J. Fluid Mech. (2019), *vol.* 865, *pp.* 281–302.
doi:10.1017/jfm.2019.62

Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control

Jean Rabault[1,†], Miroslav Kuchta[1], Atle Jensen[1], Ulysse Réglade[1,2] and Nicolas Cerardi[1,2]

[1]Department of Mathematics, University of Oslo, 0316 Oslo, Norway

[2]CEMEF, Mines ParisTech, 06904 Sophia-Antipolis, France

(Received 25 June 2018; revised 17 December 2018; accepted 14 January 2019; first published online 20 February 2019)

We present the first application of an artificial neural network trained through a deep reinforcement learning agent to perform active flow control. It is shown that, in a two-dimensional simulation of the Kármán vortex street at moderate Reynolds number ($Re = 100$), our artificial neural network is able to learn an active control strategy from experimenting with the mass flow rates of two jets on the sides of a cylinder. By interacting with the unsteady wake, the artificial neural network successfully stabilizes the vortex alley and reduces drag by approximately 8 %. This is performed while using small mass flow rates for the actuation, of the order of 0.5 % of the mass flow rate intersecting the cylinder cross-section once a new pseudo-periodic shedding regime is found. This opens the way to a new class of methods for performing active flow control.

Key words: control theory, drag reduction

1. Introduction

Drag reduction and flow control are techniques of critical interest for industry (Brunton & Noack 2015). For example, 20 % of all energy losses on modern heavy-duty vehicles are due to aerodynamic drag (of which a large part is due to flow separation on tractor pillars (see Vernet *et al.* 2014)), and drag is naturally the main source of energy losses for an airplane. Drag is also a phenomenon that penalizes animals, and Nature shows examples of drag mitigation techniques. It is, for example, thought that structures of the skin of fast-swimming sharks interact with the turbulent boundary layer around the animal, and reduce drag by as much as 9 % (Dean & Bhushan 2010). This is therefore a proof-of-existence that flow control can be achieved with benefits, and is worth aiming for.

In the past, much research has been carried out towards so-called passive drag reduction methods, for example using micro vortex generators for passive control of transition to turbulence (Fransson *et al.* 2006; Shahinfar *et al.* 2012). While it should be underlined that this technique is very different from the one used by sharks

† Email address for correspondence: jean.rblt@gmail.com

(preventing transition to turbulence by energizing the linear boundary layer, *contra* reducing the drag of a fully turbulent boundary layer), benefits in terms of reduced drag can also be achieved. Another way to obtain drag reduction is by applying an active control to the flow. A number of techniques can be used in active drag control and have been proven effective in several experiments, a typical example being to use small jets (Schoppa & Hussain 1998; Glezer 2011). Interestingly, it has been shown that effective separation control can be achieved with even quite weak actuation, as long as it is used in an efficient way (Schoppa & Hussain 1998). This underlines the need to develop techniques that can effectively control a complex actuation input into a flow, in order to reduce drag.

Unfortunately, designing active flow control strategies is a complex endeavour (Duriez, Brunton & Noack 2016). Given a set of point measurements of the flow pressure or velocity around an object, there is no easy way to find a strategy to use this information in order to perform active control and reduce drag. The high dimensionality and computational cost of the solution domain (set by the complexity and nonlinearity inherent to fluid mechanics) mean that analytical solutions and real-time predictive simulations (that would decide which control to use by simulating several control scenarios in real time) seem out of reach. Despite the considerable efforts put in to the theory of flow control, and the use of a variety of analytical and semi-analytical techniques (Barbagallo, Sipp & Schmid 2009; Barbagallo *et al.* 2012; Sipp & Schmid 2016), bottom-up approaches based on an analysis of the flow equations face considerable difficulties when attempting to design flow control techniques. A consequence of these challenges is the simplicity of the control strategies used in most published works about active flow control, which traditionally focus on either harmonic or constant control input (Schoppa & Hussain 1998). Therefore, there is a need to develop efficient control methods, that perform complex active control and take full advantage of actuation possibilities. Indeed, it seems that, as of today, the actuation possibilities are large, but only simplistic (and probably suboptimal) control strategies are implemented. To the knowledge of the authors, only a few published examples of successful complex active control strategies are available with respect to the importance and extent of the field (Pastoor *et al.* 2008; Erdmann *et al.* 2011; Gautier *et al.* 2015; Guéniat, Mathelin & Hussaini 2016; Li *et al.* 2017).

In the present work, we aim at introducing for the first time deep neural networks and reinforcement learning to the field of active flow control. Deep neural networks are revolutionizing large fields of research, such as image analysis (Krizhevsky, Sutskever & Hinton 2012), speech recognition (Schmidhuber 2015) and optimal control (Mnih *et al.* 2015; Duan *et al.* 2016). Those methods have surpassed previous algorithms in all these examples, including methods such as genetic programming, in terms of complexity of the tasks learned and learning speed. It has been speculated that deep neural networks will bring advances also to fluid mechanics (Kutz 2017), but to date those have been limited to a few applications, such as the definition of reduced-order models (Wang *et al.* 2018), the effective control of swimmers (Verma, Novati & Koumoutsakos 2018) or performing particle image velocimetry (PIV) (Rabault, Kolaas & Jensen 2017). As deep neural networks, together with the reinforcement learning framework, have allowed recent breakthroughs in the optimal control of complex dynamic systems (Lillicrap *et al.* 2015; Schulman *et al.* 2017), it is natural to attempt to use them for optimal flow control.

Artificial neural networks (ANNs) are the attempt to reproduce in machines some of the features that are believed to be at the origin of the intelligent thinking of the brain (LeCun, Bengio & Hinton 2015). The key idea consists in performing computations

using a network of simple processing units, called neurons. The output value of each neuron is obtained by applying a transfer function on the weighted sum of its inputs (Goodfellow *et al.* 2016). When performing supervised learning, an algorithm, such as stochastic gradient descent, is then used for tuning the neurons' weights so as to minimize a cost function on a training set (Goodfellow *et al.* 2016). Given the success of this training algorithm, ANNs can in theory solve any problem since they are universal approximators: a large enough feed-forward neural network using a nonlinear activation function can fit arbitrarily well any function (Hornik, Stinchcombe & White 1989), and the recurrent neural network paradigm is even Turing-complete (Siegelmann & Sontag 1995). Therefore, virtually any problem or phenomenon that can be represented by a function could be a field of experimentation with ANNs. However, the problem of designing the ANNs, and designing the algorithms that train and use them, is still the object of active research.

While the case of supervised learning (i.e. when the solution is known and the ANN should simply be trained at reproducing it, such as image labelling or PIV) is now mostly solved owing to the advance of deep neural networks and deep convolutional networks (He *et al.* 2016), the case of reinforcement learning (when an agent tries to learn through the feedback of a reward function) is still the focus of much attention (Mnih *et al.* 2013; Gu *et al.* 2016; Schulman *et al.* 2017). In the case of reinforcement learning, an agent (controlled by the ANN) interacts with an environment through three channels of exchange of information in a closed-loop fashion. First, the agent is given access at each time step to an observation o_t of the state s_t of the environment. The environment can be any stochastic process, and the observation is only a noisy, partial description of the environment. Second, the agent performs an action, a_t, that influences the time evolution of the environment. Finally, the agent receives a reward r_t depending on the state of the environment following the action. The reinforcement learning framework consists in finding strategies to learn from experimenting with the environment, in order to discover control sequences $a(t = 1, \ldots, T)$ that maximize the reward. The environment can be any system that provides the interface (o_t, a_t, r_t): it could be an Atari game emulator (Mnih *et al.* 2013), a robot acting in the physical world that should perform a specific task (Kober, Bagnell & Peters 2013), or a fluid mechanics system whose drag should be minimized in our case.

In the present work, we apply for the first time the deep reinforcement learning (DRL) paradigm (i.e. reinforcement learning performed on a deep ANN) to an active flow control problem. We use a proximal policy optimization (PPO) method (Schulman *et al.* 2017) together with a fully connected artificial neural network (FCANN) to control two synthetic jets located on the sides of a cylinder immersed in a constant flow in a two-dimensional (2D) simulation. The geometry is chosen owing to its simplicity, the low computational cost associated with resolving a 2D unsteady wake at moderate Reynolds number, and the simultaneous presence of the causes that make active flow control challenging (time dependence, nonlinearity, high dimensionality). The PPO agent manages to control the jets and to interact with the unsteady wake to reduce the drag. We have chosen to release all our code as open source, to help trigger interest in those methods and facilitate further developments. In the following, we first present the simulation environment, before giving details about the network and reinforcement framework, and finally we offer an overview of the results obtained.

