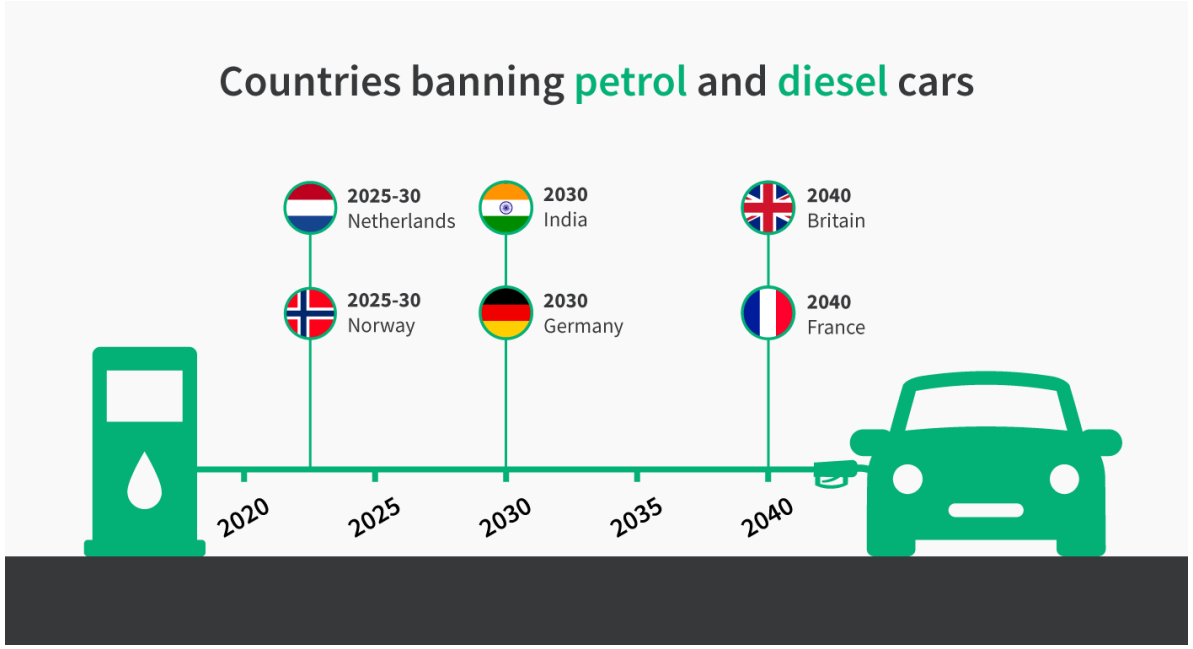
Deep Reinforcement Learning for the control of the flow past bluff bodies
ATE HEFAT 2021

P. Mudiyanselage, F. Gueniat
florimond.gueniat@bcu.ac.uk
July 2021
Birmingham City University

BIRMINGHAM CITY University

Electric vehicles' Revolution



My takeaway

Transition pitfalls :

- Compensation for fuel : massive growth of the electric grid
- Customer concerns : autonomy & emissions

There is a global need for vehicles that are more energy efficient.

It essentially means controlling the drag, which represents 40% of the consumption.

