Deep reinforcement learning for active flow control: where do we stand, and what are the perspectives for future use in physics and fluid mechanics?

Preprint · November 2023

DOI: 10.13140/RG.2.2.26313.93289

Deep reinforcement learning for active flow control: where do we stand, and what are the perspectives for future use in physics and fluid mechanics?

November 9, 2023

Jean Rabault, Independent Researcher, Oslo, Norway

Pol Suarez, FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm Sweden

Francisco Alcántara-Ávila, FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm Sweden

Luca Guastoni, FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm Sweden

Joel Vasanth, FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm Sweden

Ricardo Vinuesa, FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm Sweden

Deep reinforcement learning (DRL) is attracting increasing attention for performing a number of optimization tasks in fluid mechanics [1, 2, 3]. While active flow control (AFC) is the application we will focus on in this chapter, DRL is also being applied for other tasks within fluid mechanics [1], including, e.g., optimal route planning in turbulent environments [4], the understanding and optimization of collective behavior of swimmers [5], and shape optimization [6]. In the following, we will focus on providing an up-to-date overview of the main open research directions within DRL applications for AFC. While we will discuss the main ingredients of DRL algorithms, we will not provide a detailed introduction to DRL per se, as this is already a topic well discussed in the literature [1, 7, 8], and a detailed discussion of the implementation and technicality of DRL algorithms goes beyond the scope of the present chapter. Therefore, we will assume in the following that the reader is already familiar with the key concepts and algorithms used in DRL, and we will focus on providing insights that are both specific to DRL for AFC, and usually not presented in introductions to the topic. We will then discuss the implication of these insights in relation to the main ongoing research directions around DRL for AFC, and provide a number of practical recommendations for DRL practitioners in this context. Finally, we will give an overview of the perspectives offered by DRL for AFC, and we will discuss what challenges and opportunities lie ahead.

1 Deep reinforcement learning: a promising smart and general purpose optimizer for fluid-mechanics applications

The key ingredient of DRL lies in performing optimization based on trial-and-error interaction with the system to optimize. No knowledge of the underlying equations and mathematical properties of the system to control are needed: all what is requested is that the system to control (which we will call the "plant", though it is also referred to as the "environment" or the "system", depending on the background and scientific field) should be available for online interaction through 3 channels of communication. More specifically, the DRL agent should be able at each DRL action update time step t to i) query the state s_t of the plant, ii) provide an action a_t to apply to the plant, that will be used by the plant to advance in time, iii) collect the reward r_t after the plant evolved in time. The reward r_t should reflect the quality of the system state towards the goal of the DRL agent. The objective of all DRL algorithms is to optimize the actualized reward $R = \sum_{t=0}^{N} r_t \gamma^t$, where N is the number of interactions allowed between the DRL agent and the plant, t is the DRL interaction timestep considered, and γ is the discounting factor, typically $\gamma \in [0.95; 0.99]$.

Given this framework, a number of DRL algorithms have been developed to perform effective optimization of R through trial-and-error. As the theory around DRL has been discussed many times before, we will not present DRL algorithms in details, and we refer the reader curious of more information about the internals of DRL algorithms to a number of openly available resources, e.g. [7]. However, we will summarize a few of the key elements of DRL algorithms, that we will discuss further in the following. DRL typically optimizes based on trajectories in the phase space, i.e. series of state, action, reward tuples (s_t, a_t, r_t). Depending on the exact algorithm used, this tuple may be extended by some values at the timestamp $t+1$, such as s_{t+1}, but we will ignore these technical details in the following. Each episode, i.e. each series of triplets $\{(s_t, a_t, r_t) | t = [0...N]\}$ gathered from a single series of consecutive interactions between the DRL Agent and the plant, can be seen as a trajectory of the plant in its phase space. The optimization process typically takes place by optimizing a policy $\pi(a_t|s_t)$, that describes the probability of taking the action $a_t \in \mathscr{A}$ ($\mathscr{A}$ being the state of admissible actions), given the state observation s_t. This means that the policy function defines a probabilistic mapping from the current state s_t to the action to perform a_t. In order to do so, DRL algorithms leverage function approximators to represent the policy. While other parametrizations are possible, neural networks (NNs) are currently the preferred tools to obtain a flexible function representation, thanks to their possibility to act as universal approximators [9].

Given the requirement to solve the optimization problem that aims to find the best possible policy, there exists a number of different DRL algorithms that have been developed and improved over the years to provide fast and effective policy optimization. These mostly belong to two families: Q-learning methods [10], that take advantage of the Bellman equation, versus the policy gradient methods [11], that directly estimate the gradient of the reward depending on the policy parametrization. Recent DRL algorithms combine some elements from both categories of methods, resulting in the actor-critic category of methods [12]. While a number of key ideas underlying DRL are relatively old, their effectiveness has been demonstrated only recently in a series of groundbreaking works that have taken advantage of the increase of computation power that happened in the last few decades. This has allowed to solve increasingly complex problem, including previously outstanding challenges such as exceeding human performance at the game of Go [13].

When looking more specifically at DRL applications to fluid mechanics and AFC, numerous studies have dealt with using DRL to obtain drag reduction in bluff-body and channel flow configurations, aiming at solving longstanding problems in close-loop aerodynamics control that have proven challenging for traditional methods [14]. For bluff body flows, DRL-based drag reduction has been applied to flows past a circular cylinder in the laminar [15, 16, 17, 18, 19, 20, 21], weakly turbulent [22], and fully turbulent [23] regimes. Control of the flow around wings is also attracting interest, with clear applicability for industrial uses [24]. Problems related to fluid-structure interaction and vortex-induced vibrations have also been considered [25]. In works [26, 27], DRL-based control was applied to a realistic (fully-turbulent and 3-dimensional) channel flow configuration, also comparing the DRL-control strategy to opposition control. In particular, [26] found that DRL control largely outperforms classical opposition control, illustrating that results obtained from classical methods can be improved by leveraging DRL. The works by [28, 29] introduced DRL for the control of buoyancy-driven thermal convection and the Rayleigh-Bénard instability.

Pino et al. [30] demonstrated RL control for simple dynamical systems such as the Burgers' equation that models nonlinear traveling waves encountered in AFC, as well as a model of the frequency cross-talk problem as encountered in many turbulent flow control cases. [31] studied the use of DRL for the control of chaotic systems inspired by AFC, applying DRL to the control of the 1D Kuramoto-Sivashinsky equation. The work by [32] applied RL to the control of instability in a film of falling liquid.

A number of other applications of DRL for fluid mechanics have also been studied. Applications by [33, 6] deal with DRL-based aerodynamic shape optimisation, and [34] deals with shape optimisation of a heat-exchanger. In [34], the agent is able to maximise heat transfer while simultaneously reducing the pressure drop across the heat-exchanger, both of which are desirable outcomes for this application. DRL has also been applied to adaptive mesh refinement [35]. The advantage shown there is that, rather than the user having to hand-design a heuristic algorithm for when the mesh should be refined, the DRL agent learns to improve the mesh quality for different configurations through trial and error. DRL can also be used to improve and speed up the CFD simulations themselves, by providing for example slope limiting in finite volume schemes [36], or optimized timestepping [37]. The work by [38] illustrates the use of DRL to improve our understanding of navigation policies and schooling in fish steams. The policy from the trained agent reveals that fish can show improved propulsive efficiency if they swim in appropriate locations in the wake of other fish ahead of them. Turbulence modeling obtained leveraging DRL has also been presented, both for obtaining turbulence closure models [39, 40, 41, 42], and for automatically learning wall laws [43]. Finally, optimal navigation in turbulent conditions, corresponding to the Zermelo problem, has also been investigated using DRL [4], showing that DRL allows effective optimal navigation planning.

A number of review papers [44, 1, 2, 21, 30, 45, 46] have also been written in addition to

Application	**References**
Review papers	[44, 1, 2, 21, 30, 45, 46]
Bluff body - cylinder	[15, 16, 17, 18, 19, 20, 21, 22, 23]
Bluff body - aircraft wings	[24]
Fluid-structure interactions	[25]
Channel flow	[26]
Shape optimisation	[33, 6, 34]
Numerical Schemes	[35, 36, 37]
Navigation	[38, 4]
Model Dynamical systems	[30, 31]
Flow instability	[32, 28, 29]
Turbulence closure modelling	[39, 40, 41, 42]

Table 1: Summary of the literature of DRL for AFC as discussed in Sec. 1.

these case studies and specific applications. These combined efforts mean that the volume of literature being published on the topic is growing fast, as summarized in Table 1 that list most of the recent papers presented above.

