The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

TOWARDS ACTIVE FLOW CONTROL STRATEGIES THROUGH DEEP REINFORCEMENT LEARNING

R. MONTALÀ[1], B. FONT[2], P. SUÁREZ[3], J. RABAULT[4], O. LEHMKUHL[5], R. VINUESA[3] AND I. RODRIGUEZ[1]

[1] Turbulence and Aerodynamics Research Group (TUAREG)
Universitat Politècnica de Catalunya (UPC)
Terrassa, Spain

[2] Faculty of Mechanical Engineering
Delft University of Technology (TU Delft)
Delft, Netherlands

[3] FLOW, Engineering Mechanics
KTH Royal Institute of Technology
Stockholm, Sweden

[4] Independent Researcher
Oslo, Norway

[5] Computer Applications in Science and Engineering (CASE)
Barcelona Supercomputing Center (BSC)
Barcelona, Spain

Key words: Active Flow Control, Separation Control, Aerodynamics, Deep Reinforcement Learning

Summary. This paper presents a deep reinforcement learning (DRL) framework for active flow control (AFC) to reduce drag in aerodynamic bodies. Tested on a 3D cylinder at $Re = 100$, the DRL approach achieved a 9.32% drag reduction and a 78.4% decrease in lift oscillations by learning advanced actuation strategies. The methodology integrates a CFD solver with a DRL model using an in-memory database for efficient communication between the two instances, making it scalable to more complex flows and higher Reynolds numbers.

1 INTRODUCTION

In light of the current climate crisis, the transportation industry faces significant challenges in reducing fossil fuel emissions to mitigate the adverse effects of climate change. In particular, the aviation sector already accounts for about 3% of the global CO_2 emissions [1]. Reducing these emissions may be possible by exploring innovative methods to decrease drag. In this regard, active flow control (AFC) has demonstrated promising results in controlling the flow around wings [2, 3]. However, conventional AFC methods that rely on fixed actuation laws are inherently limited as they can only target specific frequencies within the full spectrum of turbulence scales, leading to a maximum level of drag reduction. Moreover, their fixed-loop nature prevents them from adjusting to instantaneous flow conditions, restricting their

applications to dynamical systems that are not continuously evolving. Tunning the actuation parameters can also be challenging and may require extensive trial-and-error, especially in highly turbulent flows.

The emergence of machine learning (ML), coupled with advances in computational power, has revolutionized the state of the art in scientific computing. In the context of AFC, the use of deep reinforcement learning (DRL) appears particularly well-suited for discovering more complex actuation strategies and has already shown its potential in the field of flow control [4]. As far as the author is concerned, the first successful application of DRL to AFC was the work by Rabault et al. [5], who considered a two-dimensional cylinder at a Reynolds number of $Re = 100$ [5], achieving a drag reduction of approximately 8%. The Reynolds number is defined using the fluid density ρ, the inflow velocity U_∞, the cylinder diameter D and the fluid dynamic viscosity μ as $Re = \rho U_\infty D/\mu$. Based on their previous results, Rabault and Kuhnle [6] extended their study by implementing a multi-environment approach to explore the capabilities of parallelization in DRL and hence accelerate the training, making the application of DRL affordable for more sophisticated fluid mechanics problems. Building on this data collection parallelization approach, subsequent studies extended the Reynolds number towards higher values. This is the case of Tang et al. [7], who investigated the regimes of $Re = 100$, 200, 300, and 400 with drag reductions of 5.7%, 21.6%, 32.7%, and 38.7%, respectively; and also Varela et al. [8], who extended the Reynolds number up to $Re = 2,000$ and achieved a reduction of 17.7%. The latter authors further applied their DRL set-up to control the wake of a three-dimensional cylinder for the first time, targeting $Re = 100$, 200, 300 and 400, and reducing the drag up to 8.0%, 17.2%, 15.3% and 15.1%, respectively [9]. Other research efforts also combined DRL and AFC to reduce skin friction in wall-bounded flows at $Re_\tau = 180$ [10] or to control the two-dimensional Rayleigh–Bénard convection [12]. However, all these studies were confined to canonical problems at low Reynolds numbers, indicating that this methodology is still in its early stages of development.

