Closed-loop optimal control for shear flows using reinforcement learning

O. Semeraro[1], M.A. Bucci[2], L. Mathelin[1]

semeraro@limsi.fr

1) LIMSI, CNRS, Université Paris-Saclay, Orsay (FR)
2) TAU, Inria, Université Paris-Saclay, CNRS, LRI, Orsay, (FR)

ANR/DGA FlowCon project — ANR-17-ASTR-0022

Introduction

A controlled system can be thought as a dynamical system where an **agent** interacts with an **environment** to maximize/minimize a **cost function**

$$\mathscr{J} = \min\{\textbf{drag}\}, \max\{\textbf{lift}\}, \min\{\textbf{noise}\} \ldots$$

Optimal control: the nonlinear case

$$\mathcal{J}(x(t), t) = \max_u \int_t^T r(x, u)dt \qquad st \qquad \dot{x} = f(x, u)$$

- The integrand of the **cost function** is called **reward.**
- The integral is the **value function** to be maximised
- The **control law** is usually referred to as **policy**
- Both the **policy** and the **reward** are **functions** (in the continuous case)

Hamilton-Jacobi-Bellman (HJB) equation

$$\mathcal{J}(x(t),t) = \max_u \int_t^{t+\Delta t} r(x,u)dt + \underbrace{\mathcal{J}(x(t+\Delta t), t+\Delta t)}_{\text{future value function}}$$

The principle of optimality requires that the **future value function** has to be **maximised** expanding in series and taking $\Delta t \to 0$

$$-\dot{\mathcal{J}}(x(t),t) = \max_u \{ r(x,u) + \nabla \mathcal{J}(x(t),t) f(x,u) \}$$

The **Hamilton-Jacobi-Bellman** eq. (1954) is a functional equation.
The solution is the **optimal policy** $u = \pi(x)$

HJB in discrete form

Discounted HJB

We introduce in the HJB the term $\gamma = e^{-\rho t}$ where $\rho > 0$ is the **discount factor**

$$\rho \mathcal{J}(x(t), t) = \max_u \{ r(x, u) + \nabla \mathcal{J}(x(t), t) f(x, u) \}$$

Discrete time observations

Bellman equation

$$\mathcal{J}(x_n) = \max_u \{ r(x, u) + \gamma \mathcal{J}(x_{n+1}) \}$$

Actor based RL

$$\mathcal{J}_\pi(x_n) = r(x, u) + \gamma \mathcal{J}_\pi(x_{n+1})$$

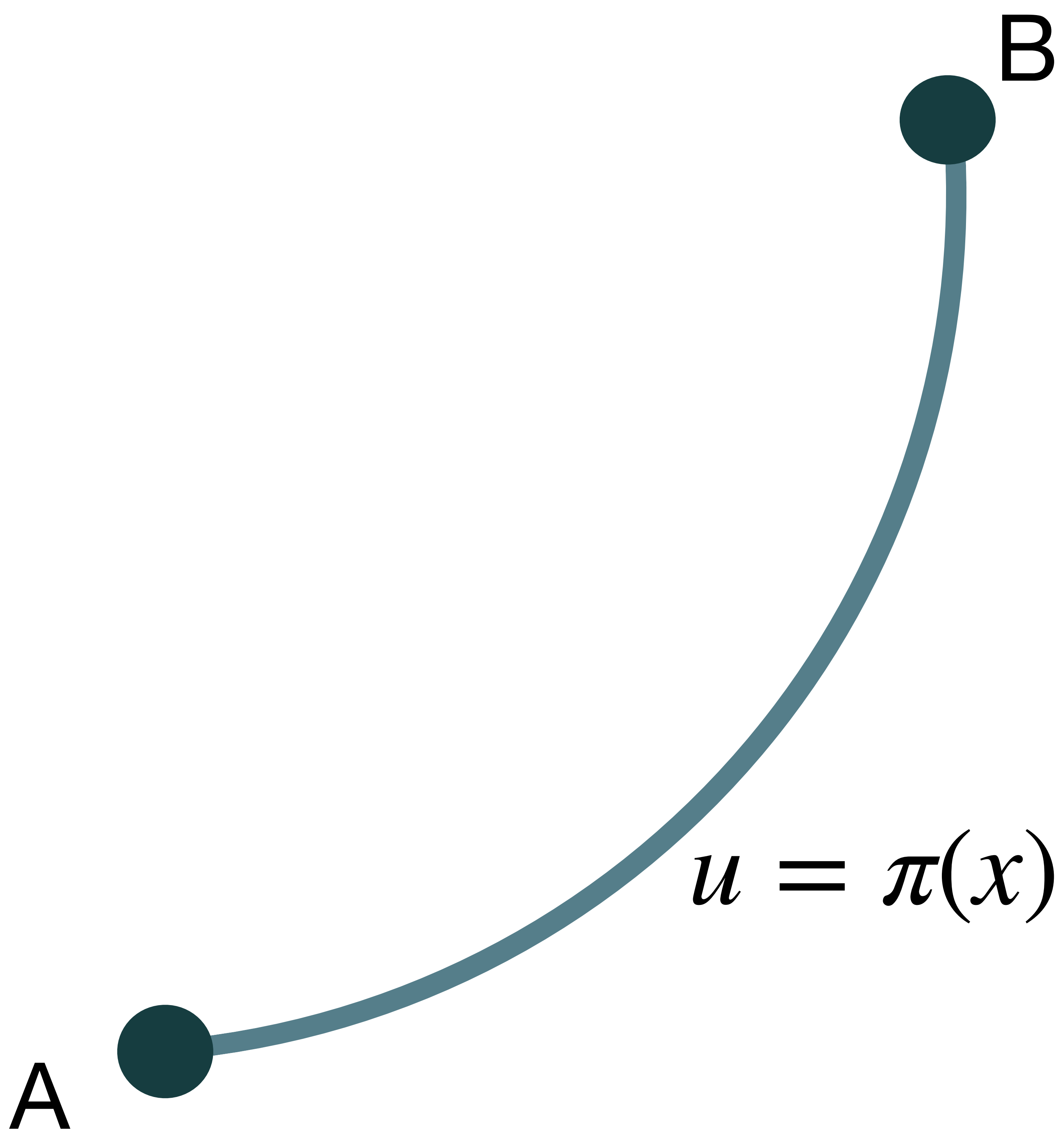
Value function

Actor-only methods (*reinforce algorithms*)