At present, state-of-the art DRL for AFC can be illustrated by, for example, the setup presented in [47], see Fig 1. As visible in Fig. 1, a DRL Agent is deployed to control a distributed input distributed output (DIDO) system, by actuating slots on the sides of a 3D cylinder. By leveraging information from probes in the neighborhood of each segment of the actuation slot, and taking advantage of the locality of the flow and reward (see the discussions about Multi Agent RL lower down), effective control can be obtained.

There is, therefore, a broad range of applications of DRL for fluid mechanics. This can be explained by both i) the generality and effectiveness of DRL algorithms, which makes them powerful general purpose optimizers, and ii) the availability of high quality frameworks that allow to deploy state-of-the-art DRL algorithms without the need to be an expert on their implementation details. In particular, frameworks such as, e.g., Tensorforce [48], TF-Agents [49], StableBaselines3 [50], or Tianshou [51], are making it possible to deploy modern DRL algorithms in a flexible way: all what is needed of the user is to encapsulate their CFD or experiment in a simple interface, that can be directly connected to the DRL

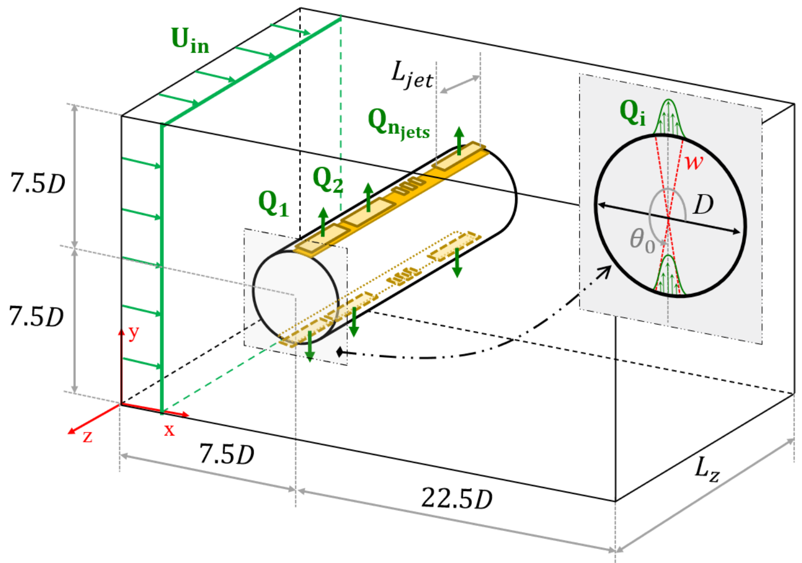

Figure 1: Illustration of 3D cylinder equipped with distributed input distributed output control. A series of actuators, sharing their policy following the Multi-Agent RL paradigm (see next section), are able to discover a control strategy that doubles the drag reduction obtained compared with traditional frequency scanning [47]. Reproduced with permission of the authors, and in agreement with the license terms, from [47], available at https://arxiv.org/pdf/2309.02462.pdf.

algorithms provided by these frameworks.

Moreover, turn key frameworks taking care of the full coupling between DRL and CFD codes are also starting to emerge, such as DRLinFluids [52], that couples several DRL libraries to the OpenFOAM solver, DRLFluent [53], that similarly couples DRL libraries to the Ansys-Fluent solver, or Relexi [54], a framework that provides general database-driven mechanisms to couple DRL algorithms and flow simulations. The only domain where extensive, turn-key frameworks are not available yet is within the deployment of DRL algorithms for experiments and real-world interaction: there is not, at the moment, an open framework solution for coupling DRL agents to actual real-world actions needed in experimental fluid mechanics flow control, despite DRL starting to emerge also for control of experiments. This can likely be explained by DRL applications having so far focused mostly on numerical works, since these allow for fast prototyping and iteration, which is key to develop new methodologies. Another explanation may be the lack of standardization for both code APIs, hardware controllers, and sensors used in experiments, which is still a hinder for the development of open source standards around tooling used in experiments. However, we expect, as the applicability of DRL for flow control is now better understood, that we will see DRL frameworks oriented towards deployment in experiments emerge in the near future, and that the successes of open source and open standards in the simulation and CFD domain will inspire similar developments in the experimental and hardware domain [1].

As DRL for fluid mechanics is undergoing rapid development we will, in the following, focus on a high level overview of the main trends and ongoing promising developments taking place in DRL for AFC application. In particular, we will summarize the main key findings that have been found to "make or break" successful application of DRL for AFC, and we will extrapolate on these trends to suggest future development directions of particular interest.

2 DRL for flow control: critical aspects for successful deployment, and promising directions for future developments

2.1 The race to higher flow and control complexity: trends and key enablers

The race to applying DRL to control flows of increasing complexity has been going on since the early DRL for flow control applications of [15, 16]. Since then, as highlighted in the

[1]Though this is a digression from the main discussion, there is ground for hope in the near future regarding the use of open source experiment setups in fluid mechanics. In particular, the use of microcontrollers and Field Programmable Gate Arrays (FPGAs) is now being democratized through open source projects such as the Arduino microcontroller ecosystem [55] and similar developements for FPGAs [56]. This allows, through both the support of open source communities and access to low cost hardware, to drastically cut costs, development time, and expertise requirements needed to build advanced mechatronics systems, and to establish fully open standards (for example, [57] was able to cut the cost of the experimental setup used therein by 2 orders of magnitude compared with commercial systems, and open source instrumentation based on Arduino is providing order-of-magnitude cost savings in experimental oceanography [58, 59]). As a consequence, open source electronics for fluid mechanics experiments bears great promises in combination to open source software tools such as, e.g., OpenPIV [60].

literature review above, DRL applications to flow control have been focusing on controlling increasingly higher Reynolds number cases (the Reynolds number, $Re = UL/\nu$, where U is the typical velocity scale of the flow, L the typical size scale, and ν the kinematic viscosity, represents the relative importance of inertial and viscous effects), going from 2D proof-of-concepts to more realistic 3D configurations, and considering an increasing number of distributed state measurements and control outputs. Depending on the application, parameters other than Re can also result in an increase in complexity. For example, in Rayleigh-Bénard convection, increase in temperature difference across the fluid leads to a larger Rayleigh number (which determines the instability regime of the system), and to more complex buoyancy-driven flow structures.

Such increases in complexity have, naturally, presented challenges for the deployment of DRL. In particular: c.i) Performing CFD simulations becomes much more expensive as the Reynolds number is increased (typically, the cost of a Direct Numerical Simulation scales approximately as Re^3 when including the need to refine meshes and reduce timesteps [61]), which makes it much more expensive to gather the same amount of episodes as Re is increased. c.ii) The dynamics of the flow get more complex with increasing non-linearities as Re increases. This makes the learning more challenging to perform and typically requires going through more episodes to learn an effective policy, as the agent needs to map a more complex phase space. c.iii) Solving complex 3D problems at Re where non-laminar behavior is observed typically requires the use of a DIDO method, which leads to a curse of dimensionality on the control space. This curse of dimensionality implies that the number of trajectories to explore by the DRL Agent grows to the power of n [32], where n is the number of components of the action vector a_t. This typically leads to a much larger (and untractable) requirement in terms of number of sampled episodes to reach an effective policy.

The challenges c.i) and c.ii) have so far been addressed effectively by using an increasingly large amount of computational power on the CFD part of the DRL for flow control training workflow. This can naturally be approached by 2 methods: s.i) one can, as it is useful to speed up CFD computations, use parallelized CFD solvers leveraging many cores. This is very visible already: while the early results of [15] used a single core CFD simulation, more recent works at higher Re have been leveraging CFD simulations running on many cores [47]. Moreover, it has been possible to s.ii) take advantage of parallelization of the gathering of episodes by running a number of CFD simulations in parallel. Indeed, DRL algorithms request a group (or batch) of episodes to be gathered between updates of the NN weights. This process can take place in a fully parallel way since different episodes do not need to communicate with each other. There too, this method is very visible through recent works: while [15] used a single CFD simulation at a time, [16] has been performing training on up to 60 simulations running independently in parallel of each others, and using such parallelism has become standard in following studies. Since s.i) and s.ii) are orthogonal of each others, and have vastly different performance requirements (in s.i), cores used within a single simulation need to be tightly coupled through effective interconnect; by contrast, in s.ii), loose coupling between the CFD simulations and the DRL framework is enough), they can be effectively combined to get multiplicative positive effect on each other.

