Reinforcement Learning for Fluid Mechanics: an overview on Fundamentals from a Control Perspective

Onofrio Semeraro

Reinforcement Learning for Fluid Mechanics: an overview on Fundamentals from a Control Perspective

Onofrio Semeraro*

Laboratoire interdisciplinaire des sciences du numérique (LISN), CNRS
Université Paris-Saclay, 91400 Orsay, France

November 20, 2024

This chapter introduces some theoretical foundations of reinforcement learning and its applications to fluid mechanics from a control theory perspective. This choice is intended to provide the reader accustomed to flow control the key ideas in a more familiar vocabulary and a perspective on a vast literature blooming with tremendous momentum. First, we set the stage by introducing flow control and re-framing motivations and goals using reinforcement learning. Next, we shift our focus on some basic concepts by discussing the Hamilton-Jacobi-Bellman equation, the dynamic programming for nonlinear optimal control, and the iterative schemes used for approximate solution. Finally, the chapter closes by reconciling these elements with the terminology of reinforcement learning practice.

*onofrio.semeraro@universite-paris-saclay.fr

Contents

1 Flow control and reinforcement learning **3**
- 1.1 Standard approaches in flow control: a brief overview . . . 4
- 1.2 Reinforcement learning in flow control . . . 5
- 1.3 Organization of the chapter . . . 7

2 From nonlinear control to reinforcement learning **8**
- 2.1 Nomenclature and state space representation . . . 8
- 2.2 Optimal control in linear system: the Riccati equation . . . 10
- 2.3 The Hamilton-Jacobi-Bellman equation . . . 12
  - 2.3.1 From the HJB equation to the Riccati equation in linear plants . . . 13
  - 2.3.2 A quick observation on the Hamiltonian function . . . 14
- 2.4 Discrete systems . . . 14
  - 2.4.1 The linear quadratic regulator (LQR) in discrete systems . . . 15
- 2.5 Bellman's principle of optimality . . . 15
  - 2.5.1 LQR using Bellman equation backward in time . . . 16
- 2.6 Approximating the Bellman equation: iterative methods . . . 18
  - 2.6.1 Policy iteration (PI) . . . 18
  - 2.6.2 Value iteration (VI) . . . 19
  - 2.6.3 Generalized policy iteration . . . 20
- 2.7 Reinforcement learning (RL) . . . 21
  - 2.7.1 Markov decision processes (MDP) . . . 21

3 Elements of reinforcement learning in practice **23**
- 3.1 A short glossary . . . 23
- 3.2 Identifying the policy . . . 24
  - 3.2.1 Temporal difference (TD) . . . 25
  - 3.2.2 $\mathcal{Q}$-learning and SARSA . . . 25
  - 3.2.3 Actor-critic algorithms . . . 25
- 3.3 Numerical approximations . . . 26

4 Essential bibliography and final remarks **28**

1 Flow control and reinforcement learning

Control applications in fluid mechanics have attracted the attention of numerous research efforts as it is nowadays recognized that the optimization of aerodynamic flows in aircraft and vehicles design may have a deep impact on the reduction of pollutant emissions, mitigation of acoustic noise or control of highly complex conditions such as separation and stall (Abergel and Temam, 1990; Gad-el Hak, 2000; Kim and Bewley, 2007; Brunton and Noack, 2015; Duriez et al., 2016; Rowley and Dawson, 2017). Several methodologies have been applied to control fluids, ranging from passive to active strategies. Here, we consider active flow control (AFC): the dynamics is modified by injecting energy into the system using actuators, acting as transducers for the flow manipulation (Cattafesta III and Sheplak, 2011). The action of the actuators is modulated by policies aimed at optimizing the performance and the dynamic response in a prescribed manner (open-loop) or as a function of some observations of the system at hand (closed-loop). The identification of these policies based on measurements and models of the physics is the objective of the control design.

In principle, AFC strategies optimize the flow in real time; in practice, these techniques are mostly used in limited numerical and experimental test cases. Indeed, approximations based on reduced-order models of the physical system can critically lose accuracy when control is applied, resulting in poor performance and lack of robustness. Addressing these challenges using machine learning (ML) tools (Brunton et al., 2020) and, more specifically, reinforcement learning (RL) can be a key factor in extending flow control to realistic cases by circumventing models limitations or lack of robustness due to their data-driven nature. A possible definition based on the main goals of RL is the following:

> *"RL studies how to use past data to enhance the future manipulation of a dynamical system"* (Recht, 2019).

The description applies equivalently to standard control theory. This suggestive similarity in definition and purpose is not coincidental: the common roots of RL and modern control theory can be found in dynamic programming (DP), a nonlinear optimization protocol based on the Bellman equation (Bellman, 1958). Starting from DP, these two disciplines evolved in parallel in the last decades, leading to the co-development of different approaches to similar problems (Bertsekas, 1995; Sutton and Barto, 2018). The solution of the Bellman equation is the value function, a nonlinear function, which is related to a score associated with a given action or controlled trajectory. Once the value function is known, one can determine the optimal policy. However, the Bellman equation results to be computationally impractical in most cases, even when models are available or direct methods applied. For this reason, a large body of literature is dedicated to the numerical approximation of the Bellman equation and the iterative schemes used for its solution. Iterative methods are particularly interesting as they can be applied with and without a prior model at hand: from a theoretical viewpoint these are the premises on the top of which RL algorithms stand (Sutton and Barto, 2018).

RL algorithms are iterative, data-driven and solely relying on limited measurements; models are completely replaced or updated during the iterative process by exploration: the state space of the system is learnt by using past data extracted from the measurements and the interactions of the system or agent with the environment. The set of all the

actions the agent can act out in an environment is called action space and a score is assigned to each action for the value function evaluation. In the limit of full knowledge of this space, the resulting policy is optimal if the Bellman equation associated with the value function is fulfilled. In that sense, RL is inspired by nature as it tries to mimic the process of learning of living beings.

A further ingredient is represented by artificial neural networks (ANN). The success of ML applications in very diverse fields, ranging from computer vision and natural language processing to medical diagnosis, is mainly due to the versatility of ANN and their effectiveness in supervised and unsupervised learning (Goodfellow et al., 2016). From a mathematical viewpoint, their versatility is motivated by their properties of universal approximators of nonlinear functions: the combination of ANN, for the approximation of the policy and the value function, with RL led to the Deep RL (DRL) framework. The first application of ANN in RL is often credited to the work by Tesauro (1994), who developed a program – TD-Gammon – combining temporal difference and ANN to play backgammon. In the same years, the application of ANN in combination with dynamic programming was discussed in seminal works on the subject by Bertsekas (1995), under the name of neuro-dynamic programming; the recent developments in the field of deep learning and the super–human performance achieved by DRL in solving games such as `go` and `shogi` (Silver et al., 2017) boosted the popularity of the approach. Together with the vast availability of open-source packages, this is also one of the reasons why DRL is often seen "only" as one of the main subfields of ML and used as a black-box tool. However, this limited perspective risks being rather simplistic: RL is well grounded in optimal control theory, and the interaction between these two disciplines could play a key role in future technological challenges such as the development of driverless cars, self-supervised learning or flow control.

1.1 Standard approaches in flow control: a brief overview

In the following, we introduce a brief, non-exhaustive overview of active flow control. From a physical point of view, the range of applications is as broad as the cases in which the presence of a fluid impacts the efficiency or performance of the dynamical system under investigation; among the examples we can cite, control mechanisms range from quenching the instabilities responsible for the transition to turbulence at relatively low Reynolds numbers (Sipp and Schmid, 2016) to the modification of the mean-flow or of the large scale structures for turbulent cases (Kühnen et al., 2018).

The DP framework provides the theoretical ground for generalizing the optimal control problem from linear to nonlinear cases (Bertsekas, 1995, 2019). A special case is the linear quadratic regulator (LQR), a standard solution of optimal control which can be derived directly from the Bellman equation and reduces to the algebraic Riccati equation in the linear, steady limit (Lewis et al., 2012); when the LQR is combined with optimal estimators, we obtain the linear quadratic Gaussian (LQG). In these hypotheses, LQR/LQG controllers are an ideal benchmark for assessing the optimality of the policies. In flow control, examples can be found in Högberg and Henningson (2002); Högberg et al. (2003); Chevalier et al. (2007). In alternative to the direct methods, one can resort to the adjoint-based formulation in the same linear/linearized limit (Luchini and Bottaro, 2014); the latter can be extended to nonlinear cases and model predictive con-

trollers (MPC) (Glad and Ljung, 2000; Bewley et al., 2001; Xiao and Papadakis, 2019) or adaptive controllers (Åström and Wittenmark, 2008). These techniques have been widely used in fluid mechanics and require for the control design a physical model, describing the behavior of the system[1].

When a physical model is available, as in the case of fluid mechanics, solving the governing equations can be too slow with respect to the dynamics at play to be useful, if not even unfeasible: for instance, direct computations of LQR controllers are limited by the degrees of freedom n that cannot exceed $n \approx 10^4$, unless resorting to iterative methods (Semeraro et al., 2013b). Alternatively, one can reduce the problem's dimensionality by identifying suitable low-order models that preserve the system's dynamics for control design while meeting computational and real-time constraints. In fluid mechanics, model reduction and system identification enjoyed widespread popularity in the last two decades, with applications ranging from balance truncation (Rowley, 2005; Ma et al., 2011) to system identification (Ljung, 1999; Noack et al., 2011; Hervé et al., 2012) or subspace iteration methods (Van Overschee and De Moor, 2012; Juillet et al., 2014). Review works can be found in Kim and Bewley (2007); Bagheri et al. (2009); Brunton and Noack (2015); Sipp and Schmid (2016); Rowley and Dawson (2017). Despite this large amount of applications, low-order models are prone to decay of performance and lack of robustness with respect to the modeling uncertainties or change of flow parameters; also, estimation and low-observability deteriorate the performance of model-based controllers when more realistic, experimental setups characterized by noise in the measurements or localized/geometrically-constrained sensors are considered. For example, we can consider the delay of laminarâĂŞturbulent transition, generally tested within the limit of small amplitudes or in highly controlled wind-tunnel tests (Semeraro et al., 2013a; Fabbiane et al., 2015). Higher amplitudes are characterized by nonlinearities limiting model-based AFC, although some improvements can be achieved by using adaptive filters on relatively slow time-scale (Sturzebecher and Nitsche, 2003; Fabbiane et al., 2014), robust control (Bewley et al., 2000; Leclercq et al., 2019) or model-predictive control (Losse et al., 2011; Goldin et al., 2013; Arbabi et al., 2018; Marra et al., 2024).

1.2 Reinforcement learning in flow control

Some of the limitations of model-based controllers can be circumvented by data-driven tools, being inherently based on measurements and past data taken from local sensors to identify control policy without relying on *a-priori* physical model. Among the early works on flow control using RL, those by Guéniat et al. (2016); Rabault et al. (2019, 2020); Fan et al. (2020) are worth mentioning. These works are representative of very different approaches within RL and allow us to quickly glance at the large variety of available techniques. In Rabault et al. (2019), an actor-based method is applied for the control of a flow developing past a cylinder using proximal policy optimization (PPO). In PPO, parametric policies are evaluated by registering the system for a long period of time and calculating the cumulative reward; optimization is performed by stochastic gradient-descent algorithms to update the policy. PPO is thus an on-policy, gradient-descent policy

[1]Note: the term *model* will refer only to *physical* models, including low-order ones. In contrast, in the ML community, the term defines approximations or parameterized functions, such as input-output relations mediated by ANN.

