CME 216 Final Report

Jeremy P. Binagia

June 11, 2020

1 Introduction

In nature, microorganisms readily move in response to environmental cues, e.g. in the direction of nutrients (chemotaxis), towards light (phototaxis), in response to gradients in flow (rheotaxis), or along the direction of gravity (gravitaxis) [1, 2, 3]. Recently, researchers have become intersted in designing artificial particles that can respond to their environment to achieve specialized tasks such as targeted drug delivery [4, 5] or minimally-invasive surgery [6]. Reinforcement learning (RL) [7] is a natural tool for designing agents necessitating such complex navigation strategies; using this methodology, smart particles would learn optimal strategies through repeated interaction with their surrounding. Indeed, RL has been used to great success in the context of active matter[8], from teaching synthetic microswimmers how to propel at low Reynolds number [9] to understanding how schools of fish develop swimming patterns that conserve energy [10].

In this study, we will focus our attention on the recent application of RL to design "smart" gyrotactic particles that can navigate complex flow fields so as to increase their average elevation [11]. For a quiescent fluid, it is clear that the optimal navigation strategy is simply to swim vertically upward. For any non-trivial fluid, however, this "naive" strategy can become quite ineffective and lead to particles becoming trapped at a given height [12, 13, 14]. Hence, the question we seek to answer in this work is whether a microswimmer can be trained to use its knowledge of its orientation and the surrounding flow to develop a unique navigation policy that maximizes the swimmer's expected long-term vertical displacement.

2 Modeling gyrotactic swimming

To model gyrotactic swimming in a complex flow, we will consider a periodic array of Taylor-Green vortices spaced a distance L apart, the velocity of which is given by [15]:

$$\mathbf{u} = (u_0/2)[-\cos(2\pi x/L)\sin(2\pi z/L), \quad \sin(2\pi x/L)\cos(2\pi z/L)] \tag{1}$$

Immersed in this flow are microscopic swimming particles, meant to represent artificial microswimmers or perhaps true gyrotactic phytoplankton. Owing to their small size, the Reynolds number for these microswimmers is on the order of $10^{-5} - 10^{-4}$ [16]. Hence, in this study, we consider the swimmers to be point-like, neutrally-buoyant particles with negligible inertia such that the surrounding flow field is completely unperturbed by their presence. In this way, the kinematics of these swimmers can be determined directly (without solving the Stokes equations); in particular, each swimmer's translational velocity $\dot{\mathbf{x}}$ is given by:

$$\dot{\mathbf{x}} = \mathbf{u} + v_s\mathbf{p} \tag{2}$$

where $\mathbf{x}$ and $\mathbf{p}$ are the swimmer's position and instantaneous orientation. This equation is essentially a statement that each particle's velocity is a linear combination of the flow velocity $\mathbf{u}$ and the swimmer's individual motion (each particle moving at a constant speed v_s). Similarly, the angular velocity of each particle, $\dot{\mathbf{p}}$, for a flow with vorticity $\boldsymbol{\omega}$ is given by [17]:

$$\dot{\mathbf{p}} = \frac{1}{2B}[\mathbf{k_a} - (\mathbf{k_a}\cdot\mathbf{p})\mathbf{p}] + \frac{1}{2}\boldsymbol{\omega}\times\mathbf{p} \tag{3}$$

Here, $\boldsymbol{k_a}$ is the swimmer's target orientation, and B is the timescale to align to $\mathbf{k_a}$. Hence, the swimmer rotates as it attempts to steer in the preferred direction $\boldsymbol{k_a}$ and as it is rotated by the surrounding vortices.

We can write these equations in dimensionless form by considering our characteristic scales for length, velocity, vorticity, and time to be $1/m$, u_0, ω_0, and $1/\omega_0$; $m = 2\pi/L$ is the wavenumber of the periodic array of vortices and $\omega_0 = 2\pi u_0/L$ denotes the maximum vorticity. With this scaling, we may rewrite our governing equations for this problem as (with an overbar denoting a dimensionless quantity):

$$\bar{\mathbf{u}} = (1/2)[-\cos\bar{x}\sin\bar{z}, \quad \sin\bar{x}\cos\bar{z}] \tag{4}$$

$$\dot{\bar{\mathbf{x}}} = \Phi\bar{\mathbf{u}} + \mathbf{p} \tag{5}$$

$$\dot{\mathbf{p}} = \frac{1}{2\Psi}[\mathbf{k_a} - (\mathbf{k_a}\cdot\mathbf{p})\mathbf{p}] + \frac{1}{2}\bar{\boldsymbol{\omega}} \times \mathbf{p} \tag{6}$$

From this analysis, we see two important dimensionless groups emerge. The first, $\Phi = v_s/u_0$, is a measure of the swimmer's speed relative to that of the background flow. The second group, $\Psi = B\omega_0$, is a ratio of the timescale to be reoriented by steering to that of vorticity. With the mathematical model for our problem now defined, we proceed to describe how RL was leveraged to train smart gyrotactic particles.

