A critical assessment of reinforcement learning methods for microswimmer navigation in complex flows

Selim Mecanna[1], Aurore Loisy[1*], Christophe Eloy[1*]

[1]Aix Marseille Univ, CNRS, Centrale Med, IRPHE, Marseille, France.

*Corresponding author(s). E-mail(s): aurore.loisy@irphe.univ-mrs.fr;
christophe.eloy@centrale-med.fr;
Contributing authors: selim.mecanna@centrale-med.fr;

Abstract

Navigating in a fluid flow while being carried by it, using only information accessible from on-board sensors, is a problem commonly faced by small planktonic organisms. It is also directly relevant to autonomous robots deployed in the oceans. In the last ten years, the fluid mechanics community has widely adopted reinforcement learning, often in the form of its simplest implementations, to address this challenge. But it is unclear how good are the strategies learned by these algorithms. In this paper, we perform a quantitative assessment of reinforcement learning methods applied to navigation in partially observable flows. We first introduce a well-posed problem of directional navigation for which a quasi-optimal policy is known analytically. We then report on the poor performance and robustness of commonly used algorithms (Q-Learning, Advantage Actor Critic) in flows regularly encountered in the literature: Taylor-Green vortices, Arnold-Beltrami-Childress flow, and two-dimensional turbulence. We show that they are vastly surpassed by PPO (Proximal Policy Optimization), a more advanced algorithm that has established dominance across a wide range of benchmarks in the reinforcement learning community. In particular, our custom implementation of PPO matches the theoretical quasi-optimal performance in turbulent flow and does so in a robust manner. Reaching this result required the use of several additional techniques, such as vectorized environments and generalized advantage estimation, as well as hyperparameter optimization. This study demonstrates the importance of algorithm selection, implementation details, and fine-tuning for discovering truly smart autonomous navigation strategies in complex flows.

Keywords: reinforcement learning, optimal navigation, partial observability, active particle, complex flow, POMDP

1 Introduction

The development of artificial microswimmers with navigation capabilities has been an intense topic of research in recent years [1–3]. When such robots with limited self-propulsion abilities are carried by a fluid flow, navigation becomes notoriously harder. This is the kind of challenge faced by robots deployed in the oceans for environmental monitoring purposes. Ideally, these drifting robots would be able to exploit background currents to travel more efficiently while relying only on data from their on-board sensors. The very same problem is also faced by plankton: these small organisms that drift with currents may be able to exploit hydrodynamic cues to migrate efficiently over long distances [4–6].

If the agent had global information about the flow, optimal control theory could be used to find optimal trajectories (a problem known as Zermelo's navigation problem [7]). But when the agent can only sense the flow *locally* (that is, has only access to a *partial observation*), optimal control theory can no longer be used. This problem becomes a model-free partially observable Markov decision process (POMDP). Such problems are usually well-suited for reinforcement learning, a data-driven alternative to optimal control that allows an agent to be trained at solving a task through repeated interactions with its environment.

In the last ten years, navigation in partially observable flows has attracted considerable attention in the fluid mechanics community [8–26], who adopted reinforcement learning techniques to develop "smart" navigation strategies. A variety of problems have been addressed, often inspired by biology. They include exploiting the flow to travel more efficiently [9, 11–13, 19], maintaining stable collective formations [8, 10, 15], catching a passive target [16], reducing chaotic dispersion [21], or targeting specific regions of the flow [14, 24, 25]. In parallel, various physical models of the agent have been used, ranging from simple self-propelled point particles to deformable bodies with fluid-structure interactions (e.g., [15, 22]).

A significant part of recent studies still use tabular Q-Learning, a classic learning algorithm [27] that has long been superseded by "deep" methods in the reinforcement learning community. A recent paper highlighted the limitations of such 'vanilla'[1] learning algorithms for discovering good strategies in complex flows, by showing that none of them could match the performance of a simple heuristic strategy obtained from physical intuition [22]. Therefore, the quality (with respect to optimality) of the learned strategies obtained so far in various navigation problems is uncertain. Previous work has shown that vanilla algorithms can find a solution to various navigation problems. But what does it take to find a *good* solution (close to optimality)?

In this paper, we provide a rigorous assessment of reinforcement learning as a tool to discover navigation strategies for microswimmers in complex flows. We compare three algorithms: two are representative of those used in prior work on smart microswimmers (Q-Learning [27] and Advantage Actor-Critic [28]), the last one is one the best modern algorithms for reinforcement learning and has demonstrated its capabilities across a wide range of domains (Proximal Policy Optimization [29]).

[1] 'vanilla' refers to the most basic, textbook implementation of an algorithm, without the use of any extra technique to improve on its performance

We benchmark these algorithms on a simple navigation problem in three different flows that are representative of those used in prior work: Taylor-Green vortices, Arnold-Beltrami-Childress flow, and two-dimensional unsteady turbulence.

Our study reveals that Q-Learning and A2C (Advantage Actor Critic) algorithms, despite being still routinely used for this purpose, actually perform rather poorly on navigation in partially observable flows. In contrast, we show that a custom implementation of PPO (Proximal Policy Optimization) allows learning a policy that matches the near-optimal performance. This work demonstrates that deep reinforcement learning is indeed a promising path toward autonomous navigation in flows, but only at the price of careful algorithm selection, implementation, and tuning.

The paper is organized as follows. We start in Section 2 by defining a well-posed benchmark navigation problem for which a near-optimal policy is known analytically, and introduce the three flow environments. In Section 3, we present the reinforcement learning algorithms used in this paper. In Section 4, we report on the performance and robustness of these algorithms in the three different flows. We conclude with a summary and discussion in Section 5.

2 Navigation in complex flows

2.1 Statement of the problem

We consider a swimming agent trying to go as far as possible in the upward $\hat{\boldsymbol{z}}$ direction. This task is representative of the diel vertical migration of plankton, and more generally of long-distance navigation to a target point (here moved to infinity) without the additional difficulties associated with sparse rewards.

The agent is modeled as an inertialess point-like particle swimming at a constant swimming speed v while being advected by the surrounding flow $\boldsymbol{u}(\boldsymbol{x}, t)$. For simplicity, we assume that the agent does not modify the background flow (one-way coupling), an approximation valid in the dilute limit as the agent perturbation to the flow is decreasing as an inverse power law. The agent can only control its swimming direction $\hat{\boldsymbol{p}}(t)$, a unit vector, every Δt (decision time). The readjustment of its swimming direction is instantaneous (no reorientation delay). Under these assumptions, the agent motion is governed by the following equation:

$$\boldsymbol{X}(t_{n+1}) = \boldsymbol{X}(t_n) + \int_{t_n}^{t_{n+1}} \left(\boldsymbol{u}(\boldsymbol{X}, t) + v\hat{\boldsymbol{p}}(t_n)\right) dt, \quad \boldsymbol{X}(t_0) = \boldsymbol{X}_0 \tag{1}$$

where $\boldsymbol{X}(t)$ is the position of the agent at time t, $\boldsymbol{u}(\boldsymbol{x}, t)$ is the flow velocity field (incompressible and with zero mean flow), and $\Delta t = t^{n+1} - t^n$. The agent initial position $\boldsymbol{X}_0$ is randomly initialized in the flow, and the starting time t_0 is also chosen randomly (when the flow is unsteady). The agent's swimming speed v is chosen as roughly half the typical flow speed (cf. Section 2.4). This choice ensures that the agent significantly drifts with the flow, while keeping learning computationally inexpensive for the purpose of this systematic benchmark.