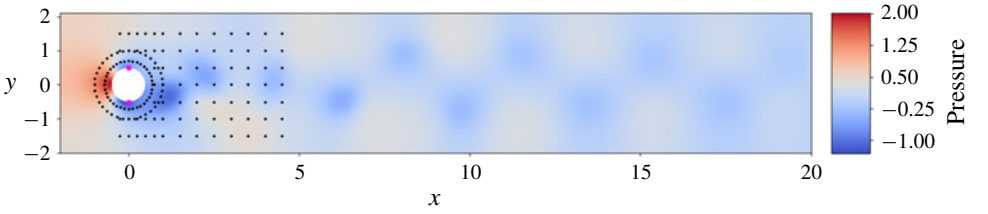

FIGURE 1. (Colour online) Unsteady non-dimensional pressure wake behind the cylinder after flow initialization without active control. The location of the velocity probes is indicated by the black dots. The location of the control jets is indicated by the red dots.

2. Methodology

2.1. *Simulation environment*

The PPO agent performs active flow control in a 2D simulation environment. In the following, all quantities are considered non-dimensionalized. The geometry of the simulation, adapted from the 2D test case of well-known benchmarks (Schäfer *et al.* 1996), consists of a cylinder of non-dimensional diameter $D = 1$ immersed in a box of total non-dimensional length $L = 22$ (along the X-axis) and height $H = 4.1$ (along the Y-axis). Similarly to the benchmark of Schäfer *et al.* (1996), the cylinder is slightly off the centreline of the domain (a shift of 0.05 in the Y-direction is used), in order to help trigger the vortex shedding. The inflow profile (on the left wall of the domain) is parabolic, following the formula (cf. 2D-2 test case in Schäfer *et al.* (1996))

$$U(y) = 6(H/2 - y)(H/2 + y)/H^2, \tag{2.1}$$

where $(U(y), V(y) = 0)$ is the non-dimensionalized velocity vector. Using this velocity profile, the mean velocity magnitude is $\bar{U} = 2U(0)/3 = 1$. A no-slip boundary condition is imposed on the top and bottom walls and on the solid walls of the cylinder. An outflow boundary condition is imposed on the right wall of the domain. The configuration of the simulation is shown in figure 1. The Reynolds number based on the mean velocity magnitude and cylinder diameter ($Re = \bar{U}D/\nu$, with ν the kinematic viscosity) is set to $Re = 100$. Computations are performed on an unstructured mesh generated with Gmsh (Geuzaine & Remacle 2009). The mesh is refined around the cylinder and is composed of 9262 triangular elements. A non-dimensional, constant numerical time step $\mathrm{d}t = 5 \times 10^{-3}$ is used. The total instantaneous drag on the cylinder C is computed as follows:

$$F_D = \int_C (\boldsymbol{\sigma} \cdot \boldsymbol{n}) \cdot \boldsymbol{e}_x \, \mathrm{d}S, \tag{2.2}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\boldsymbol{n}$ is the unit vector normal to the outer cylinder surface, and $\boldsymbol{e}_x = (1, 0)$. In the following, the drag is normalized into the drag coefficient

$$C_D = \frac{F_D}{\frac{1}{2}\rho \bar{U}^2 D}, \tag{2.3}$$

where $\rho = 1$ is the non-dimensional volumetric mass density of the fluid. Similarly, the lift force F_L and lift coefficient C_L are defined as

$$F_L = \int_C (\boldsymbol{\sigma} \cdot \boldsymbol{n}) \cdot \boldsymbol{e}_y \, \mathrm{d}S, \tag{2.4}$$

and

$$C_L = \frac{F_L}{\frac{1}{2}\rho \bar{U}^2 D}, \tag{2.5}$$

where $\boldsymbol{e}_y = (0, 1)$.

In the interest of short solution time (e.g. Valen-Sendstad *et al.* 2012), the governing Navier–Stokes equations are solved in a segregated manner. More precisely, the incremental pressure correction scheme (IPCS) method (Goda 1979) with an explicit treatment of the nonlinear term is used. More details are available in appendix B. Spatial discretization then relies on the finite-element method implemented within the FEniCS framework (Logg, Mardal & Wells 2012).

We remark that both the mesh density and the Reynolds number could easily be increased in a later study, but are kept low here as that allows for fast training on a laptop, which is the primary aim of our proof-of-concept demonstration.

In addition, two jets (1 and 2) normal to the cylinder wall are implemented on the sides of the cylinder, at angles $\theta_1 = 90°$ and $\theta_2 = 270°$ relative to the flow direction. The jets are controlled through their non-dimensional mass flow rates, Q_i, $i = 1, 2$, and are set through a parabolic-like velocity profile going to zero at the edges of the jet (see appendix B for the details). The jet widths are set to 10°. Choosing jets normal to the cylinder wall, located at the top and bottom extremities of the cylinder, means that all drag reduction observed will be the result of indirect flow control, rather than direct injection of momentum. In addition, the control is set up in such a way that the total mass flow rate injected by the jets is zero, i.e. $Q_1 + Q_2 = 0$. This synthetic jets condition is chosen as it is more realistic than a case when mass is added or subtracted from the flow, and makes the numerical scheme more stable, especially with respect to the boundary conditions of the problem. In addition, it ensures that the drag reduction observed is the result of actual flow control, rather than some sort of propulsion phenomenon. In the following, the injected mass flow rates are normalized as follows:

$$Q_i^* = Q_i / Q_{ref}, \tag{2.6}$$

where $Q_{ref} = \int_{-D/2}^{D/2} \rho U(y)\, \mathrm{d}y$ is the reference mass flow rate intercepting the cylinder. During learning, we impose that $|Q_i^*| < 0.06$. This helps in the learning process by preventing non-physically large actuation, and prevents problems in the numerics of the simulation by enforcing the Courant–Friedrichs–Lewy (CFL) condition close to the actuation jets.

Finally, information is extracted from the simulation and provided to the PPO agent. A total of 151 velocity probes, which report the local value of the horizontal and vertical components of the velocity field, are located in several locations in the neighbourhood of the cylinder and in its wake (see figure 1). This means that the network gets detailed information about the flow configuration, which is our objective, as this article focuses on finding the best possible control strategy of the vortex shedding pattern. A different question would be to assess the ability of the network to perform control with a partial observation of the system. To illustrate that this is possible with adequate training, we provide some results with an input layer reduced to 11 and five probes in appendix E, but further parameter space study and sensitivity analysis is beyond the scope of the present paper and is left to future work.

An unsteady wake develops behind the cylinder, which is in good agreement with what is expected at this Reynolds number. A simple benchmark of the simulation was performed by observing the pressure fluctuations, drag coefficient and Strouhal

number $St = fD/\bar{U}$, where f is the vortex shedding frequency. The mean value of C_D in the case without actuation (approximately 3.205) is within 1 % of what is reported in the benchmark of Schäfer *et al.* (1996), which validates our simulations, and similar agreement is found for St (typical value of approximately 0.30). In addition, we also performed tests on refined meshes, going up to approximately 30 000 triangular elements, and found that the mean drag varied by less than 1 % following mesh refinement. A pressure field snapshot of the fully developed unsteady wake is presented in figure 1.

2.2. *Network and reinforcement learning framework*

As stated in the introduction, DRL sees the fluid mechanic simulation as yet another environment to interact with through three simple channels: the observation o_t (here, an array of point measurements of velocity obtained from the simulation), the action a_t (here, the active control of the jets, imposed on the simulation by the learning agent), and the reward r_t (here, the time-averaged drag coefficient provided by the environment, penalized by the mean lift coefficient magnitude; see further in this section). Based on this limited information, DRL trains an ANN to find closed-loop control strategies deciding a_t from o_t at each time step, so as to maximize r_t.

Our DRL agent uses the PPO method (Schulman *et al.* 2017) for performing learning. PPO is a reinforcement learning algorithm that belongs to the family of policy gradient methods. This method was chosen for several reasons. In particular, it is less complex mathematically and faster than concurring trust region policy optimization (TRPO) methods (Schulman *et al.* 2015), and requires little to no metaparameter tuning. It is also better adapted to continuous control problems than deep Q network (DQN) learning (Mnih *et al.* 2015) and its variations (Gu *et al.* 2016). From the point of view of the fluid mechanist, the PPO agent acts as a black box (though details about its internals are available in Schulman *et al.* (2017) and the references therein). A brief introduction to the PPO method is provided in appendix C.

