Control of chaotic systems by deep reinforcement learning

Michele Alessandro Bucci, Onofrio Semeraro, Alexandre Allauzen, Guillaume Wisniewski, Laurent Cordier, Lionel Mathelin

Control of chaotic systems by Deep Reinforcement Learning

Michele Alessandro Bucci[a], Onofrio Semeraro[a], Alexandre Allauzen[a], Guillaume Wisniewski[a], Laurent Cordier[b], Lionel Mathelin[a]

[a] *LIMSI, CNRS, Université de Paris-Saclay, Orsay, FR*
[b] *Institut Pprime, CNRS, Université de Poitiers, ISAE-ENSMA, Poitiers, FR*

Abstract

Deep Reinforcement Learning (DRL) is applied to control a nonlinear, chaotic system governed by the one-dimensional Kuramoto-Sivashinsky (KS) equation. DRL uses reinforcement learning principles for the determination of optimal control solutions and deep Neural Networks for approximating the value function and the control policy. Recent applications have shown that DRL may achieve superhuman performance in complex cognitive tasks.

In this work, we show that using restricted, localized actuations, partial knowledge of the state based on limited sensor measurements, and model-free DRL controllers, it is possible to stabilize the dynamics of the KS system around its unstable fixed solutions, here considered as target states. The robustness of the controllers is tested by considering several trajectories in the phase-space emanating from different initial conditions; we show that the DRL is always capable of driving and stabilizing the dynamics around the target states.

The complexity of the KS system, the possibility of defining the DRL control policies by solely relying on the local measurements of the system, and their efficiency in controlling its nonlinear dynamics pave the way for the application of RL methods in control of complex fluid systems such as turbulent boundary layers, turbulent mixers or multiphase flows.

1. Introduction

With the availability of unprecedented computational resources, the maturity of modeling in many areas of engineering has yielded systems close to optimality in the context of well-known settings they were engineered for. To further improve such systems, a scientific challenge lies in rendering these systems adaptive by design to changes in the operating conditions. Enlarging the perspective from the engineering standpoint, the current environmental needs

Email address: michelealessandro.bucci [at] limsi.fr (Michele Alessandro Bucci)

have also invigorated the research effort on *flow control* applications. For example, carbon dioxide emissions are considered one of the main responsible for the global warming and any reduction of these emissions can lead to an attenuation of this effect. Increasing the efficiency of the existing technologies for the production of sustainable energy can lead to high potential benefits or to a substantial reduction of oil-consumption in the transport economy sector. Not surprisingly, a rather large body of literature is already available on the many different efforts in this realm, with varying assumptions on the system, such as linearity of the governing equations, or on the amount of information one has on the system, such as observability of a state vector or not. The interested reader is referred to the reviews by [1, 2, 3].

Active control for the optimization of the performance can be introduced via adequate strategies capable of modifying the system response in a prescribed manner (*open-loop*) or as a function of some observations of the system at hand (*closed-loop*). In both cases, the challenge is to infer an efficient and robust control strategy, hereafter termed *policy*, improving upon the retained objective function. In the usual *model-based* approach, a dynamical model is used to describe the behavior of the system.

This model allows to predict the effect of a given control action and can hence be used to derive the best control strategy leading to an optimal performance. However, a physical model[1] is not always available. Besides systems for which the governing equations are simply unknown or very poorly known, there are many situations where solving the governing equations is too slow with respect to the dynamics at play to be useful. While reduced-order models may help in solving an approximate system meeting real-time constraints, they usually lack robustness and can critically lose accuracy when control is applied, resulting in poor performance, at best.

A different line of control strategy relies on a *data-driven* approach. In this view, no model is employed and the control command is derived based on past observations only. In a training step, control actions are applied to the system and observations are made, possibly including the evaluation of the objective function. From this collection of observations, an input-output (I/O) model is built and subsequently used for controlling the system. Typical of this viewpoint are extremum seeking, [4], control strategies relying on system-identification techniques leading to auto-regressive models (AR, ARMA(X), etc.), *e.g.*, [5, 6], or the more recently proposed so-called Machine Learning Control, [7]. These techniques are more directly compatible with real systems since they do not require a physical model, nor prohibitive computational power. Moreover, they do not assume knowledge of a state vector and can efficiently deal with partial observations. Yet, they either exhibit limited performance in the general case where prior expertise knowledge is not available, or require an extensive train-

[1]The term *model* is ambiguous. Here, the term *model* refers to a *physical model*. This meaning differs from its use in the Machine Learning community where the term is more related to a parameterized function that links inputs to outputs.

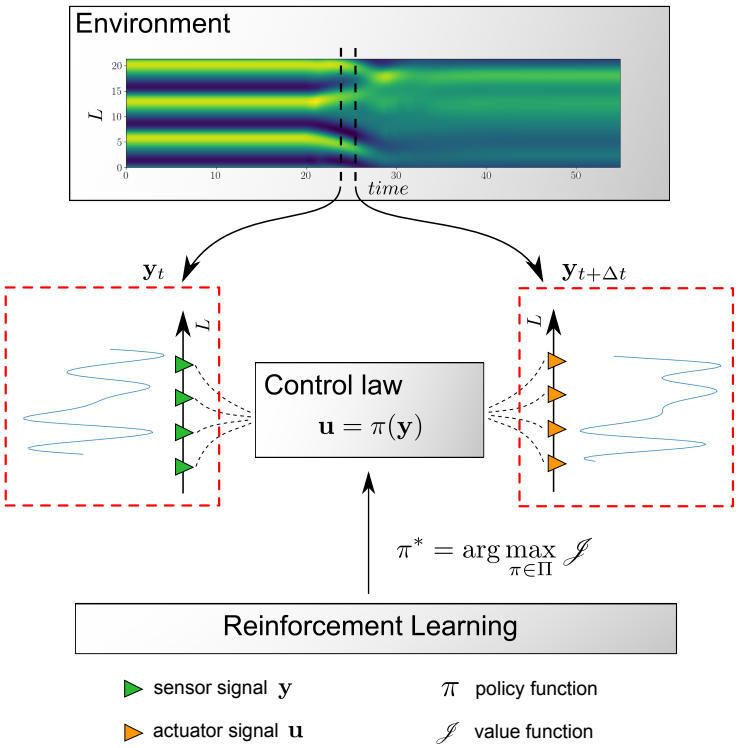

Figure 1: Overview of the Reinforcement Learning (RL) strategy for the Kuramoto-Sivashinsky equation considered in Sec. 4. In the RL framework, an *agent* interacts with an *environment* by making *observations* $\mathbf{y}$ and performing *actions* $\mathbf{u}$. In return, the agent receives a *reward* that depends directly on the changes of the environment induced by the action. The objective of RL is to determine the action $\mathbf{u}$ to impose on the environment in order to maximize a given *value function* $\mathcal{J}$ that represents the cumulative rewards over time. The *control law* $\mathbf{u}$ is determined via the *policy function* π which maps measurements $\mathbf{y}$ taken from the environment to the action space.

ing set of trial-and-errors, negatively affecting the learning time. In the present work, we introduce a Reinforcement Learning (RL) strategy [8] for the closed-loop, nonlinear control. RL is a well-established technique mainly originating from the robotics community and has gained wide-spread popularity with some recent mediatic applications, achieving super-human performances in the `go` and `shogi` games as well as self-teaching for the `StarCraft II` videogame, [9], or in revenue management [10] to cite only a few examples. As a data-driven technique, it shares the applicability of other I/O approaches, *e.g.*, low computational requirements, and the ability to use only limited sensors in contrast with a full state vector. Reinforcement learning dates back to the 1950s when the problem of optimal control was solved by Richard Bellman through the introduction of dynamic programming [11], leading – in the continuous formulation – to the Hamilton-Jacobi-Bellman (HJB) equation.

Figure 1 provides a sketch of the RL methodology for the control of dynamical system. The physical system under consideration, termed the *environment* in the reinforcement learning literature, is observed at time t by a so-called *agent* through a set of localized measurements $\mathbf{y}(t) \in \mathbb{R}^n$. The agent performs an action $\mathbf{u}(t)$ that changes the future state of the environment and, in return, receives a *reward* that represents the degree at which a state/action pair is desirable for the targeted objective function. This measure of performance to be optimized is then defined as the expectation of the (discounted) cumulative rewards over time. The action $\mathbf{u}(t) \in \mathbb{R}^m$ is evaluated from the current observations via the *control policy* π, defined as $\mathbf{u}(t) = \pi(\mathbf{y}(t))$, which represents a mapping from the observation space ($\mathbb{R}^n$) to the action space ($\mathbb{R}^m$). The policy is defined in a given class Π of multivariate functions. The *value function* $\mathcal{J} : \mathbb{R}^n \to \mathbb{R}$ or cost-to-go function of a policy π over a time horizon gives the cumulative rewards when $\mathbf{y}_t$ is the current measurement and the system follows policy π thereafter. The optimal policy, denoted $\pi^\star$, maximizes $\mathcal{J}$ over Π.

