Physics of Fluids

RESEARCH ARTICLE | JUNE 30 2023

Effective control of two-dimensional Rayleigh–Bénard convection: Invariant multi-agent reinforcement learning is all you need

Colin Vignon ; Jean Rabault ; Joel Vasanth ; Francisco Alcántara-Ávila ; Mikael Mortensen ; Ricardo Vinuesa

Physics of Fluids 35, 065146 (2023)
https://doi.org/10.1063/5.0153181

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

Effective control of two-dimensional Rayleigh–Bénard convection: Invariant multi-agent reinforcement learning is all you need

Cite as: Phys. Fluids **35**, 065146 (2023); doi: 10.1063/5.0153181
Submitted: 5 April 2023 · Accepted: 9 June 2023 ·
Published Online: 30 June 2023

Colin Vignon,[1,2] Jean Rabault,[3] Joel Vasanth,[1] Francisco Alcántara-Ávila,[1] Mikael Mortensen,[4] and Ricardo Vinuesa[1,a)]

AFFILIATIONS

[1]FLOW, Engineering Mechanics, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
[2]Mines de Paris, Universite PSL, 75005 Paris, France
[3]IT Department, Norwegian Meteorological Institute, Postboks 43, 0313 Oslo, Norway
[4]Department of Mathematics, University of Oslo, 0316 Oslo, Norway

[a)]**Author to whom correspondence should be addressed:** rvinuesa@mech.kth.se

ABSTRACT

Rayleigh–Bénard convection (RBC) is a recurrent phenomenon in a number of industrial and geoscience flows and a well-studied system from a fundamental fluid-mechanics viewpoint. In the present work, we conduct numerical simulations to apply deep reinforcement learning (DRL) for controlling two-dimensional RBC using sensor-based feedback control. We show that effective RBC control can be obtained by leveraging invariant multi-agent reinforcement learning (MARL), which takes advantage of the locality and translational invariance inherent to RBC flows inside wide channels. MARL applied to RBC allows for an increase in the number of control segments without encountering the curse of dimensionality that would result from a naive increase in the DRL action-size dimension. This is made possible by the MARL ability for re-using the knowledge generated in different parts of the RBC domain. MARL is able to discover an advanced control strategy that destabilizes the spontaneous RBC double-cell pattern, changes the topology of RBC by coalescing adjacent convection cells, and actively controls the resulting coalesced cell to bring it to a new stable configuration. This modified flow configuration results in reduced convective heat transfer, which is beneficial in a number of industrial processes. We additionally draw comparisons with a conventional single-agent reinforcement learning (SARL) setup and report that in the same number of episodes, SARL is not able to learn an effective policy to control the cells. Thus, our work both shows the potential of MARL for controlling large RBC systems and demonstrates the possibility for DRL to discover strategies that move the RBC configuration between different topological configurations, yielding desirable heat-transfer characteristics.

I. INTRODUCTION

Deep reinforcement learning (DRL) has recently emerged as a promising methodology for performing active flow control[1] (AFC). DRL is a subclass of the much more general field of RL, where deep artificial neural networks are used to parameterize the value function in Q-learning methods[2] or serve to parameterize the policy or the mapping from state to actions in policy gradient methods.[3,4] The DRL "controller," which includes the control law to be applied on the system, and algorithms around it to allow tuning of the control law and query of the control law, is called the DRL agent. The general principle of the (D)RL paradigm[5] is based on considering the system to control as a black box (the "plant" in classical control theory or the "environment" in DRL terms) and to interact with it through just three channels of communication. At a given time step t, (i) the DRL agent can query the environment for a (possibly noisy, stochastic) state estimate s_t; (ii) the DRL agent can provide to the environment a control instruction called the action in DRL terms, a_t; and (iii) the environment can provide back a reward r_t that indicates the quality of the current state of the system. Based on these three channels of interaction, the DRL paradigm aims at defining effective algorithms to learn

control strategies that maximize the combined expected rewards at all future times from the given state at time t, also known as the expected return $G_t = \sum_{k=0}^{\infty} r_{t+k+1}\gamma^k$, where γ is the discount factor, typically $\gamma \approx 0.95 - 0.99$. In order to make the computation of R_t tractable, DRL interaction with the environment takes place on an episode basis during training. An episode is a simulation with a given time extent (typically covering a few timescales of the slowest dynamics of the system) and corresponds to one individual trajectory of the system in its phase space. The optimization of the control applied by the DRL agent is performed through trial-and-error learning and direct online interaction with the environment, a process that involves stochastic exploration of the environment properties (referred to as the "training mode" of the DRL agent). This allows to explore and map the behavior of the environment and to determine the best trajectory that the environment should be actuated to follow in the phase space to optimize the reward. A number of algorithms have been developed to perform this optimization, the most famous belonging to either (i) the Q-learning paradigm[6,7] based on the Bellman equation; (ii) the policy-gradient method,[8] which relies on a direct estimate of the sensitivity of the reward to the policy parameters; and (iii) a combination of both paradigms, such as the actor-critic proximal-policy optimization (PPO) method,[9] which combines their respective strengths. Once the trial-and-error learning process has converged and the DRL agent is fully trained, the trial-and-error exploration component can be turned off, so that a deterministic controller is obtained (corresponding to the "deterministic mode" of the DRL agent).

This DRL framework is very attractive for the control of complex, high-dimensional, non-linear systems such as those obtained in AFC. Indeed, DRL does not make any assumptions on the system to control; actual DRL algorithms have been increasingly refined in the course of the last 10 years and are well adapted to solve non-linear, high-dimensional control problems from games,[7,10,11] industrial control problems,[12] or classical physical systems such as those present in fundamental fluid mechanics.[13,14] Compared with traditional approaches, such as the adjoint method,[15] DRL does not need direct insight into the full state of the system and its governing equations. Moreover, while DRL training is expensive in terms of the number of interactions with the environment needed for the trial-and-error learning to take place, once the DRL agent is trained, predicting the next action to take given the current state is very cheap computationally. These aspects make DRL an attractive method for real-world control applications, including AFC.

Over the last 5 years, DRL for AFC has been applied to increasingly complex cases. Starting from simple 2D cylinder cases[16] at low Reynolds number ($Re = UL/\nu$, which measures the relative importance of inertia and viscosity, where U is the characteristic velocity scale of the system, L its characteristic length scale, and ν the kinematic viscosity), DRL has further been used for investigating a number of different configurations involving 2D cylinders,[17–20] looking into increasingly complex stochastic wake control problems as Re is increased,[21,22] and recently, considering the control of 3D channel flows.[13,14] DRL has also been applied to a number of other AFC cases, from 2D Rayleigh–Bénard convection in a small-size channel,[23] to instabilities developing in thin fluid films,[24] engineering cases related to wings,[25,26] and other practical problems.[27–29] DRL has also been combined with flow-stability analysis to guide the placement of observation probes[20] for effective control. In this case, a stability analysis was used to reveal the structural sensitivity that determines the location of the origin of the instability, i.e., the "wavemaker" region. The structural sensitivity can also be computed using data-driven techniques.[30] Positioning the observation probes in the region of the wavemaker has shown to achieve the best possible learning curve with the reward maximization then taking place in the least number of episodes, compared with other configurations of observation probes.[20] While these studies have mostly considered numerical test cases, demonstrations of the applicability of DRL for real-world AFC have been presented in test experiments.[31]

Moreover, a number of peripheral challenges associated with discovering efficient flow control strategies have been solved over the last few years. Rabault and Kuhnle[32] demonstrated how the multi-environment paradigm can be leveraged to obtain speedups in training speed proportional to the number of environments used. In particular, they showed that DRL training speed is directly proportional to the number of (state, action, and reward) triplets generated per wall clock time unit, and that this is in turn directly proportional to the number of CFD simulations used in parallel to train the DRL algorithm and feed it with data streams. Belus *et al.*[24] have demonstrated that the curse of dimensionality on the control dimension can be effectively solved by leveraging a multi-agent reinforcement-learning (MARL) approach (although the authors did not refer to MARL under this name at the time), in cases where some general properties of the system are invariant in space. In particular, MARL was demonstrated to be an effective way to control systems containing several generally similar structures that are far enough to be mostly independent from each other, by sharing the knowledge acquired among them. This, in turn, allows to reduce the combinatorial cost and alleviate the curse of dimensionality associated with the exploration of large action spaces.

Following these advances, DRL can now be used to control different physics as the effect of non-linearity increases,[22] to perform effective control over a range of flow conditions in simple AFC test cases,[33] or to control complex three-dimensional (3D) turbulent channel flows.[14] DRL has also been demonstrated in a number of test situations involving chaotic flows, such as the one arising from the one-dimensional (1D) Kuramoto–Sivashinsky system of equations.[34,35] The placement and optimization of the sensor inputs used to drive DRL algorithms has also been investigated, see Refs. 36 and 37. As the literature is growing fast, we refer the reader curious of a more detailed overview of DRL applications to fluid mechanics in general and AFC in particular to one of the several reviews that have recently been provided on the topic.[38–40] Similarly, DRL is becoming an increasingly common topic in AFC, and the DRL methodology *per se* has now been discussed in a number of AFC studies, so we refer the reader curious of specific DRL algorithmic details and refinements to the corresponding body of literature.[41] As a consequence of this increasing popularity, DRL frameworks are starting to emerge, both in the DRL[42–44] and in the fluid-mechanics communities,[45–47] a fact that now provides frameworks for effectively coupling DRL with well-established computational-fluid-dynamics (CFD) tools. This makes it increasingly easier to apply DRL to AFC problems.

At this point, we want to emphasize that, while DRL is a promising method given that it has provided state-of-the-art results in a variety of fields as previously discussed, it is not the only method that can perform AFC by considering the flow to control as a black box to be optimized. Machine-learning control (MLC[48–50]) has, indeed, been a

topic of discussion for quite a long time, and the DRL method is, in this aspect, a newcomer in this field. Traditionally, methods such as genetic programing (GP[51,52]) or Bayesian optimization (BO[53]) have also been applied with great success to a variety of practical problems[54–57] and can be competitive with DRL in a number of applications.[58] However, we focus our efforts on DRL in the present study, as DRL is showing great flexibility in constraining the admissible structure of the control law (see, e.g., the discussions around invariants and MARL in Ref. 24). Moreover, the DRL methodology is promising thanks to its ability to use local (in space, but also in time) information to perform optimization. Compared to, e.g., a direct application of GP or BO, which only optimize based on the mean efficiency of a control law over one simulation containing several typical physical time scales of the underlying dynamics, and which do not exploit information about the high-frequency quality of the control law, DRL gets a reward at each interaction step. Thus, DRL can learn to both reproduce positive sequences of actions and avoid negative sequences of actions happening within a single episode. As a consequence, DRL is able to generate information about the quality of the control law at a higher time granularity than the episode itself and generates a larger volume of usable reward information than would be provided by monitoring for example the averaged reward over an episode.