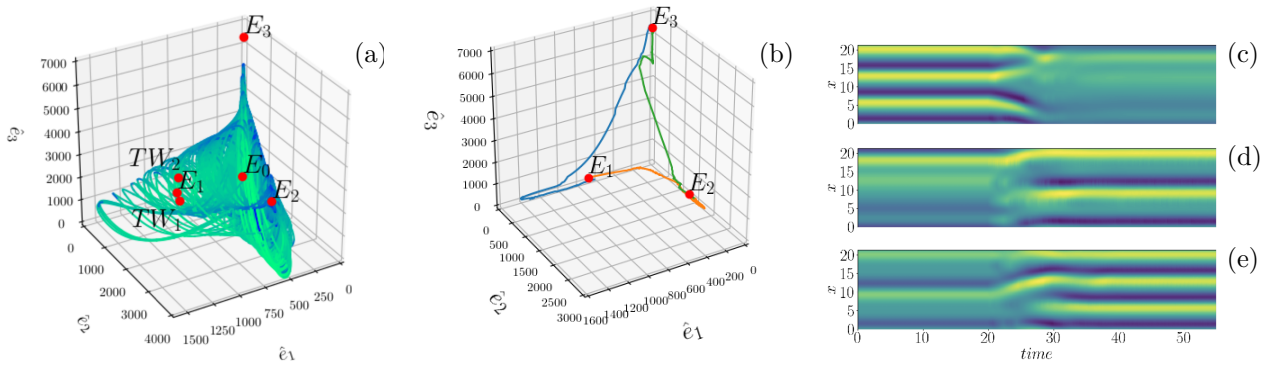

Figure 1: Example of a nonlinear dynamical system controlled using localized actuators, localized sensors, and an actor-critic algorithm (see Sec. 3); the figure is adapted from Bucci et al. (2019). In (a), the uncontrolled dynamics governed by the Kuramoto-Sivashinsky equation is shown in phase-space, by projecting on the first three Fourier modes ($\hat{e}_i$). Red dots indicate the 4 unstable equilibria (E) and 2 travelling waves (TW) characterizing the dynamics when the domain length $L = 22$ (Cvitanović et al., 2010). Chaotic behavior is observed. In (b), the system controlled by RL is shown: three policies are computed, driving the system towards each nontrivial equilibrium. The controlled trajectories are shown in the spatio-temporal plots (c)-(d)-(e) for $E_3 \rightarrow E_1$, $E_1 \rightarrow E_2$, and $E_2 \rightarrow E_3$, respectively.

method. A prototype of this class of strategies is the `reinforce` algorithm; the resulting solution in these cases does not satisfy Bellman's optimality principle. A dual approach is to approximate the state-value-action function or $\mathcal{Q}$-function (Watkins and Dayan, 1992). In these approaches, usually termed value-based, a critic is introduced such that Bellman optimality is guaranteed if the system analyzed is Markovian and completely known from observables. An early application based on $\mathcal{Q}$-learning was presented for fluid control by Guéniat et al. (2016). A disadvantage of the critic algorithms is the discrete representation of the action, in case a continuous system is considered. For this reason, these two philosophies are often found combined, resulting in a number of different solutions. One example is the previously mentioned Proximal Policy Optimization (PPO) algorithm, that incorporates a critic part estimating the value of the policy. The critic is trained using the cumulative rewards computed along the trajectories, but it does not intervene directly in the process of policy learning. Instead, the critic helps to refine the policy by providing feedback on its performance. On the other hand, algorithms where the critic is based on $\mathcal{Q}$-learning can be coupled with an actor part, in order to learn a continuous policy. A recent experimental example is provided by the work of Fan et al. (2020). With a similar strategy, in Bucci et al. (2019) an actor-critic RL algorithm, the Deep Deterministic Policy Gradient (DDPG) (Silver et al., 2014; Lillicrap et al., 2015) is used for the control of the nonlinear dynamics governed by the Kuramoto-Sivashinsky equation, without prior knowledge of the system. The work was among the first demonstrations of control of chaotic dynamical systems by RL. An overview of the results is shown in Fig. 1: the highly nonlinear dynamics (a) is stabilized (b) around unstable fixed points $(c$-$e)$ only by relying on limited measurements and localized equispaced actuators. In principle, the

combination of actor and critic allows an approximation of the policy and ensures that the function is a solution of the Bellman equation when Markiovanity conditions are met.

Besides optimal control of fluid flows, RL has also attracted attention in a variety of applications ranging from bio-mimetic optimization to numerical analysis. Among the numerous contributions available in the literature, it is worth mentioning the early work by Vergassola et al. (2007) on infotaxis as a navigation strategy without gradients; this landmark work gave the impulse to numerous works inspired by bio-mimetics where RL is used for the control of gliding or perching (Reddy et al., 2018; Novati et al., 2019), optimal swimming (Verma et al., 2018; Borra et al., 2022) and point-to-point navigation (Biferale et al., 2019). Shape optimization has been discussed in the works by Viquerat et al. (2021); Keramati et al. (2022); Dussauge et al. (2023). From the numerical analysis viewpoint, a recent trend in turbulence modeling consists of automating the closure term by multi-agent RL (Novati et al., 2021), including wall models applications (Bae and Koumoutsakos, 2022). Adaptive mesh applications through RL have been recently proposed by Yang et al. (2023) and Foucart et al. (2023).

The interested reader can find extensive reviews on data-driven control in fluid mechanics in Brunton and Noack (2015); Brunton et al. (2020). A recent review of RL in fluid mechanics was authored by Vignon et al. (2023). Finally, it is worth to mention the work by Pino et al. (2023), where a comparative analysis of ML methods for active flow control is proposed, focusing in particular on genetic programming and RL, benchmarked against global optimization techniques such as Bayesian and Lipschitz optimizations.

1.3 Organization of the chapter

The remainder of the chapter is organized as follows. In Sec. 2, we focus on the Hamilton-Jacobi-Bellman (HJB) equations and DP for nonlinear optimal control. The HJB equation is a classical result that extends the linear optimal control to nonlinear cases (Sec. 2.3). The discrete counterpart of the HJB equation – the Bellman equation – is at the heart of DP (Sec. 2.5) and RL. Iterative solutions to the Bellman equation – namely, the value iteration and policy iteration – are introduced in Sec. 2.6. To avoid an abstract discussion of these methods, we introduce the linear limits as an illustration. The chapter closes with a section where the introduced elements are reconciled with the terminology of RL practice in Sec. 3 and a short bibliography in Sec. 4.

2 From nonlinear control to reinforcement learning

In this section, we state the optimal control problem for linear and nonlinear systems, showing how the Hamilton-Jacobi-Bellman equation can be used for generalizing the results obtained in the linear framework. The discrete counterpart – the Bellman equation – is analyzed in the second step. The solution of the Bellman equation is at the heart of dynamic programming. Direct and basic iterative solutions are quickly described. Finally, reinforcement learning is introduced.

2.1 Nomenclature and state space representation

Due to numerous commonalities characterizing the two frameworks, it is interesting to introduce the basic nomenclature of control theory and RL in parallel. We start by considering the simplified block diagram in Fig. 2; this sketch is adapted from a block-diagram in the book by Sutton and Barto (2018), but it is analogous to many other diagrams that it is possible to find in control books. The sketch depicts a closed-loop system where a feedback is introduced for the control of the plant; by definition, the plant is the system to be controlled equipped with a set of actuators and sensors. Sensors detect the system and, in the case of flow control, velocity or pressure probes can be used for this purpose. Actuators introduce energy into the system through volume forcing (such as plasma actuators), blowing and suction jets or active surfaces (Cattafesta III and Sheplak, 2011).

From an analytical viewpoint, a plant corresponds to the open-loop input-output system. We consider here a time-continuous, space-discrete representation and introduce the state vector $\mathbf{x}(t) \in \mathbb{R}^n$, with n the number of spatial degrees of freedom of the system. We thus introduce a nonlinear system denoted by $\mathcal{F} : \mathbb{R}^n \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \rightarrow \mathbb{R}^n$; the state-space representation can be written as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathcal{F}(\mathbf{x}, u, d, t) \\ z(t) &= \mathbf{C}_1\mathbf{x}(t) + \nu_z \\ y(t) &= \mathbf{C}_2\mathbf{x}(t) + \nu_y. \end{aligned} \tag{1}$$

The input $u(t) \in \mathbb{R}^{n_u}$ is the control signal and it is the unknown of the control problem, with n_u entries (multi-input). The noise process or disturbance $d(t) \in \mathbb{R}^{n_d}$ is regarded as an input or forcing on the system. The outputs are $y(t) \in \mathbb{R}^{n_y}$ and $z(t) \in \mathbb{R}^{n_z}$. The sensors are modeled using the operators by $\mathbf{C}_1 \in \mathbb{R}^{n_z \times n}$ and $\mathbf{C}_2 \in \mathbb{R}^{n_y \times n}$, assuming both relations to be linear, and in both cases an additive noise process is added for modeling errors or noise in the measurements.

So far, we have assumed the most general situation where the governing equations and the interactions with control inputs are nonlinear. However, special cases are also possible. A first case consists of a linear description $\mathcal{B}_u(\mathbf{x})$ of the actuation as a function of $\mathbf{x}(t)$, written in the form

$$\dot{\mathbf{x}}(t) = \mathcal{F}(\mathbf{x}, d, t) + \mathcal{B}_u(\mathbf{x})u(t).$$

When the system can be modeled by means of linearization around the equilibrium points of the state space (also denoted as singular or stationary points), or approximated by a

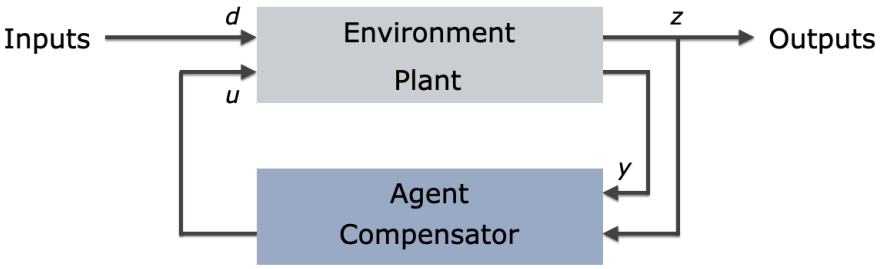

Figure 2: Simplified block diagram showing a closed loop. The system to be controlled is defined as plant or environment following control or RL terminology, respectively. The system is equipped with inputs and outputs. The outputs are $y(t)$ and $z(t)$. A feedback law is computed starting from $y(t)$ in order to minimize or maximize the measurements $z(t)$: in optimal control, we define it as a cost function; in RL is mostly found as a reward or value function. A compensator is designed to compute a control input $u(t)$ based on outputs. In RL, the compensator is called agent. Finally, the disturbance input $d(t)$ drives the system's dynamics as input noise or external forcing.

linear time-invariant model, the linear mapping is computed as $\mathbf{A} = \nabla_{\mathbf{x}}\mathcal{F}|_{x=\bar{x}}$, where $\bar{\mathbf{x}}$ denotes one of these points and $\mathbf{A} \in \mathbb{R}^{n\times n}$, assuming $u = 0$. Similarly, we obtain a representation for the actuator as $\mathbf{B}_u = \nabla_u \mathcal{F}|_{u=\bar{u}}$ at $\mathbf{x} = \bar{\mathbf{x}}$, where $\mathbf{B}_u \in \mathbb{R}^{n\times n_u}$. The resulting state equation reads as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}_u u(t) + \mathbf{B}_d d(t). \tag{2}$$

where we introduce $\mathbf{B}_d \in \mathbb{R}^{n\times n_d}$ as a model mimicking the external disturbances. The plant can be a simplified model of the environment or coinciding with that. In flow control, it is the case when Linearized Navier-Stokes equations (the plant) are used to control a nonlinear flow (the environment). The main aim of control design and RL is maximizing or minimizing quantities of interest described by a performance index. According to the context, the quantities of interest of the optimization are denoted as cost function when minimized, reward when maximize or value function. Classic quantities of interest in fluid mechanics correspond to measurements chosen for maximizing lift, reducing draft, or mitigating noise emissions. The optimization is achieved by inferring efficient and robust control strategies or policies, modulating the action of the actuators on the system. In control, the compensator provides this action; the compensator consists typically of an estimator, used for reconstructing the state based on the estimation measurements and/or augmenting its expressivity, and the controller indicated by the feedback control law $\mathbf{K} \in \mathbb{R}^{n_u\times n}$, providing the input signal based on the reconstructed state approximating the full state such that $\hat{\mathbf{x}}(t) \approx \mathbf{x}(t) \in \mathbb{R}^n$, and $u(t) = \mathbf{K}\hat{\mathbf{x}}(t)$. Note that it is not necessary to consider a full-order estimated state, if the input-output behaviour of the system is well represented by a reduced-order model and its associated state. In this case the reconstructed state can be $n_r \ll n$ and $u(t) = \mathbf{K}_r\hat{\mathbf{x}}_r(t)$, with $\mathbf{K}_r \in \mathbb{R}^{n_u\times n_r}$. A further reduction is obtained when an output feedback control is considered: in this case the control law reduces to the transfer function $u(t) = \mathbf{K}_y y(t)$, with $\mathbf{K}_y \in \mathbb{R}^{n_u\times n_y}$ (Lewis et al., 2012). In RL, the compensator roughly corresponds to the agent, while the policy

corresponds to the controller. Indeed, from a technical viewpoint, the policy is a nonlinear function $u(t) = \pi(y(t))$ based on the measurements $y(t)$.