One **policy** is evaluated based on a long-time trajectory, and explored by perturbing it

An improved version of Monte-Carlo searching

Pontryagin maximum principle a **necessary** condition for optimality



Critic based RL

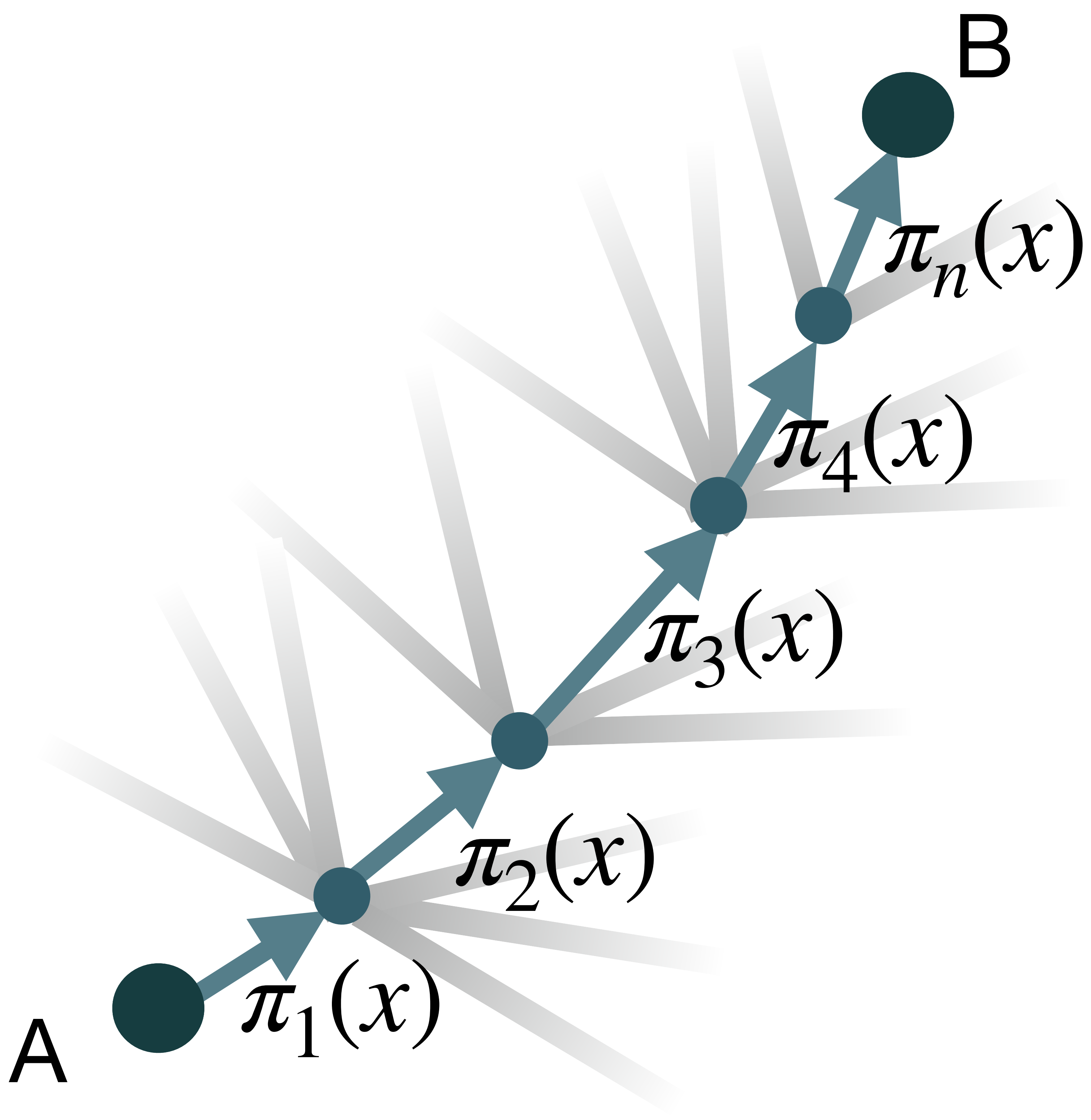
$$Q(x_n, u_n) = r(x, u) + \gamma Q(x_{n+1}, u_{n+1})$$

State-action, or **Q-function**

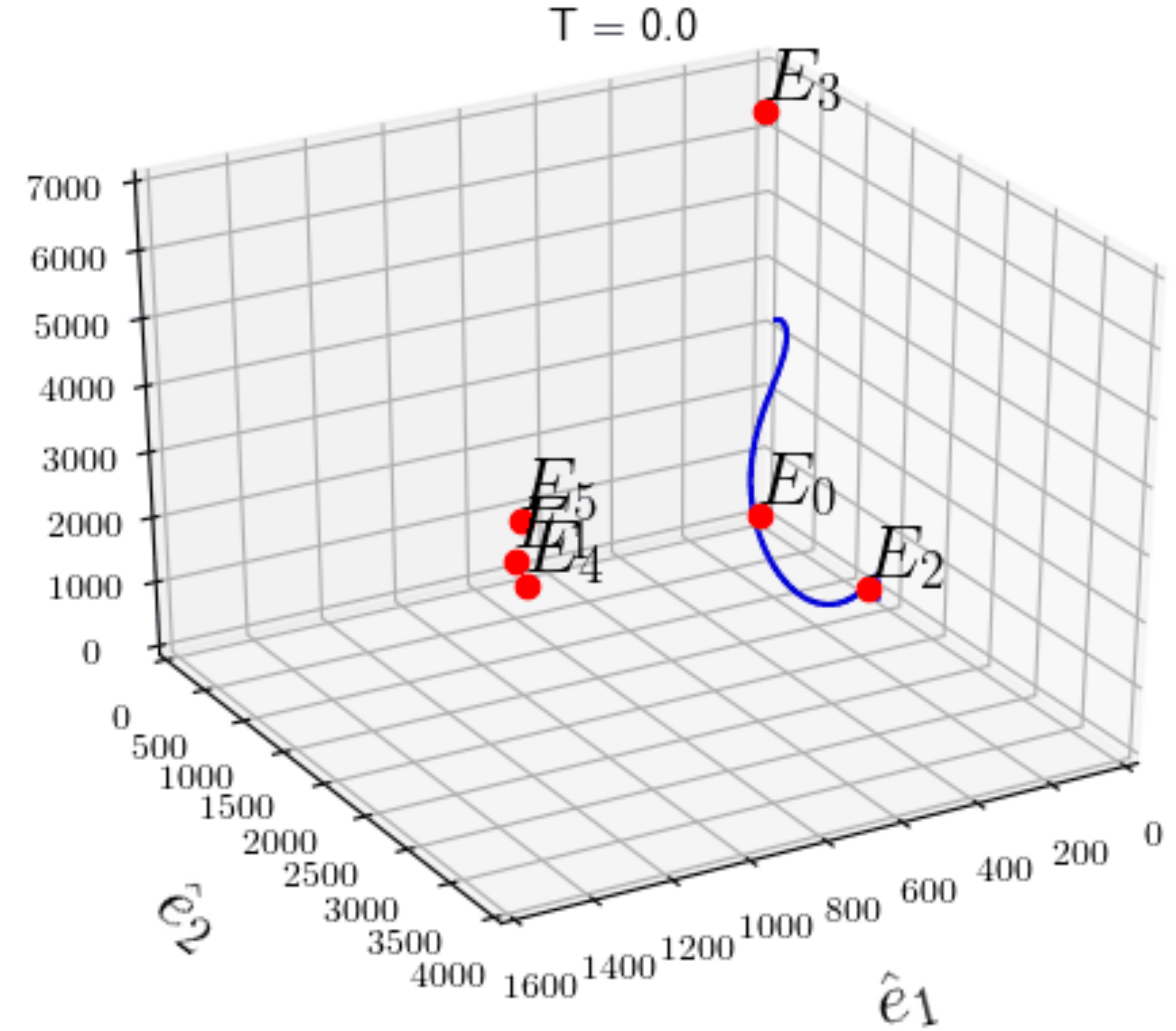
Critic-only methods (*"human-level" algorithms*)

