Exploration Strategies in Reinforcement Learning

Maximum Entropy optimisation applied to chaotic PDE control

R. Hosseinkhan Boucher , O. Semeraro, L. Mathelin

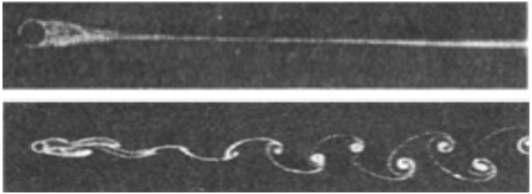
Laboratoire Interdisciplinaire des Sciences du Numérique, Université Paris-Saclay, CNRS

SIAM 2023
Amsterdam

Financed by Agence National de la Recherche (ANR) under project ANR-21-CE46-0008 Reinforcement Learning as Optimal control for Shear Flows (REASON)

Table of contents

Introduction: Real-world applications require robustness



Von Kármán vortex street in the wake of a cylinder with **Re=32** (top) and **Re=102** (bottom). **The adaptation of models to the evolution of the underlying dynamic is a property of robust models.**

Real-world applications require **robustness**

Origin of disturbances

- Noise
- Non-stationarity
- Stochasticity
- Partial Observability

Recent theoretical works about robustness in Reinforcement Learning [1]

So far applied on Robotics, what about PDE control ?

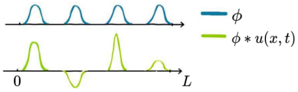
[1] B. Eysenbach, S. Levine. "Maximum Entropy RL (Provably) Solves Some Robust RL Problems", *International Conference on Learning Representations* (2022)

Controlled Kuramoto-Sivashinksy

Controlled KS: $\frac{\partial v}{\partial t}(x,t) + v(x,t)\frac{\partial v}{\partial x}(x,t) = -\frac{\partial^2 v}{\partial x^2}(x,t) - \frac{\partial^4 v}{\partial x^4}(x,t) + \phi(x) * \boldsymbol{u(t)}$

Equation is controlled through $\phi * \boldsymbol{u}$

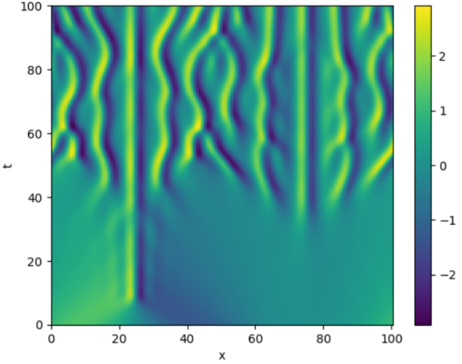
$v(x+L,t) = v(x,t)$ and $(x,t) \in [0,L] \times [0,T]$



ϕ is a **given** convolution kernel, $\boldsymbol{u}$ is the **unknown**

Properties

- Spatio-temporal chaos, 4th order non-linear
- Equilibria, relative equilibria, symmetries



Time evolution of the Kuramoto Sivashinsky equation with $L = 100$

Previous work

Extanding our previous work with *Deterministic Policy Gradient* [1]

[1] M. A. Bucci et al. "Control of Chaotic Systems by Deep Reinforcement Learning", *Proceedings of the Royal Society A* (2019)

Maximum Entropy Objective

Suppose u is a **stochastic control** with distribution $\pi(du)$

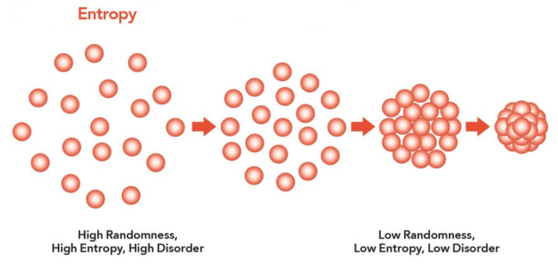
Quadratic Objective

$$J(u) = \int_0^T \left(\|v(x,t)\|^2 + \|u(x,t)\|^2\right) dt$$

Maximum Entropy Quadratic Objective

$$J(u) = \int_0^T \left(\|v(x,t)\|^2 + \|u(x,t)\|^2\right) dt - \alpha \mathcal{H}(\pi(du))$$

where $\mathcal{H}$ denotes the **Entropy**.



Question

What is the impact of considering the Maximum Entropy objective over the classic objective with Reinforcement Learning?

Modelling as Markov Decision Process

Controlled Dynamical System $\boldsymbol{x_{t+1} = G(x_t, u_t)}$, $u_t \in \mathcal{U}$, $x_t \in \mathcal{X}$.

