Developing escape strategies for copepods using Q-learning

Master's thesis in Complex Adaptive Systems

ERIK HOLMBERG

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Developing escape strategies for copepods using Q-learning

ERIK HOLMBERG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2019

Developing escape strategies for copepods using Q-learning
ERIK HOLMBERG

Supervisor: Kristian Gustavsson, Department of Physics
Examiner: Bernhard Mehlig, Department of Physics

Master's Thesis 2019
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Calanus helgolandicus, a type of copepod. Credit: Plymouth Marine Laboratory

Typeset in LaTeX
Gothenburg, Sweden 2019

Developing escape strategies for copepods using Q-learning
ERIK HOLMBERG
Department of Physics
Chalmers University of Technology

Abstract

Copepods are a type of small crustaceans with the ability to move using powerful jumps. This project seeks to develop a model for the dynamics of these copepods in the turbulent flows of the oceans and then apply Q-learning to find good navigation strategies. Contained in this project is also the development of a statistical model of a turbulent flow.
It is found that using Q-learning it is possible for the simulated copepods to learn strategies that allow them to avoid dangerous areas almost equally as well as more simple strategies that require more information.

Keywords: Machine learning, Fluid mechanics, Simulation, Copepods, Q-learning.

Contents

1

Introduction

1.1 Background

Copepods, a group of small crustaceans, have generated a lot of interest in the scientific community during the past few years. One aspect that makes them interesting is their ability to move around using powerful jumps when they detect large shear changes. This ability allows them to navigate small-scale flows of the ocean in order to escape predators [1][2]. Understanding the consequences of this behaviour has been approached from many different directions, such as exploring the effect movement by jumping has on population dynamics [3]. The strategies employed by the copepods in determining when and where to jump have also been researched, both experimentally and theoretically [4].

Ardeshiri et al. [3] develops a Lagrangian model for copepod dynamics. In this model the copepods can always measure the strain magnitude S, and will jump when this is above a certain threshold. The strain magnitude is an important property of the flow and is used as a signal for when to jump since this behaviour is observed in experiments. There are numerous reasons that the copepods could want to avoid areas of high strain magnitude as it could both indicate and mask the presence of a large predator as well as make catching prey harder. The model developed in this project is in large part based upon the one devéloped by Ardeshiri.

Another field that has seen a lot of interest recently is machine learning. The temptation to have a computer learn to solve a difficult problem has proven strong and there are few areas of research where implementing machine learning in some capacity has not been attempted [5][6]. One of the main advantages of machine learning is that it excels at analyzing huge amounts of data and learning from patterns in a way that is infeasible for a human.

Machine learning is, however, just an umbrella term for a wide array of algorithms. The specific algorithm used in this project is called Q-learning [7], which in itself is part of a branch of machine learning called reinforcement learning [8]. Reinforcement learning relies on having an agent trying to perform a task repeatedly, each time giving the agent a reward depending on its performance. The agent registers the actions taken and the reward given during each attempt and eventually learns a strategy through trial and error. Q-learning is simply a specific way to structure this learning, detailed further in chapter 3. Q-learning has been successful in constructing efficient navigation strategies in similar situations and is therefore the algorithm of choice for this project [9].

1.2 Aim

The aim of the project is to program a two-dimensional statistical model of a turbulent flow as well as a model for copepod dynamics, then use Q-learning to find strategies for the copepods to avoid regions of high strain magnitude. The project seeks to answer whether a similar or better performing strategy than the one found by Ardeshiri et al. can be found despite not giving the copepod agent the ability to directly measure the strain magnitude.

2
Model

This chapter will walk you through the process of constructing the model of the flow and the copepod agents.

2.1 Constructing a flow

The type of flow used in the model is a two-dimensional incompressible velocity field consisting of a deterministic mean part $\boldsymbol{u}_D(\boldsymbol{r})$ and an additional stochastic part $\boldsymbol{u}_S(\boldsymbol{r}, t)$.

$$\boldsymbol{u}(\boldsymbol{r}, t) = \boldsymbol{u}_D(\boldsymbol{r}) + \boldsymbol{u}_S(\boldsymbol{r}, t) \tag{2.1}$$

That the flow is incompressible means that the divergence of the velocity field is zero at all points. This is important because the aim is to´model regular Navier Stokes flows which all have this property.

2.1.1 Deterministic mean flow

For the deterministic mean, two commonly used model flows were used. The first is a Kolmogorov flow [10], defined as

$$\boldsymbol{u}_D(\boldsymbol{r}) = (u_{D0} \cos \frac{y}{L}, 0) \tag{2.2}$$

where u_{D0} is a scale factor determining the magnitude of the velocities in the flow and L is a spatial scale factor. The velocity of this flow is always parallell to the x-axis and depends only on the y coordinate, this can be seen in the left panel of figure 2.1.
The other type of deterministic flow used is a TGV flow [11], and is defined as

$$\boldsymbol{u}_D(\boldsymbol{r}) = u_{D0}(-\cos \frac{x}{L} \sin \frac{y}{L}, \cos \frac{y}{L} \sin \frac{x}{L}) \tag{2.3}$$

which manifests as a periodic pattern of spirals in interchanging directions, seen in the left panel of figure 2.2. Both of these flows are incompressible by design.

2.1.2 Stochastic flow

The stochastic flow is modelled in a way that has previously successfully explained complex dynamics observed in small inertial particles [12][13][14] . When construct-

ing the stochastic part of the field we need to make sure that it is also incompressible, this will be satisfied if it is of the form

$$\boldsymbol{u}_S(\boldsymbol{r},t) = (\partial_y\phi(\boldsymbol{r},t), -\partial_x\phi(\boldsymbol{r},t)) \tag{2.4}$$

since the requirement is $\partial_x u_{Sy} + \partial_y u_{Sx} = 0$. The following function ϕ is in our case calculated by using a Fourier decomposition with random coefficients

$$\phi(\boldsymbol{r},t) = \sqrt{2\pi}\frac{\eta^2 u_{S0}}{\sqrt{2L_xL_y}}\sum_{\boldsymbol{k}} a_{\boldsymbol{k}}(t)e^{i\boldsymbol{k}\cdot\boldsymbol{r}-k^2\eta^2/4}. \tag{2.5}$$