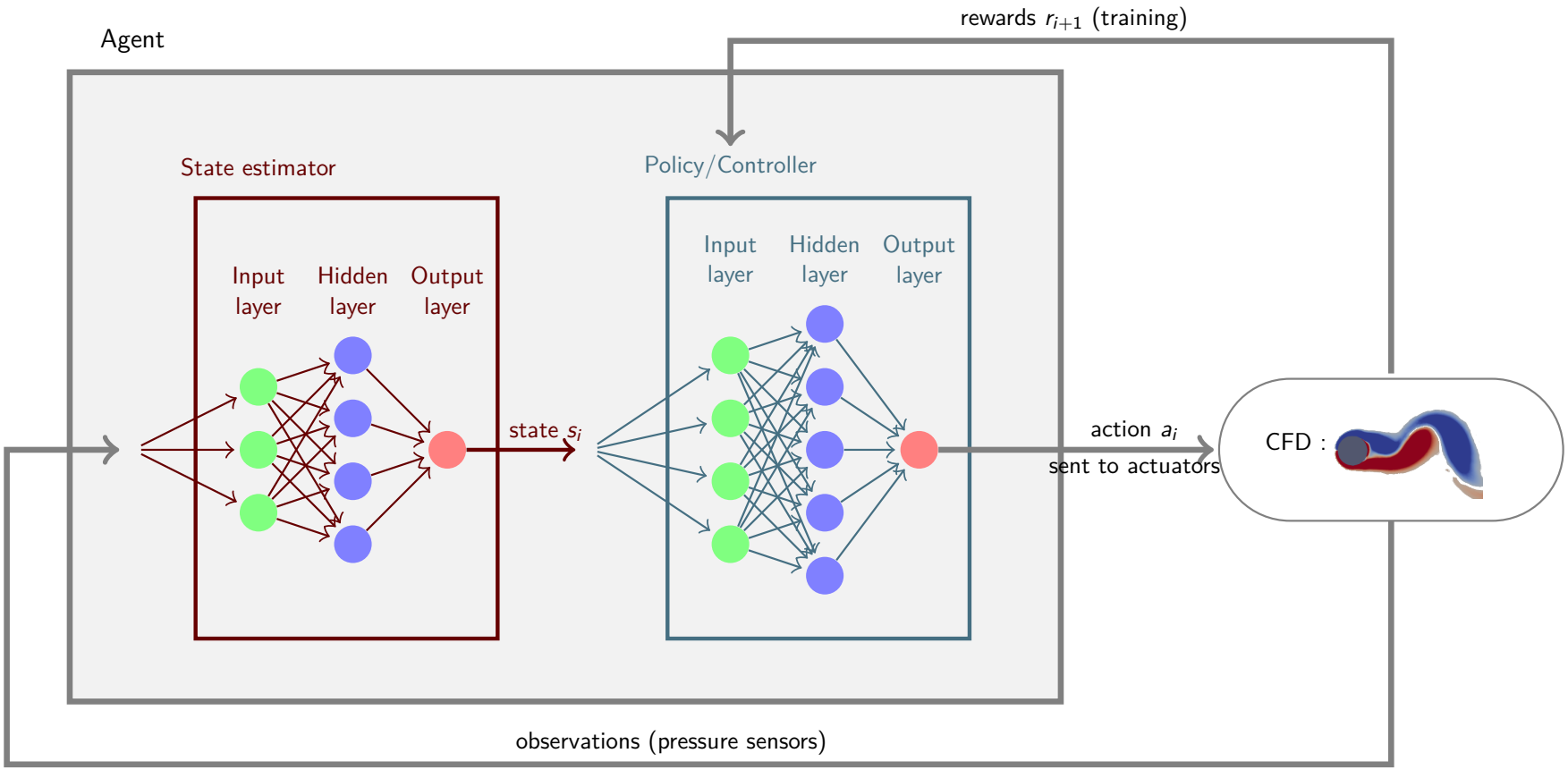
Active flow control

Despite decades of research, active drag reduction has shown little practical successes.

- Fast controller
- Cheap actuators/sensors
- Computationally expensive models
- Limited access to sensors/information

These constrains correspond to the strength of data-driven methods.