ML libraries are generally implemented in high-level programming languages, such as Python, while high-performance physics solvers typically rely on low-level languages like C++ or Fortran. This gives rise to the "two-language" problem, where efficiently linking ML models with physical simulations becomes a challenge. In DRL applications for fluid dynamics, which require solving a huge amount of degrees of freedom, the most time-consuming part is the collection of experience data. Therefore, using a fast solver that leverages GPU accelerators, now present in modern HPC clusters, is essential for tackling complex fluid environments, such as those involving intricate geometries or high Reynolds numbers. This study presents an AFC-DRL framework that effectively addresses the "two-language" problem with minimal overhead and employs a GPU-enabled code for computational fluid dynamics (CFD) simulations. This approach ensures rapid trajectory collection for DRL training, making the framework feasible for addressing more complex problems. This framework was first implemented and tested to mitigate the separation bubble in a boundary layer at $Re_\tau = 180$ [11]. In the present work, we replicate the case conducted by Suárez et al. [9], extending the validation of the framework to a three-dimensional cylinder at $Re = 100$ and evaluating its capabilities in controlling flows around bluff bodies. This represents a significant step towards applying DRL to more realistic industrial scenarios.

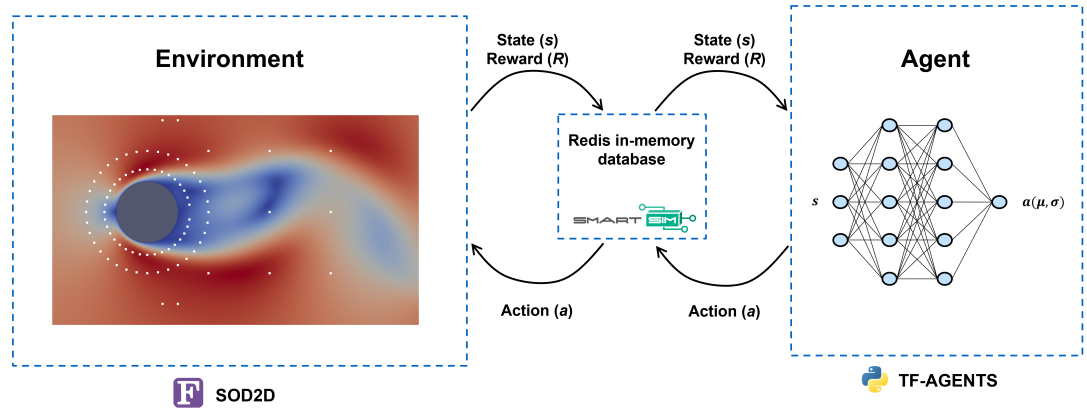


Figure 1: DRL-CFD setup

2 METHODOLOGY

2.1 DRL set-up

In DRL, two main entities can be identified: (i) The environment and (ii) the agent. In the current framework, the environment is the CFD simulation that predicts how the flow evolves with a given actuation and the agent is a neuronal network (NN) that predicts the probability distribution of possible action given the state of the environment.

For the environment, the CFD solver called SOD2D [13] is employed. This is a Spectral Element Method (SEM) and GPU-enabled code developed at the Barcelona Supercomputing Center (BSC). The incompressible Navier-Stokes equations are solved, as shown in Eq. 1 and 2.

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla p = 0 \tag{2}$$

On the other hand, the agent is built using the Python library TF-Agents [14]. To solve the so-called "two-language" problem, a Redis in-memory database is used, which is managed through the library SmartSim [15], allowing the communication between the CFD model (Fortran) and the DRL agent (Python) with minimal overhead. This workflow was initially proposed in the Relexi project [16, 17], and adapted to link with the SOD2D CFD solver in [11]. The framework is schematized in Figure 1.