Here, u_{S0} is a magnitude scale factor much like in the deterministic flow. L_x and L_y are the system sizes in the x-direction and y-direction. η is the correlation length of the flow and $\boldsymbol{k}$ is a wave vector that can take values on the form $\boldsymbol{k} = 2\pi(n_x/L_x, n_y/L_y)$ where $n_x, n_y \in \mathbb{Z}$. For each of these possible $\boldsymbol{k}$ and each time step t, there is a complex Fourier coefficient $a_{\boldsymbol{k}}(t)$ randomly drawn from a Gaussian distribution with zero mean and covariance $\langle a_{\boldsymbol{k}} a^*_{\boldsymbol{k}'}\rangle = \delta_{\boldsymbol{k}\boldsymbol{k}'}$. An additional constraint on these values is $a^*_{\boldsymbol{k}} = a_{-\boldsymbol{k}}$ in order for the sum in (2.5) to be real.
In each time step these coefficients are updated according to

$$a_{\boldsymbol{k}}(t+\delta t) = (1-\delta t/\tau)a_{\boldsymbol{k}}(t) + b_{\boldsymbol{k}}(t), \tag{2.6}$$

where δt is the time step used in the simulation, τ is the correlation time of the flow and $b_{\boldsymbol{k}}(t)$ are independent complex Gaussian random numbers with zero mean and covariance $\langle b_{\boldsymbol{k}} b^*_{\boldsymbol{k}'}\rangle = 2\delta t/\tau\delta_{\boldsymbol{k}\boldsymbol{k}'}$. This choice of $b_{\boldsymbol{k}}(t)$ yields an exponential time correlation and keep the conditions of $a_{\boldsymbol{k}}$ fulfilled, as long as $b^*_{\boldsymbol{k}} = b_{-\boldsymbol{k}}$.
Generating ϕ in this way yields a function with Gaussian space correlation and exponential time correlation:

$$\langle\phi(\boldsymbol{r},t)\phi(\boldsymbol{r}',t')\rangle = \frac{\eta^2 u_{S0}^2}{2}e^{-(\boldsymbol{r}-\boldsymbol{r}')^2/(2\eta^2)-|t-t'|/\tau}. \tag{2.7}$$

When realizing this calculation however, one has to choose how many $\boldsymbol{k}$'s to consider, since we cannot actually compute an infinite number of Fourier modes. For the tests in this project, this limit has been set at $\max n = 5, \min n = -5$ för both n_x and n_y. The left panels of figures 2.1 and 2.2 show what the velocity fields look like. In these realizations $u_{S0} = 0.025u_{D0}$, which means that the deterministic part of the velocity is significantly larger than the stochastic part at typical locations. This can be seen in the velocity maps of the flows which look very organized. For our purposes, one of the most important properties of the flow is the strain magnitude, and the strain magnitude maps, shown in the right panels of figures 2.1 and 2.2 look quite randomized in comparison to the velocity field. Here we can see several low and high strain magnitude "pockets" caused by the stochastic flow even though the overall structure of the deterministic flow is still clearly visible. From figure 2.3 we see that the mean of the strain magnitude of the stochastic flow is approximately 13 times the value of u_{S0}. Compare this to the deterministic flow where the mean of the strain magnitude is equal to u_{D0}. This relationship between u_{S0} and u_{D0} depends on a number of factors notably including the correlation length η. A larger

η would lead to lower strain magnitudes for the same values of u_{S0} as the flow would be more "smooth".
It is worth noting that the coefficient 13 in the relationship between u_{S0} and the mean strain magnitude is not the same as the one expected by analytical calculations. According to the analytical calculations the coefficient should have been $\sqrt{8}/\eta \approx 28.3$, it is unclear why this is the case but it is likely due to some error in the code.

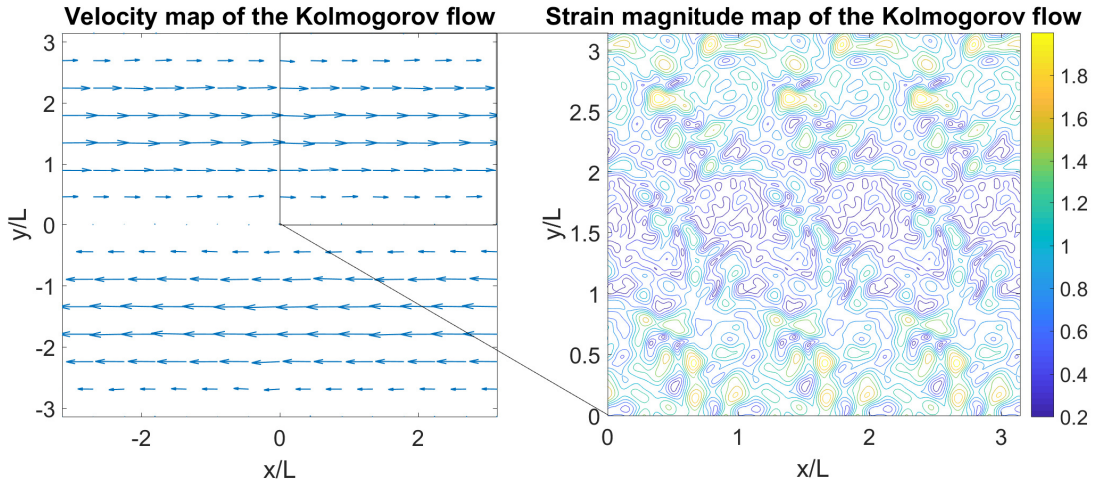

Figure 2.1: To the left is the velocity field of a typical Kolmogorov flow generated with the method described in this section, to the right is a contour map of the strain magnitude of the same flow. The top and bottom parts of the map are high strain magnitude regions due to the deterministic part of the flow but they also have low strain magnitude pockets due to the stochastic part.

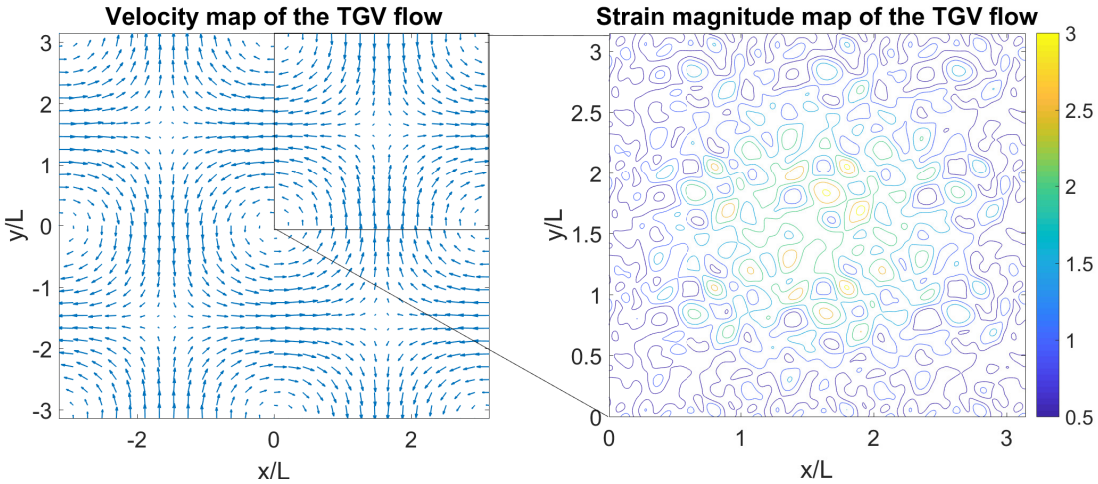

Figure 2.2: To the left is the velocity field of a typical TGV flow generated with the method described in this section, to the right is a contour map of the strain magnitude of the same flow. The center of the map is a high strain magnitude region due to the deterministic part of the flow but it also has low strain magnitude pockets due to the stochastic part.