However, the bruteforce increase of computational power provided by s.i) and s.ii) is not enough to solve more fundamental challenges arising from the curse of dimensionality corresponding to the challenge c.iii). Fortunately, most real-world systems present symmetries and invariants that allow the division of the domain into invariant subdomains. Therefore, it is possible to re-use the same control laws across each of the actuators of the different invariant subdomains. This, in turn, allows s.iii) to design effective DRL strategies by implementing these invariants into the structure of the DRL problem, leveraging what is referred to as "Multi-Agent Reinforcement Learning" (MARL). More specifically, in MARL, the different components of the action signal to be provided as part of the action $a_t = (a_t^1, ..., a_t^n)$, are seen together with the portion of the state surrounding them as individual DRL environments within the plant. The use of MARL in contexts where the control laws present invariants across the spatial extent of the plant, or symmetries over its n components, allows to learn control strategies in a constant time duration, independently of the value of n, by using a setup similar to what is presented in Fig. 2. The success of this MARL approach is demonstrated in [32, 29] and it is applicable to many cases either one considers bluff body control [47], turbulent channel flow control [26], of Rayleigh-Bénard convection [29]. As a consequence, MARL allows, in most realistic cases, to avoid the curse of dimensionality on the action space. Thus, by leveraging s.i), s.ii), and s.iii), a number of studies have been able to apply DRL for active flow control of complex situations, where a direct naive application of DRL would fail as demonstrated by, e.g., [32, 29, 62].

In addition to these key DRL-architectural ingredients, a number of well known practical aspects of DRL have been confirmed time and time again to be necessary conditions for DRL training to work. In particular, the following points are recurring critical requirements in order to allow effective DRL flow control to work in practice:

- Using reliable DRL frameworks: advanced DRL algorithms are typically hard to implement correctly and to debug, since implementation mistakes may lead to degraded learning or no learning at all, but not to a hard compiler or numerical bug. This makes it challenging to find bugs and the explanation for absence of learning, in case there is an implementation mistake on the DRL algorithm side. Therefore, we have found that relying on community-driven and well tested, high quality DRL algorithm implementations, is a critical aspect for success.

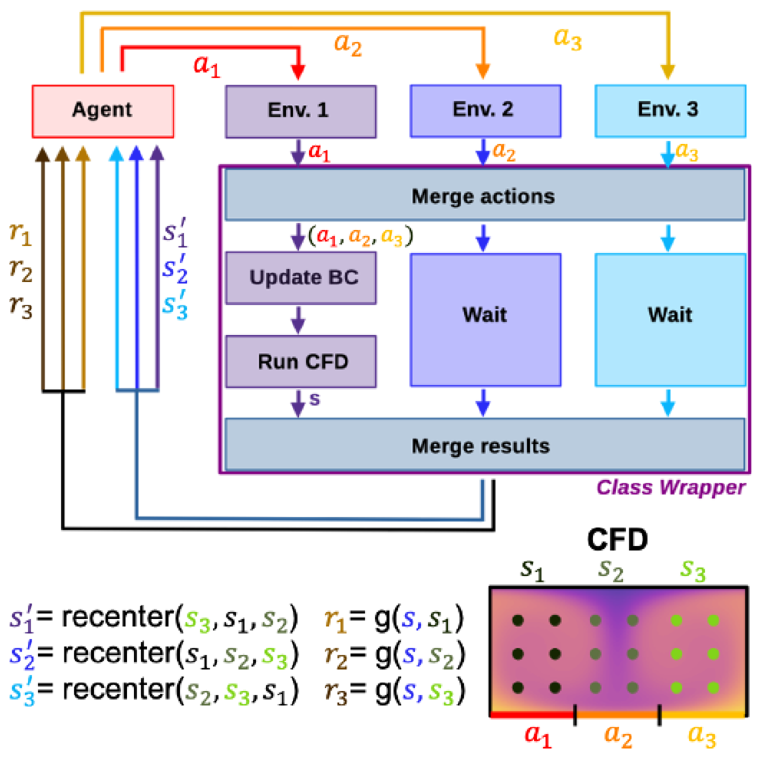

Figure 2: Illustration of the Multi Agent Deep Reinforcement Learning (MARL) framework deployed in [29]. Similarly to the results of [32], MARL is found to effectively solve the curse of dimensionality on the number of action outputs, and to allow the discovery of distributed input distributed output control laws. To do so, each local area of the plant around individual actuators is seen as a local environment, and clones of the same network, that share and reuse data and policies, are used across the plant. The figure is reproduced from [29].

- Proper normalization of the state, action, and reward, from their physically meaningful range to a normalized range typically $O([-1,1])$. The reason for this is that NNs used as universal functional approximators usually rely on nonlinear activation functions that are typically active over the range $[-1,1]$. As a consequence, to avoid network saturation, and take advantage of the non linearity abilities of modern NNs (from which they derive their universal approximator properties), it is crucial to provide NN inputs and outputs in this typical range. Moreover, the shape and formula for the reward should be chosen carefully, to push the DRL optimization algorithm towards discovering valuable strategies, while avoiding pitfalls such as large biases or parasitic flow distortions. The design of such a reward may be challenging and require in-depth human expertise.

- Choice of correct timescales for the deployment of DRL. More specifically, there are two DRL-related critical timescales that need to be chosen properly, as demonstrated in [16]. First, the duration of an individual action should be typically around a tenth of the typical timescale to control. Indeed, the DRL agent both needs to be updating the control value of the plant fast enough to be able to actually control its dynamics, while giving enough time for each action update to have a measurable impact on the state of the system and to allow exploration to effectively take place and result in measurable reward changes. Second, the duration of an episode should be long enough to allow for the plant dynamics to undergo the transient actuated behavior and reach established controlled regime. The time required to do so is largely dependent on the application, but is typically equal to around 5–10 times the dominating time scale in most flow control applications presented so far. Note that, in case there may be novel dynamics appearing for longer actuation time, it may be relevant to combine this with a stochastic episode restart, i.e., when starting a new episode, continuing the previous episode from its last state with probability β or re-starting from a non-actuated state with probability $1-\beta$. This allows to both train the agent regularly on the transitional dynamics, while also exploring plant behaviors appearing after longer actuation durations. Also note that these considerations are independent of the choice of the CFD update timestep (or "mechatronic refresh frequency" in the case of a physical plant), which is typically an additional, much shorter time scale compared with the DRL action update time scale. This implies, in particular, that an interpolation function should be designed to apply the control value at each CFD iteration, see, e.g., [17].

- As discussed above, a key enabler for effective DRL training lies in avoiding the curse of dimensionality, which can make any form for learning so expensive that it becomes completely intractable, even with advanced acceleration of the episodes gathering. Many studies have now illustrated the important of avoiding in particular the curse of dimensionality on the control space dimensionality. In particular, [32], [26] and [29] demonstrated that in the case of DIDO systems, while the Single Agent Reinforcement Learning (SARL) method, i.e. leveraging a single multiple-output DRL agent to ingest all inputs and generate all the control outputs, results in an untractable problem, MARL is able, by enforcing invariants into the deployment methodology of the DRL controller, to effectively control such systems. This simple technical trick has been the single most important enabler that has allowed, so far, to increase the complexity of DRL for flow control applications. When deploying MARL, it is also crucial to choose meaningful width of the surrounding domain as an input to each MARL Agent, and to define rewards that make sense both locally and globally to obtain effective learning.

