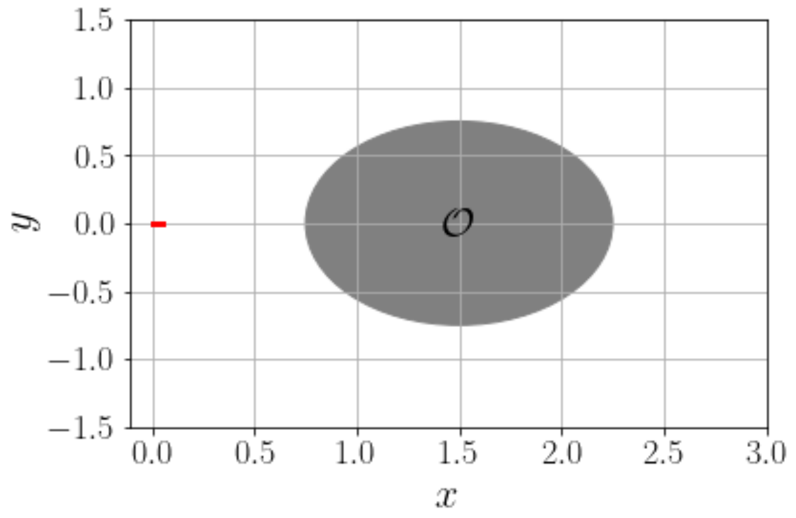
PPO and DQN

- Proximal Policy Optimization (PPO)
  - Policy-based continuous control method
  - Actor-critic approach to find a stochastic control policy $\pi^s : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$, which maps a state-actions pair to a probability
  - Control policy is found by sampling from π^s
- Deep Q-Network (DQN)
  - Action-value based discrete control method
  - The action-value function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, which maps a state-action pair to a real number, that is, the expected discounted return given the state-action pair
  - Control policy is obtained by maximizing Q, i.e., $\pi^*(s) = \arg_{u \in \mathcal{A}} \max Q(s,u)$

A Challenge to RL

- Optimal control policies generate trajectories evolving in <u>opposite</u> directions from some region of the state space for a <u>small change</u> in the state.
  - Arises when environmental rewards are *"symmetric".*

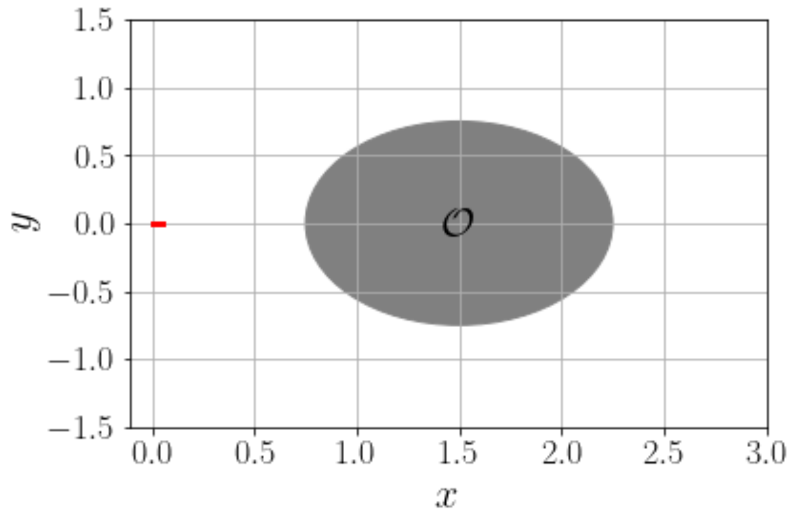
- Examples of such systems:
  - Unit circle problem
  - Obstacle avoidance problem
  - General systems on manifolds



A Challenge to RL

- Optimal control policies generate trajectories evolving in opposite directions from some region the state space for a small change in the state.
  - Arises when environmental rewards are *"symmetric".*

- Examples of such systems:
  - Unit circle problem.
  - Obstacle avoidance problem.
  - General systems on manifolds.

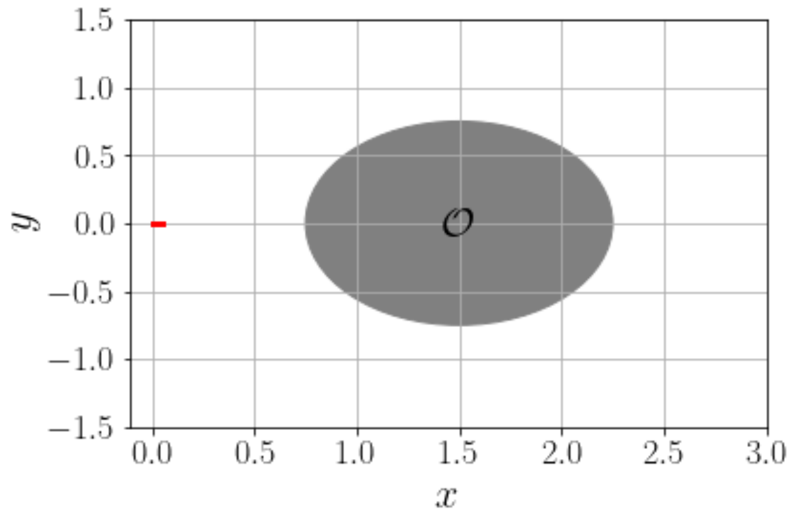


Hybrid feedback control can overcome this problem!

A Challenge to RL

- Optimal control policies generate trajectories evolving in <u>opposite</u> directions from some region the state space for a <u>small change</u> in the state.
    - Arises when environmental rewards are *“symmetric”.*

- Examples of such systems:
    - Unit circle problem.
    - Obstacle avoidance problem.
    - General systems on manifolds.



Hybrid feedback control can overcome this problem!