In the present work, we use DRL techniques to discover effective strategies to control the instabilities and the heat flux in a two-dimensional (2D) Rayleigh–Bénard convection (RBC) problem. The RBC phenomenon is intrinsic to many industrial applications and can be found in a wide range of natural phenomena.[59] The canonical initial condition in a RBC problem is a fluid at rest that is being heated from a lower wall and/or cooled from an upper wall (see Fig. 1). Because of the induced temperature gradients and due to the fluid-buoyancy effects, natural convection occurs. The dynamical state of the RBC system can be characterized by a non-dimensional parameter, the Rayleigh number Ra, which is a ratio of the timescale of heat transport by diffusion or conduction to the timescale of heat transport by buoyancy-induced convection. Beyond a critical value of $Ra = Ra_c$ when convective effects dominate, RB instability occurs through a supercritical bifurcation from an initial quiescent state, to time-dependent cellular motion.[60]

The convective cells cause a non-spatially uniform heat transfer along the domain, as well as large-scale fluid motions and structures containing significant kinetic energy, which can be detrimental in many industrial applications. Thus, control of RBC is crucial to maximize the efficiency and quality in industrial processes or systems in various sectors. The control might involve, for example, modulating the spatial distribution of the bottom-plate heating in the canonical RBC configuration. Optimizing such a control is challenging topic for classical control-theory methods. Indeed, many classical linear-control techniques such as the adjoint method have only a limited domain of applicability, owing to their local, linear nature, which limits their ability to control non-linear phenomena such as RBC.

Passive flow control has been extensively used with minor improvement in the Ra_c for which instabilities start.[61–65] On the other hand, AFC has explored more sophisticated control strategies that manage to keep a stable convective cell at considerably higher Rayleigh numbers than Ra_c. The most common way of carrying out this AFC has been with the use of jet controllers, performing small perturbations of the velocity and/or temperature, or with a linear control of the temperature in the heated boundary layer.[66–80] In configurations where the RB instability is too strong to be completely eliminated, a reduction in the strength of the RBC cells and associated flow motion, as well as kinetic energy, are also key goals.[23] Recently, the use of DRL to control the RBC has been employed for the first time in Ref. 23. Comparing the results with the linear control employed in Ref. 80, the Ra_c beyond which instabilities start to emerge in the specific configuration considered was increased from 10^4 to 3×10^4, and even in the case where RBC cells were present, their intensity was strongly reduced.

In the present work, we extend the work in Ref. 23 (which we will refer to in the rest of this paper as GB for "Gerben Beintema," from the name of the lead author), to look further into DRL control of RBC. In particular, we improve on the following aspects:

- The domain used by GB is bounded by adiabatic walls on the right and left ends, and the domain itself is relatively narrow. This in turn constrains the RBC cells obtained. Moreover, the walls presented in GB's setup can be used by the DRL agent to help move around and break the RBC cells, as they provide strong flow confinement. By contrast, we use a wider domain of height $H = 2$ and width $L = 2\pi$ (compared with the $H = 1$ and $L = 1$ used in GB) with periodic conditions on the right and left boundaries. In other words, our domain has an aspect ratio of π compared with the aspect ratio of 1 used in GB. As a consequence, our DRL agent has to control the RBC instabilities without the possibility to help itself with the walls, and the RBC cells are closer to unconstrained cells obtained in wide domains relevant for industrial use.

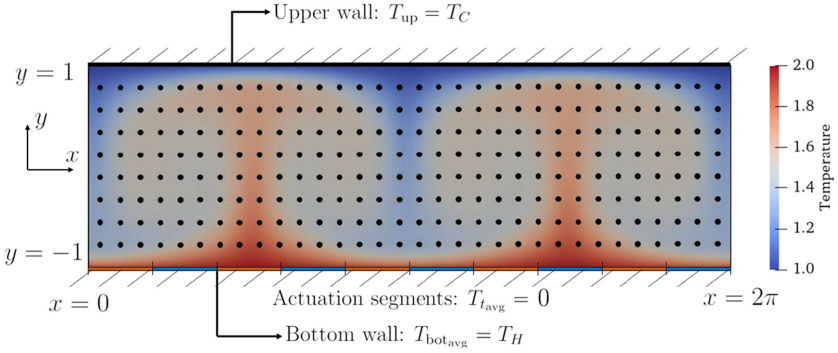

FIG. 1. Schematic representation of the domain. Dimensions are normalized by H. Temperature is controlled at the bottom wall, which is at an average temperature T_H. The upper wall is at a uniform constant temperature T_C, so that $T_H > T_C$, generating natural convection and the emergence of the convective cells. Black dots show the position of the observation probes, distributed as a uniform probe-mesh over the domain. Orange and blue segments in the lower wall correspond to the control segments.

- One limitation in GB's work is how DRL control is applied to the RBC simulation. In order to control the RBC cells, GB modulates the temperature at the lower wall, keeping its mean temperature constant. This, naturally, means that multiple outputs, i.e., temperature perturbations on bottom-wall segments, need to be applied to the N different control segments at the bottom wall. GB formulates this control problem by simply asking a single DRL agent to generate N outputs. As discussed in Ref. 24, this approach results in the curse of dimensionality on the control space, which makes learning challenging. By contrast, we apply the invariant MARL approach suggested by Ref. 24, which allows to control an arbitrary number of heating segments without resulting in the curse of dimensionality. This has a considerable impact on our ability to rapidly learn an effective control of the RBC system.
- GB uses a lattice Boltzmann method (LBM), parallelized to run on a specific graphics-processing unit (GPU), to help accelerate the simulations. In such a way, GB partially alleviated the curse of dimensionality introduced by controlling a multiple-output DRL system without using a MARL approach. Indeed, the use of a GPU-parallelized LBM allows GB to run a very large number of DRL trial-and-error steps and, therefore, to partially compensate for the very slow and challenging learning arising due to the curse of dimensionality. Note, however, that this is only applicable for narrow channels with a limited number of control values, where the curse of dimensionality remains under control. Moreover, LBM is effective, especially as they are well adapted to the parallel nature of GPUs, but understanding their accuracy can be challenging. Finally, GB's implementation is not publicly released and may be hardware dependent. By contrast, we use a spectral Galerkin solver running on standard central-processing Units (CPUs). The solver is both open source, massively parallelizable across many CPU cores, has high order and well-understood accuracy (see Sec. II for more information). Moreover, we release all our code as open-source materials (see Appendix A). Our code can, therefore, be used to further evaluate a wide range of domain sizes and non-dimensional parameters in future studies.

The structure of our study is as follows. In Sec. II, we describe the formulation of the RBC problem including the geometry, equations, and boundary conditions adopted (Sec. II. A.). We then describe the numerical method used to simulate the CFD (Sec. II B.), followed by the details of the DRL-based control (Sec. II C.). In Sec. III, we proceed to present the results. We compare the RBC obtained without control (baseline) to the case with control following a non-MARL method similar to GB and also to the case with control following a MARL approach. We discuss learning speed and robustness. We then show that the invariant MARL controller is able to effectively coalesce RBC cells in the case studied. Finally, we discuss this result and its implication for future works and industrial applications in the conclusions in Sec. IV.

II. METHODOLOGY

A. Formulation of the RBC problem

The governing equations for this problem are the well-known Navier–Stokes equations. In order to formulate these equations in a form adapted to our problem, we use the Boussinesq approximation, which considers the influence of the density gradients only in the gravitational forces term and assumes that density variations are small compared to velocity gradients. Thus, this formulation neglects density changes in the continuity equation, which then translates into a zero-divergence equation. Therefore, under the Boussinesq approximation, for a non-Newtonian incompressible flow, we obtain the continuity [Eq. (1a)], momentum [Eq. (1b)], and energy [Eq. (1c)] equations in dimensionless form[81]

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

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \sqrt{\frac{Pr}{Ra}}\nabla^2 \mathbf{u} + T\mathbf{j}, \tag{1b}$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = \frac{1}{\sqrt{RaPr}}\nabla^2 T, \tag{1c}$$

where $\mathbf{i}$ and $\mathbf{j}$ are the Cartesian unit vectors pointing along the coordinate axes x and y, respectively, and the x-axis is parallel to the walls. The corresponding velocities for each spatial coordinate are $u(x, y, t)$ and $v(x, y, t)$, respectively, $\mathbf{u}(x, y, t) = u\mathbf{i} + v\mathbf{j}$ is the velocity vector, t is the time, $p(x, y, t)$ is the pressure, and $T(x, y, t)$ is the temperature. Along with Ra, the Prandtl number Pr is another dimensionless number that governs the dynamics of the flow. These numbers are expressed as follows:

$$Ra = \frac{\alpha(T_H - T_C)gH^3}{\kappa\nu}, \tag{2a}$$

$$Pr = \frac{\nu}{\kappa}, \tag{2b}$$

where α is the thermal expansion coefficient of the fluid; H is the normalized domain half height; g is the gravitational acceleration with a downward vertical direction, i.e., in our case, perpendicular to the walls; ν is the kinematic viscosity of the fluid; and κ is the thermal diffusivity. In our setup, the flow is confined between the bottom hot wall, at a mean temperature $T_H = 2$, and the cooler upper wall, at a uniform constant temperature $T_C = 1$. The former temperature (T_H) is also uniform in what we call the "baseline" case (i.e., when no control is applied), but it varies with spatial position in what we call the "controlled" case, as discussed below. Both walls are separated by a distance $2H$, and the no-slip boundary condition is applied at both walls. Periodic boundary conditions are applied at the lateral ends of the domain, which has a normalized width $L = 2\pi H$. Figure 1 provides a schematic representation of the RBC domain.

During the entire study, the Prandtl number employed is $Pr = 0.71$, which corresponds to the Prandtl number of air. As described in the introduction (Sec. I), Ra is the ratio of the timescales associated with the thermal transport due to diffusion and convection. This implies that flows with a high Ra are more prone to instabilities due to buoyancy-driven convection. The Ra used in this work is $Ra = 10^4$, which is above the critical Ra_c when no control is applied in the flow.

Another very important non-dimensional number in heat-transfer problems, which is closely related to Ra and is the Nusselt number (Nu), defined as the ratio of the fluxes due to convective heat transfer to conductive heat transfer. For comparison purposes in the results section of the article, we will adopt the same definition as in Ref. 23,

$$Nu = \frac{\bar{q}}{\kappa(T_H - T_C)/H}, \tag{3}$$

where $\overline{q(t)}$ is the time-averaged heat flux in the domain given by

$$\bar{q} = \langle \overline{vT} \rangle_x - \frac{1}{\sqrt{RaPr}} \frac{\partial \langle \bar{T} \rangle_x}{\partial y}. \tag{4}$$

Here, the brackets, $\langle \cdot \rangle_x$, indicate averaging with respect to the indicated spatial direction and the overbar, $\bar{\cdot}$, indicates average in time. A description of the method of derivation of Eq. (4) is given in Appendix B.