Estimation of the value function unlocks avenues for a more efficient use of the available information, potentially resulting in a faster training, and improved robustness with respect to small perturbations to the system and to the sensor measurements. Applications of RL have been mostly restricted to the discrete settings, where the number of possible actions is potentially huge, but finite, [12]. A notable extension to the continuous framework includes the work by Gorodetsky and collaborators using function trains, [13]. Our earlier efforts in bringing RL to the context of fluid flows involved Q-learning, [14], and temporal difference-based continuous approaches, [15, 16].

Recent efforts from the literature involve optimizing a collective swimming strategy [17] and the control of the flow around a circular cylinder in the laminar regime [18], among others. Here, we demonstrate the performance of our methodology in terms of quality of the control strategy (achieved performance) and robustness with respect to perturbations to the system and the sensor measurements.

In this proof-of-concept work, we focus on the control of a nonlinear, chaotic dynamical system, the 1D Kuramoto-Sivashinsky (KS) equation as a model of a fluid flow to be controlled. This model is described in a space-time domain with a 4-th order partial differential equation (PDE). The KS system exhibits some of the typical features observed in flow systems at low Reynolds number, such as traveling waves, and may reach a chaotic regime closer to turbulence for certain configurations, [19, 20, 21]. Due to these peculiarities, the KS system serves as a challenging test-bed for more complex fluidic systems to be controlled such as turbulent mixers or turbulent boundary layers.

The paper is organized as follows. The basics of our control strategies are presented in Sec. 2 and 3. In Sec. 2, we review how Reinforcement Learning is connected to the classical optimal control problem. We first derive the Hamilton-Jacobi-Bellman equation from optimality principles, and next particularize in the discrete time settings to obtain the Bellman equation. The connection with linear optimal controllers is further described in App. 5. In Sec. 3.1, we

briefly discuss the different classes of RL algorithms. In Sec. 3.2, the Deep Deterministic Policy Gradient (DDPG) reinforcement learning technique we are using is then presented. The DRL methodology is illustrated with the control of the Kuramoto-Sivashinsky model in Sec. 4 where results are put in perspective with insights from the physics. The paper finalizes with conclusions in Section 5.

2. From nonlinear optimal control to Reinforcement Learning

The section briefly introduces the mathematical background of Reinforcement Learning. For a deeper introduction to RL, we refer the interested reader to the books by [8, 22]. From the methodological viewpoint, we introduce RL following the approach taken in [23, 24], starting from the definition of the optimal control (Sec. 2.1) as basis for the Hamilton-Jacobi-Bellman (HJB) equation (Sec. 2.2) and stressing the analogies with optimal control theory. Further, we show how we can derive the time-discrete counterpart of the HJB equation, the Bellman equation (Sec. 2.3), which serves as the theoretical and mathematical ground for RL applications.

2.1. Definition of the control problem

We consider a dynamical system to be controlled with localized inputs (*actuators*) and outputs (*sensors*), see for instance Fig. 1 where these elements are organized along the streamwise coordinate x. Hereafter, the combination of the system to be controlled, the actuators and the sensors will be simply referred to as the *plant*, following the classical terminology in control theory. From the mathematical view point, this corresponds to defining the state-space model that, in the most general case, reads

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \mathbf{f}\left(\mathbf{v}(t), \mathbf{u}(t), t\right), \tag{1a}$$

$$\mathbf{y}(t) = \mathbf{g}\left(\mathbf{v}(t), \mathbf{u}(t), t\right). \tag{1b}$$

Equation (1a) is the state equation, where the map $\mathbf{f}$ propagates in time the state $\mathbf{v} \in \mathbb{R}^N$, while (1b) is the output equation. In the optimal control problem [25], we aim at defining a control signal $\mathbf{u} \in \mathbb{R}^m$ feeding the actuators, based on the sensor measurements $\mathbf{y} \in \mathbb{R}^n$, such that an objective function $\mathcal{J}$ is minimized

$$\mathcal{J} = h\left(\mathbf{v}(T), T\right) + \int_0^T r\left(\mathbf{v}(\tau), \mathbf{u}(\tau), \tau\right)\, \mathrm{d}\tau, \tag{2}$$

where h is a specified function, r is a reward associated with the action $\mathbf{u}$ and T is the optimization horizon. According to [23], this optimization problem can be embedded in a larger class of problems by considering

$$\mathcal{J}(\mathbf{v}_t, t, \underset{t\le\tau\le T}{\mathbf{u}(\tau)}) = h\left(\mathbf{v}(T), T\right) + \int_t^T r\left(\mathbf{v}(\tau), \mathbf{u}(\tau), \tau\right)\, \mathrm{d}\tau, \tag{3}$$

where t can be any value less than or equal to T. Following the convention tacitly introduced in Sec. 1, we note $\mathcal{J}^\star = \max_{\mathbf{u}} \mathcal{J}$ the optimal value of the objective function. The controller (or more properly said the *compensator*) provides the mapping between the measurements $\mathbf{y}$ of the system and the control actions $\mathbf{u}$. For now, we stress that the aim of RL is to determine an optimal policy function $\pi^\star$ that describes the optimal control $\mathbf{u}^\star$ from the current observations $\mathbf{y}$ [26, 27] following

$$\mathbf{u}^\star(t) = \pi^\star\left(\mathbf{y}(t), t\right). \tag{4}$$

2.2. Hamilton-Jacobi-Bellman equation

The optimal control problem stated in (1)-(2) can be solved by maximizing an augmented Lagrangian [25] where the governing equations act as constraints. After optimal conditions are imposed on the Lagrangian, a direct-adjoint optimality system can be derived. When the function $\mathbf{f}$ is linear or can be linearized, and the objective function is quadratic, the final controller is obtained as the solution to a Riccati equation (see Appendix 5 for the derivation). Here, we focus on the general nonlinear case (2), and follow a dynamic programming approach to solve the optimal control problem. The objective is to determine a Partial Differential Equation (PDE) for the optimal objective function (or value function in the RL terminology) $\mathcal{J}^\star$ such that the corresponding action satisfies the constraint given by the state equation (1). By definition, the maximum value of the objective function is equal to

$$\begin{aligned}\mathcal{J}^\star(\mathbf{v}(t), t) &= \max_{\mathbf{u}} \mathcal{J}(\mathbf{v}_t, t, \underset{t\le\tau\le T}{\mathbf{u}(\tau)}) \\ &= \max_{\substack{\mathbf{u}(\tau)\\ t\le\tau\le T}} \left[\int_t^T r\left(\mathbf{v}(\tau), \mathbf{u}(\tau), \tau\right) \mathrm{d}\tau + h\left(\mathbf{v}(T), T\right)\right].\end{aligned} \tag{5}$$

By splitting the integrand in (5) between the immediate reward in the interval $[t,\, t+\Delta t]$ and the future value function at $t+\Delta t$, we obtain

$$\mathcal{J}^\star(\mathbf{v}(t), t) = \max_{\substack{\mathbf{u}(\tau)\\ t\le\tau\le T}} \left[\int_t^{t+\Delta t} r\,\mathrm{d}\tau + \int_{t+\Delta t}^T r\,\mathrm{d}\tau + h\left(\mathbf{v}(T), T\right)\right]. \tag{6}$$

The principle of optimality requires that

$$\mathcal{J}^\star(\mathbf{v}(t), t) = \max_{\substack{\mathbf{u}(\tau)\\ t\le\tau\le t+\Delta t}} \left[\int_t^{t+\Delta t} r\,\mathrm{d}\tau + \mathcal{J}^\star(\mathbf{v}(t+\Delta t), t+\Delta t)\right]. \tag{7}$$

Expanding $\mathcal{J}^\star(\mathbf{v}(t+\Delta t), t+\Delta t)$ in a Taylor series about $(\mathbf{v}(t), t)$ and taking the limit as $\Delta t \to 0$ gives

$$-\dot{\mathcal{J}}^\star(\mathbf{v}(t), t) = \max_{\mathbf{u}(t)} \left[r\left(\mathbf{v}(t), \mathbf{u}(t), t\right) + \mathcal{J}^\star_{\mathbf{v}}\left(\mathbf{v}(t), t\right) \mathbf{f}\left(\mathbf{v}(t), \mathbf{u}(t), t\right)\right], \tag{8}$$

where $\dot{\mathcal{J}}^\star$ is the temporal derivative of the optimal value function and $\mathcal{J}^\star_{\mathbf{v}}$ is the derivative with respect to the state. Equation (8) is the Hamilton-Jacobi-Bellman (HJB) equation. This equation is continuous in time and defined backward. The terminal condition is given by

$$\mathcal{J}^\star(\mathbf{v}(T), T) = h\left(\mathbf{v}(T), T\right). \tag{9}$$

The HJB equation is a sufficient condition for an optimum [28, 29]. Indeed, a value function might fail to satisfy the differentiability and continuity conditions that are required to solve (8) and yet still be optimal. When solved over the whole state space and when the value function is continuously differentiable, the HJB equation becomes a necessary and sufficient condition for an optimum [23]. In case of a continuous action-state space, the optimal policy $\pi^\star$ is the one that produces the optimal trajectory by obeying the HJB. However, the whole action-state space is rarely known, especially when the dimension of $\mathbf{f}$ is large.