Hybrid Reinforcement Learning

Example: Unit Circle

- Dynamics:

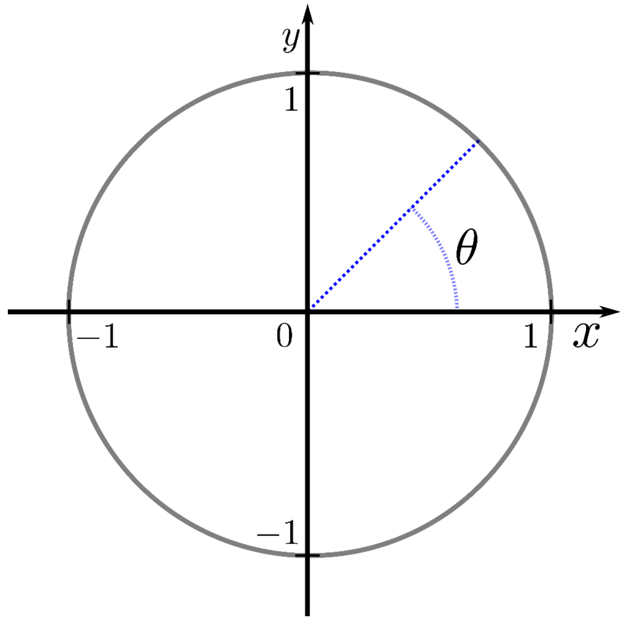
  $$\dot{\xi} = f(\xi, u) = u \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \xi \qquad \xi = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathcal{S}^1$$

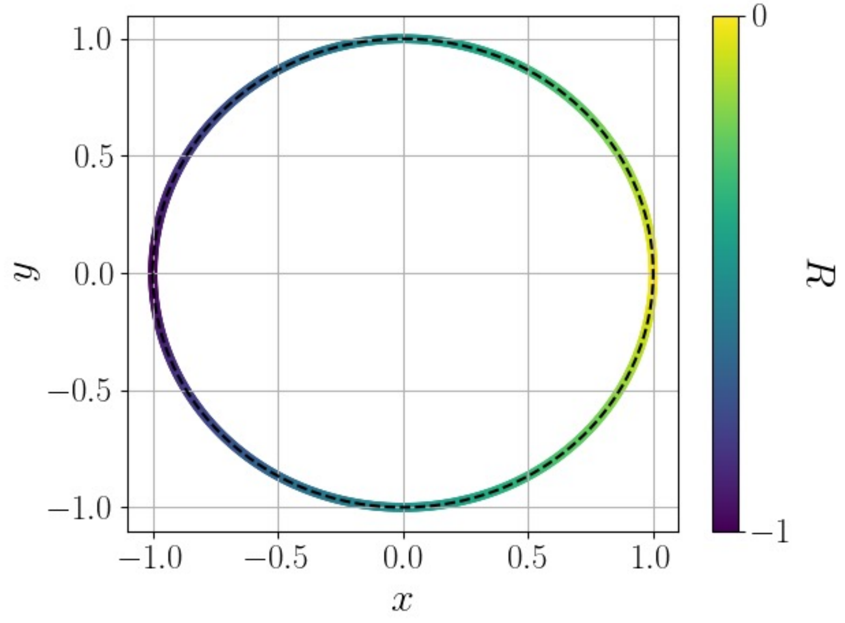
  $$\mathcal{S}^1 := \{\xi \in \mathbb{R}^2 \mid |\xi| = 1\}$$

- **Goal**: stabilize the set-point $\xi^* = (1,0)$
- Observation vector: $o(\xi) = \xi$
- Reward function:

  $$R(x, y) = -\frac{1}{\pi} |\operatorname{arctan2}(y, x)|$$

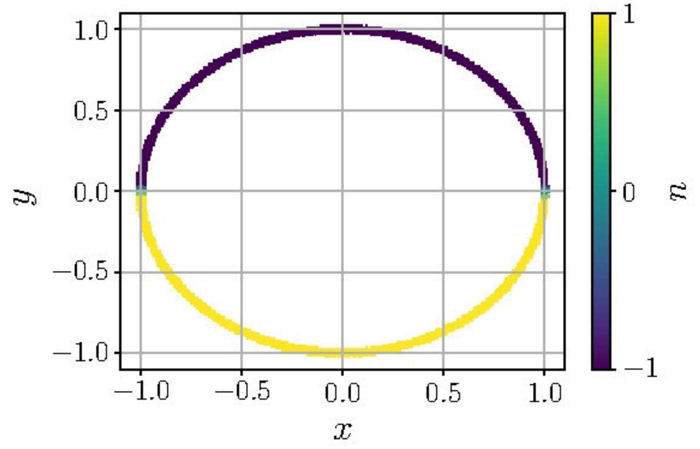
  - *Symmetric* in the sense that $R(x, y) = R(x, -y)$





Example: Unit Circle

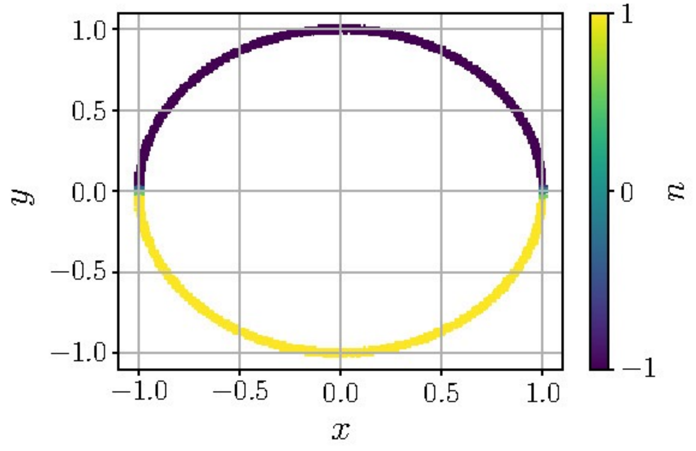
- Optimal policy is found using PPO:
  - Move clockwise if $y > 0$
  - Move counterclockwise if $y < 0$
  - Policy parameterization is continuous, hence there exists a point $\xi_c \neq \xi^*$ for which $\pi^*(\xi_c) = 0$



