Eur. Phys. J. E (2023) 46:27
https://doi.org/10.1140/epje/s10189-023-00285-8

THE EUROPEAN PHYSICAL JOURNAL E

Regular Article - Flowing Matter

Deep reinforcement learning for turbulent drag reduction in channel flows

Luca Guastoni[1,2,a], Jean Rabault[3,b], Philipp Schlatter[1,2,c], Hossein Azizpour[4,2,d], and Ricardo Vinuesa[1,2,e]

[1] FLOW, Engineering Mechanics, KTH Royal institute of Technology, 100 44 Stockholm, Sweden
[2] Swedish e-Science Research Centre (SeRC), 100 44 Stockholm, Sweden
[3] IT Department, Norwegian Meteorological Institute, Postboks 43, 0313 Oslo, Norway
[4] School of Electrical Engineering and Computer Science, KTH Royal institute of Technology, 100 44 Stockholm, Sweden

Received 24 January 2023 / Accepted 19 March 2023 / Published online: 11 April 2023

Abstract We introduce a reinforcement learning (RL) environment to design and benchmark control strategies aimed at reducing drag in turbulent fluid flows enclosed in a channel. The environment provides a framework for computationally efficient, parallelized, high-fidelity fluid simulations, ready to interface with established RL agent programming interfaces. This allows for both testing existing deep reinforcement learning (DRL) algorithms against a challenging task, and advancing our knowledge of a complex, turbulent physical system that has been a major topic of research for over two centuries, and remains, even today, the subject of many unanswered questions. The control is applied in the form of blowing and suction at the wall, while the observable state is configurable, allowing to choose different variables such as velocity and pressure, in different locations of the domain. Given the complex nonlinear nature of turbulent flows, the control strategies proposed so far in the literature are physically grounded, but too simple. DRL, by contrast, enables leveraging the high-dimensional data that can be sampled from flow simulations to design advanced control strategies. In an effort to establish a benchmark for testing data-driven control strategies, we compare opposition control, a state-of-the-art turbulence-control strategy from the literature, and a commonly used DRL algorithm, deep deterministic policy gradient. Our results show that DRL leads to 43% and 30% drag reduction in a minimal and a larger channel (at a friction Reynolds number of 180), respectively, outperforming the classical opposition control by around 20 and 10 percentage points, respectively.

1 Introduction

Turbulent flows are ubiquitous both in nature and in engineering applications, from climate and weather dynamics [1] to wind-turbine engineering [2], and from turbulent blood streams in the human body [3] to hypersonic flows around re-entry vehicles [4]. Hence, turbulence is highly relevant both for economic and environmental reasons. Depending on the intended outcome, it can be desirable to promote turbulence, for example, to enhance mixing in combustion engines [5], or to hinder it, as a mean to reduce the drag on airplane wings and thus reducing the overall fuel consumption [6]. In both cases, some form of flow control needs to be designed and deployed. Open-loop control techniques, independent of the flow state, have been proposed; however, larger gains can be obtained with the use of closed-loop solutions [7]. Recent works have also tried to leverage machine-learning techniques like genetic programming [8] and Bayesian optimization [9] for flow control. While flow control has attracted widespread attention over the years from different fields of the scientific community, turbulent flows exhibit a chaotic nature, they are multi-scale, highly nonlinear and high-dimensional phenomena; furthermore, they are very expensive to simulate numerically in an accurate way. Still today, the challenges brought by turbulence prevent us from finding effective flow control strategies in most realistic applications.

Deep reinforcement learning (DRL) is a mathematical framework that has been used to design and learn control policies, also in physics research. This framework was successfully applied in optics [10], plasma physics [11] and thermodynamics [12]. In fluid dynamics, the potential of DRL algorithms has been assessed for turbulence modelling [13] and drag reduction [14].

T.I.: Quantitative AI in Complex Fluids and Complex Flows: Challenges and Benchmarks. Guest editors: Luca Biferale, Michele Buzzicotti, Massimo Cencini

[a] e-mail: guastoni@mech.kth.se (corresponding author)
[b] e-mail: jean.rblt@gmail.com
[c] e-mail: pschlatt@mech.kth.se
[d] e-mail: azizpour@kth.se
[e] e-mail: rvinuesa@mech.kth.se (corresponding author)

Extending the latter application, in this work we introduce a RL environment to simulate a turbulent flow in a channel. The setup chosen for the simulation is more computationally efficient than other, more realistic geometries (e.g. a wing or a turbine blade), while still being able to capture all the flow features in the vicinity of the wall. The environment is based on the numerical solver SIMSON [15], which implements an efficient pseudo-spectral method to solve the Navier–Stokes equations for incompressible wall-bounded flows. The solver performs direct numerical simulations (DNSs), in which all the time and length scales are resolved without any approximation. This is essential to design a control strategy which does not exploit the limitations introduced by modelling some of the flow scales. Note that such a control policy would underperform if applied on a more realistic flow.

This article is organized as follows: in Sect. 2, we review the most recent applications of deep learning and deep reinforcement learning in fluid mechanics, as well as the different approaches to reduce drag in channel flows from the literature. In Sect. 3, we describe the simulation setup and how the numerical solver interacts with the deep reinforcement learning agent. The learning setup details are also provided. In Sect. 4, we validate our code against the opposition control results available in literature. Furthermore, the learning results in the minimal channel are reported, along with the drag reduction achieved by the DRL policy in two channel flow simulations of different sizes. Finally, in Sect. 5 we summarize our findings and outline some possible future developments.

2 Related work

2.1 Deep learning in fluid mechanics

Fluid-dynamics problems offer many opportunities and challenges for data-driven techniques. Recent research works published in machine-learning venues focused on improving the accuracy of coarse simulations [16,17] or approximating the dynamics of partial differential equations (PDEs) [18,19]. At the same time, the application of deep learning methods to turbulence requires specific domain knowledge, as testified by the large number of research works that have been published by domain specialists. A comprehensive review by Vinuesa and Brunton [20] identifies three main areas of application of deep learning for fluid mechanics: the first possible application is to accelerate direct numerical simulations. A second area includes all the studies in which deep learning is used to enhance turbulence models in simulations, in order to make them less computationally expensive. Finally, neural-network architectures, such as autoencoders (AEs), can be used to develop reduced-order models of fluid flows [21]. On top of these applications, further applications have been envisioned [22], including but not limited to temporal predictions of reduced-order models or spatial reconstructions of turbulent flows.

Reconstruction of turbulent flow fields at a given instant has been attempted using convolutional networks [23]. The possibility to perform *non-intrusive sensing* of the flow, i.e. sampling quantities without disrupting it, is an essential element to implement flow control systems, which typically rely on velocity fields sampled at a given distance from the wall, as detailed in Sect. 2.3. Convolutional neural networks can also be trained to increase the resolution of coarse flow fields [24]. Note, however, that generative adversarial networks (GANs) have been proven to be more effective for this task [25,26]. Furthermore, deep learning models have been trained to reconstruct flow fields from sparse or incomplete measurements. Also in this case, CNNs [27] and GANs [28] have shown very good performance even with very limited information as input.

2.2 Reinforcement learning in fluid mechanics

The application of (deep) reinforcement learning to fluid mechanics is still in its early phase compared with traditional supervised learning [29,30]. Two main categories of contributions can be identified: the first focuses on modelling turbulence with the aid of DRL [13], while the second focuses on active flow control. In the following, we will focus on active flow control applications.