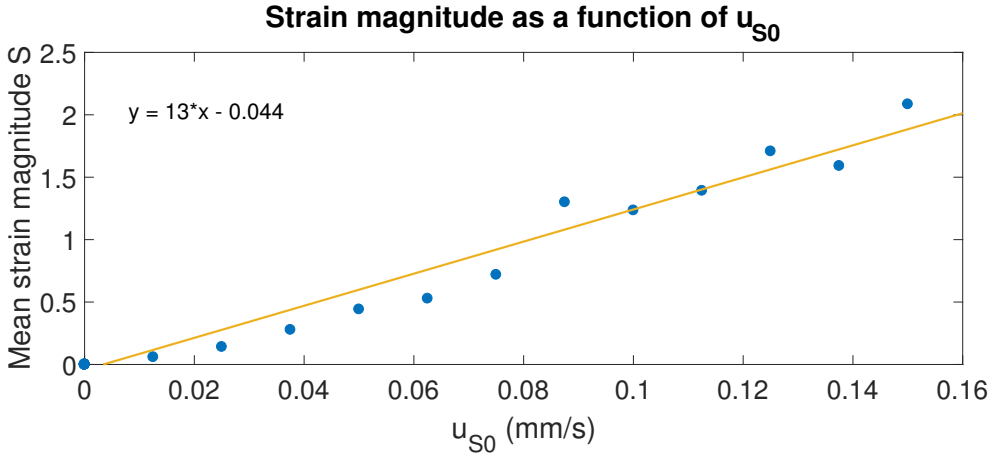

Figure 2.3: This graph shows the mean strain magnitude as a function of the scale of the velocity flow. This only takes into account the stochastic part of the flow and shows that even a small velocity perturbation will lead to a large strain perturbation.

2.2 Properties of the flow

This section will explain some concepts of fluid dynamics required to understand what it is that the copepods are trying to avoid.
Since the copepods are moving in a fluid, their movement is governed by flow of the fluid. To describe the effects of the flow on a particle, it is useful to define the tensor $\mathbb{A} = \nabla \boldsymbol{u}^{\mathrm{T}}$ which consists of the spatial derivatives of the flow $\boldsymbol{u}$. The elements of $\mathbb{A}$ are

$$A_{ij} = \frac{\partial u_i}{\partial x_j}. \tag{2.8}$$

$\mathbb{A}$ can be decomposed into a symmetric tensor $\mathbb{S}$ and an asymmetric tensor $\mathbb{O}$ with the definitions

$$\mathbb{S} = \frac{1}{2}(\mathbb{A} + \mathbb{A}^{\mathrm{T}}), \quad \mathbb{O} = \frac{1}{2}(\mathbb{A} - \mathbb{A}^{\mathrm{T}}). \tag{2.9}$$

The symmetric tensor $\mathbb{S}$ describes the strain of the flow, or the local rate of deformation. This is a very important tensor as the magnitude of the strain, defined as $S = \sqrt{2\mathbb{S} : \mathbb{S}}$, determines whether a part of the flow is desirable for the copepods or not. The anti-symmetric tensor $\mathbb{O}$ describes the rotation of the flow and has a relation to the vorticity vector of the flow $\boldsymbol{w}_f$. More specifically, for any vector $\boldsymbol{x}$

$$\mathbb{O}\boldsymbol{x} = \frac{1}{2}\boldsymbol{w}_f \times \boldsymbol{x}. \tag{2.10}$$

In terms of these tensors, the effect of a flow on the trajectory of an elliptic particle in two dimensions, which is how the model approximates the copepods, can be expressed as

$$\dot{\boldsymbol{n}} = \mathbb{O}\boldsymbol{n} + \frac{\lambda^2 - 1}{\lambda^2 + 1}(\mathbb{S}\boldsymbol{n} - (\boldsymbol{n}^{\mathrm{T}}\mathbb{S}\boldsymbol{n})\boldsymbol{n}). \tag{2.11}$$

as first derived by Jeffery [15]. Here $\boldsymbol{n}$ is the orientation vector of the particle, which in our case can be viewed as the direction the copepod is facing. λ is the shape parameter of the elliptic particle. More specifically it is the aspect ratio between the diameter parallell to the orientation vector and the diameter orthogonal to it. In this project λ is chosen to be 3.

2.3 Copepod jumps

Experimental data suggests that copepods mainly move by performing powerful "jumps" [2]. These jumps are characterised by a quick acceleration to a speed significantly greater than that of the flow, followed by a slower decrease in speed untill they are once again "at rest" and ready for another jump. In this thesis, these jumps are modeled in the way introduced by Ardeshiri et al. [3].
The copepods are assumed to be rigid, homogeneous, neutrally buoyant particles small enough to be considered perfect tracers in a flow, except during a jump. In Ardeshiris model, the movement of the copepods is described as

$$\dot{\boldsymbol{r}}(t) = \boldsymbol{u}(\boldsymbol{r}(t), t) + \boldsymbol{J}(t, t_i, t_e, S, \boldsymbol{n}) \tag{2.12}$$

where $\boldsymbol{u}(\boldsymbol{r}(t), t)$ is the flow and $\boldsymbol{J}(t, t_i, t_e, S, \boldsymbol{n})$ is an added velocity term describing the jumps that in this model is the copepods only way of moving on their own. In addition to being a function of t, it also depends on the initial and final times t_i and t_e as well as the magnitude of the strain S. The jumps are modelled as

$$\boldsymbol{J}(t, t_i, t_e, S, \boldsymbol{n}) = H[S(t_i) - S_T]H[t_e - t]u_J e^{\frac{t_i - t}{\tau_J}}\boldsymbol{n}(t_i) \tag{2.13}$$

where $H[x]$ is the Heaviside step function and S_T is the threshold of the strain magnitude where the copepods will start jumping. u_J is a parameter that determines the copepods speed at the beginning of the jump, similarily τ_J determines the duration of the jump. t_i and t_e serve as time markers keeping track of when the jump started and when it's going to end. t_i is assigned the value of t when two criteria are fulfilled, $S(t) > S_T$ and $t > t_e$. This ensures that a jump only starts when the previous jump is over and the strain magnitude is sufficiently high. At the same time, t_e is set to $t_i + c\tau_J$ where c is chosen to make sure that the amplitude of the jump has reached a negligable level before it is terminated.
In the simulations, the positions and orentations of the copepods are updated using the Runge-Kutta method with a timestep of δt.

2.3.1 Changes to the existing model

The main addition to the model made in this project is in the way the copepods decide whether to jump or not. In the previous model, the copepods could measure the magnitude of the strain directly. If this measured strain was above a fixed threshold the copepod would always jump, otherwise it would never jump. In this

project, this is replaced by a strategy developed through Q-learning that also gives a more realistic amount of information to the copepods. This will be explained in the next chapter.
Another change to the original model is the possibility of sideways jumps in addition to the existing forward jumps. These are meant to represent the copepods ability to quickly rotate before jumping[16], thus giving the simulated copepod some amount of control over the direction of the jump.

3
Q-learning

This chapter will explain the core concepts of Q-learning and show how it is implemented in the copepod model.

3.1 Reinforcement learning

Reinforcement learning is a form of machine learning, meaning that it is a type of algorithm that teaches a computer to solve a problem without explicitly telling the computer what to do. Reinforcement learning relies on the principle of trial and error to find an optimal strategy. The simulated agent will try to perform an action and will be given a reward depending on how successful it is. This reward will then impact how likely the agent is to attempt the same action again.