3 Methodology

3.1 Defining the reinforcement learning problem

In short, a RL algorithm seeks to determine the optimal action (policy) to take in a given state so as to achieve a specified long-term goal. In our specific problem, we will consider the total state space S to be the product of three coarse-grained vorticity states $S_\omega = \{\omega_-, \omega_0, \omega_+\}$ (i.e. large and negative, close to zero, large and positive) and four coarse-grained particles orientations $S_p = \{\rightarrow, \downarrow, \leftarrow, \uparrow\}$ such that $S = S_\omega \times S_p$. An "action" in this context consists of changing the preferred swimming direction, which can take on values given by $\mathbf{k}_a \in A = \{\rightarrow, \downarrow, \leftarrow, \uparrow\}$. After each action, the agent (i.e. the swimmer) moves from the present state s_n to a new state s_{n+1} and evaluates the immediate success of said action by computing a reward $r_{n+1} = z(s_{n+1}) - z(s_n)$ (i.e. its change in elevation). Ultimately, the goal of any RL algorithm is to find the optimal policy $\pi_*(a|s)$ giving the best action a to take given state s so as to maximize some long-term reward. In our case, this long-term reward is defined as the expected total vertical displacement for a swimmer taking N_s discrete steps during a give training segment (episode), E:

$$R_{tot} = \left\langle \sum_{n=1}^{N_s} r_n \right\rangle \tag{7}$$

Here $< \cdot >$ denotes the expected value, taken over different initial positions and orientations of the swimmer for an episode. Operationally, the expected value in eq. (7) is estimated using a moving average of the individual estimates of R_{tot} coming from each training episode (as opposed to say an ensemble average).

3.2 Q-learning

One of the simplest and most well-known RL algorithms is that of Q-learning [18]. As is the case for all temporal-difference (TD) methods, the goal of Q-learning is to compute the optimal action-value function $Q^*(s,a)$ from which the optimal policy directly follows; in general, the action-value function $Q^\pi(s,a)$ gives the expected return for taking action a given state s and following policy π thereafter [7]. TD methods attempt to obtain the values of the matrix $Q^*(s,a)$ through iterative updates to their most recent estimate $Q(s,a)$. The iterative update rule for Q-learning is:

$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha(r_{n+1} + \gamma \max_a Q(s_{n+1}, a) - Q(s_n, a_n)) \tag{8}$$

where α is the learning rate. $0 \leq \gamma < 1$ is the discount factor and controls the extent to which potential future gains should be weighed against immediate rewards. In our study, we select $\gamma = 0.999$.

To select the new action at step n, we use an ϵ-greedy approach, whereby with probability $1-\epsilon$ (ϵ being a small positive number less than 1) we select the greedy action a' for state s, i.e. $a' = \arg\max_a Q(s,a)$, and otherwise select a random action (selected uniformly at random with probability $1/|A|$). Note that because this behavior policy is in general different from the estimation policy (eq. (8)), Q-learning is referred to as an "off-policy" method [7]. The advantage of an off-policy method such as this is that a behavior policy promoting exploration can be achieved by setting $\epsilon > 0$, thereby ensuring that each state-action pair is visited a sufficient number of times, but that still Q will converge to the optimal *greedy* policy Q^*.

3.3 Expected Sarsa

If the term $\max_a Q(s_{n+1}, a)$ were replaced by $Q(s_{n+1}, a_{n+1})$ in eq. (8), then we would obtain the update equation for Sarsa, the classic on-policy alternative to Q-learning [19, 20]. Note that because Sarsa is on-policy, its estimate of Q will not converge to Q^* so long as an exploratory policy is used (which as we mentioned is necessary to visit each state a sufficient number of times). Typically what is done to remedy this is to decrease ϵ over time, e.g. selecting $\epsilon = 1/n$ so that the greedy policy is recovered in the limit.

The issue with Sarsa, however, is that it introduces additional variance into the update equation as a result of the stochastic behavior (and hence estimation) policy that it must utilize for exploration. To remedy this, the Expected Sarsa method was developed [21]; the insight here is to replace $Q(s_{n+1}, a_{n+1})$ in the update rule for Sarsa with its expected value $\langle Q(s_{n+1}, a_{n+1})\rangle$:

$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha(r_{n+1} + \gamma \sum_a \pi(a|s_{n+1}) Q(s_{n+1}, a_n) - Q(s_n, a_n)) \tag{9}$$

where $\pi(a|s_{n+1})$ is the probability of selecting action a given state s_{n+1} under policy π. In essence, the effect of this change is that the update for Expected Sarsa moves deterministically in the same direction that Sarsa moves in expectation. In this way, variance resulting from the policy is removed, allowing one to utilize a larger learning rate; in fact, in a purely deterministic environment the variance is zero and the maximal value of $\alpha = 1$ may be used. The connection of Expected Sarsa to Q-learning may be seen by considering the case where the policy π is different from the behavior policy such that Expected Sarsa becomes off-policy. In particular, consider the purely greedy policy whereby $\pi(a|s) = 0$ for all a except for the action where Q obtains its maximal value; in this instance, eq. (9) reduces precisely to eq. (8). Hence Expected Sarsa may be viewed as both a low-variance version of Sarsa or as a generalization of Q-learning. In terms of performance, Expected Sarsa tends to outperform Q-learning and Sarsa at least empirically [21, 7].