A number of contributions in this domain aim to control the movement of an agent in a fluid flow, for example, representing a fish swimming in a turbulent flow or in a fish school, where the reward aims to maximize the efficiency of the agent swimming [31,32]. Another category of contributions uses DRL to control the dynamics of the flow. The present work falls into this last subcategory. A notable example of such an application is provided by Ref. [14], where the flow around a cylinder is controlled using jets orthogonal to the main flow direction. This case has been used as benchmark in a number of extension works [33–38], a fact that illustrates both the growing interest of DRL for active flow control, and the importance of providing benchmark cases that can be used as a starting point by the community. Recent works [39] highlighted how different control strategies are selected by the DRL agent depending on the physical features of the flow to be controlled, and showed clearly that DRL agents can discover complex strategies not limited to simple opposition control.

Furthermore, DRL has been applied to a number of other control tasks, ranging from simple one-dimensional falling-fluid instabilities [40], convection problems [41], chaotic turbulent combustion systems [42] to a variety of engineering cases [43–46].

In this study, we extend the application of DRL to active flow control in another category of flows, namely wall-bounded turbulent flows. This is a significant jump in complexity for at least two reasons. First, the two-dimensional (2D) cylinder case previously considered exhibits a well-defined shedding pattern at a specific frequency, which is dominant with respect to all the other dynamics, and suppressing it provides a straightforward drag-reduction strategy. If a higher Reynolds number is considered, a different policy is chosen: instead of reducing the shedding, the agent energizes the boundary layer on the cylinder surface to trigger the drag crisis [39]. The channel flow, by contrast, is an inherently multi-scale phenomenon in which there are no obvious frequencies nor mechanisms that can be targeted in order to achieve drag reduction. Second, previous works focused on a 2D environment, while the channel flow is inherently 3D. This is a requisite for the development of true fully featured turbulence structures, and it provides a more nonlinear, chaotic and challenging benchmark to test novel DRL algorithms and control strategies, while also being closer to realistic full-scale configurations. Very recent works have started exploring similar flow cases, for example, in Ref. [47] the use of DRL was tested in a standard channel flow (note that in this study we consider an open channel flow, as detailed in Sect. 3.1). Another notable example is Ref. [48], where a Couette flow is controlled by means of two streamwise parallel slots. Note that in the latter case the training of the DRL agent is performed in a reduced-order model of the problem and then applied to the actual case.

2.3 Drag reduction in channel flows

Given the high importance of flow control in several fields, different techniques and frameworks have been presented in the literature. These control strategies are simple in nature, and exhibit a somewhat limited performance, but they are well established. We discuss these 'traditional' control strategies in the following subsection, and we implement them in the solver to serve as a baseline. In wall-bounded flows (such as our channel flow), the actuation is usually performed by means of a wall-normal velocity distribution applied at the wall, which corresponds to *blowing* when the wall-normal velocity is positive, and *suction* when it is negative. The control law is traditionally obtained by rescaling the wall-normal or spanwise velocity fluctuations at a given wall-normal sampling location y_s, and using it, with opposite sign, as the actuation value. The aim is to suppress the near-wall turbulent structures [49–51]. This strategy provides a drag reduction that depends on the height of the sensing plane, as detailed in Sect. 4.1. Similar drag reduction, at the corresponding Reynolds number, is reported for different flow cases, such as a turbulent boundary layer flow [52].

3 Methodology

3.1 Simulation setup

A summary of our work is shown in Fig. 1. In the left part, we show the numerical domain in which an open channel flow is simulated. The wall is represented as a no-slip condition at the lower boundary, while a symmetry condition is imposed at the upper wall. Previous works [47,49] used a standard channel flow for their investigations, with a no-slip condition applied to both the upper and lower boundaries. We opted for an open channel flow because it allows for a better isolation of the flow features close to a solid boundary. In fact, in the full channel the large turbulent structures can extend beyond the channel centreline, with an effect also on the other wall. The dynamics of the near-wall turbulence close to the lower boundary is not affected by the boundary condition on the other side, providing a meaningful comparison between the two simulations and control techniques.

Throughout the paper, we use (x, y, z) to indicate the streamwise, wall-normal and spanwise directions, respectively, and (u, v, w) are used to indicate the corresponding velocity components. We consider two different simulation domains, the first one is a *minimal channel* [53] with size $\Omega = L_x \times L_y \times L_z = 2.67\,h \times h \times 0.8\,h$ (where h is the open channel height). This domain size is large enough to simulate all the relevant near-wall statistical features of turbulence, while being computationally cheap. Note, however, that the limited spanwise dimension of the channel limits the simulation to a single low-speed streak, whereas in the larger domain, we are able to simulate the interaction of several of them. The second domain size is a larger channel of size $\Omega = L_x \times L_y \times L_z = 2\pi h \times h \times \pi h$. The friction Reynolds number is defined as $Re_\tau = u_\tau h/\nu$, where the friction velocity $u_\tau = \sqrt{\tau_w/\rho}$ (based on the the wall-shear stress τ_w and the fluid density ρ) and ν is the kinematic viscosity of the fluid. The Reynolds number represents the ratio between the inertial and viscous forces in the flow, determining the flow characteristics and how turbulent the flow is. We consider $Re_\tau = 180$ in both domains, to compare the drag reduction with the corresponding results in Ref. [52]. The solver we used, SIMSON, is a pseudo-spectral code that uses the Chebyshev polynomials in the wall-normal direction. The resolution of the simulation is given by $N_x \times N_y \times N_z$, where N_x and N_z are the number of Fourier modes in the streamwise and spanwise directions, while N_y is the number of Chebyshev modes in the wall-normal direction. The minimal channel is simulated with a resolution of $16 \times 65 \times 16$, while the resolution in the larger domain is $64 \times 65 \times 64$. The time-advancement numerical scheme is a second-order Crank–Nicholson algorithm for the linear terms and a third-order Runge–Kutta method for the nonlinear terms.

3.2 Reinforcement learning environment and algorithm

The simulation discussed above needs to be cast as a reinforcement learning problem, defining the actions that can be performed by the agent (shown on the right side of Fig. 1), the state space and the reward function. These quantities are used by the agent to determine the next action and guide the learning, respectively.

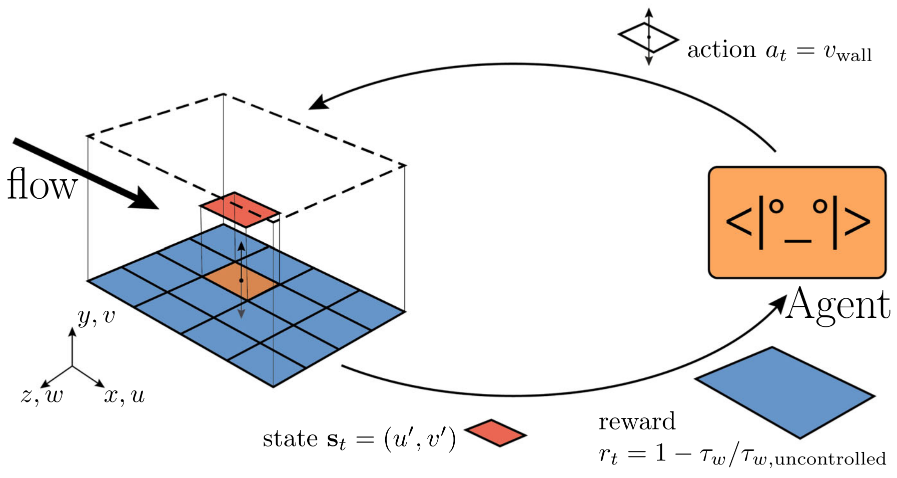
action $a_t = v_{\text{wall}}$
flow
<|°_°|>
Agent
y, v
z, w
x, u
state $\mathbf{s}_t = (u', v')$
reward
$r_t = 1 - \tau_w/\tau_{w,\text{uncontrolled}}$