Active Flow Control Workflow

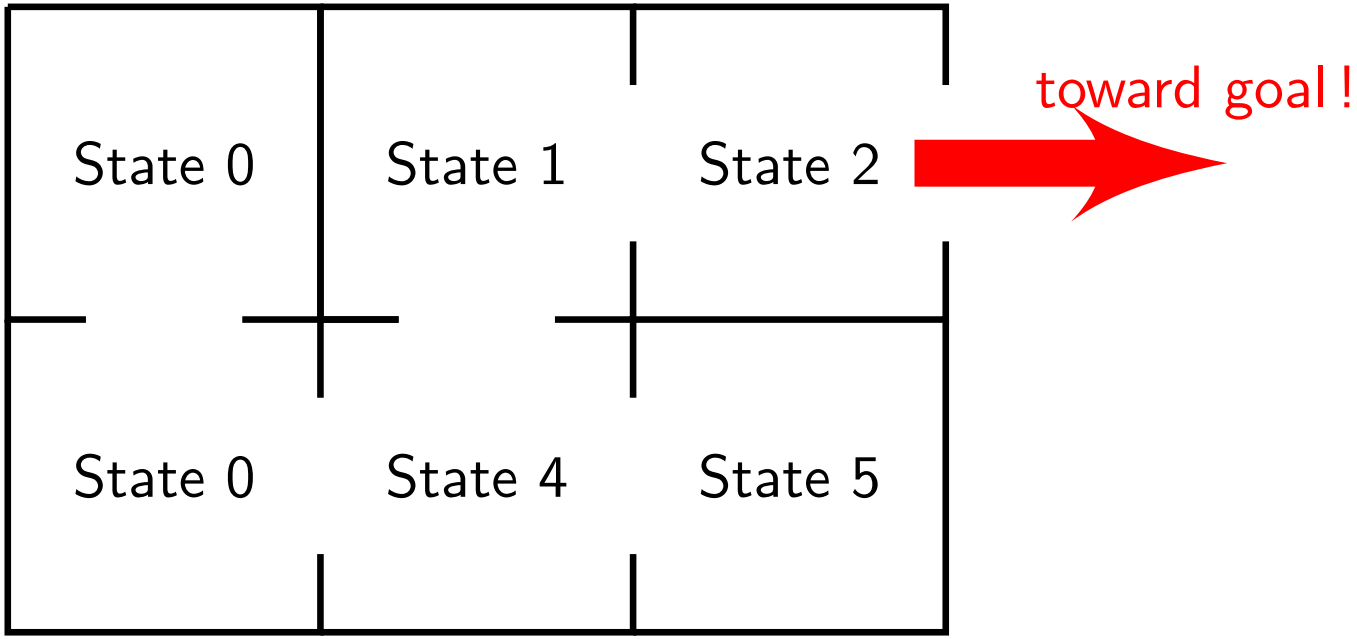


Schematic workflow of the deep reinforcement learning method applied to flow control.

Outline and contributions

- Elements of Deep Reinforcement Learning
- Application to the reduction of the drag in the flow past a cylinder

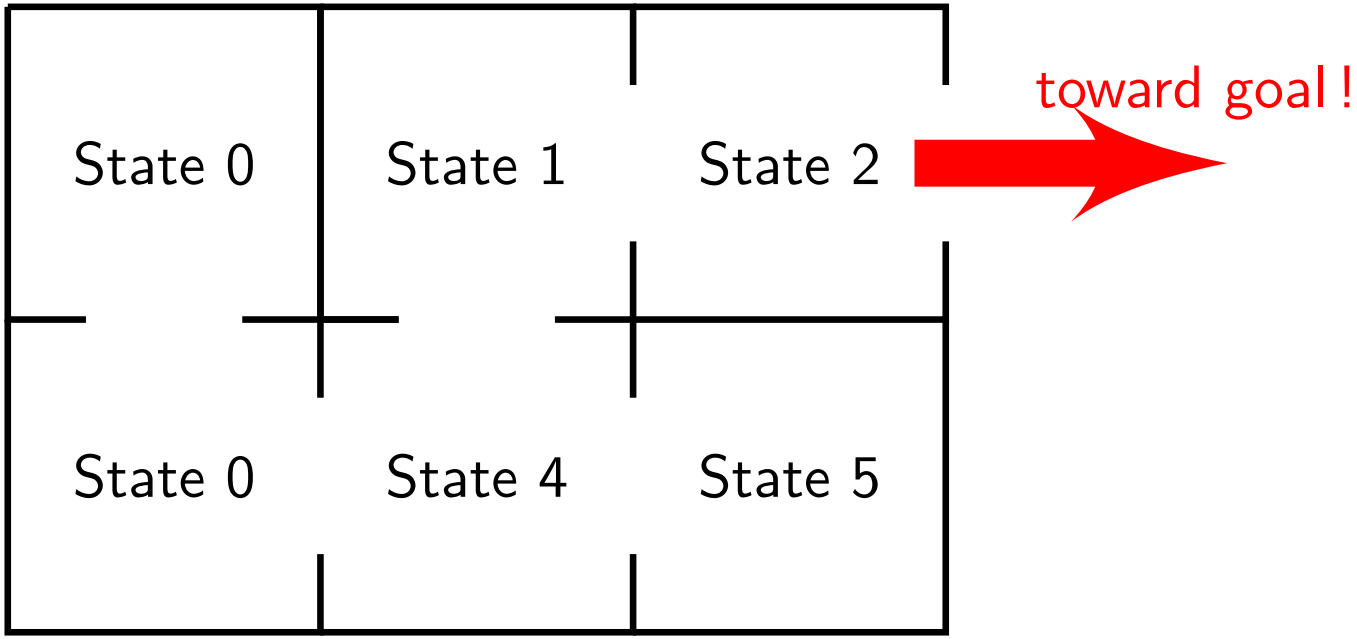
Amazing illustration



State s of the system at a given time. Directly from readings :

- temperature
- velocity
- in this work : pressure readings.