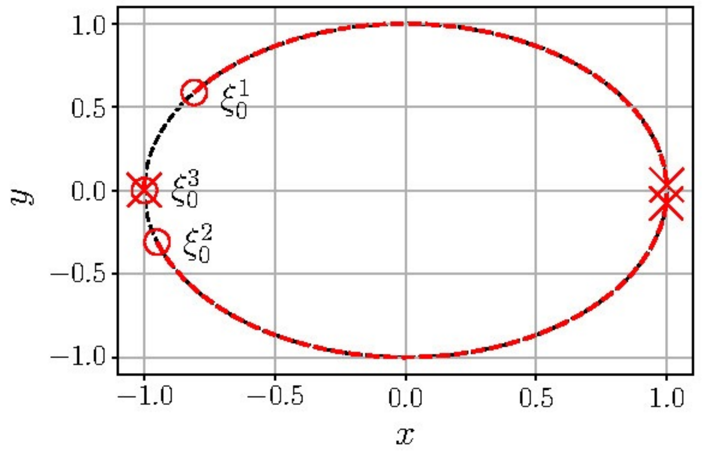
Control policy found by PPO..

Example: Unit Circle

- Optimal policy is found using PPO:
  - Move clockwise if $y > 0$
  - Move counterclockwise if $y < 0$
  - Policy parameterization is continuous, hence there exists a point $\xi_c \neq \xi^*$ for which $\pi^*(\xi_c) = 0$
- Small amount of noise cause the system to get stuck at the critical point $\xi_c = (-1, 0)$:
  - Noise is added to the angle of the state: $\angle\xi_{perturbed} = \angle\xi + \epsilon$
  - System lacks robustness against small measurement noise of magnitude $\epsilon = 0.1$

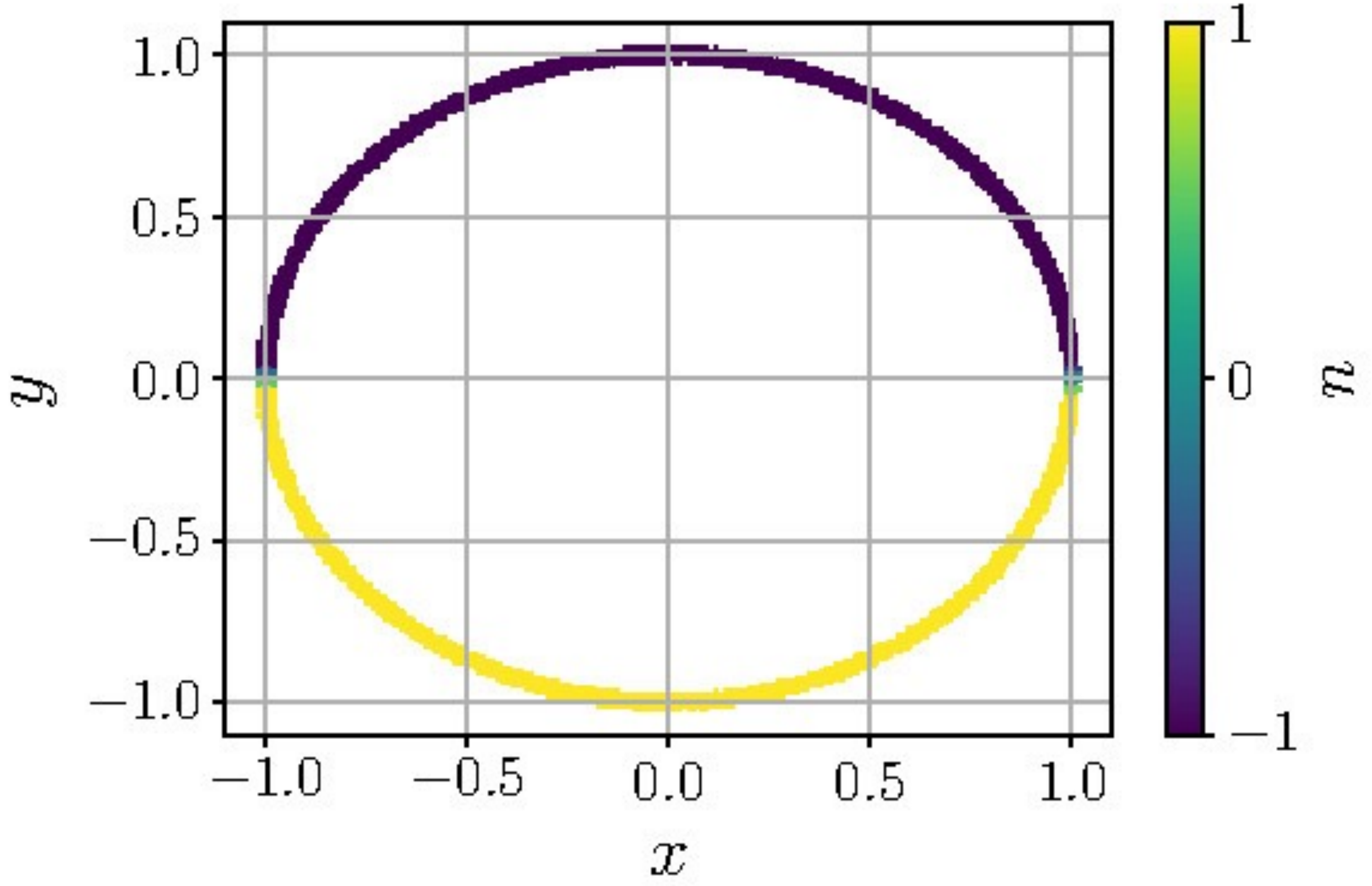


Control policy found by PPO..