B. Numerical method for the CFD

The implementation of the governing equations (1) is based on a spectral Galerkin version of the method developed by Kim *et al.*[82] for direct numerical simulations of turbulent flows. In this method, the pressure is eliminated and the 2D Navier–Stokes equations are basically reduced to the continuity equation and a fourth-order equation for the wall-normal velocity component. Adding the energy equation, the three scalar equations that are solved are Eqs. (1a) and (1c) and

$$\frac{\partial \nabla^2 v}{\partial t} = \frac{\partial^2 H_x}{\partial x \partial y} - \frac{\partial^2 H_y}{\partial x^2} + \sqrt{\frac{Pr}{Ra}} \nabla^4 v + \frac{\partial^2 T}{\partial x^2}, \tag{5}$$

where $\boldsymbol{H} = (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$ is the convection vector. Equation (5) is implemented with the four boundary conditions $v(x, \pm 1) = v'(x, \pm 1) = 0$, where the first two are due to no slip, whereas the two latter follow from the continuity equation. Periodic boundary conditions on the x direction are applied between the left and right domain boundaries, and this is implemented through the choice of the numerical elements and basis functions, as described below.

The three scalar equations (5), (1a), and (1c) are implemented using a highly accurate spectral Galerkin[83] discretization in space and a third-order implicit/explicit (IMEX) Runge–Kutta method[84] for the temporal integration. The Galerkin method makes use of tensor product basis functions constructed from Chebyshev polynomials for the wall-normal direction and Fourier exponentials for the periodic direction. The boundary conditions in both directions are built into the basis functions and as such enforced exactly. For the wall-normal direction, this requires the use of composite Chebyshev polynomials,[83] whereas the Fourier exponentials for the x-direction are already periodic. Since the continuity equation cannot be used to find u for Fourier wavenumber 0, we solve for this wavenumber the momentum equation in the x direction. All other unknowns are closed through Eqs. (5), (1a), and (1c). The convection terms $\boldsymbol{H}$ and $\boldsymbol{u} \cdot \nabla T$ are computed in physical space after expanding the number of collocation points by a factor of 3/2 in order to avoid aliasing. For $\boldsymbol{H}$, we use the rotational form $\boldsymbol{H} = -\boldsymbol{u} \times (\nabla \times \boldsymbol{u})$, with the remaining $1/2\nabla \boldsymbol{u} \cdot \boldsymbol{u}$ absorbed by the pressure, and for temperature, we use the divergence form $\boldsymbol{u} \cdot \nabla T = \nabla \cdot \boldsymbol{u}T$.

The code is implemented using the open-source spectral Galerkin framework "shenfun,"[85] where equations can be automatically discretized through a high-level scripting language closely resembling the Mathematics. The Navier–Stokes solver has been verified by reproducing the growth of the most unstable eigenmode of the Orr–Sommerfeld equations over long time integrations. The Navier–Stokes and Rayleigh–Bénard solvers are distributed as part of the shenfun software, and there is a demonstration guide published in the documentation,[86] which also provides a much more detailed description of the numerical method. Links are provided in Appendix A, and we refer the reader curious of all the technicalities of the CFD solver implementation to the resources detailed therein.

The observation probes are distributed over the domain as a uniform probe-mesh as can be seen in Fig. 1. There are 32×8 probe-mesh points in the x and y directions, respectively. The number of quadrature points in the solution approximation used by the spectral Galerkin solver is 64×96, with Fourier and Chebyshev basis functions in the x and y directions, respectively. The resulting solution is thus spatially continuous, and the observations are then evaluated from these global, continuous spectral Galerkin functions at the uniform-mesh probe locations. These observations are used in the DRL-based control methodology which we describe next.

C. Control methodology with DRL

Using the numerical methods described in the previous section, the flow is solved starting from an initial condition of a constant adverse temperature gradient in the vertical direction. The simulation is performed from this state, leading to the RB instability to occur, which develops into a two-cell configuration that becomes ultimately stationary. No control is applied up to this point, and this constitutes a "baseline" starting state upon which control can be applied. The baseline is saved and then loaded at the start of each episode in the learning phase. The DRL control is performed using the "Tensorforce"[42] framework, from which we use the PPO algorithm.

As mentioned in the Sec. I, the interaction between the agent and the RBC environment occurs through three fundamental channels of communication: the state or observation s_t, reward r_t and action a_t, where the subscript t is the time step of the CFD. The state s_t consists of sets of time-averaged velocity and temperature $(\tilde{u}_t, \tilde{v}_t, \tilde{T}_t)$ measured by computational "probes" distributed across the domain in a uniform mesh as seen in Fig. 1. The averaging is performed over the previous 4 time steps (where the tilde over the values above represents this average). The observations are flattened into one single-dimensional array and then fed to the agent's neural network input layer. The way in which the observations are passed to the agent differs in the single-agent and multi-agent frameworks, and will be described in the following.

The second channel of communication from the environment to the agent is the reward, r_t, which is the desired parameter to be optimized. The goal of the control is to reduce the convective effects, i.e., reduce the intensity of the RBC cells. Since Nu is an indicator of the magnitude of the convective heat transfer, the reward function is set to minimize Nu. Specifically, we aim to minimize the instantaneous Nusselt number Nu_{inst}, which is a function of the instantaneous heat flux $q(t)$. The expression for $q(t)$ is derived in Appendix B. This results in

$$Nu_{\text{inst}} = \frac{q(t)}{\kappa(T_H - T_C)/H}. \tag{6}$$

While our goal is to reduce Nu, DRL algorithms are formulated as an optimization problem that tends to maximize a reward value. Therefore, reward is taken as the negative value of the Nu_{inst}. This is then followed by a translation and scaling to obtain reward values

between 0 and 1 (since Tensorforce's PPO algorithm works better within this range), i.e.,

$$r_t = m(n - Nu_{\text{inst}}), \tag{7}$$

where $n = 2.67$ and $m = 1$ are the shifting and scaling, respectively, to set the reward in the range of $[0, 1]$ during the training.

The applied control actuation is a perturbation T'_t of the temperature boundary conditions on the lower hot wall. To apply this in practice, we adopt a distributed actuation framework where the lower wall is divided into N segments and N actuation values are applied independently, one for each of the segments. The raw $a(j)_t$, $j = 1 \ldots N$ that are generated by the Tensorforce agent are non-dimensional scalar values in the interval $[-1, 1]$. These are then transformed into the temperature values applied on each segment $T(j)_t$, $j = 1 \ldots N$ using a shifting and normalization process that ensures that the average temperature at the bottom wall is kept constant: first, we seek to maintain a constant mean value of temperature, T_H, at the lower wall. Changing T_H would modify the regime of the RBC flow characterized by the Ra [Eq. (2)] and would thus change the driving mechanism of the instability. Keeping a constant mean T_H prevents this. Next, since the N values of the control temperature are generated independently of each other, a shifting of the control temperature is performed to preserve the lower-wall mean temperature to be equal to T_H. The shifted values of temperature, which we denote as $T'_t(j)$, are given as

$$T'_t(j) = a_t(j) - \frac{\sum_{i=1}^{N} a_t(i)}{N}. \tag{8}$$

Next, in order to prevent nonphysical temperature perturbations, we enforce a limit on the final values of the actuations $T(j)_t < |0.75| \, \forall j$ by performing a normalization,

$$T_t(j) = \frac{T'_t(j)}{\max_j(1, |T'_t|)/C}, \tag{9}$$

where C is a scaling constant. Therefore, Eq. (8) implements the mean subtraction to make sure the temperature change applied by the controller does not change the average temperature at the bottom wall, while Eq. (9) makes sure that the maximum absolute value of the temperature change applied is within 0.75, even in the case where the mean subtraction may lead to one of the T'_t values having a magnitude slightly larger than 1.

Finally, to avoid nonphysical sharp discontinuities in the spatial variation of actuations across the segments, the left and right 10% of each segment are smoothed relatively to their immediate neighbors following a cubic transition function. In such a way, the cubic function transitions from $T_t(j)$ to $T_t(j+1)$ in a continuous and differentiable way. Figure 2 shows a schematic representation of how the temperature control is applied as a whole. An illustration of the cubic transition is shown in the zoomed Fig. 2(b).

While the general DRL setup we just described is perfectly valid from a DRL point of view, and it is actually implemented and referred to as the "single-agent-reinforcement-learning" (SARL) setup in the following, as illustrated in Fig. 3, it does not exploit the intrinsic structure of the RBC problem. Indeed, the physical mechanism driving RBC does not depend *per se* on the position along the wall direction (i.e., along the x axis). This, combined with the periodic boundary condition we implement in the CFD solver on the left and right ends of the domain, implies that the DRL-and-RBC setup is globally invariant along the x direction. This invariance can, therefore, be exploited through implementing a MARL approach, as detailed in Ref. 24, and previously discussed.

In the following, we proceed to define the "MARL-DRL" setup based on modified versions of the states, rewards, and actions described thus far. We first regard the N segments introduced previously as separate "pseudo-environments," each with their own independent state, action, and reward channels of communication with the PPO agent. Since our RBC setting is invariant in the x direction, each trajectory in the phase space obtained for each pseudo-environment provides knowledge about the dynamics to control to the DRL agent. As discussed in Ref. 24, this can drastically speed up and improve the quality of the training and learning.

We make a note here regarding the number of pseudo-environments N, which is used is a metaparameter of the MARL setup. Typically, N needs to be chosen so that the following needs are balanced:

1. The segments should be small enough to ensure sufficient spatial granularity in the control signal to be able to actually control the flow. As a consequence, there should be at least a few control segments per spatial feature of the flow. In our baseline case, given that there are 2 convection cells of length scale $\approx\pi$, we should have at least a few control segments per π distance along the x axis.
2. Despite the fact that MARL is efficient at learning even in the case of many control segments,[24] the number of segments should not be too large to overcome certain practical limitations:

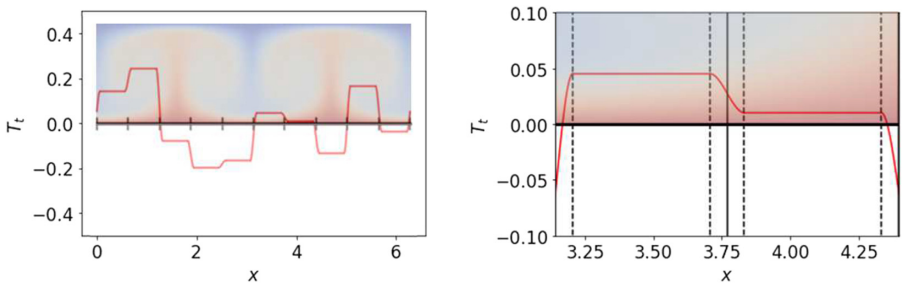

FIG. 2. Schematic representation of how the control T_t is applied to the lower boundary. (a) Global representation and (b) zoom in the transition region between control segments. Observe that $\sum_{i=1}^{N} T_t(i) = 0$, i.e., the mean temperature at the bottom wall is not modified by applying the temperature deviations described by T_t to the initially uniform bottom wall temperature T_H. Note that the temperature profile plotted in the red line is for illustration purposes and does not represent a simulation result.