Amazing illustration



Evolution from state s_i to state s_{i+1}:, it is a transition. Transitions exists under an action a_i (which can be null).

- angle of an elevator
- increasing a voltage
- in this work : mass flow rate.

Rewards and Return

To each transition, from a state s_i and under the action a_i, to the state s_{i+1} is associated a reward r_i (e.g. metrics related to the drag).

Running one experiment (aka CFD run), we record a "trajectory" :

$$\tau = \{(s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_N, a_N, r_N)\}$$

The Return is the sum of the rewards from time i :

$$R_i(\tau) = \sum_i^N \gamma^i r_i$$

R represents the "quality" of a trajectory τ starting in i.

The discount γ helps with convergence, and focusing on a reasonable horizon.

Rewards

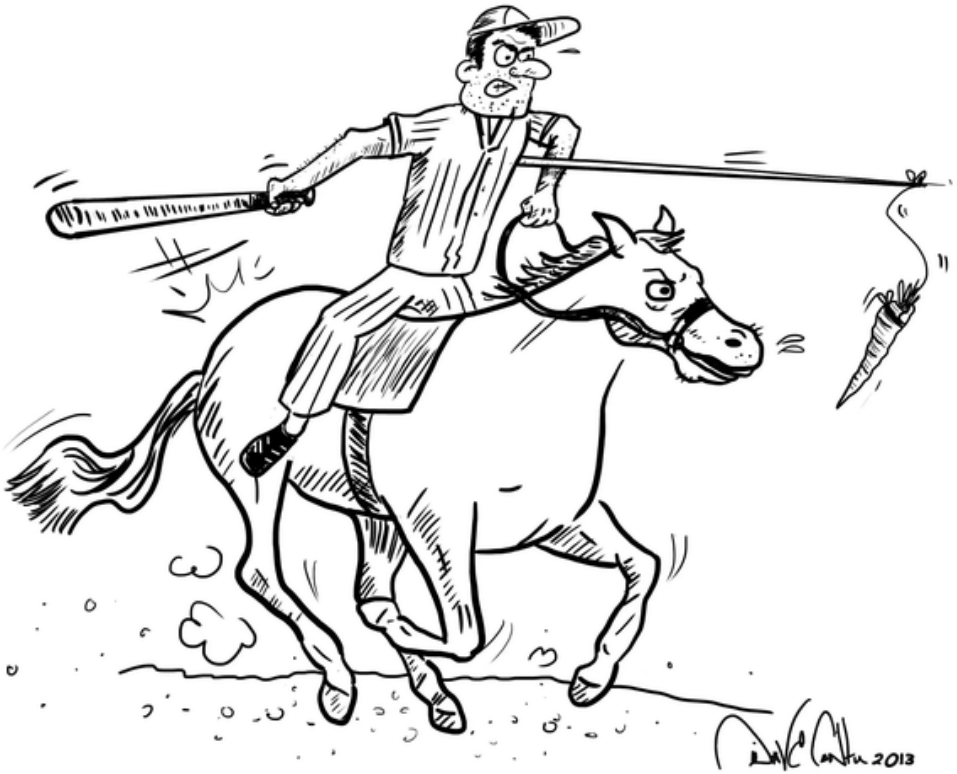
The reward function $r := r(s, a, s')$ is hand crafted. It has to measure well the goal, for instance reducing the drag. It is probably the hardest part : the control algorithm with try to break it. It should ideally be :

- positively affected by how small the distance to the objective is.
- negatively affected by the intensity of the action.

It is totally fine to have negative rewards to penalize undesired behaviours.

Carrot and stick

It is nothing else but the mathematical version of learning via punishments and rewards !