In order to choose its swimming direction $\hat{\boldsymbol{p}}(t)$ at best, the agent has access to local flow information $\boldsymbol{G}(\boldsymbol{X}(t), t)$. This observable is chosen to be the local velocity

gradient tensor $\boldsymbol{\nabla u}(\boldsymbol{X}(t), t)$ (or a related quantity, depending on the flow considered, cf. Section 2.4). Indeed only flow *gradients*, rather than the flow itself, can be measured by an agent drifting with the flow.

The goal of the agent is to maximize the total distance traveled along the target direction $\hat{\boldsymbol{z}}$ over an episode of duration $T = t_f - t_0$, and averaged over all random initial conditions. We denote this metric Z, formally defined as

$$Z =< (\boldsymbol{X}(t_f) - \boldsymbol{X}_0) \cdot \hat{\boldsymbol{z}} > \tag{2}$$

where the brackets indicate the average. To summarize, we are looking for the control (called policy in reinforcement learning) $\hat{\boldsymbol{p}}(\boldsymbol{G})$ that maximizes the objective function Z under the dynamics given by Eq. (1).

This problem is simple enough to have a known analytical approximate solution (cf. next section) while retaining the complexity inherent to plankton-like navigation. For these reasons, it provides a well-posed benchmark problem for evaluating the capabilities of reinforcement learning applied to autonomous navigation in flows.

2.2 Analytical baselines

Two heuristic policies are considered as baselines: the naive policy and the surfing policy.

The *naive* policy consists in always swimming upward:

$$\hat{\boldsymbol{p}} = \hat{\boldsymbol{z}}, \tag{3}$$

resulting in average travelled distance $Z = vT$. This naive policy is a weak baseline, representative of baselines used in prior work applying vanilla reinforcement learning algorithms for navigation in flows.

The *surfing* policy is a recently proposed policy that has been shown to significantly improve upon the naive policy [6]. It reads

$$\hat{\boldsymbol{p}} = \boldsymbol{\lambda}/\|\boldsymbol{\lambda}\|, \qquad \boldsymbol{\lambda} = [\exp(\tau^* \boldsymbol{G})] \cdot \hat{\boldsymbol{z}} \tag{4}$$

where exp is the matrix exponential. The parameter τ^* has a physical meaning: it quantifies the mean correlation time of $\boldsymbol{G}$ as observed by the agent along its trajectory. For all practical purposes τ^* can be seen as a free parameter of the surfing policy that can be manually optimized for each flow (cf. Fig. A1). The surfing policy provides a strong baseline that reinforcement learning should at least match in order to be considered, in our view, as a suitable method for autonomous navigation in flows. Its name comes from its physical interpretation [6], as the agent 'surfs' on beneficial upward currents.

These two heuristics are, respectively, zeroth- and first-order analytical approximations of the optimal control for this problem, as obtained from Pontryagin's maximum principle. Indeed, our optimization problem can be reformulated as an

ordinary differential equation for the adjoint $\boldsymbol{\lambda}$:

$$\frac{d\boldsymbol{\lambda}(t)}{dt} = -\boldsymbol{\nabla}\boldsymbol{u}(\boldsymbol{X}(t), t) \cdot \boldsymbol{\lambda}(t), \quad \boldsymbol{\lambda}(t_f) = \hat{\boldsymbol{z}} \tag{5}$$

which solution is

$$\boldsymbol{\lambda}(t) = \left[\exp\left(\int_0^{t_f - t} \boldsymbol{\nabla}\boldsymbol{u}(\boldsymbol{X}(t+\tau), t+\tau) d\tau\right)\right] \cdot \hat{\boldsymbol{z}} \tag{6}$$

where $\boldsymbol{X}(t)$ is the solution of Eq. (1). In fluid flows, $\boldsymbol{\nabla}\boldsymbol{u}$ is generally time-correlated over a finite time τ^*. This allows us to approximate the integral in Eq. (6) by $\tau^* \boldsymbol{\nabla}\boldsymbol{u}(\boldsymbol{X}, t)$. The surfing policy, given by Eq. (4), immediately follows after replacing $\boldsymbol{\nabla}\boldsymbol{u}$ by $\boldsymbol{G}$. Note that neglecting the existence of correlations in the flow amounts to setting $\tau^* = 0$, which gives the naive policy.

2.3 Flows

We consider three different carrier flows, which are canonical flows commonly used in fluid mechanics: Taylor-Green vortices (TGV), Arnold-Beltrami-Childress flow (ABC), and two-dimensional unsteady turbulence (TURB). These flows provide training environments of increasing difficulty and realism, as they exhibit an increasing number of the key features of real flows: coherent structures (TGV, ABC, TURB), chaotic dynamics (ABC, TURB), and unsteadiness (TURB). In the following we define these flows using standard Cartesian coordinates (x,y,z), with z the coordinate of the target direction.

The TGV flow, illustrated in Fig. 1 (top left), consists of a lattice of counter-rotating vortices. It is an analytical steady solution to the 2D Navier-Stokes equations. It reads:

$$\begin{aligned} u_x &= -U\cos(x)\sin(z), \\ u_z &= U\sin(x)\cos(z) \end{aligned}$$

where we set $U = 0.5$. This flow has been used in, e.g., Refs [11, 19, 22].

The ABC flow, illustrated in Fig. 2 (top left), is a 3D steady flow characterized by coherent tube-like structures separated by a chaotic region. It is a steady solution to the three-dimensional Euler equations (a particular case of the Navier-Stokes equations with zero viscosity). It reads

$$\begin{aligned} u_x &= A\sin(z) + C\cos(y), \\ u_y &= B\sin(x) + A\cos(z), \\ u_z &= C\sin(y) + B\cos(x), \end{aligned}$$

where we set $A = \sqrt{3}$, $B = \sqrt{2}$ and $C = 1$. Similar ABC flows have been used in Refs [12, 14].

The TURB flow, illustrated in Fig. 3 (top left), is an unsteady, statistically stationary two-dimensional turbulent flow obtained by numerical simulation of the Navier-Stokes equations [30]. This multiscale chaotic flow features moving vortical structures that have a finite lifetime: they unpredictably appear, evolve and vanish. We simulated the flow evolution in the direct cascade regime using a standard pseudo-spectral solver on 256^2 collocation points and a large scale stochastic forcing. The characteristic flow velocity is $u_{\text{rms}} = 3.78$ and the characteristic time scale of the flow (eddy turn-over time) is $\tau_\omega = \omega_{\text{rms}}^{-1} = 0.11$ with $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$ the vorticity. Similar 2D turbulent flows have been used in Refs [16, 31].

All these flows are 2π-periodic in all directions: when the agent is at position $\boldsymbol{X}(t)$, the flow at this location is given by $\boldsymbol{u}(\boldsymbol{X}(t) \mod 2\pi, t)$.

2.4 Environment parameters

The three flows, together with the agent swimming speed v, the size of the time step Δt, the episode duration T, and the observable $\boldsymbol{G}$, define our three environments. The parameters used are summarized in Table 1.