3.4 Specific training considerations

For each of the cases described in section 4, the Expected Sarsa method was used with a learning rate $\alpha = 1$ and an ϵ-greedy policy with $\epsilon = 1/n$. By gradually reducing ϵ over time, we allow for exploration initially while also ensuring Q converges to Q^*. Exploration is also promoted by initializing the entries of the Q matrix to large initial values, in this case setting them all equal to $L_z N_s$ where $L_z = 2\pi$ is the size of the periodic box whose domain is given by $[0, 2\pi] \times [0, 2\pi]$. This trick, sometimes referred to as "optimistic initial values" ensures that all states are visited multiple times during the beginning of the learning process [7]. Training itself is split into $N_e = 5000$ episodes each consisting of $N_s = 4000$ discrete steps, with each episode beginning with the final Q matrix obtained at the end of the previous episode and a random initialization of particle positions and orientations. Each step begins by selecting a new action ϵ-greedily, after which a new state is entered and a reward is issued. Operationally, this means integrating eq. (5) and eq. (6) to obtain a new particle position and orientation; for simplicity and to minimize computational cost, the Explicit Euler method is used with a timestep $\Delta t = 0.01$. The matrix Q is then updated according to eq. (9), at which point a new step may be taken. The ultimate performance of the smart and naive particles are compared using the expected total return (eq. (7)) and the learning gain Σ, defined as $\Sigma = R_{tot}/R_{tot,g} - 1$ where $R_{tot,g}$ is the expected return for a particle exhibiting naive gyrotaxis.

3.5 Computational implementation

The methodology described above was implemented from scratch and may be found at `https://github.com/jbinagia/cme216-final-project`. Notably, the code used for the three training experiments described below may be found in `phi_0d3_psi_0d3.ipynb`, `phi_0d3_psi_1.ipynb`, and `phi_0d3_psi_3.ipynb` respectively, with the source code for function and class definitions given in `main.ipynb`. A combination of unit tests (e.g. explicit tests of class methods such as `update_position`) and sanity checks (e.g. ensuring trivial vertical migration is obtained in the limit of $\Phi \gg 1$ and $\Psi \ll 1$) were conducted to ensure the veracity of the implementation. Furthermore, qualitative comparisons of particle trajectories to the phase map given in fig. 1 of Colabrese et al. [11] were favorable, thereby providing a validation of our approach. Despite this, the learning gains seen here are in general smaller than those found in that work (for identical values of N_s, N_e, α, etc. which are specified in their SI). This difference in performance could be a result of computing R_{tot} using actual ensembles of many particles (which proved to be impractical given the computational resources presently available to the author) or specific training considerations not stated explicitly (e.g. a custom learning rate schedule or exploration strategy).

4 Results and discussion

Using the methodology described above, we conducted RL experiments for a range of physical parameter values. In particular, we focus on the regime corresponding to $\Phi \ll 1$. Firstly, the limit of $\Phi \ll 1$ is the most interesting for our application since here navigation becomes quite difficult; this is a result of the particles primarily being advected by the flow due to their relatively small swimming speeds. In contrast, for $\Phi \gg 1$ and $\Psi << 1$, for example, navigation becomes trivial. Here the optimal strategy is simply to always steer upward since the vortices cannot adequately reorient particles away from their preferred orientation. Furthermore, the $\Phi \ll 1$ is of interest since this is often the range relevant to true swimming microorganisms; for example, plankton swimming near the ocean surface have $\Phi \sim 0.03 - 1$ [22].

Meanwhile, we will consider a range of values for Ψ to see how swimming strategies might change as the ratio of viscous to active torques is adjusted. We will see that the behaviors that emerge for each value of Ψ will be quite different despite Ψ only spanning a decade. This can be appreciated, for example, by viewing the entries of the final Q matrices for each parameter set; these matrices are displayed as heatmaps in figs. 4, 7 and 10. The approximately optimal strategies that we find in this way are quite nontrivial and would be hard to guess or design a priori. In fact, of the three Q matrices given in figs. 4, 7 and 10, selecting "up" as the preferred direction is only the selected action at most 33.3% of the time assuming a uniform distribution of states. It is for this reason that RL may prove to be a very useful tool in the design of artificial swimmers.

4.1 Weak swimming and quick realignment ($\Phi = 0.3$, $\Psi = 0.3$)

We begin by considering the case of relatively weak swimming ($\Phi = 0.3$) and ample time for particle realignment ($\Psi = 1.0$). The performance of the smart gyrotactic particle compared to that of the naive particle can be seen in fig. 5; the total reward for each episode, R_{tot}, is shown on the left while the learning gain Σ is plotted on the right. Note that each point on these curves was generated using a moving average with a window size of 1000 episodes to ameliorate the environmental stochasticity originating from the random initialization of position and orientation. From this pair of plots we see that the smart particle clearly outperforms the naive particle, with a final learning gain of approximately 70%. The gain in performance over time can be appreciated by viewing fig. 2; here, trajectories of the smart (blue) and naive (red) gyrotactic particles are shown for two episodes in the learning process. For early episodes (e.g. episode 2000), both particles easily run the risk of becoming trapped in vortex cores. Through repeated episodic experiences; however, the smart particle can develop a strategy that seemingly allows it to not only escape these vortices but also to consistently avoid them as it rises diagonally upward (c.f. the right-hand side of fig. 2).

What is the strategy learned by the smart particle that enables it to outperform the naive particle? To help answer this, we plot the smart particle's trajectory for episode 5000 in fig. 3. On the left-hand side, the trajectory is colored by the action taken at each state, i.e. what direction it selects as its desired orientation. Meanwhile, the right-hand side of the plot colors this same trajectory using the particle's instantaneous orientation, measured in degrees from the positive x-axis. What is perhaps most interesting from this set of plots is that the optimal steady-state action appears to be steering rightward (green), which when combined with the surrounding flow enables a steady orientation of approximately 130° (i.e. almost directly northwest). To reach this steady trajectory and escape the initial vortex, the particle sets its target direction first left and then right so as to efficiently move counterclockwise around the first vortex. The naive strategy of selecting $\mathbf{k_a} = \hat{\mathbf{z}}$ is used to move around the first vortex but afterwards is used sparingly, if at all.