The whole idea of the DRL is that the agent receives the state, e.g., some probes located in the domain, and this returns an action that will be applied back into the environment, e.g., the mass flow rate of the jet. However, to correctly decide the best actuation, the DRL agent needs to be previously trained. To do so, during the training, the agent also receives a reward, representing the magnitude that aims to be optimized. Then, the environment and the agent engage in a trial-and-error process structured as episodes. An episode, in this context, denotes a simulation period wherein the CFD solver and the DRL agent exchange information, including states, actions, and rewards. Following the completion of an episode, this data is used to refine the agent through a training step. The proximal policy optimization (PPO) algorithm [18] is used in this case.

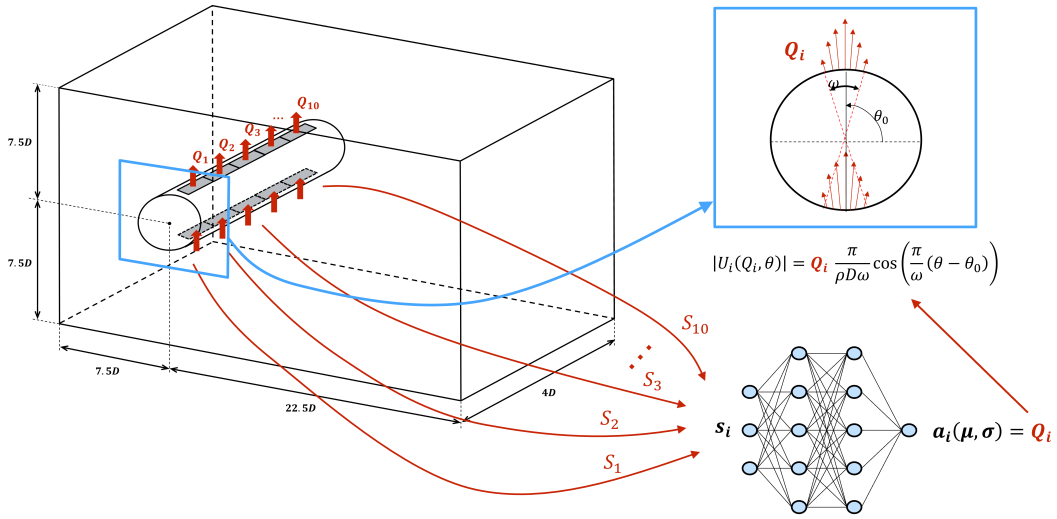


Figure 2: Case configuration

2.2 Case Configuration

The flow past a cylinder at $Re = 100$ is simulated using SOD2D. The three-dimensional domain extends $L_x = 30$, $L_y = 15D$ and $L_z = 4D$ in the streamwise, crosswise and spanwise directions, respectively. The cylinder is located approximately in the middle of the domain, i.e., at $(x, y) = (7.5D, 7.5D)$, and is infinitely long along the spanwise direction. Hence, periodic boundary conditions are applied in this direction. At the inlet, a constant freestream velocity U_∞ is imposed. The slip condition is enforced at the top and bottom surfaces, while zero-gradient conditions with constant pressure are applied at the outlet.

On the cylinder walls, the no-slip condition is applied. When AFC is activated, two sets of $n_{jet} = 10$ actuators are distributed along the spanwise direction of the cylinder. Each set contains two jets, one on the top surface of the cylinder ($\theta_{top} = 90°$) and the other on the bottom surface ($\theta_{bot} = 270°$). The two actuators are forced to have the opposite mass flow rate, i.e. $Q_{top} = -Q_{bot}$, so that the mass is conserved instantaneously. In the xy-plane, the actuators have a width of $\omega = 10°$, while in the spanwise direction they extend a width of $L_{jet} = 0.4D$.