Fig. 1 Overview of our multi-agent DRL approach to drag reduction. The simulation domain is shown on the left. The agents are organized in a grid $N_{\text{CTRLx}} \times N_{\text{CTRLz}}$. Each agents observes the velocity fluctuations in the streamwise (u') and wall-normal (v') direction. The reward is the percentage variation of the wall-shear stress τ_w. Based on the state, each agent acts by imposing a wall-normal velocity v at the wall

The control is performed by imposing a wall-normal velocity distribution. We cast the simulation as a multi-agent reinforcement learning (MARL) problem, which implies that several independent agents cooperate in order to maximize the chosen reward function. In our setup, all the agents operate locally and thus share the same actuation policy, which determines the action to perform based on the local observation of the environment provided to the agent. This approach has two important consequences: it allows us to avoid the curse of dimensionality on the DRL-control space dimension and to reuse the knowledge of the properties of the flow across the domain, as discussed in Ref. [40]. We consider a grid of $N_{\text{CTRLx}} \times N_{\text{CTRLz}}$ agents that cover the entire lower wall of the channel. Here, N_{CTRLx} and N_{CTRLz} represent the number of agents in the streamwise and spanwise directions, respectively. The policy learnt by the individual agent is translationally invariant to both the streamwise and spanwise directions. Each agent computes the wall-normal velocity intensity to be applied at each evaluation. The control policy is evaluated over a fixed time interval and the actual control varies linearly from the old value to the new one in order to avoid numerical instabilities related to sudden variation of the wall-normal velocity at the boundary. In our numerical experiments, we use $\Delta t^+ = 0.6$ (where the time units are scaled with $t^* = \nu/u_\tau^2$). The actuation value is limited to a prescribed range, in our case the agent can apply wall-normal velocities between $-u_\tau$ and u_τ. It is worth noting that our simulation is performed under the assumption of an incompressible flow. For this reason, even if the individual agents can choose the actuation value independently, we need to enforce a zero-net-mass-flux condition over the entire wall. In previous works (e.g. Ref. [14]), the dimension of the action space is reduced by one, and one of the action values is computed as a function of all the others. In our case, this could have led to a control value disproportionately large; hence, we limit ourselves to enforce a zero-average control over all the agents. This is done by removing the mean over all actuators from the control signals before they are applied in the simulation at each timestep.

The agent does not have access to the entire velocity field, and the observed state consists of a portion of one or more wall-parallel planes of sampled flow quantities. The portion of the flow field that is observable by each agent is the one above the actuation. It is possible to sample any of the three velocity components at a given wall-normal location, the wall-normal location of the sampled plane is defined in the input file of the simulation and it is rounded up to the closest Chebyshev colocation node. For the remainder of the paper, we will refer to the different sampling heights using *inner-scaled* wall-normal coordinates: $y^+ = yu_\tau/\nu = y/\ell^* \in [0, Re_\tau]$, where ℓ^* denotes the viscous length.

The same global reward value is provided to all MARL agents, defined as the percentual reduction of the wall-shear stress with respect to the uncontrolled flow, averaged over the entire wall. We do not consider the instantaneous value of this quantity, rather an averaged value between two actions is returned. The solver performs multiple timestep advancements between one action and the next one because the timestep required for the stability of the numerical method is much lower than the actuation time. The actuation interval is also a parameter that can be defined in the input file and it should be set to be small enough to allow it to react to the change of the small flow scales, and sufficiently large to observe the effect of the action at the intermediate scales [40]. Since turbulence is a multi-frequency phenomenon, the reward signal reflects this characteristic and the reward averaging helps to provide a more reliable estimate of the actual wall-shear-stress reduction than an instantaneous value would do. Note that the presence of turbulence in the flow induces a higher drag with respect to a laminar flow and the implemented control effectively reduces the level of turbulence. In fact, the control can theoretically induce a relaminarization of the flow. This condition represents an upper bound for the drag reduction achievable with the control [54]. This value depends on the chosen Reynolds number and in our case is 73.9%. Note, however, that there is no guarantee that this value is ever attainable with any control law.

The deep reinforcement learning algorithm that is used to train the agent is the deep deterministic policy gradient (DDPG) [55]. As the name suggests, this algorithm optimizes a deep neural network (called *actor*) that approximates the relation between the state and the action to be performed (i.e. a policy function). After learning, the policy is deterministic. Additionally, a second neural network (called *critic*) is used to approximate the action-value function $Q(s, a)$. In our work, we used two simple fully connected networks for both the actor and the critic. The actor has a single layer of 8 nodes, while the critic has three layers with 16, 64 and 64 nodes. During learning, Gaussian noise with zero mean and variance $0.1u_\tau$ is added to the actions computed by actor network before they are provided to the critic network. This action perturbation is performed in order to foster exploration. The policy is updated every $\Delta t^+ = 180$, with 64 mini-batch gradient updates. The mini-batches are sampled from a buffer replay that includes 5,000,000 (state, action, reward) tuples. All the other algorithm parameters have been used with the default values provided by the Stable-baselines [56] implementation. All the configuration details can also be retrieved in the code repository associated with this paper.

3.3 Solver-DRL interface

The reinforcement learning framework chosen for this work is Stable-baselines [56], while the environment is coded as a custom PettingZoo/Gym environment [57]. Both Stable-baselines and the Gym environment are written in Python; however, the fluid solver is coded in FORTRAN 77/90. An interface between the two programming languages is then needed in order to communicate the quantities (state, control values, reward) that are necessary for the learning to take place. While previous studies have coupled the solver and the RL algorithm using an input/output (I/O) stream [58], in this case the interface is based on message-passing interface (MPI). The fluid solver is spawned as a child process of the main learning process and an intercom is created between the solver and the Python main program. After the initialization, the solver waits for information requests from the main program. Once the request is received by the solver, further MPI-based messages are exchanged between the agent and the solver, depending on the requested information. Communication requests are handled using five-character strings, which determine the sequence of instructions that the solver has to perform for each interaction of the agent with the environment. The admissible values for the request string are:

- `STATE`: this string is used to request the simulation to communicate the state to the agent.
- `CNTRL`: after this request, the agent communicates the new action to the environment, in particular the new values of the controllable parameters are passed to the fluid solver.
- `EVOLVE`: once the control parameters are updated with `CNTRL`, several time iterations of the solver are computed in order to observe the effect of the chosen action on the environment. The instantaneous wall-shear stress is passed to the Python program and it is used to compute the reward.
- `TERMN`: this request interrupts the solver and closes the FORTRAN program. This request is used at the end of each episode before restarting the environment. The initial condition is the same for every episode unless stated otherwise.

The combination of these messages is used to define the required functions for the Gym environment application programming interface (API).

4 Experiments

4.1 Validation and baseline

As mentioned in Sect. 2, opposition control is a simple established control benchmark that is well understood theoretically and can be used as a reference to assess DRL-control laws. Given the velocity distribution at the sensing plane y_s, the actuation at the wall v_{wall} applied by opposition control can be computed with the expression:

$$v_{\text{wall}}(x, z, t) = -\alpha\big[v(x, y_s, z, t) - \langle v(x, y_s, z, t)\rangle\big]. \quad (1)$$

Here, $\langle v(x, y_s, z, t)\rangle$ is the spatial mean of the wall-normal velocity field, and the second term of the equation enforces zero-net-mass-flux within the domain, so that the actuation plane as a whole is not adding or removing fluid mass to the flow. α is a positive scaling parameter which is fixed in the spatial directions and time and its typical value is $\alpha = 1$, independently from the height of the sampled velocity plane. In order to validate our code implementation, we reproduced this control strategy in our solver, letting the environment evolve with constant α for a sufficiently long time, such that the drag-reduction rate has reached a stationary value. In Ref. [49], the highest drag reduction was found by sampling the wall-normal velocity at $y^+ = 10$. The computational domain had size $\Omega = L_x \times L_y \times L_z = 4\pi h \times h \times 4/3\pi h$, with resolution $N_x \times N_y \times N_z = 128 \times 129 \times 128$. The friction Reynolds number Re_τ was reported to be 180 and the Reynolds number based on the centreline velocity U_{cl} (the one at the top boundary, in the case of the open channel) is in close agreement with our setup: the reported value was $Re_{cl} = 3300$, while our simulation is performed with $Re_{cl} = 3273$. The drag reduction reported in Ref. [49] was $\approx 14\%$ when sampling at $y^+ = 10$. Our result in the larger channel is 17.73%. This is in acceptable agreement with that of the literature and further validates our simulation, considering the lower resolution and taking into account the small difference in spanwise size of the domain.