4.2 Weak swimming and moderate realignment time ($\Phi = 0.3$, $\Psi = 1.0$)

We now consider the case where swimming is again relatively weak ($\Phi = 0.3$), but the stabilizing active torque is balanced by the destabilizing viscous torque ($\Psi = 1.0$). While the total episodic rewards R_{tot} are overall smaller is magnitude than those for $\Phi = 0.3$ and $\Psi = 0.3$, the relative performance of the smart particle to that of the naive swimmer is much greater in this case with a final learning gain Σ of $\approx 120\%$.

Because vorticity can readily reorient the swimming particles at this value of $\Psi = 1$, trajectories are typically more complex than those seen in section 4.1. We can see this in fig. 6, where we have again visualized the selected actions and orientation for a smart particle's trajectory post-training. For these figures, the orange curve denotes the trajectory of a naive gyrotactic particle sharing the same initial condition and orientation. From this set of plots we observe several interesting behaviors. For example, the smart particle begins its trajectory by attempting to steer left, against the flow. While counter-intuitive, this appears to rotate the particle clockwise until it obtains a nearly vertical orientation, at which point it sets $\mathbf{k_a} = \hat{\mathbf{z}}$, presumably to escape the vortex and begin moving upwards. The maneuver of setting $\mathbf{k_a}$ counter to the flow direction appears to occur every time the particle is situated between two vortices in the vertical direction.

Finally, it is interesting to note the sparsity of the upward action in the final policy of this particle; this can be seen in fig. 6 and fig. 7, a heatmap of the particle's Q matrix. For example, viewing the normalized Q matrix shown on the left of fig. 7, we see that moving upward is the selected action (assuming a greedy policy) only 16.7% of the time (2 out of 12 total states). It should be noted though that state-action pairs for which the upward move is selected have some of the highest expected returns, i.e. $(\omega_0, \leftarrow)$ and $(\omega_-, \rightarrow)$.

4.3 Weak swimming and slow realignment ($\Phi = 0.3$, $\Psi = 3.0$)

The final experiment we consider is relatively weak swimming ($\Phi = 0.3$) with a strong destabilizing viscous torque ($\Psi = 3.0$). As one might expect, navigation becomes particularly difficult as Ψ is increased, with the particle orientation continually being changed by the flow. Indeed, in the limit of $\Psi \gg 1$, particles undergo random, undirected motion owing to their extremely small realignment time. We see this difficulty directly in fig. 8; the values of R_{tot} for each episode are the smallest we have seen thus far and the final gain is $\approx 50\%$.

As perhaps was anticipated, the trajectories for $\Psi = 3.0$ are the most irregular of those considered in this study. To explore the final policy of the smart gyrotactic particle, we visualized its trajectory alongside its selected actions and instantaneous orientation in fig. 9. Even though Ψ has only been increased by a factor of 3 compared to that in the previous subsection, the swimming strategy appears to be quite different now. For example, when situated by vortices above and below, the policy here appears to be either to try swimming up or to the left (as opposed to orienting opposite the streamlines as before). What is particularly interesting about this example is that $\mathbf{k_a} = \hat{\mathbf{x}}$ is selected as the particle moves alongside the left-side of the first vortex that is encountered. This would seem to be counterproductive, moving the particle into the center of the vortex, and indeed the naive particle undergoes this very mistake (albeit later in its trajectory). The smart particle, however, adjusts its preferred direction first leftward and then rightward so as to rotate clockwise and inevitably escape this vortex to ascend upward. This complex maneuver would be difficult to intuit a priori, and suggests the utility of the RL approach.

5 Conclusion and future work

To summarize, in this study we have designed smart gyrotactic swimmers through the use of RL that maximize their long-term vertical displacement by selecting a preferred swimming direction from a coarse idea of their current orientation and flow vorticity. We focused on the biologically relevant $\Psi \ll 1$ regime and considered a range of values of Ψ to see how navigation strategies are affected by the competing self-imposed active torque and the viscous torque applied by the surrounding flow. Besides conducting experiments for parameter values not considered in this report (i.e. $\Phi \gg 1$ or nonzero translational and rotational diffusivities), future work would likely involve relaxing some of the assumptions that were made. For example, one might consider how larger particles with finite inertia might directly interact with and change the flow velocity; alternatively, we could allow for fluids that exhibit non-Newtonian behavior (e.g. viscoelasticity or shear-thinning rheology). It is likely that the emergent strategies under these conditions would be much different than those described in this report and hence would make for an interesting avenue of research.

References

[1] Julius Adler. “Chemotaxis in bacteria”. In: *Annual review of biochemistry* 44.1 (1975), pp. 341–356.

[2] D-P Häder. “Polarotaxis, gravitaxis and vertical phototaxis in the green flagellate, Euglena gracilis”. In: *Archives of microbiology* 147.2 (1987), pp. 179–183.

[3] Kiyoshi Miki and David E Clapham. “Rheotaxis guides mammalian sperm”. In: *Current Biology* 23.6 (2013), pp. 443–452.

[4] Debabrata Patra et al. “Intelligent, self-powered, drug delivery systems”. In: *Nanoscale* 5.4 (2013), pp. 1273–1283. ISSN: 2040-3364. DOI: 10.1039/c2nr32600k. URL: http://xlink.rsc.org/?DOI=C2NR32600K.