The PPO method is episode-based, which means that it learns from performing active control for a limited amount of time before analysing the results obtained and resuming learning with a new episode. In our case, the simulation is first performed with no active control until a well-developed unsteady wake is obtained, and the corresponding state is saved and used as a start for each subsequent learning episode.

The instantaneous reward function, r_t, is computed as follows:

$$r_t = -\langle C_D \rangle_T - 0.2|\langle C_L \rangle_T|, \tag{2.7}$$

where $\langle \cdot \rangle_T$ indicates the sliding average back in time over a duration corresponding to one vortex shedding cycle. The ANN tries to maximize this function r_t, i.e. to make it as little negative as possible, therefore minimizing drag and mean lift (to take into account long-term dynamics, an actualized reward is actually used during gradient descent; see appendix C for more details). This specific reward function has several advantages compared with using the plain instantaneous drag coefficient. Firstly, using values averaged over one vortex shedding cycle leads to less variability in the value of the reward function, which was found to improve learning speed and stability. Secondly, the use of a penalization term based on the lift coefficient is necessary to prevent the network from 'cheating'. Indeed, in the absence of this penalization, the ANN manages to find a way to modify the configuration of the flow in such a way that a larger drag reduction is obtained (up to approximately 18 %

drag reduction, depending on the simulation configuration used), but at the cost of a large induced lift, which is damaging in most practical applications.

The ANN used is relatively simple, being composed of two dense layers of 512 fully connected neurons, plus the layers required to acquire data from the probes, and generate data for the two jets. This network configuration was found empirically through trial and error, as is usually done with ANNs. Results obtained with smaller networks are less good, as their modelling ability is not sufficient in regards to the complexity of the flow configuration obtained. Larger networks are also less successful, as they are harder to train. In total, our network has slightly over 300 000 weights. For more details, readers are referred to the code implementation (see appendix A).

At first, no learning could be obtained from the PPO agent interacting with the simulation environment. The reason for this was the difficulty for the PPO agent to learn the necessity to set time-correlated, continuous control signals, as the PPO first tries purely random control and must observe some improvement on the reward function for performing learning. Therefore, we implemented two tricks to help the PPO agent learn control strategies:

(i) The control value provided by the network is kept constant for a duration of 50 numerical time steps, corresponding to approximately 7.5 % of the vortex shedding period. This means, in practice, that the PPO agent is allowed to interact with the simulation and update its control only each 50 time steps.

(ii) The control is made continuous in time to avoid jumps in the pressure and velocity due to the use of an incompressible solver. For this, the control at each time step in the simulation is obtained for each jet as $c_{s+1} = c_s + \alpha(a - c_s)$, where c_s is the control of the jet considered at the previous numerical time step, c_{s+1} is the new control, a is the action set by the PPO agent for the current 50 time steps and $\alpha = 0.1$ is a numerical parameter.

Using those technical tricks, and choosing an episode duration $T_{max} = 20.0$ (which spans approximately 6.5 vortex shedding periods, and corresponds to 4000 numerical time steps, i.e. 80 actions by the network), the PPO agent is able to learn a control strategy after typically approximately 200 epochs corresponding to 1300 vortex shedding periods or 16 000 sampled actions, which requires roughly 24 hours of training on a modern desktop using one single core. This training time could be reduced easily by at least a factor of 10, using more cores to parallelize the data sampling from the epochs which is a fully parallel process. Fine tuning the policy can take a bit longer time, and up to approximately 350 epochs can be necessary to obtain a fully stabilized control strategy. A training has also been performed going up to over 1000 episodes to confirm that no more changes were obtained if the network is allowed to train for a significantly longer time. Most of the computation time is spent in the flow simulation. This set-up with simple, quick simulations makes experimentation and reproduction of our results easy, while being enough for a proof-of-concept in the context of a first application of reinforcement learning to active flow control and providing an interesting control strategy for further analysis.

3. Results

3.1. *Drag reduction through active flow control*

Robust learning is obtained by applying the methodology presented in the previous section. This is illustrated by figure 2, which presents the averaged learning curve

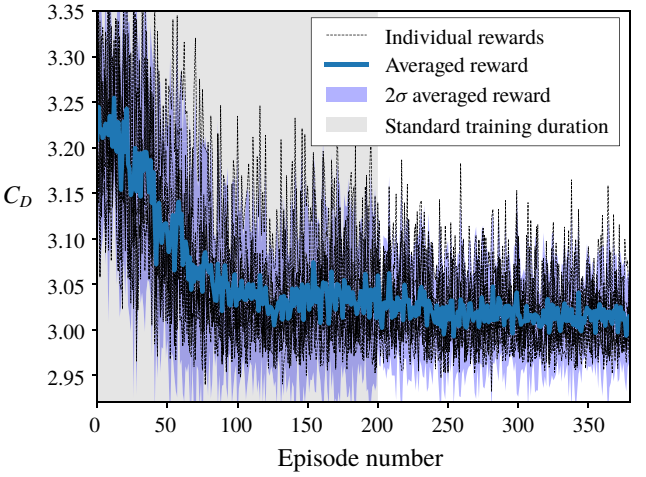

FIGURE 2. (Colour online) Illustration of the robustness of the learning process. The drag values reported are obtained at each training epoch (including exploration noise), for 10 different trainings using the same metaparameters, but different values of the random seed. Robust learning takes place within 200 epochs, with fine converged strategy requiring a few more epochs to stabilize. The drag reduction is slightly less than what is reported in the rest of the text, as these results include the random exploration noise and are computed over the second half of the training epochs, where some of the transient in the drag value is still present during training.

and the confidence interval corresponding to 10 different trainings performed using different seeds for the random number generator. In figure 2, the drag presented is obtained by averaging the drag coefficient obtained on the second half of each training epoch. This averaging is performed to smooth the effect of both vortex shedding and drag fluctuations due to the exploration. While it may include part of the initial transition from the undisturbed vortex shedding to the controlled case, it is a good relative indicator of policy convergence. Estimating at each epoch the asymptotic quality of the fully established control regime would be too expensive, which is the reason why we resort to this averaged value. Using different random seeds results in different trainings, as random data are used in the exploration noise and for the random sampling of the replay memory used during stochastic gradient descent. All other parameters are kept constant. The data presented indicate that learning takes place consistently in approximately 200 epochs, with fine convergence and tuning requiring up to approximately 400 epochs. Owing to the presence of exploration noise and the averaging being performed on a time window including some of the transition in the flow configuration from free shedding to active control, the quality of the drag reduction reported in figure 2 is slightly less than in the case of deterministic control in the pseudo-periodic actively controlled regime (i.e. when a modified stable vortex shedding is obtained with the most likely action of the optimal policy being picked up at each time step implying that, in the case of deterministic control, no exploration noise is present), which is as expected. The final drag reduction value obtained in the deterministic mode (not shown so as not to overload figure 2) is also consistent across the runs.

Therefore, it is clear that the ANN is able to consistently reduce drag by applying active flow control following training through the DRL/PPO algorithm, and that the

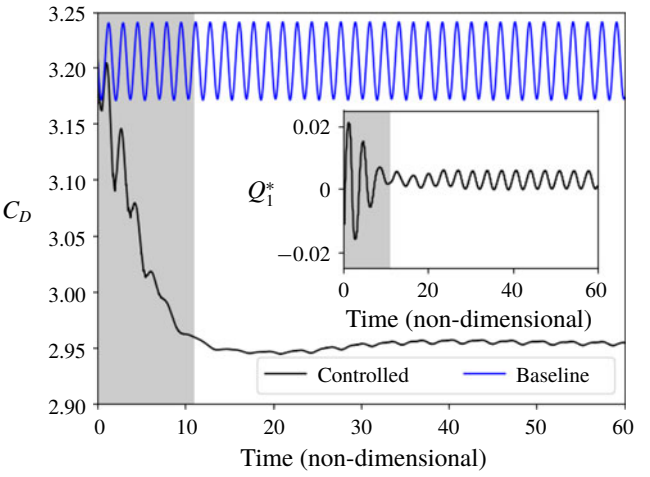

FIGURE 3. (Colour online) Time-resolved value of the drag coefficient C_D in the case without (baseline curve) and with (controlled curve) active flow control, and corresponding normalized mass flow rate of the control jet 1 (Q_1^*, inset). The effect of the flow control on the drag is clearly visible: a reduction of the drag of approximately 8 % is observed, and the fluctuations in time due to vortex shedding are drastically reduced. Two successive phases can be distinguished in the mass flow rate control: first, a relatively large control is used to change the flow configuration, up to a non-dimensional time of approximately 11, before a pseudo-periodic regime with very limited flow control is established.

learning is both stable and robust. All results presented further in both this section and the next one are obtained using deterministic prediction, and therefore exploration noise is not present in the following figures and results. The time series for the drag coefficient obtained using the active flow control strategy discovered through training in the first run, compared with the baseline simulation (no active control, i.e. $Q_1 = Q_2 = 0$), is presented in figure 3 together with the corresponding control signal (inset). Similar results and control laws are obtained for all training runs, and the results presented in figure 3 are therefore representative of the learning obtained with all 10 realizations.