Note that, for infinite horizon optimizations, it is common to introduce a *discount rate factor* $\rho > 0$, that penalizes the immediate reward in the future. In this case, (5) becomes

$$\mathcal{J}^\star(\mathbf{v}(t), t) = \max_{\mathbf{u}} \int_t^\infty e^{-\rho\tau}\, r\left(\mathbf{v}(\tau), \mathbf{u}(\tau), \tau\right)\, \mathrm{d}\tau. \tag{10}$$

If we assume that $\mathcal{J}^\star$, $\mathbf{f}$ and r do not explicitly depend on the time t, the HJB equation then finally rewrites

$$\rho\mathcal{J}^\star(\mathbf{v}) = \max_{\mathbf{u}} \left[r(\mathbf{v}, \mathbf{u}) + \mathcal{J}^\star_{\mathbf{v}}(\mathbf{v})\, \mathbf{f}(\mathbf{v}, \mathbf{u})\right]. \tag{11}$$

2.3. Bellman equation

In the derivation of the HJB equation, we have tacitly assumed that the model (1) is known or can be inferred, for instance by system identification. This is usually not the case as, most of the time, only the state $\mathbf{v}$ or a reduced-order representation at a given time t can be accessed, for instance by instantaneous measurements. In this case, the right framework is to consider a Markov Decision Process (MDP) for which the decision making is formulated by means of a transition matrix expressing the probability of evolving from a state to another under the chosen action $\mathbf{u}$. This framework introduces a discrete time stochastic control process and, as such, requires the reformulation of the objective function in terms of expectation [8]. Letting $\mathbf{v}_t$ and $\mathbf{u}_t$ being the state and the action at the discrete time t, respectively, and $\mathbf{v}_{t+\Delta t}$ be the state at $t + \Delta t$, (11) can be rewritten as

$$\mathcal{J}^\star(\mathbf{v}_t) = \max_{\mathbf{u}} \left[\Delta t\, r(\mathbf{v}_t, \mathbf{u}_t) + \gamma\mathcal{J}^\star(\mathbf{v}_{t+\Delta t})\right], \tag{12}$$

where $\gamma = \exp\left(-\Delta t\, \rho\right)$ is the discount factor. Including Δt in the definition of $r(\mathbf{v}_t, \mathbf{u}_t)$, (12) is the Bellman optimality equation [11]. This equation, which describes the evolution of the optimal value function under the optimal policy $\pi^\star$, is the foundation of the dynamic programming theory.

Let $\mathcal{J}^\pi(\mathbf{v}_t)$ denote the long-term reward achieved for the state $\mathbf{v}_t$, and following a particular policy π. With the developments made in Sec. 2.2 for the HJB equation, we derive the Bellman equation for the value function given by

$$\mathcal{J}^\pi(\mathbf{v}_t) = r(\mathbf{v}_t, \mathbf{u}_t) + \gamma \mathcal{J}^\pi(\mathbf{v}_{t+\Delta t}). \tag{13}$$

Similarly, we can derive a Bellman equation for the *state-action value function* $Q^\pi(\mathbf{v}_t, \mathbf{u}_t)$ or *Q-function*. This quantity is a measure of the long-term reward assuming the agent is in state $\mathbf{v}_t$, performs action $\mathbf{u}_t$, and then continues following some policy π. The Bellman equation for the Q-function is written

$$Q^\pi(\mathbf{v}_t, \mathbf{u}_t) = r(\mathbf{v}_t, \mathbf{u}_t) + \gamma Q^\pi(\mathbf{v}_{t+\Delta t}, \mathbf{u}_{t+\Delta t}). \tag{14}$$

At this point, a few remarks are in order:

1. Since $\rho > 0$ and $\Delta t > 0$, the discount factor $\gamma \in (0, 1)$. In the MDP framework, the value function can be represented as the cumulative discounted reward or *return* R_t defined by

$$\mathcal{J}^\pi(\mathbf{v}_t) = R_t = \sum_{l=0}^{\infty} \gamma^l r(\mathbf{v}_{t+l\,\Delta t}). \tag{15}$$

   The two benefits of introducing a discounted reward is that the return is well defined for infinite series ($l \to \infty$), and that it gives a greater weight to earlier rewards, meaning that we care more about imminent rewards and less about rewards we will receive in the future.
2. Equation (12) highlights how the discounted infinite-horizon optimal problem can be decomposed in a series of local optimal problems. This is the *Bellman's principle of optimality*, [30]: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision.
3. In (13), the model (1) does not appear explicitly. If $\mathbf{v}_t$ can be observed, and the reward r can be measured, it is possible to recover $\mathcal{J}^\pi(\mathbf{v}_t)$ by inference from the agent-environment interaction ensuring that $\mathcal{J}^\pi(\mathbf{v}_t)$ is solution of the Bellman equation (13). This is precisely what the *Reinforcement Learning* (RL) approach does. In this framework, the policy and the value function are deterministic or stochastic functions. These functions can be represented as tables when the number of states and actions are sufficiently low. When there are more state and action variables, the *curse of dimensionality* occurs [31] and there is a need to approximate these functions as parameterized functional forms. Then, the *learning* process consists in optimizing the parameters so that the value function is maximized and the constraint imposed by the Bellman optimality (12) is satisfied. In *Deep Reinforcement Learning* (DRL), the functions are represented by a Neural Network (NN).

Within the RL framework, the Bellman equation is formulated in terms of the expectation of the value function. In the value Bellman equation (13), the value function is replaced by the expected value ($\mathbb{E}\left[\mathcal{J}^{\pi}(\mathbf{v}_t)\right]$). Similarly, in the cost-value Bellman equation (14), the Q-function is replaced by $\mathbb{E}\left[Q^{\pi}(\mathbf{v}_t, \mathbf{u}_t)\right]$.

3. Reinforcement Learning

As mentioned above, the aim of RL is to find the optimal policy $\pi^{\star}$ that maximizes the return (15). A crucial aspect for our application is the possibility for RL algorithms of learning directly from the observations resulting from the interactions of the agent with the environment. In that sense, we do not rely on the full knowledge of the state, but on a few localized measurements. Hereafter, the value function, the policy and the whole formulation will be given as parameterized functions of the observable $\mathbf{y}_t$. In Sec. 4, pointwise measurements will be used for the application on the KS system. In the following, we introduce a possible classification of the RL algorithms and specify our choices.

3.1. Several classes of RL algorithms

In the literature, many RL algorithms exist. In practice, the choice of the right algorithm often depends on the type of available actuators and sensors, on the system to control and on the task to fulfill. A general classification that embeds all RL algorithms can be made. Three classes of algorithms can be identified: i) *Actor–only*, ii) *Critic–only* and iii) *Actor–Critic*, where the words *actor* and *critic* are synonyms for the policy and value function, respectively.

Actor–only methods work with a parameterized family of policies. The gradient of the value function, with respect to the parameters, is estimated, and the parameters are updated in a direction of improvement. In the DRL approach, the policy is represented by a neural network to be inferred such that $\mathbf{u} = \pi(\mathbf{y}|\boldsymbol{\omega})$, with $\boldsymbol{\omega}$ the parameters of the NN. The learning process is guaranteed by choosing $\boldsymbol{\omega}$ such that the discounted reward R_t is maximized. In such a procedure, a single policy is evaluated by recording the system for a long time. This evaluation allows the computation of R_t, and of the gradient of the value function with respect to the NN parameters of the policy, *i.e.*, $\nabla_{\boldsymbol{\omega}}\mathcal{J}^{\pi}(\mathbf{y}_t)$. The algorithms belonging to this class are referred to as `REINFORCE` algorithms, [8]. A gradient-based optimization of the policy representation is carried-out with Stochastic Gradient Descent (SGD) algorithms (or closely related SGD-like strategies, for a recent review see [32]). A possible drawback of such methods is that the gradient estimations may have a large variance.

Critic–only methods rely exclusively on value function approximation and aim at learning an approximate solution to the Bellman equation, which will then hopefully prescribe a near-optimal policy. In the "Deep" flavor of the algorithms, the state-action value function Q^{π} is approximated by a NN, leading to $Q^{\pi}\left(\mathbf{y}_t, \mathbf{u}_t|\boldsymbol{\theta}\right)$, with $\boldsymbol{\theta}$ being the parameter vector of the NN. The parameters $\boldsymbol{\theta}$ are chosen such that the function Q^{π} is solution of the Bellman equation (14).