[5] Wei Gao and Joseph Wang. “Synthetic micro/nanomotors in drug delivery”. In: *Nanoscale* 6 (2014), pp. 10486–10494. ISSN: 2040-3364. DOI: 10.1039/C4NR03124E. URL: http://dx.doi.org/10.1039/C4NR03124E.

[6] Xiaohui Yan et al. “Multifunctional biohybrid magnetite microrobots for imaging-guided therapy”. In: *Science Robotics* 2.12 (2017), pp. 1–15. ISSN: 24709476. DOI: 10.1126/scirobotics.aaq1155.

[7] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[8] Frank Cichos et al. “Machine learning for active matter”. In: *Nature Machine Intelligence* 2.2 (2020), pp. 94–103.

[9] Alan Cheng Hou Tsang et al. “Self-learning how to swim at low Reynolds number”. In: *arXiv preprint arXiv:1808.07639* (2018).

[10] Mattia Gazzola, Babak Hejazialhosseini, and Petros Koumoutsakos. “Reinforcement learning and wavelet adapted vortex methods for simulations of self-propelled swimmers”. In: *SIAM Journal on Scientific Computing* 36.3 (2014), B622–B639.

[11] Simona Colabrese et al. “Flow navigation by smart microswimmers via reinforcement learning”. In: *Physical review letters* 118.15 (2017), p. 158004.

[12] William M Durham, Eric Climent, and Roman Stocker. “Gyrotaxis in a steady vortical flow”. In: *Physical Review Letters* 106.23 (2011), p. 238102.

[13] William M Durham et al. “Turbulence drives microscale patches of motile phytoplankton”. In: *Nature communications* 4.1 (2013), pp. 1–7.

[14] Francesco Santamaria et al. “Gyrotactic trapping in laminar and turbulent Kolmogorov flow”. In: *Physics of Fluids* 26.11 (2014), p. 111901.

[15] GI Taylor. “LXXV. On the decay of vortices in a viscous fluid”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 46.274 (1923), pp. 671–674.

[16] Edward M Purcell. “Life at low Reynolds number”. In: *American journal of physics* 45.1 (1977), pp. 3–11.

[17] TJ Pedley and John O Kessler. “Hydrodynamic phenomena in suspensions of swimming microorganisms”. In: *Annual Review of Fluid Mechanics* 24.1 (1992), pp. 313–358.

[18] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[19] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[20] Richard S Sutton. “Generalization in reinforcement learning: Successful examples using sparse coarse coding”. In: *Advances in neural information processing systems*. 1996, pp. 1038–1044.

[21] Harm Van Seijen et al. “A theoretical and empirical analysis of Expected Sarsa”. In: *2009 ieee symposium on adaptive dynamic programming and reinforcement learning*. IEEE. 2009, pp. 177–184.

[22] K Gustavsson et al. “Preferential sampling and small-scale clustering of gyrotactic microswimmers in turbulence”. In: *Physical review letters* 116.10 (2016), p. 108104.

List of Figures

1 Total episodic reward and percent learning gain for $\Phi = 0.3$, $\Psi = 0.3$. 8

2 Selected trajectories for the smart (blue) and naive (red) gyrotactic particles for $\Phi = 0.3$, $\Psi = 0.3$. Darker colors denote later points in time. 8

3 Selected action (left) and orientation θ (right) for the smart gyrotactic particle swimming during the final training episode at $\Psi = 0.3$, $\Phi = 0.3$ 8

4 Normalized (left) and unnormalized (right) Q matrix for $\Phi = 0.3, \Psi = 0.3$. The normalized matrix is obtained by scaling each row of Q independently such that it ranges from 0 to 1. . . . 9

5 Total episodic reward and percent learning gain for $\Phi = 0.3$, $\Psi = 1.0$. Each data point is a local average of data from $N = 1500$ episodes 9

6 Selected action (left) and orientation θ (right) for the smart and naive (orange) gyrotactic particles swimming for an episode post-training at $\Psi = 0.3$, $\Phi = 1.0$ 9

7 Normalized (left) and unnormalized (right) Q matrix for $\Phi = 0.3, \Psi = 1.0$. The normalized matrix is obtained by scaling each row of Q independently such that it ranges from 0 to 1. . . . 10

8 Total episodic reward and percent learning gain for $\Phi = 0.3$, $\Psi = 1.0$. Each data point is a local average of data from $N = 1000$ episodes 10

9 Selected action (left) and orientation θ (right) for the smart and naive (orange) gyrotactic particles swimming for an episode post-training at $\Psi = 0.3$, $\Phi = 3.0$ 10

10 Normalized (left) and unnormalized (right) Q matrix for $\Phi = 0.3, \Psi = 3.0$. The normalized matrix is obtained by scaling each row of Q independently such that it ranges from 0 to 1. . . . 11

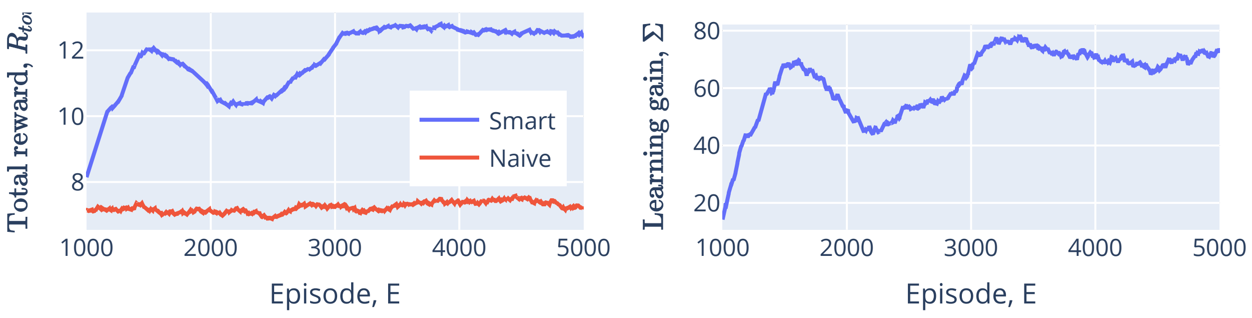

Figure 1: Total episodic reward and percent learning gain for $\Phi = 0.3$, $\Psi = 0.3$.