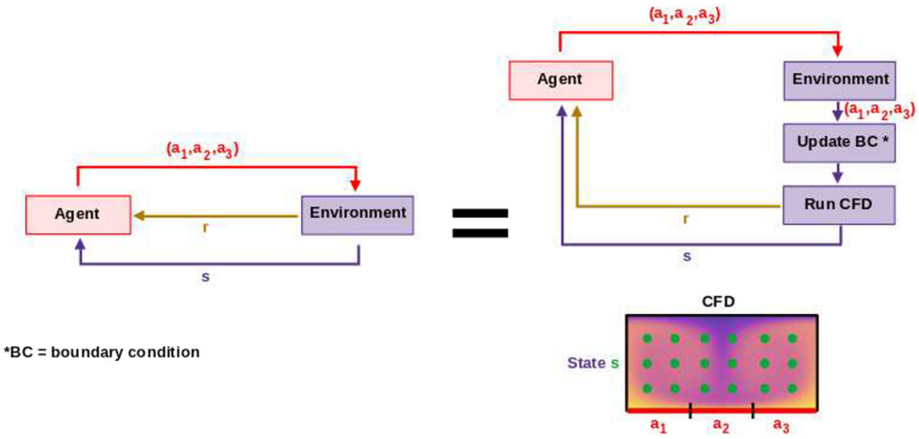

FIG. 3. Illustration of the single-agent-reinforcement-learning (SARL) control setup. In the SARL setup, the agent controls the environment as a whole in a single step. This means that the dimensionality of the action is equal to the number of control segments, which results in a high-dimensional control space and leads to the curse of dimensionality, as discussed in Ref. 24. Note that, in the present illustration, we draw a thinner channel with only 1 RBC cell and 3 control segments; this is purely for the purpose of illustrating the SARL method, since the actual channel used here is wider and has more control segments, as described above.

A large number of segments will make the control output discontinuous as a large number of points, requiring the use of more aggressive smoothing functions.

3. Each pseudo-environment corresponding to each segment will require its own stream of data to be passed to and received from the agent. From a computational viewpoint, this increases the cost of the DRL deployment.
4. Neural networks tend to learn best when learning from uncorrelated data. This means that, ideally, feeding similar data many times to the DRL agent should be avoided. Therefore, providing segments large enough so that the view on each of them is significantly different from the views on the others is beneficial for the quality of the learning.
5. Segments that are too small in size can produce an overall temperature profile (which is the control output), which is highly non-uniform. As a consequence, a Fourier decomposition of the corresponding temperature profile in space will consist of elements with a high wavenumber energy content. These high wavenumber energy content elements can make it more challenging to obtain good stability of the CFD simulation.

Considering the points above, we choose a value of $N = 10$ (Table I) as a reasonable trade-off. We note that this is the analogous (in space) to the considerations that are presented in Ref. 32 when choosing the action duration and episode length of the DRL algorithm (i.e., in time).

In the following, we resume the description of how the states, actions, and rewards communication channels are modified in the MARL-DRL setup. The state s_t of each pseudo-environment is defined as the global set of observations whose order has been rearranged in such a way that the observations from the probes directly above its respective control segment are in the center of the observation list. This "recentering" is further elaborated in the description of the schematic of Fig. 4.

In a similar way, the reward function, r_t, that is provided to each MARL environment, is slightly modified. Separate rewards are computed from each MARL pseudo-environment, and now two values of Nu contribute to the computation of each local MARL reward. The first is the global Nu, which is equal to Nu_{inst} [Eq. (7)]. This part of the reward accounts for the main goal of the agent to optimize the flow over the whole channel. This gives incentive to each pseudo-environment to improve the global flow state (rather than behaving in an egoistically greedy way). The second contribution to the reward is from a local Nusselt number, Nu_{loc}, which is calculated for each specific pseudo-environment in the column of observation probes immediately above its control segment. This local contribution incorporates information about the local effects of the actions on the flow to the reward. This, in turn, allows to provide more reward granularity to the agent during the training, by generating a local indication of the quality of the control. A constant weighting factor, $\beta = 0.0015$, is used to

TABLE I. Parameters used in the CFD simulation and DRL control. Note that for MARL cases, we make a distinction between "MARL episodes" (of which we have $N = 10$ per CFD simulation that lasts for one episode duration) and "CFD episodes" (of which we have one per CFD simulation that lasts for one episode duration). In general, episodes are defined by the duration of performing a set number of control actuations on the environment, before terminating the simulation of the environment. In a CFD episode, one actuation is a single set of N control actions for the N pseudo-environments. In a MARL episode, one actuation is one control action performed on one pseudo-environment. Given in the table that the number of actuations per episode is 200, this means there are 200 sets of actuations for a single CFD episode, which corresponds to a total of 200 $\times$ N actuations performed from the MARL actors as a whole.

Parameter	Value
Domain size $L \times H$	$2\pi \times 2$
Galerkin modes	96×64
Time step	0.05
Pr	0.7
Ra	10^4
β	0.0015
Action scaling factor, C	0.75
Number of observation probes	8×32
Number of CFD episodes	350
Number of action steps per episode	200
Number of control segments, N	10
Baseline duration	400 time units
Action duration	1.5 time units
Episode duration	300 time units

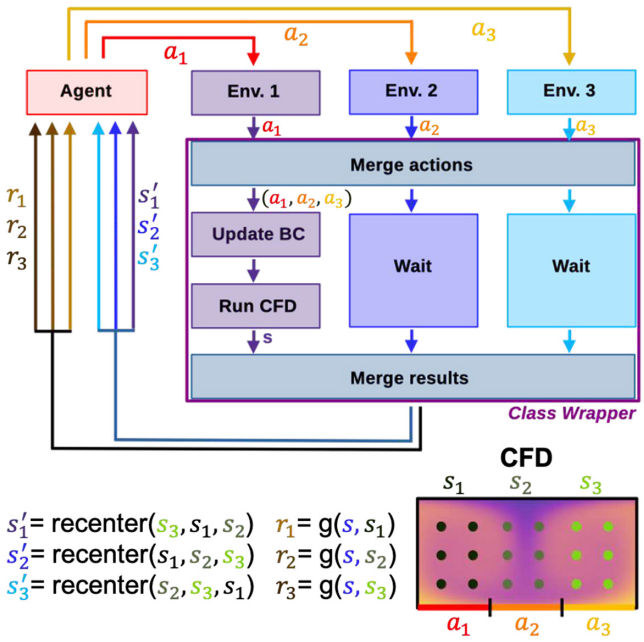

FIG. 4. Schematic of the MARL control setup, where the same agent controls local regions of the domain as if they were separate environments. This allows to (i) re-use the same policy knowledge across the domain and (ii) get a local reward information during training. This, in turn, allows to alleviate the curse of dimensionality on the action space dimension when controlling systems with invariants, as described in Ref. 24. A code wrapper is needed to ensure synchronization of the parallel local environments with the CFD simulation. Note that for illustrative purposes, the bottom right diagram shows only three pseudo-environments instead of N.

quantify the contribution of Nu_{loc} and Nu_{inst} to the total reward r_t for each pseudo-environment. The value of β appears to be low, but this is only due to the large difference in magnitudes that exists between Nu_{loc} and Nu_{inst}. This large difference in magnitude arises from the difference in the geometric width used to compute Nu_{loc} and Nu_{inst} (since we use $N = 10$ segments, each segment has a width factor of 1/10 with respect to the total domain width). We choose β such that Nu_{loc} has approximately a 10% contribution to the total reward. The final form of the reward function for each MARL pseudo-environment reads as follows:

$$r_t = m(n - (1 - \beta)Nu_{\text{inst}} - \beta Nu_{\text{loc}}), \tag{10}$$

with values of n and m as defined for Eq. (7).

A schematic representation of the MARL-based control methodology is presented in Fig. 4. In order to make the figure easier to read, only three representative pseudo-environments are shown there instead of $N = 10$. Each pseudo-environment corresponds to a localized region of the physical domain. Note that for ease of explanation, in this plot, we deviate from the notation convention used in the rest of the paper: the subscripts for s, a, and r are now the environment index $j = 1$, 2, and 3. We define s_j as the values of the observation from the probes directly above each control segment, r_j as the total reward from each pseudo-environment [Eq. (10)] and a_j as the control actions applied by each MARL pseudo-environment. The top diagram represents the control methodology. At the start of the cycle, the a_j are merged and communicated to the first pseudo-environment ($j = 1$), which is the sole pseudo-environment that runs the whole CFD simulation, while the others pseudo-environments wait until CFD simulation is finished. The CFD simulation is run for a time equal to the action duration (Table I). Once the action is finished, the entire set of observations $s = (s_1, s_2, s_3)$ is retrieved from the observation probes and communicated to all the pseudo-environments. Then, in the "Merge results" block, each pseudo-environment executes the "recenter()" function. In recenter(), a rearranging of the order of s_j is performed in each pseudo-environment, so that the s_j corresponding to pseudo-environment j is moved to the middle of the observation vector. We denote the recentered observation vectors as s_j'. The r_j is obtained from the Nu_{loc} and Nu_{inst} computed using the local s_j and the global s, respectively. This is done using Eq. (10), and it is represented for brevity as a function of g in Fig. 4. The s_j' and r_j are then passed to the respective MARL agents, which output the actions a_j and the cycle is repeated again. The MARL agents are actually implemented as data streams to a repeated DRL agent, which means that all MARL agents share the exact same policy and neural networks. This is how the control invariance across the domain is implemented in the MARL case, i.e., by re-using the exact same agent multiple times across the domain, similar to what is described in Ref. 24.