3.1.1 Q-learning

The basis for any reinforcement learning is a set of states, each with a set of possible actions. Performing an action in a certain state nets the agent a reward depending on the success of that action. The exact way that this happens depends on the specific algorithm used.
The algorithm used in this project is called Q-learning[7]. The premise of Q-learning is that every combination of state S and action A has a Q-value $Q(S_t, A_t)$ at a given time step t. If the simulated agent is in state S_t and performs action A_t at time t, this value will be updated according to

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]. \tag{3.1}$$

In this expression, R_{t+1} is the reward given to the agent at time $t+1$. This reward should be based on what the agent is trying to learn. If, for example, the agent is learning to navigate northward on a map, a suitable reward would be the distance north traveled since time t. The term $\max_a Q(S_{t+1}, a)$ can be thought of as the expected future value of ending up in state S_{t+1} as it will take the Q-value of the best action available from that state. The discount factor γ determines how much the agent values future rewards, or how long-sighted it is. α is the learning rate and determines how much the agent will let each update influence the learned value. A large α will quicken the learning process but might also make it harder for the algorithm to converge as values will fluctuate more. Both α and γ should be in the range of 0 to 1.

To explain this more conceptually: The agent, being in a certain state, performs a certain action and then in some way measures how successful this action was (the reward R_{t+1} and the value of the resulting state $\gamma \max_a Q(S_{t+1}, a)$). This is then compared to how successfull this action was previously excpected to be (the old Q-value $Q(S_t, A_t)$) and the difference multiplied by the learning rate α is added to the old Q-value.

3.1.2 Exploration and exploitation

When implementing a reinforcement learning algorithm it is important to consider the balance of exploration and exploitation, this is represented in the algorithm by the parameter ϵ. Every time the agent chooses an action, there is a probability ϵ that it will choose completely at random. When not choosing at random, it chooses the action $a : \max_a Q(S, a)$ where S is the current state of the agent. This is the action that is believed by the agent to give the best future reward.
If $\epsilon = 0$ the agent will always choose the "greedy" action that is the current estimated best option. The risk of this is that an action that is actually optimal might be discarded due to an unfortunate poor reward in an early trial or similar. A nonzero ϵ will make sure that the entire action-state-space of the Q-matrix will eventually be visited many times, which is necessary for the algorithm to converge to the optimal solution. In this project ϵ is chosen to be 0.01.

3.2 Setting up the Q-learning

In order to use the Q-learning algorithm described in section 3.1.1 in our specific model, choices need to be made regarding the state space, the action space and the learning parameters.

3.2.1 State and action space

In an attempt to model the sensory input of the copepods in a less demanding way than in Ardeshiri et al. the state space is based on two values that could be known to the copepod by measuring the velocity of the flow at four points on its body. The two values are projections of the strain $\boldsymbol{n}^T \mathbb{S} \boldsymbol{n}$ and $\boldsymbol{n}^T \mathbb{S} \boldsymbol{p}$ where $\boldsymbol{p}$ is the orientation vector orthogonal to $\boldsymbol{n}$. Each of these values will fall into one of three states: positive, negative or close to zero. This results in a total of 9 states. In comparison, Ardeshiris model gives the agents direct access to the strain magnitude S which, although it can be calculated from the projections used in our model, requires more sophisticated calculations which it is unclear whether the copepods can perform. Knowing the two projections is likely to be less useful than knowing the strain magnitude S as the 9 states in our model will "overlap", meaning that the same value of S will appear in several different states. This will make it harder for the agent to know when it is in an area of high strain magnitude. As for the action space the model has 4 available actions in each state: jump in the direction of the instantaneous orientation $\boldsymbol{n}$, rotate left and then jump, rotate right and then jump and not jumping. This means that the Q-matrix will be a 9×4 matrix.

3.2.2 Reward

The choice of reward is extremely important as it decides what the agent will try to learn. The goal of the chosen reward in this project is to emulate the desire of the copepods to avoid regions of high strain magnitude S. The chosen reward is

$$R_t = -S(t)^2. \tag{3.2}$$

This reward has two desired properties. It decreases as the strain increases, and it does so in way that disproportionately punishes large strains. The reason why this is desirable is because it is more important for the copepods to avoid the highest strain regions than find the lowest strain regions.
Since the reward will always be negative, all Q-values will be negative once the Q-learning has converged. Thus setting all initial Q-values to zero will ensure some exploration of the entire Q-matrix. This strategy of setting optimistic inital Q-values works because the algorithm will almost always pick the action with the highest Q-value which means that all state-action combinations are likely to be tried a number of times before the best one is found.

3.2.3 Training

The final steps to set up the Q-learning is to decide how to structure the updating of the Q-matrix. To simplify the model slightly, the decision was made to only update the Q-matrix at set time intervals. These intervals were set to be equal to the relaxation time of the copepods, that is the time from the start of a jump until the copepod is ready to jump again. This means that every time the Q-matrix is updated the copepod will be capable of jumping. This allows the algorithm to use these certain time steps to do a number of things at the same time:

- Determine which state the agent is currently in.
- Calculate the reward $R_t = -S(t)^2$ where $S(t)$ is the instantaneous strain magnitude.
- Update the Q-matrix.
- Choose and execute the next action.

All of these things happen at these certain time steps and never at other time steps. This is slightly unrealistic as the copepods would normally be free to jump at any time but it makes the model a lot simpler.
It is of course also important to decide the duration of the training session. The terminology used in this project has been that a training session consists of a number of episodes. Each episode starts with a copepod placed randomly in the flow and continues for a set number of timesteps. The only information that is carried over between the episodes is the Q-matrix.

3.2.4 Parameters

We recall that the updating formula for the Q-learning is

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)].$$

For all results presented in this thesis, the parameters in this formula has been set as follows: $\alpha = 0.1$, $\gamma = 0.99$, $\epsilon = 0.01$. Remember that ϵ is the probability to perform a random action rather than the most profitable action in a given timestep. The reason that a rather small ϵ is chosen is that even though we are only interested in finding the best strategy, having the agent mostly follow the optimal path will allow for training in the scenarios that it is more likely to encounter when following the finished strategy.
To summarize, the parameters used during the simulations (including the ones brought up in the previous chapter) are found in table 3.1.

Table 3.1: List of all fixed parameters used during the simulations.

Learning rate α	0.1
Discount factor γ	0.99
Exploration rate ϵ	0.01
Base mean flow velocity u_{D0} (mm/s)	1
Base stochastic flow velocity u_{S0} (mm/s)	0.025
Stochastic flow correlation length η (mm)	0.1
Stochastic flow correlation time τ (s)	0.1
Stochastic flow system size L_x, L_y (mm)	1
Jump speed u_J (mm/s)	50
Jump duration τ_J (s)	0.01
Copepod aspect ratio λ	3
Simulation timestep δt (s)	0.001
Runge-Kutta stepsize h (mm)	0.001
Timesteps per episode	10000
Episodes	500

4

Results and discussion

This section will present the results obtained during the project.