In TGV, we set $v = u_{\max}/2$ and an episode consists of 4000 time steps. The observable is $\boldsymbol{G} = \boldsymbol{\nabla u}$. Due to symmetries, only two components of the velocity gradient are independent: these two components form the observation given to the agent, making the observation space two-dimensional.

In ABC, we set $v = u_{\max}/2$ and an episode consists of 2000 time steps. The observable is the anti-symmetric part of the velocity gradient: $\boldsymbol{G} = \frac{1}{2}\left(\boldsymbol{\nabla u} - \boldsymbol{\nabla u}^T\right)$, which three independent components are proportional to the components of vorticity $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$. The observation space is therefore three-dimensional. This choice of observable is motivated by consistency with prior work on ABC flow where vorticity was chosen [12]. Note that in ABC flow, $\boldsymbol{\omega}$ and $\boldsymbol{u}$ are equal up to a constant. The same observable $\boldsymbol{G}$, rather than full velocity gradient tensor, is also used for the surfing policy in this environment.

In TURB, the agent speed is set to $v \approx u_{\text{rms}}/2$, and an episode consists of 500 time steps (the typical time scale of the flow τ_ω is roughly 11 time steps). Turbulent simulation data has been generated for a total duration of 5000 time steps, split into 4000 for training and 1000 for testing. The initial time step that defines the start of an episode is chosen randomly in [0, 3500] for training and [4000, 4500] for testing. The observable is $\boldsymbol{G} = \boldsymbol{\nabla u}$. Due to flow incompressibility, only three components are independent, making the observation space three-dimensional.

Note that reducing the observation space to independent components is not key: using all the components of $\boldsymbol{G}$ yields identical results to those presented hereinafter.

3 Reinforcement learning methods

In the language of reinforcement learning and related domains, our navigation problem is a partially observable Markov decision process (POMDP). The agent has only access to an observation o ($\boldsymbol{G}$ at its position) of the underlying state s of the environment (the entire flow and the agent's position). The action a is the agent's swimming direction $\hat{\boldsymbol{p}}$, and the reward r is the distance traveled in the target direction between two successive

environment	v	u	Δt	T	$\boldsymbol{G}$	τ^*
TGV	0.25	0.50	0.01	40.0	$\{\partial_x u_x, \partial_x u_z\}$	2.0
ABC	1.5	3.0	0.01	20.0	$\{\omega_x, \omega_y, \omega_z\}$	0.72
TURB	2.0	3.78	0.01	5.0	$\{\partial_x u_x, \partial_x u_z, \partial_z u_x\}$	0.23

Table 1 Parameters of the three environments: agent speed v, characteristic flow velocity u (max value for TGV and ABC, root-mean-square value for TURB), decision time step Δt, duration of an episode T, observable $\boldsymbol{G}$, optimal value of the parameter τ^* of the surfing policy.

Table 2 POMDP framework applied to autonomous navigation.

POMDP variable	navigation variable
s_n	$\{\boldsymbol{X}(t_n), \boldsymbol{u}(\boldsymbol{X}, t)\,\forall t\}$
a_n	$\hat{\boldsymbol{p}}(t_n)$
o_n	$\boldsymbol{G}(\boldsymbol{X}(t_n), t_n)$
r_n	$[\boldsymbol{X}(t_{n+1}) - \boldsymbol{X}(t_n)] \cdot \hat{\boldsymbol{z}}$

time steps. Table 2 maps standard POMDP variables to the corresponding navigation variables. As is usual in reinforcement learning, we apply learning algorithms designed for MDP to our POMDP, assimilating the state to the observable, although there is no guarantee anymore that these algorithms will converge to the optimum policy. In the following, we introduce the three algorithms considered in this work: Q-Learning, A2C (Advantage Actor Critic), and PPO (Proximal Policy Optimization).

Q-Learning is a value-based method, where the state-action value function (or 'Q-function') is estimated and the policy is derived directly from it. It is an off-policy algorithm: the policy used to sample the environment is different from the learned policy (in practice, an ϵ-greedy version of the learned policy is used for sampling). In classical Q-Learning, the Q-function is a table, meaning that observations and actions must be discrete. To use this algorithm, we discretize every component of the observation vector o by categorizing each of them into three possible bins such that one third of the data sampled from each environment belongs to each bin. The actions are discretized into the four (six) Cartesian directions in 2D (3D), that is, $\pm\hat{\boldsymbol{z}}$ and the two (four) orthogonal directions. Using finer or coarser discretizations may affect the results, but our Q-Learning experiments are only intended to reproduce prior work, and similar discretizations were used in [11, 12, 16, 19–21]. We use an optimistic initialization of the Q-matrix to enhance exploration, this significantly improved the results compared to an initialization with zeros.

A2C is an actor-critic method: it combines policy-based methods (actor) and value-based methods (critic). Its name stems from the fact that the critic estimates the advantage function, rather than the state-action value function. It is an on-policy algorithm: the learned policy is used to sample the environment. The actor and critic are feedforward neural networks (see Table A1), which enable us to use continuous observation and action spaces. The output of the actor is not a Gaussian distribution as in most implementations, but a von Mises-Fisher distribution to appropriately

Table 3 Performance of the best learned policies (with 95% confidence intervals) in the three flow environments. The performance is defined as the average vertical distance traveled in an episode, normalized by the same quantity for the naive agent.

	PPO	A2C	QL	Surfing	Discrete Surfing	Naive
TGV	**1.62** ± 0.01	1.13 ± 0.01	1.22 ± 0.01	1.48 ± 0.01	1.47 ± 0.01	1.00 ± 0.01
ABC	**2.35** ± 0.03	**2.32** ± 0.03	1.9 ± 0.03	2.08 ± 0.03	2.01 ± 0.03	1.00 ± 0.03
TURB	**1.51** ± 0.01	1.20 ± 0.01	1.17 ± 0.01	**1.51** ± 0.01	1.39 ± 0.01	1.00 ± 0.01

represent the orientation of the agent (in 2D or 3D). The actor network is initialized such that initially, the output distribution is close to uniform. The main reference used in the implementation of A2C is the classical book of Sutton and Barto [32].

PPO also belongs to the actor-critic on-policy family of algorithms. Compared to A2C, it comes with multiple additional techniques to improve sample efficiency, learning stability, and thereby overall performance. In our implementation, which is inspired by implementations in Refs [33–35], we use policy loss clipping, vectorized architecture, generalized advantage estimation, advantage normalization, observation normalization, and training on fixed-length trajectory segments. The actor and critic networks are identical to those used for A2C.

For each environment and algorithm, we train ten times (ten random seeds) over 10^6 episodes. In the TURB environment, we use 80% of the simulation time for training, and the remaining 20% for testing (assessing the performance of the agent in unseen flow). Hyperparameters for each algorithm were tuned manually to achieve best performance. This tuning is essential as performance is highly sensitive to some of these hyperparameters. This is the case, for example, of the learning rates but also of the parameters related to generalized advantage estimation [36] (used in PPO). The hyperparameters we used are reported in Table A2.