Having described the setup of SARL and MARL, we now proceed to the training process. The main parameters used for the DRL setup are summarized in Table I, and the hyperparameters used for the definition and training of the agent are in Table II. A total of 350 CFD episodes are performed for the training run in SARL, with 200 actuation updates per episode. Each actuation consists of a set of N actions or temperature control values for the N segments. In SARL, the policy update is performed every "batch size" number of episodes, which in our case is set to be 20. In MARL, we make a distinction between "CFD episodes" and "MARL episodes." The "CFD episode" is when the CFD simulation is run for 200 actuations, as in the SARL case. Each actuation comprises N actions applied locally by each MARL pseudo-environment. Thus, each MARL pseudo-environment, in the course of one CFD episode, applies 200 control actions, which we define as a single MARL episode. Given that there are N MARL pseudo-environments, there are therefore $350 \times N$ MARL episodes in one training run for MARL. The batch size for MARL is 20 MARL episodes, or $20/N$ CFD episodes. In the discussion of the results

TABLE II. Hyperparameters used in the definition and training of the DRL agent. The hyperparameters are common to SARL and MARL.

Hyperparameter	Value
Batch size for SARL	20 episodes
Batch size for MARL	20 MARL episodes
Learning rate	10^{-3}
Entropy regularization factor	0.01
Number of hidden layers (baseline network and policy network)	2
Number of units per hidden layer	512
Activation function	tanh
Optimizer	Adam

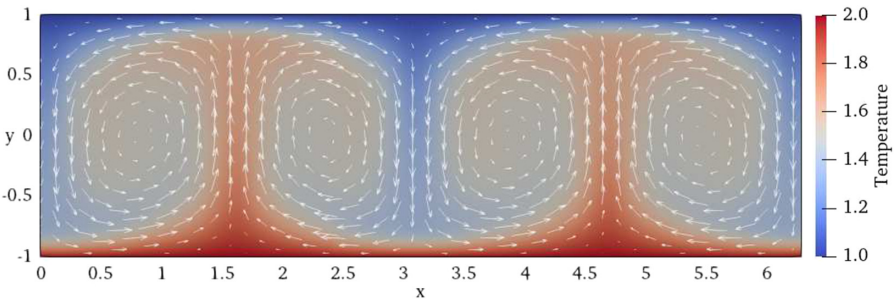

FIG. 5. Instantaneous temperature (background color) and velocity (arrows) fields from the end of the baseline simulation. This is the baseline steady state used at the start of each new episode, displaying two convective cells ($Nu = 2.68$) (see also the corresponding multimedia data from Appendix C). Multimedia available online.

section, in the context of MARL, the word "episode" refers to "CFD episode."

Training in both SARL and MARL is performed with a discount factor of $\gamma = 0.99$. Wall-clock training time for 350 episodes in SARL and 350 CFD episodes in MARL is around 34 h using a single core on a modern workstation (the wall clock duration is approximately the same in both cases, as CFD is the dominating computational cost). The duration of each episode in CFD time units is action duration $\times$ number of actions per episode $=$ 300 time units (Table I). This time is enough to capture the dynamics of the flow in the transition from the baseline state to the optimized controlled flow. Within the duration of one action (i.e., within a single control step), the control temperature applied to each control segment is established by the DRL agent through Eqs. (8) and (9). This control step is chosen to be short enough so that the agent can actuate under fast changes of the state of the flow, but also long enough so that the flow can develop according to the actuation performed, similar to what is recommended in Rabault *et al.*[16]

III. RESULTS AND DISCUSSION

In Sec. III, we analyze and discuss the results obtained. In the first place, at the start of the baseline convergence simulation, three convective cells appear. However, this is an unstable state that lasts for just a few time units and the final baseline, which includes a two-cell configuration, is achieved as visible in Fig. 5 (Multimedia view) (see also the corresponding multimedia data from Appendix C). A full video of the baseline simulation can be seen in the link provided in Appendix C.

Figure 6 shows the evolution of the Nusselt number and the kinetic energy along the duration of the baseline convergence simulation. The flow reaches a stable state after approximately 400 time units, where the Nusselt number converges to a value of $Nu = 2.68$.

In the baseline flow, two convective cells are present, with a total of two counter-rotating vortex pairs. This is different from the baseline layout obtained in GB, where a single rotating vortex was obtained. The differences in flow topology with respect to the results by GB come from the adiabatic lateral walls and the aspect ratio of 1 used in GB, compared with the periodicity in x and an aspect ratio of π used

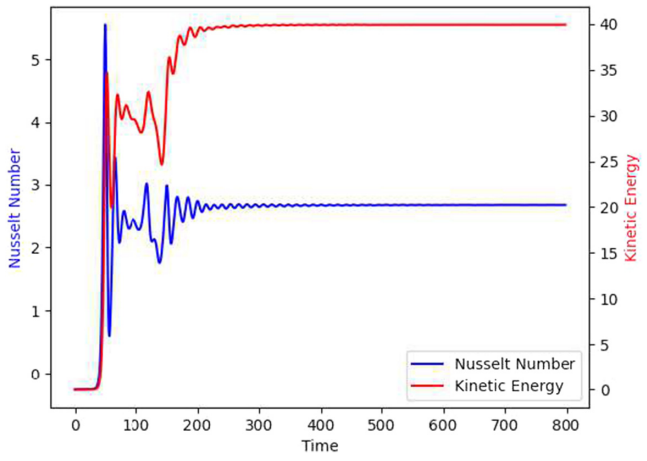

FIG. 6. Evolution of the global Nusselt number and kinetic energy during the baseline simulation with $Ra = 10^4$.

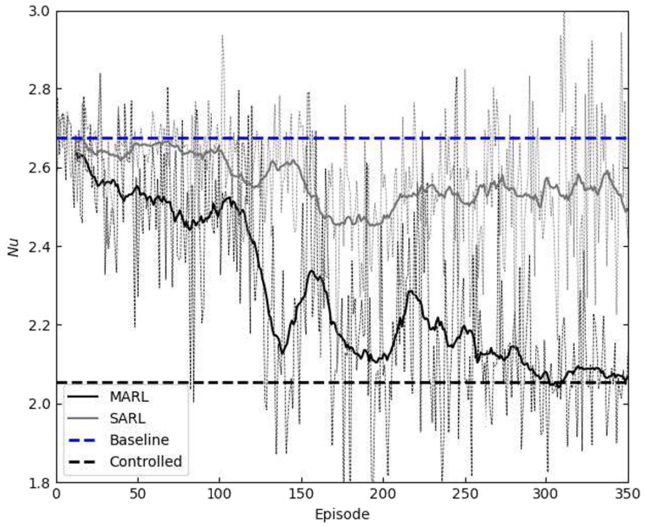

FIG. 7. Learning curves obtained in both the SARL and MARL configurations. Note that each episode in the horizontal axis corresponds to a single CFD run, which consists of $N = 10$ individual MARL episodes when the DRL algorithm is applied in the MARL configuration. The dotted lines in grayscale are the actual mean reward obtained per episode, and the full lines are moving averages over 25 episodes that illustrate the learning trends. The horizontal bold dashed lines are drawn for reference, and they represent *Nu* at the end of the baseline simulation (in blue) and the average *Nu* from an episode where MARL control is executed (in black). A movie showing control with the trained SARL agent is shown in the link provided in Appendix C. Multimedia available online.

in the present work. Therefore, we can expect different control strategies compared to those obtained in GB.

We now present the results obtained from DRL control of the Rayleigh–Bénard system. To compare the learning in the SARL and MARL frameworks, we plot both their learning curves in Fig. 7 (Multimedia view) (see also the corresponding multimedia data from Appendix C). The learning was performed for 350 CFD episodes of 200 actuations each as previously highlighted, starting from the final state of the baseline (Fig. 5, see also the corresponding multimedia data from Appendix C). This means that the same CFD duration is used to train both the SARL and MARL agents. Since CFD simulation is by far the most time- and resource-intensive part of the training process (similar to what was reported in Ref. 16), this means that the wall-clock time and resources used for performing the SARL and MARL trainings are about equal. The time-averaged *Nu* at the end of the baseline is plotted for reference, corresponding to a value of 2.68 in Fig. 7 (see also the corresponding multimedia data from Appendix C). We also show the time-averaged *Nu* at the final episode of the MARL, which is equal to 2.07, corresponding to the dashed horizontal black line.

Figure 7 (see also the corresponding multimedia data from Appendix C) clearly demonstrates that the MARL approach learns much faster and effectively than the SARL approach. This figure shows that, while the SARL approach plateaus after around 150 episodes and displays only very limited *Nu* reduction, the MARL method learns initially much faster than SARL and keeps learning for a much longer time until it reaches a far better *Nu* reduction. To be more specific, at the end of 350 episodes, the MARL approach is achievable a reduction in *Nu* of 22.7%, whereas the SARL approach achieves approximately 5.1% reduction. Moreover, a closer look at the last state of the last episodes in the SARL and MARL cases, respectively, shows that while the MARL agent has managed to change the topology of the flow and to