In summary, one can say that the agent is the compensator, while the pairs environment-plant and policy-controller can be used alternatively according to the contexts but defined similarly.

2.2 Optimal control in linear system: the Riccati equation

We can now introduce the optimal control problem by considering the time-invariant, continuous linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_u u, \tag{3}$$

already introduced in Sec. 2.1. For improved readability, the time dependence of the variables is omitted except where necessary for clarity. We aim at identifying a control law as a function of time $u(t) \in \mathbb{R}^{n_u}$ such that an associated performance index or cost function is minimized. The cost function is chosen to be quadratic and reads

$$\mathcal{J}(\mathbf{x}(t_0), t_0) = \frac{1}{2}\mathbf{x}(T)^T \mathbf{P}_T \mathbf{x}(T) + \frac{1}{2}\int_{t_0}^{T} \left(\mathbf{x}^T \mathbf{Q}\mathbf{x} + u^T \mathbf{R} u\right) dt. \tag{4}$$

The terminal cost at time T is defined by the quadratic form $\mathbf{x}^T\mathbf{P}_T\mathbf{x}$, with $\mathbf{P}_T \geq 0$ positive semi-definite. The second term is an integral in time where the integrand is a sum of two terms: an energy associated with the state vector $\mathbf{x}(t)$ and a norm of the control signal $u(t)$, with $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$ the respective weights determining the relative penalties. To ease the discussion, without loss of generality, we choose a null terminal state, *i.e.* $\mathbf{x}(T) = 0$. We aim at determining the optimal signal $u^*(t)$ defined in the interval $[t_0, T]$. An optimal quantity will be indicated with $(^*)$ in the following whenever needed.

The solution of the control problem can be obtained by introducing the following Hamiltonian function

$$\mathcal{H} = \frac{1}{2}\left(\mathbf{x}^T \mathbf{Q}\mathbf{x} + u^T \mathbf{R} u\right) + \lambda^T \left(\mathbf{A}\mathbf{x} + \mathbf{B}_u u\right), \tag{5}$$

where the first term is associated with the cost function, while the second term is the constraint represented by the dynamical system under consideration, and the state $\lambda \in \mathbb{R}^n$ is the co-state or adjoint state. Minimization is performed by considering the gradients of $\mathcal{H}$ with respect to the parameters of the problem

$$\dot{\mathbf{x}} = \frac{\partial \mathcal{H}}{\partial \lambda} \quad \rightarrow \quad \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}_u u \qquad \textit{with} \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{6}$$

$$-\dot{\lambda} = \frac{\partial \mathcal{H}}{\partial \mathbf{x}} \quad \rightarrow \quad \dot{\lambda} = -\mathbf{A}^T\lambda - \mathbf{Q}\mathbf{x} \quad \textit{with} \quad \lambda(T) = 0, \tag{7}$$

$$0 = \frac{\partial \mathcal{H}}{\partial u} \quad \rightarrow \quad 0 = \mathbf{R}u + \mathbf{B}_u^T \lambda. \tag{8}$$

By zeroing the last relation, we obtain the optimal control law

$$u^* = -\mathbf{R}^{-1}\mathbf{B}_u^T \lambda.$$

Note that Eq. 7 is the equation associated with the adjoint state and is marched backward in time. The iterative solution of the resulting system of equations provides the optimal solution $u(t)$ in Eq. 8, which is minimizing since $\partial^2\mathcal{H}/\partial u^2 = \mathbf{R} > 0$.

In alternative, one can cast the equations in Eq. 6-8 into an algebraic equation: the control algebraic Riccati equation (CARE). To this end, we assume that $\lambda(t) = \mathbf{P}\mathbf{x}(t)$, with $\mathbf{P} \geq 0$; this procedure is called sweep method (Lewis et al., 2012). The last relation can be derived in time and the relations Eq. 6-7 plugged in

$$\begin{aligned} \dot{\lambda} &= \dot{\mathbf{P}}\mathbf{x} + \mathbf{P}\dot{\mathbf{x}} \\ -\mathbf{A}^T\lambda - \mathbf{Q}\mathbf{x} &= \dot{\mathbf{P}}\mathbf{x} + \mathbf{P}\left(\mathbf{A}\mathbf{x} + \mathbf{B}_u u\right). \end{aligned}$$

The relation can be further manipulated by introducing the optimal solution Eq. 8 and the relation $\lambda(t) = \mathbf{P}\mathbf{x}(t)$

$$-\dot{\mathbf{P}}\mathbf{x} = \mathbf{A}^T\mathbf{P}\mathbf{x} + \mathbf{P}\mathbf{A}\mathbf{x} - \mathbf{P}\mathbf{B}_u\mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P}\mathbf{x} + \mathbf{Q}\mathbf{x}. \tag{9}$$

In the stationary case, $\dot{\mathbf{P}} = 0$; since the solution is valid $\forall\mathbf{x}$, the resulting CARE reads

$$0 = \mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}_u\mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P} + \mathbf{Q}. \tag{10}$$

The solution $\mathbf{P}$ enables to compute the control gain $\mathbf{K}$: it is sufficient to consider

$$u^* = -\mathbf{R}^{-1}\mathbf{B}_u^T\lambda = -\mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P}\mathbf{x}, \tag{11}$$

and define the feedback gain as $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P}$. This is the solution to the linear quadratic regulator (LQR) problem.

In perfect analogy, it is possible to define an optimal estimation problem, where we aim at minimizing the following error norm

$$\min \|y(t) - \hat{y}(t)\|, \tag{12}$$

where the measurement $\hat{y}(t)$ is associated with the estimated state $\hat{\mathbf{x}}(t)$. The combination of an optimal estimator and the LQR leads to the linear quadratic Gaussian compensator (LQG). For the separation principle, the LQG compensator is optimal (Glad and Ljung, 2000).

Further 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 is often solved instead, see for instance Bewley et al. (2016), Luchini and Bottaro (2014), Semeraro et al. (2013b) and citations therein. Additionally, optimization based on a finite sliding temporal window leads to the development of MPC (see e.g. Bewley et al., 2001). Another aspect that can be noted is on the modeling of disturbances. The LQR problem is defined based on the linear time-invariant in Eq. 3 and solved using the CARE in Eq. 10; the CARE is written using the state matrix $\mathbf{A}$, the model for the actuation $\mathbf{B}_u$, and the metric related to the cost function. In principle, a full-order, perfect model does not require any assumption on the disturbances driving or affecting the system. In practice, this becomes a critical part of the estimation process, particularly in accounting for model uncertainties. Within control theory, robust control is a significant field dedicated to addressing these limitations. Interestingly, Riccati equations can be also cast for robust control synthesis (Glad and Ljung, 2000; Zhou et al., 2002). As previously mentioned, our objective here is to tackle these challenges through data-driven solutions.

2.3 The Hamilton-Jacobi-Bellman equation

We can now introduce informally the Hamilton-Jacobi-Bellman (HJB) equation, a nonlinear partial differential equation that is at the heart of the optimal control problem in nonlinear systems and provides necessary and sufficient conditions for the optimality. Its solution is the optimal cost function. Once the cost function is known, the optimal control or policy can be computed by maximization or minimization. In this sense, we shift the focus from determining the optimal policy to the computation of the optimal cost function. This dual viewpoint characterizes also the RL applications, where the cost function is typically referred to as value function.

We consider the plant in Eq. 1 and assume the state to be fully known and not recorded from the measurements. We define the cost function through the following performance index

$$\mathcal{J}(\mathbf{x}(t_0), t_0) = h(T, \mathbf{x}(T)) + \int_t^T r(\mathbf{x}, u) d\tau, \tag{13}$$

where we are interested in determining the optimal control $u^*(t)$ in the interval $t \in [t_0, T]$. As t is the current time, a dummy variable τ is introduced in the integral. The integrand of the second term denoted $r(\mathbf{x}, u)$ is the reward function. The first term is a function associated with the terminal condition at final time T. The cost-to-go $\mathcal{J}(\mathbf{x}, t)$ can be rewritten as

$$\begin{aligned} \mathcal{J}(\mathbf{x}, t) &= \int_t^{t+\Delta t} r(\mathbf{x}, u) d\tau + \int_{t+\Delta t}^T r(\mathbf{x}, u) d\tau + h(T, \mathbf{x}(T)) \\ &= \int_t^{t+\Delta t} r(\mathbf{x}, u) d\tau + \mathcal{J}(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t), \end{aligned} \tag{14}$$

where $t + \Delta t$ is a future time close to t. For optimizing the cost-to-go $\mathcal{J}(\mathbf{x}, t)$, the integral has been broken into two terms: the first associated with the reward computed in the interval $[t, t + \Delta t]$, and a second term associated with the future rewards, appropriately replaced by the future cost function. The last expression describes all the possible values for the cost from time t to the time horizon T; however, if we know the optimal cost for all the possible states $\mathbf{x} + \Delta\mathbf{x}$ in the time interval, we can cast the following minimization

$$\mathcal{J}^*(\mathbf{x}, t) = \min_{\substack{u(\tau) \\ t \le \tau \le t+\Delta t}} \left[\int_t^{t+\Delta t} r(\mathbf{x}, u, t) d\tau + \mathcal{J}^*(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) \right]. \tag{15}$$

The expression can be manipulated by performing the integral corresponding to the instantaneous reward and introducing a Taylor series for the $\mathcal{J}^*(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)$. The resulting equation reads

$$\mathcal{J}^*(\mathbf{x}, t) = \min_{\substack{u(\tau) \\ t \le \tau \le t+\Delta t}} \left[r\Delta t + \mathcal{J}^*(\mathbf{x}, t) + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T \Delta\mathbf{x} + \frac{\partial \mathcal{J}^*}{\partial t} \Delta t \right]. \tag{16}$$

By substituting a first-order approximation $\Delta\mathbf{x} = \mathcal{F}\Delta t$, and observing that the second and fourth terms are not affected by the minimization, we obtain

$$\cancel{\mathcal{J}^*(\mathbf{x}, t)} = \min_{\substack{u(\tau) \\ t \le \tau \le t+\Delta t}} \left[r\Delta t + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T \mathcal{F}\Delta t \right] + \cancel{\mathcal{J}^*(\mathbf{x}, t)} + \frac{\partial \mathcal{J}^*}{\partial t} \Delta t \tag{17}$$

that, for $\Delta t \to 0$, results in the HJB equation

$$-\frac{\partial \mathcal{J}^*}{\partial t} = \min_{u(t)} \left[r(\mathbf{x}, u) + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T \mathcal{F}(\mathbf{x}, u) \right]. \tag{18}$$

The result of this equation is the optimal $\mathcal{J}^*$ as a function of time. Note that it is solved backward in time, as in the adjoint problem underneath the optimal control problem in the linear limit.