4 Results

The agent's goal is to travel as far as possible in the vertical direction, by taking advantage of the partially observed carrier flow. Its performance is measured by Z, the vertical distance travelled over the course of an episode, averaged over all possible random initial conditions (Eq. 2). In the following, we will present the agent performance rescaled by the naive performance: $Z/Z_{\text{naive}} = Z/(vT)$.

4.1 Robustness over training trials

Figure 4 shows the beginning of the learning curves (over 10^5 episodes) of ten trials for each environment and algorithm. These learning curves allow us to assess the robustness of each algorithm. By robustness, we refer here to how repeatable are the training experiments.

Q-Learning has the largest variance across training trials. In both ABC and TURB, a single trial outperformed all the other ones. Therefore, the performance of the best agent is not easily reproducible, and strongly depends on a 'lucky' random seed. In

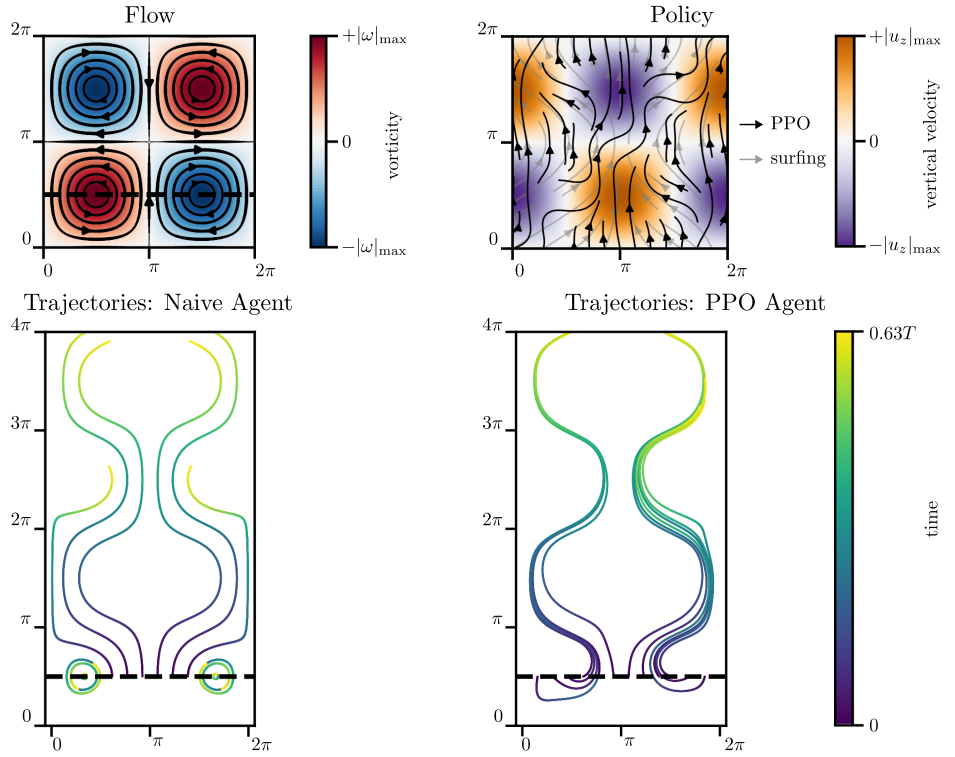

Fig. 1 Navigation in Taylor-Green vortices (TGV). The flow is represented in the upper left corner by showing the (out-of-plane component of the) vorticity ($\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$), along with streamlines. The dashed line represents the initialization of particles whose trajectories are shown in the bottom panel, for particles following the naive strategy (left) and the learned PPO strategy (right). Unlike naive agents which can be trapped on periodic orbits, PPO agents have learned to escape such trapping and all converge to a single trajectory that yields the largest vertical displacement, independently of their initial location. The PPO policy is compared to the surfing policy in the upper right corner. While both tend to diverge from downflow regions (violet) and converge to upflow regions (orange), PPO does it more aggressively, with steeper changes of direction.

general, Q-Learning learns fast, but it is very unstable. It often unlearns good strategies as shown by sudden decreases in performance.

A2C is found to be robust in TURB, but results were less reproducible in the other environments. While final performances are similar (with the exception of one trial in TGV where the agent did not learn anything), learning curves deviate strongly from one another. A2C tends to converge much more slowly than the other algorithms. This is due to the fact that both the actor and critic networks need to have small learning rates to ensure stability for this algorithm. The learning rates we used (cf. Table A2) are the largest ones allowing stable convergence.

In all the flows considered, PPO robustly reaches the same performance with very little variance across trials, compared to the other algorithms. In general, PPO converges quickly to the final solution. This is because PPO can handle high learning rates

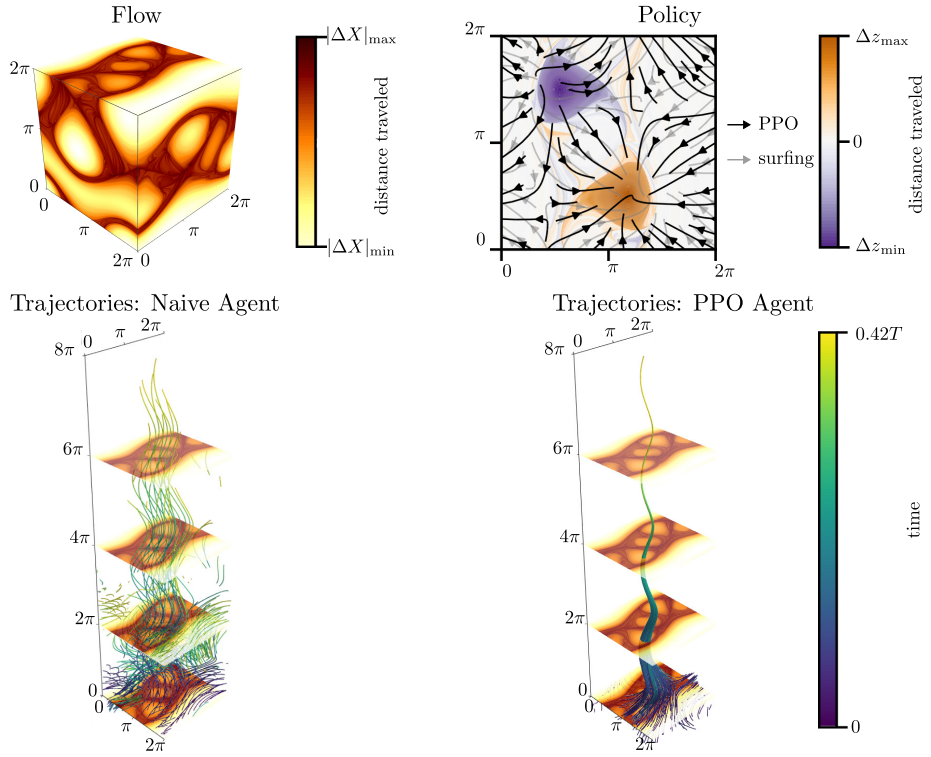

Fig. 2 Navigation in Arnold-Beltrami-Childress flow (ABC). The ABC flow is represented in the upper left corner by showing the total distance travelled by passive tracers advected by the flow (image generated with LDflow based on the LDDS package [37]). Such quantity, called Lagrangian descriptor [38], highlights flow regions with qualitatively different dynamics. This flow contains many coherent tube-like structure (light yellow) where tracers tend to cover large distances, these areas are also associated with preferential directions. They are separated by a chaotic region (dark red). In the bottom panels where trajectories are shown, agents are initialized at the $z = 0$ plane, following the naive strategy (left) and the PPO strategy (right). PPO agents have learned to converge to a particular flow structure, characterized by large upward transport. In the upper right corner, the surfing and PPO policies are projected onto a horizontal plane. The beneficial (detrimental) coherent structures are visible in the background: the orange (purple) one is associated to large upward (downward) displacement of passive tracers. Compared to surfing, PPO orients more aggressively toward the orange structure, which explains its overall superior performance.

for both the actor and the critic networks, as well as training over multiple epochs because the updated policy is guaranteed not to differ too much from the original one.