Simulations of the control policy for various initial conditions in the presence of small measurement noise of magnitude $\epsilon = 0.1$..

Example: Unit Circle



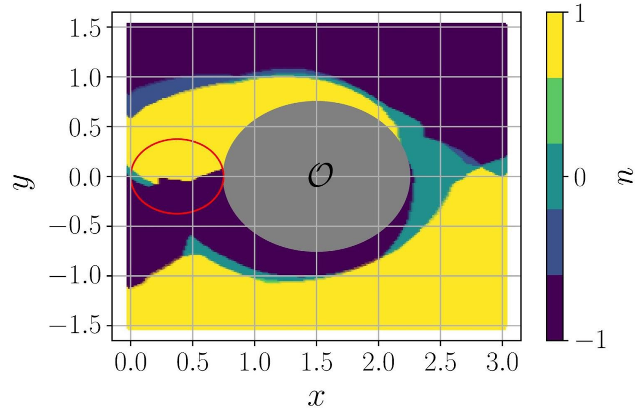
Example: Obstacle Avoidance

- Dynamics:
  $\dot{\xi} = f(u) = \begin{bmatrix} 1 \\ u \end{bmatrix} \qquad \xi = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathcal{S}$
- **Goal**:
  Drive past the obstacle and reach the set-point $\xi^* = (3,0)$.
- Observation vector: $o(\xi) = \left[d_{ob}, d_{sp}, y\right]^T$:
  - Shortest distance to the obstacle d_{ob};
  - Shortest distance to the set-point d_{sp};
  - Vertical position y.
- Reward function:
  $$R(\xi) = \max\left(-d_{sp}(\xi) - 0.1((d_{ob} - 2r_{ob})^2 - \ln(d_{ob})) + 3.5, 0\right)$$
  - *Symmetric* in the sense that $R\left(d_{ob}, d_{sp}, y\right) = R(d_{ob}, d_{sp}, -y)$.

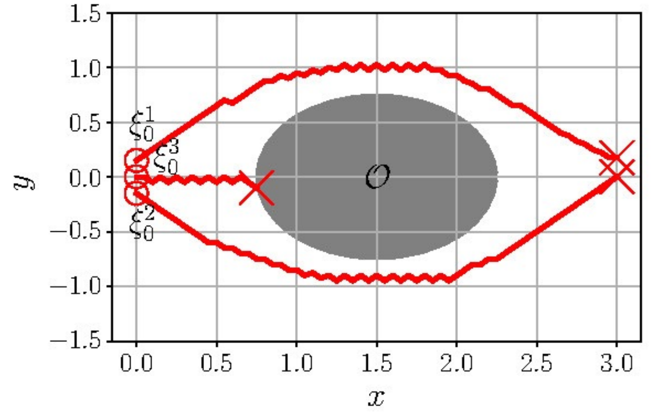
Example: Obstacle Avoidance

- Optimal policy is found using DQN:
  - Move above the obstacle if $y > 0$;
  - Move below the obstacle if $y < 0$.

- Small amount of noise cause the vehicle to crash.
  - Noise is added to the vertical position y: $y_{perturbed} = y + \epsilon$.
  - System lacks robustness against small measurement noise of magnitude $\epsilon = 0.1$.