A problem with the HBJ equation is the difficulty of computing a solution since it is a nonlinear partial differential equation, where we assume the solution $\mathcal{J}^*$ to be a differentiable function. When it is possible to solve the HJB equation, the associated policy π can be obtained by optimization. Among the possible techniques that one can employ, we could consider the expansion in power series of the functions $\mathcal{F}$ and $\mathcal{J}$, if these are real analytic close to the origin. Such a strategy allows solving the HJB in terms of the power series coefficients (Glad and Ljung, 2000), starting from the linear order, similarly to the weakly nonlinear expansions.

2.3.1 From the HJB equation to the Riccati equation in linear plants

In the previous section, we have shown that the HJB equation determines the optimal cost function for a nonlinear dynamical system with an associated reward. It is possible to show that the Eq. 18 in the linear limit is directly related to the class of optimal control problems discussed in Sec. 2.2.

We consider the linear plant in Eq. 3 associated with a quadratic cost function Eq. 4. The HJB is rewritten as following

$$-\frac{\partial \mathcal{J}^*}{\partial t} = \min_{u(t)} \left[\frac{1}{2} \left(\mathbf{x}^T \mathbf{Q} \mathbf{x} + u^T \mathbf{R} u \right) + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T (\mathbf{A}\mathbf{x} + \mathbf{B}_u u) \right]. \tag{19}$$

By minimizing the right-hand side, we get the optimal u^* in the following form

$$u^* = -\mathbf{R}^{-1} \mathbf{B}_u^T \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T. \tag{20}$$

By substituting it into the HJB equation, we obtain

$$\begin{aligned} -\frac{\partial \mathcal{J}^*}{\partial t} &= \left[\frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T \mathbf{A}\mathbf{x} \right] + \left[\frac{1}{2} (u^*)^T \mathbf{R} u^* + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T \mathbf{B}_u u^* \right] \\ &= \left[\frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T \mathbf{A}\mathbf{x} \right] - \frac{1}{2} \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right) \mathbf{B}_u \mathbf{R}^{-1} \mathbf{B}_u^T \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}} \right)^T. \end{aligned}$$

Without loss of generality, we set the terminal condition to be null at $t = T$. We now follow the sweep method and assume there exists a solution $\mathbf{P}(t)$ for $t < T$ and use it for writing the cost function as a quadratic form. By plugging in this position in the HJB, we get

$$0 = \frac{1}{2} \mathbf{x}^T \dot{\mathbf{P}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{x}^T \mathbf{P} \mathbf{A} \mathbf{x} - \frac{1}{2} \mathbf{x}^T \mathbf{P} \mathbf{B}_u \mathbf{R}^{-1} \mathbf{B}_u^T \mathbf{P} \mathbf{x}. \tag{21}$$

where the derivatives of $\mathcal{J}$ with respect to t and $\mathbf{x}$ are now written using $\mathbf{P}$. The resulting expression can be further manipulated by

- replacing the third term as $2\mathbf{PA} = \mathbf{PA} + \mathbf{A}^T\mathbf{P}$;
- observing that this relation holds for any trajectory $\mathbf{x}(t)$ emanating from all the possible initial conditions $\mathbf{x}(t_0)$;
- assuming the state to be steady.

Following these three steps, the Riccati equation is finally obtained as

$$0 = \dot{\mathbf{P}} + \mathbf{Q} + \mathbf{PA} + \mathbf{A}^T\mathbf{P} - \mathbf{PB}_u\mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P}, \tag{22}$$

with optimal signal $u^*(t) = -\mathbf{R}^{-1}\mathbf{B}_u^T\mathbf{P}\mathbf{x}(t)$. Interestingly, we can observe that the adjoint state used in the Hamiltonian formulation adopted in the linear case corresponds to the term $\left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}}\right)^T$. In other words, in the linear limit, the following equivalence holds

$$\lambda(t) = \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}}\right)^T = \mathbf{P}\mathbf{x}(t), \tag{23}$$

that allows us to stress once again the relation with adjoint solutions.

2.3.2 A quick observation on the Hamiltonian function

In the previous section, we observed that the HJB written for the linear plant in Eq. 3 reads as

$$-\frac{\partial \mathcal{J}^*}{\partial t} = \min_{u(t)} \left[\frac{1}{2}\left(\mathbf{x}^T\mathbf{Q}\mathbf{x} + u^T\mathbf{R}u\right) + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}}\right)^T (\mathbf{A}\mathbf{x} + \mathbf{B}_u u)\right]. \tag{24}$$

The reader may note that on the right-hand side we have the Hamiltonian function used in Eq. 5. This is not a coincidence, and it can be shown formally by reviewing the formulation of the optimal control problem in variational form. Further analysis are beyond the scope of this chapter, but it is interesting to observe that also for the nonlinear cases the HJB can be rewritten as

$$-\frac{\partial \mathcal{J}^*}{\partial t} = \min_{u(t)} \left[\mathcal{H}(\mathbf{x}, u, \mathcal{J}^*, t)\right] \tag{25}$$

where

$$\mathcal{H}(\mathbf{x}, u, \mathcal{J}^*, t) = r(\mathbf{x}, u) + \left(\frac{\partial \mathcal{J}^*}{\partial \mathbf{x}}\right)^T \mathcal{F}(\mathbf{x}, u). \tag{26}$$

From the physical viewpoint, the Hamiltonian function is associated to the energy content along the trajectories. The HJB equation requires that the Hamiltonian value is minimized given a prescribed policy. For time-invariant systems, the Hamiltonian function is constant along an optimal trajectory and for the differential form $\dot{\mathcal{H}} = \mathcal{H}_t = 0$.

2.4 Discrete systems

In the following sections, the introduced algorithms are formulated in a time-discrete setting. Considering a time-discrete variable $a(t)$, the corresponding discrete formulation is

$$a_k = a(k\Delta t), \qquad k = 1, 2, ... \tag{27}$$

where the sampling period Δt is constant. Accordingly, the time-discrete state-space system is defined as

$$\begin{aligned} \mathbf{x}_{k+1} &= \tilde{\mathbf{A}}\mathbf{x}_k + \tilde{\mathbf{B}}_u u_k + \tilde{\mathbf{B}}_d d_k && (28)\\ y_k &= \tilde{\mathbf{C}}_2\mathbf{x}_k + (\nu_y)_k && (29)\\ z_k &= \tilde{\mathbf{C}}_1\mathbf{x}_k + (\nu_z)_k, && (30) \end{aligned}$$

where $\tilde{\mathbf{A}} = \exp(\mathbf{A}\,\Delta t)$, $\tilde{\mathbf{B}}_{d,u} = \mathbf{B}_{d,u}\Delta t$ and $\tilde{\mathbf{C}}_{1,2} = \mathbf{C}_{1,2}$, while the remaining quantities are the time-discrete counterparts of the quantities previously introduced as time-continuous. For more details, the interested reader can refer to any control book (see e.g. Glad and Ljung, 2000). In the following, the $(\tilde{\cdot})$ will be dropped to ease the readability.

2.4.1 The linear quadratic regulator (LQR) in discrete systems

As already done in continuous formulation, the linear optimal control is introduced as a benchmark. We consider the cost function in discrete form as

$$\mathcal{J}(\mathbf{x}_k) = \frac{1}{2}\sum_{i=k}^{N-1}\left(\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i + u_i^T\mathbf{R}u_i\right) + \frac{1}{2}\mathbf{x}_N^T\mathbf{P}_N\mathbf{x}_N, \tag{31}$$

and the plant in Eq. 30. By following the sweep method and the procedure adopted in the continuous case in Sec. 2.2, it can be shown that the optimal control relation reads

$$u_k = -\mathbf{K}\mathbf{x}_k = -(\mathbf{B}_u^T\mathbf{P}\mathbf{B}_u + \mathbf{R})^{-1}\mathbf{B}_u^T\mathbf{P}\mathbf{A}\mathbf{x}_k, \tag{32}$$

with the corresponding discrete algebraic Riccati equation written as

$$0 = \mathbf{Q} - \mathbf{P} + \mathbf{A}^T\mathbf{P}\mathbf{A} - \mathbf{A}^T\mathbf{P}\mathbf{B}_u(\mathbf{B}_u^T\mathbf{P}\mathbf{B}_u + \mathbf{R})^{-1}\mathbf{B}_u^T\mathbf{P}\mathbf{A}. \tag{33}$$

Also in this case, $\mathbf{P} \in \mathbb{R}^{n\times n}$ is a symmetric matrix, positive semi-definite.

2.5 Bellman's principle of optimality

In parallel to what was done before, we consider now a nonlinear, discrete system introducing the index k for the time sampling

$$\mathbf{x}_{k+1} = \mathcal{F}(\mathbf{x}_k) + \mathcal{B}_u(\mathbf{x}_k)u_k. \tag{34}$$

Note that we consider a system where the actuation is modeled using linear relations. The performance index to be minimized is

$$\mathcal{J}_\pi(\mathbf{x}_k) = h(N, \mathbf{x}_N) + \gamma\sum_{i=k}^{N-1} r(\mathbf{x}_i, u_i). \tag{35}$$

The cost function $\mathcal{J}_\pi$ is the criterion to be minimized with respect to the policy π. Note that the second term is a sum over the rewards associated with each state $\mathbf{x}_k$ under control action u_k; also, a parameter is introduced, the discount factor $\gamma \in (0, 1)$, for guaranteeing convergences of the series. In analogy to the continuous case, we rewrite the performance

index by neglecting the final cost – without loss of generality – and separating the reward at k from the future one $i \geq k+1$

$$\begin{aligned} \mathcal{J}_\pi(\mathbf{x}_k) &= r(\mathbf{x}_k, u_k) + \gamma \sum_{i=k+1}^{N-1} r(\mathbf{x}_i, u_i) \\ &= r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}_\pi(\mathbf{x}_{k+1}). \end{aligned} \tag{36}$$

As already seen in the continuous case, the optimal cost is obtained by minimizing the following expression

$$\mathcal{J}_\pi^*(\mathbf{x}_k) = \min_\pi \left[r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}_\pi^*(\mathbf{x}_{k+1}) \right]. \tag{37}$$

Eq. 37 is the Bellman equation and describes the behavior of $\mathcal{J}$ backward in time. On the right-hand side of the equation, the minimization is performed on the sum of the two terms: the instantaneous reward $r(\mathbf{x}_k, u_k)$, function of $\mathbf{x}_k$ and the control signal u_k, and the future cost function. Moreover, we can observe that Eq. 37 enables to break the problem in a number of smaller problems. Let's assume that we can compute the optimal cost from $k+1$ to N for all the possible states $\mathbf{x}_{k+1}$: the optimal cost is thus dependent on the state $\mathbf{x}_{k+1}$. This is at the heart of the Bellman principle, that we can state using the following quote taken from Bellman (1957):

> *"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*

This principle is often found in engineering problems of navigation or optimal path. As an example, one can consider the traveling salesman problem, the aircraft routing problem or the flight path optimization. Eq. 37 allows optimizing backward in time, from N. It is called functional equation of dynamic programming (DP). The corresponding optimal policy is obtained as the minimizer of the expression. Note that these procedures are by nature *off-line planning methods*. Indeed, they involve solutions that are backward in time and knowledge of models (as in the Riccati equations case). From the historical viewpoint, it was noted within this context that the Bellman equation is the discrete counterpart of the Hamilton-Jacobi equation that, since then, is now recalled as HJB.