Each set of two actuators represents a pseudo-environment. This is schematized in Figure 2. Thus, the whole domain is divided into ten pseudo-environments with a width of L_{jet}. Moreover, four different CFD simulations are run in parallel. This allows the DRL agent to collect several experiences in parallel, as was previously done in Varela et al. [8], speeding up the training. In total, 40 trajectories (10 pseudo-environments x 4 CFD simulations) are collected after each action (batch size). The training is selected to last 50 episodes. The duration of an episode $T_{episode}$ includes six vortex sheddings and in each episode, a total of 120 actions are applied, i.e., $T_{action} = T_{episode}/120$. For the state, each pseudo-environment contains in its z-middle location a slice of 85 witness points distributed around the cylinder walls. The approximate locations of these witness points are illustrated in Figure 1. However, the size of the NN input layer is 85 x 3 = 255, as it takes also into account the state of the two neighbouring pseudo-environments. The NN consists of two hidden layers, each containing 512 neurons.

As the output, the DRL agent is responsible for predicting the optimal mass flow rate Q

to maximize the reward. This is then used to compute the velocity profile that is applied as the boundary condition along the jet surfaces, as described in Eq. 3; with the velocity perpendicular to the cylinder wall. The minimum and maximum allowable mass flow rates are set to $Q_{min,max} = [-0.176, 0.176]$.

$$[u_{jet}, v_{jet}, w_{jet}] = Q\frac{\pi}{\rho D\omega}cos(\frac{\pi}{\omega}(\theta - \theta_0))[cos\theta, sin\theta, 0] \tag{3}$$

The reward to train the model is described in Eq. 4; the first part rewarding the reduction of the drag coefficient $C_d = d/(1/2\rho U_\infty^2 S)$ with respect to the baseline scenario $C_{d,b}$, and the second penalizing the increase of the lift $C_l = l/(1/2\rho U_\infty^2 S)$, with α being a weighting factor to adjust the importance of each term. In this case, the reference surface S is defined in terms of the cylinder diameter D and the spanwise length of the jet (or pseudo-environment) L_{jet} as $S = L_{jet}D$, and the drag d and lift l forces correspond to the resulting components of the aerodynamic force in the streamwise and cross-stream directions relative to the freestream U_∞. Finally, the reward applied to the model is computed accounting for the rewards obtained in the other pseudo-environments as shown in Eq. 5, with β being a weighting factor to adjust the importance of the local reward r_i versus the mean rewards of all pseudo-environments $\sum_{j=1}^{n_{jets}} r_j$. The weighting factors in Eq. 4 and Eq. 5 are set to $\alpha = 0.3$ and $\beta = 0.8$, respectively.

$$r_i = (C_{d,b} - C_d) - \alpha|C_l| \tag{4}$$

$$R_i = \beta r_i + (1 - \beta)/n_{jets} \sum_{j=1}^{n_{jets}} r_j \tag{5}$$

Note that, to validate the results and compare with previous studies, the set-up presented here mainly mimics the configuration used by Suárez et al. [9].

