DOI: 10.13009/EUCASS2022-7350
9TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)

Action-sparsity-seeking algorithms for active flow control using deep reinforcement learning

Romain Paris⋆†, *Samir Beneddine*⋆ *and Julien Dandois*⋆
⋆*ONERA*
DAAA, ONERA, Université Paris Saclay, F-92190 Meudon - France
romain.paris@onera.fr
†Corresponding author

Abstract

This study addresses the issue of actuation sparsity for flow control, through reinforcement-learning-based control laws. A generic actuator elimination method is proposed, starting from a fully active actuator layout and sequentially disabling the least needed actuators. Three implementations of this method, using different actuator ranking metrics are implemented and tested on two test-cases, the one-dimensional Kuramoto-Sivashinsky equation and a laminar bi-dimensional flow around an airfoil at $Re_c = 1000$. Results show that these metrics solve a trade-off between accuracy and computational cost differently and thus may be chosen with respect to the characteristics of the controlled environment.

1. Introduction

Flow control encompasses a spectrum of methods from passive control to closed-loop non-linear control, with a the goal of improving aerodynamic qualities of vehicles or flows. Passive control has long been a go-to strategy thanks to its simplicity and robustness (Bruneau & Mortazavi, 2008; Evans *et al.*, 2018; Joubert *et al.*, 2013; Seshagiri *et al.*, 2009), it is now challenged by active control, reaching overall better performances (Seifert *et al.*, 1993; Seifert & Pack, 1999), especially closed-loop control that enables precise and energy efficient control actions thanks to its error-correcting feedback and is thereby a well-documented and studied domain. System dimensionality, non-linear response and simulation cost being issues specific to fluid mechanics, a wide literature proposed adapted methods (Brunton & Noack, 2015) to design closed-loop control laws tackling these challenges.

In the last few years, machine learning, and deep learning in particular, has demonstrated remarkable performances in wide variety of fields. This progress is mainly due to the rise in accessible computing power, the use of neural networks (NN) that act as quasi-universal function approximators and their straightforward optimization methods relying on gradient back-propagation (LeCun *et al.*, 2015). Hence, a flourishing literature (Brunton *et al.*, 2020; Vinuesa & Brunton, 2021) investigating the potential of such techniques in fluid mechanics can be considered as a sign of strong interest by the community. Flow control also follows this trend of increasing integration of advanced machine learning methods, notably leveraging the paradigm of reinforcement learning (RL), based on the idea of trial-and-error. RL consists in evaluating a given control law (also called policy) on a target system (referred to as the environment). The evaluation data is then used to tweak the control law to maximize a given performance metric, with the underlying idea of promoting actions or strategies that are beneficial with respect to the metric. In the case of deep reinforcement learning (DRL), the policy is embodied by a neural network structure tasked with providing a control action given observed measurement. The exploration of new control strategies plays a decisive role in the performance of such methods.

Bucci *et al.* (2019) used Deep Deterministic Policy Gradient (DDPG) to steer the Kuramoto-Sivashinksy equation to its fixed points and Shimomura *et al.* (2020) used Deep Q-Networks (DQN) to optimize the frequency of plasma actuator bursts in order to control airfoil flow separation. Proximal Policy Optimisation (PPO) was used by Rabault *et al.* (2019) to control a low Reynolds number cylinder wake using surface-mounted jets. Wang *et al.* (2022) also resorted to PPO to control a low-Reynolds confined bi-dimensional airfoil flow using three suction-side synthetic jets in order to reduce drag. In the context of flow control, with the specificities mentioned earlier, the issue of sensor and actuator location or selection becomes critical. Having an efficient and parsimonious control setup motivates the emergence of all the following methods.

Multiple studies (Li & Zhang, 2022; Natarajan *et al.*, 2016) rely on adjoint sensitivity analysis (Chomaz, 2005) and on the "wavemaker", the overlap between the direct and adjoint sensitivity modes, introduced by Giannetti &

DOI: 10.13009/EUCASS2022-7350

Luchini (2007) to derive appropriate sensor and actuator placement. Sashittal & Bodony (2021) applied a related method on a data-driven, linearised model of their systems to position their sensors. Modelling the control system as a linear plant is also used by Bhattacharjee *et al.* (2018) who take advantage of the eigensystem realisation algorithm (ERA) to compare the controllability (in a $\mathcal{H}_2$ framework) of multiple jet actuators laid on the suction of an airfoil to select the best one depending on the performance criterion (lift or angle of attack upon flow separation).

Among non-linear actuator methods, one may cite the study of Rogers (2000) who derived a set of actuator layouts on a stealth bomber to satisfy manoeuvrability goals using a genetic algorithm. Paris *et al.* (2021) proposed a sensor selection algorithm using stochastic gates and leveraging the RL paradigm to filter out sensor measurement while preserving performance as much as possible.

In the current study we aim at introducing a reinforcement-learning-based method, re-using some ideas previously introduced in Paris *et al.* (2021) but this time to select actuators instead of sensors and compare and discuss multiple candidate metrics for actuator selection. The reinforcement learning algorithm along with both test cases is introduced in section 2. Section 3 then focuses on the proposed actuator elimination method before describing the three metrics discussed in the study. At last, the results of these metrics applied on both test cases are discussed in part 4.

2. Test-cases and base RL algorithm

2.1 Proximal Policy Optimisation with Covariance Matrix Adaptation

This study is based on RL-trained (closed-loop) control laws. As shown by figure 1 (left), the environment (the system to control) provides partial state observations s_t and a reward r_t, quantifying the instantaneous fitness of the current state with respect to a predefined performance metric. The environment receives forcing actions a_t altering its dynamics when stepping forward in time: $s_{t+1} \sim T(\cdot|s_t, a_t)$, T being the (stochastic) dynamics of state s. An agent is built in a closed-loop fashion in order to provide control action a_t based on observations following its policy π: $a_t \sim \pi(\cdot, s_t)$, embodying the control law. It is also tasked with implementing the training algorithm, that uses the collected data samples (s_t, a_t, r_t) to tune the parameters of policy π in order to maximise the expected return $R_t = \mathbb{E}_{s_t \sim T, a_t \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, $\gamma \in [0, 1]$ being an actualisation scalar parameter.

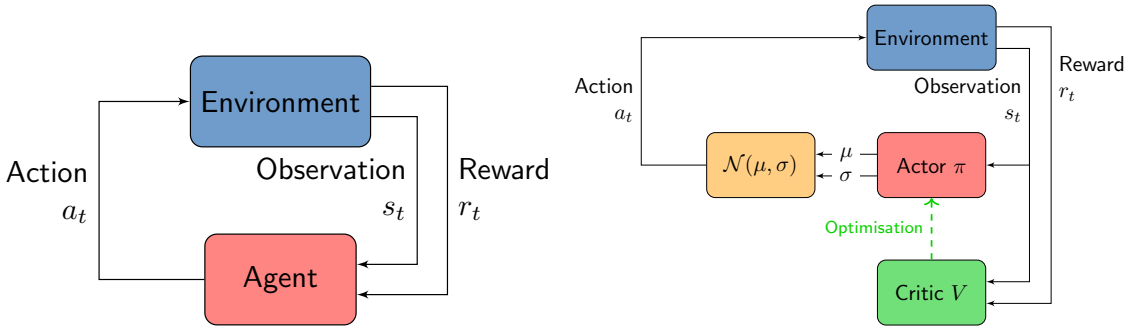

Figure 1: (left) The reinforcement learning feedback loop. (right) PPO-CMA agent structure: compared to PPO, the actor has one extra output σ which allows for a dynamic adaptation of the exploration.

Proximal Policy Optimization with Covariance Matrix Adaptation (Hämäläinen *et al.*, 2020) (PPO-CMA) is the training algorithm used here. It is a derivative version of the well-known Proximal Policy Optimization (Schulman *et al.*, 2017) (PPO). As summarized by figure 1 (right), it relies on a bicephalous neural-network structure, an actor (the policy π) and a critic (V). The critic is trained to output an estimate $V_\phi(s_t)$ of the value $V^\pi(s_t)$ of the currently observed partial state, where ϕ are the parameters of the critic neural network. This value is computed as the expected return $R_t = \mathbb{E}_{s_\tau \sim T, a_\tau \sim \pi}\left[\sum_{\tau=t}^{\infty} \gamma^\tau r_\tau\right]$ previously introduced, the expected actualized sum of the reward under the current policy π.

