8-31-2022

Low-reynolds-number locomotion via reinforcement learning

Yuexin Liu

New Jersey Institute of Technology

ABSTRACT

LOW-REYNOLDS-NUMBER LOCOMOTION VIA REINFORCEMENT LEARNING

by
Yuexin Liu

This dissertation summarizes computational results from applying reinforcement learning and deep neural network to the designs of artificial microswimmers in the inertialess regime, where the viscous dissipation in the surrounding fluid environment dominates and the swimmer's inertia is completely negligible. In particular, works in this dissertation consist of four interrelated studies of the design of microswimmers for different tasks: (1) a one-dimensional microswimmer in free-space that moves towards the target via translation, (2) a one-dimensional microswimmer in a periodic domain that rotates to reach the target, (3) a two-dimensional microswimmer that switches gaits to navigate to the designated targets in a plane, and (4) a two-dimensional microswimmer trained to navigate in a non-stationary environment.

The first and second studies focus on how reinforcement learning (specifically model-free, off-policy Q-learning) can be applied to generate one-dimensional translation (part 1) or net rotation (part 2) in low Reynolds number fluids. Through the interaction with the surrounding viscous fluid, the swimmer learns to break the time-reversal symmetry of Stokes flow in order to achieve the maximum displacement (reward) either in free-space or in a periodic domain.

In the third part of the dissertation, a deep reinforcement learning approach (proximal policy optimization) is utilized to train a two-dimensional swimmer to develop complex strategies such as run-and-tumble to navigate through environments and move towards specific targets. Proximal policy optimization contains actor-critic model, the critic estimates the value function, the actor updates the policy distribution in the direction suggested by the critic. Results show the artificial trained

swimmer can develop effective policy (gaits) such as translation and rotation, and the swimmer can move to specific targets by combining these gaits in an intelligent way. The simulation results also show that without being explicitly programmed, the trained swimmer is able to perform target navigation even under flow perturbation.

Finally, in the last part of the dissertation, a generalized step-up reinforcement method with deep learning is developed for an environment that changes in time. In this work, the traditional reinforcement learning is combined with a high confidence context detection, allowing the swimmer to be trained to navigate amphibious non-stationary environments that consist of two distinct regions. Computational results show that the swimmer trained by this algorithm adapts to the environments faster, while developing more effective locomotory strategies in both environments, than traditional reinforcement learning approaches. Furthermore, the effective policies with traditional strategies are compared and analyzed. This work illustrates how deep reinforcement learning method can be conveniently adapted to a broader class of problems such as a microswimmer in a non-stationary environment. Results from this part highlight a powerful alternative to current traditional methods for applications in unpredictable, complex fluid environments and open a route towards future designs of “smart” microswimmers with trainable artificial intelligence.

LOW-REYNOLDS-NUMBER LOCOMOTION VIA REINFORCEMENT LEARNING

by
Yuexin Liu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
and Rutgers, The State University of New Jersey – Newark
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Mathematical Sciences

Department of Mathematical Sciences
Department of Mathematics and Computer Science, Rutgers-Newark

August 2022

APPROVAL PAGE

LOW-REYNOLDS-NUMBER LOCOMOTION VIA REINFORCEMENT LEARNING

Yuexin Liu

Dr. Yuan-Nan Young, Dissertation Co-Advisor Date
Professor of Mathematical Sciences, NJIT

Dr. On Shun Pak, Dissertation Co-Advisor Date
Associate Professor of Mechanical Engineering, Santa Clara University
Santa Clara CA

Dr. Jonathan Luke, Committee Member Date
Professor of Mathematical Sciences, NJIT

Dr. Michael Siegel, Committee Member Date
Professor of Mathematical Sciences, NJIT

Dr. David Shirokoff, Committee Member Date
Associate Professor of Mathematical Sciences, NJIT

BIOGRAPHICAL SKETCH

Author: Yuexin Liu

Degree: Doctor of Philosophy

Date: August 2022

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mathematical Sciences,
  New Jersey Institute of Technology, Newark, NJ, 2022
- Master of Science in Computer Engineering,
  Florida International University, Miami, FL, 2015
- Bachelor of Science in Mathematical Sciences,
  Northeast Dianli University, Jilin, China, 2011

Major: Mathematical Sciences

Presentations and Publications:

Z. Zou, Y. Liu, A. C. H. Tsang, Y.-N. Young, O. S. Pak, "Linear 3-sphere device in a non-stationary environment via reinforcement learning context detection," Manuscript, 2022.

Z. Zou, Y. Liu, Y.-N. Young, O. S. Pak, A. C. H. Tsang, "Gait switching and targeted navigation of microswimmers via deep reinforcement learning," *Communications Physics*, **5,** 128, 2022.

Y. Liu, Z. Zou, A. C. H. Tsang, O. S. Pak, Y.-N. Young, "Mechanical rotation at low Reynolds number via reinforcement learning," *Physics of Fluids*, **33** 062007, 2021.

Z. Zou, Y. Liu, O. S. Pak, Y.-N. Young, A. C. H. Tsang, "Artificial microswimmers via reinforcement learning," *American Physical Society Division of Fluid Dynamics Meeting Abstracts*, Y11.00012, 2022.

Y. Liu, Z. Zou, O. S. Pak, Y.-N. Young, "Generating net rotational motion at low Reynolds number via reinforcement learning," *American Physical Society Division of Fluid Dynamics Meeting Abstracts*, U09.006, 2020.

Y. Liu, O. S. Pak, Y.-N. Young, "Generating net rotational motion at low Reynolds number via reinforcement learning," *73rd Annual Meeting of the American Physical Society Division of Fluid Dynamics*, November 22024, 2020.

G. Nita, M. Georgoulis, I. Kitiashvili, V. Sadykov, E. Camporeale, A. Kosovichev, H. Wang, V. Oria, J. Wang, R. Angryk, B. Aydin, A. Ahmadzadeh, X. Bai, T. Bastian, S. F. Boubrahimi, B. Chen, A. Davey, S. Fereira, G. Fleishman, D. Gary, A. Gerrard, G. Hellbourg, K. Herbert, J. Ireland, E. Illarionov, N. Kuroda, Q. Li, C. Liu, Y. Liu, H. Kim, D. Kempton, R. Ma, P. Martens, R. McGranaghan, E. Semones, J. Stefan, A. Stejko, Y. Collado-Vega, M. Wang, Y. Xu, S. Yu, "Machine learning in heliophysics and space weather forecasting: a white paper of findings and recommendations," *arXiv*, 2020.

Y. Liu, "Low-Reynolds-number locomotion via reinforcement learning," *Frontiers in Applied & Computational Mathematics (FACM)*, Newark, NJ, May 20-21, 2022, Oral Presentation.

Y. Liu, "Gait switching and targeted navigation of microswimmers via deep reinforcement learning," *Northeast Complex Fluids & Soft Matter (NCS) Workshop*, Jan 14, 2022, Oral Presentation.

Y. Liu, "Mechanical rotation at low Reynolds number via reinforcement learning," *Northeast Complex Fluids & Soft Matter (NCS) Workshop*, Aug 20, 2021, Oral Presentation.

Y. Liu, "Introduction to Q-learning," *NJIT Machine Learning Group*, Dec 10, 2020, Oral Presentation.

Y. Liu, "Using reinforcement Q-learning to train a swimmer in a viscous fluid," *NJIT MathBio Group Meeting*, Nov 18, 2020, Oral Presentation.

Y. Liu, "Training a Stokes swimmer using machine learning," *Northeast Complex Fluids & Soft Matter (NCS) Workshop*, Jan 19, 2020, Oral Presentation.

Y. Liu, "Deep learning in modeling complex systems (Project ID# F18-12)," *NJIT NSF I-Corp*, 2018, Oral Presentation.

This dissertation is dedicated to my beloved parents, Mingli Liu and Li Pan, who always love, encourage and support me unconditionally. I am also dedicated to my son, Ethan Jiang and my husband, Haodi Jiang, who bring love and joy.

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to my advisors, Dr. Yuan-Nan Young and Dr. On Shun Pak, for providing the insightful knowledge and feedback throughout my study and their invaluable guidance and support of my study and research throughout the years. Their expertise and professional dedication to scientific research have had a profound effect upon me. They have taught me, with great patience, how to conduct research project efficiently. The journey of being their student is truly precious experience and a lifelong benefit.

Second, I would like to thank Dr. Jonathan Luke, Dr. Michael Siegel and Dr. David Shirokoff, for being an important part of my scientific journey. I truly appreciate for their valuable time and suggestions for my dissertation research and serving on my dissertation committee.

It is worthwhile to note that without the generous support from the Department of Mathematical Sciences, it would be harder for me to complete the PhD program. And I would like to thank Ms. Clarisa Gonzalez-Lenahan and Dr. Sotirios Ziavras who helped me to revise this dissertation.

Besides, I would like to thank my collaborators Dr. Alan Cheng Hou Tsang and Zonghao Zou for their contributions to the work in the dissertation.

Most importantly, I truly appreciate my parents, Mingli Liu and Li Pan, for their unwavering support and belief in me. To my husband, Haodi Jiang, for being my emotional anchor. Especially to my son, Ethan Jiang, thank you for the happiness you bring to me every day and your sweetest love throughout the years. I am truly thankful for having you all in my life.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES (Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivation

Swimming at microscopic scale, the dominance of viscous forces over inertial forces imposes stringent constraints on locomotion at low Reynolds numbers (Re) [62, 79]. In the absence of inertia, common propulsion strategies become ineffective in the microscopic world. Microorganisms have developed various locomotion strategies to escape the constraints [31, 109]. In the past several decades, enormous efforts have addressed the physical principles behind cell motility [30, 60, 83], which helps to improve the general understanding of locomotion at low Re. Purcell first proposed an elegant approach to generate net translation using the kinematically irreversible motions [79]. Najafi *et al.* designed the one-dimensional three linked spheres in the low Reynolds number fluid [71]. With designed locomotory gaits, it can generate direct translation which allows the swimmer to propel along the horizontal-axis. In addition to net translation, a more recent prototype known as Purcell's "rotator" has been proposed by Dreyfus *et al.* [25] that will allow a circular three-sphere swimmer to generate net rotation in a nonreciprocal manner. With more advanced technology involved, a variety of artificial microswimmers [28, 48, 89] have engendered in recent years, which are capable of navigating biological environments showing promising opportunities for biomedical and environment applications, such as micro-structures target at cancer cells, perform optical surgery and targeted drug delivery, or use the nanobots to remove Au droplets in the fluid [33, 35, 77].

Despite the successful applications of these synthetic microswimmers over the past decades, existing microswimmers are having pre-designed locomotory gaits for a specific type of fluid medium or environmental condition [14, 67, 73]. However, the locomotory gaits that are effective in one certain fluid may turn out to be ineffective in a different medium. Thus, the performance of the microswimmers with pre-designed locomotory gaits may not be robust to the change of the environment. On the contrary, the natural microorganisms present robust performance of locomotion through changing environments by the adaptivity of their locomotory gaits to the environments [9, 65]. The adaptability of the design of the micro-structures just like the microorganisms remains formidable in complicated environment with unpredictable factors. The use of soft active materials and several approaches using modular microrobotics have been proposed to tackle these challenges [16, 47, 50, 78, 97, 101]. Here the reinforcement learning method is employed to provide a new approach in the design of the swimmers in low Re regime. Without prior knowledge, such as the fluid medium or the pre-designed locomotory gaits, the reinforcement learning approach can enable the artificial intelligent systems to perform complicated missions without explicit programming [55]. This method also sparked new directions in fluid mechanics, such as swarms of bacterial [53], wake detection [20], soaring birds [36,82], fish schooling [37, 38, 54, 100], turbulence modeling [59], and navigation planning [18, 70]. This study focuses on the use of reinforcement learning in the fundamental challenge of generating self-propulsion at low Re. This work contributes to the development of the design of the micro-structures by utilizing reinforcement learning method. Without pre-designed locomotory gaits, this method is adaptive, and can let a self-learning swimmer develop the effective propulsion strategies based on the interactions with the environments.