A different sampling plane and slightly lower Re_{cl} were used in Ref. [50]. The domain here is $\Omega = L_x \times L_y \times L_z = 4\pi h \times h \times 2\pi h$, with resolution $N_x \times N_y \times N_z = 256 \times 130 \times 256$. When rescaling the velocity at $y^+ = 15$ with $Re_{cl} = 3240$, the resulting drag reduction was $\approx 25\%$ and our own test resulted in a reduction of 25.67%, which is in very good agreement. Finally, in Ref. [52], the field at $y^+ = 15$ at $Re_\tau = 180$ is sampled. In this case, the resolution is slightly lower than those of the previous studies the wall-parallel directions ($N_x \times N_y \times N_z = 160 \times 257 \times 128$). On the other hand, the channel size is also lower, being exactly the same as our larger channel. They reported $\approx 24\%$ drag reduction, and our result is also in good agreement with theirs. Overall, the results from our implementation match the ones reported in the literature, giving us confidence in our simulation setup and control implementation. The drag reduction obtained with these settings is used as baseline for comparison with the DRL results.

4.2 Minimal channel learning and testing

We first consider the smaller domain, i.e. the minimal channel configuration. In this case, the number of agents is set to $N_{\text{CTRLx}} = N_x$ and $N_{\text{CTRLz}} = N_z$. The learning is performed using several initial conditions for 400 episodes. The streamwise and wall-normal components of the velocity at $y^+ = 15$ are provided to the agents as the observable state. In each learning, all the episodes are initialized with the same flow field. Figure 2 highlights the sensitivity of the learning to the initial condition of the episode. One important remark is that (D)RL algorithms are based on a trial-and-error approach to the reward optimization. This means that not all learning runs provide a drag-reducing policy, as shown in the aforementioned figure. Note that the unsuccessful runs (shown in the left panel) are removed from the average for clarity.

Even though the learning has a strong sensitivity on the initialization of the episode, the learnt policies perform consistently during testing, regardless of the selected initial condition. Figure 3 shows the drag reduction achieved with one of the best-performing policy. The policy test is repeated for six different initial conditions, in order to assess the generalization of the learnt control policy. The performance is compared with the baseline strategy (opposition control, with sensing plane at $y^+ = 15$) for the same set of initial conditions. Initially, the DRL policy produces a strong increase in the drag for a brief time, corresponding to a negative value for the drag reduction. After this drag increase, it is possible to observe how the average drag reduction after the initial transient is consistently higher than the one obtained with opposition control: the DDPG policy provides 43% drag reduction, while opposition control is limited to 26%, as mentioned in Sect. 4.1.

Figure 3 (right) shows the effect of the control on the velocity-fluctuations distribution. Using the friction velocity for the uncontrolled case, it is possible to observe how the two control approaches produce different changes in the fluctuations distribution. With opposition control, the range of the fluctuations is reduced, but the overall shape of the distribution is unchanged. In fact, from the perspective of quadrant analysis [59,60], sweeps and ejections remain the dominant features in the near-wall region. On the other hand, DRL has a more significant effect on the distribution: at the sensing plane, we can observe a distribution that is almost symmetric with respect to the wall-normal fluctuations. With the DRL control, the range of the streamwise fluctuations is greatly reduced, while the wall-normal fluctuations are increased. Furthermore, the predominance of sweeps and ejections vanishes, with a more even distribution of events among the four quadrants. Consequently, DRL learns a control strategy with a profound impact on the flow physics. Analysing the sensing plane at $y^+ = 15$ when the control is applied reveals that the wall streak in the minimal channel is significantly attenuated by a streamwise travelling wave.

4.3 Larger channel testing

One of the relevant features of the learnt policy is that it is local and translational invariant, meaning that it can be applied with no modification to different domain sizes and flow cases, provided that the size of the state observations and of the reward are the same. The most important assumption behind the application of the same policy is that the underlying physical features exploited by the agent in the environment where the learning is performed are also present in the new environment. When applying the policy to a larger domain, we still considered $N_{\text{CTRLx}} = N_x$ and $N_{\text{CTRLz}} = N_z$. Note, however, that this implies that a larger number of agents cooperate within a single instance of the simulation. The left panel of Fig. 4 shows the drag reduction achieved in the larger domain when the policy learnt in the minimal channel is applied. Remarkably, the policy provides a drag reduction that is higher than the opposition control baseline, regardless of the flow initial condition. The effectiveness of the control policy is tested without any further tuning in the larger domain. The success of this transfer application shows that the DRL agent has been able to learn a control strategy that robustly exploits the features in the flow, so that the learnt policy can be effective in different cases which have a similar physical characterization. The magnitude of the reduction in this case is smaller than the one obtained in the minimal channel, with the DDPG policy providing 30% drag reduction, still performing better than opposition control, which yields 20%. One possible explanation for this result is the fact that the policy is learnt in the minimal channel, where some physical features, such as the interaction of the streaks, cannot be experienced by the agent. Further evidence of this difference can be found by assessing the effect of the two control strategies on the velocity-fluctuations

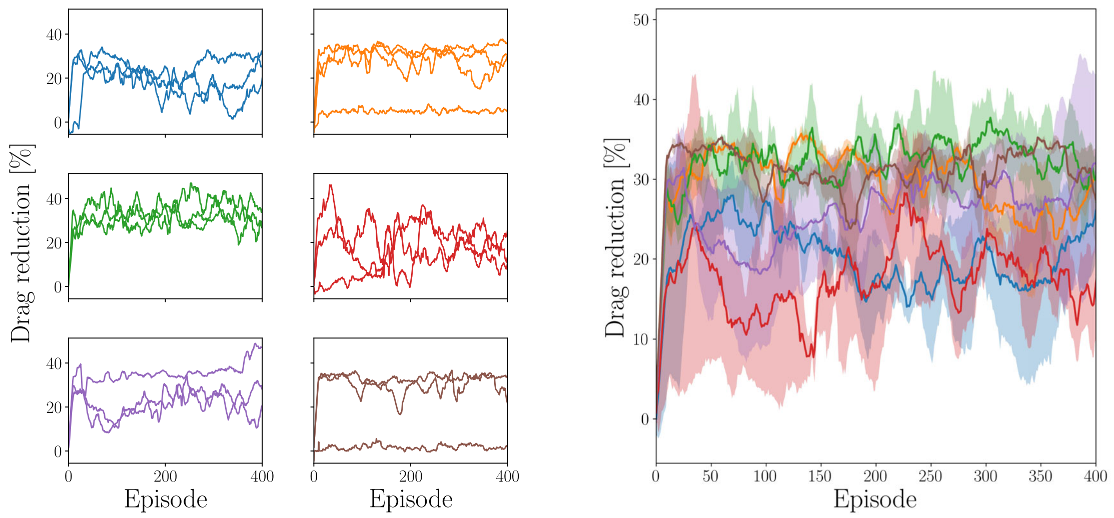

Fig. 2 (Left) Running mean of the drag reduction with respect to the reference uncontrolled case during agent learning in the minimal channel. Each panel shows 3 different learning runs for 6 initial conditions. (Right) Running mean of the drag reduction with respect to the reference uncontrolled case during learning in the minimal channel, averaged over different learning runs. The shaded area represents the variance of the drag reduction with the different runs

distribution shown in the right panel of Fig. 4. For the DRL control, the symmetry with respect to the wall-normal fluctuations is more pronounced than before, showing a high probability of strong wall-normal fluctuations, coupled with weak streamwise fluctuations. In this case, the predominance of sweeps and ejections is also reduced by the DRL in the larger domain, but it is not eliminated to the extent that is observed in the minimal channel.