2.2 The 1D Kuramoto-Sivashinsky equation

The first test case considered is the control of the one-dimensional Kuramoto-Sivashinsky (KS) equation. This case is interesting because it is computationally inexpensive, allowing to perform a brute-force search of the optimal actuator layouts, yet complex when analyzing the obtained optimal layouts. The KS equation is a well-studied fourth-order partial differential equation exhibiting a chaotic behavior and describing the unstable evolution of flame fronts

DOI: 10.13009/EUCASS2022-7350

(Sivashinsky, 1980). On a periodic domain of length $L = 22$, the KS equation reads:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = a, \qquad \forall\, t : u(0,t) = u(L,t), \tag{1}$$

where a is the control action forcing later described. For $L = 22$, the KS equation exhibits three fixed points (named E1, E2 and E3) and low-dimensional instabilities similarly to some low-Reynolds number Navier-Stokes flows, as illustrated by figure 2.

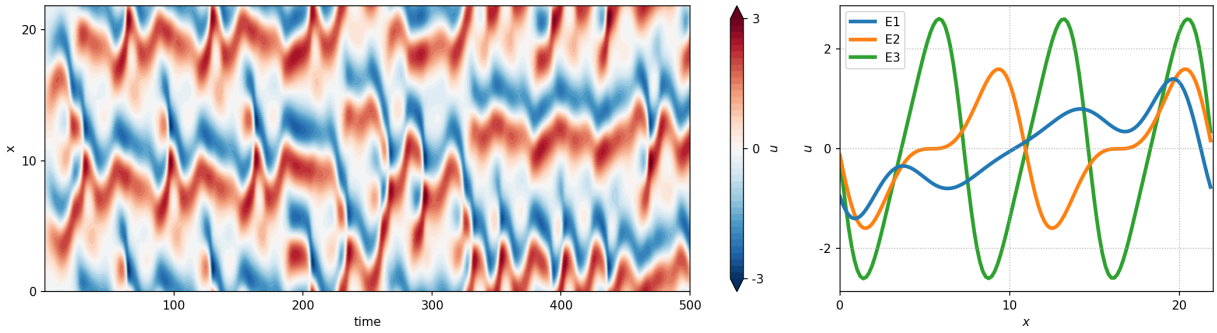

Figure 2: (left) Spatio-temporal representation of the dynamics of the KS equation on 500 non-dimensional time units, corresponding to 2000 control steps. (right) Shape of the three fixed points of the KS equation.

The numerical setup is based on the work of Bucci *et al.* (2019) and a code from pyKS with a time-step of 0.05. The control term is also designed to mimic spatially localized Gaussian forcing actions:

$$a(x,t) = \sum_{i=0}^{n-1} a_i(t) \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - x_i^{act})^2}{2\sigma^2}\right),$$

where n is the number of control actions, x_i^{act} $i \in \{0, ..., n-1\}$ the locations of the centers of Gaussian kernels and a_i the amplitude of each forcing implemented around x_i. The forcing action has 8 forcing components implemented at locations $(x_i^{act} \in \{0, 1, ..., 7\}L/8)$, $a_i \in [-0.5, 0.5]$ and $\sigma = 0.4$. The partial state observations are provided by measurements of u interspersed between control action locations so that $x_i^{obs} \in \{1, 3, 5, 7, 9, 11, 13, 15\}L/16$. A control step is made of an update of a_t, then 5 time-steps and the measurement of the observations and reward. A run of the KS equation lasts for 500 control steps. The reset state is seeded using a Gaussian noise and ran for a random number of control steps without control action, so that control starts on a fully developed instability. The aim of the control is to stabilize u around the fixed point E_1, and therefore the reward r_t is defined as:

$$MSE_t = \|u(\cdot,t) - u_{E1}\|_2 = \sqrt{\frac{1}{L}\int_0^L (u(x,t) - u_{E1})^2\, dx},$$
$$r_t = \frac{MSE_t - MSE_{ref}}{|MSE_{ref}|} - 0.1\|\underline{a}_t\|^2$$

where u_{E1} describes the fixed point E_1 (refer to figure 2 (right)), MSE_{ref} is the time-averaged reference mean squared error of the uncontrolled state and $\underline{a}_t$ is the control action at time t. This way r_t ranges over $]-\infty, 1]$.

2.3 2D flow around a stall NACA airfoil

The second considered test case is a bi-dimensional flow around a stalled NACA 0012 at a chord-wise Reynolds number $Re_c = 1000$. Airfoil flow separation control (Wu *et al.*, 1998) has been a matter of interest for a long time, with studies using a wide variety of control methods at various flow and stall regimes (Amitay & Glezer, 2002; Seifert *et al.*, 1996; Shimomura *et al.*, 2017; Yeh & Taira, 2019).

As illustrated by figure 3, the computational domain is "C-shaped" and built in the reference frame of the airfoil, meaning that the angle of attack (α) is imposed by the upstream flow conditions. The free-stream flow is uniform at $M_\infty = 0.1$. In the following, all quantities are made non-dimensional by the characteristic length C, the inflow density ρ_∞, the velocity U_∞ and the static temperature T_∞. The flow solution is computed via direct numerical solving using

DOI: 10.13009/EUCASS2022-7350

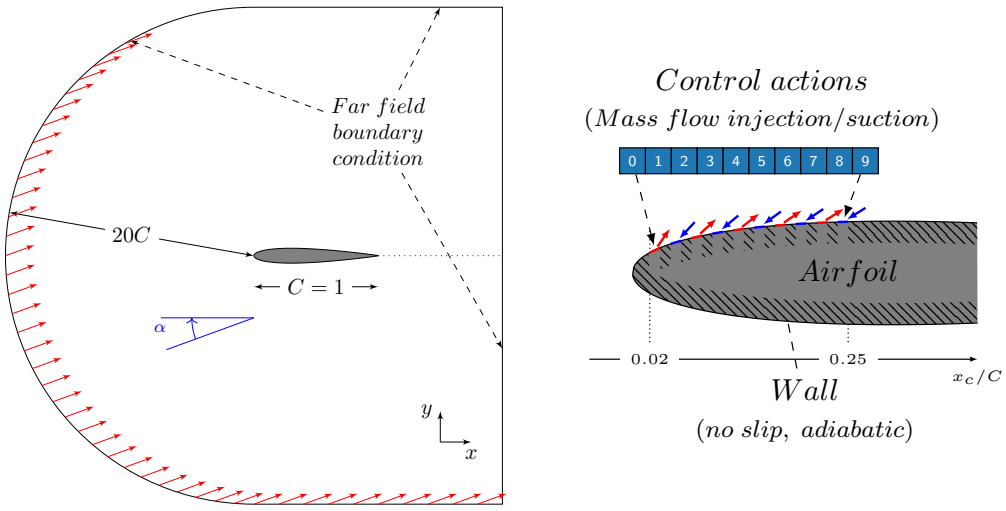

Figure 3: (left) Flow domain geometry, not at true scale. α denotes the angle of attack and C is the (unitary) chord length. (right) Boundary conditions on the airfoil, with $n_{act} = 10$. Blue numbered boxes symbolize actuators, number 0 being the upstream-most actuator, number 9 being the downstream-most one.

ONERA's FastS finite volume method solver (Dandois *et al.*, 2018), with a second-order-accurate AUSM+(P) scheme (Edwards & Liou, 1998) and a second-order implicit Euler time scheme ($dt = 1.3 \times 10^{-3}$). The structured mesh is made of 120,000 nodes and refined in the vicinity of the airfoil and in its wake are properly resolved.