3 RESULTS AND DISCUSSION

3.1 Training mode

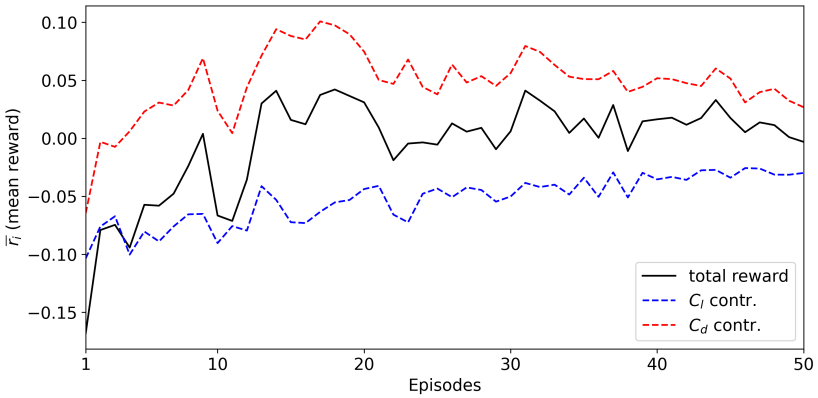
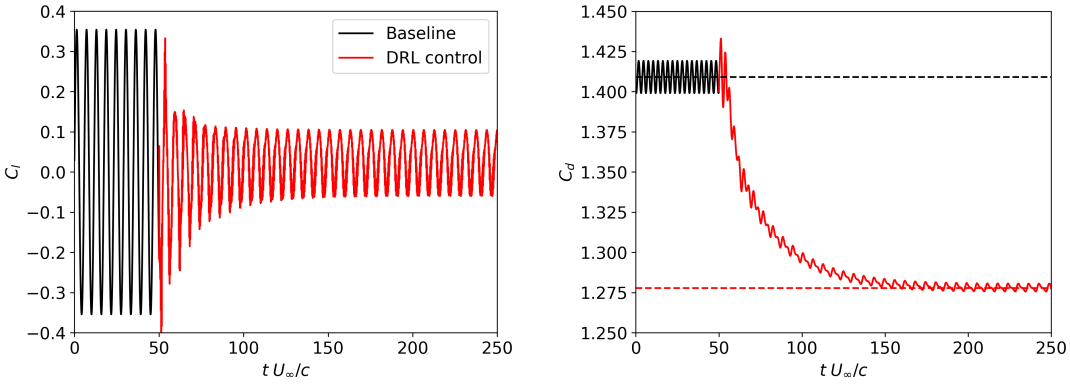


Figure 3: Evolution of the averaged reward $\overline{r_i}$ across the 40 pseudo-environments during the training

During the training, the DRL agent adjusts the weights and biases of the NN to maximize the reward described in Eq. 4 and Eq. 5. Thus, as can be observed from these equations, the DRL control aims at reducing the drag coefficient and also reducing the lift oscillations. The curves showing how the reward evolves along the different training episodes are depicted in Figure 3, where the contribution of the lift and drag terms are plotted separately as well. This figure shows the mean reward $\overline{r_i}$ described in Eq. 4 among the forty pseudo-environments. It can be observed that the agent successfully learns a strategy that reduces the drag coefficient and also the oscillations of the lift coefficient. At the beginning of the training, a considerable drag reduction is achieved. As the episodes continue, the lift contribution to the reward progressively increases, while the drag contribution slightly decreases. Overall, the total reward increases. To adjust which is the final objective of the training, e.g., only minimize the drag, the α factor in Eq. 4 could be tuned. To evaluate the learnt actuation law, the model has to be run in deterministic mode and hence apply the learnt policy without any further exploration.

3.2 Deterministic mode



(a) (b)

Figure 4: Lift C_l (a) and drag C_d (b) coefficients before and after the DRL control is applied

Once the DRL agent is run in deterministic mode, the obtained lift and drag coefficients are shown in Figure 4. Note that the actuation is activated at $tU_\infty/D = 50$, which allows to compare the baseline case ($tU_\infty/D < 50$) against the DRL control ($tU_\infty/D > 50$). After the actuation is applied, the drag coefficient starts to decrease considerably, as well as the amplitude of the lift coefficient. The simulation is run until the control converges into a periodic behaviour. In Table 1, the mean lift and drag coefficients, with the pressure and viscous contributions, are reported. This table also shows the Strouhal number St_{C_l} and the standard deviation σ_{C_l} of the lift coefficient signal. The results indicate that the DRL control achieves a 9.32% reduction in the drag coefficient and a 78.4% decrease in the standard deviation of the lift coefficient. This drag reduction is in close agreement with the 9.4% reduction reported by Suárez et al. [9]. It is important to note that all statistics reported here are computed after the control has reached a statistically steady state.