4.1 Tuning the learning

As mentioned in chapter 3, the state space of the Q-matrix is made up of three different levels of $S_{\parallel} = n^T \mathbb{S} n$ and three different levels of $S_{\perp} = n^T \mathbb{S} p$. The decision remaining to be made is where the thresholds between these levels are. Assuming that the levels are distributed as shown in table 4.1, a sweep of possible values for the threshold T was performed, the results of which is shown in figure 4.1. From this sweep we can conclude that the optimal threshold is probably around 0.37 and this is the value used for T in all upcoming results. This sweep was done in the Kolmogorov flow but a very similar result was obtained in the TGV flow. Figure 4.2 shows how the reward progresses through a training session.

Table 4.1: Table illustrating how the state space is structured with regards to the threshold parameter T.

	$S_{\perp} < -T$	$\lvert S_{\perp}\rvert < T$	$S_{\perp} > T$
$S_{\parallel} < -T$	s_{--}	s_{-0}	s_{-+}
$\lvert S_{\parallel}\rvert < T$	s_{0-}	s_{00}	s_{0+}
$S_{\parallel} > T$	s_{+-}	s_{+0}	s_{++}

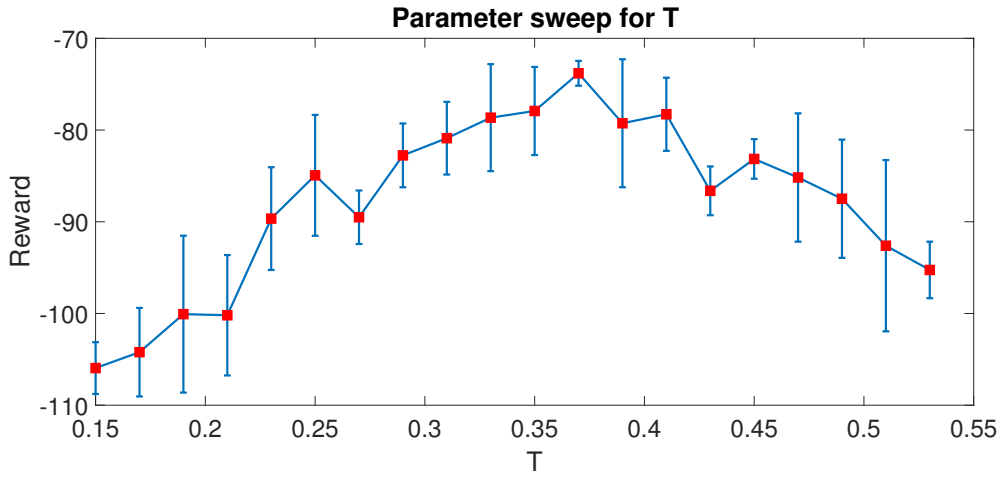

Figure 4.1: Results of a parameter sweep over the strain threshold T. For each value of T, 4 training sessions were run. The mean total reward for the last part of these trainings are plotted against T with the standard deviations shown as error bars. The most efficient T is likely around 0.37.

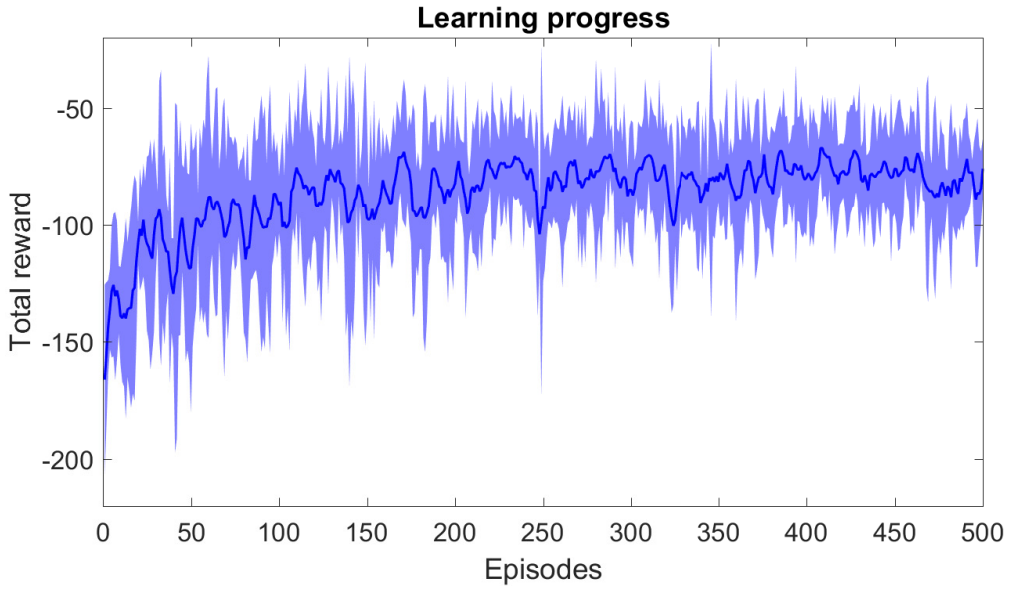

Figure 4.2: The total reward earned by an agent in each episode of a training session, averaged over five training sessions. The opaque line shows the mean reward and the shaded area shows the standard deviation. As the training proceeds the strategy improves from an average reward around -150 to an average reward around -80. Learning parameters are $\alpha = 0.1$,$\epsilon = 0.01$,$\gamma = 0.99$.

4.2 Learned strategies

After a training session the Q-matrix will in most cases have stabilized. The strategy that has then been learned is defined by the actions that have the highest Q-value

for each state. To answer the question of whether the same strategy is learned every time, the results of many training sessions are aggregated into tables 4.2 and 4.3 for the two flows. The tables also summarize the strategies generally learned during the training and we can observe that in each case only three show any signinficant correlation between runs. An observation can be made that in the Kolmogorov flow, the three conclusive states all share the same perpendicular component and only differ in the parallell component. In the TGV flow, they all instead share the same parallell component and differ in the perpendicular one.

Table 4.2: The left table shows the sum of all learned strategies over 40 training sessions in the Kolmogorov flow. For each state and action, the number in the corresponding box is the number of training sessions that ended with a strategy performing that action in that state. The right table shows a simplified summary of the conclusions one can draw from this result. States where results vary wildly are marked with ? while states where the same action prevails often are marked with that action.

	$\cdot$	$\uparrow$	$\leftarrow$	$\rightarrow$
s_{--}	7	9	15	9
s_{0-}	3	11	9	17
s_{+-}	7	11	13	9
s_{-0}	1	30	7	2
s_{00}	38	0	1	1
s_{+0}	0	34	1	5
s_{-+}	8	13	8	11
s_{0+}	3	13	19	5
s_{++}	10	9	10	11

State	Action
s_{--}	?
s_{0-}	?
s_{+-}	?
s_{-0}	$\uparrow$
s_{00}	$\cdot$
s_{+0}	$\uparrow$
s_{-+}	?
s_{0+}	?
s_{++}	?