The actuation applied at the wall by opposition control and the DRL policy are very different, as shown in Fig. 5. The DRL policy takes advantage of the available control range. The control is often switching from the maximum to the minimum value (or vice versa), in a way that reminds *bang-bang* (or *2-step*) control strategies [61]. On the other hand, the actuation in opposition control is limited by the fluctuations in the wall-normal direction at the sensing plane, that the control aims to suppress. With this approach, the actuation intensity decreases as the fluctuations are reduced by the control. In turn, by reducing the control intensity, the wall-normal fluctuations start growing again (as seen at $t^+ \approx 400$), affecting again the control in a cyclic increase and decrease of the fluctuations.

Despite yielding a smaller drag reduction, opposition control also has a smaller actuation intensity. This means that it could potentially yield a better performance if the actuation cost is factored in. To this end, we followed the approach of Ref. [62], where the input power is calculated as:

$$w_{\text{in}} = \frac{1}{2}\|v_{\text{wall}}\|^3, \tag{2}$$

and the net-energy saving is computed as

$$S = \frac{c_{f,\text{uncontrolled}} - (c_f + w_{\text{in}})}{c_{f,\text{uncontrolled}}}. \tag{3}$$

Here $c_f = 2\tau_w/\rho U_b^2$ is the friction coefficient (and U_b is the bulk velocity in the open channel, which is constant because we are simulating a flow with fixed mass-flow rate). Both c_f and w_{in} are averaged over all the agents and in time after the initial transient ($t^+ > 500$). Note that, by definition, the ratio of the friction coefficient in the controlled and uncontrolled case is equivalent to the ratio of the wall-shear stress. Hence, the drag reduction R in Ref. [62] is equivalent to our reward:

$$R = 1 - c_f/c_{f,\text{uncontrolled}} = 1 - \tau_w/\tau_{w,\text{uncontrolled}}. \tag{4}$$

The order of magnitude of the control is $\mathcal{O}(10^{-2})$, as shown in Fig. 5. This means that the input power will be $w_{\text{in}} \sim \mathcal{O}(10^{-6})$. On the other hand, the friction coefficient is $c_f \sim \mathcal{O}(10^{-3})$. From this initial estimate based on the order of magnitude of the different terms, the input power is comparatively small with respect to the controlled c_f, similarly to Ref. [62]. Hence, the net-energy saving obtained with the two control strategies is very similar to the drag-reduction figures reported before. In the DRL case, the net-energy saving is around 29% (the drag reduction per se is 30%), while in the opposition control case both drag reduction and net-energy saving are about 20%, and these are essentially equal as they differ by only around 0.1%. Even when the cost of actuation is taken into account, the DRL

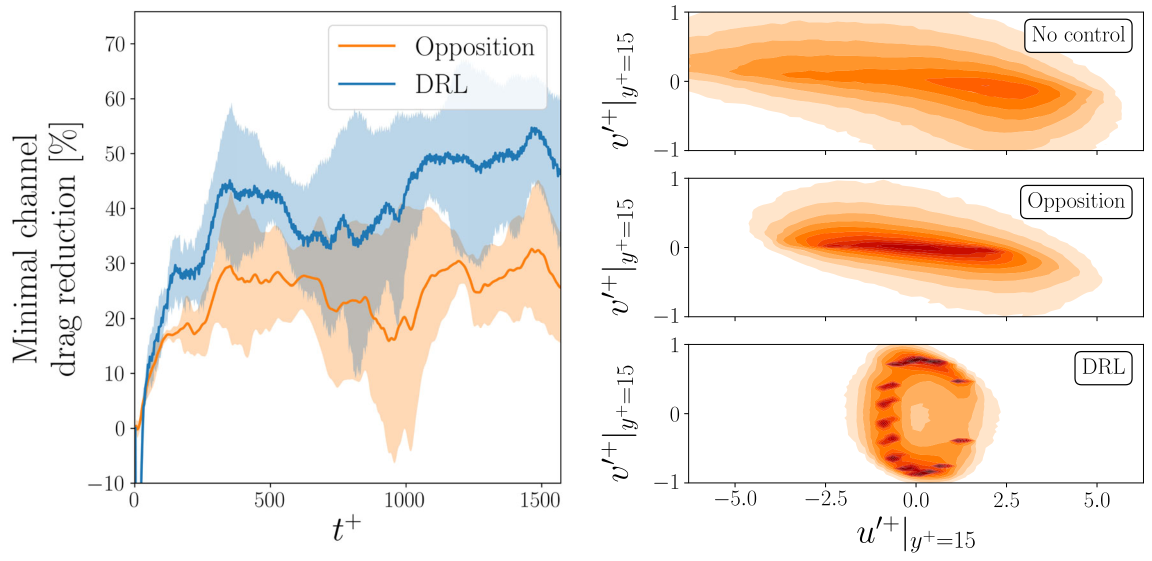

Fig. 3 (Left) Drag reduction with respect to the uncontrolled case obtained in the minimal channel, when using the policy learnt using DRL or opposition control. The result is averaged over 6 different initial conditions. The shaded area represents the variance of the drag reduction with the different initial conditions. (Right) Distribution of the inner-scaled velocity-fluctuation components after the initial transient ($t^+ > 500$) in the streamwise (u) and wall-normal (v) directions, for the uncontrolled case (top), with opposition control (middle) and when using DRL (bottom)