The applied actuation by the DRL agent can be visualized in Figure 5. After the initial

Table 1: Lift and drag coefficients statistics

	C_l	σ_{C_l}	St_{C_l}	C_d	$C_{d,press.}$	$C_{d,visc.}$
Baseline	0.012	0.250	0.170	1.409	1.051	0.358
DRL control	0.029	0.054	0.161	1.278	0.946	0.331

transient, the DRL actuation converges to a periodic signal with a frequency of $St = 0.161$. This is approximately the 95% of the baseline vortex-shedding frequency, delaying the vortex-shedding of the controlled scenario (see St_{C_l} in Table 1). The minimum and maximum actions applied by the DRL agent are $Q \in [-0.01, 0.01]$. Nevertheless, the predicted action is not a perfect sinusoidal signal and, at the beginning of each period, a little hump is detected. This is also manifested in the spectral domain, where a second harmonic at $St = 0.320$ is obtained. Compared to a classical periodic control, where a simple sinusoidal actuation would be considered, in this case the agent has learned a double-lobed signal, hence demonstrating that DRL can help to learn more complex control actions that allow to push the limits of drag reduction using AFC. It is worth pointing out that, despite being a three-dimensional domain, the wake of the cylinder at this low Re is essentially two-dimensional. Thus, the applied Q in all the pseudo-domains is exactly the same, as well as the obtained C_l and C_d.

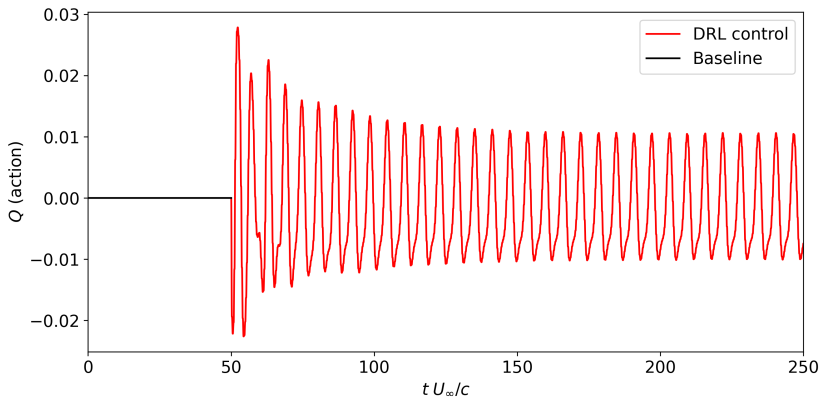
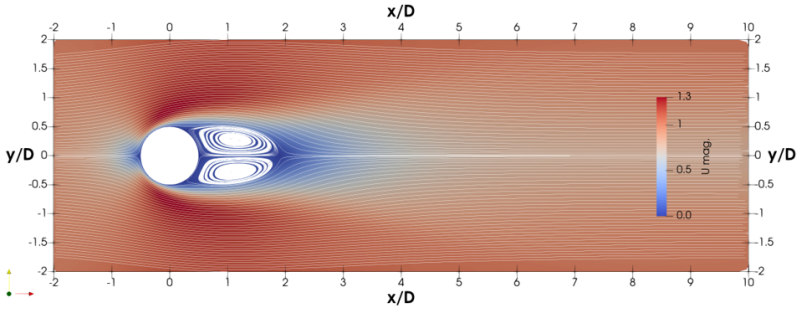