A Q-learning algorithm [33] may be used to approximate the optimal action-value function. The core of the algorithm is to update iteratively the function by using a weighted average of the old value and the new information:

$$\begin{aligned} Q^{\pi}\left(\mathbf{y}_t, \mathbf{u}_t|\boldsymbol{\theta}\right) \quad &\longleftarrow \quad Q^{\pi}\left(\mathbf{y}_t, \mathbf{u}_t|\boldsymbol{\theta}\right) \\ &+ \quad \alpha\left(r(\mathbf{y}_t, \mathbf{u}_t) + \gamma \max_{\mathbf{u}} Q^{\pi}\left(\mathbf{y}_{t+\Delta t}, \mathbf{u}|\boldsymbol{\theta}\right) - Q^{\pi}\left(\mathbf{y}_t, \mathbf{u}_t|\boldsymbol{\theta}\right)\right), \end{aligned} \tag{16}$$

where α is the learning rate ($0 < \alpha \leq 1$).

The use of NN in the Q-learning procedure establishes the Deep Q-Networks (DQN), [34]. This class of algorithms is often referred in the specialized literature as the first *"human-level"* control algorithm. The optimal policy does not need a function representation as it shall consist of choosing the action $\mathbf{u}$ that maximizes the optimal state-action value. For a finite number of possible actions, the Q-function is computed for each of these and the action that ensures the maximum is then chosen. The use of NN imposes the implementation of some strategies to regularize the learning process, such as memory, prioritized experience, target neural network, etc., [34].

From a mathematical point of view, the difference between the Actor–only and Critic–only approaches relies on the difference between the Pontryagin's maximum principle and the Bellman principle. The first principle is based on a *perturbative* approach: the neighborhood of the controlled trajectory is explored by perturbing the states in order to further minimize the cost function (curve optimization). The second principle is based on the solution of the HJB equation under the assumption of Markovianity of the system (function optimization) as explained in the previous paragraphs. While the Pontryagin's maximum principle is a necessary condition for the optimality, the fulfillment of the HJB equation is a necessary and sufficient condition if and only if the action-state is fully known (Legendre-Clebsch condition, see [35]).

The last class of algorithms, the Actor–Critic ones, combines the advantages of the two aforementioned approaches. The critic uses an approximation architecture to learn a value function, which is then used to update the actor's policy parameters in a direction of performance improvement. In DRL, two NNs are employed simultaneously, one for the policy and a second one for the value function. In that sense, a possible implementation consists in combining the `REINFORCE` and the Q-learning procedures such that the optimal controller is obtained, [36]. From the algorithmic viewpoint, the coupling of the two NNs is due to the update of the networks that is performed simultaneously by sharing the same expectation of the discounted reward. With a parameterized policy function, the limitation of the application of the Critic–only approach to a discrete action space is circumvented.

3.2. Deep Deterministic Policy Gradient as an actor-critic algorithm for RL

In this work, we consider the Deep Deterministic Policy Gradient (DDPG) algorithm [37] as an Actor–Critic, model free algorithm (see the graphical summary in Fig. 2). The advantage of DDPG compared to the Deep Q Network

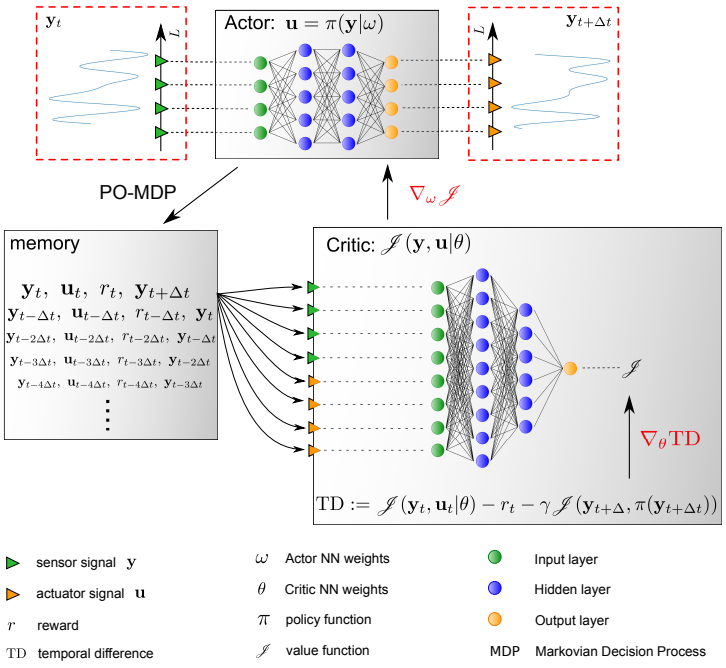

Figure 2: Graphical sketch for the Deep Deterministic Policy Gradient algorithm. The strategy is an Actor–Critic controller, where each of the part is implemented by means of a neural network, as depicted in the sketch.

considered in [34] is to handle continuous action domain that are often encountered in physical control tasks. Essentially, the DDPG algorithm adapts the theoretically-grounded Deterministic Policy Gradient (DPG) algorithm introduced in [38] to the Deep Reinforcement Learning setting where NNs are employed to represent the policy and value functions. The basic idea of DPG is to update the policy parameter $\boldsymbol{\omega}$ in the direction of the gradient of Q^π, rather than globally maximizing Q^π as it is done classically in policy gradient methods. After application of the chain rule to $Q^\pi\left(\mathbf{y}_t, \pi(\mathbf{y}_t|\boldsymbol{\omega})\right)$, we can show that the policy improvement at each iteration number k can be decomposed into the gradient of the state-action value with respect to actions, and the gradient of the policy with respect to the policy parameters (see Eqs. 6 and 7 in page 3 of [38]; *ibid.* the policy π is indicated as μ). We finally obtain:

$$\boldsymbol{\omega}^{k+1} = \boldsymbol{\omega}^k + \alpha \mathbb{E}\left[\nabla_{\boldsymbol{\omega}} \pi(\mathbf{y}_t|\boldsymbol{\omega})\ \nabla_{\mathbf{u}_t} Q^{\pi^k}\left(\mathbf{y}_t, \mathbf{u}_t\right)\Big|_{\mathbf{u}_t = \pi(\mathbf{y}_t|\boldsymbol{\omega})}\right], \tag{17}$$

where α is a learning rate.

The tuple corresponding to one time discrete transition $\{\mathbf{y}_t, \mathbf{u}_t, r_t, \mathbf{y}_{t+\Delta t}\}$ defines a Markov Decision Process (MDP) or Partially Observable Markov Decision Process (PO-MDP) in case only a partial measurement of the state is ac-

cessible. At each iteration, the PO-MDP is stacked in memory and successively used by the critic optimizer to reduce the Temporal Difference error defined as

$$Q^{\pi}\left(\mathbf{y}_t, \mathbf{u}_t|\boldsymbol{\theta}\right) - r(\mathbf{y}_t, \mathbf{u}_t) - \gamma\, Q^{\pi}\left(\mathbf{y}_{t+\Delta t}, \mathbf{u}_{t+\Delta t}|\boldsymbol{\theta}\right). \tag{18}$$

Regardless of the optimized critic NN, the actor NN is also optimized according to (17).

The optimality of the control is guaranteed by the Bellman principle under the hypothesis that the state-action space is known. For this reason, the state-action space needs to be explored. The exploration is carried out by perturbing the parameters of the policy as

$$\mathbf{u}_t = \pi(\mathbf{y}_t|\boldsymbol{\omega} + \mathcal{N}_1) + \mathcal{N}_2, \tag{19}$$

where $\mathcal{N}_1$ and $\mathcal{N}_2$ are two noise processes. $\mathcal{N}_1$ is essential at the beginning of the exploration process to introduce unbiased and uncorrelated random actions. The NN is a strongly non-linear representation of the true optimal policy function and a small variation of the parameters can thus lead to drastic changes in the output action. Both noise processes $\mathcal{N}_1$ and $\mathcal{N}_2$ vary over time. $\mathcal{N}_1$ is damped to a desired standard deviation of the resulting action $\mathbf{u}$, while the amplitude of $\mathcal{N}_2$ is given as a function of an Ornstein-Uhlenbeck process. This different choice introduces two different time-scales in the exploration process: when $\mathcal{N}_1$ is already damped, the noise process $\mathcal{N}_2$ still allows to explore efficiently the control landscape in the vicinity of the converged policy. We refer to DDPG articles for further technical information about its implementation, [38, 37, 39].