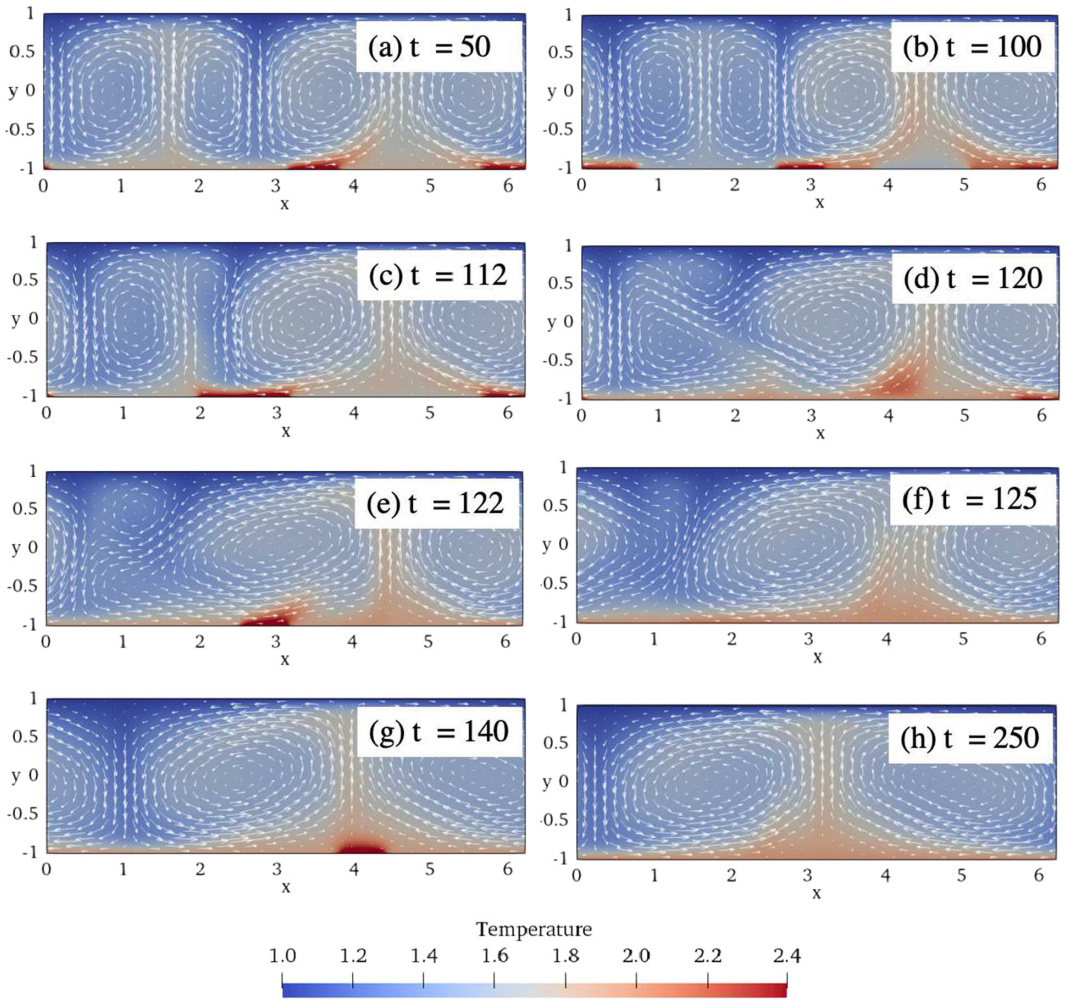

FIG. 8. Sequence of temperature (background color) and velocity (arrows) fields showing the dynamics of the control action taken by the trained agent during a single episode without exploration noise (deterministic, or evaluation, mode). Starting from the baseline double-RBC-cell flow, the agent destabilizes the flow, leading to coalescence into a single cell. This cell is then controlled until it reaches a stable configuration. Each frame in the current figure corresponds to the snapshots of the flow field at time instants marked with red dot symbols in Fig. 9, over the different phases of flow development (see also the corresponding multimedia data from Appendix C). Multimedia available online.

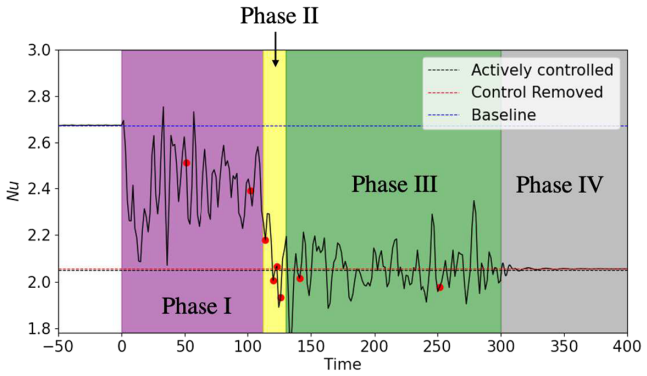

FIG. 9. Evolution of the Nusselt number showing the control achieved by the DRL agent in a single deterministic run (i.e., without exploration noise in the DRL agent policy). The shaded area shows the different phases of the flow evolution. The red point markers are the time instants at which the frames in Fig. 8 are sampled. The black horizontal dashed line is the mean *Nu* of the actively controlled single-cell configuration from the current episode. The red horizontal dashed line is the *Nu* achieved at the end state of the system after the control is removed and the naturally stable single-cell RBC configuration is observed (corresponding to the flow visible in Fig. 10).

reach a coalesced RBC convection cell state (more details below), the SARL approach has only a limited effect on the flow topology (visible from the videos in Appendix C). This implies that, while the MARL agent has learned an effective policy and control law, the agent in the SARL framework has failed to do so. This is very similar to the findings reported in Ref. 24, as previously discussed. The reader curious of more details about the flow dynamics in each case is referred to the videos in Appendix C to see the difference in the controlled-flow configurations. Furthermore, insight into the MARL control mechanism is provided in the next paragraphs.

The price to pay for the DRL ability to discover and optimize non-linear control laws is the lack of direct interpretability of the DRL agent policy. Given a trained DRL agent, and in particular, its policy network weights, it is difficult to derive a detailed, quantitative understanding of the control law using a bottom-up approach, and the DRL policy *per se* is effectively a black box. This contrasts with, e.g., linear flow-control methods, where properties such as the eigenmodes and eigenvectors of the associated operators can be used to extract some form of explainability.[87,88] However, a higher-level, phenomenological understanding of the DRL agent policy is still obtainable, by applying expert knowledge to the analysis of the control laws exhibited by the DRL agent. Such a phenomenological understanding is what we discuss in the following paragraphs.

More specifically, different phases are clearly visible when the DRL agent controls RBC, as highlighted in Fig. 8 (Multimedia view) (see also the corresponding multimedia data from Appendix C). Starting from the baseline flow, the DRL agent starts by weakening the RBC cells through modulation of the bottom wall temperature. We will refer to this destabilization of the baseline regime as the Phase I. The destabilization is performed by the agent through applying (i) increased heating on the regions of the bottom plate where a falling plume of low temperature fluid is observed, and (ii) decreased heating (or relative cooling) on the regions of the bottom plate where a rising plume of high temperature fluid is observed. This is visible in Figs. 8(a) and 8(b) (see also the corresponding multimedia data from Appendix C). This, in practice, weakens the driving mechanism of the baseline RBC cells, by producing excess buoyancy in the downwelling part of the RBC plume. As a result, the control on Phase I reduces the excess of buoyancy in the rising part of the RBC plume, hence, opposing the buoyancy driven flow motion.

This opposition to and weakening of the baseline flow results in an unstable situation, where the double RBC cells are broken. We recognize this transition as the phase II and it can be observed in Figs. 8(c)–8(f) (see also the corresponding multimedia data from Appendix C) and in Fig. 9. This Phase II ends with a recombination of the two initial convective cells into a single coalesced RBC cell, thus leading to the Phase III. Phase III is distinguished by a single coalesced bubble regime, which is visible in Figs. 8(g) and 8(h) (see also the corresponding multimedia data from Appendix C).

Therefore, it appears that the DRL agent has learnt to "navigate" in the phase space of the problem. While the phase space configuration corresponding to the baseline configuration is stable, it leads to a suboptimal reward. In particular, the DRL agent has found a strategy (applied during the transition phases I and II) that can bring the system away from the baseline configuration and into the neighborhood of a new topological flow configuration, i.e., the coalesced bubble observed in phase III, which provides higher reward.

Interestingly, the single coalesced bubble obtained in phase III is naturally stable once it is established and is in fact providing the optimal reward, even without control. This is what we define as phase IV in Fig. 9, i.e., the flow obtained upon the re-imposition of the lower-wall constant temperature boundary condition (as in the baseline) after the end of episode. A snapshot of this final state is observed in Fig. 10 (Multimedia view) (see also the corresponding multimedia data from Appendix C).

Figure 11 shows the Nusselt number obtained through three different episodes at different stages of the learning. While at episode 45, the DRL agent was not able to coalesce the two convective cells into one, at episodes 275 and 350, it has already learned the strategy to

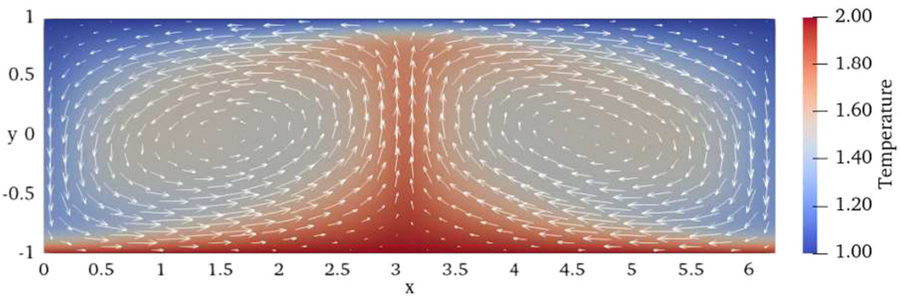

FIG. 10. Illustration of the single-cell configuration maintained by the system after the DRL-control is removed and the uniform hot-wall boundary condition re-imposed. Interestingly, once the DRL controller has moved the flow to this new configuration, it is intrinsically stable and yields the best reward, even when left unactuated (see also the corresponding multimedia data from Appendix C). Multimedia available online.

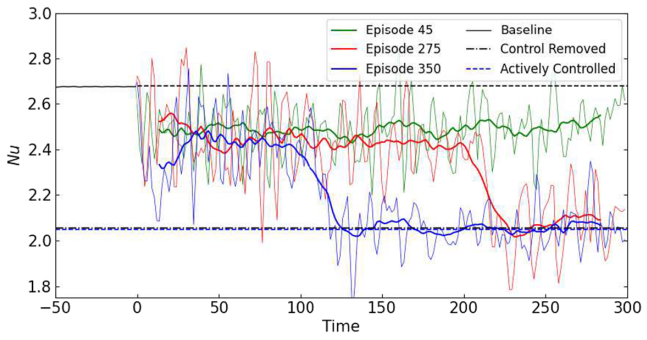

FIG. 11. Evolution of the Nusselt number in three different episodes (episode numbers 45, 275, and 350), chosen from the MARL learning curve in Fig. 7. The black horizontal line (dashed for positive times) represents the value of *Nu* at the baseline. The dash-dotted horizontal line is the *Nu* number at phase IV. The control corresponding to episode 45 only leads to a minor change in *Nu*. By contrast, the controls corresponding to episodes 275 and 350 lead to much better *Nu* reduction, corresponding to the transition to a single coalesced bubble. The difference between episodes 275 and 350 lies in the ability of the controller obtained at episode 350 to reach the coalesced state faster than what is the case at episode 275.

perform the coalescence. This is indicated by the large jumps in the Nusselt number at time units 115 and 215 for episodes 350 and 275, respectively. We can also observe that the agent trained for more episodes (episode 350) is able to reach the Phase III (single-cell configuration) earlier than the agent trained for fewer episodes (episode 275). However, once the flow configuration is a single cell, no matter when it is reached, the value of the Nusselt number converges to a minimum of approximately 2.1. This is related to the fact that the single-cell configuration is a stable state with a higher reward, i.e., lower *Nu*, than the two-cell configuration. If once on phase III the control is turned off, i.e., in phase IV, the Nusselt number achieved gradually converges to 2.05, which is the best *Nu* we ever observe. Therefore, any small control applied during Phase III is noise (either exploration noise during training, or agent "uncertainty" during evaluation). The black dash-dotted horizontal line in Fig. 11 shows the value of the Nusselt number for phase IV.