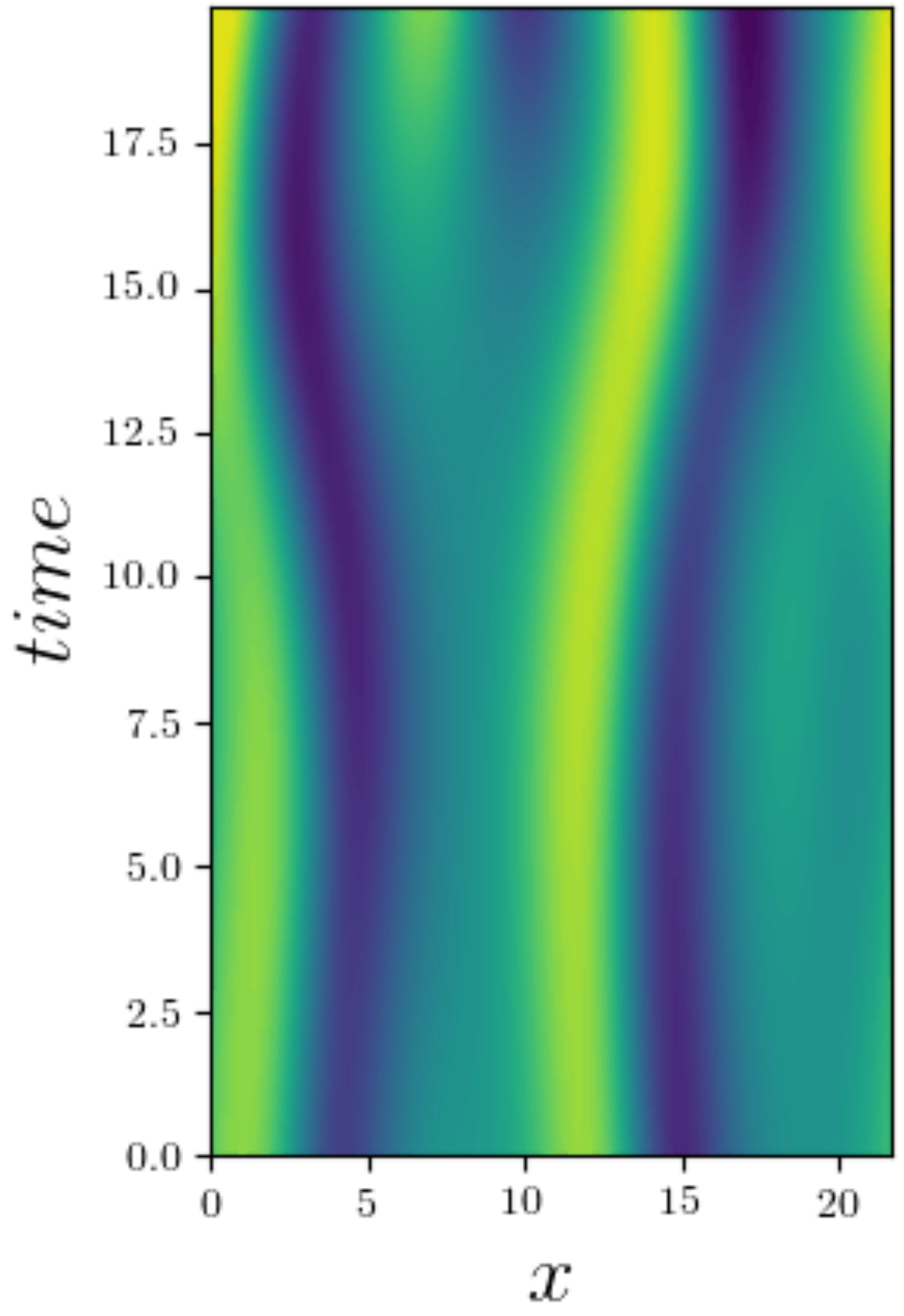
The discounted infinite-horizon optimal problem is decomposed in local optimal problems.

In principle, satisfies the optimal **Bellman equation** if the case is **Markovian**. It is a **sufficient** and **necessary** condition for optimality



KS system





Kuramoto-Sivashinsky (KS) equation models **diffusive instabilities** in flame front and **phase turbulence**.