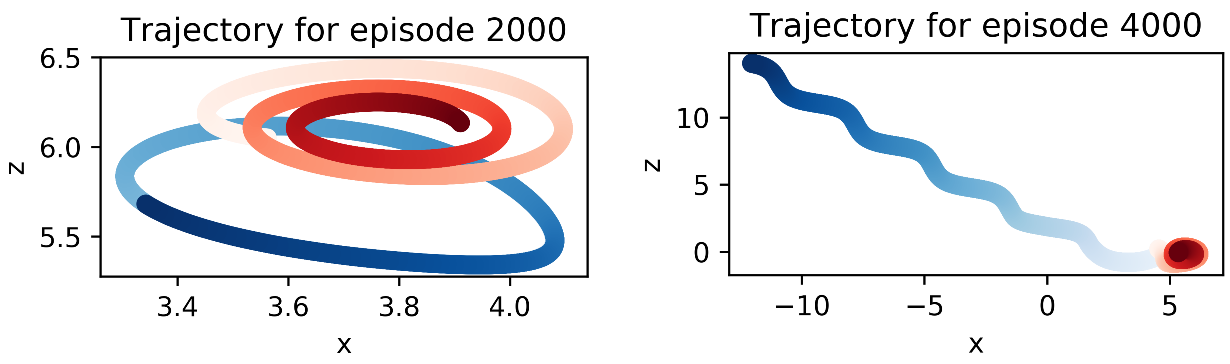

Figure 2: Selected trajectories for the smart (blue) and naive (red) gyrotactic particles for $\Phi = 0.3$, $\Psi = 0.3$. Darker colors denote later points in time.

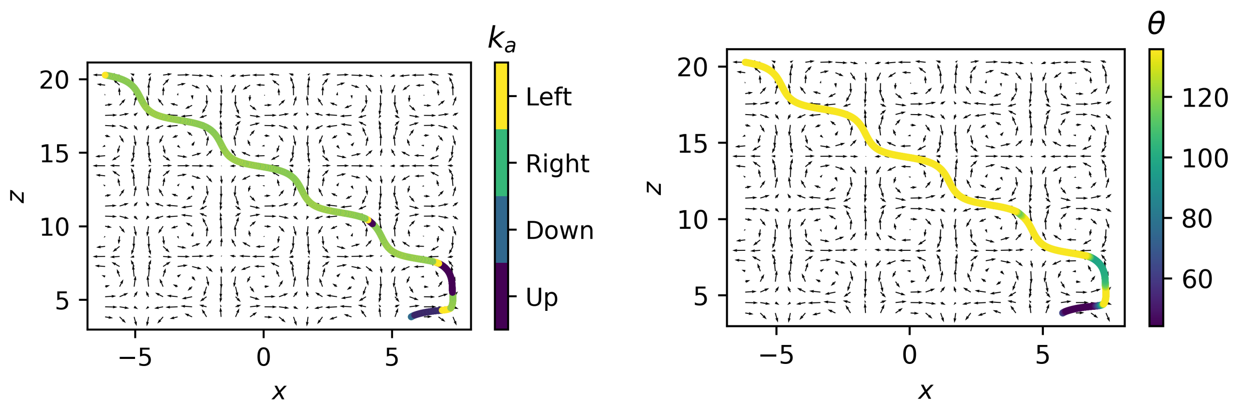

Figure 3: Selected action (left) and orientation θ (right) for the smart gyrotactic particle swimming during the final training episode at $\Psi = 0.3$, $\Phi = 0.3$

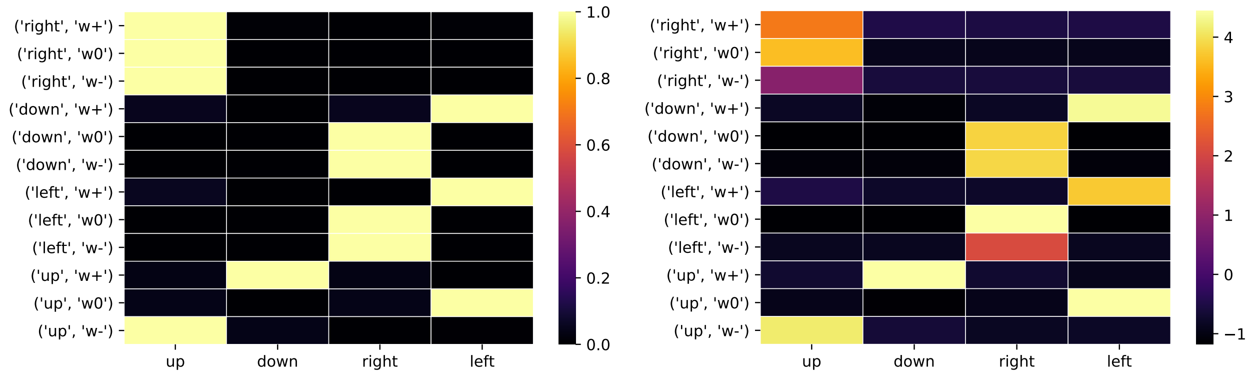

Figure 4: Normalized (left) and unnormalized (right) Q matrix for $\Phi = 0.3, \Psi = 0.3$. The normalized matrix is obtained by scaling each row of Q independently such that it ranges from 0 to 1.