- One additional fundamental DRL requirement is the necessity to learn from non-correlated plant trajectories in its phase space. Indeed, most DRL algorithms need to sample independent trajectories in the phase space in order to build robust stochastic estimates for the quality of their policy during the learning phase. In other words, this means that the trajectories obtained should be a representative sample of the behavior of the plant across its full behavior range. Moreover, NNs as function approximators are famously known for their challenges learning from self-correlated data, with doing so leading to a number of training issues and biases. On the one hand this non-correlation condition is usually naturally enforced in SARL DRL deployments, as a batch of quite different training trajectories naturally arises from the sampling of different episodes. This occurs due to the fact that episodes are randomized through the stochastic exploration term used to explore the phase space of the plant. However, on the other hand, this is not always the case when deploying MARL. More specifically, when a MARL approach is used, different trajectories sampled by neighboring agents are usually strongly correlated with each others, due to the natural spatial correlation present in most systems. This implies that, if a MARL training case provides for example 10 trajectories per CFD simulation, a commonly used DRL training batch size of 20 trajectories may be too small to ensure that enough diversity of non-correlated trajectories are sampled (i.e., the 20 MARL episodes collected truly contain roughly only 2 independent system trajectories), and one may need to increase the batch size to cover more CFD simulations.

- Identifying the computational bottlenecks appearing in DRL deployments is another crucial aspect of DRL for AFC. Typically, DRL is not the bottleneck, at least in CFD-driven flow control applications. [15] finds, for example, that over 99% of the computational cost is happening in the CFD, rather than the DRL. However, the computational cost of the DRL may still explode if applied improperly. In particular when large

DIDO systems are being controlled, failure to apply MARL with restricted neighboring DRL input may result in deploying many MARL agents having very large inputs. This can cause the cost of the DRL part to explode, both in terms of memory and CPU use. This is usually suboptimal and not needed, as most DIDO systems present some form of locality, so that a reduced-size, localized input is better suited than providing the full global state to each of the local MARL agents that are being deployed. Moreover, providing a local input may help the agent to use relevant information by not overloading it with lots of non meaningful input data, and speed up the learning as a consequence. Note that the synchronization of the several MARL actions involved in a specific CFD simulation needs attention, so that the global CFD action is applied consistently - this may require some specific synchronization mechanisms to be added, as illustrated in Fig. 2.

- How the action is applied into the plant is also important. In particular, action update, if not properly smoothened and applied, may result in jumps in the boundary conditions used in the CFD solvers. As a consequence, care must be taken in the transitions between consecutive actions in order not to introduce discontinuities that can produce sudden numerical failures. It should be noted that in CFD simulations, abrupt changes in boundary conditions can also lead to a considerable increase in computational cost. In practice, these issues are solved by using a smooth transition function between action updates (with the smoothing active both in time and space), either it is linear [17], or more sophisticated and based on infinitely continuous functions as in [47] (which uses smooth transition functions).

- Providing sufficient input variability to the DRL agent is also necessary to obtain effective control laws. Typically, most DRL agents rely on simple feedforward NNs.

This implies that, while there may be some variability in the output of the agent during the training phase due to the exploration noise, most common DRL agents will only present variability in their output during deterministic evaluation if there exists variability in their input. This is especially crucial for applications where the system is initially in a stable equilibrium situation with little or no variability taking place, and for which a non-trivial actuation, for example a phase of initial periodic forcing, is necessary to trigger significant state variations: due to its feedforward nature, it may be difficult for a DRL agent to generate the corresponding output variations if its inputs remain close to constant. A number of solutions can be deployed in practice to mitigate this difficulty. One can, for example, deploy recurrent neural networks using Long Short Term Memory (LSTM) layers. This, however, presents a number of technical challenges and is usually quite computationally expensive. Another simple solution is to provide the action a_t of the DRL agent as one of the state components for the next state observation s_{t+1} - the DRL agent can then apply, for example, a continuous increase to the action, or any other function to link the action at step t to the one at step $t+1$. Other techniques can also be considered, such as applying attention-based mechanisms [63]. Finally, the DRL practitioner can, in cases where a clear frequency of interest is identified, e.g. through linear analysis techniques, provide a hand-crafted synthetic input to the DRL agent (either for example a sinusoidal input tuned at the right frequency, or a signal ramp aliasing in time of the form $ramp(t) = \alpha t[MOD]1$, where $[MOD]1$ indicates that the value 1 is subtracted from the ramp as many times as needed so that the obtained value is in the range $[0,1]$, and α is used to control the wrap-up aliasing time of the ramp).

Naturally, these specific technicalities should be applied together with choices for the state, action, reward, that are physically meaningful and contain enough information for the optimization process to take place.

Applying a combination of MARL setup, fast parallelized CFD, experience gathering from several environments in parallel, and the technical tricks and tips mentioned above, has already allowed to apply DRL for a number of complex flow control situations at increasingly higher Reynolds numbers and complexity levels [64, 26]. Deploying these in ever more challenging cases is an ongoing process in the DRL for flow control community, with great activity currently taking place. However, even a combination of these will ultimately prove too computationally expensive to yield further progress on their own. In the following, we speculate on what further ingredients can be brought to continue advancing the use of DRL for flow control, when the next complexity cost wall is hit.

2.2 Overcoming the next "complexity wall": possibilities opened by offline learning, Group-Convolutional Neural Networks, and deployment in experiments

It is likely that DRL for flow control trained through CFD simulations will face computational cost challenges in the years to come as the Reynolds number and complexity of the flows being studied are further increased. Indeed, the cost of the CFD simulations is the dominating computational cost of DRL for AFC training, and current state-of-the-art applications such as [47] or [26] already require using up to several hundreds of CPU cores for several weeks in a row to generate the CFD online training data. As a consequence, large amounts of computational power are already needed to perform training, and a hard CFD cost bottleneck will likely soon be encountered as Re is continuously increased. Methods based on reduced order modeling and surrogate models of the flow dynamics can be used to accelerate learning [65, 66], however, these are relying on the ability to build an effective and accurate model of the flow dynamics, which is a famously difficult problem as the

Reynolds number and flow complexity is increased.

Fortunately, there are several more techniques that can still be implemented to alleviate the cost of learning DRL control strategies for higher complexity flows. First, the MARL approach only takes advantage, at the moment, of the global invariants present in the flow to control. I.e., the MARL technique allows to enforce, for example, translational invariance of the control policy across the domain location, but it does not take advantage of local symmetries of the flow, such as right-left symmetries. Such a category of local symmetries, which is usually presented as a group-equivariant property in the mathematically oriented literature [67], and is illustrated in Fig. 3, can be effectively enforced using Group Convolutional Neural Networks (G-CNNs). G-CNNs, and other similar networks, leverage dimension lifting of the NN state to enforce groups of invariants or symmetries on the local policy itself [68], as illustrated in Fig. 4 in a simple example case. We note that, at the moment, the literature on this topic is still consolidating, with different nomenclature being used. For example Group Convolutional Neural Networks [68, 69, 70], homomorphic neural networks [71], or equivariant neural networks [72, 67, 73, 74], are all terms describing the same family of ideas. Though deploying the same general ideas, exact implementations may slightly differ, from leveraging simple lifting layer technical tricks [68], to implementing more sophisticated continuous equivariances through, for example, steerable kernel basis functions [75]. G-CNNs have already proved their ability to increase the computational efficiency of a number of image recognition [68], robotics [76], and drone control [77], tasks, and similar gains could be expected when using G-CNNs within DRL for flow control.