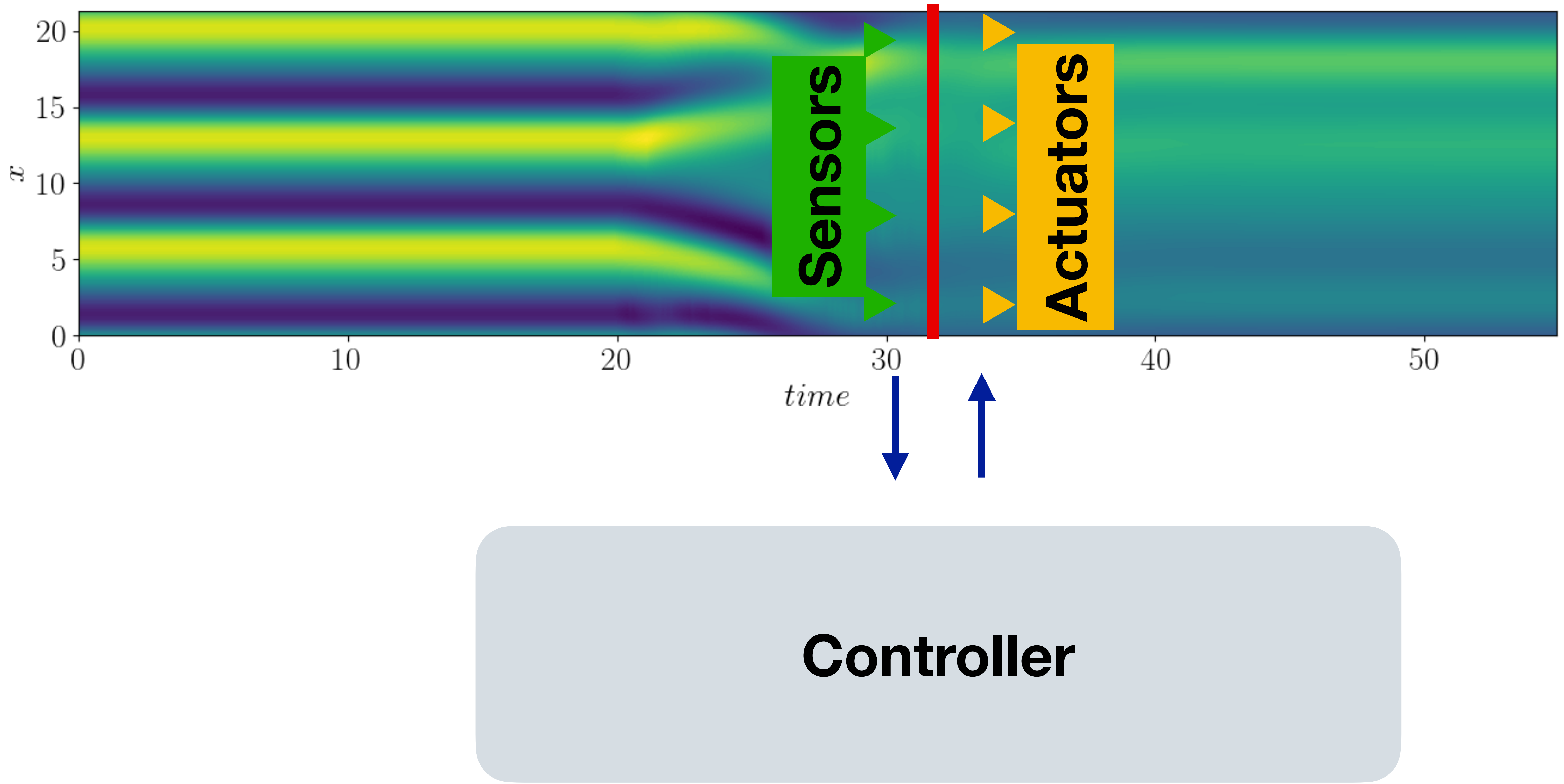
Different regimes: steady, periodic, chaos.

$$\frac{\partial x}{\partial t} = -\nabla^4 x - \nabla^2 x - \frac{1}{2}|\nabla x|^2$$

We consider a **chaotic case**, where the critical parameter (domain length)

Cvitanović, P., et al. (2010). SIAM Journal on Applied Dynamical Systems

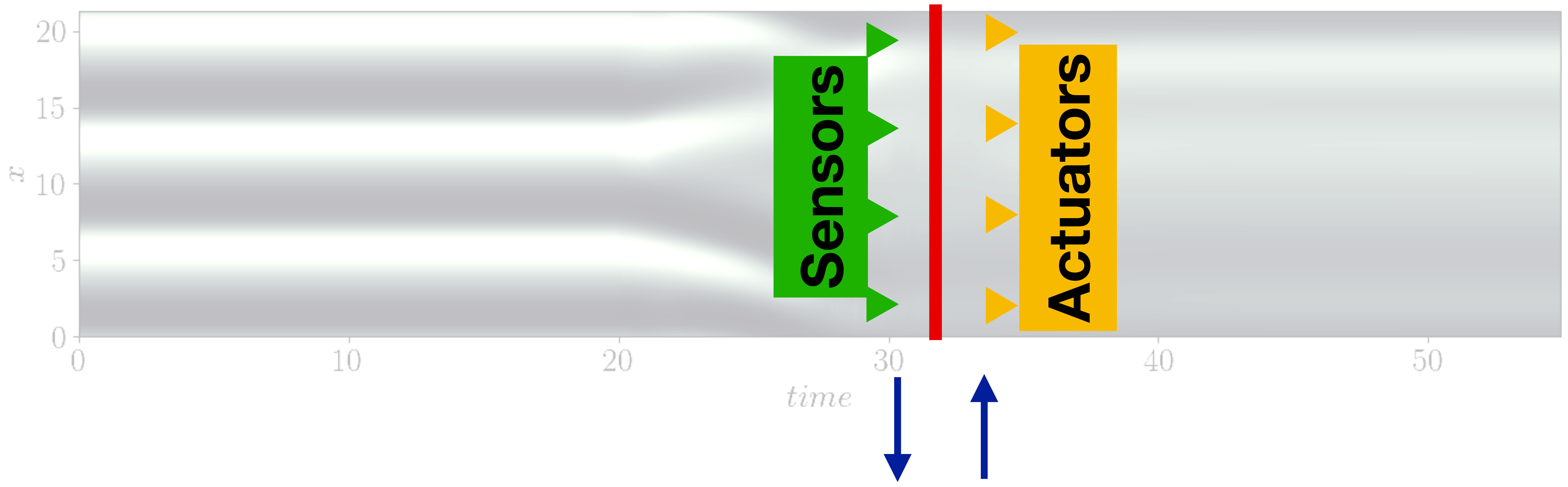
KS system as a plant



Plant setup
8 sensors (local measurements)
4 actuators (Gaussian body forcing)

$$\frac{\partial x}{\partial t} = -\nabla^4 x - \nabla^2 x - \frac{1}{2}|\nabla x|^2 + f$$