1.2 Main Results and Dissertation Structure

The remaining of this dissertation is organized as follows. Chapter 2 introduces the use of reinforcement learning, specifically Q-learning, to the design of the locomotory gaits of a one-dimensional microswimmer in free-space that moves towards the target via translation. The results demonstrate that, without requiring any prior knowledge on low Re locomotion, the linear swimmer can generate effective strategies to escape Purcell's scallop constraints for self-propulsion and can recover the strategy proposed by Najafi and Golestanian [71]. This method provides a new approach to solve the challenge of the microswimmers in complex environments.

Chapter 3 focuses on a one-dimensional microswimmer in a periodic domain that rotates to reach the target. The use of reinforcement learning method to the design of three-sphere rotator in net rotation generation. The result is consistent with the strategy proposed by Dreyfus *et al.* [25] and this alternative approach in this work is particularly desirable when a machine explores an environment with unknown properties or bypasses the challenging of designing locomotory gaits in advance in these situations.

Chapter 4 presents a two-dimensional microswimmer that switches gaits to navigate to the designated targets in a plane. This work utilizes the deep reinforcement learning method to let a three-sphere swimmer self-learns three distinct locomotory gaits: steering, transition and translation. The result shows the swimmer is capable of complex geometry tracing and shown to be robust against flow perturbation. This ability of targeted navigation via adaptive gait-switching is particularly desirable for the development of smart artificial microswimmers that can perform complex biomedical tasks such as targeted drug delivery and microsurgery in an autonomous manner.

Chapter 5 discusses a two-dimensional microswimmer trained to navigate in a non-stationary environment based on the context detection and deep reinforcement learning method. In contrast to previous works that utilize reinforcement learning for a single environment, this study focuses on the device's ability to navigate non-stationary environments without prior knowledge of when an environment change would occur. The results demonstrate that the reinforcement learning context detection method enables the device to master specialized locomotory gaits for each environment and detect a context change quickly. The context detection and deep RL approach present here offers a new avenue for designing artificial devices in navigating complex fast changing environments.

Finally, in Chapter 6, the dissertation ends up with a few detailed discussions to summarize the main results mentioned earlier in this dissertation, the conclusions to be drawn, and the possible directions for future work.

CHAPTER 2

USING Q-LEARNING TO TRAIN A STOKESIAN SWIMMER

2.1 Background and Related Work

Machine learning has been applied to an increasing range of physics and engineering, and recently it has been applied in zero-Reynolds number flow for flow control and designing of swimmers [36–38, 55, 59, 69, 70, 82, 100, 101]. Artificial micro-swimmers in engineering and medical applications such as drug delivery and cell manipulation show great success [33, 35, 77], however, hydrodynamic interactions in low Reynolds number environments and the uncontrolled environmental factors will also influence the swimmer's behavior [79, 94]. Here, we present a reinforcement learning paradigm to design a new set of self-learning, adaptive linear N-sphere swimmer and Purcell's rotator [24] in a viscous Stokes fluid environment. Different from the typical designed autonomous swimmers [71, 79], the traditional microswimmers are typically designed to have fixed locomotory gaits for a particular type of medium or environmental condition. However, gaits that are optimal in one medium may become ineffective in a different medium; hence, locomotion performance of synthetic microswimmers with fixed locomotory gaits may not be robust to environmental changes [16, 47, 50, 78, 97, 101]. In contrast, natural organisms show robust locomotion performance across varying environments by adapting their locomotory gaits to the surroundings. Thus, we do not prescribe any propulsion gaits but allow the swimmer to self-learn its own strategy based on its interactions with the surrounding environment through reinforcement learning. We show the ability of the linear swimmer to obtain the optimal propulsion policies. Our study illustrates the potential of reinforcement learning in fluid mechanics and provides a new way for designing smart artificial swimmers.

Locomotion at the microscopic scale encounters stringent constraints due to the absence of inertia. As a result of kinematic reversibility, Purcell showed that animals such as scallops that are equipped with a single hinge cannot swim using the simple opening and closing procedure [79], since the motion is reversible, after finishing a cycle the scallop will end up being where it initially was. He also argued that swimming strategies can only be successful in this regime if they involve a cyclic and non-time-reversible motion. Here we apply a recent framework based on reinforcement learning [99] to generate net translational motion at low Reynolds numbers. Without prior knowledge of locomotion, the system develops effective policies based on its interactions with the surrounding environment. We compare the results with previously known strategies and remark on the possibility of more complex maneuvers.

2.2 Formulation

2.2.1 Hydrodynamics

The interaction between the spheres and the surrounding viscous fluid is governed by the Stokes equation subject to the incompressible condition:

$$\begin{cases} \nabla p = \eta \nabla^2 \mathbf{u}, \text{in } \Omega, \\ \nabla \cdot \mathbf{u} = 0, \text{in } \Omega, \\ \mathbf{u} = \mathbf{V_i}, \text{ on } \Gamma, \\ \mathbf{u} \to 0, \text{when } r \to \infty, \end{cases} \tag{2.1}$$

where p represents the pressure field in the medium, $\mathbf{u}$ is the velocity field. Γ is the boundary of the sphere and Ω is the exterior domain. Sphere i is moving inside the fluid with velocity vectors $\mathbf{V_i}$ (with the index i denoting sphere i). The above equations are solved with zero velocity at infinity and no-slip boundary conditions on the spheres.

The variables that determine the dynamics of the spheres are their velocities $\mathbf{V}_i$ and the forces $\mathbf{F}_i$ acting on them. By solving the above equations, we will be able to obtain the fluid velocity in the medium, and the corresponding stress tensor will give us the required forces on the spheres. Since the Stokes equation is linear, the velocity fields produced by each of the spheres simply add up, and we can express the relation in the general form:

$$\mathbf{V}_i = \sum_{j=1}^{N} H_{ij}\mathbf{F}_j. \tag{2.2}$$

Assuming the separations between the spheres are sufficiently larger than the sizes of the spheres. The Oseen tensor H_{ij} is given by

$$H_{ij} = \begin{cases} \mathbf{I}/6\pi\eta R & \text{if } i = j, \\ \\ (1/8\pi\eta\,|\mathbf{r}_{ij}|)\,(\mathbf{I} + \mathbf{r}_{ij}\mathbf{r}_{ij}/|\mathbf{r}_{ij}|^2) & \text{otherwise,} \end{cases} \tag{2.3}$$

to the leading order. Here η is the fluid viscosity, $\mathbf{I}$ is the identity matrix, and $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$ is the distance between spheres i and j.

Including the condition that there are no external forces such as gravity, the system of spheres should be force-free and torque-free:

$$\begin{cases} \sum_{i=1}^{3} \mathbf{F}_i = 0, \\ \sum_{i=1}^{3} \mathbf{F}_i \times \mathbf{r_i} = \mathbf{0}. \end{cases} \tag{2.4}$$

The Najafi and Golestanian swimmer [71] is shown in Figure 2.1, that consists of three spheres with radius R and are connected by negligible arms along the horizontal direction.

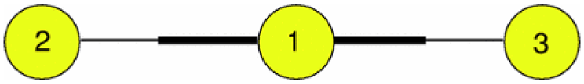


Figure 2.1 Three linked spheres connected by two rods of negligible thickness.
Source: [71]

2.2.2 Reinforcement Learning

Reinforcement learning is a type of Machine Learning algorithm which allows software agents and machines to automatically determine the ideal behavior within a specific context, to maximize its performance. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

Reinforcement learning differs from other types of machine learning paradigm. As shown in Figure 2.2, in Supervised Learning, we are given a dataset, which consists of data x and labels y. In this setting, for each example, we are provided the correct label (classification problems) or a correct output (regression problems). Our goal is to learn a model that takes the input: data x and learns to predict the labels y. The example here is we have a picture of an apple and we want to train the model to predict, there is an apple in the picture (it's just a classification problem). In contrast, if no labels are provided and we only have access to the data. Then we move to unsupervised learning which refers to the methods that to find the underlying structure in some data. In this case, we give two pictures of apples but the machine don't know what they are because no labels are here, by analyzing the structure of these two, the model knows these two things are the same even if we don't know that they are specifically apples. However, in RL, we are given data in the form of what are called state action pairs, we are dealing with making decisions and comparing actions that could be taken, rather than making predictions, A RL agent may interact with the world and receive some immediate, partial feedback signal, commonly called a reward for each interaction. The agent here has no knowledge of the background, the agent somehow learns to pick actions that will maximize a long term cumulative reward. Because of the incomplete feedback provided by the reward signal we can consider the RL to lie somewhere between supervised learning which gives strong feed back with labeled data and unsupervised learning, with no feedback or labels.

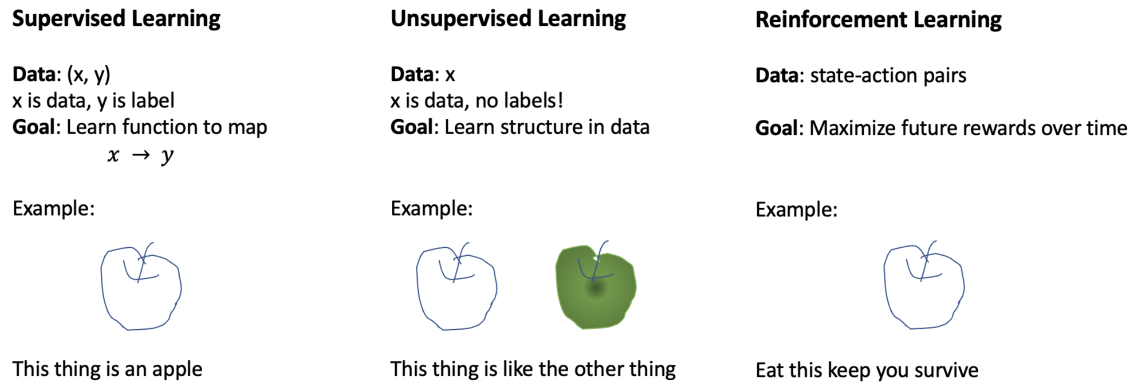


Figure 2.2 Example of classes of learning problems.

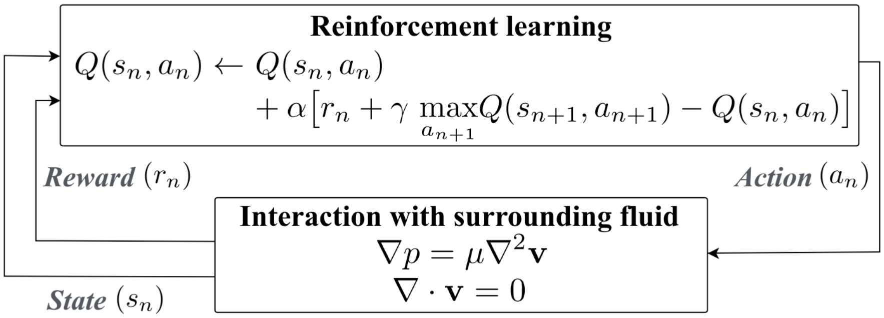

Reinforcement learning
$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha\left[r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)\right]$
Reward (r_n)
Action (a_n)
Interaction with surrounding fluid
$\nabla p = \mu \nabla^2 \mathbf{v}$
$\nabla \cdot \mathbf{v} = 0$
State (s_n)

Figure 2.3 Schematic of reinforcement learning of a swimmer that progressively learns how to swim by interacting with the surroundings.
Source: [99].

Using reinforcement learning can let the swimmer progressively learn how to act by interacting with the surrounding medium. The schematic of the method is shown in Figure 2.3 [99]. For a given configuration of the swimmer (the state, s_n) in the n-th learning step, the swimmer can perform one translational (extend or shorten the arm) or rotational (contract or increase the angle between spheres) action (the action, a_n) to transforms from the current state to the next state/new state. Such an action results in a displacement of the swimmer's body centroid (the reward, r_n), which measures the immediate quality of the action relative to its final goal.

The implementation is carried out by a standard Q-learning algorithm.The experiences obtained by the agent/swimmer is stored in a Q-matrix, with $Q(s_n, a_n)$ is an action-value function that captures the expected long-term reward for taking the action $a - n$ given the corresponding state s_n. At each training step, the Q entries will be updated using the following equation,

$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)], \tag{2.5}$$

with:

1. α: learning rate ($0 \leq \alpha \leq 1$), determines to what extent new information overrides old information in the Q-matrix. Fixed $\alpha = 1$ to maximize learning in a fully deterministic system.