In the case without actuation (baseline), the drag coefficient C_D varies periodically at twice the vortex shedding frequency, as should be expected. The mean value for the drag coefficient is $\langle C_D \rangle \approx 3.205$, and the amplitude of the fluctuations of the drag coefficient is approximately 0.034. By contrast, the mean value for the drag coefficient in the case with active flow control is $\langle C_D' \rangle \approx 2.95$, which represents a drag reduction of approximately 8 %.

To put this drag reduction into perspective, we estimate the drag obtained in the hypothetical case where no vortex shedding is present. For this, we perform a simulation with the upper half-domain and a symmetric boundary condition on the lower boundary (which cuts the cylinder through its equator). More details about this simulation are presented in appendix D. The steady-state drag obtained on a full cylinder in the case without vortex shedding is then $C_{Ds} = 2.93$ (see appendix D), which means that the active control is able to suppress approximately 93 % of the drag increase observed in the baseline without control compared with the hypothetical reference case where the flow would be kept completely stable.

In addition to this reduction in drag, the fluctuations of the drag coefficient are reduced to approximately 0.0016 by the active control, i.e. a factor of roughly 20 compared with the baseline. Similarly, fluctuations in lift are reduced, though by a more modest factor of approximately 5.7. Finally, a Fourier analysis of the drag coefficients obtained shows that the actuation slightly modifies the characteristic frequency of the system. The actively controlled system has a shedding frequency approximately 3.5 % lower than the baseline.

Several interesting points are visible from the active control signal imposed by the ANN presented in figure 3. Firstly, the active flow control is composed of two phases. In the first one, the ANN changes the configuration of the flow by performing a relatively large transient actuation (non-dimensional time ranging from 0 to approximately 11). This changes the flow configuration, and sets the system in a state in which less drag is generated. Following this transient actuation, a second regime is reached in which a smaller actuation amplitude is used. The actuation in this new regime is pseudo-periodic. Therefore, it appears that the ANN has found a way to both set the flow in a modified configuration in which less drag is present, and keep it in this modified configuration at a relatively small cost. In a separate simulation, the small actuation present in the pseudo-periodic regime once the initial actuation has taken place was suppressed. This led to a rapid collapse of the modified flow regime, and the original base flow configuration was recovered. As a consequence, it appears that the modified flow configuration is unstable, though only small corrections are needed to keep the system in its neighbourhood.

Secondly, it is striking to observe that the ANN resorts to quite small actuations. The peak value for the norm of the non-dimensional control mass flow rate Q_1^*, which is reached during the transient active control regime, is only approximately 0.02, i.e. a factor 3 smaller than the maximum value allowed during training. Once the pseudo-periodic regime is established, the peak value of the actuation is reduced to approximately 0.006. This is an illustration of the sensitivity of the Navier–Stokes equations to small perturbations, and a proof that this property of the equations can be exploited to actively control the flow configuration, if forcing is applied in an appropriate manner.

3.2. *Analysis of the control strategy*

The ANN trained through DRL learns a control strategy by using a trial-and-error method. Understanding which strategy an ANN decides to use from the analysis of its weights is known to be challenging, even on simple image analysis tasks. Indeed, the strategy of the network is encoded in the complex combination of the weights of all its neurons. A number of properties of each individual network, such as the variations in architecture, make systematic analysis challenging (Schmidhuber 2015; Rauber *et al.* 2017). Through the combination of the neuron weights, the network builds its own internal representation of how the flow in a given state will be affected by actuation, and how this will affect the reward value. This is a sort of private, 'encrypted' model obtained through experience and interaction with the flow. Therefore, it appears challenging to directly analyse the control strategy from the trained network, which should be considered rather as a black box in this regard.

Instead, we can look at macroscopic flow features and how the active control modifies them. This pinpoints the effect of the actuation on the flow and separation happening in the wake. Representative snapshots of the flow configuration in the baseline case (no actuation), and in the controlled case when the pseudo-periodic

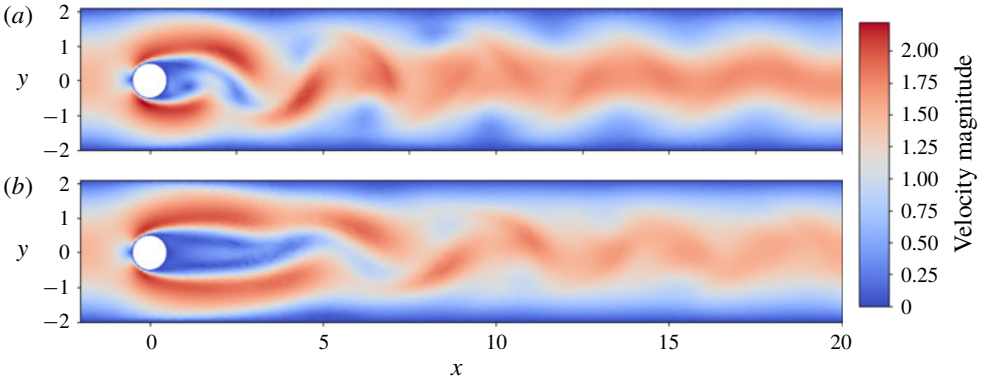

FIGURE 4. (Colour online) Comparison of representative snapshots of the velocity magnitude in the case without actuation (*a*), and with active flow control (*b*). The lower panel corresponds to the established pseudo-periodic modified regime, which is attained after the initial transient control.

regime is reached (i.e. after the initial large transient actuation), are presented in figure 4. As can be seen in figure 4, the active control leads to a modification of the 2D flow configuration. In particular, the Kármán alley is altered in the case with active control and the velocity fluctuations induced by the vortices are globally less strong, and less active close to the upper and lower walls. More strikingly, the extent of the recirculation area is dramatically increased. Defining the recirculation area as the region in the downstream neighbourhood of the cylinder where the horizontal component of the velocity is negative, we observe a 130 % increase in the recirculation area, averaged over the pseudo-period. The recirculation area in the active control case represents 103 % of what is obtained in the hypothetical stable configuration of appendix D (so, the recirculation area is slightly larger in the controlled case than in the hypothetical stable case, though the difference is so small that it may be due to a side effect such as slightly larger separation close to the jets, rather than a true change in the extent of the developed wake), while the recirculation area in the baseline configuration with vortex shedding is only 44 % of this same stable configuration value. This is, similarly to what was observed for C_D, an illustration of the efficiency of the control strategy at reducing the effect of vortex shedding.

To go into more details, we look at the mean and the standard deviation (STD) of the flow velocity magnitude and pressure, averaged over a large number of vortex shedding periods (in the case with active flow control, we consider the pseudo-periodic regime). Results are presented in figure 5. Several interesting points are visible from both the velocity and pressure data. Similarly to what was observed on the snapshots, the area of the separated wake is larger in the case with active control than in the baseline. This is clearly visible from the mean value plots of both velocity magnitude and pressure. This feature results in a lower mean pressure drop in the wake of the cylinder in the case with active control, which is the cause of the reduced drag. This phenomenon is similar to boat tailing, which is a well-known method for reducing the drag between bluff bodies. However, in the present case, this is attained through applying small controls to the flow rather than modifying the shape of the obstacle. The STD figures also clearly show a decreased level of fluctuations of