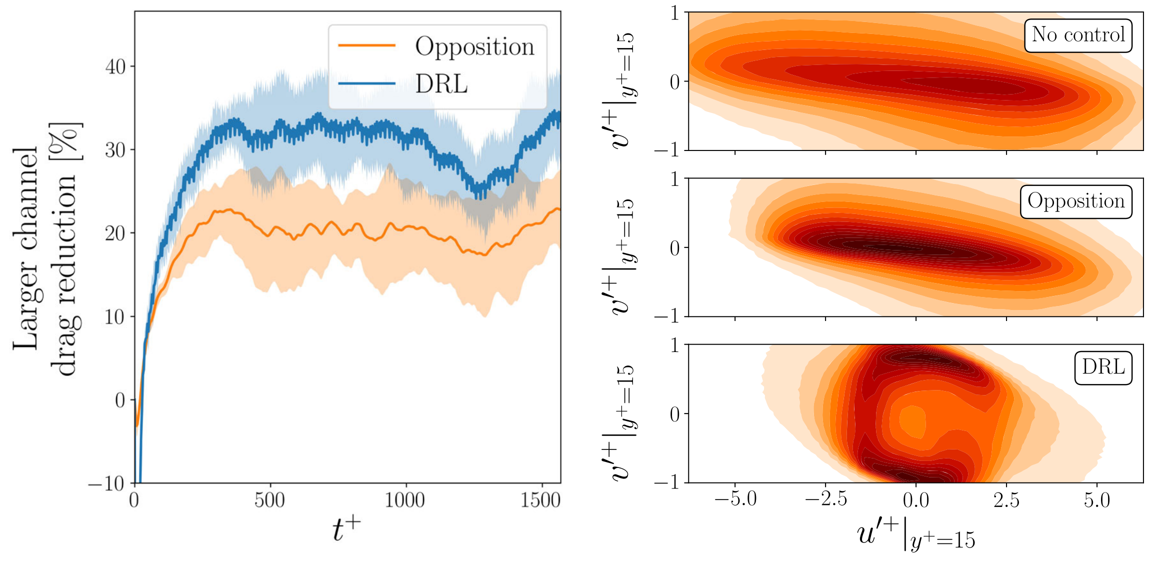

Fig. 4 (Left) Drag reduction with respect to the uncontrolled case obtained in the larger channel, when using the DRL policy learnt in the minimal channel versus using opposition control. The result is averaged over 6 different initial conditions. The shaded area represents the variance of the drag reduction with the different initial conditions. (Right) Distribution of the inner-scaled velocity-fluctuation components after the initial transient ($t^+ > 500$) in the streamwise (u) and wall-normal (v) directions, for the uncontrolled case (top), with opposition control (middle) and when using DRL (bottom)

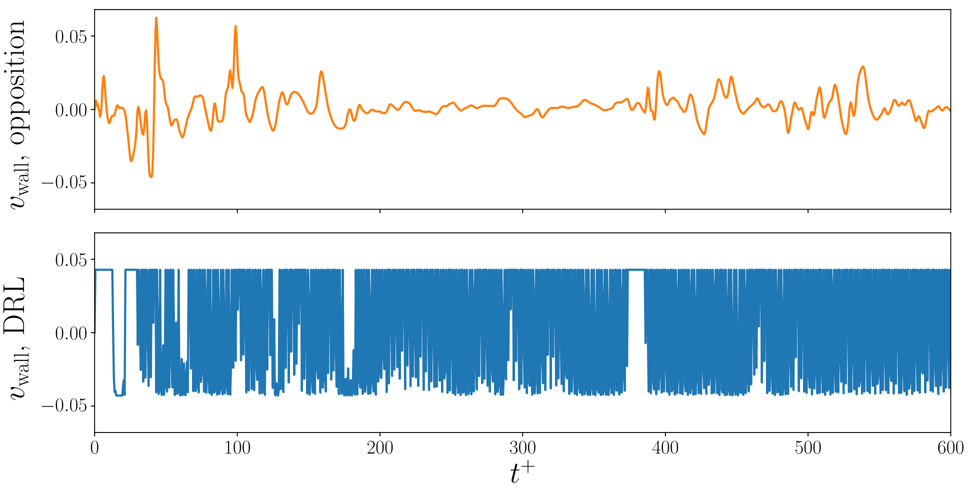

Fig. 5 History of the control applied at the wall by a single agent (up to $t^+ = 600$) using opposition control (top) and the DRL policy (bottom) in the larger channel, when using the policy trained in the minimal channel

policy still provides a better performance than opposition control.

5 Conclusions

In this paper, we have described the implementation of a pseudo-spectral solver for DNS of fluid flows as a reinforcement learning environment, with which the agents interact to maximize the drag reduction within the domain. The environment supports three-dimensional, fully turbulent flow simulations, allowing for the discovery of physically accurate control policies. The environment can be customized with different flow cases and adapted to tasks of varying difficulty, selecting a different number of input quantities or a different number of agents. For instance, acting on the vorticity can be an effective strategy to reduce small-scale turbulent fluctuations [63]. This variable could be included in the state observed by the DRL agents, in order to assess its effect on the learnt policy.

Using the environment, we applied DRL for drag reduction in an open channel flow, with two different domain sizes and resolutions. The DDPG algorithm allows the identification of control policies that on average show a higher drag reduction than the one provided by opposition control, used here as baseline. In the minimal channel, the control strategy yields 43% drag reduction, while in the larger domain the achieved reduction is smaller (30%), but still consistently better than the baseline. The performance improvement with respect to the baseline is around 20 percentage points in the minimal channel and 10 percentage points in the larger domain. None of the policies learnt or used as baseline considers the energetic cost of the actuation, and the learnt DRL policy uses a higher amount of energy, compared to opposition control. Note, however, that this is not affecting the overall performance difference between the two strategies, as the actuation energy is much smaller compared with the additional drag reduction. Nonetheless, reinforcement learning could in further work help designing more efficient control strategies by incorporating explicitly the energy cost of the actuation as part of the reward function. Currently, the learnt policies are not able to relaminarize the flow. Since this condition represents the upper bound for drag reduction, an improvement of the control policy is still theoretically possible, although there is no guarantee that this is attainable through DRL control or other techniques. The current study focuses on a open channel flow with uncontrolled friction Reynolds number $Re_\tau = 180$. Our setup can also be adapted to different flow cases, paving the way to applying similar techniques to increasingly complex systems such as boundary layers or higher Reynolds numbers. Simulating increasingly high Reynolds numbers becomes progressively more computationally expensive; however, the simulation wall-clock time can be reduced by using the MPI parallelization of the solver and HPC clusters. The use of different RL techniques, such as offline RL [64], could also help reducing the overall cost of the learning. The possibility to change the Reynolds number of the flow simulation represents a way to 'tune' the difficulty of the control problem, making the drag reduction in a open channel flow an ideal benchmark to test new policies. Increasing the Reynolds number is necessary step in order to design a control strategy that can be applied in practical applications; on the other hand, it also represents an appealing research direction as the maximum achievable drag reduction increases with the Reynolds number. It must be noted that a control policy that yields drag reduction at a given Reynolds number, may not be as effective at a higher Re because different physical mechanisms can be responsible for the drag [65]. The policy learnt at a given Reynolds number will likely require some form of adjustment in order to perform in a satisfactory manner at a different Reynolds. In this regard, deep reinforcement learning constitutes a promising framework, thanks to its end-to-end approach: all the physical features of the turbulent flow are represented in the environment, allowing for new and possibly more effective control strategies.

Acknowledgements The authors acknowledge the Swedish National Infrastructure for Computing (SNIC) for providing the computational resources by PDC, used to carry out the numerical simulations. The authors acknowledge Prof. Yosuke Hasegawa for insightful discussions on the DRL-algorithm setup.

Funding Open access funding provided by Royal Institute of Technology. This work is supported by the founding provided by the Swedish e-Science Research Centre (SeRC), ERC grant no. '2021-CoG-101043998, DEEPCONTROL' and the Knut and Alice Wallenberg (KAW) Foundation.

Author contribution statement

The general idea of the project was developed by Ricardo Vinuesa (RV). The deep reinforcement learning (DRL) interface for the flow simulation was developed by Luca Guastoni (LG), with input by RV, Philipp Schlatter (PS) and Jean Rabault (JR). The learning runs and post-processing of the results were performed by LG. JR, RV and Hossein Azizpour (HA) provided feedback. The paper was written by LG and edited by RV and JR. Comments were provided by RV, JR, HA and PS.

Declarations

Code availability The datasets generated during the current study as well as all the codes are included in the repository https://github.com/KTH-FlowAI/MARL-drag-reduction-in-wall-bounded-flows.