2. r_n: immediate reward.

3. $\max_{a_{n+1}} Q(s_{n+1}, a_{n+1})$: maximum future reward at the next state.

4. γ: discount factor, assigns a weight to immediate versus future rewards a_{n+1} ($0 \leq \gamma < 1$).

   (a) when γ is small, the agent/swimmer is shortsighted and tends to maximize the immediate reward;

   (b) when γ is large, the agent/swimmer is farsighted and focuses more on future rewards.

5. $\epsilon-$greedy scheme (trade-off between exploration and exploitation): in each learning step, the swimmer chooses the best action advised by the Q-matrix with a probability $1 - \epsilon$ or takes a random action with a small probability ϵ, which allows the swimmer to explore new solutions and avoids being trapped in only locally optimal policies. $\epsilon-$greedy is a simple method to balance exploration and exploitation by choosing between exploration and exploitation randomly. The $\epsilon-$greedy, where ϵ refers to the probability of choosing to explore, exploits most of the time with a small chance of exploring.

$$\textit{Action at learning step}(t) \begin{cases} \max Q_t(a) & \textit{with probability } 1 - \epsilon \\ \textit{any action}(a) & \textit{with probability } \epsilon \end{cases} \tag{2.6}$$

Pseudo Code:

```
p = random()
if p < epsilon:
    pull random action
else:
    pull current-best action
```

2.2.3 Reinforcement Learning for Three-sphere Swimmer

2.2.4 Problem Setup

1. Goal: learn how to act by interacting with the fluid environment;

2. Agent(the N-sphere swimmer): N-spheres connected with $N-1$ extensible rods of negligible diameters, each sphere has a radius R and each rod has length l that can contract/extend by e (in the paper, they set $l = 10R, e = 4R$);

3. State(s_n): configuration of the swimmer, N spheres has total 2^{N-1} configurations and each configuration can transition to $N-1$ configurations by extending or contracting one of the connecting rod, for example 3-sphere swimmer has 4 states;

4. Action(a_n): extend/contract one of its rods;

5. Reward(r_n): certain displacement of the body centroid of the swimmer(c_n);

$$r_n = \hat{e}\Delta c_n$$

   (a) body centroid of the swimmer is defined as : $c_n := \sum_{i=1}^{N} \frac{x_i(n)}{N}$

   (b) $x_i(n)$: position vector of i-th sphere;

   (c) $\hat{e}$: desired direction (unit vector);

   (d) Δc_n: transformation between states displaces c_n, $\Delta c_n = c_{n+1} - c_n$

6. D: cumulative displacement of the body centroid

$$D = \sum_n \hat{e}\Delta c_n$$

7. update $Q(s_n, a_n)$

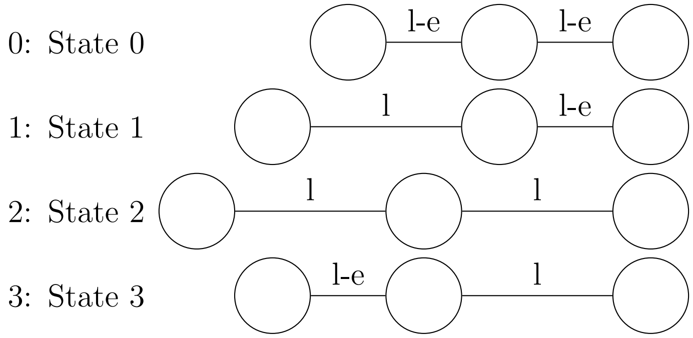
$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[r_n + \gamma \max Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)] \qquad (2.7)$$

(a) α is the learning rate ($0 \leq \alpha \leq 1$), which determines to what extent new information overrides old information in the Q-matrix. Unless otherwise specified, we fixed $\alpha = 1$ to let the learning occur quickly;

(b) γ is the discount factor, which assigns a weight to immediate versus future rewards ($0 \leq \gamma < 1$). When γ is small, the swimmer is shortsighted and tends to maximize the immediate reward; when γ is large, the swimmer is farsighted and focuses more on future rewards;

8. In addition, we add an ϵ-greedy selection scheme (trade-off between exploration and exploitation): in each learning step, the swimmer chooses the best action advised by the Q-matrix with a probability $1 - \epsilon$ or takes a random action with a small probability ϵ, which allows the swimmer to explore new solutions and avoids being trapped in only locally optimal policies. $\epsilon-$greedy is a simple method to balance exploration and exploitation by choosing between exploration and exploitation randomly. The $\epsilon-$greedy, where $\epsilon = 0.05$ refers to the probability of choosing to explore, exploits most of the time with a small chance of exploring.

To implement the algorithm, let's start by recollecting the states and map each of the states to numbers:



Specify the parameters of the Q-Learning algorithm: ϵ, γ (discount factor), number of steps n and add $\epsilon-$greedy policy for the following steps

- if $c = 0, p = 1 - \epsilon = 0.95$, the agent will choose the max Q-value;
- if $c = 1, p = \epsilon = 0.05$, the agent will act randomly.

The flowchart in Figure 2.4 shows the training process of Q-learning. Each of the blue-colored box is one step. First is to initialize a Q-matrix. A Q-matrix or

Q-table is just a fancy name for the lookup table where we calculate the maximum expected future rewards for an action at certain state. Basically, this table will guide us to the best action at each state.

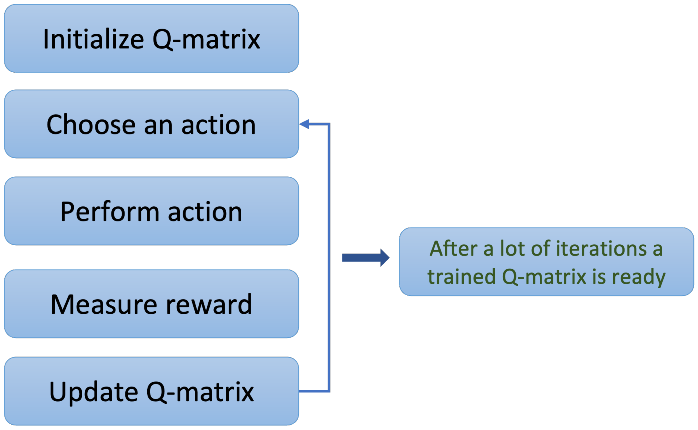
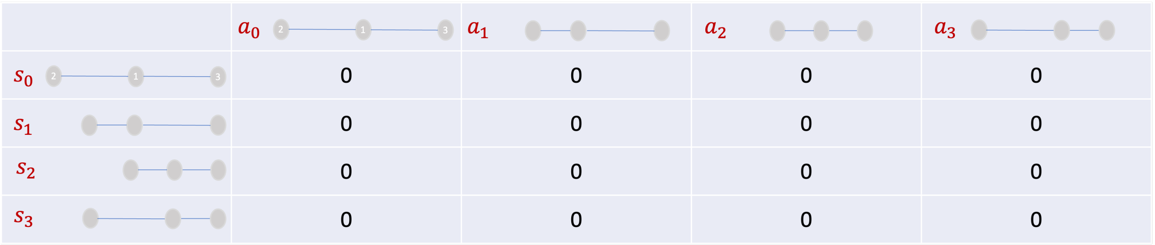


Figure 2.4 Flowchart of Q-learning method.

First, we will use the linear three sphere swimmer as an example to build a Q-table. It consists of n rows, with n represents the number of states of the linear three sphere swimmer and m columns, where m denotes the number of actions for each state. For the linear three sphere swimmer, it has four states and four actions at each state. For instance, when a swimmer is at state 1, it can either extend the left arm to transit to state 0 or it can contract the right arm to transit to state 2. In our study, only one degree of freedom is allowed for each learning step, which means, we can only perform one actuation at each step.

In the beginning, all the values of Q-entries are zeros, as shown in Figure 2.5, then the training start and this Q-table will be improved at each iteration.

For instance, at the first learning step, the swimmer is at state 0 and knows nothing about the environment, it picks a random action, a_1, which is contract the left arm. Due to this particular action, we calculate the displacement for the case of $D = 10R$, arm length change is $4R$, the middle sphere will have a negative x-direction displacement about $-1.35R$. The corresponding Q-entry, $Q(s_0, a_1)$ can be updated

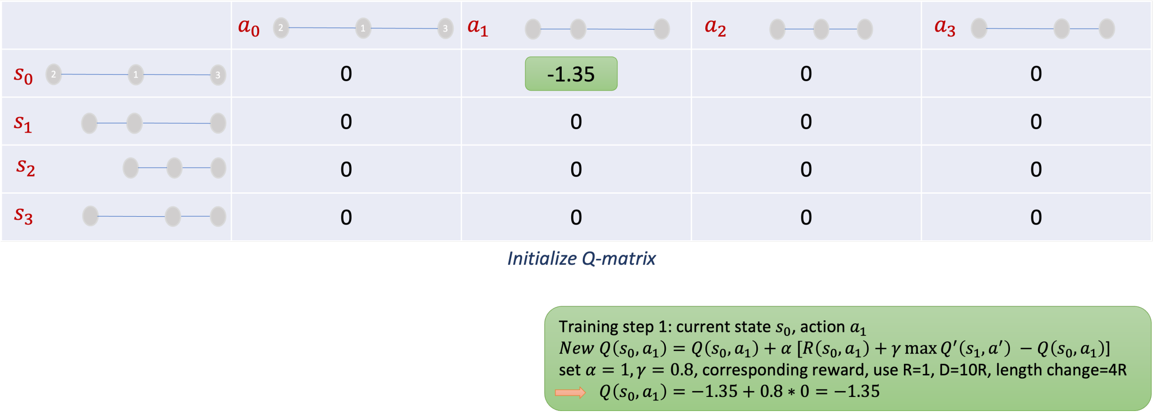


	a_0	a_1	a_2	a_3
s_0	0	0	0	0
s_1	0	0	0	0
s_2	0	0	0	0
s_3	0	0	0	0

Figure 2.5 Initialize Q-table. In the beginning of the Q-learning, all the entries in this Q-table are initialized to zeros.

using Equation 2.7, with learning rate $\alpha = 1$, discount factor $\gamma = 0.8$.

$$Q(s_0, a_1) = Q(s_0, a_1) + \alpha[r_n + \gamma \max Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)]$$
$$\xrightarrow[\gamma=0.8]{\alpha=1} r_n + 0.8 \max Q(s_{n+1}, a_{n+1}).$$



	a_0	a_1	a_2	a_3
s_0	0	-1.35	0	0
s_1	0	0	0	0
s_2	0	0	0	0
s_3	0	0	0	0

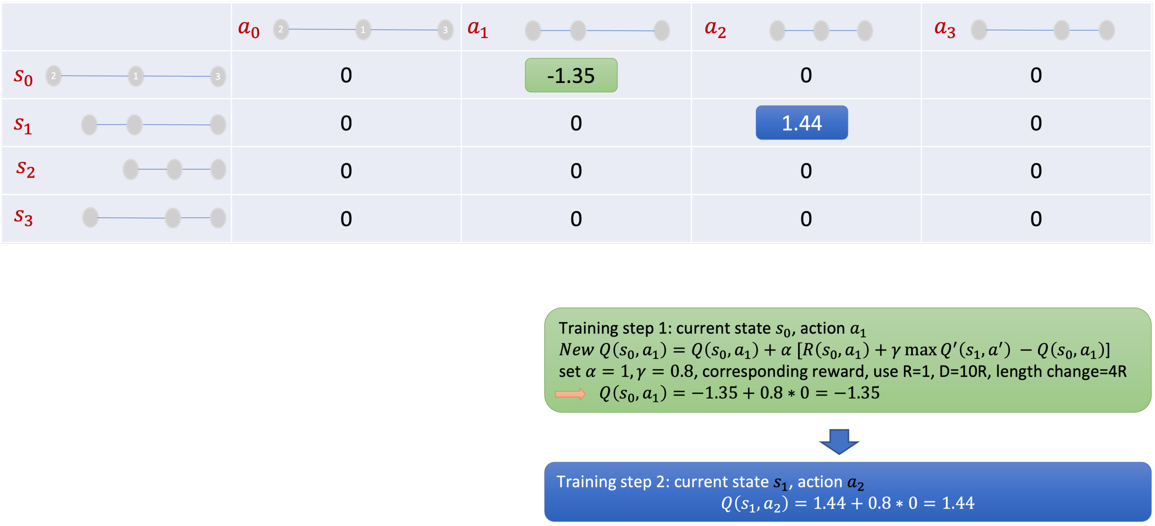
Initialize Q-matrix

Training step 1: current state s_0, action a_1
$New\ Q(s_0, a_1) = Q(s_0, a_1) + \alpha\ [R(s_0, a_1) + \gamma \max Q'(s_1, a') - Q(s_0, a_1)]$
set $\alpha = 1, \gamma = 0.8$, corresponding reward, use R=1, D=10R, length change=4R
$\Rightarrow Q(s_0, a_1) = -1.35 + 0.8 * 0 = -1.35$

Figure 2.6 Q-table update at the first learning step.