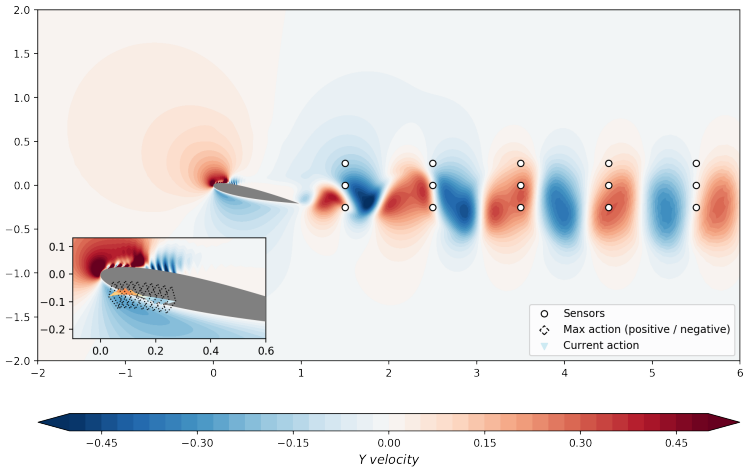

Figure 4: Instantaneous Y velocity flow field, with an arbitrary control action (here with $n_{act} = 20$), in the **free-stream reference frame**. White dots represent the sensor locations. The colored triangles nearby the airfoil depict the action, their heights and colors representing each action amplitude. The dashed diamond shapes mark off maximum actions (both positive and negative). The strong variations in velocity in the vicinity of the actuators are due to the presence of interspersed wall boundary conditions in-between actuators.

The control step ($\Delta t = 58\, dt = 7.9 \times 10^{-2}$ time units $\approx 1/50$ vortex shedding period) is chosen to both discretize the observation signal properly (avoiding aliasing) and to keep the "impact horizon" of a given action within a relatively short-term future for the agent (generally < 100 control steps). Control action is performed on the airfoil suction side through a series of n_{act} independent jet inlets (refer to figure 3 (right)). Negative control actions correspond to suction and positive to blowing at an angle of $-80°$ with respect to the local wall normal. The control action command ranges $[-2, 2]^{n_{act}}$ and a 52-iteration interpolation ramp between previous and current action is used in order to avoid abrupt changes that may not be handled by the numerical solver, in a similar fashion as Paris *et al.* (2021) and Rabault *et al.* (2019) did. Figure 4 illustrates a standard setup for this case. Both drag and lift coefficients (C_d and C_l) are computed by integration around the airfoil on a closed circulation, in the presence of actuators.

In this study, the angle of attack α is set to 15 degrees. At $Re_c = 1000$, the flow is unsteady and displays a

DOI: 10.13009/EUCASS2022-7350

laminar vortex shedding (Wu *et al.*, 1998). This instability causes both lift and drag coefficients to vary periodically, yielding undesired alternated loads on the airfoil. For any angle of attack α, one can define the characteristic period $T = 1/St(\alpha)$ of the unstable phenomenon. This time unit is later used in the study to size the control step. The main goal of the controller is to minimize lift fluctuations using as little control power as possible. Thus, the reward r_t is defined as:

$$r_t = -S\,(C_l)_{2T} - S\,(C_d)_{2T} - 0.05\frac{1}{n_{act}}\sum_{i=0}^{n_{act}-1}|\langle a_i\rangle_{2T}|,$$

where $S\,(C_l)_{2T}$ and $S\,(C_d)_{2T}$ are the standard deviation of the lift and drag coefficients computed over two characteristic periods and $|\langle a_i\rangle_{2T}|$ the absolute value of the averaged i^{th} action component also over 2 periods.

3. Actuator ranking

As stated in the introduction we aim at tackling the specific issue of optimized actuator location. The approach used here is to start from fully active actuator layouts (in the sense that all actuators are allowed to act) and to proceed to a successive elimination of the least "useful" ones until a prescribed number of remaining ones is reached. The previously introduced RL paradigm is well-suited for non-brute-force search of reduced actuator layouts, with the aim of preserving the control performance as much as much possible. Hence, it allows to derive non-linear (and potentially very efficient) control laws via a gradual optimization of the policy that takes advantage of the plasticity of neural networks.

3.1 A generic gating mechanism and procedure

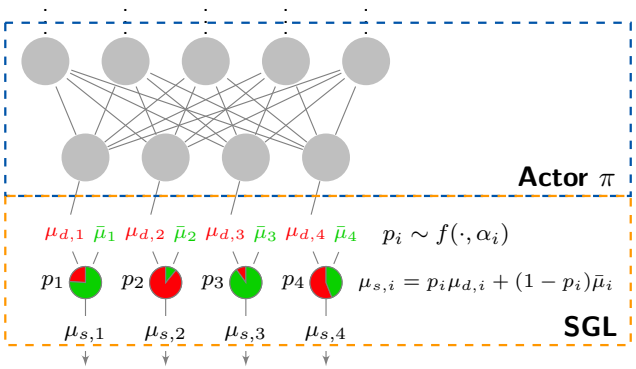

Figure 5: Structure of the Stochastic Gated Layer (SGL) used to filter the actor output. Here μ_d and μ_s respectively stand for μ_{dense} and μ_{sparse}.

The proposed methods rely on a common masking mechanism clipping action components downstream the policy. If $a_{dense} \sim \pi_{dense}$ is the stochastic action provided by the actor, the aim is to learn a binary mask $\underline{p} \in \{0,1\}^{n_{act}}$ reducing the number of non-null actions components to a prescribed amount. As illustrated by figure 5, where the gated (or clipped) action a_{sparse} can be defined as:

$$\underline{\mu}_{sparse} = \underline{p} \odot \underline{\mu}_{dense} + (1-\underline{p}) \odot \underline{\bar{\mu}}$$
$$\text{and } \underline{a}_{sparse} = \mathcal{N}\left(\underline{\mu}_{sparse}, \underline{\sigma}\right),$$

with $\odot$ being the scalar dot product, and $\bar{\mu}$ being substitution values (in our case $\bar{\mu}$ is a null vector). This structure is a simplified version of the Stochastic Gating Layer (SGL) by Louizos *et al.* (2018). $\underline{p}$ aims at being a deterministic, binary-valued vector, yet during training, this vector is sampled so that:

$$p_i = f(\alpha_i, u_i) = \begin{cases} 1, & \text{if } u_i \geq \alpha_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

DOI: 10.13009/EUCASS2022-7350

where $\underline{u} \in [0,1]^{n_{act}}$ is a random vector and $\underline{\alpha}$ is a trainable vector that sets the probability for the gate to be open, as shown in figure 5. The aim is then to derive a relevant loss $\mathcal{L}_{SGL}$, updating $\underline{\alpha}$ by a gradient descent method in order to reach the prescribed number of clipped action components. The considered loss aims at balancing two terms, the first promoting the opening of the most "useful" gate components, the second encouraging gate closing:

$$\begin{aligned}\mathcal{L}_{SGL} &= -\sum_{i=1}^{n_{act}} \mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right]\alpha_i + \lambda \sum_{i=1}^{n_{act}} P(p_i > 0) \\ &= \underline{\alpha} \odot \sum_{i=1}^{n_{act}} \left(-\mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right] + \lambda\right)\underline{e_i}\end{aligned}$$

where λ is a scalar weighting the penalization term, $\underline{e_i}$ the i^{th} standard unit vector of $\mathbb{R}^{n_{act}}$ and $\mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right]$ is a measure of the "usefulness" of the i^{th} action component for control. Its definition is discussed in the following sections. This way, λ is algorithmically scheduled to control the number of gates to close to reach a given user-prescribed number of actuators n_{target}. Indeed, the gradient of the loss with respect to each gate parameter α_i reads:

$$\nabla_{\alpha_i}\mathcal{L}_{SGL} = \lambda - \mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right].$$

Following a classical gradient minimization, α_i grows if $\nabla_{\alpha_i}\mathcal{L}_{SGL}$ is negative and decreases otherwise, consequently opening or closing the corresponding gate.

The action-sparsity-seeking algorithm training phases unwind as follows:

1. A first standard training phase where the policy is trained with all its action components (SGL gates are kept "open" during this phase), until a steady (maximum) performance is reached.
2. A second phase where all metrics $\mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right]$ (whatever their definition) are learned or estimated. SGL gates are still being kept frozen in this phase.
3. A third phase where, roll-outs are only performed to update the SGL α_i values using $\mathcal{L}_{SGL}$ computed thanks to frozen values of $\mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right]$. π and V are trained alongside $\underline{\alpha}$ in order to adapt the policy and its value estimation to the effective modification of $\underline{\alpha}$.
4. Restart to step 1 until the prescribed number of actuators is reached.