4.2 Performance of trained agents

Table 3 shows the performance of each algorithm in each environment. It is used to assess the ability of algorithms to discover good strategies in various flows. We selected the best agent for evaluation, that is, the one that achieved the highest performance at any point during training. Therefore, performance collapse during training does not affect this evaluation. In the TURB environment, we use the 20% portion of the simulation that was not used in learning to evaluate the performance.

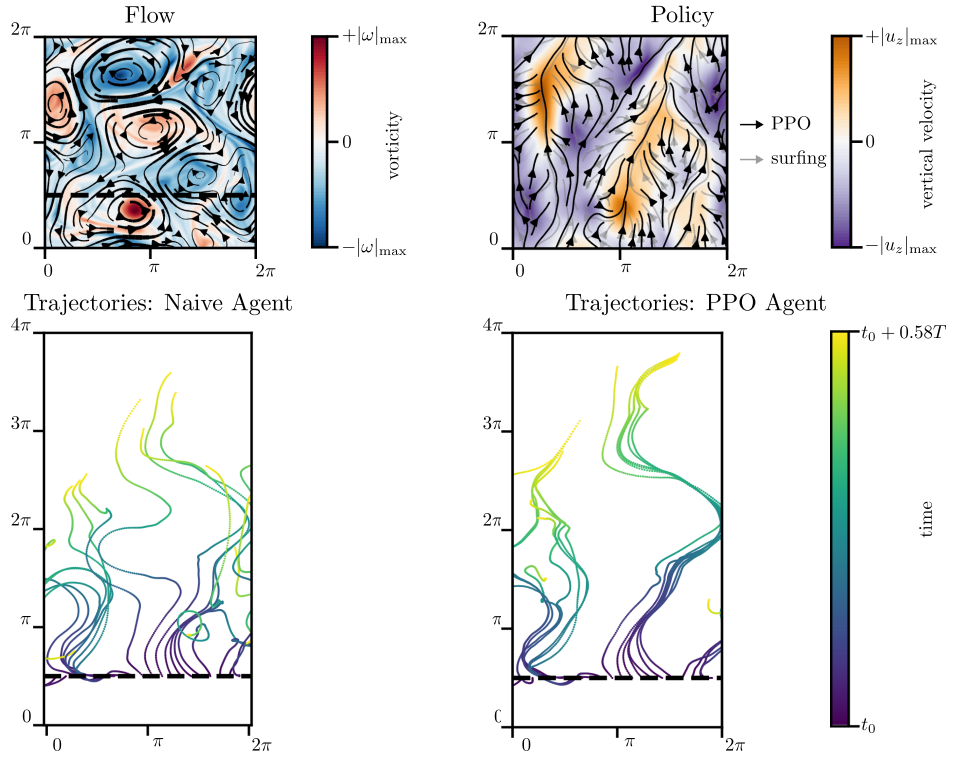

Fig. 3 Navigation in a two-dimensional turbulent flow (TURB). A snapshot of the time-dependent turbulence simulation is represented in the upper left: the (out-of-plane) vorticity is shown in the background, along with the streamlines. The snapshot corresponds to a randomly chosen time $t = t_0$ at which agents are initialized on the dashed line, their trajectories are shown in the bottom panel. Compared to the naive strategy (bottom left), the PPO strategy (bottom right) yields trajectories that tend to clump together to benefit from upward flow. This is visible in the policy representation (top right), where PPO is compared to surfing: both tend to diverge from downflow regions (violet) and converge to upflow regions (orange). While not strictly identical, these two policies are very similar to each other.

The evaluation was done using the deterministic version of the policies. For Q-Learning, the action chosen is the one corresponding to the highest Q-value. For A2C and PPO, instead of sampling from the von Mises-Fisher distribution that the network has converged to, we choose the action corresponding to the mean value of the distribution. Although not justified theoretically, this is common practice, and we found that the deterministic versions of the policies yield slightly better performance than the stochastic versions. Note that with PPO, the policies have essentially converged to deterministic ones, while this is not the case for A2C.

PPO is unambiguously the best-performing algorithm in all flow environments. It is the only algorithm able to outperform or match the performance of the surfing policy, our challenging baseline derived analytically, in all flows. In contrast, the strategies learned by A2C are far from optimal, except in ABC flow. As both PPO and A2C are actor-critic algorithms with identical networks, this performance gap illustrates the

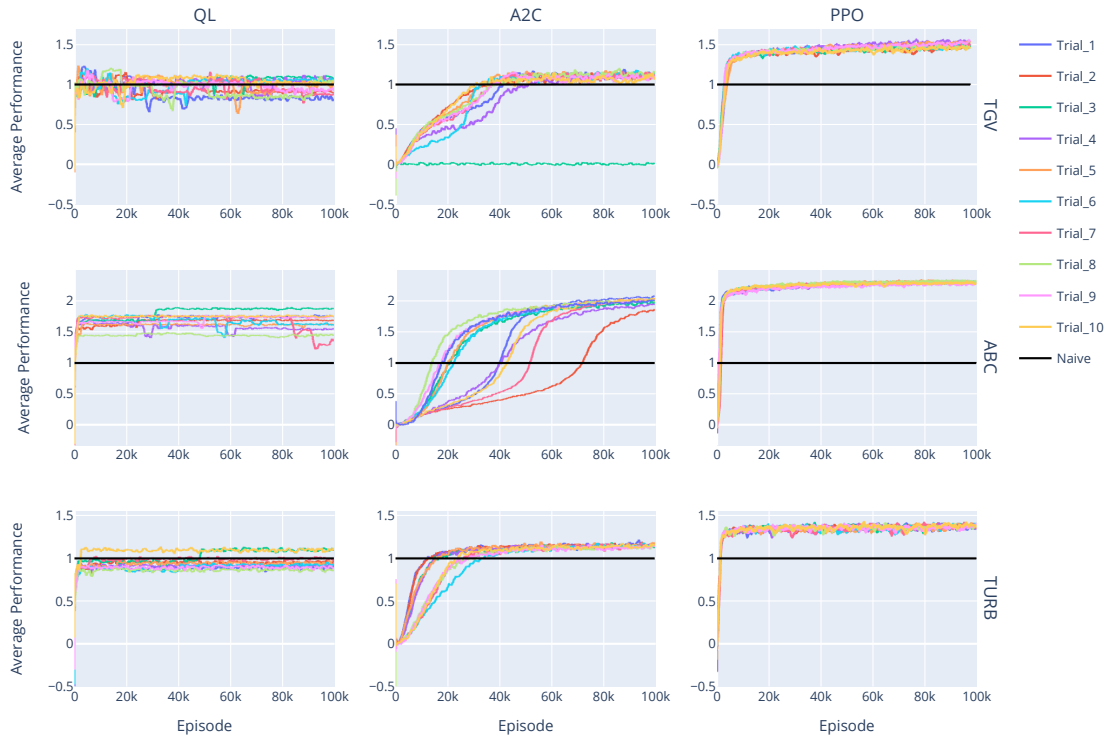

Fig. 4 Learning curves for each algorithm (columns) in each environment (rows): performance (distance traveled in the vertical direction normalized by vT) as a function of the number of episodes used for training. The performance of the naive policy is shown by a black line. To smooth out large episode-to-episode fluctuations and enhance readability, we use a moving averaging window of 1000 episodes. Only the first 10^5 episodes are shown, training is continued over a total of 10^6 episodes before evaluating performance in Table 3.

importance of using additional techniques (as in PPO) on top of vanilla algorithms (like A2C).