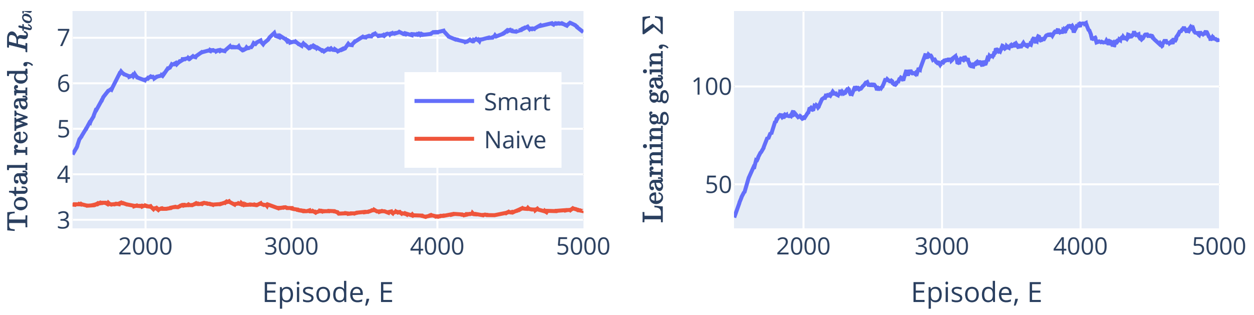

Figure 5: Total episodic reward and percent learning gain for $\Phi = 0.3$, $\Psi = 1.0$. Each data point is a local average of data from $N = 1500$ episodes

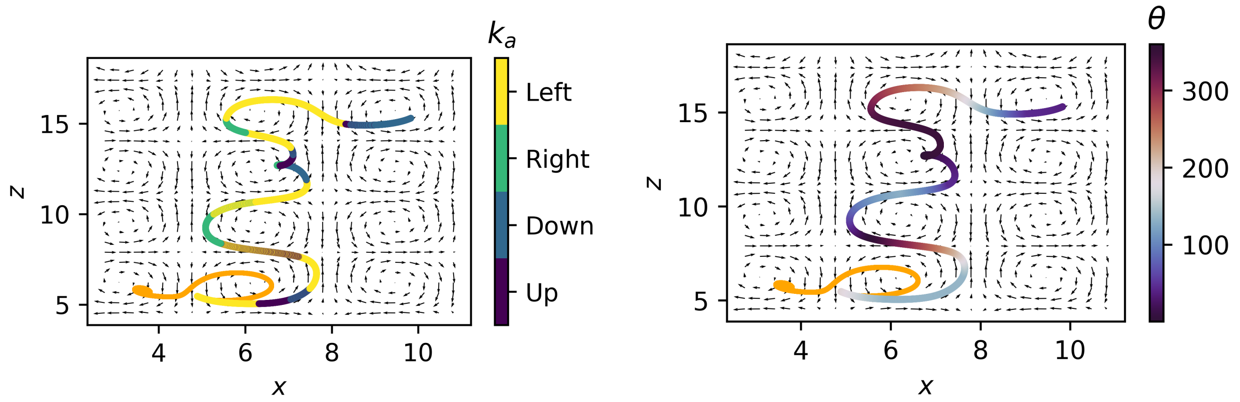

Figure 6: Selected action (left) and orientation θ (right) for the smart and naive (orange) gyrotactic particles swimming for an episode post-training at $\Psi = 0.3$, $\Phi = 1.0$

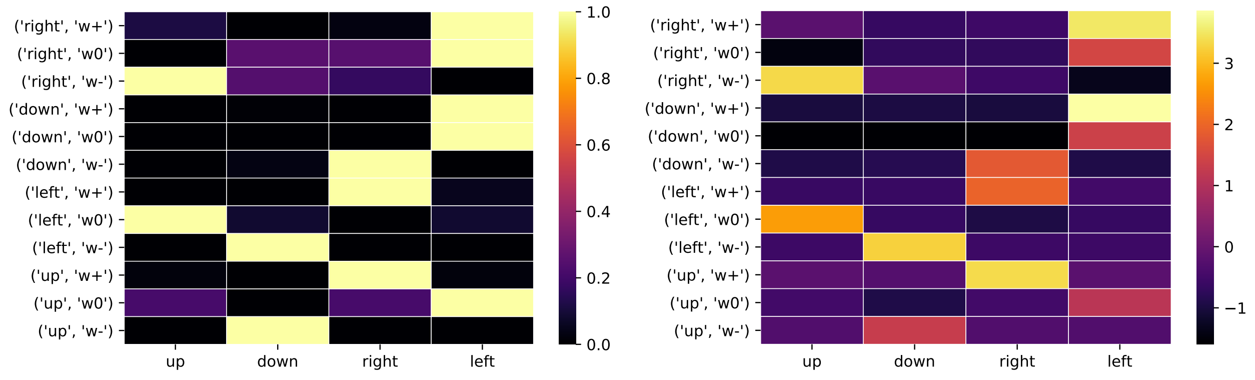

Figure 7: Normalized (left) and unnormalized (right) Q matrix for $\Phi = 0.3, \Psi = 1.0$. The normalized matrix is obtained by scaling each row of Q independently such that it ranges from 0 to 1.