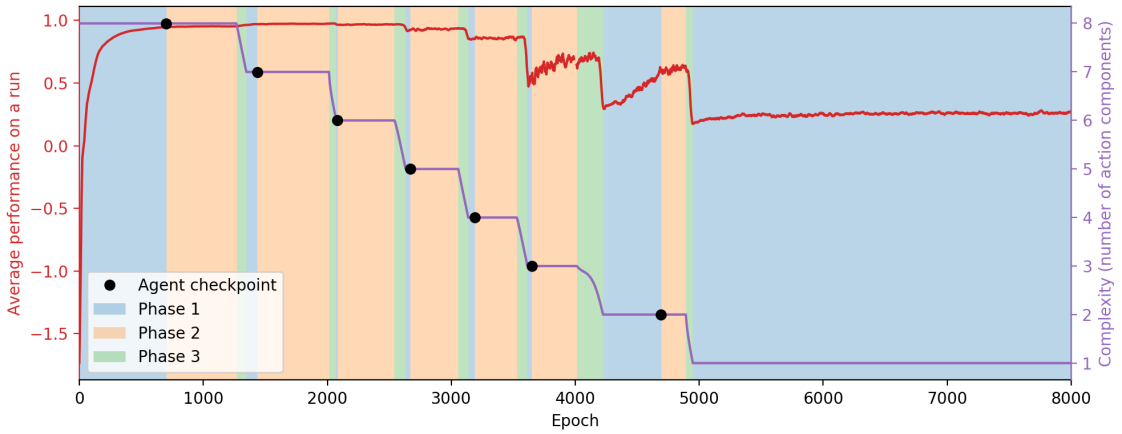

Figure 6: Example of the one-by-one elimination strategy drawn from a Kuramoto-Savishinsky test case (section 2.2). The performance curve (red line) has been smoothed using rolling average of length 20 for the sake of readability. Shaded areas in the background illustrate learning phases. Black dots represent the evaluation runs and the checkpoint back-ups of the agent performed during training.

Figure 6 provides an illustrative example of the elimination process. One can first notice that phases 1, 2 and 3 are of varying lengths (in number of training epochs) depending on the current number of action components (as called complexity). This is due to the fact that once the i^{th} component is eliminated, $\mathbb{E}_{s\sim T, a\sim\pi}\left[|\Delta_i(s)|\right]$ no longer needed to be computed. In order to spare useless training roll-outs, the corresponding training epochs are skipped and the corresponding gradient component is set to a value guaranteeing that the component is always selected among the eliminated set of components forever-after. A second factor comes from the fact that switches from phase 1 to 2 and 2 to 3 are conditioned to the stability of the performance. If this criterion is not met, training remains in the current phase. This action-sparsity-seeking variant of PPO-CMA is denoted AS-PPO-CMA thereafter.

DOI: 10.13009/EUCASS2022-7350

3.2 Choosing the ranking metric

Defining the ranking metric $\mathbb{E}_{s\sim T,a\sim\pi}\left[|\Delta_i(s)|\right]$ is the last task at hand. The impact on performance of eliminating a given action component is hard to evaluate for multiple reasons. A control action has an effect on the next state transition but also on the following ones. Strictly speaking, the dynamics T of our system satisfies the Markov property (i.e the Markov decision process is memoryless: $T(s_{t+1}|s_t, at) = T(s_{t+1}|s_t, at, s_{t-1}, a_{t-1}, \dots)$), this means the long-term impact should be entirely "encoded" in the next transition only. Yet, this impact may be hard to quantify, all the more that we resort to estimators trained in noisy data. Second, the elimination of an actuator puts the agent off-balance, and it "recovers" again by converging to a new extremum, by adapting its policy. It is only after this policy adaptation that the complete impact of eliminating an action component can be measured. Lastly, the order in which components are removed yields a potentially important impact on the final solution since, some actuators may have a different "importance" depending on the gating state of other ones and especially neighbors that might substitute for their removal or not. As we chose to proceed to a sequential elimination, this last issue cannot be solved satisfactorily and results must be considered with that in mind.

Thus we derive the following criteria, that an ideal metric would comply with:

1. **Stable & reproducible**: The metric must find the right balance between rapid adaptation to a significant evolution of the policy π but also enable a consistent and stable ranking of actuators once the control performance plateaus.

2. **Computationally cheap**: The computation overhead of the metric should be as limited as possible, both in terms of extra training epochs to run and of structures/estimators to compute.

3. **Far-sighted**: The long-term effects (on a run) of clipping an action component should be taken into account by the metric.

4. **One-move-ahead**: The metric should be able to consider the policy adaptation resulting of the elimination of an action component.

5. **Context-decoupled**: The metric should be able to accurately estimate the value of any combination of remaining action components at any time in the process. This last one is obviously not realizable with the current hypotheses.

All the candidate metrics are described in the following sections. Their compliance to these criteria and their performance on the two test-cases is compared and discussed later on.

3.3 Value function analysis

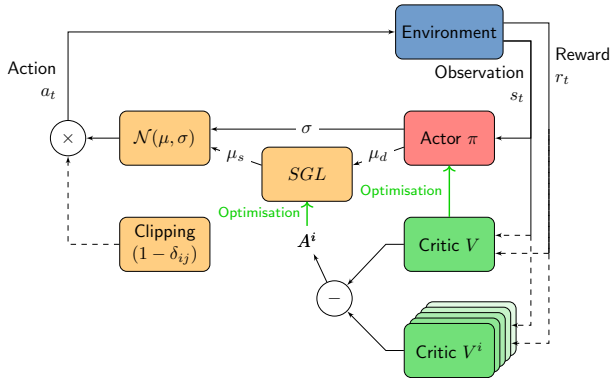

Figure 7: AS-PPO-CMA agent structure with value function estimation. The second training phase see the alternation of policy π and main critic V training with specific training for each critic V^i. The third phase focuses on tuning the SGL gate opening probability on the eliminated action component.

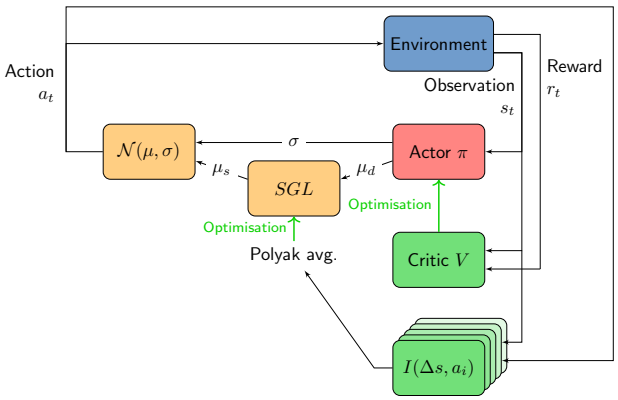

Figure 8: AS-PPO-CMA agent structure with mutual information estimation. In that case, both first and second training phases are performed simultaneously since mutual information estimation can be performed "on-the-fly" without any extra data collection roll-out.

DOI: 10.13009/EUCASS2022-7350

This first of the candidate metrics relies on a "what-if" analysis performed on the value function. A Polyak averaged estimation of $\mathbb{E}_{s\sim T,a\sim\pi}\left[|\Delta_i(s)|\right]$ is considered and $\Delta_i(s)$ is defined as:

$$\Delta_i(s) = V^{\pi}(s) - V^{\pi^i}(s)$$

where V^{π} estimates the expected return under policy π with the current gating mask and is already estimated by the standard PPO-CMA training process. V^{π^i} estimates the projected value function under the clipped policy $\pi^i = (1-\delta_{ij})\pi$, where the i^{th} action component is clipped systematically. The proposed method is rather straightforward as V^{π^i} is directly estimated using observed return values R_t of roll-outs (whose data is stored in buffer $\mathcal{B}_i$) performed using π^i as policy, thanks to a simple action-clipping mechanism downstream of the actor. Thus, in addition to the previously introduced neural structures, n extra neural networks V^i are built and tasked with providing accurate values for V^{π^i} for each $i \in \{0, ..., n-1\}$ as illustrated by figure 7. The training loss for each action component i is defined as:

$$\mathcal{L}_{V^i} = \frac{1}{|\mathcal{B}_i|} \sum_{(s_t,R_t)\in\mathcal{B}_i} \left(R_t - V^i(s_t)\right)^2 .$$

3.4 Mutual information analysis

The second metric is also deeply rooted into reinforcement learning hypotheses, and more precisely to the Markov Decision Process underlying the control problem. This metric proposes to rank actuators by the mutual information $I(\Delta s, a_i)$ computed between their action signal $a_{t,i}$ and the partial state transition $\Delta s_t = s_{t+1} - s_t$. Mutual information measures the dependence between two random variables, by giving the amount of information the observation of one variable gives about the other. I is 0 if both variables are independent. Given X and Y two random variables, one can write their mutual information as:

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

with H being the entropy of the random variable. This way, using $I(\Delta s, a_i)$ as an indicator of the actuator "importance" relies on the hypothesis that the flow partial dynamics $s_{t+1} \sim T(\cdot|s_t, a_t)$ complies well enough to the Markov property so that s_{t+1} contains all the relevant information about the forcing effect of $a_{t,i}$. The heuristic is that, if a given action component is "useless", it won't notably influence the state dynamics, contrary to an "important" action component that will "drive" the dynamics. In the latter case, the knowledge of such an action will reduce the uncertainty (H) about the state transition far more than in the first case.