Conflict of interest The authors report no conflicts of interest

A Effect of the episode length

In Sect. 4.2, we consider episodes of 3000 interactions between the environment and the agent. This corresponds to a simulation time of $t^+ \approx 1500$. Here, we verify the effect of the episode length on the training by reducing the number of interactions to 1000 and 2000, as shown in Fig. 6. The initial

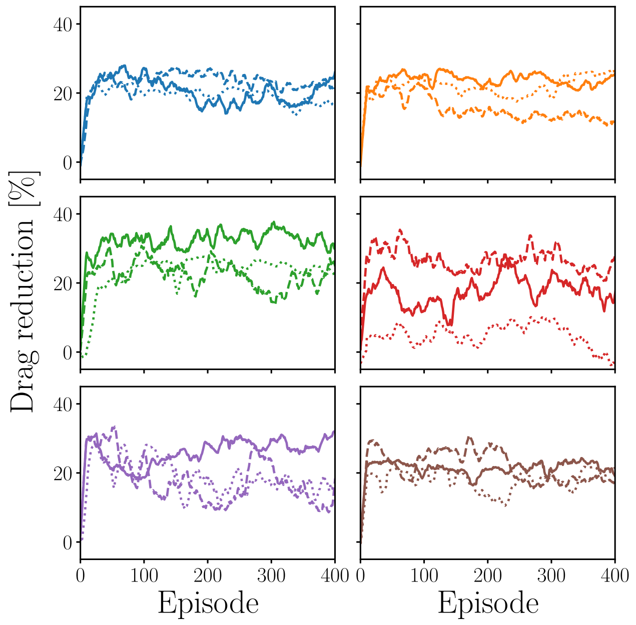

Fig. 6 Running mean of the drag reduction with respect to the reference uncontrolled case during learning in the minimal channel, averaged over different learning runs. Each panel shows 3 different learning runs for 6 initial conditions

choice of the episode length is designed to include the initial transient (typically $t^+ \approx 500$) and a sufficiently long time after that. However, in our experiments, longer episodes do not provide a significant advantage in terms of drag reduction. It is possible to observe how the learning appears to improve faster at the very beginning of the learning but this is simply related to the higher number of interactions per episode. Shorter episode might provide a way to reduce the overall time and computational cost of the learning, which increase quickly with the number of agents and the resolution/Reynolds number, respectively.

References

1. J.R. Garratt, The atmospheric boundary layer. Earth Sci. Rev. **37**(1–2), 89–134 (1994)
2. M.J. Churchfield, S. Lee, J. Michalakes, P.J. Moriarty, A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics. J. Turbulence **13**, 14 (2012)
3. P.D. Stein, H.N. Sabbah, Turbulent blood flow in the ascending aorta of humans with normal and diseased aortic valves. Circ. Res. **39**(1), 58–65 (1976)
4. S.P. Schneider, Hypersonic laminar-turbulent transition on circular cones and scramjet forebodies. Prog. Aerosp. Sci. **40**(1–2), 1–50 (2004)
5. I. Celik, I. Yavuz, A. Smirnov, Large eddy simulations of in-cylinder turbulence for internal combustion engines: a review. Int. J. Engine Res. **2**(2), 119–148 (2001)
6. P.R. Spalart, J.D. McLean, Drag reduction: enticing turbulence, and then an industry. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **369**(1940), 1556–1569 (2011)

7. S.L. Brunton, B.R. Noack, Closed-loop turbulence control: progress and challenges. Appl. Mech. Rev. (2015). https://doi.org/10.1115/1.4031175
8. T. Duriez, S.L. Brunton, B.R. Noack, *Machine Learning Control - Taming Nonlinear Dynamics and Turbulence* (Springer, Cham, 2017)
9. F. Pino, L. Schena, J. Rabault, M.A. Mendez, Comparative analysis of machine learning methods for active flow control (2022). https://doi.org/10.48550/ARXIV.2202.11664
10. J. Nousiainen, C. Rajani, M. Kasper, T. Helin, Adaptive optics control using model-based reinforcement learning. Opt. Express **29**(10), 15327–15344 (2021). https://doi.org/10.1364/OE.420270
11. 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 **602**(7897), 414–419 (2022)
12. C. Beeler, U. Yahorau, R. Coles, K. Mills, S. Whitelam, I. Tamblyn, Optimizing thermodynamic trajectories using evolutionary and gradient-based reinforcement learning. Phys. Rev. E **104**, 064128 (2021). https://doi.org/10.1103/PhysRevE.104.064128
13. G. Novati, H.L. de Laroussilhe, P. Koumoutsakos, Automating turbulence modeling by multi-agent reinforcement learning. Nature Mach. Intell. **3**, 87–96 (2021)
14. J. Rabault, M. Kuchta, A. Jensen, U. Réglade, N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. J. Fluid Mech. **865**, 281–302 (2019)
15. M. Chevalier, P. Schlatter, A. Lundbladh, D.S. Henningson, A pseudospectral solver for incompressible boundary layer flows. Technical report, TRITA-MEK 2007:07. KTH Mechanics, Stockholm, Sweden (2007)
16. D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning-accelerated computational fluid dynamics. Proc. Natl. Acad. Sci. **118**(21) (2021)
17. M. Zhang, J. Wang, J. Tlhomole, M.D. Piggott, Learning to estimate and refine fluid motion with physical dynamics (2022)
18. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations (2020)
19. N. Wandel, M. Weinmann, R. Klein, Unsupervised deep learning of incompressible fluid dynamics (2020). arXiv:2006.08762
20. R. Vinuesa, S. Brunton, Enhancing computational fluid dynamics with machine learning. Nature Comput. Sci. **2**, 358–366 (2022)
21. H. Eivazi, S. Le Clainche, S. Hoyas, R. Vinuesa, Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows. Expert Syst. Appl. **202**, 117038 (2022)
22. R. Vinuesa, S. Brunton, Emerging trends in machine learning for computational fluid dynamics (2022)
23. L. Guastoni, A. Güemes, A. Ianiro, S. Discetti, P. Schlatter, H. Azizpour, R. Vinuesa, Convolutional-network models to predict wall-bounded turbulence from wall quantities. J. Fluid Mech. **928**, 27 (2021)
24. K. Fukami, K. Fukagata, K. Taira, Super-resolution reconstruction of turbulent flows with machine learning. J. Fluid Mech. **870**, 106–120 (2019). https://doi.org/10.1017/jfm.2019.238
25. A. Güemes, S. Discetti, A. Ianiro, B. Sirmacek, H. Azizpour, R. Vinuesa, From coarse wall measurements to turbulent velocity fields through deep learning. Phys. Fluids **33**(7), 075121 (2021). https://doi.org/10.1063/5.0058346
26. H. Kim, J. Kim, S. Won, C. Lee, Unsupervised deep learning for super-resolution reconstruction of turbulence. J. Fluid Mech. **910**, 29 (2021). https://doi.org/10.1017/jfm.2020.1028
27. K. Fukami, R. Maulik, N. Ramachandra, K. Fukagata, K. Taira, Global field reconstruction from sparse sensors with voronoi tessellation-assisted deep learning. Nature Mach. Intell. **3**(11), 945–951 (2021). https://doi.org/10.1038/s42256-021-00402-2
28. M. Buzzicotti, F. Bonaccorso, P.C. Di Leoni, L. Biferale, Reconstruction of turbulent data with deep generative models for semantic inpainting from turb-rot database. Phys. Rev. Fluids **6**, 050503 (2021). https://doi.org/10.1103/PhysRevFluids.6.050503
29. P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, E. Hachem, A review on deep reinforcement learning for fluid mechanics. Comput. Fluids **225**, 104973 (2021). https://doi.org/10.1016/j.compfluid.2021.104973
30. J. Rabault, F. Ren, W. Zhang, H. Tang, H. Xu, Deep reinforcement learning in fluid mechanics: a promising method for both active flow control and shape optimization. J. Hydrodyn. **32**(2), 234–246 (2020)
31. L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, K. Gustavsson, Zermelo's problem: optimal point-to-point navigation in 2d turbulent flows using reinforcement learning. Chaos: Interdiscip. J. Nonlinear Sci. **29**(10), 103138 (2019). https://doi.org/10.1063/1.5120370
32. S. Verma, G. Novati, P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning. Proc. Natl. Acad. Sci. **115**(23), 5849–5854 (2018). https://doi.org/10.1073/pnas.1800923115
33. J. Rabault, A. Kuhnle, Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach. Phys. Fluids **31**(9), 094105 (2019)
34. R. Paris, S. Beneddine, J. Dandois, Robust flow control and optimal sensor placement using deep reinforcement learning. J. Fluid Mech. **913** (2021)
35. H. Tang, J. Rabault, A. Kuhnle, Y. Wang, T. Wang, Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning. Phys. Fluids **32**(5), 053605 (2020)
36. D. Fan, L. Yang, Z. Wang, M.S. Triantafyllou, G.E. Karniadakis, Reinforcement learning for bluff body active flow control in experiments and simulations. Proc. Natl. Acad. Sci. **117**(42), 26091–26098 (2020)
37. F. Ren, J. Rabault, H. Tang, Applying deep reinforcement learning to active flow control in weakly turbulent conditions. Phys. Fluids **33**(3), 037121 (2021)
38. H. Xu, W. Zhang, J. Deng, J. Rabault, Active flow control with rotating cylinders by an artificial neural network trained by deep reinforcement learning. J. Hydrodyn. **32**(2), 254–258 (2020)