Value function and Q-function

One experiment is not enough to ensure a policy works well. We define two functions :

$$V(s) = \mathbb{E}_\tau \left[R_0(\tau) | s = s_0 \right]$$

$$Q(s, a) = \mathbb{E}_\tau \left[R_0(\tau) | s = s_0, a = a_0 \right]$$

Assuming the actions are taken from a policy : $\pi(s_i) = a_i$.
V^π represents the "quality" of a state s when following the policy π.

Value function and Q-function

One experiment is not enough to ensure a policy works well. We define two functions :

$$V(s) = \mathbb{E}_\tau \left[R_0(\tau) | s = s_0 \right]$$

$$Q(s, a) = \mathbb{E}_\tau \left[R_0(\tau) | s = s_0, a = a_0 \right]$$

It is actually the discrete form of the Bellman's equation :

$$V^\pi(s) = r^\pi\big(\pi(s)\big) + \gamma \sum_{s'} p\big(s, \pi(s), s'\big) V^\pi_{s'}.$$

Value function and Q-function

One experiment is not enough to ensure a policy works well. We define two functions :

$$V(s) = \mathbb{E}_\tau \left[R_0(\tau) | s = s_0 \right]$$

$$Q(s, a) = \mathbb{E}_\tau \left[R_0(\tau) | s = s_0, a = a_0 \right]$$

The Q-function Q^π represents the "quality" of an action a taken in state s, or of the tuplet (s, a), and then following the policy π.

Objective : optimizing the controller - the policy

The actions are taken from a policy : $a_i = \pi(s_i)$. It is strictly equivalent of a control law.

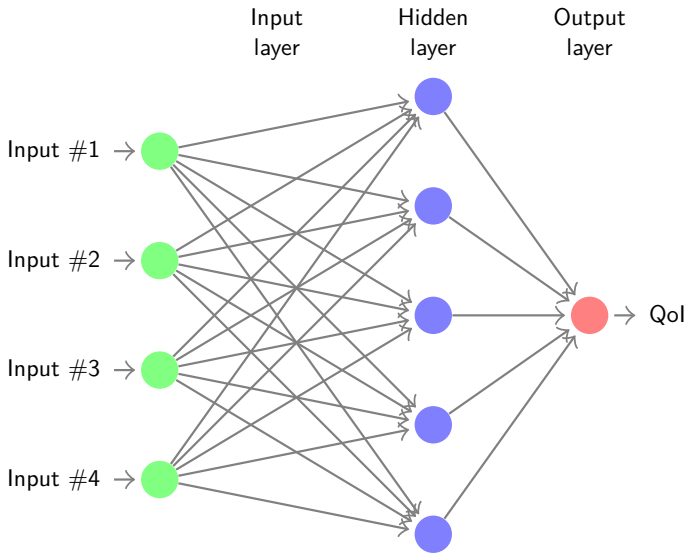
Reducing the drag now means to maximize the value $J(\pi) := V^\pi = \mathbb{E}[R^\pi]$, by finding the policy π :

$$\pi = \arg\max_\pi J(\pi)$$

Generalization via deep learning

Neuralnets allow to approximate functions.

The training will correspond in fitting the parameters θ of nets to approximate $\pi_\theta(s)$, $V^{\pi_\theta}(s)$ and $Q^{\pi_\theta}(s, a)$.



Fitting the Q-function

The Q-function is trained using temporal difference. Having access to a transition (s, a, r, s', a') :

$$Q(s,a) \leftarrow \underbrace{Q(s,a)}_{\text{old value}} + \alpha \left(r(s,a,s') + \gamma \underbrace{Q(s',a')}_{\text{predicted Q}} - \underbrace{Q(s,a)}_{\text{old value}} \right)$$

Note that it is purely data driven as it depends on a transition ! We can estimate how good a policy is, but how to update the policy ?

Policy gradient & Trust Region Policy Optimization

The policy is represented by a network with parameters θ.
It is possible to train the policy via the policy gradient :

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \mathbb{E}\left[\sum_{i=1}^{N} R_i(\tau)\nabla_\theta \log \pi_\theta(a_i|s_i)\right] \\ &\approx \sum_{i=1}^{N} R_i(\tau)\nabla_\theta \log \pi_\theta(a_i|s_i) \end{aligned}$$

Note that it is purely data driven as it depends on the trajectory τ !

Practically, the TRPO update the policy only in the regions where there is little mismatch between the old and the new policy.