Both entropy measures cannot be directly computed using the collected data. However, expressed as a Kullback-Leibler divergence $I(X,Y) = D_{KL}(P(X,Y)||P(X)\otimes P(Y))$, the Donsker-Varadhan variational formulation (Donsker & Varadhan, 1975) enables to compute a neural-network estimator of I:

$$I(\Delta s, a_i) = \sup_{\phi\in\mathcal{M}_b(\Omega)} \left(\mathbb{E}_{P(\Delta s,a_i)}\left[\phi\right] - \mathbb{E}_{P(\Delta s)\otimes P(a_i)}\left[e^{\phi}\right]\right),$$

where Ω is the sample space of $(\Delta s, a_i)$, $\mathcal{M}_b(\Omega)$ the set of all bounded measurable functions of Ω. Thus the function ϕ can be embodied by a neural-network and be trained to maximize the argument of the supremum. This method has been successfully used by Belghazi *et al.* (2018), Hjelm *et al.* (2018) and numerous other studies in the domain of image classification. As illustrated by figure 8, each mutual information $I(\Delta s, a_i)$ is estimated by a neural network (one per action component) and a Polyak average is run on these quantities to ensure stability throughout training. In that case, the maximum gate opening probability is not set to 1 but to 0.95 for reasons that are discussed later in part 4.6.

3.5 Norm-based

This last candidate metrics is based on the assumption that injected energy by the forcing is somehow proportionally linked to its impact on the flow. Thus, quantifying by a norm the forcing action appears relevant as ranking indicator. For a given actuation component i, one can estimate the expectation of a_i L_2-norm N_i on a control run:

$$N_i = \mathbb{E}_{s\sim T,a\sim\pi}\left[||a_i||_2\right].$$

Here the choice of an L_2-norm is rather arbitrary and can be discussed. This metric can simply be computed on the roll-out data, using a Polyak averaging to ensure an increased stability of the measurement.

DOI: 10.13009/EUCASS2022-7350

4. Results and discussion

In this section, the results of the three metrics on both test-cases are described and discussed. The study on the KS equation is performed using batches of respectively 200, 100 and 100 test-cases for the value function, the mutual information and the action norm metrics. The low computational cost of this environment allows for large batch sizes and thus quantitative and statistical analyses. A systematic study, where all 256 possible actuators layouts have been trained has been performed to be used as reference point. On the other hand, the slow convergence and large computational cost of the NACA test-case does not allow a systematic study enables only qualitative analyses, illustrating the potential of each method for flow control, batch sizes are respectively 5, 10 and 10 for the value function, the mutual information and the action norm metrics.

4.1 Comparison between metrics

Figures 9 and 10 compares the performances of the three metrics on the test cases. One can first notice that the mutual information performance for large numbers of actuators is significantly lower than the other performances on both test cases. This is due to the maximum open gate probability value set to 0.95 instead of 1 for the other metrics. It has been verified (refer to part 4.6) that with fully open gates, mutual information performs as well as the other methods. On the case of the KS equation (figure 9), all methods seem to perform rather equivalently within each other and match the performances of the systematic study (blue line), considered as almost topmost reachable performance. All the three metrics seem to be performing slightly better than the systematic study for one actuator, which is likely due to a different convergence (1000 training epochs for the systematic study versus around 2500 epochs from the start for action-sparsified cases).

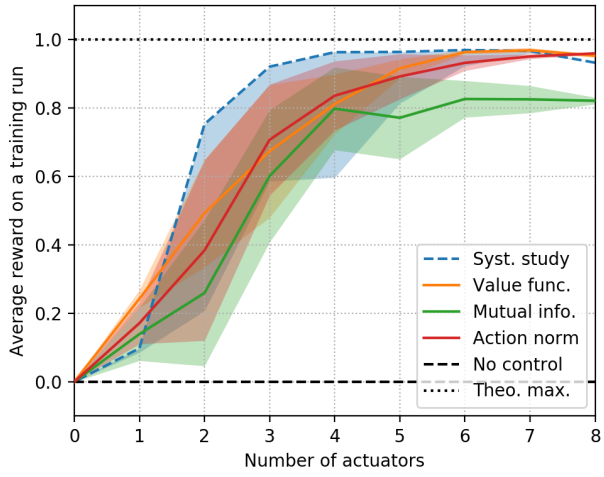

Figure 9: Average training performance with respect to the number of active action components on the KS test-case. The results of the proposed metrics (orange, green and red lines) are compared with the ones of the systematic study (blue line). Shaded areas illustrate ensemble standard deviations. Both maximum theoretical reward (dotted line) and baseline (no control - dashed line) performances are reported in black lines.

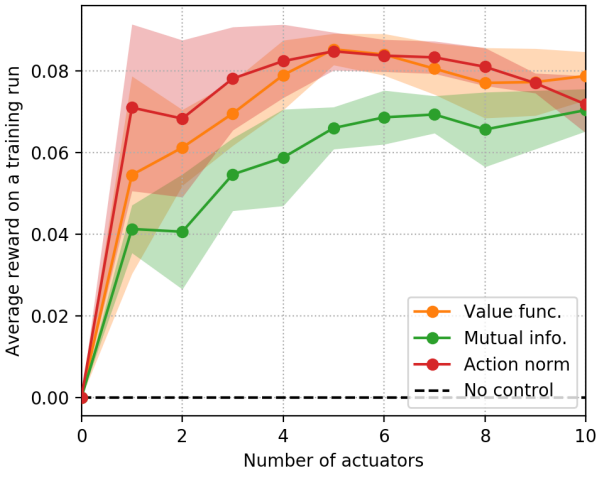

Figure 10: Average training performance with respect to the number of active action components on the NACA test-case. The solid lines indicate averages while, shaded areas illustrate ensemble standard deviations. The reward scaling is arbitrary since no re-attached base-flow has been computed to estimate the maximum theoretical reward value, but the average baseline reward computed is used as reference.

On the case of the NACA, the action norm and the value function seem to peak in performance for 5 actuators. This remains to be completely explained but one can hypothesize that it comes from the conjunction of reward formulation promoting stability, and the exploration noise, added to each action, whose impact thus decreases with the number of active components. This way, one can assume that a competition between these two factors reaches an optimized trade-off with 5 actuators. On the other hand concerning the mutual information metric, the open gate probability being less than one, this adds an extra source of action noise. This may be the reason of not having the same peak in performance as the two others.

DOI: 10.13009/EUCASS2022-7350

4.2 Statistical study of the results on the KS equation

A frequency analysis has been led to illustrate the different behaviors of the proposed metrics. Results are presented in figure 11. First, the norm-based analysis appears to behave differently than the two others with much diverse eliminations choices. The only major trend is the early removal of actuators #3 and #4. Both the value function and mutual information metrics remove the same first actuator (#0) but they quickly diverge. Overall, the mutual information encourages more spread-out layouts (refer to $n_{target} = 4$), with a evenly likely elimination of these remaining actuators afterwards, while the value function analysis quickly removes actuators #3 and #4, consistently with the action-norm metric, but afterwards proceeds more reproducibly and end-ups with most likely keeping actuator #7.