Figure 5: Applied mass flow rate Q

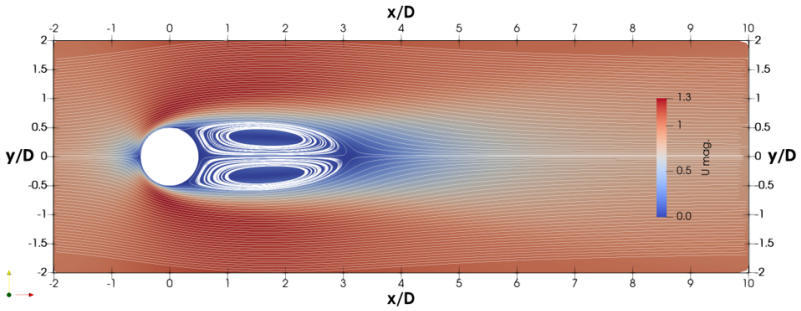
As observed in Figure 6, the DRL control increases the streamwise length of the wake recirculation bubble, reducing the intensity of the shear layers developed on the cylinder top and bottom surfaces and hence reducing the vortex-shedding frequencies. As depicted in Figure 7, this increases the pressure coefficient C_p on the rear part of the cylinder ($\theta > 50°$), finally leading to the drag and lift reductions described before.

4 CONCLUSIONS

This study extends the application of the DRL framework initially implemented in Font et al. [11], where an in-memory database enables the linking of the CFD solver SOD2D with a Python-based DRL model with minimal overhead. In this work, we apply a DRL control



(a)



(b)

Figure 6: Velocity streamlines of the baseline (a) and controlled (b) cases

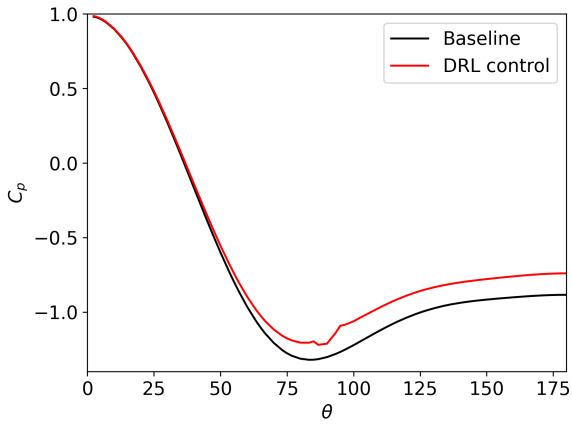


Figure 7: Pressure coefficient C_p along the cylinder wall

strategy to the flow around a three-dimensional cylinder at $Re = 100$, following the set-up by Suárez et al. [9]. This previous work serves as a benchmark to validate the current framework.

According to the results, the methodology demonstrates its capability to control the flow

around a bluff body. The cylinder drag is reduced by approximately 9.32% and the lift oscillations by 78.4%. This is consistent with the drag reduction reported by Suárez et al. [9]. This improvement is accomplished by controlling the intensity of the shear layers formed around the cylinder top and bottom regions, thereby delaying the onset of instabilities.

Despite the relative simplicity of this case, the results highlight the potential of DRL to find more advanced actuation strategies compared to traditional periodic controls. Additionally, integrating the DRL model with the GPU-accelerated SOD2D CFD code allows for the rapid collection of a larger volume of experiences, making the framework scalable and feasible for more challenging cases involving higher Reynolds numbers and/or complex geometries.

REFERENCES

[1] Liu, Z., Deng, Z., Davis, S. and Ciais, P. "Monitoring global carbon emissions in 2022". *Nat. Rev. Earth Environ.* (2023) **4**:205–206. https://doi.org/10.1038/s43017-023-00406-z

[2] Rodriguez, I., Lehmkuhl, O. and Borrell, R. "Effects of the actuation on the boundary layer of an airfoil at Reynolds number Re = 60,000". *Flow Turbul. Combust.* (2020) **105**:607–626. https://doi.org/10.1007/s10494-020-00160-y

[3] Atzori, M., Vinuesa, R., Stroh, A., Gatti, D., Frohnapfel, B. and Schlatter, P. "Uniform blowing and suction applied to nonuniform adverse-pressure-gradient wing boundary layers". *Phys. Rev. Fluids* (2021) **6**:113904. https://doi.org/10.1103/PhysRevFluids.6.113904

[4] Garnier, P., Viquerat, J., Rabault, J., Larcher, A., Kuhnle, A. and Hachem, E. "A review on deep reinforcement learning for fluid mechanics". *Comput. Fluids* (2021) **225**:104973. https://doi.org/10.1016/j.compfluid.2021.104973