39. P. Varela, P. Suárez, F. Alcántara-Ávila, A. Miró, J. Rabault, B. Font, L.M. Garcáa-Cuevas, O. Lehmkuhl, R. Vinuesa, Deep reinforcement learning for flow control exploits different physics for increasing Reynolds number regimes. Actuators **11**(12) (2022). https://doi.org/10.3390/act11120359
40. V. Belus, J. Rabault, J. Viquerat, Z. Che, E. Hachem, U. Reglade, Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film. AIP Adv. **9**(12), 125014 (2019)
41. G. Beintema, A. Corbetta, L. Biferale, F. Toschi, Controlling Rayleigh–Bénard convection via reinforcement learning. J. Turbul. **21**(9–10), 585–605 (2020)
42. M.A. Bucci, O. Semeraro, A. Allauzen, G. Wisniewski, L. Cordier, L. Mathelin, Control of chaotic systems by deep reinforcement learning. Proc. R. Soc. A **475**(2231), 20190351 (2019)
43. M.T. Henry de Frahan, N.T. Wimer, S. Yellapantula, R.W. Grout, Deep reinforcement learning for dynamic control of fuel injection timing in multipulse compression ignition engines. Int. J. Engine Res., 14680874211019345 (2021)
44. H. Korb, H. Asmuth, M. Stender, S. Ivanell, Exploring the application of reinforcement learning to wind farm control. J. Phys: Conf. Ser. **1934**(1), 012022 (2021). https://doi.org/10.1088/1742-6596/1934/1/012022
45. C. Zheng, T. Ji, F. Xie, X. Zhang, H. Zheng, Y. Zheng, From active learning to deep reinforcement learning: Intelligent active flow control in suppressing vortex-induced vibration. Phys. Fluids **33**(6), 063607 (2021). https://doi.org/10.1063/5.0052524
46. R. Vinuesa, O. Lehmkuhl, A. Lozano-Durán, J. Rabault, Flow control in wings and discovery of novel approaches via deep reinforcement learning. Fluids (2022). https://doi.org/10.3390/fluids7020062
47. T. Sonoda, Z. Liu, T. Itoh, Y. Hasegawa, Reinforcement learning of control strategies for reducing skin friction drag in a fully developed channel flow. arXiv:2206.15355 (2022)
48. K. Zeng, A.J. Linot, M.D. Graham, Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning. Proc. R. Soc. A: Math. Phys. Eng. Sci. **478**(2267), 20220297 (2022). https://doi.org/10.1098/rspa.2022.0297
49. H. Choi, P. Moin, J. Kim, Active turbulence control for drag reduction in wall-bounded flows. J. Fluid Mech. **262**, 75–110 (1994)
50. E.P. Hammond, T.R. Bewley, P. Moin, Observed mechanisms for turbulence attenuation and enhancement in opposition-controlled wall-bounded flows. Phys. Fluids **10**(9), 2421–2423 (1998). https://doi.org/10.1063/1.869759
51. Y. Chang, S.S. Collis, S. Ramakrishnan, Viscous effects in control of near-wall turbulence. Phys. Fluids **14**(11), 4069–4080 (2002). https://doi.org/10.1063/1.1509751
52. A. Stroh, B. Frohnapfel, P. Schlatter, Y. Hasegawa, A comparison of opposition control in turbulent boundary layer and turbulent channel flow. Phys. Fluids **27**(7), 075101 (2015). https://doi.org/10.1063/1.4923234
53. J. Jiménez, P. Moin, The minimal flow unit in near-wall turbulence. J. Fluid Mech. **225**, 213–240 (1991). https://doi.org/10.1017/S0022112091002033
54. K. Fukagata, K. Sugiyama, N. Kasagi, On the lower bound of net driving power in controlled duct flows. Physica D **238**(13), 1082–1086 (2009). https://doi.org/10.1016/j.physd.2009.03.008
55. T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning (2015)
56. A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-baselines3: Reliable reinforcement learning implementations. J. Mach. Learn. Res. **22**(268), 1–8 (2021)
57. J.K. Terry, B. Black, A. Hari, L.S. Santos, C. Dieffendahl, N.L. Williams, Y. Lokesh, C. Horsch, P. Ravi, Pettingzoo: Gym for multi-agent reinforcement learning (2020). arXiv:2009.14471
58. Q. Wang, L. Yan, G. Hu, C. Li, Y. Xiao, H. Xiong, J. Rabault, B.R. Noack, DRLinFluids: an open-source Python platform of coupling deep reinforcement learning and OpenFOAM. Phys. Fluids **34**(8), 081801 (2022). https://doi.org/10.1063/5.0103113
59. J.M. Wallace, H. Eckelmann, R.S. Brodkey, The wall region in turbulent shear flow. J. Fluid Mech. **54**(1), 39–48 (1972). https://doi.org/10.1017/S0022112072000515
60. S.S. Lu, W.W. Willmarth, Measurements of the structure of the Reynolds stress in a turbulent boundary layer. J. Fluid Mech. **60**(3), 481–511 (1973). https://doi.org/10.1017/S0022112073000315
61. I. Flugge-Lotz, *Discontinuous Automatic Control*, vol. 2166 (Princeton University Press, Princeton, NJ, 2015)
62. Y. Kametani, K. Fukagata, R. Örlü, P. Schlatter, Effect of uniform blowing/suction in a turbulent boundary layer at moderate Reynolds number. Int. J. Heat Fluid Flow **55**, 132–142 (2015). https://doi.org/10.1016/j.ijheatfluidflow.2015.05.019
63. M. Buzzicotti, L. Biferale, F. Toschi, Statistical properties of turbulence in the presence of a smart small-scale control. Phys. Rev. Lett. **124**, 084504 (2020). https://doi.org/10.1103/PhysRevLett.124.084504
64. S. Levine, A. Kumar, G. Tucker, J. Fu, Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems (2020). https://doi.org/10.48550/ARXIV.2005.01643
65. I. Marusic, D. Chandran, A. Rouhi, M.K. Fu, D. Wine, B. Holloway, D. Chung, A.J. Smits, An energy-efficient pathway to turbulent drag reduction. Nature Commun. **12**(5805) (2021)