IV. CONCLUSIONS

In the present work, we demonstrate effective Rayleigh–Bénard Convection (RBC) control by using deep reinforcement learning (DRL). Compared with previous works, e.g., Ref. 23, we consider a domain with a higher aspect ratio, i.e., a wider domain. This leads to the existence of several rolls or RBC cells, and the need for using more separate actuators at the bottom walls. We show that a naive single-agent DRL method fails to learn an effective control policy. This is due to the difficulty presented by having to control several actuators simultaneously, which results in the curse of dimensionality in the control space. By contrast, we demonstrate that leveraging invariant multi-agent reinforcement learning (MARL), which takes advantage of the invariant structure of the underlying RBC control problem, allows to discover an effective control strategy in the case studied. We demonstrate that, in practice, "multi-agent reinforcement learning is all you need" to control our multiple-input, multiple-output fluid-dynamics problem. This finding is similar to previous results,[24] and we expect to keep observing it repeatedly when applying DRL to control physical systems with strong underlying structures and symmetries. Using a MARL approach results in a much faster learning than what was obtained in Ref. 23, and is key to our ability to control wider channels with more actuators.

Going into more details, our MARL controller finds a complex, non-trivial control strategy in the RBC configuration under consideration. Through trial and error, the MARL approach discovers that adequate control of the lower wall temperature profile can destabilize the double RBC cell pattern observed in our baseline flow, and force these RBC cells to coalesce into a single convection cell. This convection cell is, in turn, intrinsically stable and yields better heat exchange performance, i.e., a lower value of the Nusselt number *Nu*, following the definition of our optimization problem.

The present study is a first application of MARL to RBC control, and we provide a strong framework for further studies through the release of all the codes and scripts as open material. Many interesting questions remain open to discussion in future works, and we expect that more studies of the RBC controllability using MARL will take place. In particular, this framework is well adapted to studying problems such as the effect of the Rayleigh number *Ra*, the channel aspect ratio, the strength of the control authority, etc., on controllability and optimal states of the RBC system. We believe that this work is also a first step toward demonstrating RBC MARL control in both 3D configurations, and in geometries relevant for industrial applications.

ACKNOWLEDGMENTS

Part of the deep-learning-model training was enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS). RV acknowledges financial support from ERC Grant No. "2021-CoG-101043998, DEEPCONTROL." Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Colin Vignon: Data curation (equal); Investigation (lead); Validation (equal); Visualization (equal); Writing – review & editing (equal). **Jean Rabault:** Conceptualization (equal); Supervision (lead); Validation (equal); Writing – original draft (lead); Writing – review & editing (equal). **Joel Varun Vasanth:** Data curation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Francisco Alcántara Ávila:** Supervision (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Mikael Mortensen:** Software (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal). **Ricardo Vinuesa:** Conceptualization (equal); Funding acquisition (equal); Resources (equal); Supervision (equal); Validation (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX A: CODE RELEASE

All the codes, scripts, and post-processing tools used in this work are made available on Github, together with readmes and user instructions, at the following address: https://github.com/KTH-FlowAI/DeepReinforcementLearning_RayleighBenard2D_Control.

The shenfun CFD case setup is described in great details, including all the numerical implementation considerations, elements chosen, and code walk-through, on the shenfun RBC documentation page: https://shenfun.readthedocs.io/en/latest/rayleighbenard.html. Note that the conventions used in the present paper and in this documentation page are slightly different: while the present paper uses x as the horizontal ("wall parallel") description and y as the vertical ("wall normal") direction, the documentation page uses the opposite conventions. This has no influence on the CFD results *per se*. The code uses the same conventions as the shenfun documentation, and only the present paper uses separate conventions, for simplicity of the writing and conformity with the literature.

APPENDIX B: HEAT-FLUX DERIVATION

In this section, we derive the expression for the non-dimensional time-averaged heat flux q as provided in Eq. (4). We begin with the non-dimensional energy equation [Eq. (1c)]. To this end, we add Eq. (1a) to obtain

$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{u}T) = \frac{1}{\sqrt{RaPr}} \nabla \cdot \nabla T. \tag{B1}$$

The above equation is volumetric and can thus be integrated over the entire volume of the domain. We then apply the Gauss divergence theorem to obtain

$$\int_V \frac{\partial T}{\partial t} \mathrm{dV} + \int_S (\mathbf{u}\mathrm{T}) \cdot \hat{\mathbf{n}}\, \mathrm{dS} = \frac{1}{\sqrt{\mathrm{RaPr}}} \int_S \nabla \mathrm{T} \cdot \hat{\mathbf{n}}\, \mathrm{dS}, \tag{B2}$$

where S and V are the boundaries and volume of the domain, and $\mathrm{d}S$ and $\mathrm{d}V$ represent elementary surface and volume units, respectively. Also note that $\hat{\mathbf{n}}$ is the unit vector normal to S. The second term on the LHS and the term on the RHS represent surface-integrated "flux terms" corresponding to the convective and conductive heat fluxes q_{conv} and q_{cond} across S, respectively.

Since the domain in the current study is two-dimensional, integrals in the z direction evaluate to a constant and the surface integrals reduce to line integrals over the domain boundaries. Given that the boundary conditions are periodic in x, the component of the flux terms integrated over the vertical boundaries, i.e., from $y = -1$ to 1, at $x = 0$ and $x = 2\pi$ sum to 0. Hence, we are left with the terms integrated over the horizontal boundaries or walls of the domain. The second term on the LHS of Eq. (B2) becomes

$$\int_x (\mathbf{u}T) \cdot \hat{\mathbf{j}}\, \mathrm{d}x = \int_x vT\, \mathrm{d}x, \tag{B3}$$

and the RHS becomes

$$\frac{1}{\sqrt{RaPr}} \int_x \nabla T \cdot \hat{\mathbf{j}}\, \mathrm{d}x = \frac{1}{\sqrt{\mathrm{RaPr}}} \int_x \frac{\partial \mathrm{T}}{\partial \mathrm{y}}\, \mathrm{dx}. \tag{B4}$$

Note that the integrands in Eqs. (B3) and (B4) are evaluated at $y = \pm 1$, and the limits of integration in x are from 0 to 2π. The sum of Eqs. (B3) and (B4) equates to the sum of the convective and conductive surface-integrated heat fluxes,

$$q(t) = q_{\mathrm{conv}}(t) + q_{\mathrm{cond}}(t) = \int_0^{2\pi} \left(vT - \frac{1}{\sqrt{RaPr}} \frac{\partial T}{\partial y} \right) \mathrm{dx}. \tag{B5}$$

Multiplying Eq. (B5) with $1/L = 1/2\pi$, the resulting terms are then averages of the integrands over x at $y = -1$ and 1. We represent averages using the angle bracket notation as

$$q(t) = \langle vT \rangle_x - \frac{1}{\sqrt{RaPr}} \frac{\partial \langle T \rangle_x}{\partial y}, \tag{B6}$$

where we used the Leibnitz integral rule for the last term to interchange the integral and derivative operators. Equation (4) is then obtained by performing a temporal average of Eq. (B6).

APPENDIX C: EXTRA MATERIALS

Extra materials are provided as a series of videos on YouTube, which illustrate

- The development and settling of the baseline flow (Fig. 5): see https://www.youtube.com/watch?v=aSKZ1qNgMWk
- The RBC undergoing control by the SARL controller, starting from the baseline flow, which illustrates that SARL fails to learn a topology-changing control strategy (Fig. 7, SARL episode 350): see https://www.youtube.com/watch?v=N9wTi4K2uJY
- The RBC undergoing successful control by the MARL controller, starting from the baseline flow, which illustrates the bubble coalescence phase and the large bubble phase (Fig. 8): see https://www.youtube.com/watch?v=_HYmWee3P0A
- The modified flow configuration corresponding to a single stable RBC cell, obtained by re-applying a uniform bottom wall temperature at the end of the MARL control sequence (Fig. 10): see https://www.youtube.com/watch?v=MEGl2IiybSg

REFERENCES

[1] J. Rabault and A. Kuhnle, "Deep reinforcement learning applied to active flow control," in *Data-Driven Fluid Mechanics: Combining First Principles and Machine Learning*, edited by M. A. Mendez, A. Ianiro, B. R. Noack, and S. L. Brunton (Cambridge University Press, 2023), pp. 368–390.
[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller *et al.*, "Human-level control through deep reinforcement learning," Nature **518**, 529 (2015).
[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).
[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2018).
[5] Y. Li, "Deep reinforcement learning: An overview," arXiv:1701.07274 (2017).
[6] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in Proceedings of the AAAI Conference on Artificial Intelligence, 2016, Vol. 30.

[7]V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," arXiv:1312.5602 (2013).

[8]D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning* (PMLR, 2014), pp. 387–395.

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

[10]D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," Science **362**, 1140–1144 (2018).

[11]O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in Starcraft II using multi-agent reinforcement learning," Nature **575**, 350–354 (2019).

[12]J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," Nature **602**, 414–419 (2022).

[13]T. Sonoda, Z. Liu, T. Itoh, and Y. Hasegawa, "Reinforcement learning of control strategies for reducing skin friction drag in a fully developed channel flow," arXiv:2206.15355 (2022).

[14]L. Guastoni, J. Rabault, P. Schlatter, H. Azizpour, and R. Vinuesa, "Deep reinforcement learning for turbulent drag reduction in channel flows," Eur. Phys. J. E: Soft Matter Biol. Phys. **46**(4), 27 (2023).

[15]M. Chevalier, "Adjoint based control and optimization of aerodynamic flows," Ph.D. thesis (Mekanik, 2002).

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

[17]H. Xu, W. Zhang, J. Deng, and J. Rabault, "Active flow control with rotating cylinders by an artificial neural network trained by deep reinforcement learning," J. Hydrodyn. **32**, 254–258 (2020).

[18]S. Qin, S. Wang, J. Rabault, and G. Sun, "An application of data driven reward of deep reinforcement learning by dynamic mode decomposition in active flow control," arXiv:2106.06176 (2021).

[19]M. Tokarev, E. Palkin, and R. Mullyadzhanov, "Deep reinforcement learning control of cylinder flow using rotary oscillations at low Reynolds number," Energies **13**, 5920 (2020).

[20]J. Li and M. Zhang, "Reinforcement-learning-based control of confined cylinder wakes with stability analyses," J. Fluid Mech. **932**, A44 (2022).

[21]F. Ren, J. Rabault, and H. Tang, "Applying deep reinforcement learning to active flow control in weakly turbulent conditions," Phys. Fluids **33**, 037121 (2021).

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

[23]G. Beintema, A. Corbetta, L. Biferale, and F. Toschi, "Controlling Rayleigh–Bénard convection via reinforcement learning," J. Turbul. **21**, 585–605 (2020).

[24]V. Belus, J. Rabault, J. Viquerat, Z. Che, E. Hachem, and U. Reglade, "Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film," AIP Adv. **9**, 125014 (2019).

[25]Y.-Z. Wang, Y.-F. Mei, N. Aubry, Z. Chen, P. Wu, and W.-T. Wu, "Deep reinforcement learning based synthetic jet control on disturbed flow over airfoil," Phys. Fluids **34**, 033606 (2022).