2.5.1 LQR using Bellman equation backward in time

As an exercise, we derive in the following the discrete solution of the LQR problem, based on the Bellman equation. We consider the quadratic, time-discrete performance index introduced in Eq. 31

$$\mathcal{J}(\mathbf{x}_k) = \frac{1}{2} \sum_{i=k}^{N-1} \left(\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i + u_i^T \mathbf{R} u_i \right) + \frac{1}{2} \mathbf{x}_N^T \mathbf{P}_N \mathbf{x}_N,$$

with $\mathbf{Q} \geq 0$, $\mathbf{R} > 0$ and $\mathbf{P}_N \geq 0$. We want to minimize the performance index by determining u_k^*. In order to do that, we proceed by going backward from the last instant of time N. At $k = N$, the cost reduces to the terminal cost

$$\mathcal{J}_N = \frac{1}{2} \mathbf{x}_N^T \mathbf{P}_N \mathbf{x}_N, \tag{38}$$

acting here as boundary condition. Stepping backward at $k = N-1$, the cost is

$$\mathcal{J}_{N-1} = \frac{1}{2}\mathbf{x}_N^T \mathbf{P}_N \mathbf{x}_N + \frac{1}{2}\left(\mathbf{x}_{N-1}^T \mathbf{Q}\mathbf{x}_{N-1} + u_{N-1}^T \mathbf{R} u_{N-1}\right). \tag{39}$$

In the evaluation at $N-1$, we can substitute the terms at $k = N$ and the linear mapping, obtaining

$$\begin{aligned}\mathcal{J}_{N-1} &= \frac{1}{2}\left(\mathbf{A}\mathbf{x}_{N-1} + \mathbf{B}_u u_{N-1}\right)^T \mathbf{P}_N \left(\mathbf{A}\mathbf{x}_{N-1} + \mathbf{B}_u u_{N-1}\right) \\ &+ \frac{1}{2}\left(\mathbf{x}_{N-1}^T \mathbf{Q}\mathbf{x}_{N-1} + u_{N-1}^T \mathbf{R} u_{N-1}\right).\end{aligned} \tag{40}$$

The optimization is unconstrained, so we perform the gradient $\partial \mathcal{J}_{N-1}/\partial u_{N-1} = 0$, and get the expression

$$0 = \mathbf{B}_u^T \mathbf{P}_N \left(\mathbf{A}\mathbf{x}_{N-1} + \mathbf{B}_u u_{N-1}\right) + \mathbf{R} u_{N-1}, \tag{41}$$

from which we get the optimal action at $N-1$. We can collect the terms appearing in the last relation introducing the feedback control law $\mathbf{K}_{N-1}$ as

$$u_{N-1}^* = -\mathbf{K}_{N-1}\mathbf{x}_{N-1} \tag{42}$$

$$\mathbf{K}_{N-1} = \left[(\mathbf{R} + \mathbf{B}_u^T \mathbf{P}_N \mathbf{B}_u)^{-1} \mathbf{B}_u^T \mathbf{P}_N \mathbf{A}\right]. \tag{43}$$

We can know get the optimal cost at $k = N-1$,

$$\mathcal{J}_{N-1}^* = \frac{1}{2}\mathbf{x}_{N-1}^T \underbrace{\left[\left(\mathbf{A} - \mathbf{B}_u \mathbf{K}_{N-1}\right)^T \mathbf{P}_N \left(\mathbf{A} - \mathbf{B}_u \mathbf{K}_{N-1}\right) + \mathbf{K}_{N-1}^T \mathbf{R} \mathbf{K}_{N-1} + \mathbf{Q}\right]}_{\mathbf{P}_{N-1}} \mathbf{x}_{N-1} \tag{44}$$

$$= \frac{1}{2}\mathbf{x}_{N-1}^T \mathbf{P}_{N-1} \mathbf{x}_{N-1}. \tag{45}$$

The optimal action u_{N-2}^* can be found by proceeding as done before for $N-1$. So by recursion, proceeding backward as $k = N-1, \ldots, 0$, we finally find for the k-th step

$$u_k^* = -\mathbf{K}_k \mathbf{x}_k \tag{46}$$

$$\mathbf{K}_k = \left[(\mathbf{R} + \mathbf{B}_u^T \mathbf{P}_{k+1} \mathbf{B}_u)^{-1} \mathbf{B}_u^T \mathbf{P}_{k+1} \mathbf{A}\right] \tag{47}$$

$$\mathbf{P}_k = \left[\left(\mathbf{A} - \mathbf{B}_u \mathbf{K}_k\right)^T \mathbf{P}_{k+1} \left(\mathbf{A} - \mathbf{B}_u \mathbf{K}_k\right) + \mathbf{K}_k^T \mathbf{R} \mathbf{K}_k + \mathbf{Q}\right] \tag{48}$$

with optimal cost given by

$$\mathcal{J}_k^* = \frac{1}{2}\mathbf{x}_k^T \mathbf{P}_k \mathbf{x}_k. \tag{49}$$

The solution is equivalent to the Joseph stabilized version of the Riccati equation (Lewis et al., 2012). It is interesting to observe that the sequence $\mathbf{P}_k$ is time varying. As a consequence a time-varying feedback control law is found also if the system is linear time invariant. This sequence can have different behaviours, as it can converge or not to a steady-state matrix. When it converges to the limit $\mathbf{P}_k = \mathbf{P}_{k+1}$, we get a stationary policy with control law $u_k = -\mathbf{K}\mathbf{x}_k$. In this case, Eq. 48 corresponds to the algebraic Riccati equation.

2.6 Approximating the Bellman equation: iterative methods

The solution of the Bellman equation is impractical in most cases because its direct, exact computations can be performed when a model is known but not too large as the computational costs quickly become prohibitive; it is an example of curse of dimensionality. However, the properties of the Bellman equation in Eq. 37 can be used for approximating the exact solutions. In particular, we can make use of the Bellman principle that – as already seen – allows to break the optimal problem into a set of sub-optimal problems and observe that the Bellman equation is a contracting, fixed point equation.

In this section we introduce two fundamental iterations: the policy iteration and the value iteration. The difference between the two strategies is rather subtle, and relies on the evaluation that is performed first: in one case the focus is on the update of the policy, while in the second case we consider updates on the value function. We remind that the cost function is frequently referred to as the value function in the context of maximization problems. For consistency with the RL literature, regardless of the optimization problem, we preferably refer to $\mathcal{J}$ in Eq. 37 as value function in the following.

2.6.1 Policy iteration (PI)

In policy iteration, we start by choosing an arbitrary policy. We introduce two indexes: a temporal index k related to the time sampling and an iteration index j related to the policy iteration. The aim is to cast an iteration process where, at each step j, the policy is improved. For a given time k, we consider the Bellman equation in Eq. 37 at the iteration $j+1$ as

$$\mathcal{J}_{j+1}(\mathbf{x}_k) = r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}_{j+1}(\mathbf{x}_{k+1}), \tag{50}$$

based on the policy estimated at the step j. This step is called policy evaluation and consists of the solution of the Bellman equation in Eq. 50. Based on the computed value function, it is possible to estimate the associated policy by minimization (Sec. 2.5). Thus, the policy is evaluated and improved at iteration $j+1$ as

$$\pi_{j+1}(\mathbf{x}_k) = \arg\min_{\pi} \mathcal{J}_{j+1}(\mathbf{x}_k) = \arg\min_{\pi} \left[r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}_{j+1}(\mathbf{x}_{k+1}) \right]. \tag{51}$$

This optimization corresponds to the policy improvement. The resulting policy $j+1$ is used at the following step of the policy iteration.

In summary, at each time k we cast an iteration process consisting of two phases: the policy evaluation through the solution of the Bellman equation Eq. 50 and the policy improvement based on the minimization of Eq. 51. The policy computed at the *j-th* step is improved until convergence. Once the policy is converged, we move forward in time.

Example: policy iteration in LQR As an example, we consider the iterative solution of the linear LQR using the policy iteration. This algorithm is described in Hewer (1971) and Lewis et al. (2012) and allows the computation of the steady state gain $\mathbf{K}$ by forward iteration instead of solving the discrete Riccati equation. We start from the Bellman equation written for the linear case

$$\mathcal{J}_{j+1}(\mathbf{x}_k) = \frac{1}{2}\left(\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + u_k^T \mathbf{R} u_k\right) + \mathcal{J}_{j+1}(\mathbf{x}_{k+1}) \tag{52}$$

with $\gamma = 1$. The optimal cost, being quadratic, is written by introducing the variable $\mathbf{P}$

$$\mathcal{J}^*_{j+1}(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k^T \mathbf{P}_{j+1}\mathbf{x}_k, \tag{53}$$

from which the Eq. 52 is rewritten

$$\mathbf{x}_k^T \mathbf{P}_{j+1}\mathbf{x}_k = \left(\mathbf{x}_k^T \mathbf{Q}\mathbf{x}_k + u_k^T \mathbf{R} u_k\right) + \mathbf{x}_{k+1}^T \mathbf{P}_{j+1}\mathbf{x}_{k+1}. \tag{54}$$

We assume a stationary, linear policy $\mathbf{K}$ such that the control action is given by $u_k = -\mathbf{K}\mathbf{x}_k$ and recall that the discrete state equation reads $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}_u u_k$. The policy evaluation is thus obtained by solving the Lyapunov equation

$$\mathbf{P}_{j+1} = \mathbf{Q} + \mathbf{K}_j^T \mathbf{R}\mathbf{K}_j + \left(\mathbf{A} - \mathbf{B}_u\mathbf{K}_j\right)^T \mathbf{P}_{j+1}\left(\mathbf{A} - \mathbf{B}_u\mathbf{K}_j\right), \tag{55}$$

corresponding to the Bellman equation in the linear case. The policy improvement is obtained by introducing the minimization

$$\arg\min_{u_k} \left[\mathbf{x}_k^T \mathbf{Q}\mathbf{x}_k + u_k^T \mathbf{R} u_k + \left(\mathbf{A}\mathbf{x}_k + \mathbf{B}_u u_k\right)^T \mathbf{P}_{j+1}\left(\mathbf{A}\mathbf{x}_k + \mathbf{B}_u u_k\right)\right],$$

performed with respect to u_k. This minimization leads to the following relation

$$\mathbf{R}u_k + \mathbf{B}_u^T \mathbf{P}_{j+1}\mathbf{B}_u u_k + \mathbf{B}_u^T \mathbf{P}_{j+1}\mathbf{A}\mathbf{x}_k = 0,$$

reordering with respect of u_k, we get

$$\mathbf{K}_{j+1} = \left(\mathbf{R} + \mathbf{B}_u^T \mathbf{P}_{j+1}\mathbf{B}_u\right)^{-1} \mathbf{B}_u^T \mathbf{P}_{j+1}\mathbf{A}.$$

In summary, at each step j of the policy iteration, a Lyapunov equation is solved and $\mathbf{P}_j$ computed; based on $\mathbf{P}_j$, the policy is updated until convergence to the solution of the discrete Riccati equation defined at time index k. In case of stationary policies, the solution is steady; the reader can compare this procedure with the solution of the backward algorithm analyzed in Eq. 2.5.1.