[5] Rabault, J., Kuchta, M., Jensen, A., Réglade, U. and Cerardi, N. "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control". *J. Fluids Mech.* (2019) **865**:281–302. https://doi.org/10.1017/jfm.2019.62

[6] Rabault, J. and Kuhnle, A. "Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach". *Phys. Fluids* (2019) **31**:094105. https://doi.org/10.1063/1.5116415

[7] Tang, H., Rabault, J., Kunhle, A., Wang, Y. and Wang, T. "Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning". *Phys. Fluids* (2020) **32**:053605. https://doi.org/10.1063/5.0006492

[8] Varela, P., Suárez, P., Alcántara-Ávila, F., Miró, A., Rabault, J., Font, B., García-Cuevas, L. M., Lehmkuhl, O. and Vinuesa, R. "Deep reinforcement learning for flow control exploits different physics for increasing Reynolds number regimes". *Actuators* (2022) **11**:359. https://doi.org/10.3390/act11120359

[9] Suárez, P., Alcántara-Ávila, F., Miró, A., Rabault, J., Font, B., Lehmkuhl, O. and Vinuesa, R. "Active flow control for three-dimensional cylinders through deep reinforcement learning". In: *14th International ERCOFTAC Symposium on Engineering, Turbulence, Modelling and Measurements* (2023)

[10] Guastioni, L., Rabault, J., Schlatter, P., Azizpour, H. and Vinuesa, R. "Deep reinforcement learning for turbulent drag reduction in channel flows". *Eur. Phys. J. E* (2023) **46**. https://doi.org/10.1140/epje/s10189-023-00285-8

[11] Font, B., Alcántara-Ávila, F., Rabault, J., Vinuesa, R. and Lehmkuhl, O. "Active flow control of a turbulent separation bubble through deep reinforcement learning". *J. Phys. Conf. Ser.* (2024) **2753**: 012022. https://doi.org/10.1088/1742-6596/2753/1/012022

[12] Vignon, C., Rabault, J., Vasanth, J., Alcántara-Ávila, F., Mortensen, M. and Vinuesa, R. "Effective control of two-dimensional Rayleigh–Bénard convection: Invariant multi-agent reinforcement learning is all you need". *Phys. Fluids* (2023) **35**:065146. https://doi.org/10.1063/5.0153181

[13] Gasparino, L., Spiga, F. and Lehmkuhl, O. "SOD2D: A GPU-enabled spectral finite elements method for compressible scale-resolving simulations". *Comput. Phys. Comun.* (2024) **297**:109067. https://doi.org/10.1016/j.cpc.2023.109067

[14] Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V. and Brevdo, E. "TF-Agents: A library for Reinforcement Learning in TensorFlow". (2018). https://github.com/tensorflow/agents

[15] Partee, S., Ellis, M., Rigazzi, A., Shao, A. E., Bachman, S., Marques, G. and Robbins, B. "Using machine learning at scale in HPC simulations with SmartSim: An application to ocean climate modeling". *J. Comput. Sci.* (2022) **62**:101707. https://doi.org/10.5281/zenodo.4682270

[16] Kurz, M., Offenhäuser, P., Viola, D., Resch, M. and Beck, A. "Relexi - A scalable open source reinforcement learning framework for high-performance computing". *Software Impacts* (2022) **14**:100422. http://doi.org/10.1016/j.simpa.2022.100422

[17] Kurz, M., Offenhäuser, P. and Beck, A. "Deep reinforcement learning for turbulence modelling in large eddy simulations". *Int. J. Heat Fluid Flow* (2023) **99**:109094. https://doi.org/10.1016/j.ijheatfluidflow.2022.109094

[18] Schulman, F., Wolski, P., Dhariwal, A., Radford, A. and Klimov, O. "Proximal policy optimization algorithms". (2017). https://doi.org/10.48550/arXiv.1707.06347