Table 4.3: The left table shows the sum of all learned strategies over 40 training sessions in the TGV flow. For each state and action, the number in the corresponding box is the number of training sessions that ended with a strategy performing that action in that state. The right table shows a simplified summary of the conclusions one can draw from this result. States where results vary wildly are marked with ? while states where the same action prevails often are marked with that action.

	$\cdot$	$\uparrow$	$\leftarrow$	$\rightarrow$
s_{--}	5	7	12	16
s_{0-}	1	38	1	0
s_{+-}	2	21	6	11
s_{-0}	2	9	6	23
s_{00}	40	0	0	0
s_{+0}	3	18	13	6
s_{-+}	5	3	20	12
s_{0+}	1	35	3	1
s_{++}	2	13	17	8

State	Action
s_{--}	?
s_{0-}	$\uparrow$
s_{+-}	?
s_{-0}	?
s_{00}	$\cdot$
s_{+0}	?
s_{-+}	?
s_{0+}	$\uparrow$
s_{++}	?

Figures 4.3 and 4.4 shows what the execution of the learned strategies look like in practice. A core element of the strategy is of course to jump when the agent is in an area with a high strain magnitude and to stay still when in an area with a low strain magnitude, this can be seen in both examples. In the Kolmogorov example we can see that the agent sometimes jumps away from the optimal line even after it finds it. This could be due to finding high strain magnitude pockets caused by the stochastic field. It does however find its way back eventually. Figures 4.5 and 4.6 illustrate the results of the learned strategies in a slightly different way. These figures contain the distribution of copepod agents after they have followed the learned strategies for an episode and we can observe that the low strain regions are in all cases more densely populated than the high strain areas. In each of these figures, the right panel shows the results of following a strategy that is a "mirroring" of the strategy used in the left panel, meaning that all left turns are replaced by right turns and vice versa. Noting the qualitative similarity between the two results, it is likely that mirroring a strategy has no bearing on the effectiveness of the strategy. This symmetry should be taken into considration when evaluating the results in tables 4.2 and 4.3.

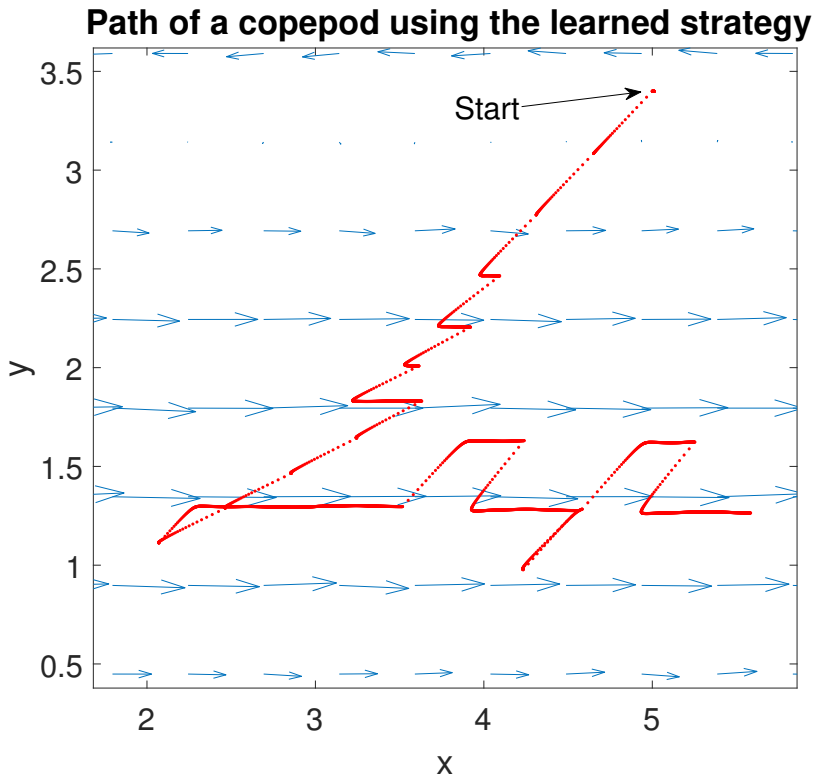

Figure 4.3: An example of the path an agent can take in the Kolmogorov flow after learning a strategy. Each dot represents the agents position at one time step, the places where the dots are further apart are places where the agent performed a jump. In this example, the agent starts out in an area with a high strain magnitude and moves to find an area with a low strain magnitude.

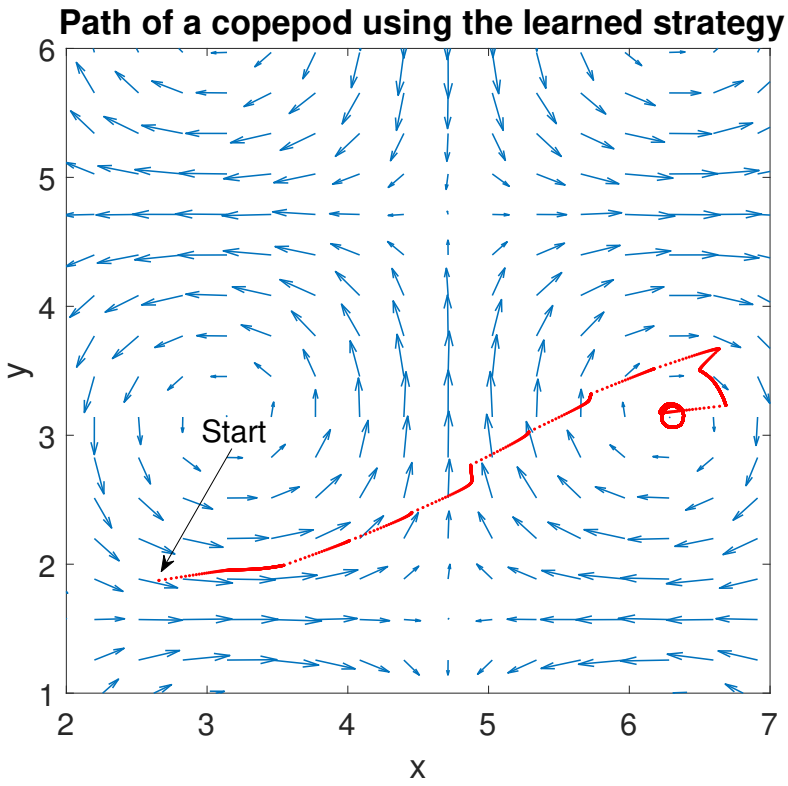

Figure 4.4: An example of the path an agent can take in the TGV flow after learning a strategy. Each dot represents the agents position at one time step, the places where the dots are further apart are places where the agent performed a jump. In this example, the agent manages to find its way to the center of a spiral where the strain magnitude is low.