It acts as a regularization : the updated policy's behavior is not drastically different from the current policy's behavior.

Configuration

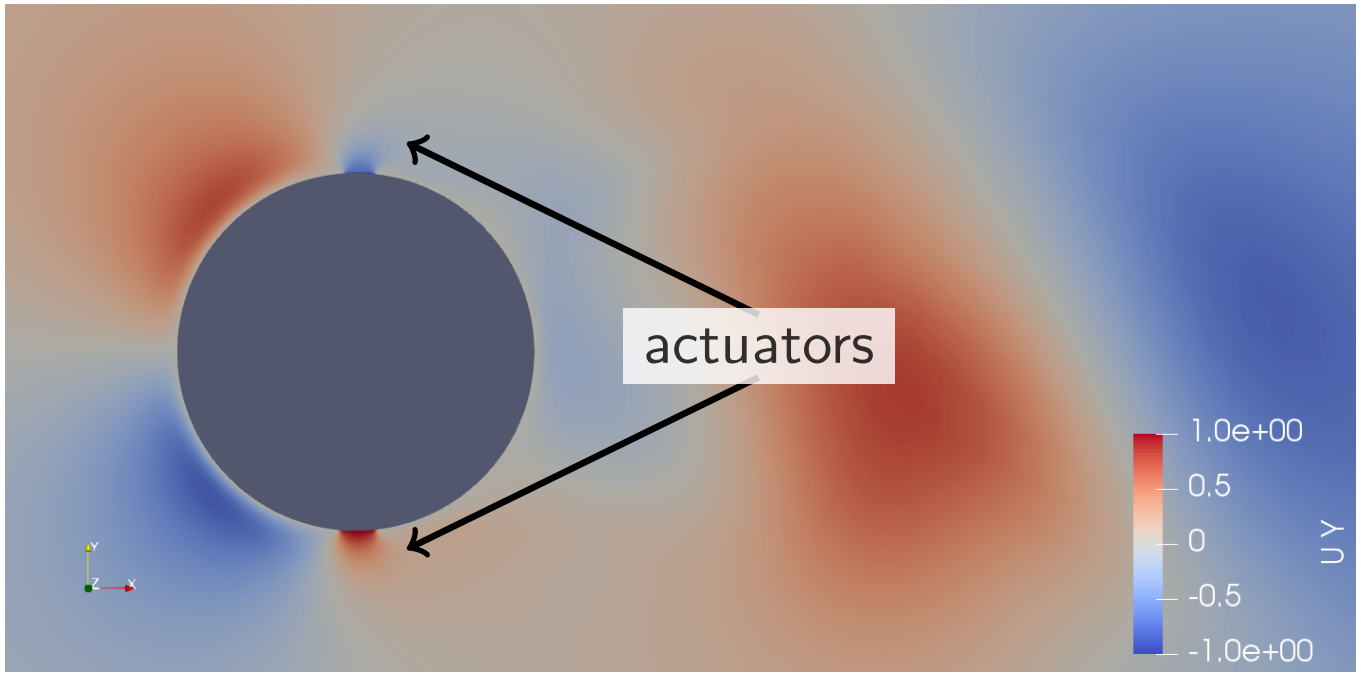
Flow past a cylinder at $Re = 200; \nu = 0.05$.

- Observations - 18 wall pressure measurements, evenly distributed.
- Actuations - 2 suction and blowing actuators (normal velocity BC, amplitude max : $5m/s$)

This case allows to know an oracle : constant removal of fluids from the boundary layer is optimal.

Solver	Turbulence model	# Cells
pimpleFoam	Realizable $k - \epsilon$	149029

Configuration



Colors encode the vertical velocity.

Rewards

$$r = r_{\mathcal{C}_d} + r_a + r_{reg}$$

- $r_{\mathcal{C}_d} = -w_d(\mathcal{C}_d - 1.55)$. Low drag coefficient $\Rightarrow r_{\mathcal{C}_d} > 0$.
- $r_a = -w_a||a||_2$
- $r_{reg}(i) = -w_{reg} \sum_{k=0}^{N_{reg}} |\mathcal{C}_{d\,i-k} - \mathcal{C}_{d\,i-k-1}|$. It penalizes oscillations.

w_d	w_a	w_{reg}	N_{reg}
1	0.1	0.1	5

Note : compromised between $\mathcal{C}_d$ and a, Oracle does not consider the actions.