4. Control of the Kuramoto-Sivashinsky model

The DDPG algorithm introduced in the previous section is tested on the one-dimensional (1D) Kuramoto-Sivashinsky (KS) equation. The 1D KS equation is used for the description of flame fronts and reaction-diffusion systems. It is well known and referenced as being one of the simplest nonlinear PDEs exhibiting spatio-temporal chaos [19, 21], thus providing an appropriate test-bed for the proposed control strategy. In the following, we first describe the dynamics of the KS in the phase-space (Sec. 4.1), before a description of the controlled cases is given in Sec. 4.2.

4.1. Dynamics of the Kuramoto-Sivashinsky equation

The time evolution of the velocity $v = v(x,t)$ on a periodic domain of length L is given by

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = -\frac{\partial^2 v}{\partial x^2} - \frac{\partial^4 v}{\partial x^4} + g, \tag{20}$$

where g is a spatio-temporal forcing term. The equation is characterized on the left-hand side by a nonlinear convective term and on the right-hand side by

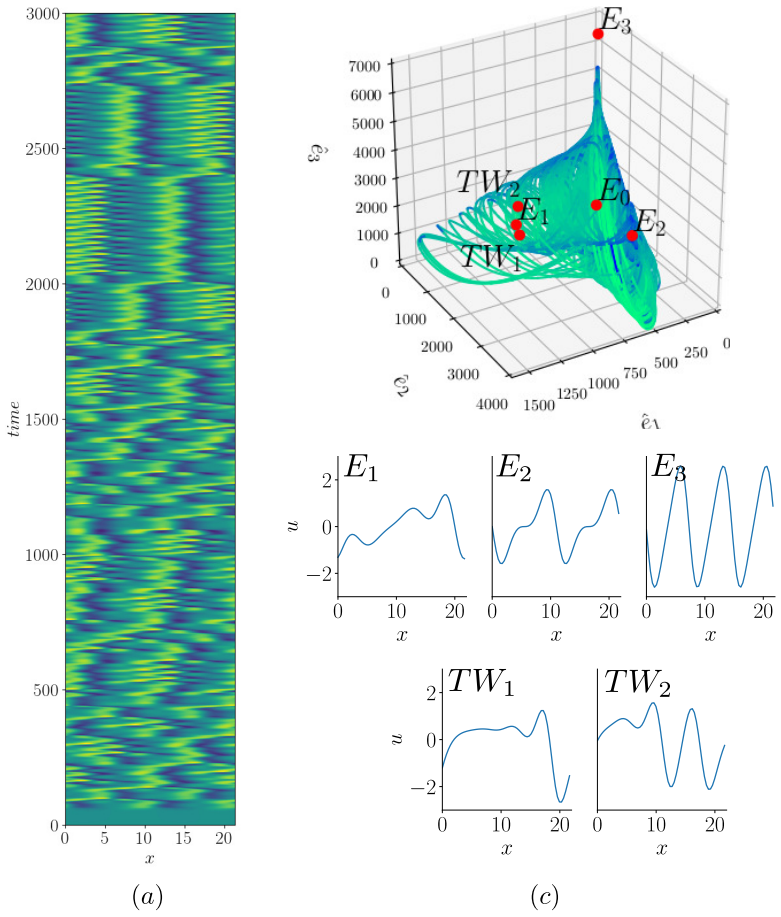

Figure 3: Dynamics of the Kuramoto-Sivashinsky (KS) equation on a time interval of 3000 time-units. The contour plot in (a) shows the space-time behavior of a trajectory emanating from the perturbed state E_0. The same trajectory is shown in (b) in the phase-space spanned by the dominant Fourier coefficients $\{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$. Red spots indicate: the trivial solution E_0, the three non-trivial invariant solutions E_i, $i = 1, 2, 3$, and the two traveling waves TW_1 and TW_2. The spatial distributions of the solutions are given in (c). Note how the trajectory is attracted by E_3, although never visited, while the other two equilibria are often visited.

a diffusion term expressed by two addends: a $2nd$-order derivative related to energy production, and a $4th$-order derivative acting as an hyper-diffusion.

The length of the domain dictates different regimes for the solution. Introducing the trivial solution $E_0 = 0$, it can be shown that for $L < L_c = 2\pi$, the dynamics is stable and converges towards E_0, while for $L > L_c$ a chaotic dynamics emerges (in a Lyapunov sense). In this work, we focus on the dynamics of the KS equation for a domain length of $L = 22$, identical to that studied in [21]. This length is large enough for exhibiting many of the features typical of turbulent dynamics observed in large KS systems, see [40], but sufficiently small for a thorough analysis of the state-space dynamics as performed by [41, 21].

4.1.1. Numerical Simulations

Numerical simulations are used for solving (20). As already mentioned, periodic boundary conditions are imposed, $v(x,t) = v(x+L,t)$. The spatial periodicity allows us to project the instantaneous solution onto Fourier modes and to perform spatial derivatives in the Fourier space. For $L = 22$, $N = 64$ Fourier collocation points have been used. This number of collocation points is deemed sufficient for an accurate representation of the spatio-temporal dynamics (see the related discussion in [21]). Time marching is carried out with a 3rd-order semi-implicit Runge-Kutta scheme [42]. The linear operator on the right-hand side of (20) is marched in time by an implicit scheme, whereas the non-linear and forcing terms are marched in time explicitly. For all numerical simulations, a time step of 0.05 was adopted.

In Fig. 3(a), a sample of the chaotic dynamics is shown when the solution is initialized with the trivial state E_0 perturbed by a Gaussian white noise with an amplitude of order 10^{-4}. The trajectory evolves in time as shown in the corresponding phase-space 3(b) obtained by projecting the dynamics onto the set of the dominant Fourier modes $\hat{e}_1$, $\hat{e}_2$, $\hat{e}_3$. Indeed, due to the periodicity of the domain, the velocity fields are characterized by different phases, with respect of which the Fourier projection is independent. In Fig. 3(b), the different invariant solutions computed by Newton iterations are shown, namely the three unstable, fixed points indicated by E_i $(i = 1, 2, 3)$ and the two traveling waves indicated by TW_i $(i = 1, 2)$. Apart for the relevant role in the dynamics, the invariant solutions E_i are used as target of our control strategy in Sec. 4.2. A detailed analysis of the resulting dynamics shown in Fig. 3(a)−(b) is beyond the scope of the present paper and already well described in [21]. However, it is interesting to observe how the system evolves by frequently visiting the vicinity of the solutions E_1 and E_2, as well as the two traveling waves; on the other hand, the dynamics is "attracted" by the unstable solution E_3, along the manifold associated with it, without ever reaching it.

4.2. Controlled system

As mentioned in Sec. 2, the plant is the system to be controlled, including the inputs (actuators) and the outputs (sensors) of the system. The number of actuators and sensors, their distribution and the combination of the two define a

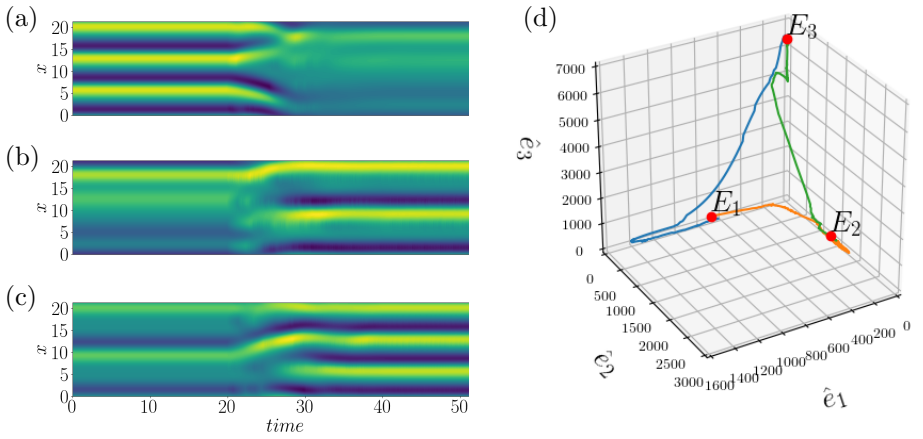

Figure 4: Closed-loop dynamics for the three control test cases: (a) $E_3 \to E_1$, (b) $E_1 \to E_2$ and (c) $E_2 \to E_3$. The controller is switched on at $t = 20$. The control strategy used in each test case is the optimal RL policy. The trajectories are shown in the Fourier phase-space in (d).