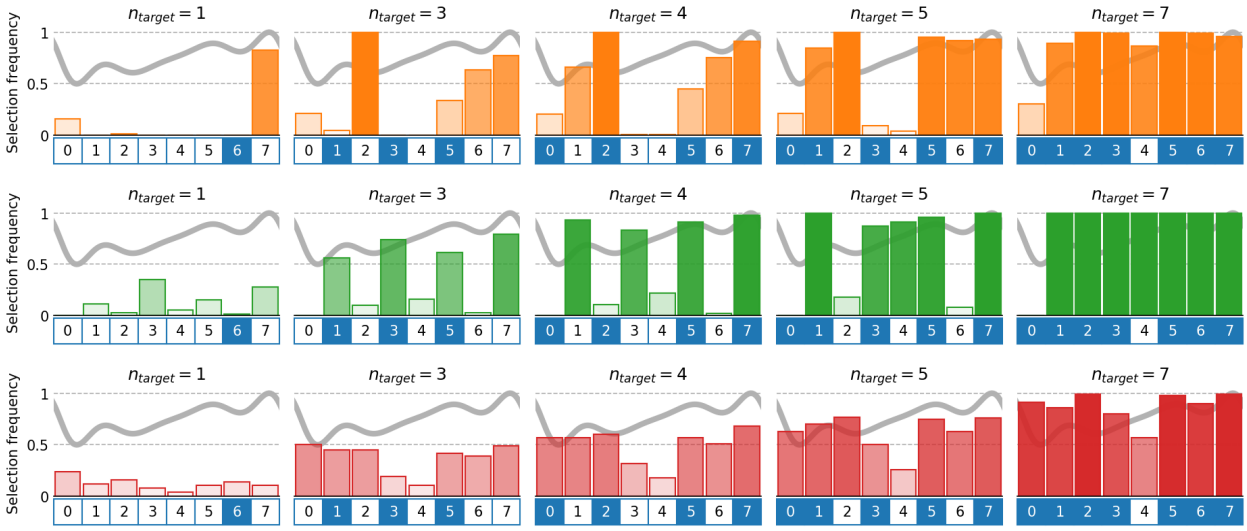

Figure 11: Frequency histograms of the obtained layouts (bar plots) for the value function (top, orange bars), the mutual information (middle, green bars) and action-norm (bottom, red bars) metrics. These can be compared to the best layout from the systematic study (blue numbered boxes). Fixed point E_1 has been plotted in the background as a reference.

None of the proposed metrics match the optimal layout from the systematic study. This is to be considered with both the fact that multiple layouts provide nearly optimal performances and the sequential aspect of the methods that, for instance, prevents the transition from the optimal layout with 4 actuators to the one with 3.

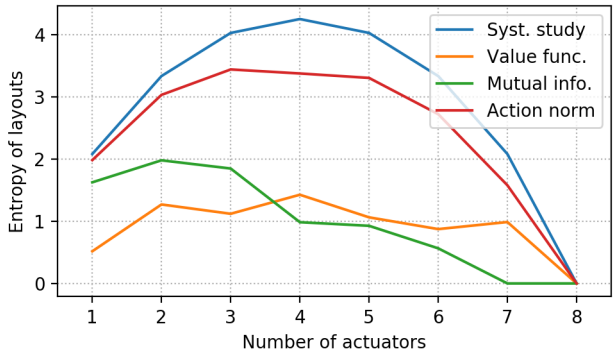

Figure 12: Entropy of obtained layouts with respect to the number of active actuators. The systematic study displays the maximum reachable entropy.

These trends are further confirmed by figure 12 showing the observed layout entropy with respect to the number of actuators. The entropy H of the layouts is computed as follows:

$$H = \sum_{l \in \text{layouts}} f_l \log(f_l),$$

where f_l is the observed frequency of a given actuator layout over the batch. The first deterministic choice of the mutual information metric complies with its entropy value remaining null. The overall trend of the action-norm method to

DOI: 10.13009/EUCASS2022-7350

produce a wider variety of layouts is confirmed by its entropy values, almost as high as the maximum value. The other tendency of the mutual information method being more reproducible than the value function method at the beginning of the elimination then becoming less deterministic is also confirmed. For instance, both value function and mutual information methods propose 10 different layouts having 4 active actuators, but the distribution is more even in the case of the value function, thus leading to a higher entropy.

4.3 Computational cost

The three proposed metrics require various computational overheads, both in terms of increased training duration (in epochs) than in neural network optimization. Since all neural networks have the same size and number of layers, the impact of the architecture of the neural structures being out of the scope of the present study. The computational overhead is thus counted in number of extra epochs and/or average number of neural networks optimized per training epoch. Both measures are segregated since the overhead cost is made of both the cost of running the environment and of the optimization phase. The total computational overhead ratio is defined as:

$$\text{total overhead ratio} = (1 + \text{extra epoch ratio})\,(1 + \text{computational overhead ratio}) - 1$$

$$\text{where} \quad \text{extra epoch ratio} = \frac{\text{average number of extra epochs}}{\text{average reference number of epochs}}$$

$$\text{and} \quad \text{computational overhead ratio} = \frac{\text{computational overhead}}{\text{environment cost + optimization cost}} \quad \text{(per epoch)}.$$

The environment cost per epoch (in floating-point operations) is hard to estimate especially in the case of a CFD simulation, but compared to a unitary optimization cost, it can be considered negligible in the case of the KS equation and dominating for the NACA test case. Table 1 summarizes the computational overheads in the case of the KS equation (env. cost $\ll$ optim. cost) and the NACA flow (env. cost $\gg$ optim. cost), for the three proposed metrics:

Metric	Extra epoch ratio	Comp. overh. ratio (KS vs NACA)		Total overh. ratio (KS vs NACA)	
Value function	92%	−24%	∼ 0%	46%	∼ 92%
Mutual information	0%	310%	∼ 0%	310%	∼ 0%
Action norm	0%	0%		0%	

Table 1: Comparison of computational overheads for the three proposed metrics.

The value function yields a negative computational overhead ratio per epoch simply because runs of the second training phase are only dedicated to training one V^i network. Yet the total overall ratio is much larger than one due to the larger extra number of epochs required to run approximately n_{act} extra "what-if" analysis roll-outs. Despite some skipping mechanisms allowing to reduce the computational burden on already converged value functions and eliminated components, this can represent a major drawback for costly environments such as CFD simulations. The mutual information analysis runs "on-the-fly", thus no extra epochs are needed. The increased number of neural networks is the only source of overhead. In the case of the KS equation, the environment running cost is supposed null, thus giving a clear advantage to the value function metric, but for NACA flow, the value function becomes much less advantageous (since extra epoch ratio costs scale with environment costs), and the global overhead induced by the mutual information method shrinks to a negligible value. Other considerations such as the stability of the metrics, impact the lengths of the training phases or initial number of actuators may have a small impact on the results, but would not impact the orders of magnitudes and thus, the conclusions.

4.4 Qualitative study of the results on the NACA test-case

Figure 13 illustrates the evolution of both drag (C_d) and lift (C_l) coefficients with the number of active action components. Compared to the value function analysis, both mutual information and action norm metrics tend to provide a lower performance for large numbers of actuators. This is likely due to a shorter second phase of training (where ranking estimators are converged) for these two methods, where the agent is further optimized. Mutual information appears to clearly perform worse than the other metrics on the average lift, this fact remains to be explained. This performance indicator not being directly embedded in the reward formulation, one can see it as a secondary objective that is expected to increase as the flow reattaches. One should note that these performances are measured during the elimination process. It is likely that performances achieved by the obtained actuator layouts trained from scratch would be different.