2.6.2 Value iteration (VI)

A drawback of policy iteration scheme described in the previous section is the policy evaluation step consisting of the solution of the Bellman equation in Eq. 50. In the linear limit, corresponding to the LQR problem, this step corresponds to the solution of the Lyapunov equation in Eq. 55. In both cases, the solution can be challenging, for instance when the dimension of the problem is large. In alternative, we can leverage the properties of the Bellman operator, that is a contraction map: the solution iteratively converges towards a fixed point being the solution of the equation. Thus, instead of solving the Bellman equation in Eq. 37, we consider a value iteration for circumventing it. The main idea is to replace this direct solution with a recursion. By introducing the index i, for avoiding mis-interpretations, the following relation is obtained

$$\mathcal{J}^{i+1}(\mathbf{x}_k) = r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}^i(\mathbf{x}_{k+1}). \tag{56}$$

We note that the left-hand side is evaluated at step $i+1$, computed on the estimates of the previous step i. This marks a substantial difference as the Eq. 56 is not anymore a Bellman equation. The policy is computed based on $\mathcal{J}^{i+1}(\mathbf{x}_k)$ by minimizing the following relation

$$\pi^{i+1}(\mathbf{x}_k) = \arg\min_{\pi} \left[r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}^{i+1}(\mathbf{x}_{k+1}) \right]. \tag{57}$$

Instead of evaluating and then improving the value function, the value iteration algorithm updates the state value function iteratively by calculating all possible rewards in the minimization by looking ahead.

Both policy iteration and value iteration are guaranteed to converge to the optimal values as they imply the fulfilment of the Bellman equation; policy iteration converges quicker than value iteration to the optimal solution, but it requires the solution of a Bellman equation at each step of the loop. In this sense, value iteration is a slower process, but it has the advantage of approximating by recursion the Bellman solution.

Example: value iteration in LQR As mentioned in Sec. 2.6.1, in the linear limit, the solution of the Bellman equation corresponds to the solution of an associated Lyapunov equation. Analogously, value iteration is a Lyapunov recursion. We consider the value function with quadratic reward and $\gamma = 1$

$$\mathcal{J}^{i+1}(\mathbf{x}_k) = \frac{1}{2} \left(\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + u_k^T \mathbf{R} u_k \right) + \mathcal{J}^i(\mathbf{x}_k). \tag{58}$$

We rewrite the value function as a quadratic form by means of $\mathbf{P}$. Following the same steps used for the Hewer algorithm, we get to the relation

$$\mathbf{P}^{i+1} = \mathbf{Q} + (\mathbf{K}^i)^T \mathbf{R} \mathbf{K}^i + \left(\mathbf{A} - \mathbf{B}_u \mathbf{K}^i \right)^T \mathbf{P}^i \left(\mathbf{A} - \mathbf{B}_u \mathbf{K}^i \right). \tag{59}$$

Once again, it is worth stressing that on the left-hand side the index is $i+1$, while on the right-hand side we are considering quantities computed at the previous iteration step i. So, this is not a Lyapunov equation, but the corresponding recursive form based on the idea that the solution converges to a fixed point. The algorithm proceeds by computing the policy associated with the updated $\mathbf{P}$.

2.6.3 Generalized policy iteration

The value iterations allows to avoid the solution of the Bellman equation. We can borrow this idea for generalizing the policy iteration. As already mentioned, the policy iteration consists of two steps: the policy evaluation and the policy improvement. Let's reconsider the policy evaluation at step j: we replace the Bellman equation with a recursion over the index i, using the value iteration trick. As a consequence the following relation is obtained

$$\mathcal{J}_j^{i+1}(\mathbf{x}_k) = r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}_j^i(\mathbf{x}_{k+1}) \tag{60}$$

that is iterated until convergence of the value function. Once $\mathcal{J}_j^{i+1}$ associated with the policy π_j is converged, we improve the policy using Eq. 51

$$\pi_j(\mathbf{x}_k) = \arg\min_{\pi} \mathcal{J}_j^{i+1}(\mathbf{x}_k) = \arg\min_{\pi} \left[r(\mathbf{x}_k, u_k) + \gamma \mathcal{J}_j^{i+1}(\mathbf{x}_{k+1}) \right]$$

and get π_{j+1}. This process is called iterative policy iteration: it shares the same scheme of the policy iteration, but includes the recursive scheme of the value iteration in the policy evaluation step.

The iterative policy iteration is characterized by two nested iterations. In practice, it is not necessary to iterate until convergence the evaluation step. This idea leads to the concept of generalized policy iteration, where the simultaneous processes of policy evaluation and policy improvement are performed; the interested reader can refer for further details to the review work by Bertsekas (2011).

2.7 Reinforcement learning (RL)

DP allows us to find the optimal value and policy backward in time. Approximations are possible using the time-forward iterations analyzed in the previous section. However, we can take a further step and observe that the policy iteration and the value iteration can also be applied without a physical model at hand. By inspecting the Bellman equation, we observe that estimates of the value function $\mathcal{J}$ and the scores associated with the reward r can be obtained by solely registering an observable of the state y_k

$$\mathcal{J}_\pi(y_k) = \min_\pi \left[r(y_k, u_k) + \gamma \mathcal{J}_\pi(y_{k+1}) \right]. \tag{61}$$

We can do a parallel with standard tools designed on identified model of the system to be controlled: it is not necessary to estimate the full state $\mathbf{x}_k$, but it is sufficient to identify a proxy $\hat{\mathbf{x}}_k$ or directly compute the transfer functions capturing the correct input-output behavior of the system (see *i.e.* Kim and Bewley, 2007). In perfect analogy, we note that it suffices to observe y_k and measure the reward r for recovering $\mathcal{J}_\pi(y_k)$ from the interactions of the system with the environment under the policy π. This enables to circumvent the need of a large model of the system, while preserving the possibility of approximating optimal solutions. In fact, using iterative methods, it is possible to approximate the DP by learning a policy interacting with the system, forward-in-time, and without the knowledge of a model a-priori. These elements define the perimeter of reinforcement learning (RL), where the agent interacts with the environment and learns from it the optimal or suboptimal policies based on sequential decisions that improve the control actions on the experience by observing the response of the system. In principle, the identified policy is optimal if the associated $\mathcal{J}_\pi(y_k)$ is an approximate solution of the Bellman equation.

2.7.1 Markov decision processes (MDP)

For conciseness and easiness of the discussion, we have described how, from standard control, it is possible to get to the definition of the Bellman equation and how the latter can be approximated iteratively. We have applied deterministic linear or nonlinear descriptions of the systems at hand. However, the most natural framework for studying RL is statistical and is represented by Markov decision processes (MDP). MDP can also be applied for describing the numerous control strategies mentioned so far, such as optimal control, adaptive control or model-predictive control. As such, it represents an ideal framework for a unifying perspective.

We introduce a set of states $\mathcal{X}$ (or state space), a set of actions $\mathcal{U}$ (or action space). The transition probability $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ provides the conditional probability

$$\mathcal{P}^{u}_{\mathbf{x},\mathbf{x}'} = p\left\{\mathbf{x}'|\mathbf{x}, u\right\}, \tag{62}$$

transitioning from the state $\mathbf{x} \in \mathcal{X}$ to the state $\mathbf{x}' \in \mathcal{X}$ under a given action $u \in \mathcal{U}$. We also define a reward $\mathcal{R} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ that provides a score associated with the transition. The MDP is thus defined by the four elements $(\mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R})$. The Markov property (or Markovianity) refers to the temporal dependence of the transition probabilities solely related to the transitions from the state $\mathbf{x}$ to the following $\mathbf{x}'$ without being a function of previous steps. Within this context, the policy is a map $\pi : \mathcal{X} \times \mathcal{U} \to [0, 1]$ with conditional probability

$$\pi(\mathbf{x}, u) = p\left\{\mathbf{x}|u\right\} \tag{63}$$

of taking an action $u \in \mathcal{U}$ when the MDP is in $\mathbf{x} \in \mathcal{X}$. Note that the policy is stochastic and is defined by a distribution. If the probability is 1, the policy is deterministic.

Starting from these elements, it is possible to rewrite the policy iteration, its generalized form, and the value iteration, which lead to numerous versions of RL. We refer to the books by Bertsekas and Tsitsiklis (1996); Puterman (2014); Sutton and Barto (2018) for a detailed analysis.

3 Elements of reinforcement learning in practice

The overview proposed in the previous section enables to capture the traits in common between control and reinforcement learning (RL), starting from dynamic programming (DP). In RL, the learning step of the policy is performed interacting with the system, forward-in-time, and without the knowledge of a model a-priori. If we are able to identify policies that are approximate solutions of the Bellman equation, we can also guarantee these policies to be sub-optimal within a threshold or optimal. Some definitions were already introduced in Sec. 2.1 and in Sec. 2.7.1, where we have seen that the most appropriate framework for describing RL is Markov Decision Processes (MDP). However, as the reader may notice, the common practice of reinforcement is characterized by a specific jargon. We aim here at reconciling at least partially this gap by introducing some definitions and concepts to help the reader establish connections with the common practice, and categorize the different algorithms currently available in literature.

3.1 A short glossary

In the following, we provide some definitions commonly found in RL. The first distinction is related to the nature of the policies. In particular, we define as

- **Target policy**, the policy that an RL agent learns for fulfilling the value function;
- **Behavior policy**, a policy that the agent learns during the exploration, *i.e.* when the agent interacts with the environment.

The distinction between target and behavior policies allows to define on-policy and off-policy learning. In particular

- **On-policy learning** seeks or improves the policy during exploration: behavior and target policies are identical. Examples of on-policy are provided by Monte-Carlo searching or the Proximal Policy Optimization (PPO), see Schulman et al. (2017).
- **Off-policy learning** is characterized by a learning process where behavior policies differs from target policies. A landmark example is provided by $\mathcal{Q}$-learning (see Watkins and Dayan, 1992, and below).

On-policy and off-policy are naturally related to the concepts of exploration and exploitation, already mentioned in this chapter. We can now define them as follows

- **Exploration** allows the agent to improve the knowledge of the environment and the interaction between action and environment in order to maximize/optimize long-term decisions. Behavior policies are key in this process.
- **Exploitation** tends to get the best short-reward by exploiting the current estimates of the action-state space. It is associated with the notion of greedy action, as it is characterized by a short-term reward. However, this may lead to a sub-optimal behavior.

These two strategies are often seen as opposites, giving rise to the so-called exploration-exploitation dilemma. Balancing these strategies is essential in RL problems, as exploration and exploitation replace modeling when no model is available. Currently, both model-free and model-based RL algorithms can be found in the literature, with the second class gaining momentum; an example is RL based on modeling Gaussian processes informed (Chua et al., 2018). Among the classical strategies, we can distinguish three classes (Lewis et al., 2012)

- **Exact computations** If complete knowledge of the system is available, for instance by system identification or reduced-order modeling, one can run exact computations. The controller is designed upon the preliminary identification step, typically in an open loop – thus off policy. Literature in flow control insofar has been mostly based on this approach. In principle, dynamic programming (DP) can be applied, including iterative approximations.

- **Monte Carlo (MC) learning** MC learning strongly relies on the idea of episodic task and exploration: we sample different trajectories of the system emanating from prescribing the initial state and assess it. The expected values are approximated by repetitively averaging along the sampled paths, thus convergence is guarantee if we go through all the states many time. In order to do this, we assume the Markov decision process (MDP) model to be ergodic. Iterative learning control and machine learning control are closely related. This sampling methodology is fully model-free.

- **Temporal difference methods** These methods lie at the intersection of DP and MC methods: they are model-free (like MC) but can be implemented online with step-by-step computations (unlike MC). Combined with the policy iteration and value iteration, it constitutes the basis of a large body of algorithms included in RL. More details are given in the next section.

3.2 Identifying the policy

Among the possible classifications of RL algorithms, one of the most common ways is to consider the update of the policy and if the associated value function is a solution of an approximated Bellman equation or not; we thus introduce the actor and critic.

- **Actor-based algorithms** This approach characterizes the policy-based methods, where the policy is learnt without learning the expected outcomes of different policies. Optimality of the policy is not guaranteed. An example is given by the `reinforce` algorithm (Williams, 1992).

- **Critic-based algorithms** In this approach, we introduce a state-value function that we learn during the training process and aim at maximizing. The actions are chosen by evaluating each state; thus, we don't identify a policy as the actions are discrete in time. The resulting action can be associated with optimal value functions, evaluated using temporal difference (TD). An example of this technique is the $\mathcal{Q}$-learning (Watkins and Dayan, 1992).