Q-Learning manages to learn better-than-naive strategies, as reported in most prior studies on microswimmer navigation. However, we are able to show here that these learned strategies are vastly suboptimal. Since Q-Learning requires discrete sets of observations and actions, we evaluated the surfing policy constrained to the same set of discrete actions (while keeping continuous observations, as discretizing observations would not make sense). Results are reported under 'Discrete Surfing' in Table 3, and show that, in all flows, Q-Learning is unable to learn a policy that matches this discrete version of the quasi-optimal strategy.

4.3 Interpretation of the strategies learned with PPO

We now comment on the best strategies learned for each flow, which have been obtained with the PPO algorithm.

In TGV (Fig. 1), naive swimmers can be trapped on periodic orbits. The occurrence of trapping, and therefore the performance of the naive agent in a given episode, is entirely determined by its initial position in the flow. In contrast, PPO agents forget their initial positions: they all converge to a single trajectory, the one that yields the largest vertical displacement by the background flow. This behaviour also prevents them from being trapped. The PPO policy and the surfing policy are similar: both diverge from downflow regions and converge to upflow regions. PPO does it more aggressively, with steeper changes of direction, resulting its slightly higher performance. We remark that, unlike surfing, the PPO policy is not symmetric with respect to symmetric inputs; this is common in reinforcement learning when no additional technique is used to enforce symmetries.

In ABC (Fig. 2), there exists a tube-like structure where passive tracers are trapped and are transported upward at a rapid rate (this structure is essentially an 'elevator' [12]). Therefore, the best strategy is to get into this structure as quickly as possible from the initial position, and then get essentially carried by the upward flow. This is exactly what the PPO agent has learned to do, and its higher performance compared to other agents is directly related to its ability to reach this structure faster than all the other strategies, on average.

In TURB (Fig. 3), the agent needs to find and stay in regions with upward flow, without overfitting to the specific flow used for training since flow structures are random and transient. The policy learned by PPO is very similar to the analytically derived one (surfing), though not identical. To interpret this difference, we trained an agent acting according to a generalized version of the surfing policy, where the parameter τ^* is an unknown function (represented by a neural network) of the input $\boldsymbol{G}$, rather than a constant. The learned $\tau^*(\boldsymbol{G})$ varies significantly with the input values, yet the performance of this agent at the task is identical to that of the original surfing policy (and that of the PPO agent). Furthermore, we found that the policy of this generalized-surfing agent is essentially identical to that of the PPO agent. In conclusion, PPO has learned a generalized version of surfing, with a variable parameter τ^*. We speculate that there is a family of functions $\tau^*(\boldsymbol{G})$ that perform as well as surfing in turbulent flows.

5 Conclusions

We have introduced a POMDP that models a navigation task relevant to robotic microswimmers and planktonic organisms. Despite its apparent simplicity, this task is challenging because it combines complex (possibly chaotic) state dynamics with partial observability. It is nevertheless well-posed mathematically and comes with a near-optimal analytical solution. It is therefore particularly well suited as a benchmark problem for a quantitative evaluation of reinforcement learning algorithms applied to navigation in partially observable flows.

We have implemented A2C and Q-Learning with similar features as in prior studies on navigation in flows, and shown that these algorithms perform poorly on this benchmark. In contrast, our custom implementation of PPO robustly achieves near-optimal theoretical performance. The satisfactory performance obtained with PPO is encouraging regarding the ability of reinforcement learning to discover and fully exploit flow features without having direct knowledge of them. We expect this version of PPO to be a good starting point for solving more challenging navigation tasks in partially observable flows (e.g., agents with memory).

These results highlight the importance of algorithm selection and implementation details when applying reinforcement learning to such navigation problems. We hope that these choices will be discussed more in the future, and that our study will encourage more quantitative assessments and comparisons. This could be done by comparing the performance obtained with various algorithms or by developing challenging heuristics as baselines.

The turbulent flow simulation considered here was modest to make this systematic benchmark feasible. Learning in very turbulent 2D flows and in 3D turbulent flows remains an open challenge. Analytical heuristics such as the surfing policy [6] or its recent generalizations [39, 40] provide strong baselines to which learned strategies should be compared to. As the cost of running the environment increases, PPO may become inefficient. Off-policy algorithms, such as SAC (Soft Actor Critic [41]) and TD3 (Twin-Delayed Deep Deterministic policy gradient [42]) should be considered and benchmarked for such problems where sample efficiency is likely to be of critical importance.

Partial observability is sometimes counteracted by providing the agent with some form of memory. While memory is unnecessary for the navigation task considered here, it is crucial to other navigation problems such as olfactory search in turbulent flows [43–47], a much harder problem on which reinforcement learning has started to be used [48–50]. It remains to be shown whether model-free, deep reinforcement learning is a viable tool for discovering good strategies in such memory-based navigation tasks in the presence of a realistic turbulent flow.

Acknowledgments. We thank Rémi Monthiller for fruitful discussions on optimal planktonic navigation, Jérémie Bec for his help with early developments of the DNS code for turbulent flow simulations, and Vladimír Krajňák for his help with visualizing coherent structures in ABC flow using Lagrangian descriptors. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 834238).

Statements and Declarations

Competing interests

The authors have no competing interests to declare.

Code and data availability

The code used for this study and the trained models are available at https://github.com/C0PEP0D/RLfl0w.

Authors' contributions

SM, AL and CE designed the study, analyzed the results and edited the manuscript. SM developed the code, performed numerical experiments and processed results. AL drafted the manuscript. CE obtained funding.