Parameters for DRL

Each network is composed of two layers of dimensions 64 with tanh-activations, followed by an LSTM layer of dimensions 256.

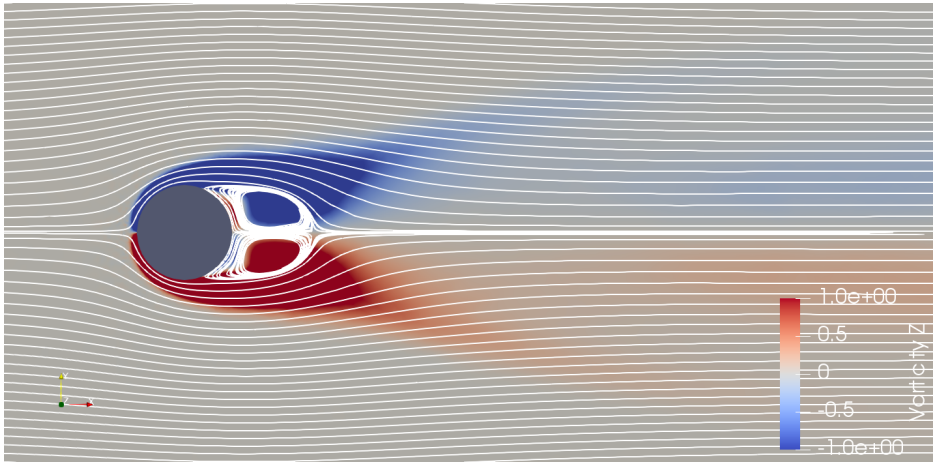
- "only" 500 runs
- one run is 20s, dt=0.01

only 50 000 time steps. Policy are only converged because the optimal policy has a simple structure.

Already a few days on a regular laptop.

Cylinder flow

case	baseline	oracle	present
value	0.16	4.64	5.22
average drag coefficient	1.531 ± 0.035	1.028 ± 0.006	1.086 ± 0.008
average actuation	(0,0)	(-1,1)	(-0.79,0.83)
recirculation length	0.85L	1.05L	1.01L
recirculation height	1L	0.76L	0.82L

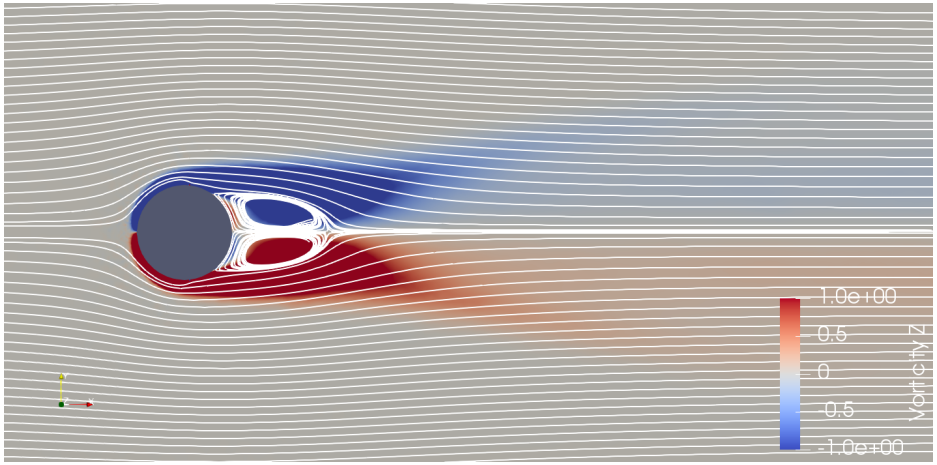


baseline

Streamlines + Colors encode the vorticity.