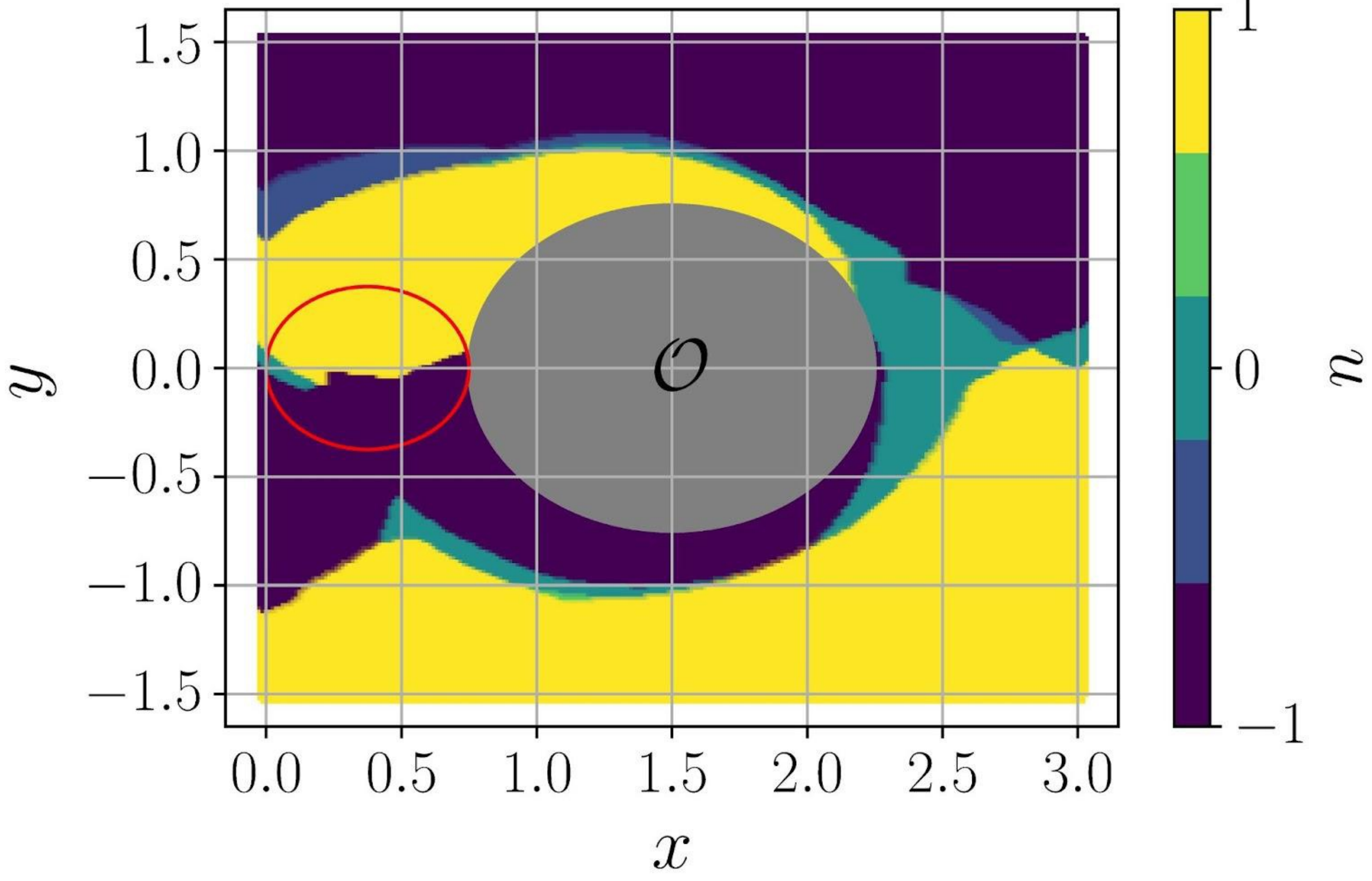


The policy found by DQN: solutions evolve in opposite directions for a small change in y.



Simulation of the DQN policy for various initial conditions in presence of noise signal of magnitude $\epsilon = 0.1$ on y. The vehicle crashes into the obstacle due to the noise.

Lack of Robustness: Obstacle Avoidance



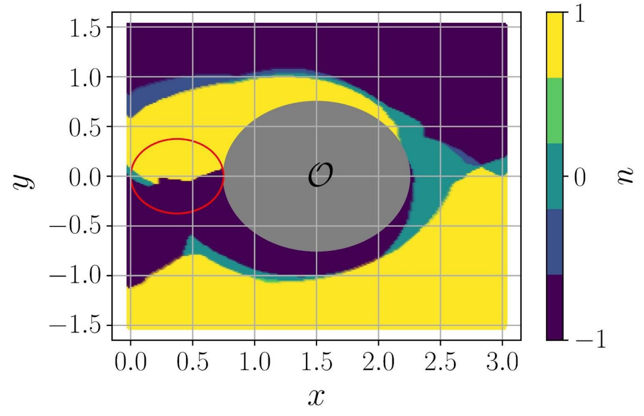
Overview: A Hybrid RL (HyRL) Algorithm

- **Goal**:
  Obtain robustness against measurement noise
- Steps:
  - The environment is split up into two overlapping sets where the intersection is the set of critical points.
  - The overlapping sets are extended.
  - The RL method of choice is used to find two new control policies for each of the extended overlapping sets.
  - A hybrid system is built that incorporates a hysteresis switching effect between the two newly found control policies.

Walkthrough HyRL Algorithm

1. **Run an RL algorithm to find a control policy.**
2. Determine the set of critical points.
3. Split up the environment.
4. Extend the overlapping sets.
5. Find two new policies for each extended set.
6. Build the hybrid system.

Obstacle Avoidance Example

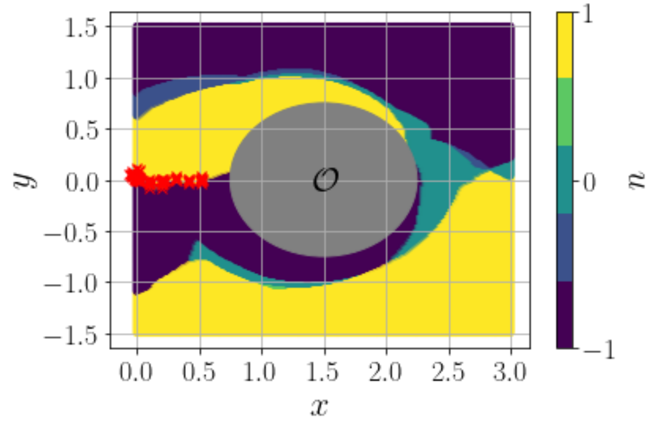


Step 1) The policy found by DQN: solutions evolve in opposite directions for a small change in y.

Walkthrough HyRL Algorithm

1. Run an RL algorithm to find a control policy.
2. **Determine the set of critical points.**
3. Split up the environment.
4. Extend the overlapping sets.
5. Find two new policies for each extended set.
6. Build the hybrid system.

Obstacle Avoidance Example

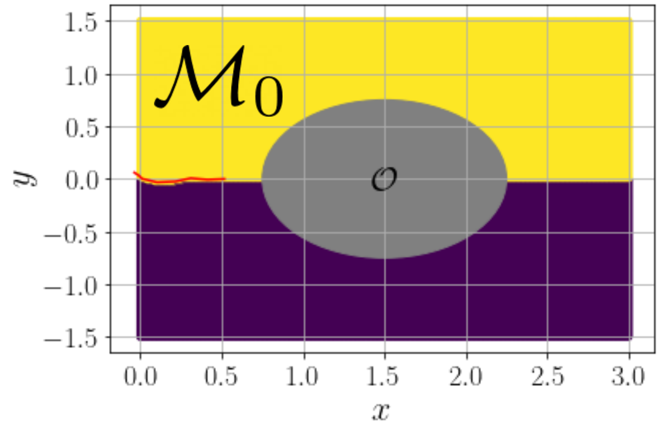


Step 2) Finding the critical points. More details on github.com/HybridSystemsLab/ObstacleAvoidanceHyRL.