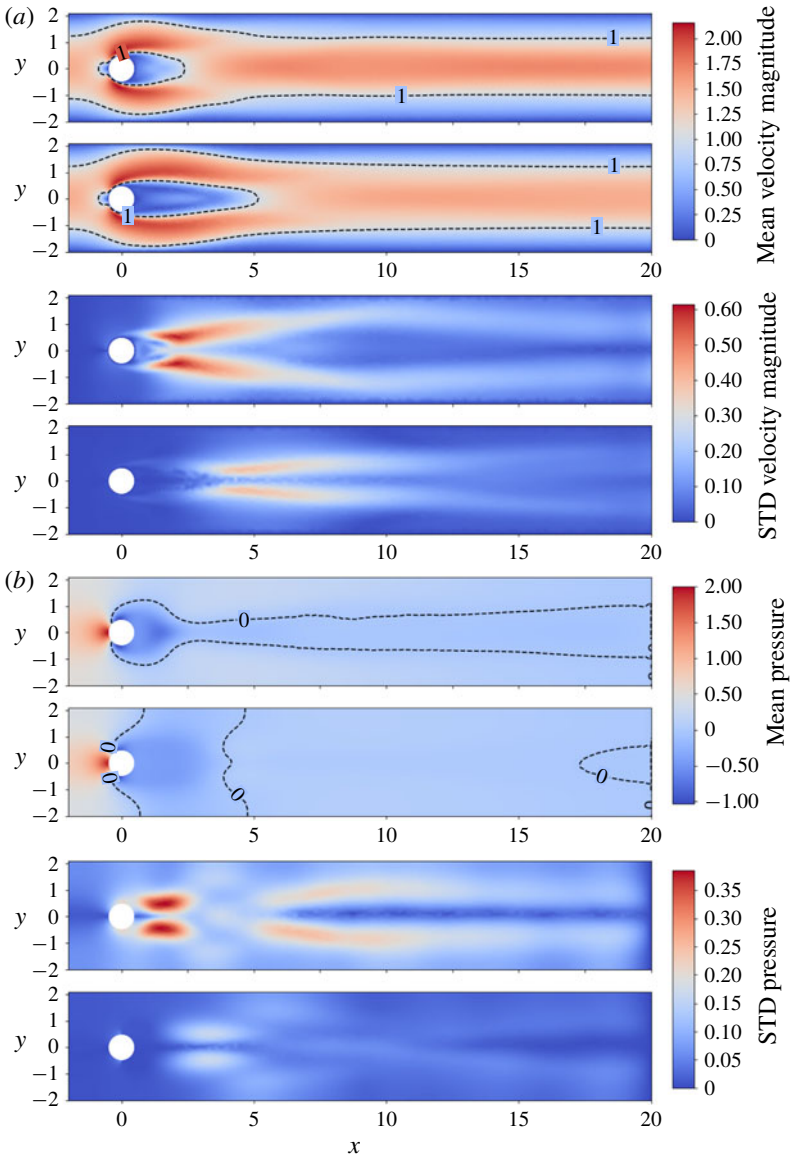

FIGURE 5. (Colour online) Comparison of the flow morphology without (top part of each double panel) and with (bottom part of each double panel) actuation. (*a*) Velocity magnitude comparisons: mean (upper double panel) and STD (lower double panel). (*b*) Pressure comparisons: mean (upper double panel) and STD (lower double panel). The colour bar is common to both parts of each double panel. A clear increase in size of the recirculation area is observed with actuation, which is associated with a lower pressure drop behind the cylinder.

both the velocity magnitude and the pressure in the wake, as well as a displacement downstream of the cylinder of the regions where highest flow variations are recorded.

4. Conclusion

We show for the first time that the deep reinforcement learning (DRL) paradigm, and more specifically the proximal policy optimization (PPO) algorithm, can discover an active flow control strategy for synthetic jets on a cylinder, and control the configuration of the 2D Kármán vortex street. From the point of view of the artificial neural network (ANN) and DRL, this is just yet another environment to interact with. The discovery of the control strategy takes place through the optimization of a reward function, here defined from the fluctuations of the drag and lift components experienced by the cylinder. A drag reduction of up to approximately 8 % is observed. In order to reduce drag, the ANN decides to increase the area of the separated region, which in turn induces a lower pressure drop behind the cylinder, and therefore lower drag. This brings the flow into a configuration that presents some similarities with what would be obtained from boat tailing. The value of the drag coefficient and extent of the recirculation bubble when control is turned on are very close to what is obtained by simulating the flow around a half-cylinder using a symmetric boundary condition at the lower wall, which allows one to estimate the drag expected around a cylinder at comparable Reynolds number if no vortex shedding was present. This implies that the active control is able to effectively cancel the detrimental effect of vortex shedding on drag. The learning obtained is remarkable, as little metaparameter tuning was necessary, and training takes place in about one day on a laptop. In addition, we have resorted to strong regularization of the output of the DRL agent through under-sampling of the simulation and imposing a continuous control for helping the learning process. It could be expected that relaxing those constraints, i.e. giving more freedom to the network, could lead to even more efficient strategies.

These results are potentially of considerable importance for fluid mechanics, as they provide a proof that DRL can be used to solve the high dimensionality, analytically untractable problem of active flow control. The ANN and DRL approach has a number of strengths which make it an appealing methodology. In particular, ANNs allow for an efficient global approximation of strongly nonlinear functions, and they can be trained through direct experimentation of the DRL agent with the flow, which makes it in theory easily applicable to both simulations and experiments without changes in the DRL methodology. In addition, once trained, the ANN requires only a few calculations to compute the control at each time step. In the present case when two hidden layers of width 512 are used, most of the computational cost comes from a matrix multiplication, where the size of the matrices to multiply is $[512, 512]$. This is much less computationally expensive than the underlying problem. Finally, we are able to show that learning takes place in a timely manner, requiring a reasonable number of vortex shedding periods to obtain a converged strategy.

This work opens a number of research directions, including applying the DRL methodology to more complex simulations, for example more realistic three-dimensional large-eddy simulations or direct numerical simulations on large computer clusters, or even applying such an approach directly to a real-world experiment. In addition, a number of interesting questions arise from the use of ANNs and DRL. For example, can some form of transfer learning be used between simulations and the real world if the simulations are realistic enough (i.e. can one train an ANN in a simulation, and then use it in the real world)? The use of DRL for active flow control may provide a technique to finally take advantage of advanced, complex flow actuation possibilities, such as those allowed by complex jet actuator arrays.

Acknowledgement

The help of T. Kvernes for setting up the computational infrastructure used in this work is gratefully acknowledged. In addition, we want to thank Professors T. Coupez and E. Hachem for stimulating discussions, and helping organize the visit of Ulysse Réglade and Nicolas Cerardi to the University of Oslo. Funding from the Norwegian Research Council through the Petromaks 2 grants 'WOICE' (grant number 233901) and 'DOFI' (grant number 280625), and through the project 'Rigspray' (grant number 256435) is gratefully acknowledged. We want to thank the three reviewers, whose constructive feedback has greatly contributed to improving the quality of our manuscript.

Appendix A. Open source code

The source code of this project is released as open source on the GitHub of the author: https://github.com/jerabaul29/Cylinder2DFlowControlDRL. The simulation environment is based on the open-source finite-element framework FEniCS (Logg *et al.* 2012) version 2017.2.0. The PPO agent is based on the open-source implementation provided by Tensorforce (Schaarschmidt, Kuhnle & Fricke 2017), which builds on top of the Tensorflow framework for building artificial neural networks (Abadi *et al.* 2016). More details about the simulation environment and the DRL algorithm are presented in appendices B and C, respectively.

Appendix B. Details of simulation environment

The response of the environment to the action tuple (Q_1, Q_2) provided by the agent is determined by computing a solution for the Navier–Stokes equations in the computational domain Ω:

$$\left.\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot(\nabla \boldsymbol{u}) &= -\nabla p + Re^{-1}\Delta \boldsymbol{u} && \text{in } \Omega, \\ \nabla\cdot\boldsymbol{u} &= 0 && \text{in } \Omega. \end{aligned}\right\} \tag{B 1}$$

To close (B 1), the boundary of the domain is partitioned (see also figure 6) into an inflow part Γ_I, a no-slip part Γ_W, an outflow part Γ_O and the jet parts Γ_1 and Γ_2. Following this decomposition, the system is considered with the following boundary conditions:

$$\left.\begin{aligned} -p\boldsymbol{n} + Re^{-1}(\boldsymbol{n}\cdot\nabla\boldsymbol{u}) &= 0 && \text{on } \Gamma_O, \\ \boldsymbol{u} &= 0 && \text{on } \Gamma_W, \\ \boldsymbol{u} &= \boldsymbol{U} && \text{on } \Gamma_I, \\ \boldsymbol{u} &= f_{Q_i}, && \text{on } \Gamma_i,\ i = 1, 2. \end{aligned}\right\} \tag{B 2}$$

Here, $\boldsymbol{U}$ is the inflow velocity profile (2.1) while f_{Q_i} are radial velocity profiles which mimic suction or injection of the fluid by the jets. The functions are chosen such that the prescribed velocity continuously joins the no-slip condition imposed on the Γ_W surfaces of the cylinder. More precisely, we set $f_{Q_i} = A(\theta;\, Q_i)(x, y)$ where the modulation depends on the angular coordinate θ (cf. figure 6), such that for the jet with width ω and centred at θ_0 placed on the cylinder of radius R the modulation is set as

$$A(\theta;\, Q) = Q\frac{\pi}{2\omega R^2}\cos\left(\frac{\pi}{\omega}(\theta - \theta_0)\right). \tag{B 3}$$

We remark that with this choice the boundary conditions on the jets are in fact controlled by single scalar values Q_i. Negative values of Q_i correspond to suction.