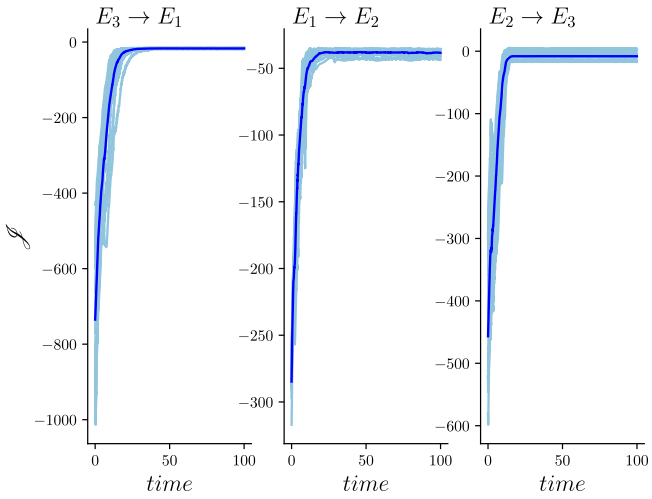

Figure 5: The value function is plotted for each of the three control test cases considered in Fig. 4. For each of the test case, the robustness of the policy is assessed by running 15 different experiments emanating from different initial conditions of the control. The darker blue line indicates the average of the value as a function of time. For each case, it can be observed that - regardless of the initial conditions - the controller is capable of reaching the target in a robust manner.

rather large parametric space. In this work, we consider 4 actuators equispaced along x. The support of each actuator is assumed Gaussian-shaped and the forcing term in (20) is given by

$$g\left(x;\{x_i^a\}_i\right) = \sum_{i=1}^{4} (2\pi\sigma)^{-1/2} \exp\left(-\frac{(x-x_i^a)^2}{2\sigma^2}\right), \tag{21}$$

where $x_i^a \in \{0, L/4, L/2, 3L/4\}$ is the location of the actuators along the x-axis and $\sigma = 0.4$ is the variance defining their spatial distribution. Regarding the sensors, we make the realistic assumption that real-life controllers rely only on partial information. We then introduce 4 equispaced sensors measuring the local velocity v. These sensors are staggered with respect of the actuator's locations and are located at $x_i^s \in \{L/8, 3L/8, 5L/8, 7L/8\}$.

4.2.1. Implementation of the control policies

The Deep Deterministic Policy Gradient approach (see Sec. 3.2) is implemented using scripts in `PyTorch`[2]. The critic and actor parts are both implemented by means of a neural network. The neural network of the critic part consists of an input layer of dimension 8, with 4 nodes dedicated to the actuator's signals and 4 to the sensor's ones, and a single scalar output layer; two hidden layers are introduced, with 256 and 128 nodes, respectively, both featuring the `swish` activation function, [43]. The neural network for the actor part approximates the control law. As such, the input layer is fed with the sensor measurements and is of dimension 4, while the output layer provides the control signal feeding the 4 actuators. Two hidden layers are used, of dimensions 128 and 64, with activation functions `swish` and `tanh`, respectively. The last layer acts as a saturating function. The maximum amplitude of the output is $g_{\max} = 0.5$. The training is performed using the `Adam` optimization algorithm, with learning rate of 0.001 for both the networks, and mini-batch with 200 examples for the optimization of the critic-network. Finally, the discount factor is set to $\gamma = 0.99$.

4.2.2. Results

We want to demonstrate the ability of the DDPG-based policies to drive the chaotic system towards the unstable fixed points and keep the dynamics in their vicinity, using the localized sensors and actuators discussed before. From the engineering viewpoint, this would correspond to leading a fluid system to a given operating condition only relying on some measurements of the environment. Specifically, we introduce three control test cases, each of them targeting to minimize the distance between the state v and the non-trivial invariant solutions, namely E_1, E_2 and E_3, respectively. In the following, for simplicity, we label the different policies with the invariant solution considered in their objective function.

[2] https://pytorch.org/

First, we consider each individual policy for driving the dynamics of the system from one invariant solution to another. In Fig. 4, the spatio-temporal behavior of the solution is shown in the inserts (a)-(c), and as a phase-space representation in (d). In each case, the RL controller is active for $t > 20$. For each control case, the numerical experiments are repeated 15 times with different initial conditions of the control, to assess the robustness of the policy. In Fig. 5, we report the value as a function of time for each experiment (light-blue). The darker line indicates the average of the runs. By comparing Fig. 4(d) and Fig. 5, we can observe that the controller is capable of driving the system in about 10 time-units towards the target solution. We also show that the control action of the policies is robust since the general behavior is comparable for all the runs considered.

To further verify the robustness of the policies, we consider a second set of numerical experiments, where the initial condition is randomly chosen in the phase space. This is shown in Fig. 6, where we consider 10 different initial conditions randomly and run the simulation under the control policy π_{E_3}. Five of these examples are reported in the inserts (a)-(e). The controller is capable of controlling the system in about 15 time-units in all but one cases (c) where it took about 40 time-units. The phase-space view is shown in (f), where the trajectories in (a)-(e) are shown in colors, while the others are in gray. The time evolutions of the corresponding values are shown in (g). Large excursions of $\mathcal{J}$ roughly correspond to fluctuations in the phase-space, although all trajectories finally converge towards E_3. In that sense, the high independence from the initial conditions – as compared to linear control methods such as the model-based LQG – and the ability to maximize the value function demonstrate the level of performance and robustness of the DDPG-based strategy applied to the KS system.

5. Conclusions

We have presented a deep reinforcement learning (DRL) algorithm successfully controlling the chaotic dynamics governed by the well-known nonlinear Kuramoto-Sivashinsky (KS) equation. The DRL combines reinforcement learning principles with deep Neural Networks for approximating the value function and the control policy. Among the different available strategies, we have tested an actor-critic algorithm, the Deep Deterministic Policy Gradient (DDPG). This choice is related to the possibility of directly tailoring the classical strategies from the nonlinear optimal control theory and, in particular, the solution of the Hamilton-Jacobi-Bellman equation, with the resulting approximation obtained by DDPG. From this perspective, our approach departs from recent efforts found in the flow-control literature as it allows knowledge of the cost landscape in the vicinity of the system trajectory, hence bringing robustness with respect to perturbations, while allowing for high control performance. Further, our off-policy framework is not episode-based and can then potentially require less training data (faster learning).

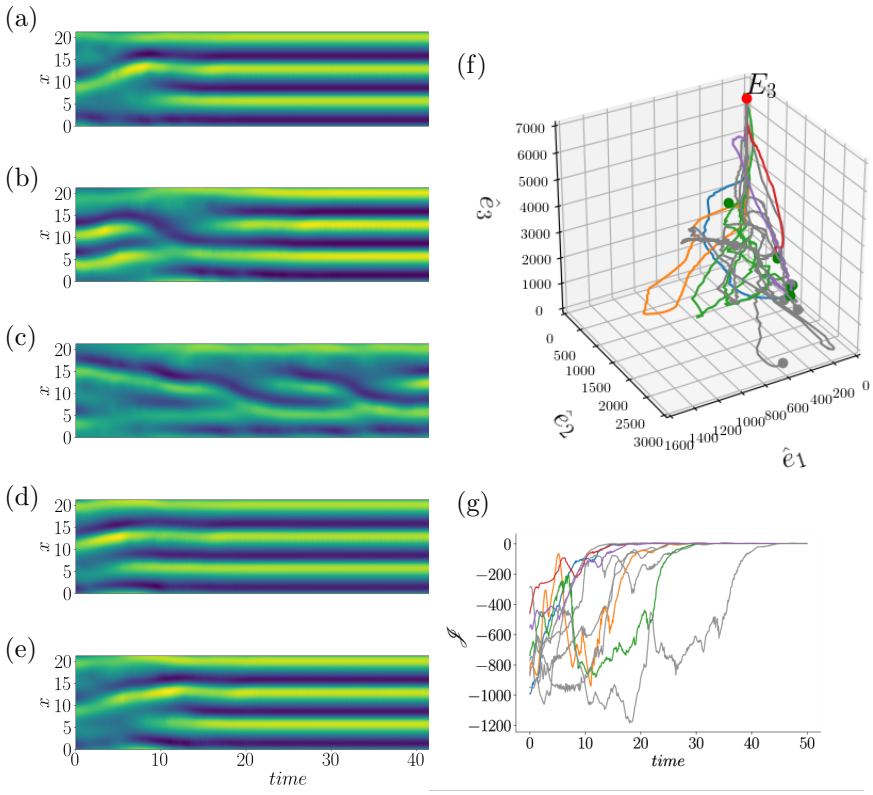

Figure 6: Illustration of the robustness of the closed-loop dynamics with respect to changes of the initial conditions. Ten different initial conditions are randomly chosen for the optimal policy driving the dynamics towards the unstable state E_3. Five of these examples are reported in the figures (a)-(e). It is shown that, except for the trajectory shown in (c), the controller is capable to control the system in approximately 15 time-units. The corresponding phase-space is shown in (f), where the trajectories in (a)-(e) are shown in colors, while the others are in gray. The corresponding values are shown in (g) as a function of time.

We emphasize that, in the present work, the controlled system exhibits a chaotic behavior in the analyzed regime. This configuration is significantly more involved than systems exhibiting a periodic, or quasi-periodic, dynamics. Nonetheless, we demonstrate that, using a limited, localized set of actuators and sensors and model-free DRL controllers, it is possible to drive and stabilize the dynamics of the KS system around its unstable fixed solutions, here considered as target states. Also, we show that the controllers are robust with respect of the initial conditions by considering numerous trajectories emanating from them; for all the tested cases, the performances are essentially unchanged with respect of the initial conditions, in contrast with the application of more classic, linear controllers where this choice is often critical.