Similarly for the next learning step, now the swimmer is at state 1, and randomly pick an action a_2. According to this particular action, the middle of the swimmer will have a displacement in the positive x-direction by an amount of $1.44R$. Substitute the values in the update formula, we can get the corresponding Q-entry, $Q(s_1, a_2) = 1.44$, as shown in Figure 2.7.

To use this Q-table as a reference to dictate the motion of the swimmer. For instance, in Figure 2.7, suppose we are currently at state 1, since only one actuation is



	a_0	a_1	a_2	a_3
s_0	0	-1.35	0	0
s_1	0	0	1.44	0
s_2	0	0	0	0
s_3	0	0	0	0

Training step 1: current state s_0, action a_1

$New\ Q(s_0, a_1) = Q(s_0, a_1) + \alpha\ [R(s_0, a_1) + \gamma \max Q'(s_1, a')\ - Q(s_0, a_1)]$

set $\alpha = 1, \gamma = 0.8$, corresponding reward, use R=1, D=10R, length change=4R

⇨ $Q(s_0, a_1) = -1.35 + 0.8 * 0 = -1.35$

Training step 2: current state s_1, action a_2

$Q(s_1, a_2) = 1.44 + 0.8 * 0 = 1.44$

Figure 2.7 Q-table update at the second learning step.

allowed for each step, we can only transfer to state 0 or state 2 in this constraint. As the value of $Q(s_1, a_0)$ is less than $Q(s_1, a_2)$, therefore, at state 2, action 2 is preferred than action 1.

Above are the iterative process of updating the values, as we start to explore the surrounding environment, this Q-function gives up better and better approximations by continuously updating the Q-entries in the Q-table.

2.3 Results and Discussion

2.3.1 Linear Three-sphere Swimmer

Several authors have described models of swimmers at low Reynolds number. In 2004, Najafi and Golestanian [42] proposed a one dimensional swimmer comprising three connected spheres. Their model uses the cyclic motion that breaks time reversibility. The swimmer is composed of three-spheres with fixed radius R. The central sphere is connected to two other spheres by negligible rods and separated by an angle of 180° along the x-direction. The swimmer is immersed in the low-Reynolds number fluid with viscosity η. There are two internal engines on the middle sphere, which can make a nonreciprocal motion that is needed to propel the whole system. The microstructure moves by shortening and extending the lengths of the arms in a time irreversible and periodic manner. The parameters for this device are,

1. D: the distance between the central sphere and an outer sphere at the maximum arm length.

2. ε: the distance the arm shortens.

3. W: the speed at which the arms change their lengths.

4. R: the radius of each sphere.

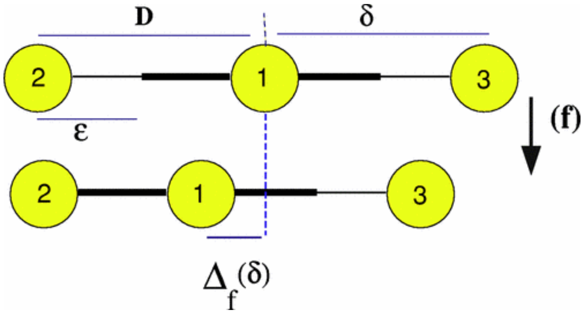


Figure 2.8 An auxiliary (fictitious) movement in which the right arm has a constant length δ while the left arm changes its length from D to $D-\varepsilon$. During this movement the middle sphere will be displaced by an amount $\Delta_f(\delta)$.
Source: [42].

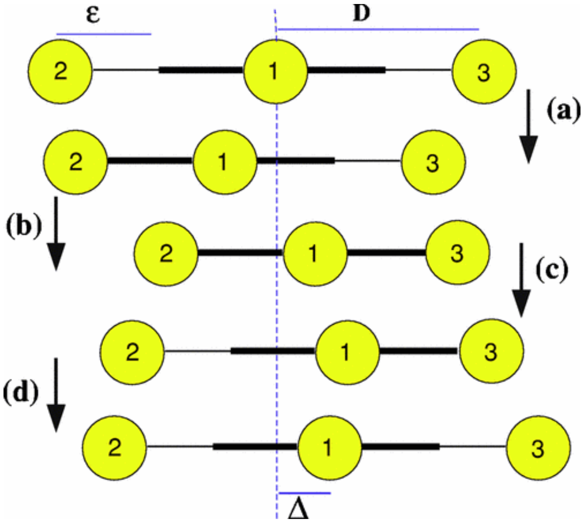


Figure 2.9 The four step, cyclic motion of the linear three sphere swimmer [42]. To analyze the motion of the system during one complete period of the non-reciprocal cycle, introduce the auxiliary stroke: during one stroke, the right arm has a constant length δ while the left arm changes it length from D to $D-\varepsilon$ with the constant velocity W. By symmetry, we can related all the four steps in the non-reciprocal cycle to the above stroke as follow: step(a): setting the length of right arm is $\delta = D$ while the left arm changes length from D to $D-\varepsilon$ with a constant velocity W; step(b): apply a reflection transformation with $\delta = D-\varepsilon$; step(c): apply a time-reversal transformation on the auxiliary stroke with the right arm is $\delta = D-\varepsilon$ and the left arm is D; step(d): apply a reflection transformation by a time-reversal transformation with $\delta = D$.

Consider the initial state of the system such that the spheres numbered 2 and 3 are in equal distance D from the middle sphere and divide a complete cycle of the nonreciprocal motion into four parts as above (see Figure 2.9).

1. In the first step of the motion, the right arm has fixed length, and the length of the left arm is decreased with a constant relative velocity W, using one of the internal engines in the middle sphere. Denote the relative displacement of the spheres 1 and 2 in this stage by ε.

2. For the second step, the left arm is fixed and the right arm decreases its length with the same constant relative velocity W as before. The relative displacement of the spheres 1 and 3 is again ε, like the previous stage.

3. During the third step, while the right arm is kept fixed, the left arm increases its length with the same relative velocity W to reach its original length D.

4. Finally, in the last step, the left arm is kept fixed and the right arm increases to its original length with the same constant velocity W. The system is now in its original configuration.

The result of this cyclic, time irreversible motion is a net translation of the swimmer along the x-axis; we define Δ as the distance the swimmer translates in one complete cycle. To obtain a net translational motion, the above cycle can be repeated continuously.

In the auxiliary stroke one arm has a fixed length, δ, where δ is either D or $D - \varepsilon$, and the other arm changes length from D to $D - \varepsilon$. We choose the x-axis to be parallel to the line linking the spheres and directed away from sphere 2 (see Figure 2.8). During the auxiliary stroke $v_1 = v_3$ and $W = v_2 - v_1$. To obtain the net displacement of the middle sphere in the real problem, it is thus enough to solve the dynamical equation for a single auxiliary movement. If we define the net displacement of the middle sphere during the auxiliary step by $\Delta_f(\delta)$, then by considering the above arguments we can calculate the total displacement Δ of the real system through a complete cycle as

$$\Delta = 2[\Delta_f(D) - \Delta_f(D - \varepsilon)] \tag{2.8}$$

2.3.2 Analytic Result

Velocity of middle sphere Since we are only interested in the dynamics of the spheres, we can equivalently solve the set of Equation (2.2) and Equation (2.4).

$$\begin{vmatrix} H_{11} & H_{12} & H_{13} \\ H_{12} & H_{11} & H_{23} \\ H_{13} & H_{23} & H_{11} \\ 1 & 1 & 1 \end{vmatrix} \begin{vmatrix} F_1 \\ F_2 \\ F_3 \end{vmatrix} = \begin{vmatrix} v_1 \\ v_1 + W \\ v_1 \\ 0 \end{vmatrix},$$

with $H_{11} = \frac{1}{6\pi\eta R}$, $H_{12} = \frac{1}{4\pi\eta(D-Wt)}$, $H_{13} = \frac{1}{4\pi\eta\delta}$, $H_{23} = \frac{1}{4\pi\eta(\delta+D-Wt)}$.

After Gaussian elimination, solve for v_1 and divide it by H_{11}:

$$v_1 = \frac{-W(H_{11} - H_{12} - H_{23} + \frac{H_{12}H_{13}+H_{13}H_{23}-H_{13}^2}{H_{11}})}{[3H_{11} - 2(H_{12} + 2H_{13} + 2H_{23}) - \frac{H_{12}^2+H_{13}^2+H_{23}^2-2H_{12}H_{13}-2H_{12}H_{23}-2H_{13}H_{23}}{H_{11}}]}, \quad (2.9)$$

collecting terms in $O(1), O(\xi), O(\xi^2)$:

$$\begin{cases} O(1) : -W(H_{11} - H_{12} - H_{23})(3H_{11} - 2(H_{12} + H_{13} + H_{23}) \\ O(\xi) : -W(H_{12} + H_{13} + H_{23})(3H_{11} - 2(H_{12} + H_{13} + H_{23}) \\ \qquad -W(H_{11} - H_{12} - H_{23})(H_{12} + H_{13} + H_{23} - 2H_{12} - 2H_{13} - 2H_{23}) \\ O(\xi^2) : -W(H_{12} + H_{13} + H_{23})(H_{12} + H_{13} + H_{23} - 2H_{12} - 2H_{13} - 2H_{23}) \end{cases}$$

Then, we drop the term in $O(\varepsilon)$ and higher to get v_1 in the form:

$$v_1 \sim \frac{-W(H_{11} - H_{12} - H_{23})}{3H_{11} - 2(H_{12} + H_{13} + H_{23})} \quad (2.10)$$

divide the above expression of v_1 by H_{11}:

$$v_1 \sim \frac{-W(H_{11} - H_{12} - H_{23})}{3H_{11} - 2(H_{12} + H_{13} + H_{23})} \xrightarrow{\text{divide by } H_{11}} \frac{-W}{3}[\frac{1 - \frac{H_{12}}{H_{11}} - \frac{H_{23}}{H_{11}}}{1 - \frac{2}{3}(\frac{H_{12}}{H_{11}} + \frac{H_{13}}{H_{11}} + \frac{H_{23}}{H_{11}})}],$$

with $\frac{H_{13}}{H_{11}} = \frac{3R}{2\delta}$, $\frac{H_{12}}{H_{11}} = \frac{3R}{2(D-Wt)}$, $\frac{H_{23}}{H_{11}} = \frac{3R}{2(\delta+D-Wt)}$.

Ignoring terms of order $(R/D)^2$ and greater the velocity of the middle sphere, in the limit that the swimmer undergoes small deformations, is

$$v_1(\delta) \sim -\frac{W}{3}[1 - \frac{R}{2(D-Wt)} + \frac{R}{\delta} - \frac{R}{2(\delta + D - Wt)}], \quad (2.11)$$

δ is the distance between sphere 1 and sphere 3, R is the radius of sphere, D is the distance between sphere 1 and sphere 2, W is the constant velocity of left arm(distance between sphere 2 and 1) to decrease, t is the time.

Displacement of the Middle Sphere Integrating v_1 over time t gives the displacement over the auxiliary stroke:

$$\Delta_a(\delta) = \int_0^{\varepsilon/W} v_1(\delta)dt,$$

$\Delta_a(\delta)$ is the displacement of the middle sphere, ε/W is the time for the stroke, $v_1(\delta)$ is the velocity of middle sphere.