Deterministic Policy Gradient (DPG)



Actor Policy

$$u_t = \pi(x_t \,|\, \omega)$$

Deterministic Policy Gradient (DPG)

POMDP: Partial Observable Markov Decision Process

TD: Temporal difference

Sensors

Actor Policy

$$u_t = \pi(x_t \,|\, \omega)$$

Actuators

POMDP

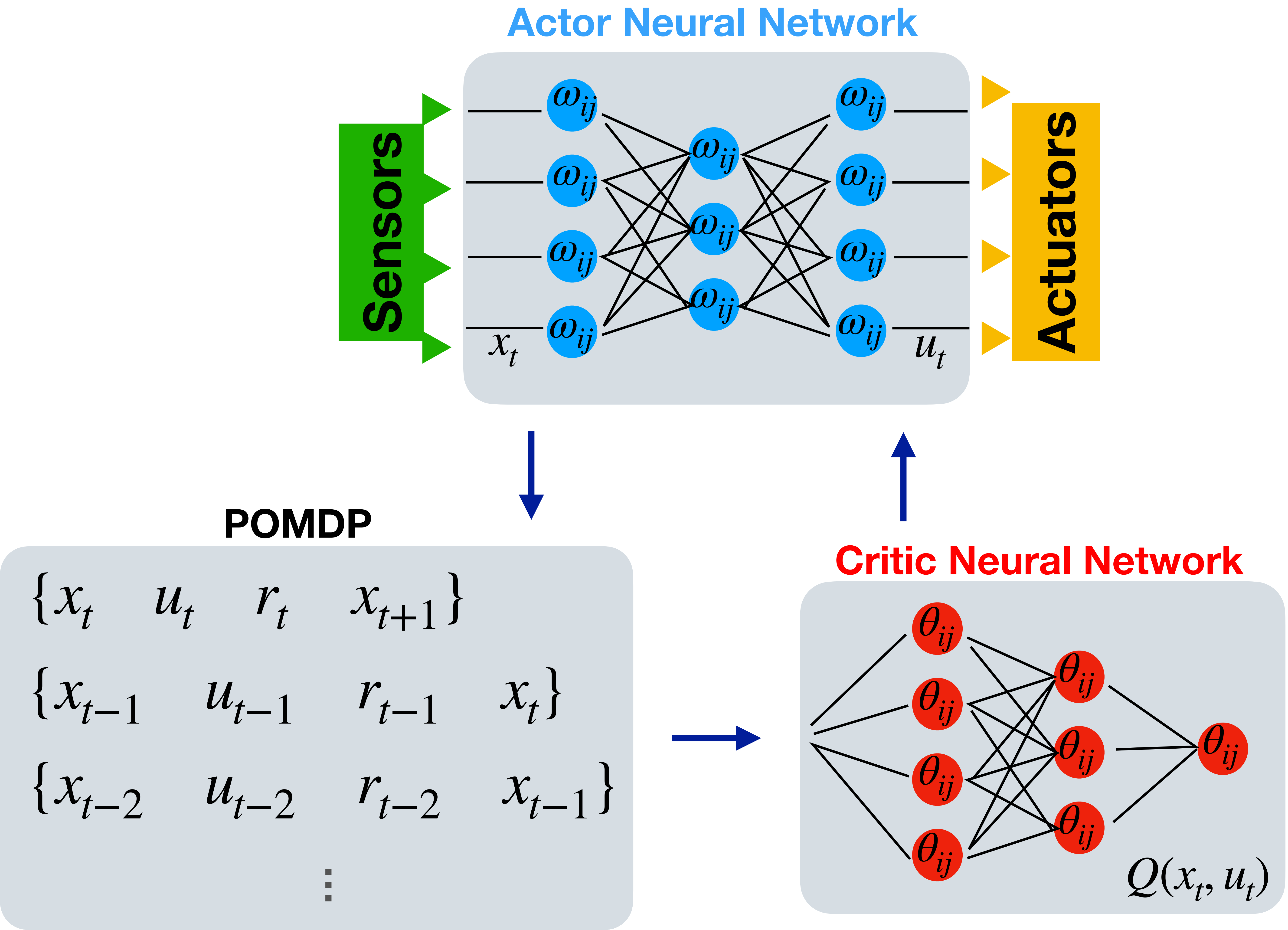
$x_t \quad u_t \quad r_t \quad x_{t+1}$

$\nabla_\omega Q$

Critic

$$TD := Q(x_t, u_t \,|\, \theta) - r_t - \gamma Q(x_{t+1}, \pi(x_{t+1}) \,|\, \theta) \longrightarrow \nabla_\theta TD \longrightarrow Q(x_t, u_t \,|\, \theta)$$