Appendix A Parameters used for the surfing policy, the actor-critic neural networks, and the reinforcement learning algorithms

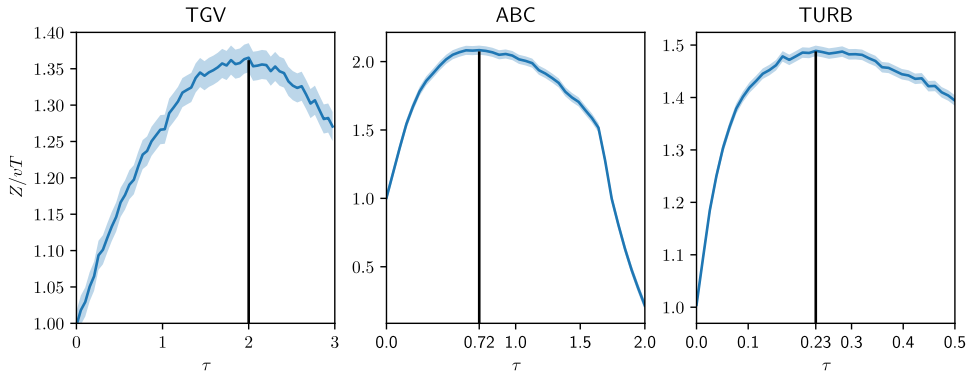

Fig. A1 Surfing policy: optimization over τ in the different flows. In TGV $\tau^* = 2.0$, in ABC $\tau^* = 0.72$, and in TURB $\tau^* = 0.23$

Table A1 Parameters of the actor and critic neural networks, used in both A2C and PPO. The two networks are independent (no shared layer).

Actor network	
Number of hidden layers	2
Neurons per hidden layer	40
Type of layers	Dense
Initialization	Glorot-uniform
Activation	ELU
Use feature normalization	True
Optimizer	Adam
Output distribution	von Mises-Fisher

Critic network	
Number of hidden layers	2
Neurons per hidden layer	100
Type of layers	Dense
Initialization	Glorot-uniform
Activation	ELU
Use feature normalization	True
Optimizer	Adam

Table A2 Hyperparameters used for the learning algorithms.

Hyperparameter of QL	TGV	ABC	TURB
Learning rate	0.8	0.8	0.8
Anneal learning rates	True	True	True
Epsilon (for ϵ-greedy exploration)	0.1	0.1	0.1
Discount factor	0.95	0.95	0.99

Hyperparameter of A2C	TGV	ABC	TURB
Learning rate actor	10^{-6}	10^{-6}	10^{-6}
Learning rate critic	10^{-4}	10^{-4}	10^{-4}
Anneal learning rates	False	False	False
Discount factor	0.95	0.99	0.99

Hyperparameter of PPO	TGV	ABC	TURB
Learning rate actor	10^{-4}	10^{-4}	10^{-4}
Learning rate critic	10^{-3}	10^{-3}	10^{-3}
Anneal learning rates	True	True	True
Discount factor	0.99	0.99	0.99
Number of parallel environments	100	10	10
Rollout length	10	10	100
GAE lambda	1.0	1.0	1.0
Number of minibatches	5	5	5
Epochs	4	4	4
Clip coefficient	0.1	0.1	0.1
Entropy coefficient	0.0	0.0	0.0
Target KL divergence	0.02	0.02	0.02

References

[1] B. Dai, J. Wang, Z. Xiong, X. Zhan, W. Dai, C. Li, S. Feng, J. Tang, Programmable artificial phototactic microswimmer. Nature Nanotechnology **11**, 1087–1092 (2016)

[2] H. Huang, F. Uslu, P. Katsamba, E. Lauga, M. Sakar, B. Nelson, Adaptive locomotion of artificial microswimmers. Science Advances **5** (2019)

[3] S. Muiños-Landin, A. Fischer, V. Holubec, F. Cichos, Reinforcement learning with artificial microswimmers. Science Robotics **6**(52) (2021). https://doi.org/10.1126/scirobotics.abd9285

[4] J. Wheeler, E. Secchi, R. Rusconi, R. Stocker, Not just going with the flow: The effects of fluid flow on bacteria and plankton. Annual Review of Cell and Developmental Biology **35**, 213–237 (2019)

[5] T. Kiørboe, E. Saiz, A. Visser, Hydrodynamic signal perception in the copepod Acartia tonsa. Marine Ecology Progress Series **179**, 97–111 (1999)

[6] R. Monthiller, A. Loisy, M.A.R. Koehl, B. Favier, C. Eloy, Surfing on Turbulence: A Strategy for Planktonic Navigation. Physical Review Letters **129**(6), 064502 (2022). https://doi.org/10.1103/PhysRevLett.129.064502

[7] E. Zermelo, Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung. Journal of Applied Mathematics and Mechanics **11**, 114–124 (1931)

[8] M. Gazzola, B. Hejazialhosseini, P. Koumoutsakos, Reinforcement Learning and Wavelet Adapted Vortex Methods for Simulations of Self-propelled Swimmers. SIAM Journal on Scientific Computing **36**(3), B622–B639 (2014). https://doi.org/10.1137/130943078

[9] G. Reddy, A. Celani, T.J. Sejnowski, M. Vergassola, Learning to soar in turbulent environments. Proceedings of the National Academy of Sciences of the United States of America **113**(33), E4877–84 (2016). https://doi.org/10.1073/pnas.1606075113. 27482099

[10] M. Gazzola, A.A. Tchieu, D. Alexeev, A. de Brauer, P. Koumoutsakos, Learning to school in the presence of hydrodynamic interactions. Journal of Fluid Mechanics **789**, 726–749 (2016). https://doi.org/10.1017/jfm.2015.686

[11] S. Colabrese, K. Gustavsson, A. Celani, L. Biferale, Flow Navigation by Smart Microswimmers via Reinforcement Learning. Physical Review Letters **118**(15), 158004 (2017). https://doi.org/10.1103/PhysRevLett.118.158004

[12] K. Gustavsson, L. Biferale, A. Celani, S. Colabrese, Finding efficient swimming strategies in a three-dimensional chaotic flow by reinforcement learning. The European Physical Journal E **40**(12), 110–110 (2017). https://doi.org/10.1140/epje/i2017-11602-9

[13] G. Reddy, J. Wong-Ng, A. Celani, T.J. Sejnowski, M. Vergassola, Glider soaring via reinforcement learning in the field. Nature **562**(7726), 236–239 (2018). https://doi.org/10.1038/s41586-018-0533-0

[14] S. Colabrese, K. Gustavsson, A. Celani, L. Biferale, Smart inertial particles. Physical Review Fluids **3**(8), 084301 (2018). https://doi.org/10.1103/PhysRevFluids.3.084301

[15] S. Verma, G. Novati, P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning. Proceedings of the National Academy of Sciences **115**(23), 5849–5854 (2018). https://doi.org/10.1073/pnas.1800923115

[16] J.K. Alageshan, A.K. Verma, J. Bec, R. Pandit, Machine learning strategies for path-planning microswimmers in turbulent flows. Physical Review E **101**(4), 043110–043110 (2020). https://doi.org/10.1103/PhysRevE.101.043110

[17] P. Gunnarson, I. Mandralis, G. Novati, P. Koumoutsakos, J.O. Dabiri, Learning efficient navigation in vortical flow fields. Nature Communications **12**(1), 7143 (2021). https://doi.org/10.1038/s41467-021-27015-y

[18] I. Mandralis, P. Weber, G. Novati, P. Koumoutsakos, Learning swimming escape patterns for larval fish under energy constraints. Physical Review Fluids **6**(9), 093101 (2021). https://doi.org/10.1103/PhysRevFluids.6.093101

[19] J. Qiu, N. Mousavi, K. Gustavsson, C. Xu, B. Mehlig, L. Zhao, Navigation of micro-swimmers in steady flow: The importance of symmetries. Journal of Fluid Mechanics **932** (2022). https://doi.org/10.1017/jfm.2021.978