Then the velocity for the middle sphere can be express as:

$$\begin{cases} v_1(\delta) & \sim -\frac{W}{3}[1 - \frac{R}{2(D-Wt)} + \frac{R}{\delta} - \frac{R}{2(\delta+D-Wt)}] \\ v_1(D) & \sim -\frac{W}{3}[1 - \frac{R}{2(D-Wt)} + \frac{R}{D} - \frac{R}{2(2D-Wt)}] \\ v_1(D-\varepsilon) & \sim -\frac{W}{3}[1 - \frac{R}{2(D-Wt)} + \frac{R}{D-\varepsilon} - \frac{R}{2(2D-Wt-\varepsilon)}] \end{cases}$$

Integrate Equation (2.11) gives the displacement over the auxiliary stroke and calculate the total displacement Δ after complete cycle based on Equation (2.8),

$$\Delta = -\frac{2W}{3}[[\frac{R}{D} - \frac{R}{D-\varepsilon}]\frac{\varepsilon}{W} + \frac{R}{2W}[-ln|2D-2\varepsilon| + 2ln|2D-\varepsilon| - ln|2D|]].$$

We can expand all the quantities in terms of ε/D,

$$\begin{cases} \frac{1}{1-\frac{\varepsilon}{D}} & = 1 + \frac{\varepsilon}{D} + (\frac{\varepsilon}{D})^2 + (\frac{\varepsilon}{D})^3 + O((\varepsilon/D)^4) \\ ln|2D(1-\frac{\varepsilon}{D})| & = ln2D - \frac{\varepsilon}{D} - \frac{1}{2}(\frac{\varepsilon}{D})^2 - \frac{1}{3}(\frac{\varepsilon}{D})^3 + h.o.t \\ ln|2D(1-\frac{1}{2}\frac{\varepsilon}{D})| & = ln2D - \frac{1}{2}\frac{\varepsilon}{D} - (\frac{1}{8}\frac{\varepsilon}{D})^2 - \frac{1}{24}(\frac{\varepsilon}{D})^3 + h.o.t \end{cases}$$

collecting terms in $O(1), O((\frac{\varepsilon}{D})), O((\frac{\varepsilon}{D})^2), O((\frac{\varepsilon}{D})^3)$:

$$\begin{cases} O(\frac{\varepsilon}{D}) & : 1 - 1 = 0 \\ O(\frac{\varepsilon}{D})^2 & : (-\frac{2}{3})R[-1 + \frac{1}{4} - \frac{1}{8}] = (-\frac{2}{3})R(-\frac{7}{8}) = \frac{7}{12}R \\ O(\frac{\varepsilon}{D})^3 & : (-\frac{2}{3})R[-1 + \frac{1}{2}(\frac{1}{3} - \frac{1}{12})] = (-\frac{2}{3})R(-\frac{7}{8}) = \frac{7}{12}R \end{cases}$$

Finally, the total displacement after the four step cycle, to second order in ε/D, as

$$\Delta = \frac{7}{12}R[(\frac{\varepsilon}{D})^2 + (\frac{\varepsilon}{D})^3]. \tag{2.12}$$

In the limit of small internal deformation of the linear three-sphere swimmer, the expression obtained by Najafi and Golestanian is as follows,

$$v_s = 0.7W(\frac{R}{D})(\frac{\varepsilon}{D})^2, \tag{2.13}$$

$$\Delta = 2.8R(\frac{\varepsilon}{D})^3. \tag{2.14}$$

This appears to give good agreement for larger values of ε/D. While the result is misleading, as in the limit of small ε/D it does not converge to the theoretical solution, as shown in the figure below, and it should not be valid at higher values of ε/D due to the assumptions made in the derivation. Figure 2.10 indicates the theoretical solution (blue dotted curve) converges to the simulation result better. If we consider the third stroke of the motion as shown in Figure 2.9, the swimmer must translate in the same direction, while, Equation (2.14) suggests that the swimmer propel in the reversed direction which is not correct.

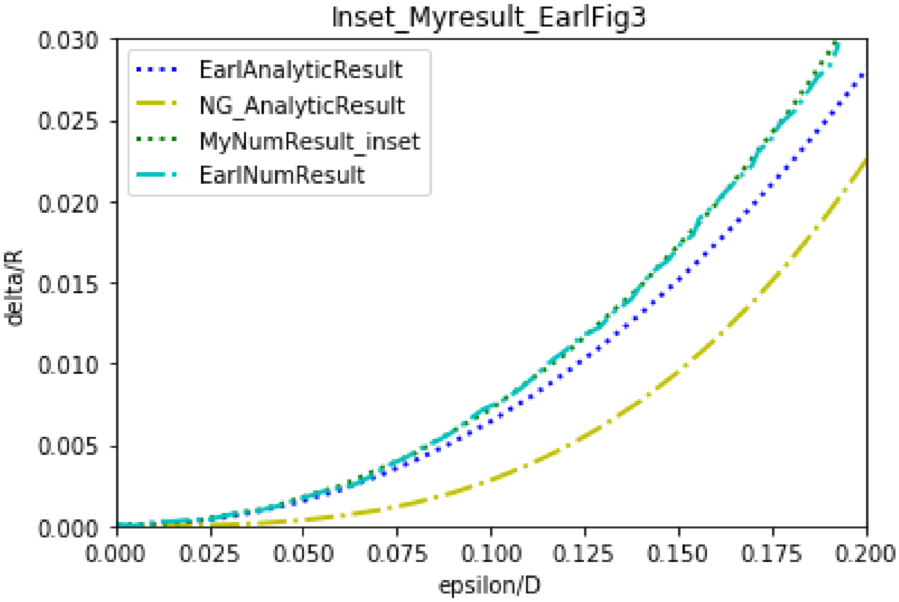


Figure 2.10 The net translation per cycle of the linear three-sphere swimmer, Δ as a function of the sphere displacement amplitude, ε. The region of x-axis is $\varepsilon/D = 0.2$. The parameters used are: $D = 25$ and $R = 3$. The blue dotted curve is the theoretical result presented by Earl *et al.*, the green dot-dashed curve is presented by Najafi and Golestanian, the cyan dotted curve is my numerical result and the cyan dot-dashed curve is Earl *et al.* simulation result.
Source: [27], [42]

2.3.3 Numerical Implementation and Results

Numerical implementation The Oseen tensor allows us to consider the hydrodynamic interaction, in the limit of zero Reynolds number, between spheres that are spaced far apart ($R \ll D$),

Set parameters:

1. D: distance between the central sphere and an outer sphere at the maximum arm length;

2. R: the radius of each sphere;

3. W: constant velocity for length change;

4. ε: sphere displacement amplitude $\varepsilon < D$.

Divide the interval(range) of ε into n subintervals, for each ε_i, $\varepsilon_i = \frac{\varepsilon}{n}i$, follow the four steps as shown in Figure 2.9.

1. Step a:

   (a) use the parameters above to compute the stokeslet $H_{ii} = \frac{1}{6\pi\eta R_i}$, $H_{ij} = \frac{1}{4\pi\eta r_{ij}}$, with $r_{12} = D - \varepsilon_i$, $r_{13} = D$, $r_{23} = 2D - \varepsilon_i$;

   (b) plug into equation $v_i(\varepsilon_i)_a = \sum H_{ij}F_j$ and the force-free condition, solve for $v_i(\varepsilon_i)_a$;

   (c) for small ε, use equation $\Delta_a \sim \sum_{i=1}^{n} v_i(\varepsilon_i)_a \frac{\varepsilon_i}{w}$ to solve for the displacement after step a.

2. Step b:

   (a) fix the distance $r_{12} = D - \varepsilon$, update the stokeslet $H_{ii} = \frac{1}{6\pi\eta R_i}$, $H_{ij} = \frac{1}{4\pi\eta r_{ij}}$, with $r_{12} = D - \varepsilon$, $r_{13} = D - \varepsilon_i$, $r_{23} = 2D - \varepsilon - \varepsilon_i$;

   (b) plug into equation $v_i(\varepsilon_i)_b = \sum H_{ij}F_j$ and the force-free condition, solve for $v_i(\varepsilon_i)_b$;

   (c) for small ε, use equation $\Delta_b \sim \sum_{i=1}^{n} v_i(\varepsilon_i)_b \frac{\varepsilon_i}{w}$ to solve for the displacement after step b.

3. Step c:

   (a) fix the distance $r_{13} = D - \varepsilon$, update the stokeslet $H_{ii} = \frac{1}{6\pi\eta R_i}$, $H_{ij} = \frac{1}{4\pi\eta r_{ij}}$, with $r_{12} = D - \varepsilon + \varepsilon_i$, $r_{13} = D - \varepsilon$, $r_{23} = 2D - 2\varepsilon + \varepsilon_i$;

   (b) plug into equation $v_i(\varepsilon_i)_c = \sum H_{ij}F_j$ and the force-free condition, solve for $v_i(\varepsilon_i)_c$;

   (c) for small ε, use equation $\Delta_c \sim \sum_{i=1}^{n} v_i(\varepsilon_i)_c \frac{\varepsilon_i}{w}$ to solve for the displacement after step c.

4. Step d:

   (a) fix the distance $r_{12} = D$, update the stokeslet $H_{ii} = \frac{1}{6\pi\eta R_i}$, $H_{ij} = \frac{1}{4\pi\eta r_{ij}}$, with $r_{12} = D$, $r_{13} = D - \varepsilon + \varepsilon_i$, $r_{23} = 2D - \varepsilon + \varepsilon_i$;

   (b) plug into equation $v_i(\varepsilon_i)_d = \sum H_{ij}F_j$ and the force-free condition, solve for $v_i(\varepsilon_i)_d$;

   (c) for small ε, use equation $\Delta_d \sim \sum_{i=1}^{n} v_i(\varepsilon_i)_d \frac{\varepsilon_i}{w}$ to solve for the displacement after step d.

5. compute the total displacement after a complete cycle:

$$\Delta = \Delta_a + \Delta_b + \Delta_c + \Delta_d.$$

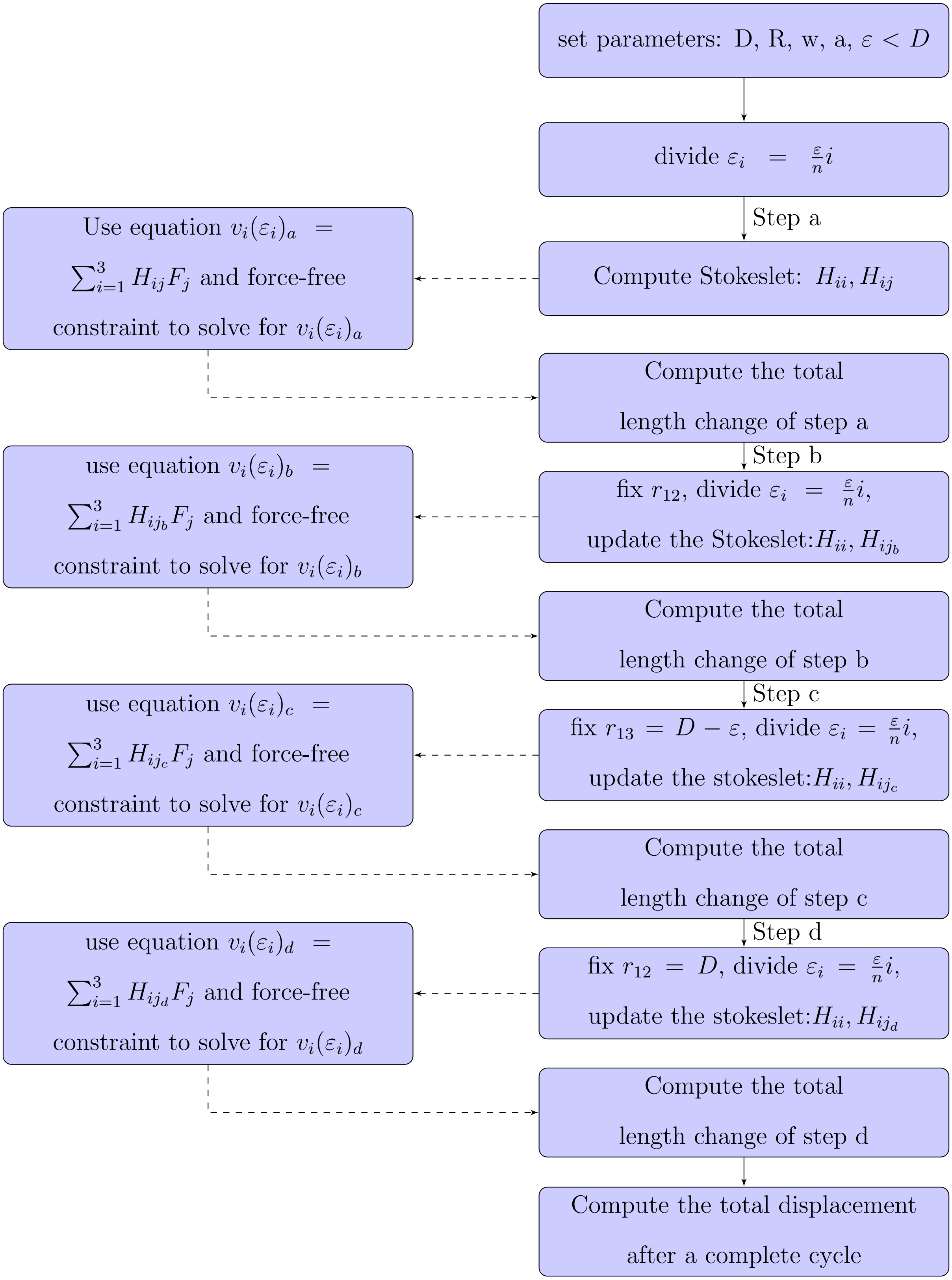

set parameters: D, R, w, a, $\varepsilon < D$
divide $\varepsilon_i = \frac{\varepsilon}{n} i$
Step a
Compute Stokeslet: H_{ii}, H_{ij}
Use equation $v_i(\varepsilon_i)_a = \sum_{i=1}^{3} H_{ij} F_j$ and force-free constraint to solve for $v_i(\varepsilon_i)_a$
Compute the total length change of step a
Step b
fix r_{12}, divide $\varepsilon_i = \frac{\varepsilon}{n} i$, update the Stokeslet:$H_{ii}, H_{ij_b}$
use equation $v_i(\varepsilon_i)_b = \sum_{i=1}^{3} H_{ij_b} F_j$ and force-free constraint to solve for $v_i(\varepsilon_i)_b$
Compute the total length change of step b
Step c
fix $r_{13} = D - \varepsilon$, divide $\varepsilon_i = \frac{\varepsilon}{n} i$, update the stokeslet:$H_{ii}, H_{ij_c}$
use equation $v_i(\varepsilon_i)_c = \sum_{i=1}^{3} H_{ij_c} F_j$ and force-free constraint to solve for $v_i(\varepsilon_i)_c$
Compute the total length change of step c
Step d
fix $r_{12} = D$, divide $\varepsilon_i = \frac{\varepsilon}{n} i$, update the stokeslet:$H_{ii}, H_{ij_d}$
use equation $v_i(\varepsilon_i)_d = \sum_{i=1}^{3} H_{ij_d} F_j$ and force-free constraint to solve for $v_i(\varepsilon_i)_d$
Compute the total length change of step d
Compute the total displacement after a complete cycle

Figure 2.11 Flowchart for linear three-sphere numerical scheme.

Numerical results Najafi and Golestanian introduced the linear three-sphere swimmer and calculated its swimming velocity by solving the linear governing equations. The swimmer uses the periodic internal motion to propel itself under low Reynolds number environment. The advantage of this model, as compared to previously known model swimmers, is that the analysis of the hydrodynamics problem can be performed easily.

Figure 2.12 gives the results for a single linear three-sphere swimmer. The graph shows how the total displacement of the swimmer over one cycle, Δ, varies as a function of the amplitude of the stroke ε. As we increase the amplitude of the stroke ε, the net translational displacement will also increase over one cycle. Our numerical result is consistent with Najafi and Golestanian's result.

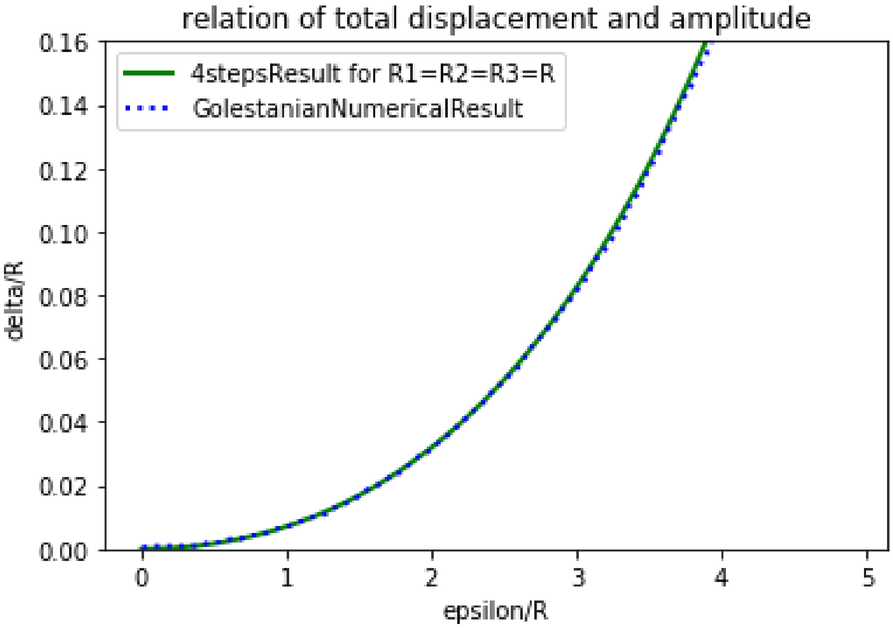


Figure 2.12 Dimensionless displacement of the swimmer in a complete cycle as a function of the dimensionless relative displacement between neighboring spheres. The solid green curve is obtained by solving the Oseen tensor interaction between the spheres, the blue dotted line shows the simulation result from Najafi and Golestanian, the parameters used were $D = 10R$.

Figure 2.13 shows how the total displacement of the swimmer over one cycle, Δ, varies as a function of the amplitude of the stroke ε. The parameters used were D=25 and R=3 for the Oseen tensor. The green dotted line was obtained by numerically solving the Oseen tensor equation and the cyan dash-dotted line is the simulation result obtained by Earl *et al.* [27]. The graph shows our numerical result is consistent with Earl *et al.* simulation result.

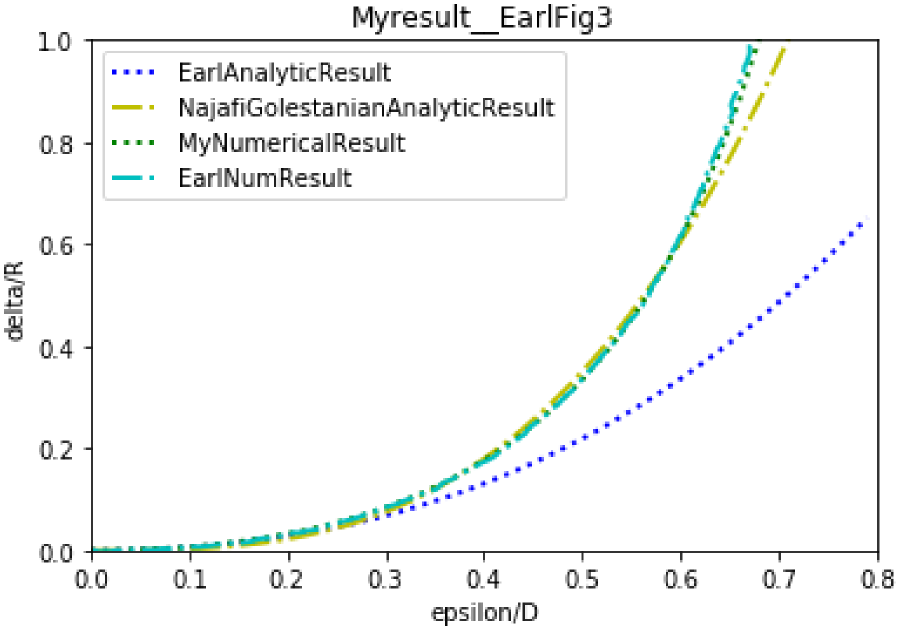


Figure 2.13 The shift per cycle of the linear three sphere swimmer, Δ, as a function of the sphere displacement amplitude, ε. The parameters used were D=25 and R=3 for the Oseen tensor. The blue dotted line is the theoretical expression given in Equation (2.8), the olive dash-dotted line is the expression proposed by Golestanian, the green dotted line was obtained by numerically solving the Oseen tensor equation and the cyan dash-dotted line is the simulation result obtained by Earl *et al.*. *Source: [27], [42].*

2.3.4 Results for Self-learning Linear Three-sphere Swimmer

The swimmer struggles to find a policy to propel in the positive direction at the beginning, therefore it will moves back and forth for the first 60 training steps, with D remains close to 0. Then the swimmer keeps exploring the surrounding environment by taking different actions and updating its propulsion policy. After accumulating enough experiences, the linear three-sphere swimmer develops an effective propulsion policy that repeats the same sequence of actions, except when a random action is chosen, and propel in the positive direction with increasing D. The propulsion policy obtained by Q-learning algorithm for a three-sphere swimmer is consistent with Najafi-Golestanian's swimming gaits, which indicates Q-learning can train a swimmer to obtain its propulsion gaits without prior knowledge of low Reynolds number locomotion.

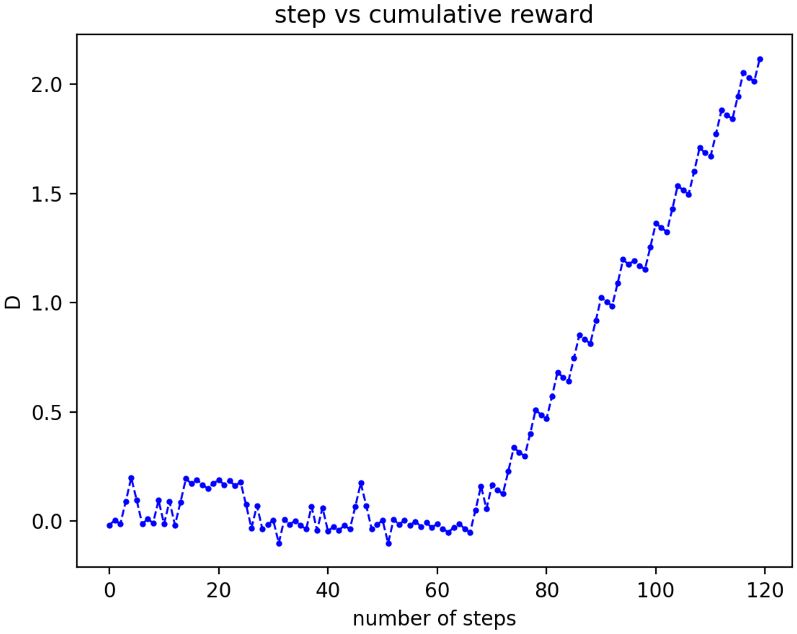


Figure 2.14 A typical learning process of a self-learning swimmer, the dimensionless cumulative displacement D of the swimmer evolves over learning steps. The x-axis is the number of learning step, the y-axis is the cumulative displacement of the body centroid.

Figure 2.15 shows ten learning processes of a self-learning three-sphere swimmer, the dimensionless cumulative displacement D of the swimmer evolves over learning steps. The swimmer initially struggles to find a policy to propel in the positive direction after sufficient steps (*e.g.*, learning step around 60), then the swimmer develops an effective propulsion policy and moves in the positive direction with increasing D.