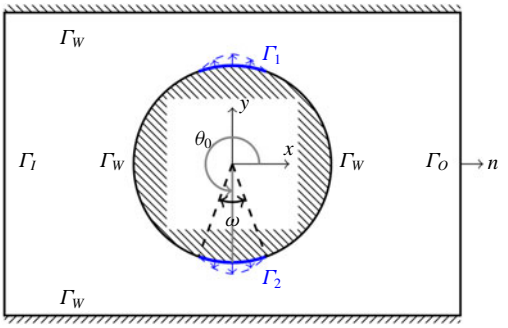

FIGURE 6. (Colour online) Solution domain Ω (not to scale) for the Navier–Stokes equations of the simulation environment. On parts of the cylinder boundary (in blue) velocity boundary conditions determined by Q_i are prescribed.

To solve (B 1)–(B 2) numerically, the incremental pressure correction scheme (IPCS) method (Goda 1979) with explicit treatment of the nonlinear term is adopted. Let δt be the step size of the temporal discretization. Then the velocity u and pressure p for the next temporal level are computed from the current solutions u_0 and p_0 in three steps: the tentative velocity step

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}_0}{\delta t} + \boldsymbol{u}_0 \cdot (\boldsymbol{\nabla} \boldsymbol{u}_0) = -\boldsymbol{\nabla} p_0 + Re^{-1} \Delta \frac{\boldsymbol{u}^* + \boldsymbol{u}_0}{2} \quad \text{in } \Omega, \tag{B 4}$$

the pressure projection step

$$-\Delta(p - p_0) = -\delta t^{-1} \boldsymbol{\nabla} \cdot \boldsymbol{u}^* \quad \text{in } \Omega, \tag{B 5}$$

and the velocity correction step

$$\boldsymbol{u} - \boldsymbol{u}^* = -\delta t^{-1} \boldsymbol{\nabla}(p - p_0) \quad \text{in } \Omega. \tag{B 6}$$

The steps (B 4) and (B 6) are considered with the boundary conditions (B 2), while for pressure projection (B 5) a Dirichlet boundary condition $p = 0$ is used on Γ_O and the remaining boundaries have $\boldsymbol{n} \cdot \boldsymbol{\nabla} p = 0$.

Discretization of the IPCS scheme (B 4)–(B 6) relies on the finite-element method. More specifically, the velocity and pressure fields are discretized, respectively, in terms of the continuous quadratic and continuous linear elements on triangular cells. Because of the explicit treatment of the nonlinearity in (B 4), all the matrices of linear systems in the scheme are assembled (and their solvers set up) once prior to entering the time loop in which only the right-hand side vectors are updated. In our implementation the solvers for the linear systems involved are the sparse direct solvers from the UMFPACK library (Davis 2004). We remark that the finite-element mesh used for training consists of 9262 elements and gives rise to systems with 37 804 and 4820 unknowns in (B 4), (B 6) and (B 5), respectively.

Once u and p have been computed, the drag and lift are integrated over the entire surface of the cylinder. In particular, the jet surfaces are included.

Appendix C. Deep reinforcement learning, policy gradient method and PPO

In this appendix, we give a brief overview of the policy gradient and PPO methods. This is a summary of the main lines of the algorithms presented in the corresponding literature (Lillicrap *et al.* 2015; Duan *et al.* 2016), and the reader should consult these references for further details.

In all the following, the usual DRL framework is used: an ANN controlled by a DRL agent interacts with a complex system (the environment) through three channels: a noisy, partial observation of the system (o_t), an action applied on the system by the ANN (a_t), and a reward provided by the system depending on its state (r_t). The detailed internal state s_t of the system is usually not available. The interaction takes place at discrete time steps.

As previously stated, the algorithm used in this work belongs to the policy gradient class. The aim of this method is to directly obtain the optimal policy $\pi^*(a_t|o_t)$, i.e. the distribution probability of action a_t given the observation o_t for maximizing the long-term actualized reward $R(t) = \sum_{i>t} \gamma^{i-t} r_i$, where $0 < \gamma < 1$ is a discount factor in time. In the case of policy gradient methods, the policy is directly modelled by the ANN. This is in contrast to methods such as Q-learning, where an indirect description of the policy (in the case of Q-learning, the quality function Q, i.e. the expected return for each action) is modelled by the ANN. Policy gradient methods have better stability and convergence properties than Q-learning, and are a more natural solution for continuous control cases. However, this comes at the cost of slightly worse exploration properties.

In the following, we will formulate the learning problem as finding all the weights of the ANN, collectively described by the Θ variable, such as to maximize the expected return:

$$R_{max} = \max_{\Theta} \mathbb{E}\left[\sum_{t=0}^{H} R(s_t)|\pi_\Theta\right], \tag{C 1}$$

where π_Θ is the policy function described by the ANN, when it has the weights Θ, and s_t is the (hidden) state of the system.

Following this optimization formulation, one has naturally $\pi^* = \pi_{\Theta=\Theta^*}$, where Θ^* is the set of weights obtained through the maximization (C 1).

The maximization (C 1) is solved through gradient descent performed on the weights Θ of the ANN, following experimental sampling of the system through interaction with the environment. Sophisticated gradient descent batch methods, such as Adagrad or Adadelta, allow one to automatically set the learning rate in accordance to the local speed of the gradient descent and provide stabilization of the gradient by adding momentum. More specifically, if we denote τ a (s–a–r) sequence,

$$\tau = (s_0, a_0, r_0), (s_1, a_1, r_1), \ldots, (s_H, a_H, r_H), \ldots, \tag{C 2}$$

and we overload the R operator as $R(\tau) = \sum_i \gamma^i r_i$, then the value function obtained with the weights Θ, which is the quantity that should be maximized, can be written as

$$V(\Theta) = \mathbb{E}\left[\sum_{t=0}^{H} R(s_t, u_t)|\pi_\theta\right] = \sum_{\tau} \mathbb{P}(\tau, \Theta) R(\tau). \tag{C 3}$$

From this point, elementary manipulations lead to

$$\begin{aligned}\nabla_\Theta V(\Theta) &= \sum_\tau \nabla_\Theta \mathbb{P}(\tau, \Theta) R(\tau) \\ &= \sum_\tau \frac{\mathbb{P}(\tau, \Theta)}{\mathbb{P}(\tau, \Theta)} \nabla_\Theta \mathbb{P}(\tau, \Theta) R(\tau) \\ &= \sum_\tau \mathbb{P}(\tau, \Theta) \frac{\nabla_\Theta \mathbb{P}(\tau, \Theta)}{\mathbb{P}(\tau, \Theta)} R(\tau) \\ &= \sum_\tau \mathbb{P}(\tau, \Theta) \nabla_\Theta \log\left(\mathbb{P}(\tau, \Theta)\right) R(\tau). \end{aligned} \tag{C 4}$$

The last expression represents a new expected value, which can be empirically sampled under the policy π_Θ and used as the input to the gradient descent.

In this new expression, one needs to estimate the log-prob gradient $\nabla_\Theta \log(\mathbb{P}(\tau, \Theta))$. This can be performed in the following way:

$$\begin{aligned}\nabla_\Theta \log(\mathbb{P}(\tau^{(i)}, \Theta)) &= \nabla_\Theta \log\left[\prod_t \mathbb{P}(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) \pi_\Theta(a_t^{(i)} | s_t^{(i)})\right] \\ &= \nabla_\Theta \left[\sum_t \log \mathbb{P}(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) + \sum_t \log \pi_\Theta(a_t^{(i)} | s_t^{(i)})\right] \\ &= \nabla_\Theta \sum_t \log \pi_\Theta(a_t^{(i)} | s_t^{(i)}). \end{aligned} \tag{C 5}$$

This last expression depends only on the policy, not the dynamic model. This allows effective sampling and gradient descent. In addition, one can show that this method is unbiased.