Deep DPG



Critic update

θ_{ij} are the weights of the **Critic Neural Network**

256 Neurons, 3 Layers

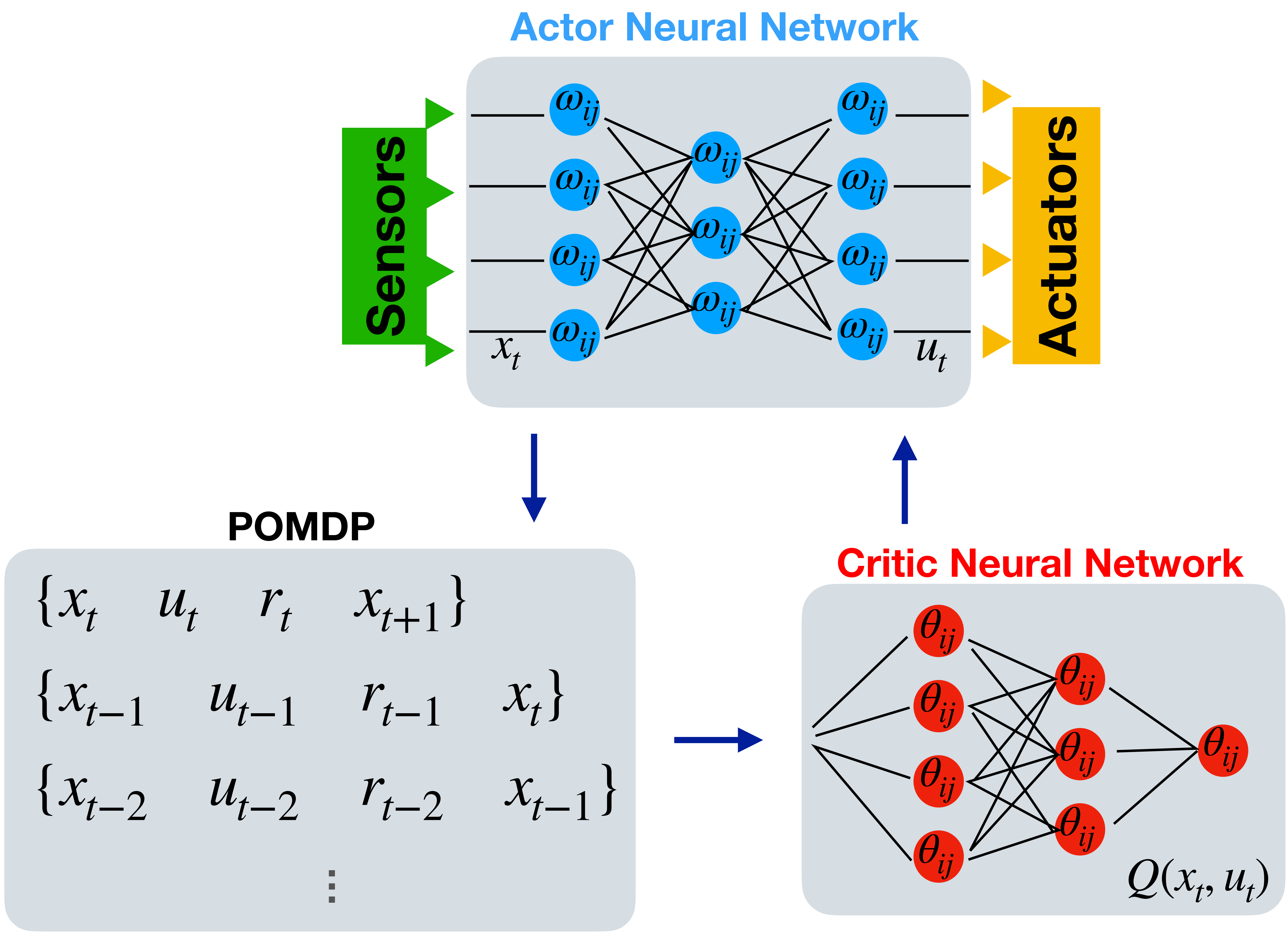
Actor update

ω_{ij} are the weights of the **Actor Neural Network**

128 Neurons, 3 Layers

Silver, D., et al. (2014, June). Deterministic policy gradient algorithms. In *ICML*.

Update of the models



Critic update

$$\nabla_\theta TD = \frac{\partial \| Q_t - (r_t + Q_{t+1}) \|}{\partial \theta_i}$$

$$\theta_i' = \theta_i - \alpha \nabla_\theta TD$$

Actor update

$$\nabla_\omega Q = \frac{\partial Q_t}{\partial u_t} \frac{\partial \pi_t}{\partial \omega_i}$$

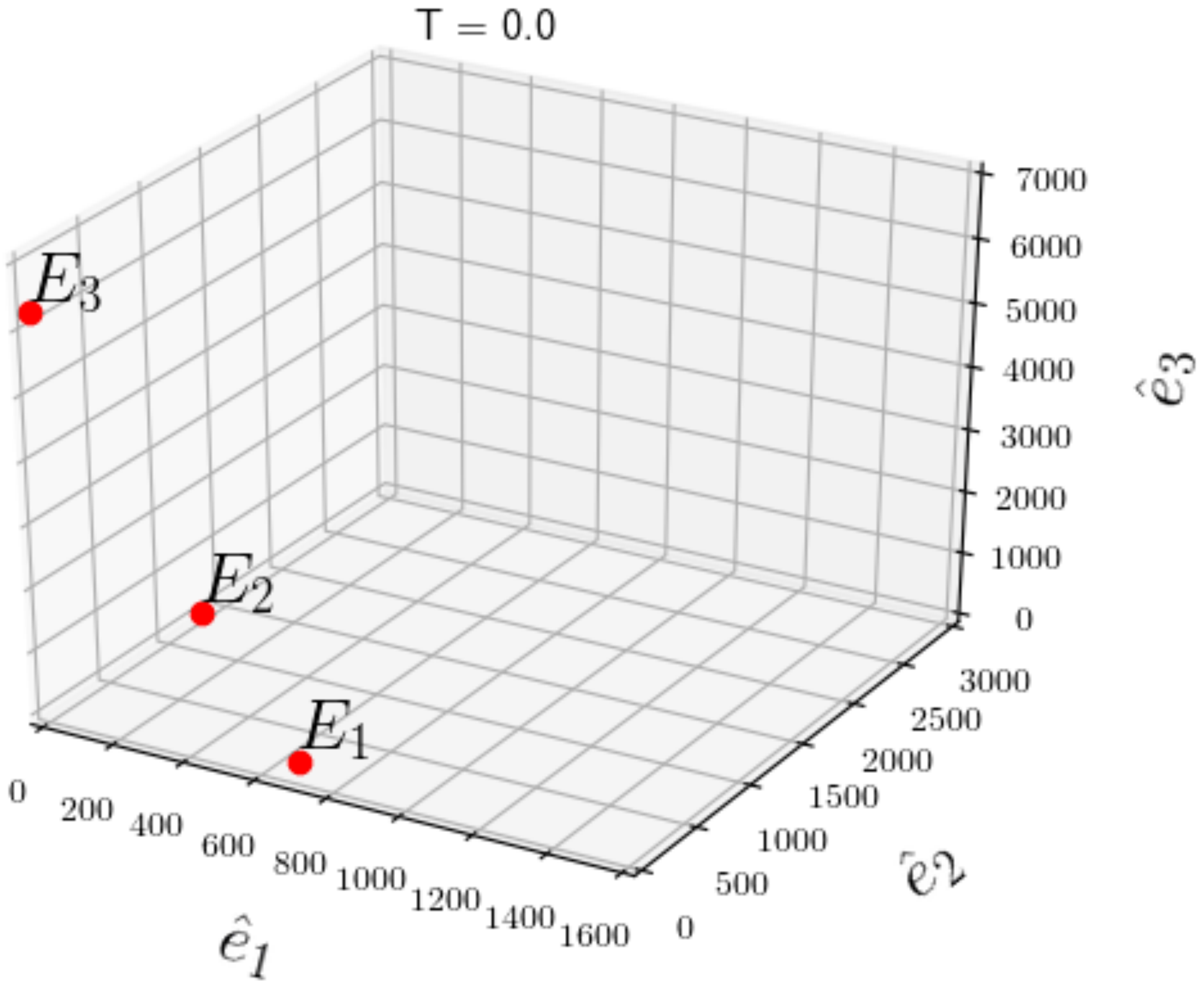
$$\omega_i' = \omega_i + \alpha \nabla_\omega Q$$