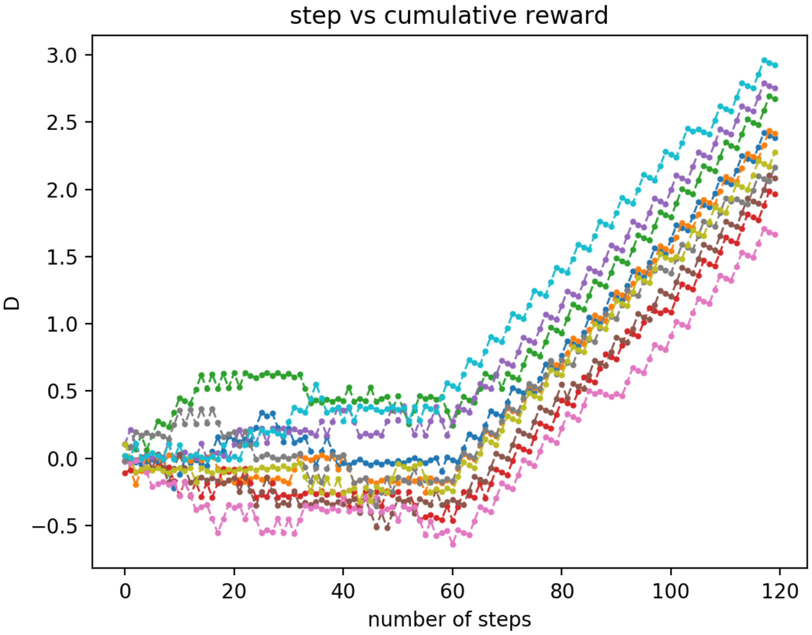


Figure 2.15 Ten typical learning processes of a self-learning three-sphere swimmer with the x-axis is the learning step and the y-axis is the cumulative displacement of the body centroid.

Figure 2.16 indicates as the learning step greater than 120, the difference of each row in the Q-matrix becomes steady.

Figure 2.17 illustrates the effective propulsion policy for the linear three-sphere swimmer is in the "travelling wave" pattern which is consistent with Najafi-Golestanian's swimmer [42].

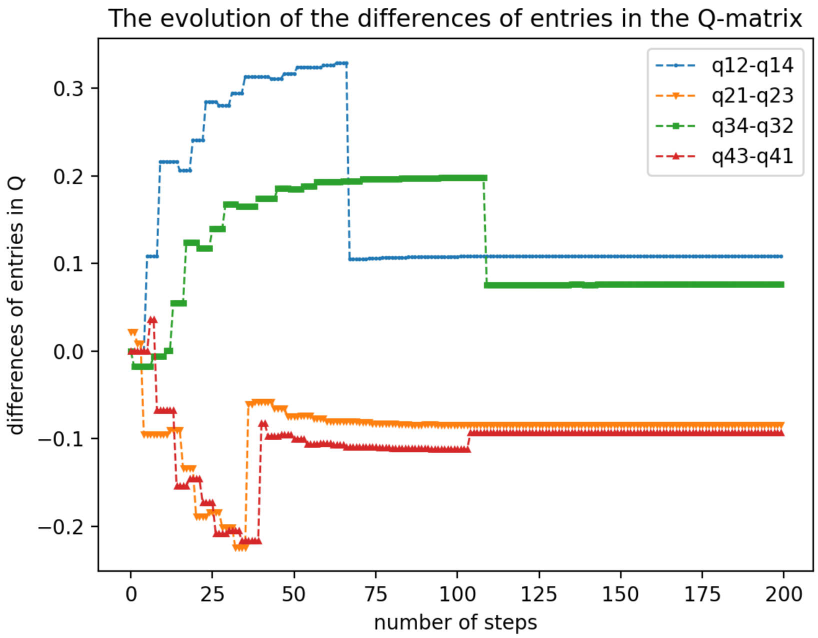


Figure 2.16 Evolution of the differences of entries in the Q-matrix. As the learning steps increases, the Q-matrix becomes steady.

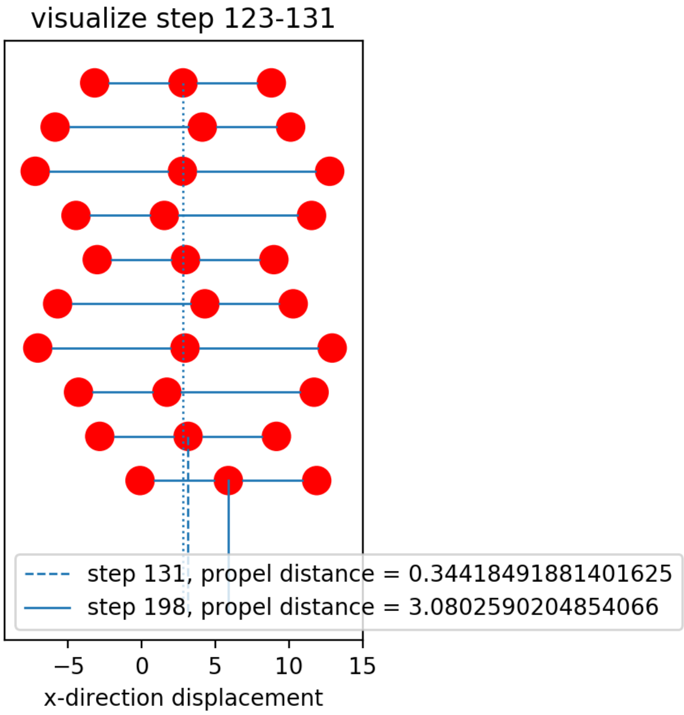


Figure 2.17 Configurations of the linear three-sphere swimmer from learning step 123 to 131, the net translation is around 0.34. The bottom state is learning step 198, the cumulative propel distance from step 123 to step 198 is approximately 3.08.

CHAPTER 3

MECHANICAL ROTATION VIA REINFORCEMENT LEARNING

3.1 Background and Related Work

Swimming microorganisms inhabit a world dominated by the viscous force. The Reynolds number, $\mathrm{Re} = \rho U \ell / \mu$ (where ℓ and U represent the characteristic length and speed of the swimmer, and ρ and μ are fluid density and dynamic viscosity, respectively), falls in the range of 10^{-4} to 10^{-2} for swimming bacteria and spermatozoa [31, 60, 92]. The inertial force is therefore negligible compared with the viscous force. At such low Reynolds numbers, common swimming strategies based on inertia at the macroscopic scales become largely ineffective [62, 109]. Microorganisms have evolved different strategies, including the use of flagellar rotary motors [11] or the action of molecular motors within flagella [84], to swim effectively in their microscopic world. There are growing interests in developing artificial microscopic machines that can self-propel like their biological counterparts for potential biomedical and environmental applications [34, 74]. However, without sophisticated biological molecular machines possessed by microorganisms, it remains a challenge to design micromachines for complex maneuvers in the viscously dominated flow limit [28].

Purcell's work popularized the fundamental fluid dynamical aspects of swimming at low Reynolds numbers [79]. In particular, his scallop theorem rules out any reciprocal motion–sequence of motions with time reversal symmetry (*e.g.*, opening and closing the hinge of a single-hinged scallop) for self-propulsion without inertia. To escape from the constraints by the scallop theorem, Purcell designed a three-link swimmer that can perform kinematically irreversible cyclic motions for net translation [10, 79]. Najafi and Golestanian [71] proposed another ingenious design consisting of three linked spheres, which can translate by modulating the distances between the spheres; the mechanism inspired a wide variety of variants [6–8, 27, 41, 72, 91, 103, 104]. In addition to net translation, the design of mechanisms that can produce net rotation at the microscale is important to the development of micromachines. To this end, Dreyfus *et al.* proposed a mechanism (also known as Purcell's "rotator") [25], which consists of three spheres linked like the spokes on a wheel (Figure 3.1), as the rotational analogue of Purcell's three-link swimmer for translation. The rotator performs a prescribed sequence of motions that exploit the hydrodynamic interaction between the spheres to produce net rotation. The mechanism of Purcell's rotator shares similarity with the conformational changes of some molecular motors undergoing ATP- or photochemically-driven rotational movements [25, 57, 58].

These ingenious designs rely on knowledge of the surrounding environment and the physics of locomotion within the environment, which may not be complete or clear in more complex scenarios. In particular, for biological applications, the properties of some highly complex, heterogeneous biological environments may not be known *a priori*, posing additional challenges on the design of effective self-propelled micromachines. Recent approaches have exploited the prowess of machine learning in the studies of different aspects of locomotion in fluids [17, 97], including individual and collective motion of fish [13, 37, 38, 54, 75, 100, 107] and birds [81, 82], as well as different navigation [2, 70, 80, 108] and cloaking [68] problems of self-propelled objects. In particular, an alternative framework based on reinforcement learning has enabled a microswimmer to learn effective locomotory gaits based on its interactions with the surrounding low-Reynolds-number environment [99]. Without any prior knowledge of locomotion, such a "self-learning" microswimmer is able to acquire a previously known propulsion strategy by Najafi and Golestanian [71] for net translation and adapt its locomotory gaits in different media.

Similar in spirit, in this work we employ a reinforcement learning approach to generate mechanical rotation at low Reynolds numbers. We adopt the mechanical configuration of the Purcell's rotator shown in Figure 3.1 [25]; however, instead of prescribing the locomotory gaits of Purcell's rotator, we allow the machine to progressively learn how to exploit hydrodynamic interactions to produce net rotation via reinforcement learning on its own. We will examine the locomotion strategies acquired by the learning process and consider more complex scenarios when the number of spheres in the machine increases. This work is organized as follows: in Section 3.2 we present the geometric setup (Section 3.2.1), formulation of the hydrodynamic (Section 3.2.2) and the reinforcement learning (Section 3.2.3) problems used in this work. We discuss the results in Section 3.3 for a three-sphere rotator (Section 3.3.1), before extending the studies to configurations with a higher number of spheres (Section 3.3.2). We conclude the investigation with some remarks in Section 3.4.

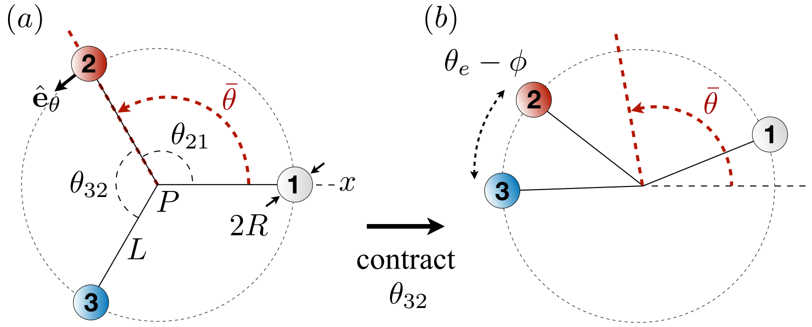


Figure 3.1 Schematic diagram and notations of a mechanical setup based on Purcell's rotator by Dreyfus *et al.* [25]. The machine consists of three spheres of radius R connected to the center P with connecting rods of length L. The spheres are connected to the center of the circle P with connecting rods. (a) In its initial configuration, the three spheres have equal angular spacing, $\theta_e = 2\pi/3$. There exist active elements that can contract the angle θ_{21} or θ_{32} by an amount ϕ or expand by the same amount to return to the value θ_e. In (b), we illustrate the configuration of the machine after it contracts the angle θ_{32}, which results an overall change of the angular centroid of the machine, $\bar{\theta}$ (indicated by the red dashed lines).

3.2 Formulation

3.2.1 Geometric Setup

We first illustrate the geometric setup using a three-sphere configuration similar to Purcell's rotator (Figure 3.1a), before considering systems with an increased number of spheres. We place three spheres of radius R on an imaginary circle of radius L. The spheres are individually connected to the center of the circle $\mathbf{P}$ with connecting rods. Figure 3.1 shows an initial configuration with equal angular spacing ($\theta_e = 2\pi/3$) between the spheres, where the angle between spheres 2 and 1 (θ_{21}) and the angle between spheres 3 and 2 (θ_{32}) attain their fully extended values ($\theta_{21} = \theta_{32} = \theta_e$). There exist two internal active elements that can contract θ_{21} or θ_{32} (referred to as active angles here) by an amount ϕ (Figure 3.1b), or expand an angle back to its fully extended value θ_e. The remaining angle between spheres 3 and 1 (θ_{13}) only reacts passively to the contraction and expansion. To measure the net rotation of the machine, we define the angular centroid $\bar{\theta} = \sum_1^3 \theta_i/3$, which is the average of the angles of all spheres θ_i measured from the x-axis. The angular centroid of the initial configuration shown in Figure 3.1a is given by $\bar{\theta} = 2\pi/3$, as indicated by the red dashed line. Actuating (contracting or expanding) any of the active angles will alter the angular centroid of the machine as illustrated in Figure 3.1b. The goal of the machine is to generate net rotation (*i.e.*, a net increase in the angular centroid $\bar{\theta}$) in the anti-clockwise direction by choosing different actions of the active elements. Without requiring prior knowledge of low-Reynolds-number locomotion, we will demonstrate a reinforcement learning approach in achieving this goal. We next present the formulation of the hydrodynamic problem in Section 3.2.2 and its integration with a reinforcement learning algorithm in Section 3.2.3.