In the case of continuous control, as is performed in the present work, the ANN is used to predict the parameters of a distribution with compact support (i.e. the distribution is 0 outside of the range of admissible actions; in the present case, a Γ distribution is used, but other choices are possible), given the input o_t. The distribution obtained at each time step describes the probability distribution for the optimal action to perform at the corresponding step, following the current belief encoded by the weights of the ANN. When performing training, the action effectively taken is sampled following this distribution. This means that there is a level of exploration randomness when an action is chosen during training. The more uncertain the network is about which action to take (i.e. the wider the distribution), the more the network will try different controls. This is the source of the random exploration during training. By contrast, when the network is used in pure prediction mode, the DRL agent extracts the action with the highest probability and uses it for the action, so there is no randomness any longer in the control.

In addition to these main lines of the policy gradient algorithm that we have just described, a number of technical tricks are implemented to make the method more easily converge. In particular, a replay memory buffer (Mnih *et al.* 2013) is used to store the data empirically sampled. When training is performed, a random subset of the replay memory buffer is used. This allows the ANN to perform gradient descent

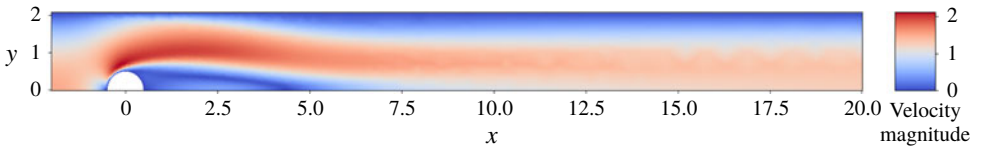

FIGURE 7. (Colour online) Illustration of the converged flow obtained around a centred half-domain, using a symmetric boundary condition at the lower boundary. The lower boundary cuts the cylinder through its equatorial plane. This results in a configuration where no vortex shedding is present, which constitutes a 'hypothetical' no-shedding baseline to which we can compare our results. The mesh is heavily refined in all the recirculation area.

on a mostly uncorrelated dataset, which yields better convergence results. In addition, the PPO method resorts to a several heuristics to improve stability. The most important one consists in gradient clipping. This makes sure that only small updates of the policy are performed at each gradient descent. The rationale behind gradient clipping is to avoid the model to overfit lucky coincidences in the training data.

In the present implementation, Tensorflow is the open-source library providing the facilities around ANN definition and the gradient descent algorithm, while Tensorforce (which builds upon Tensorflow) is the open-source library which implements the DRL algorithm.

Appendix D. Baseline simulation of a half-cylinder without vortex shedding

As vortex shedding is the process which participates in creating drag on the cylinder that is being mitigated by our active flow control, a modified baseline value of the drag without vortex shedding can be used to assess the efficiency of the control strategy. For estimating the modified baseline drag value, we perform a simulation where the cylinder is placed symmetrically at the centreline of the domain (recall that, by contrast, in the configuration used in the rest of the article, a slight offset is present), and further only the upper half of the domain is simulated (cf. figure 7), with symmetric boundary conditions enforced on the lower boundary (in particular, $v = 0$ at the lower domain boundary).

More precisely, referring to the streamwise and vertical velocity components as u and v, the boundary conditions on the lower boundary are: $v = 0$ and $(\partial u/\partial y) = 0$ for (B 4), $(\partial p/\partial y) = 0$ in (B 5) and $v = 0$ for (B 6).

As a consequence, the vortex shedding is killed and a modified baseline showing the drag behind the cylinder when the shedding is absent is obtained. This simulation is validated through a mesh refinement analysis that is distinct from the one performed with the full domain.

In this configuration, we obtain an asymptotic drag coefficient for a half-cylinder which corresponds to a virtual drag coefficient on the full cylinder in a hypothetical steady state without vortex shedding $C_{Ds} = 2.93$. This value can be compared with the drag obtained when active flow control is turned on, to estimate how efficient the control is at reducing the negative effect of vortex shedding on drag. Similarly, the asymptotic recirculation area for a half-cylinder without vortex shedding is obtained and the value can be extended to the hypothetical steady case with a full cylinder ($A_s = 2.41$).

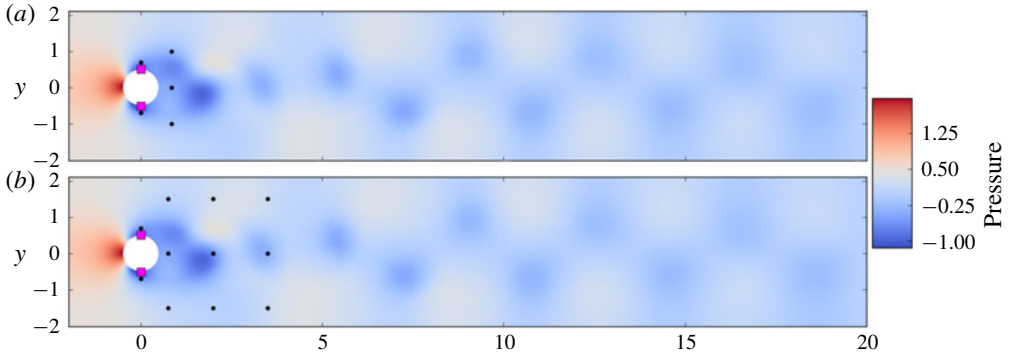

FIGURE 8. (Colour online) Unsteady non-dimensional pressure wake behind the cylinder after flow initialization without active control and position of the pressure probes in the case with five (*a*) and 11 (*b*) probes. Both the resolution and the regions of the flow on which information is provided are much reduced compared with the main body of the text. In both cases, the position of the probes is indicated by black dots, while the position of the jets is indicated by red squares.

Appendix E. Control with partial system information

All results presented in the main body of the article are obtained with a high number of velocity probes (151), which provide the network with a relatively detailed flow description. While presenting a detailed analysis of the sensitivity of the learning to the number, type, position and noise properties of the probes is outside the focus of this work, this section illustrates that the network is able to perform learning with much more partial observation.

More specifically, we performed two trainings with two modified configurations. In these configurations, either five or 11 pressure probes were located in the vicinity of the cylinder (the case with five probes), or in the vicinity of the cylinder and the near wake (the case with 11 probes). The size and structure of the network are otherwise the same as in the main body of the text. The configuration of the probes is visible in figure 8. In the case with 11 probes the network receives information about the flow in a region which includes the neighbourhood of the cylinder and the near wake. In the case with five probes the network receives only information about the dynamics in the immediate neighbourhood of the cylinder.

Results obtained by performing control (in deterministic mode, i.e. without exploration noise), after a training phase similar to what is described in the main body of the text, are presented in figure 9. As can be seen in figure 9, the networks with both five and 11 probes are able to learn some valid control strategies, though the results are a bit less good than what was obtained with 151 velocity probes (typically, mean values of C_D reach 3.03 for five probes, and 2.99 for 11 probes in the pseudo-periodic regime). This can probably be attributed to the absence of information about the configuration of the developed wake. This provides an illustration of the fact that ANNs can also be used to perform efficient control even when only partial, under-sampled flow information is available.

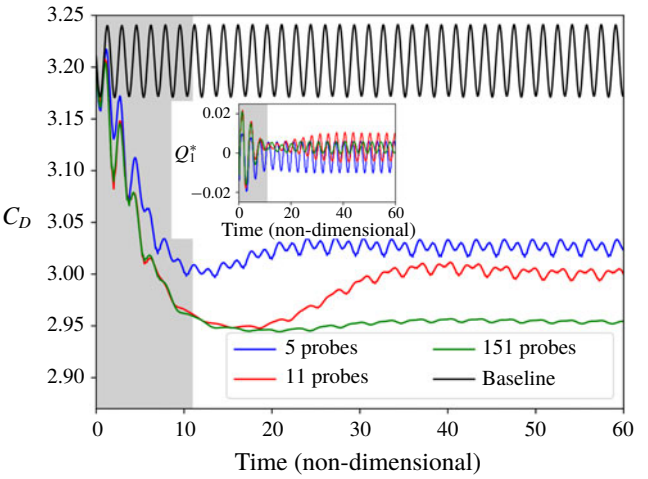

FIGURE 9. (Colour online) Time-resolved value of the drag coefficient C_D in the case without (baseline curve) and with active flow control (controlled curves, cases with five, 11, 151 probes; the last one is the same as was reported in the main body of the text), and corresponding normalized mass flow rates of the control jet 1 (Q_1^*, inset for both cases). This figure is plotted in a similar way to figure 3. As visible here, the asymptotic drag reduction in the case of a reduced input information size is a bit less good than with a full flow information. This may be due to both the absence of information about the far-wake configuration, and the lesser spatial resolution of the sampling.