Markov Decision Process representation

Consider G as a **stochastic process** $X_{t+1} = G(X_t, U_t)$

Transition Probability

$P((x_t, u_t), dx_{t+1})$ is a distribution over $\mathcal{X}$ given $(x_t, u_t) \in \mathcal{X} \times \mathcal{U}$

Example (Deterministic case)

$P((x_t, u_t), dx_{t+1}) = \delta_{G(x_t, u_t)}(dx_{t+1})$, the transition is determined by $\boldsymbol{G}$

Policy

$\pi(x, du)$ is a distribution over $\mathcal{U}$ given $x \in \mathcal{X}$

Deterministic PDE: randomness is induced by the control $\boldsymbol{U}$!

Standard Objective vs. Maximum Entropy Objective

Policy

$\pi(x, du)$ is a distribution over $\mathcal{U}$ given $x \in \mathcal{X}$

Example (Gaussian)

$\pi(x, du) \sim \mathcal{N}(\mu_x, \sigma_x^2)$

Cost-per-step

$c : \mathcal{X} \times \mathcal{A} \to \mathbb{R}_+$

Example (energy)

$c(x, a) = \|x\|^2 + \|u\|^2$

Standard Objective	Max Entropy Objective
$J_x^\pi := \mathbb{E}_x^\pi \left[\sum_{t=1}^\infty \gamma^t c\left(X_t, U_t\right)\right]$	$J_x^\pi := \mathbb{E}_x^\pi \left[\sum_{t=1}^\infty \gamma^t c\left(X_t, U_t\right)\right] - \alpha\mathcal{H}(\pi(x, du))$

Optimal policy

$\pi^* := \underset{\pi\in\Pi}{\arg\min}\ J^\pi$

Goal: find a policy π^* such that an objective is minimised

Functional approximation

Parametric Statistics

Distribution is parametrised by $\theta \in \Theta$, $\pi := \pi_\theta$

Objective $J^\pi := J^{\pi_\theta} = J(\theta)$

Optimal policy π_{θ^*} where $\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} J_\theta(x)$

Trajectories are sampled to estimate the process distribution

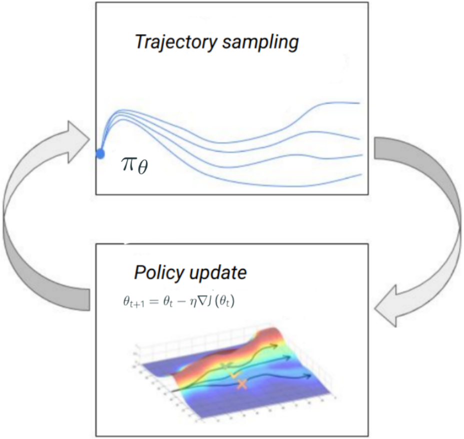
Monte Carlo method (Estimation)

$h^i = \left(x_1^i, u_1^i, x_2^i, \ldots, x_{T-1}^i, u_{T-1}^i, x_T^i\right)$

$J_x(\theta) = E_x^\pi\left[\sum_{t=0}^\infty \alpha^t c\left(x_t, a_t\right)\right] \simeq \frac{1}{N}\sum_{i=1}^N\left[\sum_{t=0}^\infty \alpha^t c\left(x_t^i, a_t^i\right)\right]$

Optimisation (Gradient Descent)

$\theta_{t+1} = \theta_t - \eta\nabla J\left(\theta_t\right)$



Reinforcement Learning: Kuramoto-Sivashinsky setting

State and control spaces

$$\mathcal{X} = L^2([0, L]) \simeq \mathbb{R}^d$$
$$\mathcal{U} = L^2([-a, a]) \simeq [-a, a]^b$$

with $d, b \in \mathbb{N}$, discretization dimensions (*e.g.* $d = 64$)

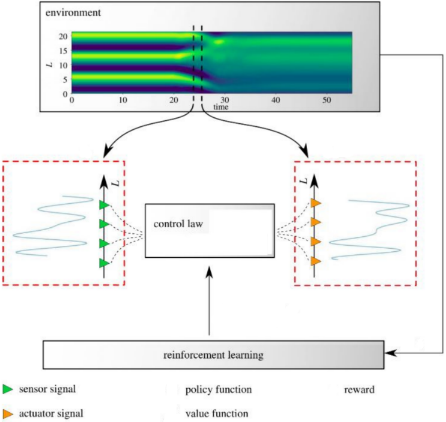
Cost is the energy of the system

$$c(x) = \lambda \|x\|_2^2 + \beta \|u\|_2^2$$

Control is a gaussian mixture weighted by $U_t \sim \pi(X_t, \cdot)$

$$\phi(x) * U(t) = \sum_{i=1}^{b} U_i \frac{1}{2\pi\sigma} \exp\left(-\frac{(x - x_i^{\mathrm{a}})^2}{2\sigma^2}\right)$$

System evolution: spatial discretisation with exponential time-differencing.