Another promising direction consists in leveraging offline DRL [78, 79]. At present, most DRL applications (and all DRL for flow control applications that we know of) rely purely on online training (possibly slightly accelerated by transfer learning, as presented first in [22]), i.e., all the training is performed by directly interacting with the plant. This

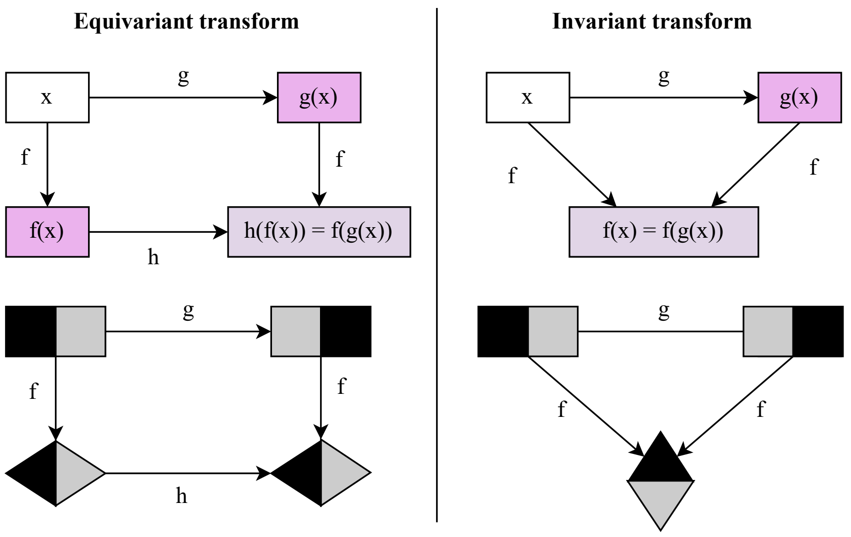

Figure 3: Illustration of equivariant vs. invariant transforms. While the laws of physics enforce invariants in the underlying equations, this usually results in equivariant policies when performing control of physical systems. Though the terms equivariant and invariant have been used somewhat interchangeably within DRL applications to AFC [32], the more rigorous mathematically-focused fundamental RL literature usually uses the equivariant nomenclature. Taking advantage of invariance or equivariance (depending on what category applies to a given problem) allows to considerably reduce the cost of exploration and policy learning. This can be obtained by leveraging for example MARL approaches when some reward locality is also enforced and the optimal policy should be reproduced across the width of the domain [32, 26, 29], or by using homomorphic (also sometimes known as "equivariant" or "group convolutional") neural networks when local symmetries are present [71]. Combining MARL and group convolutional neural networks to take advantage of both policy invariance across the domain, reward locality, and local symmetries in the physics, is expected to be a promising milestone for future DRL for AFC studies.

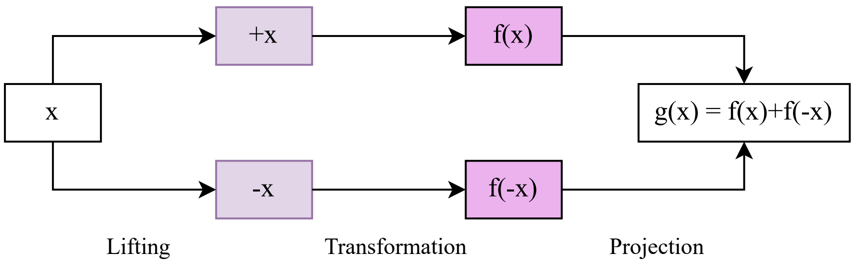

Figure 4: Simple illustration of the "lifting, transformation, projection" method for implementing group invarinats, equivariants, and symmetries in neural networks. By lifting the state x to $(+x, -x)$, and by applying a similar transformation on both lifted states before projecting, a function g that enforces the symmetries corresponding to the lifting group of transformation (here, symmetry around 0, i.e. parity) can be obtained by construction. This allows to build local symmetries in, for example, policy and Q-networks. Imposing such symmetries in the architecture of networks, rather than learning these directly from data for a non-constrained network, is both more robust and more data-efficient [68], which is critical in the case of online DRL when sampling plant trajectories can be expensive. Applying the same transformation on the different lifted states can be obtained, for example, either by applying the same fully connected neural network f on all lifted states, or by applying a single convolutional neural network with its convolution dimension acting on the lifting dimension (hence, the expression Group Convolutional Neural Network: convolutions acting on the lifting dimension can be used to enforce group equivariance or invariance).

allows to deploy simpler DRL algorithms that do not need to overly consider issues arising from off-policy sampling and distribution shift, but comes at a high cost: every new DRL training needs to start learning from scratch and to gather large amounts of online interaction data, which is usually expensive to gather, and it is not possible to easily leverage previously gathered data. Offline learning, by contrast, allows to train DRL agents by querying a database of previous examples of (state, action, reward) triplets transitions [78]. As a consequence, the dataset available for offline training of a DRL agent in a given case is as big as all simulations that have been run for this case under actuation in the past, rather than limited to the current online interaction data that are being generated by the specific ongoing training. Since it is common to need quite many iterations and trials-and-errors to find the right DRL setup, metaparameters, and MARL strategy, in order to obtain effective learning, offline DRL methods could potentially offer an order-of-magnitude gain

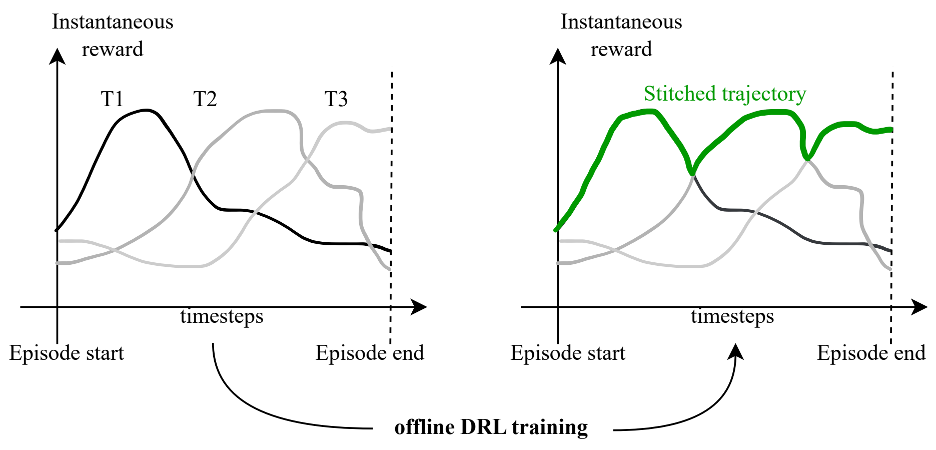

Figure 5: Idealized illustration of the trajectory stitching concept, which is one of the advantages of offline DRL compared to, for example, imitation learning. More specifically, offline learning can stitch together locally-optimal trajectory segments, and learn a policy better than any policy or trajectory present in the offline training database. This is illustrated here by stitching segments from the trajectory dataset $\{T1, T2, T3\}$ into a stitched trajectory that presents equal or higher instantaneous reward than any offline dataset trajectory, and is, therefore, more successful than any of the policies provided in the dataset.

in the volume of data available to train each individual agent by pooling data across specific training runs. While offline DRL has historically been challenging, new algorithms [80, 81, 82, 83] and frameworks [84] are now making it a credible technical solution. Offline DRL can, naturally, be fine-tuned through online interaction if necessary. As a side note, offline DRL learning is different, and potentially more powerful, than imitation DRL learning [85, 86]. Indeed, while imitation DRL only learns to reproduce examples of strategies that may be suboptimal (and require the user to know one such strategy in the first place), offline DRL is, in theory, fully able to learn a policy that is more optimal than any policy that has been used to generate its training dataset, by leveraging a number of mechanisms, such as trajectories stitching [87, 88, 89], as illustrated in Fig. 5.

Therefore, there is hope to sustain the trend in increased DRL-in-CFD-flow-control complexity for a few more years by leveraging larger CPU and graphics processing unit (GPU) clusters to run the CFD itself [42], as well as taking advantage of effective MARL strategies, G-CNNs, and offline learning. However, approaches based on deploying DRL for flow control in CFD simulations will always, ultimately, be limited by our ability to simulate flows at high Reynolds number. As a consequence, DRL for flow control will have to increasingly move from controlling CFD simulations, to control real-world flows in experiments, an approach that has already been demonstrated [23, 90].

Interestingly enough, this may actually, by contrast to what is observed in for example robotics application of DRL, improve our ability to collect large amounts of data. Indeed, in the world of robotics, robots are typically moving and actuated at a relatively low speed that is limited by their physical shape and inertia. As a consequence, simulations are often used within robotics as a way to gather larger amounts of data in limited time than what is allowed using physical test robots, by taking advantage of relatively inexpensive solid dynamics simulations. In fluid mechanics and active flow control, by contrast, CFD simulations are so computationally expensive that they usually run much slower than real-world experiments. The main motivation for applying DRL in simulations has been simpler deployment and iteration of the DRL setup, and easier access to direct online measurements in whatever location of the flow is needed, since properly setting up an experiment takes significant time overhead, is not as easily adapted, and both flow sensors and actuators are complex to build and install.