[20] J. Qiu, N. Mousavi, L. Zhao, K. Gustavsson, Active gyrotactic stability of microswimmers using hydromechanical signals. Physical Review Fluids **7**(1), 014311 (2022). https://doi.org/10.1103/PhysRevFluids.7.014311

[21] C. Calascibetta, L. Biferale, F. Borra, A. Celani, M. Cencini, Taming Lagrangian chaos with multi-objective reinforcement learning. The European Physical Journal E **46**(3), 9 (2023). https://doi.org/10.1140/epje/s10189-023-00271-0

[22] Z. El Khiyati, R. Chesneaux, L. Giraldi, J. Bec, Steering undulatory micro-swimmers in a fluid flow through reinforcement learning. The European Physical Journal E **46**(6), 43 (2023). https://doi.org/10.1140/epje/s10189-023-00293-8

[23] K. Sankaewtong, J.J. Molina, M.S. Turner, R. Yamamoto, Learning to swim efficiently in a nonuniform flow field. Physical Review E **107**(6), 065102 (2023). https://doi.org/10.1103/PhysRevE.107.065102

[24] P. Gunnarson, J.O. Dabiri, Fish-inspired tracking of underwater turbulent plumes. Bioinspiration & Biomimetics **19**(5), 056024 (2024). https://doi.org/10.1088/1748-3190/ad7181

[25] N. Mousavi, J. Qiu, L. Zhao, B. Mehlig, K. Gustavsson, Short term vs. long term: Optimization of microswimmer navigation on different time horizons. Physical Review Research **7**(1), 013258 (2025). https://doi.org/10.1103/PhysRevResearch.7.013258

[26] Y. Jiao, H. Hang, J. Merel, E. Kanso, Sensing flow gradients is necessary for learning autonomous underwater navigation. Nature Communications **16**(1), 3044 (2025). https://doi.org/10.1038/s41467-025-58125-6

[27] C.J.C.H. Watkins, Learning from delayed rewards. Ph.D. thesis, University of Cambridge, England (1989)

[28] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, *Asynchronous Methods for Deep Reinforcement Learning*, in *Proceedings of The 33rd International Conference on Machine Learning* (PMLR,

2016), pp. 1928–1937

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Proximal Policy Optimization Algorithms (2017). https://doi.org/10.48550/arXiv.1707.06347

[30] G. Boffetta, R.E. Ecke, Two-Dimensional Turbulence. Annual Review of Fluid Mechanics **44**(1), 427–451 (2012). https://doi.org/10.1146/annurev-fluid-120710-101240

[31] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, K. Gustavsson, Zermelo's problem: Optimal point-to-point navigation in 2D turbulent flows using reinforcement learning. Chaos: An Interdisciplinary Journal of Nonlinear Science **29**(10), 103138–103138 (2019). https://doi.org/10.1063/1.5120370

[32] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, 2nd edn. (MIT Press, Cambridge, MA, 2018)

[33] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, A. Madry, *Implementation Matters in Deep RL: A Case Study on PPO and TRPO*, in *ICLR 2019 - Eighth International Conference on Learning Representations* (2019)

[34] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, O. Bachem, *What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study*, in *ICLR 2021 - Ninth International Conference on Learning Representations* (Vienna, Austria, 2021)

[35] H. Shengyi, D. Julien, R. Atonin, K. Anssi, W..I.B. Track. The 37 implementation details of proximal policy optimization (2022). URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/

[36] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, *High-Dimensional Continuous Control Using Generalized Advantage Estimation*, in *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)* (arXiv, San Juan, Puerto Rico, 2016). https://doi.org/10.48550/arXiv.1506.02438

[37] B. Aguilar-Sanjuan, V.J. García-Garrido, V. Krajňák, S. Naik, S. Wiggins, Ldds: Python package for computing and visualizing lagrangian descriptors for dynamical systems. Journal of Open Source Software **6**(65), 3482 (2021). https://doi.org/10.21105/joss.03482. URL https://doi.org/10.21105/joss.03482

[38] M. Agaoglou, B. Aguilar Sanjuan, V.J. García-Garrido, F. Gonzalez Montoya, M. Katsanikas, V. Krajňák, S. Naik, S.R. Wiggins, *Lagrangian Descriptors: Discovery and Quantification of Phase Space Structure and Transport* (Zenodo, 2020). https://doi.org/10.5281/zenodo.3958985

[39] C. Calascibetta, L. Biferale, F. Borra, A. Celani, M. Cencini, Optimal tracking strategies in a turbulent flow. Communications Physics **6**(1), 1–10 (2023). https://doi.org/10.1038/s42005-023-01366-y

[40] L. Piro, A. Vilfan, R. Golestanian, B. Mahault, Energetic cost of microswimmer navigation: The role of body shape. Physical Review Research **6**(1), 013274 (2024). https://doi.org/10.1103/PhysRevResearch.6.013274

[41] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*, in *Proceedings of the 35th International Conference on Machine Learning* (PMLR, Stockholm, Sweden, 2018), pp. 1861–1870

[42] S. Fujimoto, H. Hoof, D. Meger, *Addressing Function Approximation Error in Actor-Critic Methods*, in *Proceedings of the 35th International Conference on Machine Learning* (PMLR, Stockholm, Sweden, 2018), pp. 1587–1596

[43] M. Vergassola, E. Villermaux, B.I. Shraiman, "Infotaxis" as a strategy for searching without gradients. Nature **445**(7126), 406–409 (2007). https://doi.org/10.1038/nature05464

[44] A. Loisy, C. Eloy, Searching for a source without gradients: How good is infotaxis and how to beat it. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **478**(2262), 20220118 (2022). https://doi.org/10.1098/rspa.2022.0118

[45] A. Loisy, R.A. Heinonen, Deep reinforcement learning for the olfactory search POMDP: A quantitative benchmark. The European Physical Journal E **46**(3), 17 (2023). https://doi.org/10.1140/epje/s10189-023-00277-8

[46] G. Reddy, V.N. Murthy, M. Vergassola, Olfactory Sensing and Navigation in Turbulent Environments. Annual Review of Condensed Matter Physics **13**(1), 191–213 (2022). https://doi.org/10.1146/annurev-conmatphys-031720-032754

[47] A. Celani, E. Panizon, in *Target Search Problems*, ed. by D. Grebenkov, R. Metzler, G. Oshanin (Springer Nature Switzerland, 2024), pp. 711–732. https://doi.org/10.1007/978-3-031-67802-8_30

[48] K.V.B. Verano, E. Panizon, A. Celani, Olfactory search with finite-state controllers. Proceedings of the National Academy of Sciences **120**(34), e2304230120 (2023). https://doi.org/10.1073/pnas.2304230120

[49] S.H. Singh, F. van Breugel, R.P.N. Rao, B.W. Brunton, Emergent behaviour and neural dynamics in artificial agents tracking odour plumes. Nature Machine Intelligence **5**(1), 58–70 (2023). https://doi.org/10.1038/s42256-022-00599-w

[50] M. Rando, M. James, A. Verri, L. Rosasco, A. Seminara, Q-learning with temporal memory to navigate turbulence. eLife **13** (2025). https://doi.org/10.7554/eLife.102906.2