Experiments 1: Stabilising the dynamics

With spatial domain $x \in [0, 22]$, the PDE has 4 **steady-state solutions** $E_i(x),\ i = 0, \cdots, 3$

Task

Minimise $J_x^\pi :=$
$$\mathbb{E}_x^\pi \left[\sum_{t=1}^{\infty} \gamma^t c\left(X_t, U_t\right)\right] - \alpha \mathcal{H}(\pi(x, du))$$

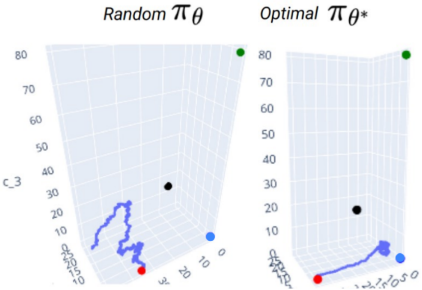
with $c(x) = \lambda \|x\|_2^2 + \beta \|u\|_2^2$

Configuration

Method Proximal Policy Optimisation[1] (PPO)

Time horizon $t \in [0, 20]$

Data 2000 trajectories from random initial conditions with shifting distribution



Fourier representation of time-independant solutions $E_i(x)$ with random (left) and optimal (right) controlled trajectories.
Representation of the equilibria E_0, E_1, E_2, E_3

[1] J. Schulman et al. "Proximal Policy Optimization Algorithms", arXiv preprint (2017).

Experiment 1: Stabilising the dynamics

Objective

Minimise $J_x^\pi := \mathbb{E}_x^\pi \left[\sum_{t=1}^{\infty} \gamma^t c\left(X_t, U_t\right)\right] - \alpha \mathcal{H}(\pi(x, du))$

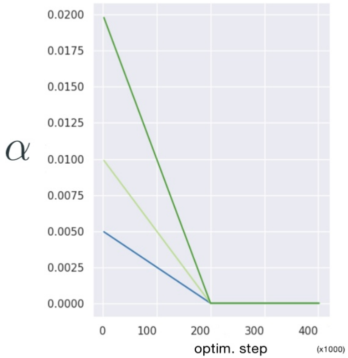
$$\text{with } c(x) = \lambda \|x\|_2^2 + \beta \|u\|_2^2$$

Random initial condition $X_0 \sim \mathcal{N}(E_2, \sigma^2)$

Control the state x_t towards the equlibrium $E_0 = 0$

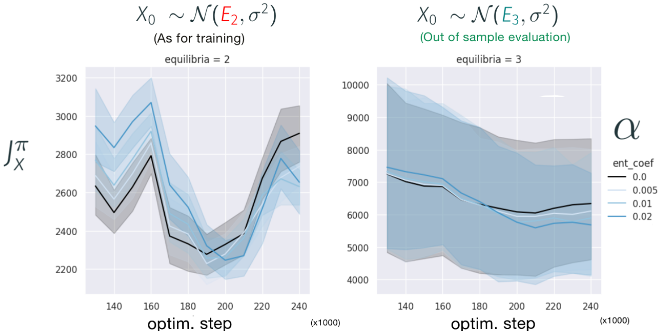
Experiment

- Fix 3 **different levels of entropy** α
- Optimise 10 seeds (decrease incertainty) for each of the α
- Entropy **linear decay** during optimisation
- Test policy on new initial condition distribution $X_0 \sim \mathcal{N}(E_3, \sigma^2)$



Decay of α during optimisation

Result 1: Maximising Entropy improves generalisation



Optimal $\mathbb{E}^\pi_x \left[\sum_{t=1}^\infty \gamma^t \|X_t\|^2 + \|U_t\|^2\right]$ for different levels of α

Black curve: $\alpha = 0$
Blue curves: $\alpha > 0$

Average over 10 models
for each of the α
(total 40 models θ^*)

Observations

- No-entropy objective converges **faster**
- Entropy improves **generalisation** performances (lower energy on **out of sample** distribution)