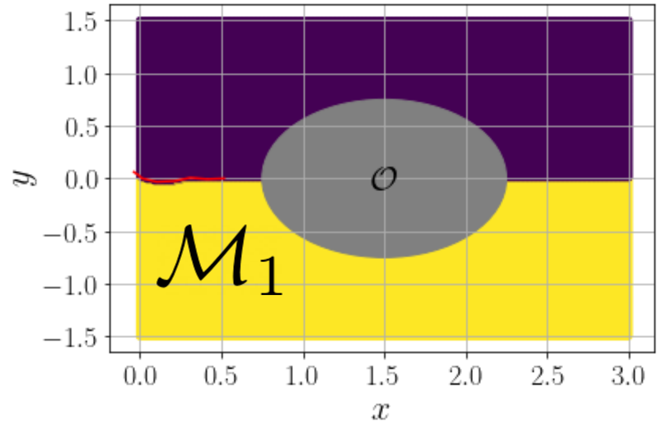
Walkthrough HyRL Algorithm

1. Run an RL algorithm to find a control policy.
2. Determine the set of critical points.
3. **Split up the environment.**
4. Extend the overlapping sets.
5. Find two new policies for each extended set.
6. Build the hybrid system.

Obstacle Avoidance Example



Step 3) In yellow, the splitted set.

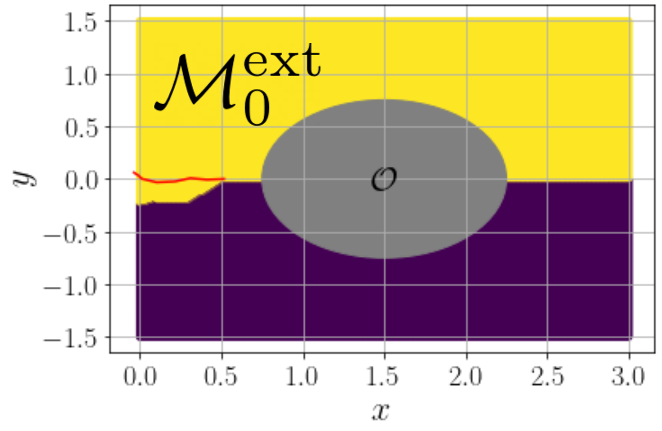


Step 3) In yellow, the splitted set.

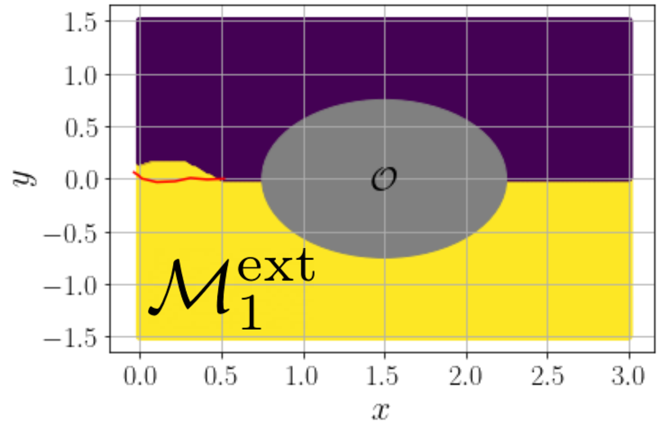
Walkthrough HyRL Algorithm

1. Run an RL algorithm to find a control policy.
2. Determine the set of critical points.
3. Split up the environment.
4. **Extend the overlapping sets.**
5. Find two new policies for each extended set.
6. Build the hybrid system.

Obstacle Avoidance Example



Step 4) In yellow, the extended set.

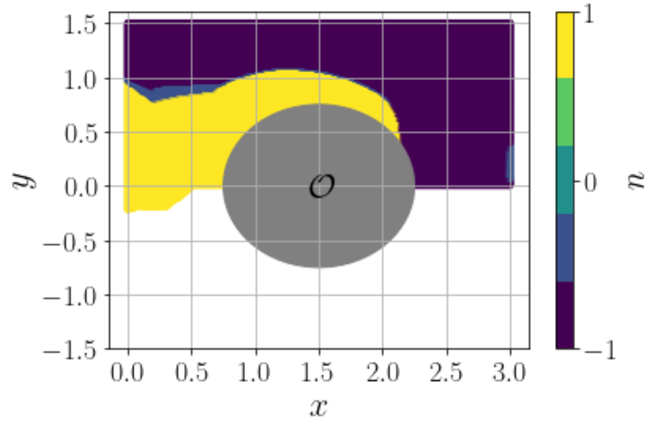


Step 4) In yellow, the extended set.

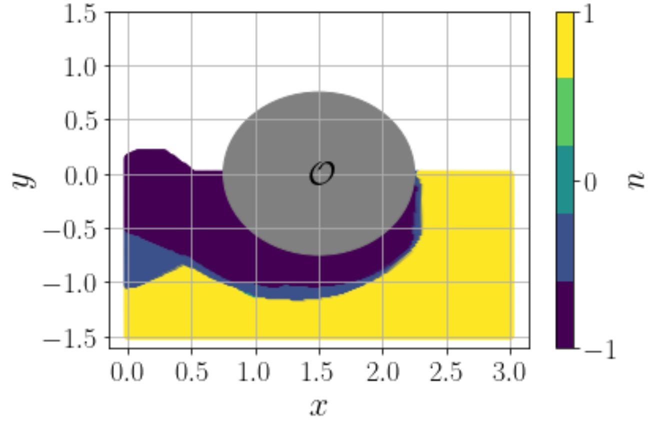
Walkthrough HyRL Algorithm

1. Run an RL algorithm to find a control policy.
2. Determine the set of critical points.
3. Split up the environment.
4. Extend the overlapping sets.
5. **Find two new policies for each extended set.**
6. Build the hybrid system.