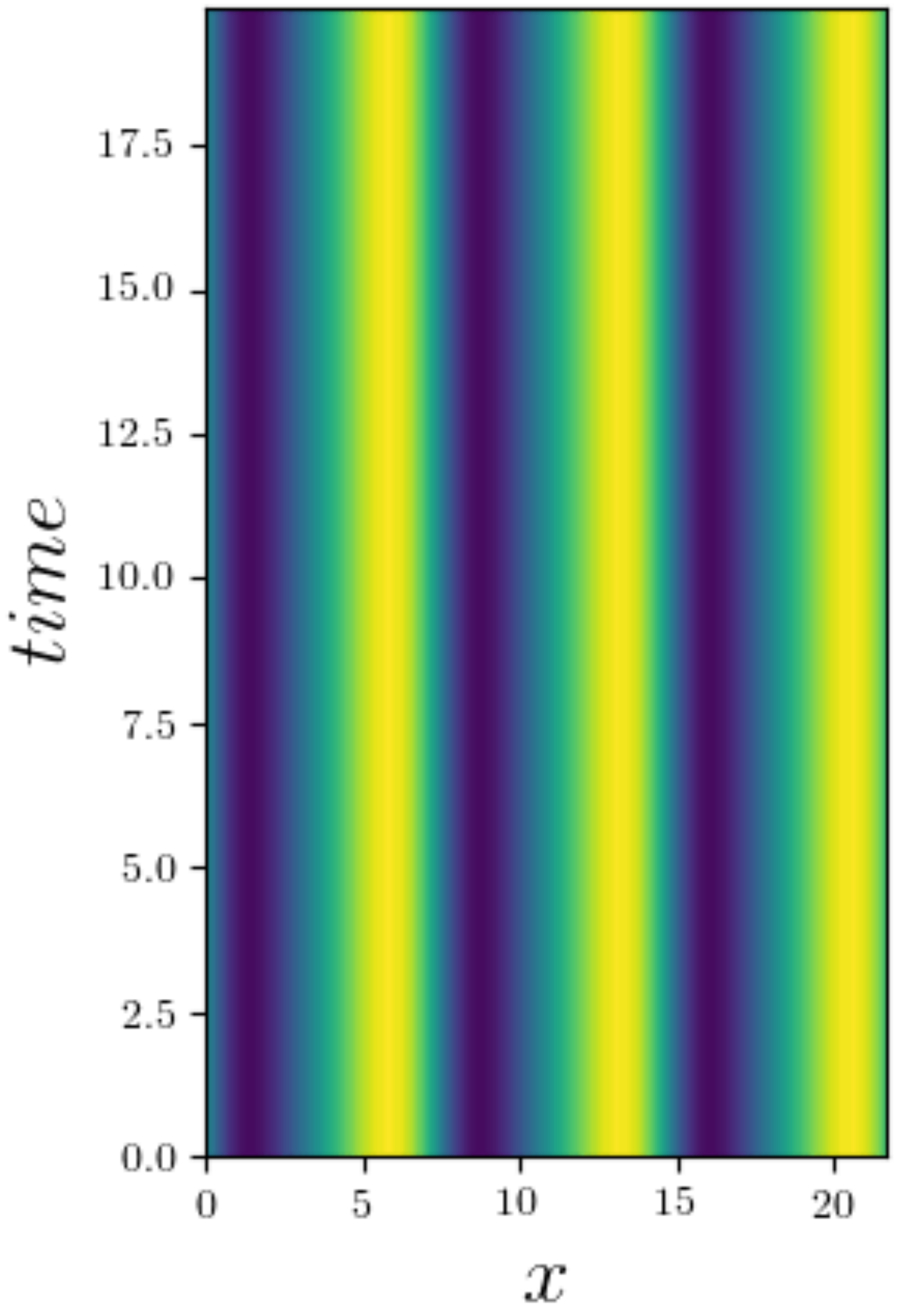
Learn from the observation

Explore the action-state space

Perturbation of the parameters of the policy, Ornstein-Uhlenbeck process

Controlled KS system





Actor-Critic algorithm: **deep deterministic policy gradient** (DDPG)