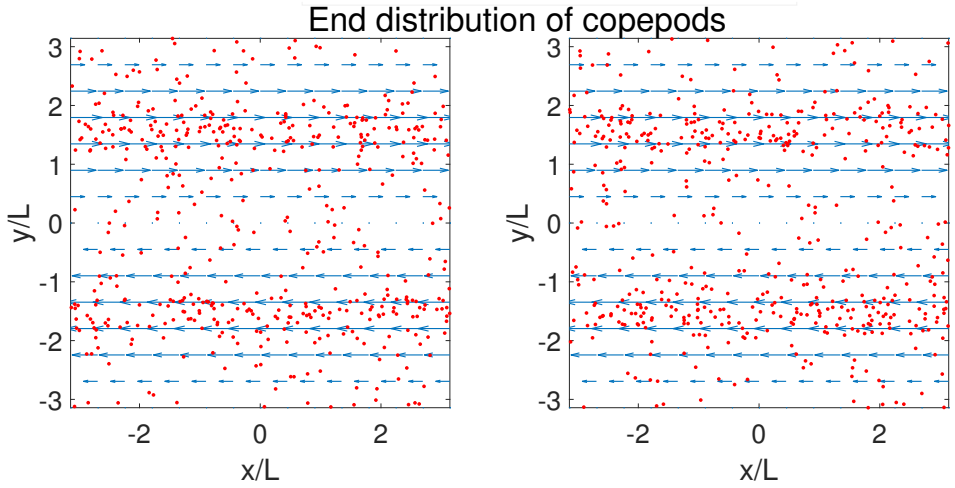

Figure 4.5: These images show the distribution of 500 copepod positions after each copepod has navigated for a complete episode using a Q-learning developed strategy in the Kolmogorov flow. The left panel shows the results of using the developed strategy while the right panel shows the results of using a "mirrored" strategy, meaning that all left turns are replaced with right turns and vice versa. The similarity between the two panels show that mirrored strategies perform at a similar level.

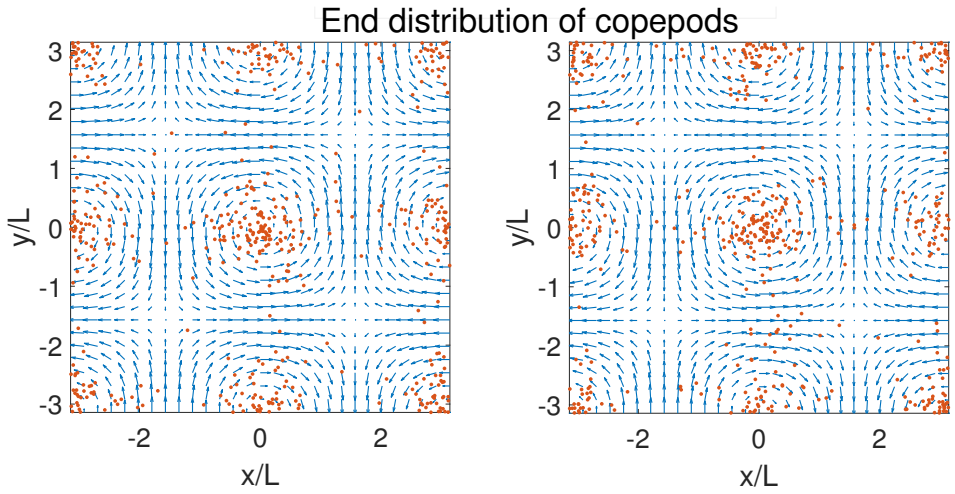

Figure 4.6: These images show the distribution of 500 copepod positions after each copepod has navigated for a complete episode using a Q-learning developed strategy in the TGV flow. The left panel shows the results of using the developed strategy while the right panel shows the results of using a "mirrored" strategy, meaning that all left turns are replaced with right turns and vice versa. The similarity between the two panels show that mirrored strategies perform at a similar level.

To shed some light on why these particular strategies emerge and why they differ from field to field, I took a closer look on how exactly the values $\boldsymbol{n}^T\mathbb{S}\boldsymbol{n}$ and $\boldsymbol{n}^T\mathbb{S}\boldsymbol{p}$ change with the orientation of the agent. Assuming $\boldsymbol{n} = (\cos\alpha, \sin\alpha)^T$ and $\boldsymbol{p} = (-\sin\alpha, \cos\alpha)^T$ the values for the Kolmogorov flow (only considering the mean part of the flow) will be:

$$\boldsymbol{n}^T \mathbb{S} \boldsymbol{n} = -\frac{u_{D0}}{L} \cos\frac{y}{L} \sin 2\alpha \tag{4.1}$$

$$\boldsymbol{n}^T \mathbb{S} \boldsymbol{p} = -\frac{u_{D0}}{L} \cos\frac{y}{L} \cos 2\alpha \tag{4.2}$$

While for the TGV flow it will be:

$$\boldsymbol{n}^T \mathbb{S} \boldsymbol{n} = \frac{u_{D0}}{L} \sin\frac{x}{L} \sin\frac{y}{L} \cos 2\alpha \tag{4.3}$$

$$\boldsymbol{n}^T \mathbb{S} \boldsymbol{p} = \frac{u_{D0}}{L} \sin\frac{x}{L} \sin\frac{y}{L} \sin 2\alpha \tag{4.4}$$

This doesn't immediately give an answer but it allows for some speculation. The terms can be thought of as consisting of two factors. One is the one that is the same for both $\boldsymbol{n}^T \mathbb{S} \boldsymbol{n}$ and $\boldsymbol{n}^T \mathbb{S} \boldsymbol{p}$, this one depends on the coordinates of the agents. The other one depends on the orientation angle α, and in both flows one of the values takes the form $\cos 2\alpha$ and the other is $\sin 2\alpha$. What these factors mean in practice is that the magnitude of the term in question will be larger when the orientation of the agent is horizontal or vertical ($\cos 2\alpha$) or diagonal ($\sin 2\alpha$). The interesting part is that the places are reversed between the flows.

Connecting this to observations made regarding the learned strategies we remember that the agent in the Kolmogorov flow learned strategies most consistently when $\boldsymbol{n}^T \mathbb{S} \boldsymbol{p}$ was close to zero. This is in that case the term with $\cos 2\alpha$. In the TGV flow the same was true for $\boldsymbol{n}^T \mathbb{S} \boldsymbol{n}$ wich in that case is also the term with $\cos 2\alpha$. It is tempting to find a connection here. $\cos 2\alpha$ being close to zero means that the agent is oriented diagonally, it could be that the agent spends most of its time oriented in such a way due to some property in the rotational dynamics of the flow (2.11). Numerical data from the simulations support this, as shown in figure 4.7 where the left panel correlates to the Kolmogorov flow and the right to the TGV flow. This tendency of the agent to be oriented diagonally offers an explanation as to why the states where $\cos 2\alpha$ is close to zero are seemingly more important parts of the strategy. The results from the TGV flow also shows that two of the peaks are significantly larger than the two others, the exact cause of this is unknown but is likely rooted in the stability of the rotational dynamics. It should be noted that the prevalence of diagonal orientations is a property of the flows and is not inherent in the copepod model.