Therefore, the CFD-driven approach has allowed to quickly iterate in order to better understand how limited flow information can be leveraged, by, for example, taking better advantage of a limited number of surface probes [19, 91, 92, 93]. As a next step, moving to real world flow control experiments now that we better understand how DRL control should be applied will allow to better scale up against flow complexity than what can be obtained in simulation. For example, in the case of turbulent channels, while the intrinsic flow complexity of course evolves similarly in simulations and experiments, the computational resources needed to simulate the flow increases at best as around Re^3 as previously mentioned, while by contrast, increasing the Reynolds number in an experiment by increasing the flow speed reduces the typical timescales at play, and allows to gather more relevant flow dynamics realizations for a constant experiment duration. Therefore, it remains an open question to discover how the time requirement for discovering effective DRL flow control strategies in experiments will scale with higher Reynolds number, since it is hard, in particular, to estimate a priori how complex a given flow is from a control point of view. However, careful optimism can still be held on the topic, and the progresses coming in the next few years will be critical to better understanding the challenges associated with these aspects.

3 DRL for flow control in a broader scientific context: advancing practical applications and fundamental physical knowledge

As a word of conclusion, we want to offer a few thoughts about the importance and future role of DRL for flow control in fluid mechanics. Due to the nonlinear nature of many flows, and the difficulties this presents for most traditional control methods, it remains challenging to assess the potential gains that can be achieved by active flow control - we do not know in advance "how good" and "how efficient" an active flow control strategy can, fundamentally, be obtained in a given flow configuration.

However, we have observed in several domains the potential of DRL for finding previously unknown optima and strategies in complex non-linear systems. The canonical example on this aspect is the game of Go [13]: not only has DRL offered an algorithm that is able to beat the best human players (a feat that was thought, as recently as early 2017, to still be several decades ahead in the future), but AlphaGo has also discovered fundamentally novel strategies and moves, especially in the opening phase of the game, that very clearly outperform the best strategies that had been discovered by humans over several millennia of playing Go. This has led to a true revolution in how humans understand and play Go, to the point where a good-level professional Go player from 2023, trained by AlphaGo to understand and apply these new strategies, could likely beat the best world player from 2017, who was unaware of these playing strategies [94].

There are reasons to hope that DRL for flow control can potentially lead to similar revolutions in active flow control at large (including, for example, fields such as plasma control [95, 96] or instability control). For example, DRL is able to significantly improve on classical harmonic strategies for the control of 3D cylinder wakes [47]. Interestingly, the strategies found by the DRL agent are relatively close to the best harmonic forcing strategies - however, the minor differences introduced by the DRL control law allow to nearly double the drag reduction achieved. Similarly, DRL applications to the control of 3D turbulent channel flows [26] has revealed that simple strategies found by DRL, such as bang-bang like controllers, can widely overperform compared with the opposition control derived from linear analysis and commonly used so far - a finding that seems to be recurrent for several classes of systems that behave as low-pass filters [97]. These more effective strategies are not just artifacts, but they do correspond to fundamental changes in the properties and dominating physical phenomena happening in the flow undergoing control, as illustrated in Fig. 6.

Therefore, DRL for flow control offers both an opportunity to improve on state-of-the-art control strategies, and to gain fundamentally new knowledge about the dynamics of the systems that it controls - since a good way to gain understanding about the wider phase-space behavior of a non-linear system is to control it to go out of its natural stable dynamics. From a practical point of view, DRL is a promising practical method for intelligent close-loop control, since, once trained, the DRL controller is computationally cheap to deploy, and

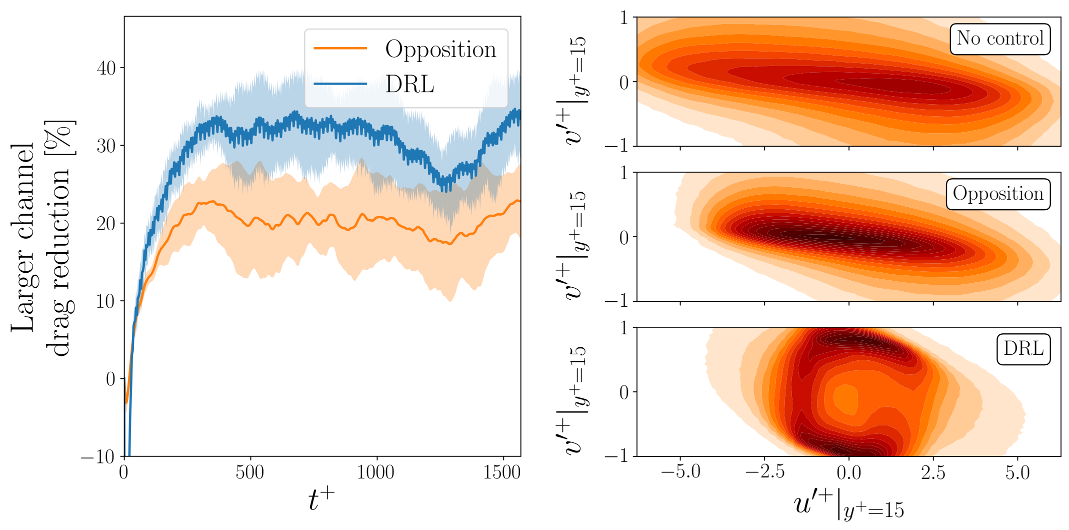

Figure 6: Illustration of the effectiveness of DRL control compared to state-of-the-art opposition control in reducing the drag in a turbulent channel. While the diagnostic plots show that opposition control reduces, but does not suppress, ejection events, DRL is able to fundamentally change the properties of the flow. In-depth analysis shows that the DRL control law is more akin to a bang-bang controller, which introduces strong nonlinearity in the control law [26] compared with the more traditional opposition control usually deployed. Figure reproduced from [26].

does not request access to the full details of the flow - unlike, for example, adjoint-based methods.

To conclude, DRL for flow control has been attracting increasing attention over the last few years. Since its first applications in 2018-2019, DRL for flow control has been applied to more and more complex flows, by increasing the Reynolds number, and moving from simplistic 2D flows towards more and more realistic 3D configurations. These early successes of DRL for flow control have made it into a fast moving, promising methodology, with a growing number of groups from around the world looking into how to push it further. Challenges and opportunities lie ahead, and it remains an open question to determine how much further the complexity of DRL flow control laws obtained in CFD simulations can be pushed. Indeed, gathering enough trajectories of flow systems from CFD simulations is getting quickly very computationally expensive, due to the intrinsic cost of running CFD simulations of complex flows. However, there is hope: in particular, the rapid progresses of DRL for flow control have been made possible largely by the existence of a growing community of users sharing their codes and developments as open source materials alongside their publications, and we can hope that similar trends will emerge also when DRL is increasingly applied to the control of experimental flows. If successful, DRL flow control in complex experiments could contribute to bringing new insights to our understanding of flow stability and controllability.

References

[1] C. Vignon, J. Rabault, and R. Vinuesa, "Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions," *Physics of Fluids*, vol. 35, no. 3, 2023.

[2] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem, "A review on deep reinforcement learning for fluid mechanics," *Computers & Fluids*, vol. 225, p. 104973, 2021.

[3] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual review of fluid mechanics*, vol. 52, pp. 477–508, 2020.

[4] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, and K. Gustavsson, "Zermelo's problem: optimal point-to-point navigation in 2D turbulent flows using RL," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 10, p. 103138, 2019.

[5] S. Verma, G. Novati, and P. Koumoutsakos, "Efficient collective swimming by harnessing vortices through deep reinforcement learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 23, pp. 5849–5854, 2018.

[6] J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, and E. Hachem, "Direct shape optimization through deep reinforcement learning," 2020.

[7] J. Rabault and A. Kuhnle, *Deep Reinforcement Learning Applied to Active Flow Control*, p. 368–390. Cambridge University Press, 2023.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[10] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[11] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[12] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[14] S. L. Brunton and B. R. Noack, "Closed-loop turbulence control: Progress and challenges," *Applied Mechanics Reviews*, vol. 67, no. 5, p. 050801, 2015.

[15] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control," *Journal of Fluid Mechanics*, vol. 865, pp. 281–302, 2019.