DOI: 10.13009/EUCASS2022-7350

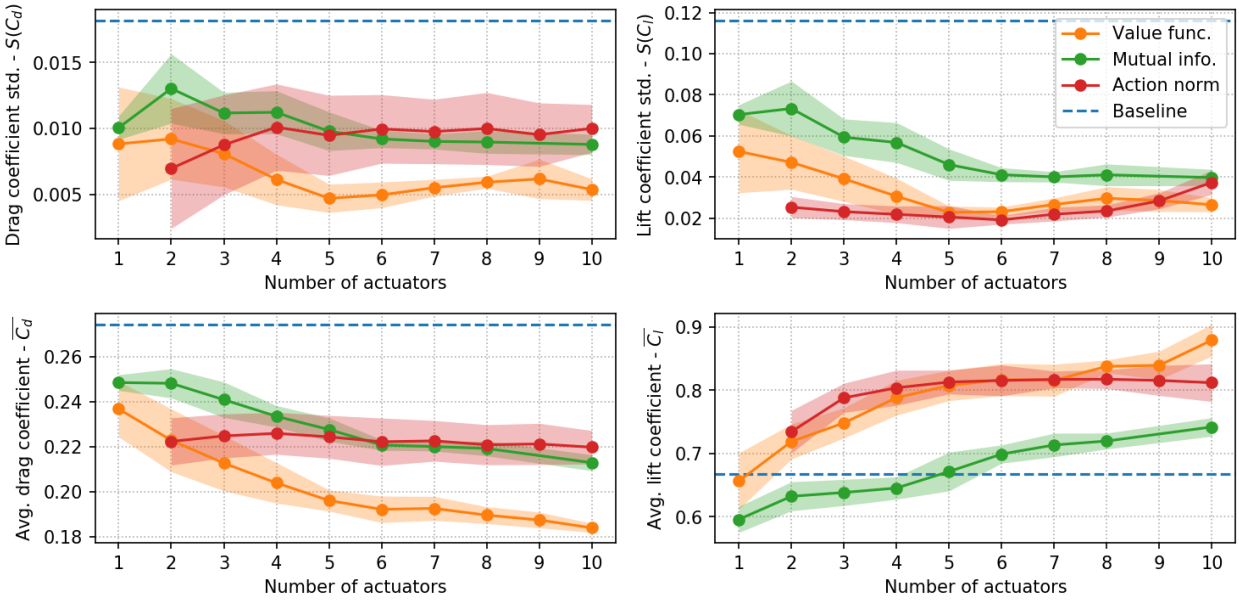

Figure 13: Averaged training performance indicators with respect to the number of active action components. Ensemble averages are computed on these data points (batch-size 5, 10 and 10 respectively). The blue dashed line represents the uncontrolled (baseline) performance, lines denote the evolution of the ensemble average and shaded areas illustrate the standard deviation across the batch. All indicators are computed on the last 40 control steps of a training run. (Upper left and right) Evolution of the standard deviation of the drag $S(C_d)$ and lift $S(C_l)$ coefficients. (Lower left and right) Time-averaged drag C_d and lift C_l coefficients.

4.5 Actuator neighboring effect

Figure 14 illustrates the impact of the elimination of a given actuator on the perceived importance of its neighbors on the KS equation. The elimination of the first actuator starts as soon as both performance and mutual information estimates are stabilized. From epoch 450 onward, the first actuator (actuator #0, located at $x = 0.0625$) is eliminated and consequently the mutual information of its two neighbors (actuators #1 and #7) drastically increases and stabilizes as the first two highest mutual informations, indicating their modified importance. This phenomenon also occurs during the second elimination (of actuator #6) where both actuators #5 and #7 see their importance modified.

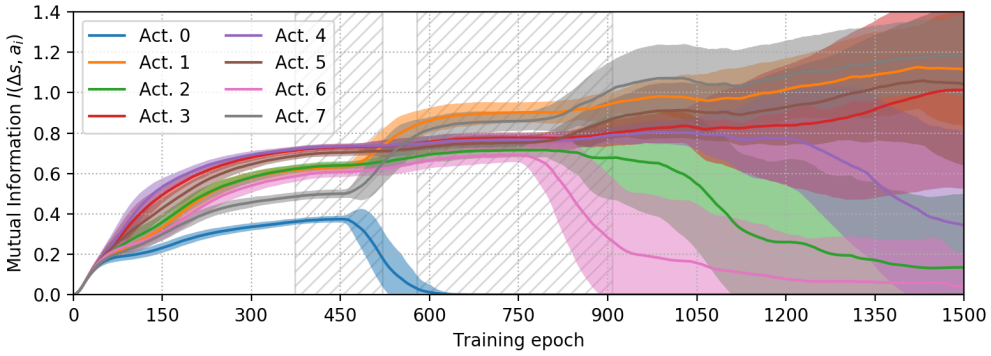

Figure 14: Evolution of the estimated mutual informations $I(\Delta s, a_i)$ for each actuator during training on the KS test-case. Averages are performed on a batch of 100 test-cases. Color-shaded areas correspond to the ensemble standard variation. Gray-hatched zones indicate the periods during which an action component is eliminated. Only the first two eliminations are reported here, for the sake of readability.

These observations support the fact that a simultaneous elimination would yield widely different actuator layouts and performances since for instance actuator #7 would be the second one eliminated with this method instead of being kept longer using sequential elimination.

DOI: 10.13009/EUCASS2022-7350

4.6 Correlation versus causation

Multiple ranking indicators are proposed in the current study. They all rely on indirect links between a given estimated quantity and the true quantity of interest which is the maximum performance (expected return) a given actuator layout would enable with a fully converged agent. One may then question if these links exploit a real causation or if it simply uses a possibly biased correlation.

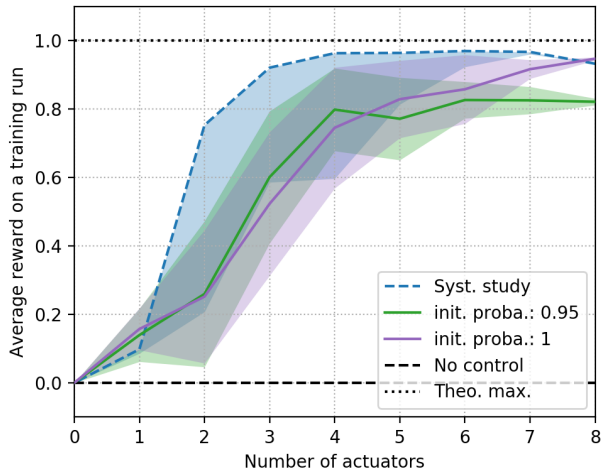

Figure 15: Average training performance with respect to the number of active action components on the KS test-case for he mutual information metric obtained with an initial gate opening probability of 0.95 (green) and 1 (purple).

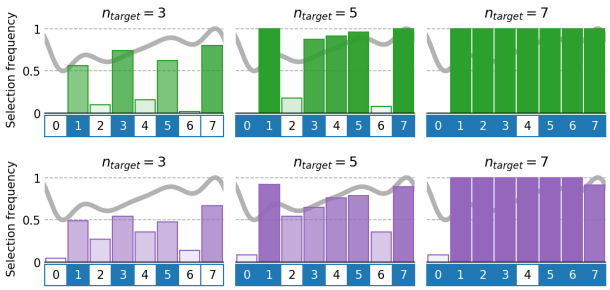

Figure 16: Frequency histograms of the obtained layouts (bar plots) for the mutual information metric obtained with an initial gate opening probability of 0.95 (top, green) and 1 (bottom, purple). These can be compared to the best layout from the systematic study (blue numbered boxes). Fixed point E_1 has been plotted in the background as a reference.

The action-norm-based metric uses the assumption that action norm (proportional to the injected forcing power in the system) correlates with the importance of the actuator. This correlation may obviously be spurious, for instance if an actuator is located somewhere insensitive to forcing. Value function analysis breaks the correlation of policy-produced action components by imposing null action values and thus directly performing the removal experiment all other things held constant. This method directly ensures causation. At last, for the mutual information metric, correlation may become more and more biased as the policy converges it becomes increasingly deterministic (π^* is not stochastic). Thus, action components become correlated with one another, but the causation between a given action component a_i and a state transition Δs cannot be asserted, since this transition could be caused by another correlated action component. This explains why the maximum and initial gate opening probability is set to 0.95 instead of 1. This uncertainty indeed slightly decorrelates the action components at the expense of a lower performance for five and more actuators as shown by figure 15. Figure 16 shows that this decorrelation enables a more deterministic/reproducible choice of layouts, most likely indicating a better choice quality.

5. Conclusion

In this study, three candidate metrics quantifying the importance of each actuator have been tested on two different test-cases: the 1-D Kuramoto-Siavshinky equation, which is a cheap environment displaying dynamics close to low Reynolds flows and presents intrinsic challenges to actuator selection and a bi-dimensional NACA airfoil flow, closer to practical control issues in fluid mechanics.

Criterion	Perfo.	Stable & Repro.	Comput.	Far-sighted	1-move-ahd.	Context-dec.
Value function	✓	~	✗	✓	✗	✗
Mutual information	~	✓	✓	✓*	✗	✗
Norm-based	~	✗	✓	~*	✗	✗

Table 2: Qualitative comparison of the fitness of the three introduced metrics to the previously presented quality criteria. *Depends on the compliance of the observation dynamics to the Markov property.