The complexity of the KS system and the possibility of computing a DRL

control policy by solely using local measurements of the system suggest the extension of this application to the control of more realistic configurations such as turbulent boundary layers or mixers. In this sense, this work represents a first effort toward these applications. A number of developments are currently made to apply RL to these configurations addressing – among other limitations – limited and noisy observables, low measurement sampling frequency, high-dimensional state-action space and unknown and non-stationary time-delays.

Authors' contributions. The project has been initiated by LM and LC. MAB implemented all the routines used in this work for the analysis of the dynamics governed by the KS equation, the approximation and control by RL, with supervision from OS and LM, and feedbacks from AA. The paper was written by MAB, OS, LM and LC, with feedbacks from GW and AA.

Competing Interests. We declare we have no competing interests linked to this study.

Acknowledgements. This work has benefited from discussions with Charles Pivot whom is gratefully acknowledged. The authors are also thankful to Sylvain Cailllou for his support in the numerical implementation of the algorithm.

Funding. This project was generously funded by the French Agence Nationale pour la Recherche (ANR) and Direction Générale de l'Armement (DGA) via the FlowCon project (ANR-17-ASTR-0022).

Appendix A. Solving the linear optimal problem: Variational and HJB approaches

In this section we briefly summarize how a classic Linear Quadratic Regulator (LQR) problem can be derived from the more general formulation of the Hamilton-Jacobi-Bellman equation developed in Sec. 2.2. This Appendix is not meant to be exhaustive, but instead puts in perspective the application of the Reinforcement Learning with respect to standard approaches from control theory applied in the last decade in flow control. For more details on the optimal control theory, the interested reader can refer to [25] and to the reviews [1, 44, 2].

To begin with, we consider the assumption of a linear, time-invariant, state-space model

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = A\,\mathbf{v} + B\,\mathbf{u}, \tag{A.1a}$$

$$\mathbf{y} = C\,\mathbf{v} + D\,\mathbf{u}. \tag{A.1b}$$

In this case, the system matrix $A \in \mathbb{R}^{N\times N}$ is obtained from the discretization in space of the analyzed model, including boundary conditions; the spatial

distribution of the m actuators is indicated by the matrix $B \in \mathbb{R}^{N\times m}$, while $C \in \mathbb{R}^{n\times N}$ is the matrix including n sensors and $D \in \mathbb{R}^{n\times m}$ is the feedthrough (or feedforward) matrix. The aim of LQR is to define a control signal $\mathbf{u}$, such that the quadratic objective function

$$\mathcal{J}(\mathbf{v}_t, \mathbf{u}_t) = \frac{1}{2}\int_0^T \left(\mathbf{v}^\mathsf{T} Q_1 \mathbf{v} + \mathbf{u}^\mathsf{T} Q_2 \mathbf{u}\right)\, \mathrm{d}\tau \tag{A.2}$$

is minimized, [25]. The first term of (A.2) is related to the energy of the state $\mathbf{v}$, while the second one penalizes the energy expense of the control action. In this definition, the matrices $Q_1 \succcurlyeq 0 \in \mathbb{R}^{N\times N}$ and $Q_2 \succ 0 \in \mathbb{R}^{m\times m}$ respectively represent state and control penalties.

The objective is then to determine $\mathcal{J}^\star = \min_{\mathbf{u}} \mathcal{J}$ under the constraints of the state equation (A.1a). This optimal control problem can be solved by minimizing the augmented Lagrangian

$$\mathcal{L} = \frac{1}{2}\int_0^T \left(\mathbf{v}^\mathsf{T} Q_1 \mathbf{v} + \mathbf{u}^\mathsf{T} Q_2 \mathbf{u}\right)\, \mathrm{d}\tau + \int_0^T \mathbf{p}^\mathsf{T}\left(\dot{\mathbf{v}} - A\,\mathbf{v} - B\,\mathbf{u}\right)\, \mathrm{d}\tau, \tag{A.3}$$

where the governing equations with initial conditions $\mathbf{v} = \mathbf{v}_0$ act as constraint and the *co-state* or *adjoint state* $\mathbf{p}$ can be interpreted as a Lagrangian multiplier. The *optimality system* is then obtained by imposing the optimality conditions on $\mathcal{L}$. It leads to the following system of coupled equations:

$$\mathcal{L}_\mathbf{p} = \mathbf{0} \quad \Longrightarrow \quad \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = A\,\mathbf{v} + B\,\mathbf{u}, \qquad \mathbf{v}(0) = \mathbf{v}_0, \qquad \text{Direct equation} \tag{A.4a}$$

$$\mathcal{L}_\mathbf{v} = \mathbf{0} \quad \Longrightarrow \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -A^\mathsf{T}\,\mathbf{p} + Q_1\,\mathbf{v}, \qquad \mathbf{p}(T) = \mathbf{0}, \qquad \text{Adjoint equation} \tag{A.4b}$$

$$\mathcal{L}_\mathbf{u} = \mathbf{0} \quad \Longrightarrow \quad \mathbf{0} = B^\mathsf{T}\mathbf{p} + Q_2\mathbf{u}. \qquad \text{Optimality condition} \tag{A.4c}$$

The direct-adjoint system (A.4a)-(A.4b) is solved iteratively, by marching forward in time the direct equation, and backward in time the adjoint equation, forced by the second term on the right-hand side of (A.4b). The optimal condition is updated using the gradient given by (A.4c). It can be shown that the optimality system can be directly solved by means of an associated continuous time algebraic Riccati equation (CARE), [25].

We have shown in Sec. 3 that the HJB equation determines the minimum value function for a given dynamical system with an associated reward. This equation is directly linked to the class of optimal control problem treated previously. For this reason, we aim at obtaining the CARE starting from the HJB equation rewritten for the linear system as

$$-\dot{\mathcal{J}}^\star(\mathbf{v}, t) = \min_{\mathbf{u}} \mathcal{H}\left(\mathbf{v}(t), \mathbf{u}(t), \mathcal{J}^\star_\mathbf{v}, t\right) \tag{A.5}$$

$$= \min_{\mathbf{u}} \left[\frac{1}{2}\left(\mathbf{v}^\mathsf{T} Q_1\,\mathbf{v} + \mathbf{u}^\mathsf{T} Q_2\,\mathbf{u}\right) + \mathcal{J}^{\star\mathsf{T}}_\mathbf{v}\,\left(A\,\mathbf{v} + B\,\mathbf{u}\right)\right], \tag{A.6}$$

where $\mathcal{H}$ by definition is the Hamiltonian. The right-hand side of (A.6) is minimized for $\mathcal{H}_{\mathbf{u}} = \mathbf{0}$, leading[3] to the optimal control

$$\mathbf{u}^\star = -Q_2^{-1} B^\mathsf{T} \mathcal{J}_{\mathbf{v}}^\star. \tag{A.7}$$

Plugging this expression in (A.6), we get

$$-\dot{\mathcal{J}}^\star = \frac{1}{2}\mathbf{v}^\mathsf{T} Q_1\, \mathbf{v} - \frac{1}{2}\mathcal{J}_{\mathbf{v}}^{\star\mathsf{T}} B\, Q_2^{-1}\, B^\mathsf{T} \mathcal{J}_{\mathbf{v}}^\star + \mathcal{J}_{\mathbf{v}}^{\star\mathsf{T}} A\mathbf{v}. \tag{A.8}$$

At this point, the value function $\mathcal{J}^\star$ can be expressed as a function of the unknown, real symmetric positive-definite matrix $S(t)$

$$\mathcal{J}^\star(\mathbf{v}(t), t) = \frac{1}{2}\mathbf{v}^\mathsf{T}(t) S(t)\, \mathbf{v}(t). \tag{A.9}$$

This expression can be derived in time $\dot{\mathcal{J}}^\star = \frac{1}{2}\mathbf{v}^\mathsf{T}\dot{S}\,\mathbf{v}$, and with respect to the state $\mathbf{v}$, leading to the vector[4] $\mathcal{J}_{\mathbf{v}}^\star = S\mathbf{v}$. Introducing this term in (A.8), and considering that only the symmetric part of $S(t)A$ contributes to the result, we obtain

$$-\mathbf{v}^\mathsf{T}\dot{S}(t)\mathbf{v} = \mathbf{v}^\mathsf{T} Q_1\, \mathbf{v} - \mathbf{v}^\mathsf{T} S(t)\, B\, Q_2^{-1}\, B^\mathsf{T} S(t)\, \mathbf{v} + \mathbf{v}^\mathsf{T}\left(S(t)A + A^\mathsf{T} S(t)\right)\mathbf{v}, \tag{A.10}$$

from which it is possible to obtain the differential Riccati equation since the expression above is valid $\forall \mathbf{v}$:

$$-\dot{S}(t) = S(t)A + A^\mathsf{T} S(t) - S(t)\, B\, Q_2^{-1}\, B^\mathsf{T} S(t) + Q_1. \tag{A.11}$$

The optimal action is finally obtained as

$$\mathbf{u}^\star = -Q_2^{-1} B^\mathsf{T} S\, \mathbf{v} =: K\, \mathbf{v} \tag{A.12}$$

where $K(t)$ is the optimal control gain obtained as solution to the LQR problem. The corresponding time-invariant optimal control gain is obtained for $T \to \infty$, leading to the continuous time algebraic Riccati equation. In perfect analogy, it is possible to define an optimal estimation problem that, using the LQR and thanks to the separation principle, leads to the Linear Quadratic Gaussian compensator [26, 25].