REFERENCES

ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., RUDLUR, M., LOVENBERG, J., MONGA, R., MOORE, S., STEINER, B., TUCHU, P., VASUDEJON, V., WARDEN, P., WICKE, M., YU, Y. & ZHENG, X. 2016 Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, vol. 16, pp. 265–283.

BARBAGALLO, A., DERGHAM, G., SIPP, D., SCHMID, P. J. & ROBINET, J.-C. 2012 Closed-loop control of unsteadiness over a rounded backward-facing step. *J. Fluid Mech.* **703**, 326–362.

BARBAGALLO, A., SIPP, D. & SCHMID, P. J. 2009 Closed-loop control of an open cavity flow using reduced-order models. *J. Fluid Mech.* **641**, 1–50.

BRUNTON, S. L. & NOACK, B. R. 2015 Closed-loop turbulence control: progress and challenges. *Appl. Mech. Rev.* **67** (5), 050801.

DAVIS, T. A. 2004 Algorithm 832: UMFPACK v4.3 – an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.* **30** (2), 196–199.

DEAN, B. & BHUSHAN, B. 2010 Shark-skin surfaces for fluid-drag reduction in turbulent flow: a review. *Phil. Trans. R. Soc. Lond.* A **368** (1929), 4775–4806.

DUAN, Y., CHEN, X., HOUTHOOFT, R., SCHULMAN, J. & ABBEEL, P. 2016 Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338.

DURIEZ, T., BRUNTON, S. L. & NOACK, B. R. 2016 *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer.

ERDMANN, R., PÄTZOLD, A., ENGERT, M., PELTZER, I. & NITSCHE, W. 2011 On active control of laminar–turbulent transition on two-dimensional wings. *Phil. Trans. R. Soc. Lond.* A **369** (1940), 1382–1395.

FRANSSON, J. H. M., TALAMELLI, A., BRANDT, L. & COSSU, C. 2006 Delaying transition to turbulence by a passive mechanism. *Phys. Rev. Lett.* **96** (6), 064501.

GAUTIER, N., AIDER, J.-L., DURIEZ, T., NOACK, B. R., SEGOND, M. & ABEL, M. 2015 Closed-loop separation control using machine learning. *J. Fluid Mech.* **770**, 442–457.

GEUZAINE, C. & REMACLE, J.-F. 2009 Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Intl J. Numer. Meth. Engng* **79** (11), 1309–1331.

GLEZER, A. 2011 Some aspects of aerodynamic flow control using synthetic-jet actuation. *Phil. Trans. R. Soc. Lond.* A **369** (1940), 1476–1494.

GODA, K. 1979 A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *J. Comput. Phys.* **30** (1), 76–95.

GOODFELLOW, I., BENGIO, Y., COURVILLE, A. & BENGIO, Y. 2016 *Deep Learning*, vol. 1. MIT Press.

GU, S., LILLICRAP, T., SUTSKEVER, I. & LEVINE, S. 2016 Continuous deep Q-learning with model-based acceleration. In *Intl Conference on Machine Learning*, pp. 2829–2838.

GUÉNIAT, F., MATHELIN, L. & HUSSAINI, M. Y. 2016 A statistical learning strategy for closed-loop control of fluid flows. *Theor. Comput. Fluid Dyn.* **30** (6), 497–510.

HE, K., ZHANG, X., REN, S. & SUN, J. 2016 Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 770–778.

HORNIK, K., STINCHCOMBE, M. & WHITE, H. 1989 Multilayer feedforward networks are universal approximators. *Neural Networks* **2** (5), 359–366.

KOBER, J., BAGNELL, J. A. & PETERS, J. 2013 Reinforcement learning in robotics: a survey. *Intl J. Robotics Res.* **32** (11), 1238–1274.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. 2012 Imagenet classification with deep convolutional neural networks. *Adv. Neural Inform. Proc. Syst.* pp. 1097–1105.

KUTZ, J. N. 2017 Deep learning in fluid dynamics. *J. Fluid Mech.* **814**, 1–4.

LECUN, Y., BENGIO, Y. & HINTON, G. 2015 Deep learning. *Nature* **521**, 436–444.

LI, R., NOACK, B. R., CORDIER, L., BORÉE, J. & HARAMBAT, F. 2017 Drag reduction of a car model by linear genetic programming control. *Exp. Fluids* **58** (8), 103.

LILLICRAP, T. P., HUNT, J. J., PRITZEL, A., HEESS, N., EREZ, T., TASSA, Y., SILVER, D. & WIERSTRA, D. 2015 Continuous control with deep reinforcement learning. arXiv:1509.02971.

LOGG, A., MARDAL, K.-A. & WELLS, G. 2012 *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*, vol. 84. Springer.

MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D. & RIEDMILLER, M. 2013 Playing Atari with deep reinforcement learning. arXiv:1312.5602.

MNIH, V., KAVUKCUOGLU, K., SILVER, D., RUSU, A. A., VENESS, J., BELLEMARE, M. G., GRAVES, A., RIEDMILLER, M., FIDJELAND, A. K., OSTROVSKI, G., PETERSON, S., BEATIE, C., SADIH, A., ANTONOGLOU, I., KING, H., RUMANRON, D., WIERSTA, D., LEGG, S. & HASSABIS, D. 2015 Human-level control through deep reinforcement learning. *Nature* **518** (7540), 529.

PASTOOR, M., HENNING, L., NOACK, B. R., KING, R. & TADMOR, G. 2008 Feedback shear layer control for bluff body drag reduction. *J. Fluid Mech.* **608**, 161–196.

RABAULT, J., KOLAAS, J. & JENSEN, A. 2017 Performing particle image velocimetry using artificial neural networks: a proof-of-concept. *Meas. Sci. Technol.* **28** (12), 125301.

RAUBER, P. E., FADEL, S. G., FALCAO, A. X. & TELEA, A. C. 2017 Visualizing the hidden activity of artificial neural networks. *IEEE Trans. Vis. Comput. Graphics* **23** (1), 101–110.

SCHAARSCHMIDT, M., KUHNLE, A. & FRICKE, K. 2017 Tensorforce: a tensorflow library for applied reinforcement learning. https://github.com/tensorforce/tensorforce.

SCHÄFER, M., TUREK, S., DURST, F., KRAUSE, E. & RANNACHER, R. 1996 Benchmark computations of laminar flow around a cylinder. In *Flow Simulation with High-Performance Computers II* (ed. E. H. Hirschel), pp. 547–566. Springer.

SCHMIDHUBER, J. 2015 Deep learning in neural networks: an overview. *Neural Networks* **61**, 85–117.

SCHOPPA, W. & HUSSAIN, F. 1998 A large-scale control strategy for drag reduction in turbulent boundary layers. *Phys. Fluids* **10** (5), 1049–1051.

SCHULMAN, J., LEVINE, S., MORITZ, P., JORDAN, M. I. & ABBEEL, P. 2015 Trust region policy optimization. CoRR abs/1502.05477, arXiv:1502.05477.

SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A. & KLIMOV, O. 2017 Proximal policy optimization algorithms. arXiv:1707.06347.

SHAHINFAR, S., SATTARZADEH, S. S., FRANSSON, J. H. & TALAMELLI, A. 2012 Revival of classical vortex generators now for transition delay. *Phys. Rev. Lett.* **109** (7), 074501.

SIEGELMANN, H. T. & SONTAG, E. D. 1995 On the computational power of neural nets. *J. Comput. Syst. Sci.* **50** (1), 132–150.

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

VALEN-SENDSTAD, K., LOGG, A., MARDAL, K.-A., NARAYANAN, H. & MORTENSEN, M. 2012 A comparison of finite element schemes for the incompressible Navier–Stokes equations. In *Automated Solution of Differential Equations by the Finite Element Method*, pp. 399–420. Springer.

VERMA, S., NOVATI, G. & KOUMOUTSAKOS, P. 2018 Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proc. Natl Acad. Sci. USA*; http://www.pnas.org/content/early/2018/05/16/1800923115.full.pdf.

VERNET, J., ÖRLÜ, R., ALFREDSSON, P. H., ELOFSSON, P. & SCANIA, A. B. 2014 Flow separation delay on trucks a-pillars by means of dielectric barrier discharge actuation. In *First International Conference in Numerical and Experimental Aerodynamics of Road Vehicles and Trains (Aerovehicles 1), Bordeaux, France*, pp. 1–2.

WANG, Z., XIAO, D., FANG, F., GOVINDAN, R., PAIN, C. C. & GUO, Y. 2018 Model identification of reduced order fluid dynamics systems using deep learning. *Intl J. Numer. Meth. Fluids* **86** (4), 255–268.