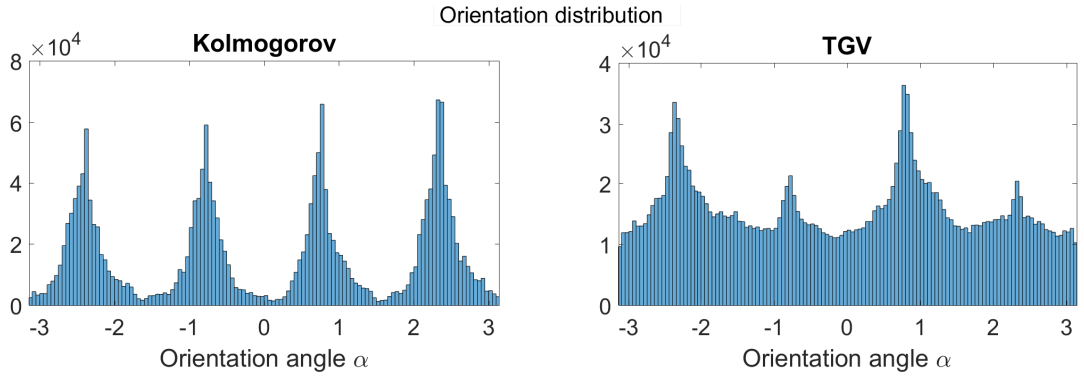

Figure 4.7: These figures show the distribution of the orientation angle α in a histogram measured during 200 episodes using the learned strategy. The left panel shows data from the Kolmogorov flow while the right is from the TGV flow. Both distributions show clear peaks at $-\frac{3\pi}{4}$, $-\frac{\pi}{4}$, $\frac{\pi}{4}$ and $\frac{3\pi}{4}$. The distribution from the TGV flow has a substantially higher frequency of all other angles, this is because the agents will rotate continuously once they find their way to a spiral.

4.3 Performance compared to previous results

It is not obvious how best to compare the strategies developed in this project to the simple strategy used by Ardeshiri et al.[3]. As a reminder, in that report the copepod agents could measure the strain magnitude directly and would jump whenever the measured strain magnitude was higher than a certain threshold. This is hard to compare fairly to this model for a number of reasons:

- The agents don't have access to the same information. The Ardeshiri agents know the absolute strain magnitude while the ones in this project only know two projections of the strain.
- They don't have the same freedom of movement. The agents in this project could only jump at fixed timesteps while the Ardeshiri ones could jump at any time as long as the previous jump had ended. On the other hand, the agents in this project can jump in different directions which the Ardeshiri ones could not.
- It is unclear what the best metric to measure them against each other is. The agents in this project are specifically trained to optimize the value of $R_t = -S(t)^2$ at certain timesteps so measuring their performance by something else might be unfair.

Even though it might not be completely fair, it seems that the best comparison would be to measure the reward that the agents in this project has been trained with and to allow the agents using the Ardeshiri strategy to optimize their strain magnitude threshold to achieve the same goal. By doing a parameter sweep similar to the one done in figure 4.1 the optimal value for the strain magnitude threshold is determined to be 0.87. Running the simulations with this threshold gives the comparison in table 4.4:

Table 4.4: Results of a comparison between the simple Ardeshiri strategy and one learned through Q-learning.

	Mean total reward	Standard deviation
Q-learning	-76.8	2.10
Ardeshiri	-68.4	1.90

The results of the comparison favor the Ardeshiri strategies, though it is important to remember that those agents have an advantage in information. The question still remains which sensory input most closely resembles reality, it could be possible to determine this experimentally by observing the trajectories of copepods as the two different sensory input should lead to distinctly different trajectories.

4.4 Conclusion

The model was succesfully programmed and simulates the movement of the copepods in the intended way. In the end, the learning algorithm did not find a strategy that outperformed the simple Ardeshiri strategy. It did, however, come reasonably close with a less demanding sensory input. The fact that no advanced strategy emerged to beat the simple Ardeshiri strategy can be used to argue that the assumptions made by Ardeshiri et al. regarding the validity of their strategy were correct.
Some interesting patterns emerged in the the strategies learned by the copepods depending in the flow. It was deemed likely that this was due to properties in the two flows studied that made the agents more likely to be oriented diagonally.

4.4.1 Future work

There are a lot of areas where this model could be expanded in the future. A larger Q-matrix with more states would allow for even more sophisticated strategies. A 3D model might be able to explore aspects of the copepods movement in ways not possible in two dimensions. The model could also be made more advanced in a lot of other ways. For example, predator agents could be introduced and copepods could be made to interact with each other. It could also be of interest to test the model in new flows, notably the channel flow.The model could also be modified to simulate other kinds of copepods. The ones in this project moved only by jumping, but there are other kinds of copepods that also or only move by swimming at a more constant pace.
There are also some unsolved problems in this project that could be investigated in the future, mainly the discrepancy between analytical and experimental results regarding the relation of the stochastic scale factor and the mean strain magnitude discussed in section 2.1.2 as well as the size different of the peaks in figure 4.7.

Bibliography

[1]I. Fouxon, S. Souissi, and M. Holzner, "Zooplankton can actively adjust their motility to turbulent flow", 1–9 (2017).

[2]E. J. Buskey, P. H. Lenz, and D. K. Hartline, "Escape behavior of planktonic copepods in response to hydrodynamic disturbances : high speed video analysis", **235**, 135–146 (2002).

[3]H. Ardeshiri, I. Benkeddad, F. G. Schmitt, S. Souissi, F. Toschi, and E. Calzavarini, "Lagrangian model of copepod dynamics: Clustering by escape jumps in turbulence", Physical Review E **93**, 1–11 (2016).

[4]B. J. Gemmell, D. Adhikari, and E. K. Longmire, "Volumetric quantification of fluid flow reveals fish's use of hydrodynamic stealth to capture evasive prey", Journal of the Royal Society Interface **11** (2014) `10.1098/rsif.2013.0880`.

[5]K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction", CSBJ **13**, 8–17 (2015).

[6]S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning", 250–257 (2005).

[7]C. J. C. H. Watkins and P. Dayan, "Q-learning", Machine Learning **8**, 279–292 (1992).

[8]Sutton, *Reinforcement Learning: An Introduction* (2018).

[9]S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale, "Flow Navigation by Smart Microswimmers via Reinforcement Learning", 1–5 (2017).

[10]E. D. Fylladitakis, "Kolmogorov Flow: Seven Decades of History", Journal of Applied Mathematics and Physics **06**, 2227–2263 (2018).

[11]G. Taylor and A. Green, "Mechanism of the Production of Small Eddies from Large Ones", **151** (1936).

[12]K. Gustavsson and B. Mehlig, "Statistical models for spatial patterns of heavy particles in turbulence", Advances in Physics **65**, 1–57 (2016).

[13]K. Gustavsson, F. Berglund, P. R. Jonsson, and B. Mehlig, "Preferential sampling and small-scale clustering of gyrotactic microswimmers in turbulence", Phys. Rev. Lett. **116**, 108104 (2016).

[14]M. Borgnino, K. Gustavsson, F. D. Lillo, G. Boffetta, M. Cencini, and B. Mehlig, "Alignment of non-spherical active particles in chaotic flows", 1–5.

[15]G. B. Jeffrey, "The motion of ellipsoidal particles immersed in a viscous fluid", Proceedings of the Royal Society of London **102**, 011008 (1922).

[16]T. Kiørboe, A. Andersen, V. J. Langlois, H. Henrik, and T. Bohr, "Mechanisms and feasibility of prey capture in ambush-feeding zooplankton", 6–11 (2009).