Cylinder flow

case	baseline	oracle	present
value	0.16	4.64	5.22
average drag coefficient	1.531 ± 0.035	1.028 ± 0.006	1.086 ± 0.008
average actuation	(0,0)	(-1,1)	(-0.79,0.83)
recirculation length	0.85L	1.05L	1.01L
recirculation height	1L	0.76L	0.82L

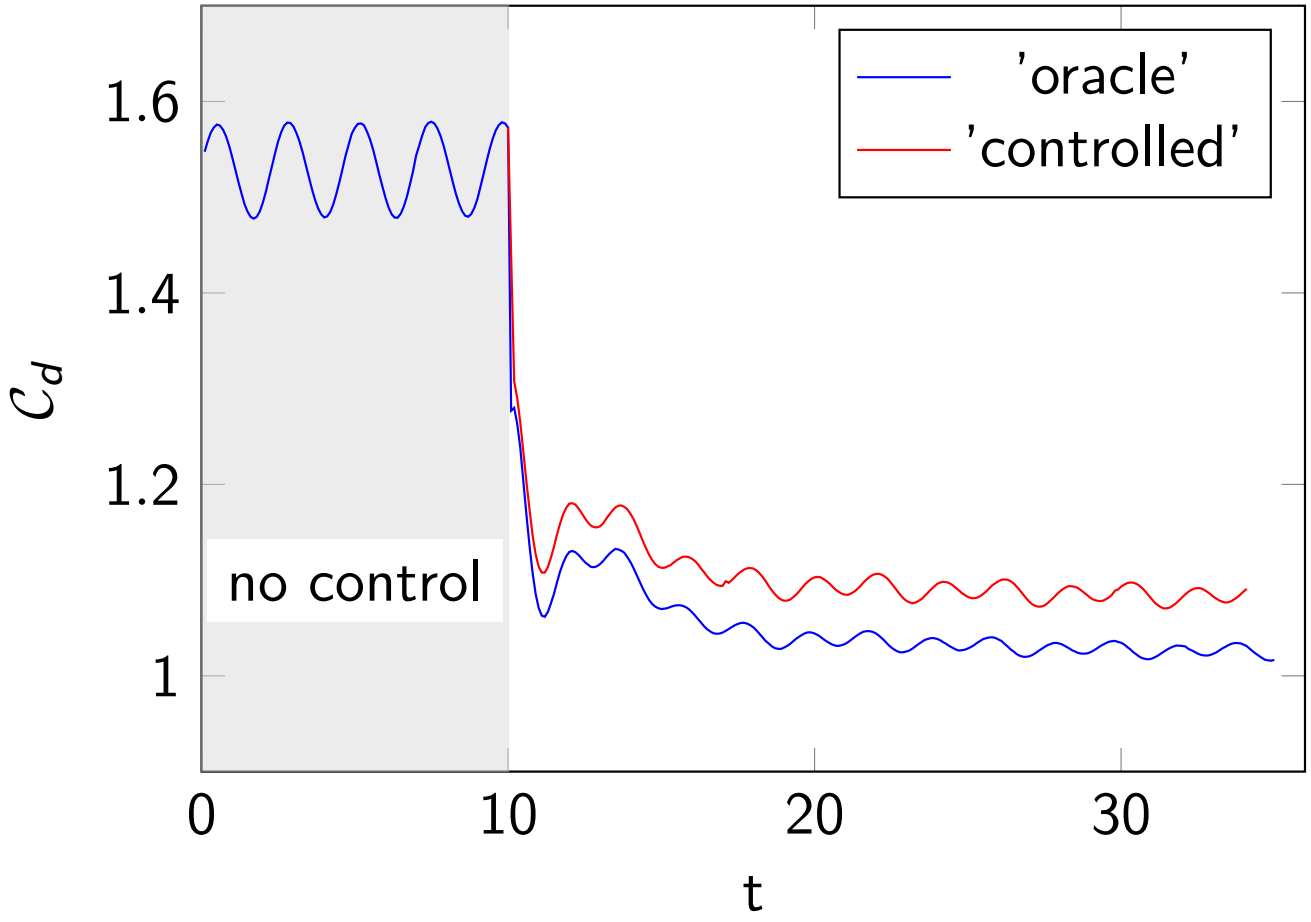


present

Streamlines + Colors encode the vorticity.

Effect on drag

The drag is reduced by almost 30% !



Take away and future work

- Deep Reinforcement Learning can be seen as a black box.
- DRL works as expected, but is extremelly computationally expensive/sample inefficient.
- It seems well suited for experimental work that can be automatized.

We plan to do experiments in the near future.

Any questions ?

Drag Reduction : preliminary experimental results

Preliminary work in our teaching wind tunnel, using an Ahmed body.

We achieved an increase in the wall pressure of 2%, and a drag reduction of 7%.