Obstacle Avoidance Example



Step 5) The new policy π_0^*.



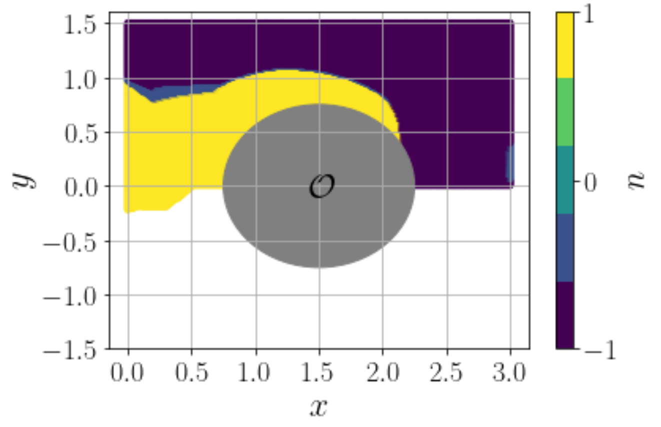
Step 5) The new policy π_1^*.

Walkthrough HyRL Algorithm

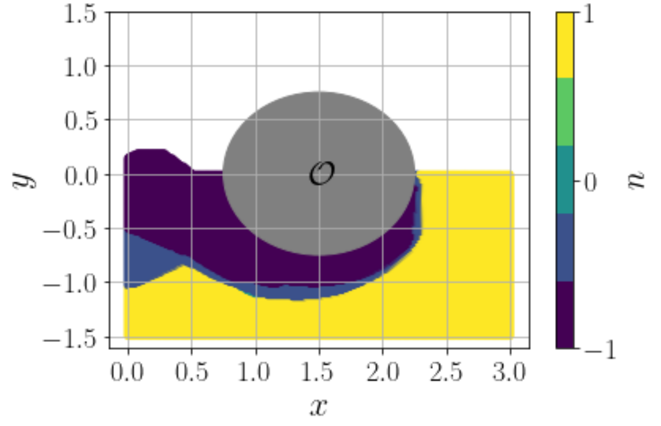
6. Build the hybrid system $\mathcal{H} = (C, F, D, G)$:

- Hybrid state $z = (\xi, q) \in \mathcal{S} \times \{0,1\}$
- Flow map $\begin{bmatrix} \dot{\xi} \\ \dot{q} \end{bmatrix} = F(z) := \begin{bmatrix} f(\xi, \pi_q^*(o(\xi))) \\ 0 \end{bmatrix} \quad z \in C,$
- Flow set $C := \bigcup_{q \in \{0,1\}} (\mathcal{M}_q^{ext} \times \{q\})$
- Jump map $\begin{bmatrix} \xi^+ \\ q^+ \end{bmatrix} = G(z) := \begin{bmatrix} \xi \\ \begin{cases} 1 \text{ if } q = 1 \\ 0 \text{ if } q = 0 \end{cases} \end{bmatrix} \quad z \in D,$
- Jump set $D := \bigcup_{q \in \{0,1\}} \left(\overline{(\mathcal{S} \setminus \mathcal{M}_q^{ext})} \times \{q\} \right)$

Obstacle Avoidance Example



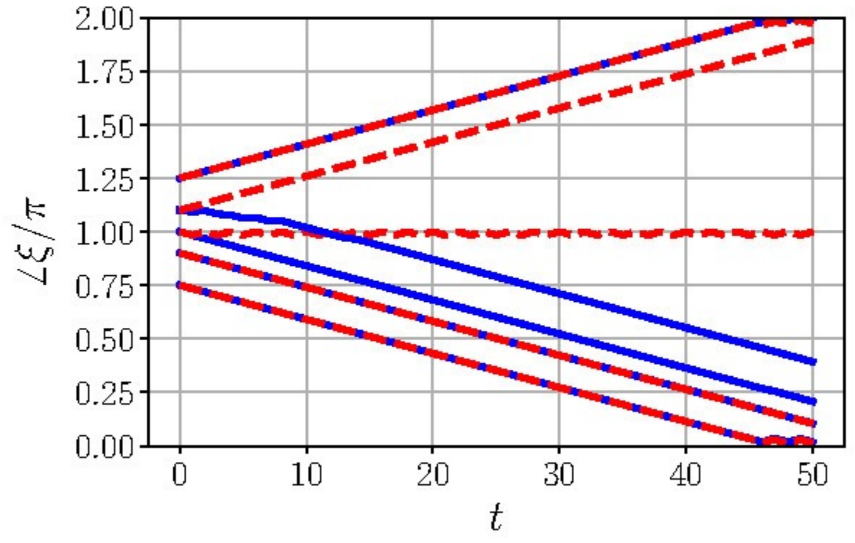
Step 6) The policy when $z \in \mathcal{M}_0^{ext} \times \{0\}$.



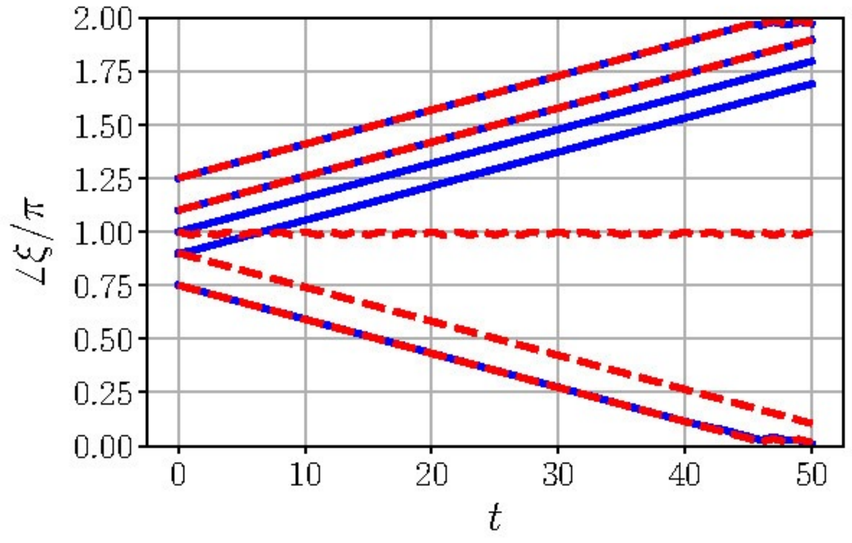
Step 6) The policy when $z \in \mathcal{M}_1^{ext} \times \{1\}$.