Three policies are identified

Reward defined with respect of the target states

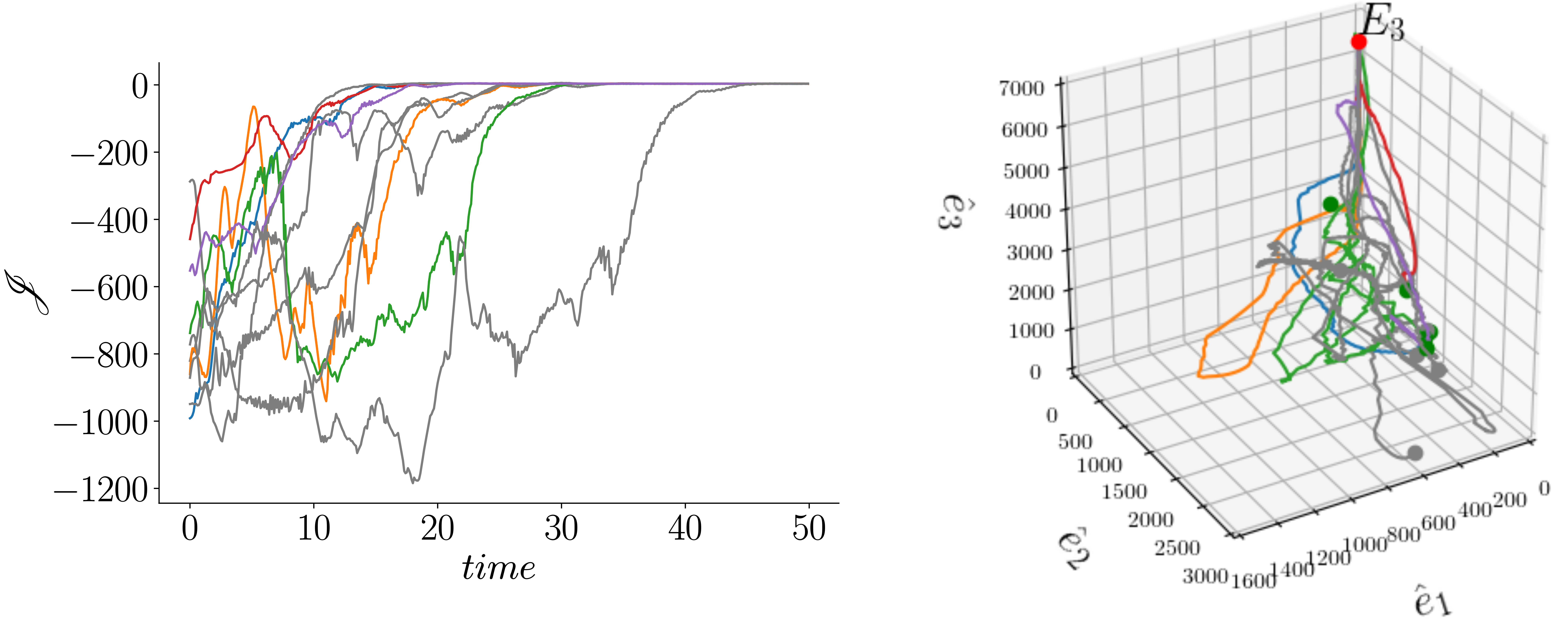
$$r_t = -\|x_t - x_{Target}\|$$

Discount factor $\gamma = 0.99$

Max time: **1 hour training** (Intel I3 CPU, 2015)

Silver, D., et al. (2014, June), In *ICML*
Bucci, M.A. et al. (2019), 10.1098/rspa.2019.0351

Controlled KS system



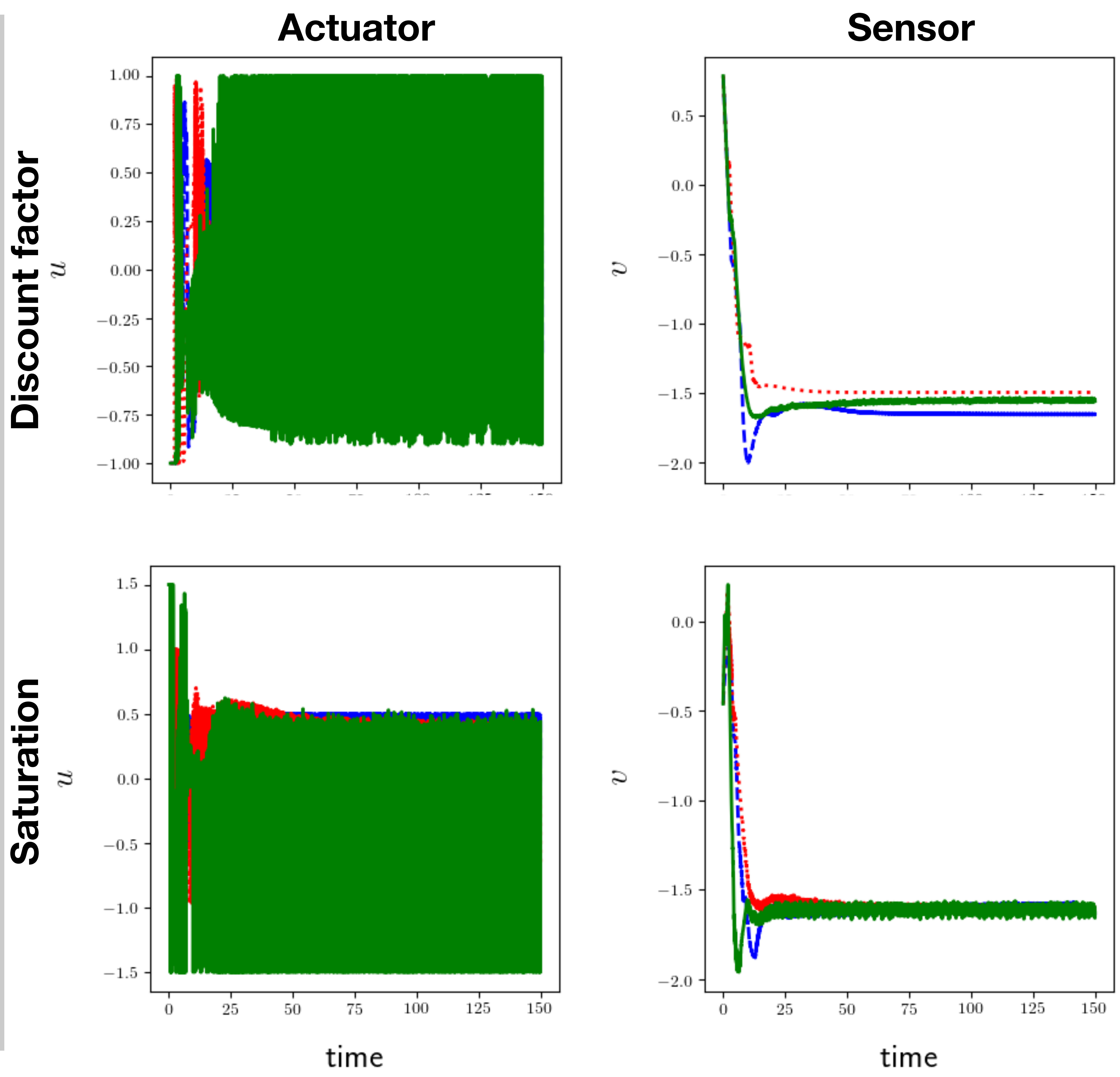
The identified policies are **robust** with respect of the initial conditions due to the **Markiovanity** of the **Q-function**, and the **exploration**

Controlled KS system

We identify complex control laws

Glass half full: Discovery of new, nontrivial control laws

Glass half empty: It is **not** the optimal solution obtained in KS by Riccati-based LQR.



Conclusions

Deep reinforcement learning is a powerful method for **non-linear control**

- **Full knowledge** of the system is **not required**
- **In principle**, the policy is a **global optimum** if the cost function is solution of the Bellman equation. **In practice this is not guaranteed.**
- Successfully tested on the KS chaotic system

More info? Click here!

—> **Application to NS equations**

Abstract: R01.00032 ***Control by Deep Reinforcement Learning of a separated flow*** by Thibaut Guegan et al.