[16] J. Rabault and A. Kuhnle, “Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach,” *Physics of Fluids*, vol. 31, no. 9, p. 094105, 2019.

[17] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang, “Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through DRL,” *Physics of Fluids*, vol. 32, no. 5, p. 053605, 2020.

[18] H. Xu, W. Zhang, J. Deng, and J. Rabault, “Active flow control with rotating cylinders by an artificial neural network trained by deep reinforcement learning,” *Journal of Hydrodynamics*, vol. 32, no. 2, pp. 254–258, 2020.

[19] R. Paris, S. Beneddine, and J. Dandois, “Robust flow control and optimal sensor placement using deep reinforcement learning,” *Journal of Fluid Mechanics*, vol. 913, p. A25, 2021.

[20] J. Li and M. Zhang, “Reinforcement-learning-based control of confined cylinder wakes with stability analyses,” *Journal of Fluid Mechanics*, vol. 932, p. A44, Feb. 2022.

[21] J. Rabault, F. Ren, W. Zhang, H. Tang, and H. Xu, “Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization,” *Journal of Hydrodynamics*, vol. 32, no. 2, pp. 234–246, 2020.

[22] F. Ren, J. Rabault, and H. Tang, “Applying deep reinforcement learning to active flow control in weakly turbulent conditions,” *Physics of Fluids*, vol. 33, no. 3, p. 037121, 2021.

[23] D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, “Reinforcement learning for bluff body active flow control in experiments and simulations,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 42, pp. 26091–26098, 2020.

[24] R. Vinuesa, O. Lehmkuhl, A. Lozano-Durán, and J. Rabault, “Flow control in wings and discovery of novel approaches via deep reinforcement learning,” *Fluids*, vol. 7, no. 2, p. 62, 2022.

[25] W. Chen, Q. Wang, L. Yan, G. Hu, and B. R. Noack, “Deep reinforcement learning-based active flow control of vortex-induced vibration of a square cylinder,” *Physics of Fluids*, vol. 35, no. 5, 2023.

[26] L. Guastoni, J. Rabault, P. Schlatter, H. Azizpour, and R. Vinuesa, “DRL for turbulent drag reduction in channel flows,” *The European Physical Journal E*, vol. 46, no. 4, p. 27, 2023.

[27] T. Sonoda, Z. Liu, T. Itoh, and Y. Hasegawa, “Reinforcement learning of control strategies for reducing skin friction drag in a fully developed turbulent channel flow,” *Journal of Fluid Mechanics*, vol. 960, p. A30, 2023.

[28] G. Beintema, A. Corbetta, L. Biferale, and F. Toschi, “Controlling rayleigh–bénard convection via reinforcement learning,” *Journal of Turbulence*, vol. 21, no. 9-10, pp. 585–605, 2020.

[29] C. Vignon, J. Rabault, J. Vasanth, F. Alcántara-Ávila, M. Mortensen, and R. Vinuesa, “Effective control of two-dimensional Rayleigh–Bénard convection: Invariant multi-agent reinforcement learning is all you need,” *Physics of Fluids*, vol. 35, no. 6, 2023.

[30] F. Pino, L. Schena, J. Rabault, and M. A. Mendez, “Comparative analysis of machine learning methods for active flow control,” 2022.

[31] M. A. Bucci, O. Semeraro, A. Allauzen, G. Wisniewski, L. Cordier, and L. Mathelin, “Control of chaotic systems by deep reinforcement learning,” *Proceedings of the Royal Society A*, vol. 475, no. 2231, p. 20190351, 2019.

[32] V. Belus, J. Rabault, J. Viquerat, Z. Che, E. Hachem, and U. Reglade, “Exploiting locality and translational invariance to design effective DRL control of the 1-dimensional unstable falling liquid film,” *AIP Advances*, vol. 9, no. 12, p. 125014, 2019.

[33] X. Yan, J. Zhu, M. Kuang, and X. Wang, “Aerodynamic shape optimization using a novel optimizer based on machine learning techniques,” *Aerospace Science and Technology*, vol. 86, pp. 826–835, 2019.

[34] H. Keramati, F. Hamdullahpur, and M. Barzegari, “Deep reinforcement learning for heat exchanger shape optimization,” *International Journal of Heat and Mass Transfer*, vol. 194, p. 123112, 2022.

[35] C. Foucart, A. Charous, and P. F. Lermusiaux, “Deep reinforcement learning for adaptive mesh refinement,” *Journal of Computational Physics*, vol. 491, p. 112381, 2023.

[36] J. Keim, A. Schwarz, S. Chiocchetti, C. Rohde, and A. Beck, “A reinforcement learning based slope limiter for two-dimensional finite volume schemes,” in *International Conference on Finite Volumes for Complex Applications*, pp. 209–217, Springer, 2023.

[37] M. Dellnitz, E. Hüllermeier, M. Lücke, S. Ober-Blöbaum, C. Offen, S. Peitz, and K. Pfannschmidt, “Efficient time-stepping for numerical integration using reinforcement learning,” *SIAM Journal on Scientific Computing*, vol. 45, no. 2, pp. A579–A595, 2023.

[38] S. Verma, G. Novati, and P. Koumoutsakos, “Efficient collective swimming by harnessing vortices through deep reinforcement learning,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, pp. 5849–5854, June 2018.

[39] M. Kurz, P. Offenhäuser, and A. Beck, “DRL for turbulence modeling in large eddy simulations,” *International Journal of Heat and Fluid Flow*, vol. 99, p. 109094, 2023.

[40] G. Novati, H. L. de Laroussilhe, and P. Koumoutsakos, “Automating turbulence modeling by multi-agent reinforcement learning,” *Nature Machine Intelligence*, vol. 3, pp. 87–96, 2021.

[41] A. Beck and M. Kurz, “Toward discretization-consistent closure schemes for large eddy simulation using reinforcement learning,” *arXiv preprint arXiv:2309.06260*, 2023.

[42] M. Kurz, P. Offenhäuser, and A. Beck, “Deep reinforcement learning for turbulence modeling in large eddy simulations,” *International Journal of Heat and Fluid Flow*, vol. 99, p. 109094, 2023.

[43] H. J. Bae and P. Koumoutsakos, “Scientific multi-agent reinforcement learning for wall-models of turbulent flows,” *Nature Communications*, vol. 13, no. 1, p. 1443, 2022.

[44] T. Duriez, S. L. Brunton, and B. R. Noack, *Machine learning control-taming nonlinear dynamics and turbulence*, vol. 116. Springer, 2017.

[45] R. Vinuesa and S. L. Brunton, “Enhancing computational fluid dynamics with machine learning,” *Nature Computational Science*, vol. 2, no. 6, pp. 358–366, 2022.

[46] R. Vinuesa, O. Lehmkuhl, A. Lozano-Durán, and J. Rabault, “Flow control in wings and discovery of novel approaches via deep reinforcement learning,” *Fluids*, vol. 7, no. 2, 2022.

[47] P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, O. Lehmkuhl, and R. Vinuesa, “Active flow control for three-dimensional cylinders through deep reinforcement learning,” *arXiv preprint arXiv:2309.02462*, 2023.

[48] A. Kuhnle, M. Schaarschmidt, and K. Fricke, "Tensorforce: a tensorflow library for applied reinforcement learning." Web page, 2017.

[49] J. Davidson, E. Holly, T. Boyd, S. Yue, R. Ormandi, K. Lee, A. Greenberg, A. Yazdanbakhsh, Y. Lu, G. Jain, C. Angermueller, M. Daoust, and A. Wood, "Tf-agents: A reliable, scalable and easy to use tensorflow library for contextual bandits and reinforcement learning." Web page, 2017.

[50] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[51] J. Weng, H. Chen, D. Yan, K. You, A. Duburcq, M. Zhang, Y. Su, H. Su, and J. Zhu, "Tianshou: A highly modularized deep reinforcement learning library," *Journal of Machine Learning Research*, vol. 23, no. 267, pp. 1–6, 2022.

[52] Q. Wang, L. Yan, G. Hu, C. Li, Y. Xiao, H. Xiong, J. Rabault, and B. R. Noack, "Drlinfluids: An open-source python platform of coupling deep reinforcement learning and openfoam," *Physics of Fluids*, vol. 34, no. 8, 2022.