[26]R. Vinuesa, O. Lehmkuhl, A. Lozano-Durán, and J. Rabault, "Flow control in wings and discovery of novel approaches via deep reinforcement learning," Fluids **7**, 62 (2022).

[27]F. Ren, C. Wang, and H. Tang, "Active control of vortex-induced vibration of a circular cylinder using machine learning," Phys. Fluids **31**, 093601 (2019).

[28]F. Ren, C. Wang, and H. Tang, "Bluff body uses deep-reinforcement-learning trained active flow control to achieve hydrodynamic stealth," Phys. Fluids **33**, 093602 (2021).

[29]C. Jiang-Li, C. Shao-Qiang, R. Feng, and H. Hai-Bao, "Artificially intelligent control of drag reduction around a circular cylinder based on wall pressure feedback," Acta Phys. Sin. **71**, 084701 (2022).

[30]A. Corrochano and S. Le Clainche, "Structural sensitivity in non-linear flows using direct solutions," Comput. Math. Appl. **128**, 69–78 (2022).

[31]D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Reinforcement learning for bluff body active flow control in experiments and simulations," Proc. Natl. Acad. Sci. U. S. A. **117**, 26091–26098 (2020).

[32]J. Rabault and A. Kuhnle, "Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach," Phys. Fluids **31**, 094105 (2019).

[33]H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang, "Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning," Phys. Fluids **32**, 053605 (2020).

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

[35]D. Xu and M. Zhang, "Reinforcement-learning-based control of convectively unstable flows," J. Fluid Mech. **954**, A37 (2023).

[36]R. Paris, S. Beneddine, and J. Dandois, "Robust flow control and optimal sensor placement using deep reinforcement learning," J. Fluid Mech. **913**, A25 (2021).

[37]R. Paris, S. Beneddine, and J. Dandois, "Reinforcement-learning-based actuator selection method for active flow control," J. Fluid Mech. **955**, A8 (2023).

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

[39]P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem, "A review on deep reinforcement learning for fluid mechanics," Comput. Fluids **225**, 104973 (2021).

[40]C. Vignon, J. Rabault, and R. Vinuesa, "Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions," Phys. Fluids **35**, 031301 (2023).

[41]Y. Matsuo, Y. LeCun, M. Sahani, D. Precup, D. Silver, M. Sugiyama, E. Uchibe, and J. Morimoto, "Deep learning, reinforcement learning, and world models," Neural Networks **152**, 267 (2022).

[42]A. Kuhnle, M. Schaarschmidt, and K. Fricke, "Tensorforce: A Tensorflow library for applied reinforcement learning" (2017), see https://github.com/tensorforce/tensorforce.

[43]A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," J. Mach. Learn. Res. **22**, 12348–12355 (2021), see https://dl.acm.org/doi/abs/10.5555/3546258.3546526.

[44]J. Weng, H. Chen, D. Yan, K. You, A. Duburcq, M. Zhang, Y. Su, H. Su, and J. Zhu, "Tianshou: A highly modularized deep reinforcement learning library," arXiv:2107.14171 (2021).

[45]Q. Wang, L. Yan, G. Hu, C. Li, Y. Xiao, H. Xiong, J. Rabault, and B. R. Noack, "DRLinFluids: An open-source Python platform of coupling deep reinforcement learning and OpenFOAM," Phys. Fluids **34**, 081801 (2022).

[46]M. Kurz, P. Offenhäuser, D. Viola, O. Shcherbakov, M. Resch, and A. Beck, "Deep reinforcement learning for computational fluid dynamics on HPC systems," J. Comput. Sci. **65**, 101884 (2022).

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

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

[49]T. Duriez, S. L. Brunton, and B. R. Noack, *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence* (Springer, 2017), Vol. 116.

[50]B. R. Noack, "Closed-loop turbulence control-from human to machine learning (and retour)," in *Fluid-Structure-Sound Interactions and Control: Proceedings of the 4th Symposium on Fluid-Structure-Sound Interactions and Control* (Springer, 2019), pp. 23–32.

[51]A. Debien, K. A. F. F. von Krbek, N. Mazellier, T. Duriez, L. Cordier, B. R. Noack, M. W. Abel, and A. Kourta, "Closed-loop separation control over a sharp edge ramp using genetic programming," Exp. Fluids **57**, 40 (2016).

[52]R. Li, B. R. Noack, L. Cordier, J. Borée, and F. Harambat, "Drag reduction of a car model by linear genetic programming control," Exp. Fluids **58**, 103 (2017).

[53]A. B. Blanchard, G. Y. Cornejo Maceda, D. Fan, Y. Li, Y. Zhou, B. R. Noack, and T. P. Sapsis, "Bayesian optimization for active flow control," Acta Mech. Sin. **37**, 1786 (2021).
[54]J.-L. Aider, "Closed-loop separation control using machine learning," J. Fluid Mech. **770**, 442–457 (2015).
[55]Y. Zhou, D. Fan, B. Zhang, R. Li, and B. R. Noack, "Artificial intelligence control of a turbulent jet," J. Fluid Mech. **897**, A27 (2020).
[56]C. Raibaudo, P. Zhong, B. R. Noack, and R. J. Martinuzzi, "Machine learning strategies applied to the control of a fluidic pinball," Phys. Fluids **32**, 015108 (2020).
[57]G. Y. C. Maceda, Y. Li, F. Lusseyran, M. Morzyński, and B. R. Noack, "Stabilization of the fluidic pinball with gradient-enriched machine learning control," J. Fluid Mech. **917**, A42 (2021).
[58]F. Pino, L. Schena, J. Rabault, and M. A. Mendez, "Comparative analysis of machine learning methods for active flow control," J. Fluid Mech. **958**, A39 (2023).
[59]A. V. Getling, *Rayleigh–Bénard Convection: Structures and Dynamics* (World Scientific Publishing Co. Private Ltd., 1998).
[60]J. Tang and H. H. Bau, "Stabilization of the no-motion state in the Rayleigh–Bénard problem," Proc. R. Soc. A **447**, 587–607 (1994).
[61]S. H. Davis, "The stability of time-periodic flows," Annu. Rev. Fluid Mech. **8**, 57–74 (1976).
[62]R. J. Donnelly, "Externally modulated hydrodynamic systems," in *Nonlinear Evolution of Spatio-Temporal Structures in Dissipative Continuous Systems*, edited by F. H. Busse and L. Kramer (Springer, 1990), pp. 31–43.
[63]R. E. Kelly, "Stabilization of Rayleigh–Bénard convection by means of a slow nonplanar oscillatory flow," Phys. Fluids A **4**, 647 (1992).
[64]R. M. Carbo, R. W. M. Smith, and M. E. Poese, "A computational model for the dynamic stabilization of Rayleigh–Bénard convection in a cubic cavity," J. Acoust. Soc. Am. **135**, 654–668 (2014).
[65]A. Swaminathan, S. L. Garrett, M. E. Poese, and R. W. M. Smith, "Dynamic stabilization of the Rayleigh–Bénard instability by acceleration modulation," J. Acoust. Soc. Am. **144**, 2334 (2018).
[66]J. Singer and H. H. Bau, "Active control of convection," Phys. Fluids A **3**, 2859 (1991).
[67]Y. Z. Wang, J. Singer, and H. H. Bau, "Controlling chaos in a thermal convection loop," J. Fluid Mech. **237**, 479 (1992).
[68]J. Tang and H. H. Bau, "Stabilization of the no-motion state in Rayleigh–Bénard convection through the use of feedback control," Phys. Rev. Lett. **70**, 1795–1798 (1993).
[69]J. Tang and H. H. Bau, "Feedback control stabilization of the no-motion state of a fluid confined in a horizontal porous layer heated from below," J. Fluid Mech. **257**, 485 (1993).
[70]J. Tang and H. H. Bau, "Stabilization of the no-motion state of a horizontal fluid layer heated from below with joule heating," J. Heat Transfer **117**, 329–333 (1995).
[71]T. A. Shortis and P. Hall, "On the effect of feedback control on Bénard convection in a Boussinesq fluid," Report No. NASA-CR-198280 (NASA, 1996).
[72]L. E. Howle, "Active control of Rayleigh–Bénard convection," Phys. Fluids **9**, 1861 (1997).
[73]L. E. Howle, "Linear stability analysis of controlled Rayleigh–Bénard convection using shadowgraphic measurement," Phys. Fluids **9**, 3111 (1997).
[74]J. Tang and H. H. Bau, "Experiments on the stabilization of the no-motion state of a fluid layer heated from below and cooled from above," J. Fluid Mech. **363**, 153–171 (1998).
[75]J. Tang and H. H. Bau, "Numerical investigation of the stabilization of the no-motion state of a fluid layer heated from below and cooled from above," Phys. Fluids **10**, 1597 (1998).
[76]L. E. Howle, "The effect of boundary properties on controlled Rayleigh–Bénard convection," J. Fluid Mech. **411**, 39–58 (2000).
[77]A. C. Or, L. Cortelezzi, and J. L. Speyer, "Robust feedback control of Rayleigh–Bénard convection," J. Fluid Mech. **437**, 175–202 (2001).
[78]A. C. Or and J. L. Speyer, "Active suppression of finite-amplitude Rayleigh–Bénard convection," J. Fluid Mech. **483**, 111–128 (2003).
[79]A. C. Or and J. L. Speyer, "Gain-scheduled controller for the suppression of convection at high Rayleigh number," Phys. Rev. E **71**, 046302 (2005).
[80]M. C. Remillieux, H. Zhao, and H. H. Bau, "Suppression of Rayleigh–Bénard convection with proportional-derivative controller," Phys. Fluids **19**, 017102 (2007).
[81]A. Pandey, J. D. Scheel, and J. Schumacher, "Turbulent superstructures in Rayleigh–Bénard convection," Nat. Commun. **9**, 2118 (2018).
[82]J. Kim, P. Moin, and R. Moser, "Turbulence statistics in fully developed channel flow at low Reynolds number," J. Fluid Mech. **177**, 133–166 (1987).
[83]J. Shen, T. Tang, and L.-L. Wang, *Spectral Methods - Algorithms, Analysis and Applications* (Springer-Verlag, Berlin, Heidelberg, 2011).
[84]U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, "Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations," Appl. Numer. Math. **25**, 151–167 (1997).
[85]M. Mortensen, "Shenfun: High performance spectral Galerkin computing platform," J. Open Source Software **3**, 1071 (2018).
[86]M. Mortensen, see https://shenfun.readthedocs.io for "Shenfun's Documentation."
[87]S. C. Reddy, P. J. Schmid, and D. S. Henningson, "Pseudospectra of the Orr–Sommerfeld operator," SIAM J. Appl. Math. **53**, 15–47 (1993).
[88]P. J. Schmid and D. S. Henningson, "Optimal energy density growth in Hagen–Poiseuille flow," J. Fluid Mech. **277**, 197–225 (1994).