DOI: 10.13009/EUCASS2022-7350

Numerous issues ranging from performance to computational cost have been discussed and are qualitatively summarized in table 2. While value function appears as the most reliable metric in terms of actuator elimination and performance, it becomes prohibitively expensive for costly environments compared to the mutual information metric which is slightly less efficient but guarantees stable computational costs. At last, the action norm metric is the simplest indicator but is clearly biased by the spurious correlation between action amplitude and action importance to the control law.

All the previously introduced methods rely on a model-free policy-iteration algorithm (PPO-CMA). Indirect optimizations using value-iteration (such as DDPG or DQN), that are normally considered less efficient since not principled to the control objective, could become interesting in the context of actuator elimination that requires off-policy data. Model-based RL algorithms could also introduce a relevant approach to this issue.

Acknowledgments

This work is funded by the French Agency for Innovation and Defence (AID) via a PhD scholarship, their support is gratefully acknowledged.

References

Amitay, Michael & Glezer, Ari 2002 Role of actuation frequency in controlled flow reattachment over a stalled airfoil. *AIAA Journal* **40** (2), 209–216.

Belghazi, Mohamed Ishmael, Baratin, Aristide, Rajeshwar, Sai, Ozair, Sherjil, Bengio, Yoshua, Courville, Aaron & Hjelm, Devon 2018 Mutual information neural estimation. pp. 531–540. PMLR.

Bhattacharjee, Debraj, Hemati, Maziar, Klose, Bjoern & Jacobs, Gustaaf 2018 Optimal actuator selection for airfoil separation control. In *9th AIAA Flow Control Conference, 2018*, p. 3692.

Bruneau, Charles-Henri & Mortazavi, Iraj 2008 Numerical modelling and passive flow control using porous media. *Computers & Fluids* **37** (5), 488–498.

Brunton, Steven L & Noack, Bernd R 2015 Closed-loop turbulence control: progress and challenges. *Applied Mechanics Reviews* **67** (5), 050801.

Brunton, Steven L., Noack, Bernd R. & Koumoutsakos, Petros 2020 Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics* **52**, 477–508.

Bucci, Michele Alessandro, Semeraro, Onofrio, Allauzen, Alexandre, Wisniewski, Guillaume, Cordier, Laurent & Mathelin, Lionel 2019 Control of chaotic systems by deep reinforcement learning. *Proceedings of the Royal Society A* **475** (2231), 20190351.

Chomaz, Jean-Marc 2005 Global instabilities in spatially developing flows: non-normality and nonlinearity. *Annu. Rev. Fluid Mech.* **37**, 357–392.

Dandois, Julien, Mary, Ivan & Brion, Vincent 2018 Large-eddy simulation of laminar transonic buffet. *Journal of Fluid Mechanics* **850**, 156–178.

Donsker, Monroe D. & Varadhan, S. R. Srinivasa 1975 On a variational formula for the principal eigenvalue for operators with maximum principle. *Proceedings of the National Academy of Sciences* **72** (3), 780–783.

Edwards, Jack R. & Liou, Meng-Sing 1998 Low-diffusion flux-splitting methods for flows at all speeds. *AIAA Journal* **36** (9), 1610–1617.

Evans, Humberto Bocanegra, Hamed, Ali M., Gorumlu, Serdar, Doosttalab, Ali, Aksak, Burak, Chamorro, Leonardo P. & Castillo, Luciano 2018 Engineered bio-inspired coating for passive flow control. *Proceedings of the National Academy of Sciences* **115** (6), 1210–1214.

Giannetti, Flavio & Luchini, Paolo 2007 Structural sensitivity of the first instability of the cylinder wake. *Journal of Fluid Mechanics* **581**, 167–197.

Hämäläinen, Perttu, Babadi, Amin, Ma, Xiaoxiao & Lehtinen, Jaakko 2020 Ppo-cma: Proximal policy optimization with covariance matrix adaptation. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE.

DOI: 10.13009/EUCASS2022-7350

Hjelm, R. Devon, Fedorov, Alex, Lavoie-Marchildon, Samuel, Grewal, Karan, Bachman, Phil, Trischler, Adam & Bengio, Yoshua 2018 Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* .

Joubert, Gilles, Le Pape, Arnaud, Heine, Benjamin & Huberson, Serge 2013 Vortical interactions behind deployable vortex generator for airfoil static stall control. *AIAA Journal* **51** (1), 240–252.

LeCun, Yann, Bengio, Yoshua & Hinton, Geoffrey 2015 Deep learning. *Nature* **521** (7553), 436–444.

Li, Jichao & Zhang, Mengqi 2022 Reinforcement-learning-based control of confined cylinder wakes with stability analyses. *Journal of Fluid Mechanics* **932**, A44.

Louizos, Christos, Welling, Max & Kingma, Diederik P. 2018 Learning sparse neural networks through l_0 regularization. *Sixth International Conference on Learning Representations, Vancouver Canada, Monday April 30-Thursday May 03, 2018* .

Natarajan, Mahesh, Freund, Jonathan B. & Bodony, Daniel J. 2016 Actuator selection and placement for localized feedback flow control. *Journal of Fluid Mechanics* **809**, 775–792.

Paris, Romain, Beneddine, Samir & Dandois, Julien 2021 Robust flow control and optimal sensor placement using deep reinforcement learning. *Journal of Fluid Mechanics* **913**, A25.

Rabault, Jean, Kuchta, Miroslav, Jensen, Atle, RÃ©glade, Ulysse & Cerardi, Nicolas 2019 Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics* **865**, 281–302.

Rogers, James 2000 A parallel approach to optimum actuator selection with a genetic algorithm. In *AIAA guidance, navigation, and control conference and exhibit*, p. 4484.

Sashittal, Palash & Bodony, Daniel J. 2021 Data-driven sensor placement for fluid flows. *Theoretical and Computational Fluid Dynamics* **35** (5), 709–729.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec & Klimov, Oleg 2017 Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Seifert, A., Bachar, T., Koss, D., Shepshelovich, M. & Wygnanski, I. 1993 Oscillatory blowing: a tool to delay boundary-layer separation. *AIAA Journal* **31** (11), 2052–2060.

Seifert, Avraham, Darabi, A. & Wyganski, Israel 1996 Delay of airfoil stall by periodic excitation. *Journal of Aircraft* **33** (4), 691–698.

Seifert, A. & Pack, L. G. 1999 Oscillatory control of separation at high reynolds numbers. *AIAA Journal* **37** (9), 1062–1071.

Seshagiri, Amith, Cooper, Evan & Traub, Lance W. 2009 Effects of vortex generators on an airfoil at low reynolds numbers. *Journal of Aircraft* **46** (1), 116–122.

Shimomura, Satoshi, Ogawa, Takuto, Sekimoto, Satoshi, Nonomura, Taku, Oyama, Akira, Fujii, Kozo & Nishida, Hiroyuki 2017 Experimental analysis of closed-loop flow control around airfoil using DBD plasma actuator. In *ASME 2017 Fluids Engineering Division Summer Meeting, FEDSM 2017*, , vol. 58066, p. V01CT22A004. American Society of Mechanical Engineers.

Shimomura, Satoshi, Sekimoto, Satoshi, Oyama, Akira, Fujii, Kozo & Nishida, Hiroyuki 2020 Closed-loop flow separation control using the deep Q network over airfoil. *AIAA Journal* **58** (10), 4260–4270.

Sivashinsky, Gregory I. 1980 On flame propagation under conditions of stoichiometry. *SIAM Journal on Applied Mathematics* **39** (1), 67–82.

Vinuesa, Ricardo & Brunton, Steven L. 2021 The potential of machine learning to enhance computational fluid dynamics. *arXiv preprint arXiv:2110.02085* .

Wang, Yi-Zhe, Mei, Yu-Fei, Aubry, Nadine, Chen, Zhihua, Wu, Peng & Wu, Wei-Tao 2022 Deep reinforcement learning based synthetic jet control on disturbed flow over airfoil. *Physics of Fluids* **34** (3), 033606.

DOI: 10.13009/EUCASS2022-7350

Wu, Jie-Zhi, Lu, Xi-Yun, Denny, Andrew G., Fan, Meng & Wu, Jain-Ming 1998 Post-stall flow control on an airfoil by local unsteady forcing. *Journal of Fluid Mechanics* **371**, 21–58.

Yeh, Chi-An & Taira, Kunihiko 2019 Resolvent-analysis-based design of airfoil separation control. *Journal of Fluid Mechanics* **867**, 572–610.