Actor-based algorithms are often referred to as policy search (PS) algorithms, as the policy is sought via optimization techniques (gradient-based or gradient-free). Critic-based algorithms instead are closely related to value iteration and policy iteration, combined with TD.

3.2.1 Temporal difference (TD)

Temporal difference (TD) stands between Monte Carlo methods and dynamic programming (DP): in TD, the learning process is performed by bootstrapping from the current estimate of the value function, thus we sample from the environment (like in MC), but perform updates based on current estimates (like DP methods). For simplicity, we consider the Bellman equation along one trajectory, associated with a given policy π as

$$\mathcal{J}_\pi(\mathbf{x}_k) = r_k + \gamma \mathcal{J}_\pi(\mathbf{x}_{k+1})\,. \tag{64}$$

We introduce the so-called TD, as the error between the left hand side and the right hand side of Eq. 64

$$e_k = r_k + \gamma \mathcal{J}_\pi(\mathbf{x}_{k+1}) - \mathcal{J}_\pi(\mathbf{x}_k). \tag{65}$$

In principle, the Bellman equation is fulfilled when the error $e_k \to 0$. In practice, we can achieve optimality of the solution if we minimize the expression in Eq. 65. TD can be applied in policy iteration or value iteration.

3.2.2 $\mathcal{Q}$-learning and SARSA

An example of TD algorithm is represented by the $\mathcal{Q}$-learning (Watkins and Dayan, 1992). $\mathcal{Q}$-learning is an off-policy method, which – we remind – means that the optimal action-value function $\mathcal{Q}$ is estimated independently of the current policy (exploration). An optimal $\mathcal{Q}$-learning step can be expressed as

$$\mathcal{Q}_{k+1}(\mathbf{x}_k, u_k) \leftarrow \mathcal{Q}_k(\mathbf{x}_k, u_k)(1-\alpha) + \alpha \left[r_{k+1} + \gamma \min_{u_{k+1}} \mathcal{Q}_k(\mathbf{x}_{k+1}, u_{k+1}) \right], \tag{66}$$

where $0 \leq \alpha \leq 1$ is the learning rate and $0 \leq \gamma \leq 1$ is the discount factor. The action u is discrete.

It is interesting to compare this strategy with the SARSA (State-Action-Reward-State-Action) algorithm (Sutton and Barto, 2018). In contrast to $\mathcal{Q}$-learning, which is an off-policy algorithm, SARSA considers the actual action taken by the agent, incorporating the policy's exploration behavior into its updates. Thus, it is an on-policy algorithm relying on TD where the updates of the action-value function are based on the current action taken by the agent. This often leads to more conservative policies in environments with high variability.

3.2.3 Actor-critic algorithms

This approach combines policy-based (actor) and value-based (critic) methods, allowing for simultaneous learning of both the policy and the value function. In these architectures,

the policy and the value function are defined by distinct parametric structures. Different learning procedures can be considered.

In policy search algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), the agent interacts with the environment, and a score is assigned to each trajectory. These scores enable the update of the critic, which evaluates the quality of the actions taken. Consequently, the critic is trained based on the values estimated during the rollout of the policies.

A different approach is taken in algorithms like the Deep Deterministic Policy Gradient (DDPG) (Silver et al., 2014) or the soft actor-critic (SAC) (Haarnoja et al., 2018). These algorithms are more closely related to value-based strategies. In these cases, the TD error is utilized for updating the critic, as described by (Sutton and Barto, 2018)

$$\mathcal{Q}_{k+1}(\mathbf{x}_k, u_k) \leftarrow \mathcal{Q}_k(\mathbf{x}_k, u_k) + \alpha e_{TD} \tag{67}$$

$$e_{TD} = r_{k+1} + \gamma \mathcal{Q}_k(\mathbf{x}_{k+1}, u_{k+1}) - \mathcal{Q}(\mathbf{x}_k, u_k). \tag{68}$$

The target $\mathcal{Q}$-value is computed using the reward and the estimated $\mathcal{Q}$-value of the next state by minimizing e_{TD}. Importantly, the actor is updated by using the gradients of the $\mathcal{Q}$-value with respect to the parameters defining the actor's policy.

The sequence of updates highlights a significant distinction between the two approaches. In the value-based actor-critic algorithms (DDPG, SAC), the policy update is directly guided by the critic, allowing for the potential acquisition of optimal policies if the Bellman equation is accurately approximated during the TD process. In contrast, in actor-based actor-critic algorithms (PPO), the policies are selected with the help of the critic but optimality is not guaranteed through a proxy of the Bellman equation.

3.3 Numerical approximations

A final aspect that is crucial in the practice of RL is the numerical approximation of the value function and the policy. So far we have referred to the approximation of analytical solutions using the appropriate iterative schemes. In practice, however, these functions must be represented and updated during the learning process with the appropriate expansion, whether linear or nonlinear.

A first example of approximation, in presence of linear or quadratic terms in the value function, is provided by the product of vectors. The quadratic form can be manipulated as follows

$$\mathcal{J}(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k^T \mathbf{P} \mathbf{x}_k = \frac{1}{2} vec(\mathbf{P})^T (\mathbf{x}_k \otimes \mathbf{x}_k) = vec(\mathbf{P})^T \phi(\mathbf{x}_k), \tag{69}$$

where the operator $vec(\mathbf{P})$ stacks the column of the matrix $\mathbf{P}$ in a vector. The Kronecker product $\otimes$ allows to define ϕ, where the entries are the pairwise products of the vector components (Lewis et al., 2012). Due to the redundant terms, given n the dimension of the vector $\mathbf{x}_k$, the resulting new vectors are of dimension $n(n+1)$. Assuming the Bellman equation has a smooth solution, locally, higher order approximation can also be obtained as

$$\mathcal{J}(\mathbf{x}_k) \approx \sum_{i=1}^{L} w_i \phi_i(\mathbf{x}_k) + \varepsilon_L(\mathbf{x}_k) = \mathbf{W}^T \Phi(\mathbf{x}_k) + \varepsilon_L(\mathbf{x}_k), \tag{70}$$

where ε_L is the approximation error. When introducing the basis vector Φ, the problem is translated into the identification of the coefficients of expansion $\mathbf{W}$. The basis can be represented by polynomials or Volterra series; recent applications of RL using sparse identification of nonlinear dynamics (SINDy) rely on this idea, see Arora et al. (2022); Zolman et al. (2023).

As mentioned in the introduction, artificial neural networks (ANN) are the current working-horse for the function representation. The use of ANN was already suggested in early form the 90is by Bertsekas and Tsitsiklis (1996) (neuro-dynamic programming) because of the remarkable versatility in representing strongly nonlinear functions. In the case of ANN, L is the number of hidden-layer neurons, the basis is composed by the activation functions (sigmoid, hyperbolic tangent, Gaussian radial basis functions...), and the weights are the ones associated with the ANN. The drawback of ANN, however, is the tendency to data hungriness as well as the risk of introducing more parameters than are actually needed for the problem. In other words, the expressivity of the ANN can prove detrimental in terms of data required during the training and lack of generalization (over-fitting problem). A first remedy is to limit the number of layers and neurons to the necessary, possibly including physics constraints (linear policies, quadratic cost functions). In alternative, one can choose parsimonious representations such as tensor-based composition (Gorodetsky, 2017).

4 Essential bibliography and final remarks

Machine learning (ML) applications have gained tremendous momentum in the last decade; thus, it is rather natural to expect a quick evolution of standard approaches of analysis and optimization of fluid flows boosted by statistical learning, and more specifically RL. However, some caveats are in order. As mentioned in these pages, reinforcement learning shares elements from numerous related fields, ranging from applied mathematics and statistics to control, engineering, and, more recently deep learning. Because of the vastness of the topic and the different angles one can use for tackling the problem, the risk is to approach RL in naïve way: we believe that a simple combination of off-the-shelf algorithms and environments to be controlled while opening new venues of research in terms of possible applications, can result detrimental on the longer run in the full deployment of RL's potential. Also, we like to emphasize the importance of a preliminary analysis of the open-loop system, since the physics to be controlled may limit the performance and robustness of the closed loop and thus require specific choices of sensing and control authorities, as well as the need for improved modeling and algorithms.

As mentioned in this chapter, solutions in RL were found in parallel with control solutions, departing from the same roots, namely the Hamilton-Jacobi-Bellman equation and dynamic programming (DP). Not surprisingly, model predictive control can be analyzed as part of reinforcement learning (RL), while a note written by Sutton et al. (1992) highlights the close relation between adaptive filters and actor-based RL controllers; remarkably, introducing a critic into adaptive control one may guarantee optimality. This is currently a rather active field of research that opened up at the crossing between control and RL, which, in fact, now can be reconciled.

This chapter is far from being exhaustive, as it provides only a quick overview of some of the main concepts behind RL. Nonetheless, we hope it can trigger the curiosity to develop solutions better suited for fluid mechanics problems. The interested reader can explore the subject further starting from the works of F. L. Lewis and collaborators in optimal control (Lewis et al., 2012) and the numerous references by D. Bertsekas, among which it is worth citing Bertsekas (1995) and Bertsekas (2019). For a viewpoint from statistics, being the framework inherently based on Markov decision processes, a landmark reference is the book by Puterman (2014). Together with the classic by Sutton and Barto (2018), these works are excellent starting points. Numerous resources are available online, including recordings of classes. We can mention the material provided by S. Levine. From a fluid mechanics perspective – besides research papers – the interested reader can start from the review by Brunton et al. (2020) and the book by Brunton and Kutz (2022) for a larger perspective on the need for data-driven tools in the current engineering practice. A review in RL for fluids can be found in Vignon et al. (2023), while a first comparative analysis with different tools is carried out by Pino et al. (2023). The packages in `OpenFoam` by Wang et al. (2022) and the one in `FEniCS` by Paehler et al. (2023) provide convenient platforms for testing RL in fluids, complementing the numerous implementation available online such as Gymnasium from OpenAI or Tensorforce.

We believe this is only the beginning of an exciting venue of research with several open questions. Among these issues, we can note that optimality of the solution is not always guaranteed in practice, and a detailed analysis of the robustness of these policies has been so far elusive. However, a successful protocol based on RL should require the

assessment of the performance with respect to the uncertainties of the system, such as the evolution of the environment parameters, or the impact that plant limitation has such as the presence of input-output time-delays. In this sense, the certification problem must be considered. Among the numerous venues of research for future works, safe RL is particularly appealing to ensure reasonable system performance with respect to the constraints during the exploration/exploitation through external knowledge, physics or the guidance of a risk metric (Garcıa and Fernández, 2015).

Acknowledgments The author would like to thank Lionel Mathelin (CNRS-LISN) for sharing ideas on numerous data-driven subjects and providing comments on the chapter; Miguel A. Mendez (VKI-ULB) and Alessandro Parente (ULB) for suggestions on the final version of this manuscript; Rémy Hosseinkhan-Boucher, Amine Saibi, and past collaborators Michele Alessandro Bucci (Safran-Tech) and Nicoló Fabbiane (Onera) for their stimulating viewpoints. Part of this research was funded under grants ANR-DGA FlowCon (project-ANR-17-ASTR-0022) and ANR-JCJC REASON (ANR-21-CE46-0008).

References

Abergel, F. and Temam, R. (1990). On some control problems in fluid mechanics. *Theoretical and Computational Fluid Dynamics*, 1(6):303–325.

Arbabi, H., Korda, M., and Mezić, I. (2018). A data-driven koopman model predictive control framework for nonlinear partial differential equations. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6409–6414. IEEE.

Arora, R., da Silva, B. C., and Moss, E. (2022). Model-based reinforcement learning with SINDy. *arXiv preprint arXiv:2208.14501*.