Experiment 2: Policy evaluation under noisy observations

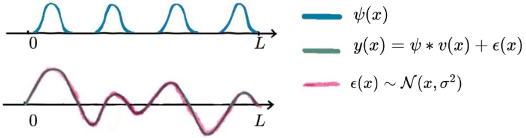
Controlled KS: $\frac{\partial v}{\partial t}(x,t) + v(x,t)\frac{\partial v}{\partial x}(x,t) = -\frac{\partial^2 v}{\partial x^2}(x,t) - \frac{\partial^4 v}{\partial x^4}(x,t) + \phi(x) * u(t)$

In practice: partial observability

PDE controlling term $\phi(y) * u(t)$

Noisy observable $y(x) = \psi * v(x) + \epsilon(x)$

Sensor noise $\epsilon(x) \sim \mathcal{N}(x, \sigma^2)$



Noisy observable

Hypothesis

Maximum entropy solutions are robust to noise

Observation noise decreases performances

Experiment

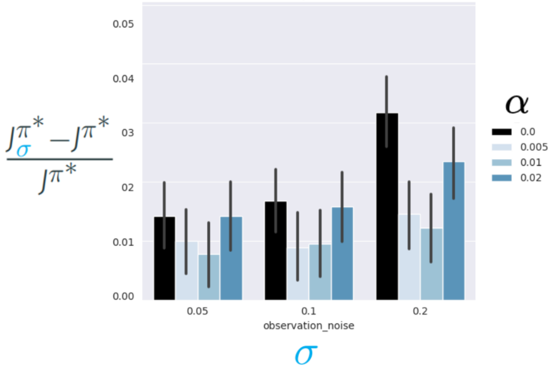
- *Test* policy with *different level of noise* σ on y
- Compare evolution of $J_\sigma^{\pi^*}$ w.r.t. J^{π^*}: $\frac{J_\sigma^{\pi^*} - J^{\pi^*}}{J^{\pi^*}}$

with $\boldsymbol{J^{\pi^*}} = \mathbb{E}_x^\pi\left[\sum_{t=1}^\infty \gamma^t \left\|\boldsymbol{X_t}\right\|^2 + \left\|\boldsymbol{U_t}\right\|^2\right]$

and $J_\sigma^{\pi^*}$ same quantity evaluated

with noisy observables

Result 2: Entropy improves noise robustness



Variation of the objective to minimise **after noise introduction**

Black bar: $\alpha = 0$
Blue bars: $\alpha > 0$

Average over 10 models
for each of the α
(total 40 models θ^*)

Noisy observable $y(x) = \psi * v(x) + \epsilon(x)$

Sensor noise $\epsilon(x) \sim \mathcal{N}(x, \sigma^2)$

Observations

- Noise introduction globally **increases** the cost function
- The classic objective is the more sensitive to noise (up to 3x.)
- Adding the entropy constraint α improves robustness

Conclusion: Entropy Objective defines a Robustness/Performance trade-off

Performance Penalised objective $\neq$ standard objective

Generalisation State space exploration

Robustness Noise introduction

Further work Model regularity properties (Lipschitz continuity),

Related References

- T. Haarnoja et al. "Reinforcement Learning with Deep Energy-Based Policies", *International Conference on Machine Learning* (2017)
- Z. Ahmed et al. "Understanding the Impact of Entropy on Policy Optimization", *International Conference on Machine Learning* (2019)
- B. Eysenbach, S. Levine. "Maximum Entropy RL (Provably) Solves Some Robust RL Problems", *International Conference on Learning Representations* (2022)

Modelling as Markov Decision Process

Controlled Dynamical System $x_{t+1} = G(x_t, u_t)$, $u_t \in \mathcal{U}$, $x_t \in \mathcal{X}$.

Markov Decision Process representation

Consider G as a **stochastic process** $X_{t+1} = G(X_t, U_t)$

Transition Probability

$P((x_t, u_t), dx_{t+1})$ is a distribution over $\mathcal{X}$ given $(x_t, u_t) \in \mathcal{X} \times \mathcal{U}$

Example (Deterministic case)

$P((x_t, u_t), dx_{t+1}) = \delta_{G(x_t, u_t)}(dx_{t+1})$

Policy

$\pi(x, du)$ is a distribution over $\mathcal{U}$ given $x \in \mathcal{X}$

Process Distribution

$$P^{\pi}\left(dx_0, du_0, dx_1, du_1 \ldots, dx_t\right) = \nu\left(dx_0\right) \pi\left(x_0,\ du_0\right) P\left(dx_2 \mid x_1,\ u_1\right) \pi\left(x_2,\ du_2\right) \cdots$$
$$\pi\left(x_{t-1},\ du_{t-1}\right) P\left(dx_t \mid x_{t-1},\ u_{t-1}\right)$$