[53] Y. Mao, S. Zhong, and H. Yin, "Drlfluent: A distributed co-simulation framework coupling deep reinforcement learning with ansys-fluent on high-performance computing systems," *Journal of Computational Science*, p. 102171, 2023.

[54] M. Kurz, P. Offenhäuser, D. Viola, M. Resch, and A. Beck, "Relexi—a scalable open source reinforcement learning framework for high-performance computing," *Software Impacts*, vol. 14, p. 100422, 2022.

[55] S. Monk, *Programming Arduino: getting started with sketches*. McGraw-Hill Education, 2023.

[56] A. Saiz-Vela, P. Fontova, T. Pallejá, M. Tresanchez, J. A. Garriga, and C. Roig, "A low-cost development platform to design digital circuits on fpgas using open-source software and hardware tools," in *2020 XIV Technologies Applied to Electronics Teaching Conference (TAEE)*, pp. 1–8, IEEE, 2020.

[57] J. Rabault, G. Sutherland, A. Jensen, K. H. Christensen, and A. Marchenko, "Experiments on wave propagation in grease ice: combined wave gauges and particle image velocimetry measurements," *Journal of Fluid Mechanics*, vol. 864, pp. 876–898, 2019.

[58] J. Rabault, T. Nose, G. Hope, M. Müller, Ø. Breivik, J. Voermans, L. R. Hole, P. Bohlinger, T. Waseda, T. Kodaira, *et al.*, "Openmetbuoy-v2021: An easy-to-build, affordable, customizable, open-source instrument for oceanographic measurements of drift and waves in sea ice and the open ocean," *Geosciences*, vol. 12, no. 3, p. 110, 2022.

[59] J. T. Ølberg, P. Bohlinger, Ø. Breivik, K. H. Christensen, B. R. Furevik, L. R. Hole, G. Hope, A. Jensen, F. Knoblauch, N.-T. Nguyen, *et al.*, "Wave measurements using open source ship mounted ultrasonic altimeter and motion correction system during the one ocean expedition," *arXiv preprint arXiv:2310.03101*, 2023.

[60] A. Liberzon, D. Lasagna, M. Aubert, P. Bachant, T. Käufer, jakirkham, A. Bauer, B. Vodenicharski, C. Dallas, J. Borg, tomerast, and ranleu, "OpenPIV/openpiv-python: OpenPIV - Python (v0.22.2) with a new extended search PIV grid option," July 2020.

[61] H. Choi and P. Moin, "Grid-point requirements for large eddy simulation: Chapman's estimates revisited," *Physics of Fluids*, vol. 24, p. 011702, 2012.

[62] S. Peitz, J. Stenner, V. Chidananda, O. Wallscheid, S. L. Brunton, and K. Taira, "Distributed control of partial differential equations using convolutional reinforcement learning," *arXiv preprint arXiv:2301.10737*, 2023.

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[64] P. Varela, P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, L. M. García-Cuevas, O. Lehmkuhl, and R. Vinuesa, “DRL for flow control exploits different physics for increasing Reynolds number regimes,” *Actuators*, vol. 11, no. 12, 2022.

[65] X. Dong, H. Hong, X. Deng, W. Zhong, and G. Hu, “Surrogate model-based deep reinforcement learning for experimental study of active flow control of circular cylinder,” *Physics of Fluids*, vol. 35, no. 10, 2023.

[66] A. J. Linot, K. Zeng, and M. D. Graham, “Turbulence control in plane couette flow using low-dimensional neural ode-based models and deep reinforcement learning,” *International Journal of Heat and Fluid Flow*, vol. 101, p. 109139, 2023.

[67] N. Keriven and G. Peyré, “Universal invariant and equivariant graph neural networks,” in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[68] E. J. Bekkers, M. W. Lafarge, M. Veta, K. A. Eppenhof, J. P. Pluim, and R. Duits, “Roto-translation covariant convolutional networks for medical image analysis,” in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, pp. 440–448, Springer, 2018.

[69] R. Liu, F. Lauze, E. Bekkers, K. Erleben, and S. Darkner, “Group convolutional neural networks for dwi segmentation,” in *Geometric Deep Learning in Medical Image Analy-*

sis, pp. 96–106, PMLR, 2022.

[70] D. M. Knigge, D. W. Romero, and E. J. Bekkers, “Exploiting redundancy: Separable group convolutional networks on lie groups,” in *International Conference on Machine Learning*, pp. 11359–11386, PMLR, 2022.

[71] E. van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling, “Mdp homomorphic networks: Group symmetries in reinforcement learning,” in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 4199–4210, Curran Associates, Inc., 2020.

[72] T. Cohen and M. Welling, “Group equivariant convolutional networks,” in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 2990–2999, PMLR, 20–22 Jun 2016.

[73] C. Esteves, “Theoretical aspects of group equivariant neural networks,” *arXiv preprint arXiv:2004.05154*, 2020.

[74] T. S. Cohen, M. Geiger, and M. Weiler, “A general theory of equivariant cnns on homogeneous spaces,” in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[75] J. Brandstetter, R. Hesselink, E. van der Pol, E. J. Bekkers, and M. Welling, “Geometric and physical quantities improve e (3) equivariant message passing,” *arXiv preprint arXiv:2110.02905*, 2021.

[76] D. Wang, M. Jia, X. Zhu, R. Walters, and R. Platt, “On-robot learning with equivariant models,” *arXiv preprint arXiv:2203.04923*, 2022.

[77] J. Huang, W. Zeng, H. Xiong, B. R. Noack, G. Hu, S. Liu, Y. Xu, and H. Cao, "Symmetry-informed reinforcement learning and its application to the attitude control of quadrotors," *IEEE Transactions on Artificial Intelligence*, 2023.

[78] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[79] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*, pp. 104–114, PMLR, 2020.

[80] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[81] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[82] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[83] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," *Advances in neural information processing systems*, vol. 34, pp. 28954–28967, 2021.

[84] T. Seno and M. Imai, "d3rlpy: An offline deep reinforcement learning library," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 14205–14224, 2022.

[85] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, 2004.

[86] A. Kumar, J. Hong, A. Singh, and S. Levine, "Should i run offline reinforcement learning or behavioral cloning?," in *International Conference on Learning Representations*, 2021.

[87] H. Xu, L. Jiang, L. Jianxiong, and X. Zhan, "A policy-guided imitation approach for offline reinforcement learning," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 4085–4098, Curran Associates, Inc., 2022.

[88] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine, "Cog: Connecting new skills to past experience with offline reinforcement learning," *arXiv preprint arXiv:2010.14500*, 2020.

[89] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 1273–1286, Curran Associates, Inc., 2021.

[90] E. Amico, G. Cafiero, and G. Iuso, "Deep reinforcement learning for active control of a three-dimensional bluff body wake," *Physics of Fluids*, vol. 34, p. 105126, 10 2022.

[91] R. Castellanos, G. Cornejo Maceda, I. De La Fuente, B. Noack, A. Ianiro, and S. Discetti, "Machine-learning flow control with few sensor feedback and measurement noise," *Physics of Fluids*, vol. 34, no. 4, 2022.

[92] Q. Wang, L. Yan, G. Hu, W. Chen, and B. R. Noack, "Dynamic feature-based deep reinforcement learning for flow control of circular cylinder with sparse surface pressure sensing," *arXiv preprint arXiv:2307.01995*, 2023.

[93] R. Paris, S. Beneddine, and J. Dandois, “Reinforcement-learning-based actuator selection method for active flow control,” *Journal of Fluid Mechanics*, vol. 955, p. A8, 2023.

[94] J. Kang, J. S. Yoon, and B. Lee, “How ai-based training affected the performance of professional go players,” in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2022.

[95] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, *et al.*, “Magnetic control of tokamak plasmas through deep reinforcement learning,” *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[96] J. Seo, Y.-S. Na, B. Kim, C. Lee, M. Park, S. Park, and Y. Lee, “Feedforward beta control in the kstar tokamak by deep reinforcement learning,” *Nuclear Fusion*, vol. 61, no. 10, p. 106010, 2021.

[97] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus, “Is bang-bang control all you need? solving continuous control with bernoulli policies,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 27209–27221, 2021.