Åström, K. J. and Wittenmark, B. (2008). *Adaptive control*. Courier Corporation.

Bae, H. J. and Koumoutsakos, P. (2022). Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nature Communications*, 13(1):1443.

Bagheri, S., Henningson, D. S., Hoepffner, J., and Schmid, P. J. (2009). Input-output analysis and control design applied to a linear model of spatially developing flows. *App. Mech. Rev*, 62(2).

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684.

Bellman, R. (1958). Dynamic programming and stochastic control processes. *Inf. Control.*, 1(3):228–239.

Bertsekas, D. (2011). Approximate policy iteration: a survey and some new methods. *J. Control Theory Appl.*

Bertsekas, D. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*. Athena scientific Belmont, MA.

Bertsekas, D. P. (2019). *Reinforcement learning and optimal control*. Athena Scientific.

Bewley, T., Luchini, P., and Pralits, J. (2016). Methods for solution of large optimal control problems that bypass open-loop model reduction. *Meccanica*, 51(12):2997–3014.

Bewley, T., Temam, R., and Ziane, M. (2000). A general framework for robust control in fluid mechanics. *Physica D: Nonlinear Phenomena*, 138:360–392.

Bewley, T. R., Moin, P., and Temam, R. (2001). Dns-based predictive control of turbulence: an optimal benchmark for feedback algorithms. *Journal of Fluid Mechanics*, 447:179–225.

Biferale, L., Bonaccorso, F., Buzzicotti, M., Clark Di Leoni, P., and Gustavsson, K. (2019). Zermelo's problem: Optimal point-to-point navigation in 2D turbulent flows using reinforcement learning. *Chaos*, 29(10):103138.

Borra, F., Biferale, L., Cencini, M., and Celani, A. (2022). Reinforcement learning for pursuit and evasion of microswimmers at low reynolds number. *Physical Review Fluids*, 7(2):023103.

Brunton, S. L. and Kutz, J. N. (2022). *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press.

Brunton, S. L. and Noack, B. R. (2015). Closed-loop turbulence control: Progress and challenges. *Appl. Mech. Rev.*, 67(5):050801.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.*, 52:477–508.

Bucci, M. A., Semeraro, O., Allauzen, A., Wisniewski, G., Cordier, L., and Mathelin, L. (2019). Control of chaotic systems by deep reinforcement learning. *Proc. R. Soc. A*, 475(2231):20190351.

Cattafesta III, L. N. and Sheplak, M. (2011). Actuators for active flow control. *Annu. Rev. Fluid Mech.*, 43:247–272.

Chevalier, M., Hœpffner, J., Åkervik, E., and Henningson, D. (2007). Linear feedback control and estimation applied to instabilities in spatially developing boundary layers. *J. Fluid Mech.*, 588:163–187.

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765.

Cvitanović, P., Davidchack, R. L., and Siminos, E. (2010). On the state space geometry of the Kuramoto–Sivashinsky flow in a periodic domain. *SIAM Journal on Applied Dynamical Systems*, 9(1):1–33.

Duriez, T., Brunton, S. L., and Noack, B. R. (2016). *Machine Learning Control: Taming Nonlinear Dynamics and Turbulence*. Springer.

Dussauge, T. P., Sung, W. J., Pinon Fischer, O. J., and Mavris, D. N. (2023). A reinforcement learning approach to airfoil shape optimization. *Scientific Reports*, 13(1):9753.

Fabbiane, N., Semeraro, O., Bagheri, S., and Henningson, D. S. (2014). Adaptive and model-based control theory applied to convectively unstable flows. *App. Mech. Rev*, 66(6):060801.

Fabbiane, N., Simon, B., Fischer, F., Grundmann, S., Bagheri, S., and Henningson, D. S. (2015). On the role of adaptivity for robust laminar flow control. *J. Fluid Mech.*, 767.

Fan, D., Yang, L., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. (2020). Reinforcement learning for bluff body active flow control in experiments and simulations. *PNAS*, 117(42):26091–26098.

Foucart, C., Charous, A., and Lermusiaux, P. F. (2023). Deep reinforcement learning for adaptive mesh refinement. *Journal of Computational Physics*, 491:112381.

Gad-el Hak, M. (2000). *Flow Control: Passive, Active, and Reactive Flow Management*. Cambridge University Press.

Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16(1):1437–1480.

Glad, T. and Ljung, L. (2000). *Control Theory*. Taylor & Francis, London.

Goldin, N., King, R., Pätzold, A., Nitsche, W., Haller, D., and Woias, P. (2013). Laminar flow control with distributed surface actuation: damping tollmien-schlichting waves with active surface displacement. *Experiments in fluids*, 54:1–11.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*. MIT press Cambridge.

Gorodetsky, A. A. (2017). *Continuous low-rank tensor decompositions, with applications to stochastic optimal control and data assimilation*. PhD thesis, Massachusetts Institute of Technology.

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

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. (2018). Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hervé, A., Sipp, D., Schmid, P. J., and Samuelides, M. (2012). A physics-based approach to flow control using system identification . *J. Fluid Mech.*, 702:26–58.

Hewer, G. (1971). An iterative technique for the computation of the steady state gains for the discrete optimal regulator. *IEEE Transactions on Automatic Control*, 16(4):382–384.

Högberg, M., Bewley, T. R., and Henningson, D. S. (2003). Linear feedback control and estimation of transition in plane channel flow. *Journal of Fluid Mechanics*, 481:149–175.

Högberg, M. and Henningson, D. S. (2002). Linear optimal control applied to instabilities in spatially developing boundary layers. *Journal of Fluid Mechanics*, 470:151–179.

Juillet, F., McKeon, B., and Schmid, P. J. (2014). Experimental control of natural perturbations in channel flow. *J. Fluid Mech.*, 752:296–309.

Keramati, H., Hamdullahpur, F., and Barzegari, M. (2022). Deep reinforcement learning for heat exchanger shape optimization. *International Journal of Heat and Mass Transfer*, 194:123112.

Kim, J. and Bewley, T. R. (2007). A linear systems approach to flow control. *Annu. Rev. Fluid Mech.*, 39:383–417.

Kühnen, J., Song, B., Scarselli, D., Budanur, N. B., Riedl, M., Willis, A. P., Avila, M., and Hof, N. (2018). Destabilizing turbulence in pipe flow. *Nat. Phys.*, 14(4):386–390.

Leclercq, C., Demourant, F., Poussot-Vassal, C., and Sipp, D. (2019). Linear iterative method for closed-loop control of quasiperiodic flows. *Journal of Fluid Mechanics*, 868:26–65.

Lewis, F. L., Vrabie, D., and Syrmos, V. L. (2012). *Optimal control*. John Wiley & Sons.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Ljung, L. (1999). *System identification*. Wiley Online Library.

Losse, N. R., King, R., Zengl, M., Rist, U., and Noack, B. R. (2011). Control of tollmien–schlichting instabilities by finite distributed wall actuation. *Theoretical and Computational Fluid Dynamics*, 25:167–178.

Luchini, P. and Bottaro, A. (2014). Adjoint equations in stability analysis. *Ann. Rev. Fluid*, 46:493–517.

Ma, Z., Ahuja, S., and Rowley, C. W. (2011). Reduced-order models for control of fluids using the eigensystem realization algorithm. *Theor. Comp. Fluid Dyn.*, 25(1-4):233–247.

Marra, L., Meilán-Vila, A., and Discetti, S. (2024). Self-tuning model predictive control for wake flows. *arXiv preprint arXiv:2401.10826*.

Noack, B. R., Morzynski, M., and Tadmor, G. (2011). *Reduced-order modelling for flow control*, volume 528. Springer Science & Business Media.

Novati, G., de Laroussilhe, H. L., and Koumoutsakos, P. (2021). Automating turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, 3(1):87–96.

Novati, G., Mahadevan, L., and Koumoutsakos, P. (2019). Controlled gliding and perching through deep-reinforcement-learning. *Physical Review Fluids*, 4(9):093902.

Paehler, L., Callaham, J., Ahnert, S., Adams, N., and Brunton, S. (2023). Hydrogym: A reinforcement learning control framework for fluid dynamics. *Bulletin of the American Physical Society*.

Pino, F., Schena, L., Rabault, J., and Mendez, M. A. (2023). Comparative analysis of machine learning methods for active flow control. *Journal of Fluid Mechanics*, 958:A39.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rabault, J., Kuchta, M., Jensen, A., Réglade, U., and Cerardi, N. (2019). Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *J. Fluid Mech.*, 865:281–302.

Rabault, J., Ren, F., Zhang, W., Tang, H., and Xu, H. (2020). Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. *J Hydrodynam B .*, 32:234–246.

Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annu. Rev. Control Robot. Auton. Syst.*, 2:253–279.

Reddy, G., Wong-Ng, J., Celani, A., Sejnowski, T. J., and Vergassola, M. (2018). Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239.

Rowley, C. W. (2005). Model reduction for fluids, using balanced proper orthogonal decomposition. *Int. J. Bifurcation Chaos*, 15(03):997–1013.

Rowley, C. W. and Dawson, S. T. (2017). Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Semeraro, O., Bagheri, S., Brandt, L., and Henningson, D. S. (2013a). Transition delay in a boundary layer flow using active control. *J. Fluid Mech.*, 731(9):288–311.

Semeraro, O., Pralits, J. O., Rowley, C., and Henningson, D. S. (2013b). Riccati-less approach for optimal control and estimation: an application to two-dimensional boundary layers. *J. Fluid Mech.*, 731:394–417.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning*, pages –.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

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

Sturzebecher, D. and Nitsche, W. (2003). Active cancellation of Tollmien–Schlichting instabilities on a wing using multi-channel sensor actuator systems. *Intl J. Heat and Fluid Flow*, 24:572–583.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., Barto, A. G., and Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2):19–22.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219.

Van Overschee, P. and De Moor, B. (2012). *Subspace Identification for Linear Systems: Theory – Implementation – Applications*. Springer Science & Business Media.

Vergassola, M., Villermaux, E., and Shraiman, B. I. (2007). Infotaxis as a strategy for searching without gradients. *Nature*, 445(7126):406–409.

Verma, S., Novati, G., and Koumoutsakos, P. (2018). Efficient collective swimming by harnessing vortices through deep reinforcement learning. *PNAS*, 115(23):5849–5854.

Vignon, C., Rabault, J., and Vinuesa, R. (2023). Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions. *Physics of Fluids*, 35(3).

Viquerat, J., Rabault, J., Kuhnle, A., Ghraieb, H., Larcher, A., and Hachem, E. (2021). Direct shape optimization through deep reinforcement learning. *Journal of Computational Physics*, 428:110080.

Wang, Q., Yan, L., Hu, G., Li, C., Xiao, Y., Xiong, H., Rabault, J., and Noack, B. R. (2022). Drlinfluids: An open-source python platform of coupling deep reinforcement learning and openfoam. *Physics of Fluids*, 34(8).

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Xiao, D. and Papadakis, G. (2019). Nonlinear optimal control of transition due to a pair of vortical perturbations using a receding horizon approach. *Journal of Fluid Mechanics*, 861:524–555.

Yang, J., Dzanic, T., Petersen, B., Kudo, J., Mittal, K., Tomov, V., Camier, J.-S., Zhao, T., Zha, H., Kolev, T., et al. (2023). Reinforcement learning for adaptive mesh refinement. In *International Conference on Artificial Intelligence and Statistics*, pages 5997–6014. PMLR.

Zhou, K., Doyle, J. C., and Glover, K. (2002). *Robust and Optimal Control*. Prentice Hall, New Jersey.

Zolman, N., Fasel, U., Kutz, N., and Brunton, S. (2023). Sindy-rl: Interpretable and efficient reinforcement learning for fluid flow control. *Bulletin of the American Physical Society*.