As a conclusion, some observations can be made. First of all, since the Riccati equation is not directly solvable for systems characterized by a large number of degrees of freedom, say $N > 10^4$, the direct-adjoint system (A.4a)-(A.4b) is often solved instead, see for instance [45] and [46]. Optimization based on a finite sliding temporal window leads to the class of Model Predictive Controllers, often characterized by a nonlinear loop; examples in fluid mechanics are provided by the seminal work by[47] and the recent application by [48].

[3] Since Q_2 is a positive definite matrix, the existence of Q_2^{-1} is guaranteed.

[4] Note also that if we compare the optimal control (A.7) determined with the Hamiltonian and the one that can be deduced from the optimality condition (A.4c), we arrive to $\mathbf{p} = \mathcal{J}_{\mathbf{v}}^\star = S\,\mathbf{v}$.

References

[1] J. Kim, T. R. Bewley, A Linear Systems Approach to Flow Control, Ann. Rev. Fluid Mech. 39 (2007) 39–383. doi:10.1146/annurev.fluid.39.050905.110153.

[2] D. Sipp, P. J. Schmid, Linear closed-loop control of fluid instabilities and noise-induced perturbations: A review of approaches and tools, Appl. Mech. Rev. 68 (2) (2016) 020801. doi:10.1115/1.4033345.

[3] S. Brunton, B. Noack, Closed-loop turbulence control: Progress and challenges, Appl. Mech. Rev. 67 (5) (2015) 050801–050801–48.

[4] R. Becker, R. King, R. Petz, W. Nitsche, Adaptive closed-loop control on a high-lift configuration using extremum seeking, AIAA Journal 45 (6) (2007) 1382–1392.

[5] M. A. Kegerise, R. H. Cabell, L. N. Cattafesta, Real time feedback control of flow-induced cavity tones - part 1: Fixed-gain control, Journal of Sound and Vibration 307 (2007) 906–923.

[6] M. A. Kegerise, R. H. Cabell, L. N. Cattafesta, Real time feedback control of flow-induced cavity tones - part 2: Adaptive control, Journal of Sound and Vibration 307 (2007) 924–940.

[7] N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, M. Abel, Closed-loop separation control using machine learning, J. Fluid Mech. 770 (2015) 442–457. doi:10.1017/jfm.2015.95.

[8] R. S. Sutton, A. G. Barto, Introduction to reinforcement learning, Vol. 135, MIT press Cambridge, 1998.

[9] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, Nature 550 (2017) 354–371.

[10] R. Lawhead, A. Gosavi, A bounded actor-critic reinforcement learning algorithm applied to airline revenue management, Engineering Applications of Artificial IntelligenceTo appear.

[11] R. Bellman, Dynamic programming and stochastic control processes, Information and control 1 (3) (1958) 228–239.

[12] A. A. Gorodetsky, S. Karaman, Y. M. Marzouk, Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition, in: Robotics: Science and Systems, 2015, pp. –.

[13] A. A. Gorodetsky, S. Karaman, Y. M. Marzouk, High-dimensional stochastic optimal control using continuous tensor decompositions, The International Journal of Robotics Research 37 (2-3) (2018) 340–377.

[14] F. Guéniat, L. Mathelin, M. Hussaini, A statistical learning strategy for closed-loop control of fluid flows, Theo. Comput. Fluid Dyn. 30 (2016) 1–14.

[15] C. Pivot, L. Cordier, L. Mathelin, F. Guéniat, B. R. Noack, A continuous reinforcement learning strategy for closed-loop control in fluid dynamics, in: 35th AIAA Applied Aerodynamics Conference, Denver, CO, USA, 2017, p. 3566.

[16] C. Pivot, A. A. Gorodetsky, L. Mathelin, L. Cordier, Toward control of weakly observed nonlinear dynamical systems using reinforcement learning, in: SIAM UQ 18, Garden Grove, CA, USA, 2018, pp. –.

[17] S. Verma, G. Novati, P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning, Proceedings of the National Academy of Sciences 115 (23) (2018) 5849–5854. arXiv:https://www.pnas.org/content/115/23/5849.full.pdf, doi:10.1073/pnas.1800923115.
URL https://www.pnas.org/content/115/23/5849

[18] J. Rabault, M. Kuchta, A. Jensen, U. Reglade, N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control, J. Fluid Mech. 865 (2019) 281–302. doi:10.1017/jfm.2019.62.

[19] P. Holmes, J. Lumley, G. Berkooz, Turbulence Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, 1996.

[20] T. Bohr, M. H. Jensen, G. Paladin, A. Vulpiani, Dynamical systems approach to turbulence, Cambridge University Press, 2005.

[21] P. Cvitanović, R. L. Davidchack, E. Siminos, On the state space geometry of the kuramoto–sivashinsky flow in a periodic domain, SIAM Journal on Applied Dynamical Systems 9 (1) (2010) 1–33.

[22] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[23] D. E. Kirk, Optimal control theory: an introduction, Courier Corporation, 2012.

[24] B. Recht, A tour of reinforcement learning: The view from continuous control, Annu. Rev. Control Robot Auton. Syst.

[25] F. L. Lewis, D. Vrabie, V. L. Syrmos, Optimal control, John Wiley & Sons, 2012.

[26] T. Glad, L. Ljung, Control Theory, Taylor & Francis, London, 2000.

[27] S. Skogestad, I. Postlethwaite, Multivariable Feedback Control, Analysis to Design, 2nd Edition, Wiley, 2005.

[28] D. P. Bertsekas, Dynamic programming and optimal control, Vol. 1, Athena Scientific Belmont, Massachusetts, 1996.

[29] R. F. Stengel, Optimal control and estimation, Dover, 1994.

[30] R. Bellman, A markovian decision process, Journal of Mathematics and Mechanics (1957) 679–684.

[31] R. Bellman, Adaptive control processes: a guided tour, Princeton University Press, 1961.

[32] S. Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747.

[33] C. J. Watkins, P. Dayan, Q-learning, Machine learning 8 (3-4) (1992) 279–292.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.

[35] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, Cambridge, MA, 2005.

[36] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International conference on machine learning, 2016, pp. 1928–1937.

[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv:1509.02971.

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic Policy Gradient Algorithms, in: International Conference on Machine Learning, 2014, pp. –.

[39] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, arXiv:1511.05952.

[40] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach, Physical review letters 120 (2) (2018) 024102.

[41] J. Greene, J.-S. Kim, The steady states of the kuramoto-sivashinsky equation, Physica D: Nonlinear Phenomena 33 (1-3) (1988) 99–120.

[42] S. K. Kar, A semi-implicit runge–kutta time-difference scheme for the two-dimensional shallow-water equations, Monthly weather review 134 (10) (2006) 2916–2926.

[43] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv:1710.05941.

[44] N. Fabbiane, O. Semeraro, S. Bagheri, D. S. Henningson, Adaptive and model-based control theory applied to convectively unstable flows, Appl. Mech. Rev. 66 (6) (2014) 060801. doi:10.1115/1.4027483.

[45] T. Bewley, P. Luchini, J. O. Pralits, Methods for solution of large optimal control problems that bypass open-loop model reduction, Meccanica 51 (12) (2016) 2997–3014. doi:10.1007/s11012-016-0547-3.

[46] P. Luchini, A. Bottaro, Adjoint equations in stability analysis, Ann. Rev. Fluid 46 (2014) 493–517. doi:10.1146/annurev-fluid-010313-141253.

[47] T. R. Bewley, P. Moin, R. Temam, Dns-based predictive control of turbulence: an optimal benchmark for feedback algorithms, J. Fluid Mech. 447 (2) (2001) 179–225. doi:10.1017/S0022112001005821.

[48] S. Cherubini, J.-C. Robinet, P. De Palma, Nonlinear control of unsteady finite-amplitude perturbations in the blasius boundary-layer flow, J. Fluid Mech. 737 (2013) 440–465. doi:10.1017/jfm.2013.576.