Robustness for Unit Circle

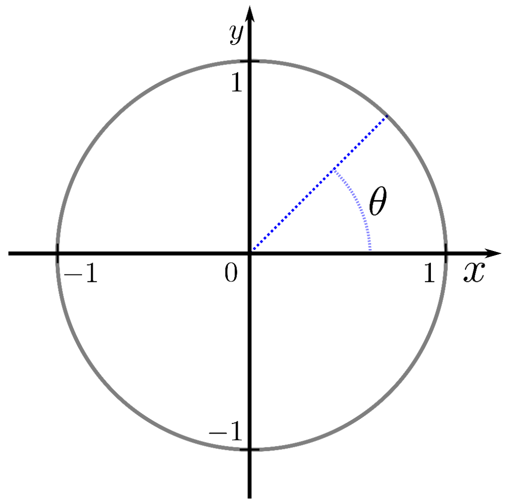
- HyRL vs DQN in presence of the same noise signal of $\epsilon = 0.1$



Initialized with $q_0 = 0.$



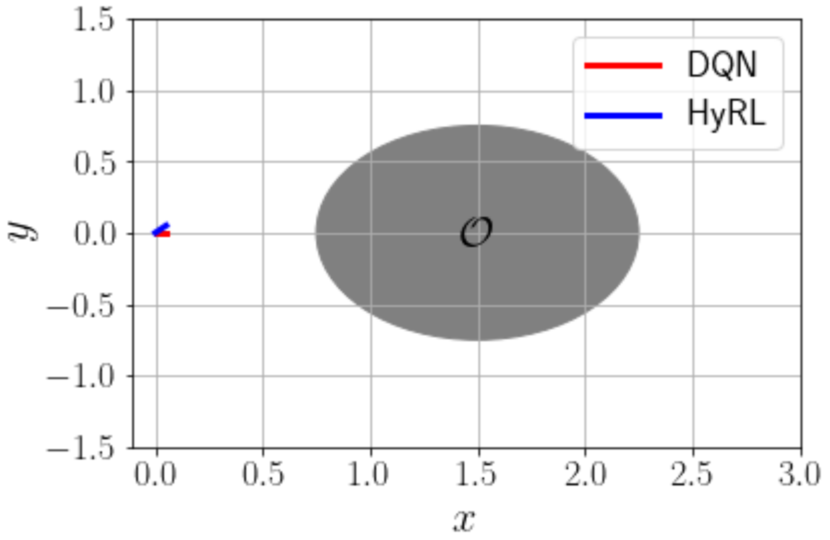
Initialized with $q_0 = 1.$



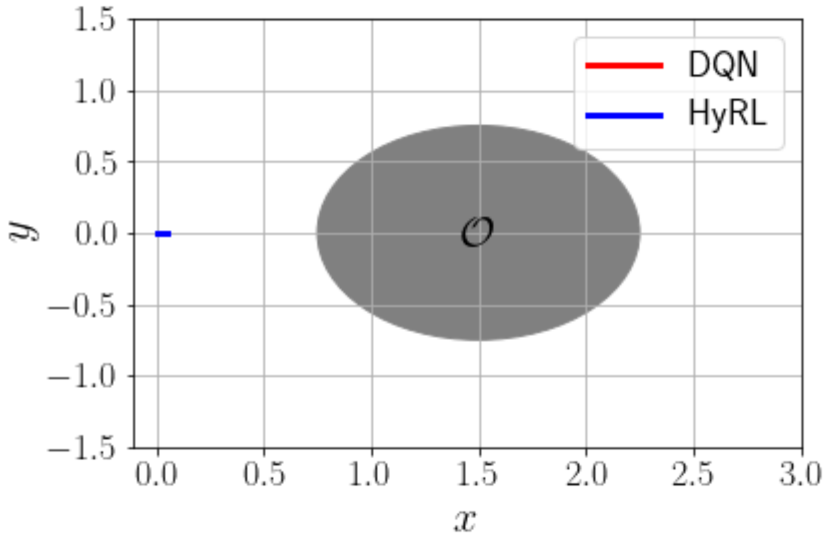
- Hybrid basic conditions are satisfied
- The overlapping region is larger than ϵ, hence the closed-loop hybrid system is <u>robust</u> against the measurement noise of magnitude ϵ

Robustness for Obstacle Avoidance

- HyRL vs DQN in presence of the same noise signal of $\epsilon = 0.1$



Initialized with $q_0 = 0$.



Initialized with $q_0 = 1$.

- Hybrid basic conditions are satisfied
- The overlapping region is larger than ϵ, hence the closed-loop hybrid system is <u>robust</u> against the measurement noise of magnitude ϵ

Future Work

Thank you for your attention

Research funded by AFOSR under Grant no. FA9550-19-1-0169

- Formal proofs:
  - (Global) asymptotic stability;
  - Robustness.
- Safe learning:
  - Guaranteed stable and robust behavioral policy during training.
- Algorithm improvements:
  - Finding the set of critical points;
  - Inflating the overlapping sets $\mathcal{M}_0$ and $\mathcal{M}_1$.
- Hybrid policy parameterization.