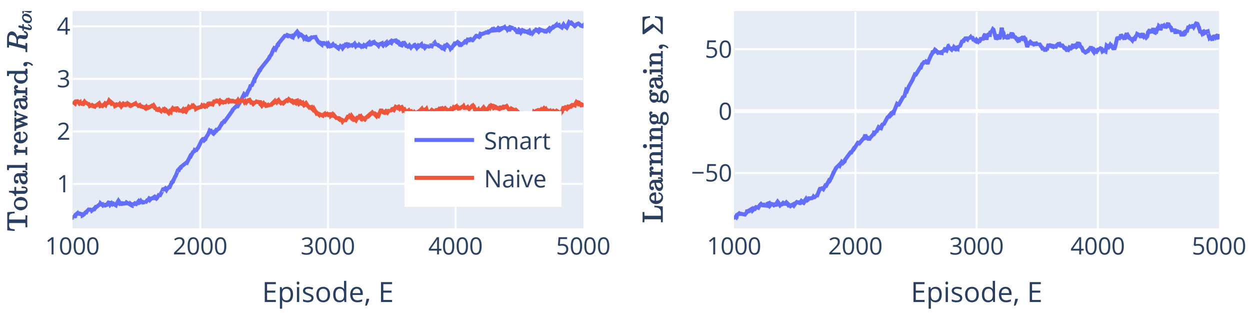

Figure 8: Total episodic reward and percent learning gain for $\Phi = 0.3$, $\Psi = 1.0$. Each data point is a local average of data from $N = 1000$ episodes

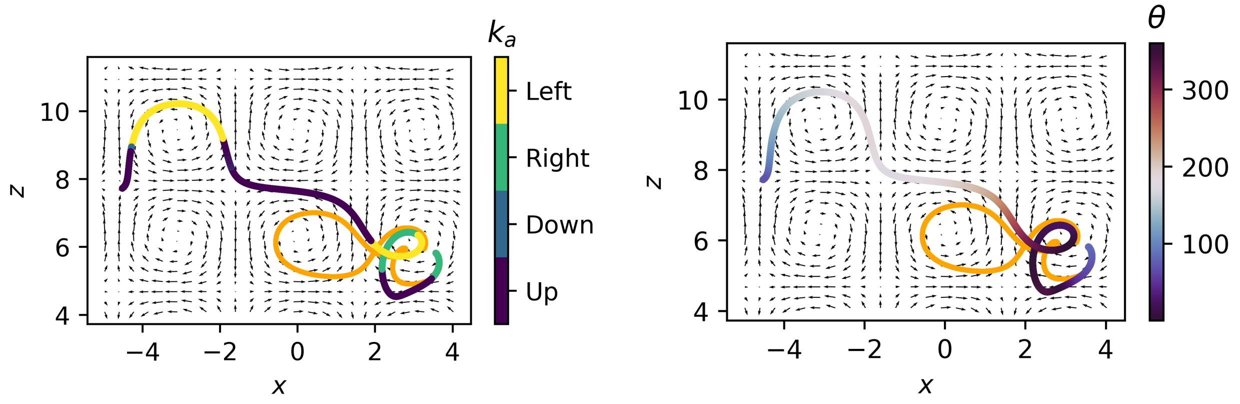

Figure 9: Selected action (left) and orientation θ (right) for the smart and naive (orange) gyrotactic particles swimming for an episode post-training at $\Psi = 0.3$, $\Phi = 3.0$

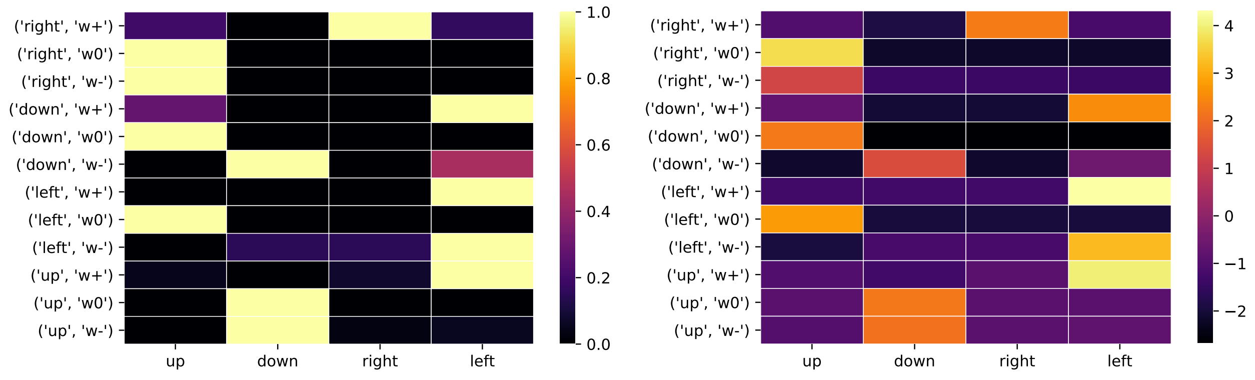

Figure 10: Normalized (left) and unnormalized (right) Q matrix for $\Phi = 0.3, \Psi = 3.0$. The normalized matrix is obtained by scaling each row of Q independently such that it ranges from 0 to 1.