3.2.2 Low-Reynolds-number Hydrodynamics

We consider the hydrodynamics governed by the Stokes equation in the low Reynolds number regime. Here we neglect the hydrodynamic influence of the connecting rods and account for the leading-order hydrodynamic interaction between the spheres via the Oseen tensor [25,44,71] in the limit $R/L \ll 1$. The forces $\mathbf{F}_i$ and velocities $\mathbf{V}_i$ of the spheres ($i = 1, 2, 3$) are related as

$$\mathbf{F}_i = \sum_{j=1}^{3} \mathbf{H}_{ij} \mathbf{V}_j, \tag{3.1}$$

where

$$\mathbf{H}_{ij} = \begin{cases} -6\pi\mu R\,\mathbf{I}, & \text{if } \ i = j \\ 6\pi\mu R \frac{3R}{4R_{ij}} (\mathbf{I} + \hat{\mathbf{R}}_{ij}\hat{\mathbf{R}}_{ij}), & \text{if } \ i \neq j \end{cases} \tag{3.2}$$

and $R_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$, $\mathbf{r}_i$ is the position of sphere i from the center P, $\hat{\mathbf{R}}_{ij} = (\mathbf{r}_i - \mathbf{r}_j)/R_{ij}$, and $\mathbf{I}$ is the identity matrix. The hydrodynamic torque about the origin in the laboratory frame is given by $\mathbf{\Gamma}_i = \mathbf{D}_i \times \mathbf{F}_i = \mathbf{D}_i \times \sum_{j=1}^{3} \mathbf{H}_{ij} \mathbf{V_j}$, where $\mathbf{D}_i$ is the position vector of each spheres in the laboratory frame. Here we focus on pure rotation of the machine and thus fix its center P to the origin in the laboratory frame. If the center is not kept fixed, the machine can undergo both translation and rotation [25]. The velocity of the spheres $\mathbf{V}_i = L\dot{\theta}_i \ \hat{\mathbf{e}}_\theta$ are therefore purely tangential to the imaginary circle, where $\hat{\mathbf{e}}_\theta$ is the unit vector tangent to the circle. In the absence of an external torque, the system is torque-free

$$\sum_{i=1}^{3} \mathbf{\Gamma}_i = \mathbf{0}. \tag{3.3}$$

The machine is allowed to actuate any one of the active elements in each step to contract or expand the angle at a rate ω. For instance, in Figure 3.1 from (a) to (b), the machine contracts the angle θ_{32} by an amount ϕ (*i.e.*, $\dot{\theta}_3 - \dot{\theta}_2 = -\omega$), while maintaining the angle θ_{21} fixed (*i.e.*, $\dot{\theta}_2 = \dot{\theta}_1$). Such action results an overall change of the angular centroid of the machine, $\bar{\theta}$ (indicated by the red dashed lines in Figure 3.1). These kinematic constraints close the system of equations, which can be numerically solved to determine the rotational dynamics of the machine for each action taken. We remark that the linearity and time-independence of the Stokes equation leads to the property of rate independence [62, 79]: any translational or rotational displacement of the machine resulting from its configuratonal changes (contraction/expansion of active angles) does not depend on the rate of configurational changes but only on the sequence of the changes. We therefore follow Dreyfus *et al.* [25] and assume a uniform rate of expansion and contraction ω in this work.

Allowing the rotator to both translate and rotate, we need to satisfy both force and torque free condition.

$$\sum_{i=1}^{3} \boldsymbol{F}_i = \mathbf{0}, \quad \sum_{i=1}^{3} \mathbf{\Gamma}_i = \mathbf{0}. \tag{3.4}$$

To solve those system of equations, we parameterize the velocities of each sphere by separating its translational and angular velocities in the laboratory frame:

$$\mathbf{V}_i = \mathbf{V}_P + L\dot{\theta}_i \mathbf{U}_{\theta_i}, \tag{3.5}$$

where $\mathbf{V}_p$ is the translational velocity of the circle's center $\mathbf{P}$, $\dot{\theta}_i$ is the rotational velocity of each sphere, and $\mathbf{U}_{\theta_i}$ is the rotational velocity direction which is always tangent to the circle.

Two more equations are needed to fully constraint the system. Those equations are determined based on which element is active. The stroke shown in Figure 3.1(b) is a contraction in θ_{23}. We can write the following:

$$\dot{\theta}_3 - \dot{\theta}_2 = -\omega \tag{3.6}$$

$$\dot{\theta}_2 = \dot{\theta}_1 \tag{3.7}$$

and ω is the constant speed of contraction, where $\phi = \Delta t \omega$.

With the above construction, we allow both translation and rotation of our rotator. However, as shown in by Dreyfus *et al.* [25], fixing $\mathbf{P}$ generates more net rotation. We will therefore limit our reinforcement study on the no translation case only. To fix $\mathbf{P}$, we need to introduce an equal and opposite force at $\mathbf{P}$, which indicates the force free condition is no longer valid. Furthermore, we can set $\mathbf{D}_i = \mathbf{r}_i$,$\mathbf{V}_p = 0$.

3.2.3 Reinforcement Learning

The goal of the machine is to generate net rotation by performing an effective sequence of strokes. Instead of prescribing the sequence of strokes in the conventional approach, here we use a simple reinforcement learning algorithm to enable the machine to acquire effective locomotion strategies by itself. Such an approach does not rely on prior knowledge of locomotion but allows the machine to learn and adapt its locomotion strategies based on its experience interacting with the surroundings. Here we implement the Q-learning algorithm for its simplicity and expressiveness compared with other reinforcement learning algorithms [105].

Many model-free reinforcement learning algorithms have show their capabilities in producing sub-optimal policies to mathematically complex and intractable problems. For the rotator shown in Figure 3.1, We can mathematically trace out the effective rotating policy for the model. However,the problem will quickly become intractable when more choices are introduced with increasing spheres. Hence, we will employ reinforcement learning. For all the results follow, we will implement Q-learning algorithm due to its simplicity and expressiveness comparing to other potential algorithms.

We first introduce four Q-learning concepts: states, actions, rewards, and Q-matrix. States (s_n) is a set containing all possible position of the model, where n represents the number of steps. For our problem, we consider states as a set of all geometric configurations. (A total of four combinations of contraction and extension in θ_{12} and θ_{23}.) Actions (a_n) are all choices the model can make given a state, s_n. When making an action a_n, the rotator will move to the next state, s_{n+1}. Two sets of actions are available. The rotator can either choose to extend/contract θ_{12} or θ_{23}. Rewards (r_n) directly relate to action the rotator takes given a state. We define rewards as the angular centroid difference after taking action, a_n, at state, s_n. $(\bar{\theta}_{n+1} - \bar{\theta}_n)$.

In a given learning step in Q-learning (for example, the n-th step in Figure 3.2), the machine performs an action (a_n, contracting or expanding one of the active angles), taking the machine from the current configuration state (s_n) to the next state (s_{n+1}). The "success" of action a_n is measured by reward r_n, which is defined as the resulting difference of the angular centroid ($i.e.$, $r_n = \bar{\theta}_{n+1} - \bar{\theta}_n$). The expected long-term reward for taking the action a_n given the state s_n is quantified by the Q-matrix, $Q(s_n, a_n)$, which is an action-value function that encodes the adaptive decision-making intelligence of the machine. After each learning step, the Q-matrix evolves based on the experience gained by the machine,

$$\begin{aligned} Q(s_n, a_n) \leftarrow & Q(s_n, a_n) + \\ & \alpha[r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)], \end{aligned} \tag{3.8}$$

where α is the learning rate ($0 \leq \alpha \leq 1$) that determines to what extent new information overrides old information and therefore control the learning speed of the machine. Here we fixed $\alpha = 1$ to maximize the learning speed. The discount factor γ ($0 < \gamma < 1$) determines the trade-off between immediate reward r_n and maximum future reward at the next state $\max_{a_{n+1}} Q(s_{n+1}, a_{n+1})$. When γ is small, the machine is shortsighted and tends to maximize the immediate reward; when γ is large, the swimmer is farsighted and takes actions that maximize the long-term reward. In order to avoid the machine from being trapped in locally optimal policies, we implemented an ϵ-greedy selection scheme: In each learning step, the machine chooses the best action recommended by the Q-matrix with a probability $1 - \epsilon$ or takes a random action with a small probability ϵ, which allows the machine to explore new solutions.

We fixed $\alpha = 1$ to maximize learning. γ is the discount factor $(0 \leq \gamma \leq 1)$ determines how far-slightness of the rotator. With large γ, the rotator will more likely to take actions maximizing long term reward (net angular rotation). In addition, we will also include an $\epsilon-$greedy scheme: in each step, the rotator has an ϵ probability to take a random action, and $1 - \epsilon$ probability to take the best action advise by the current Q-matrix, which allows the rotator to explore new choices and avoids being trapped in sub-optimal policies. For all runs, we will use $\alpha = 1.0$, $\gamma = 0.9$, and $\epsilon = 0.15$ unless otherwise specified.

As a remark, the configuration states considered here correspond to the shape space in the literature, which contains all possible shapes of the machine without considering the positions and orientations of the rotator.

The goal of the machine is to generate net rotation by performing different configurational changes. Instead of designing a sequence of locomotory gaits in advance, here we leverage a simple reinforcement learning algorithm (Q-learning) to enable the machine to acquire effective locomotion strategy based on its interaction with the surroundings. In each learning step, the machine performs an action a_n (contracting or expanding on the active angles) to transform from one configuration state s_n to the next s_{n+1}. The reward r_n, defined as the resulting difference of the angular centroid $(\bar{\theta}_{n+1} - \bar{\theta}_n)$, measures the success of each action. The reinforcement learning process progressively updates the Q-matrix, which encodes the adaptive decision-making intelligence of the machine [105].

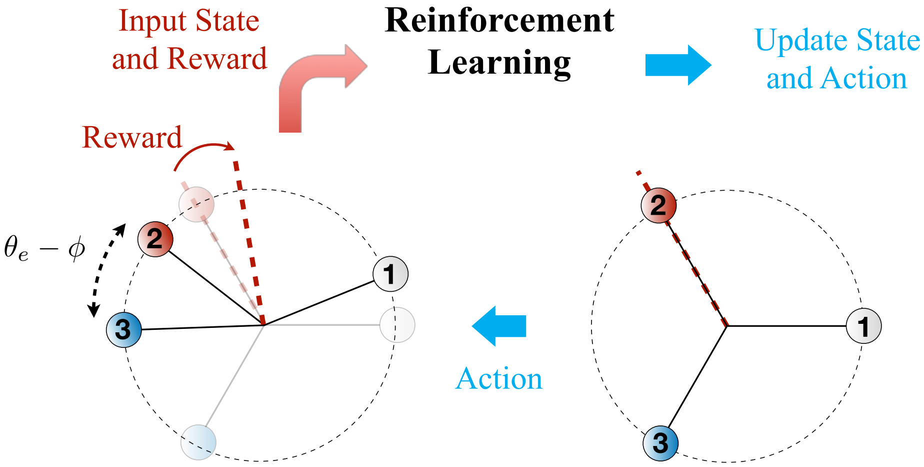


Figure 3.2 Mechanical rotation at low Reynolds numbers via reinforcement learning. The goal of the machine is to generate net rotation by performing different configurational changes. Instead of designing a sequence of locomotory gaits in advance, here we leverage Q-learning algorithm to enable the machine to acquire effective locomotion strategy based on its interaction with the surroundings. In each learning step, the machine performs an action a_n (contracting or expanding on the active angles) to transform from one configuration state s_n to the next s_{n+1}. The reward r_n, defined as the resulting difference of the angular centroid $(\bar{\theta}_{n+1} - \bar{\theta}_n)$, measures the success of each action. The reinforcement learning process progressively updates the Q-matrix, which encodes the adaptive decision-making intelligence of the machine.
Source: [105].

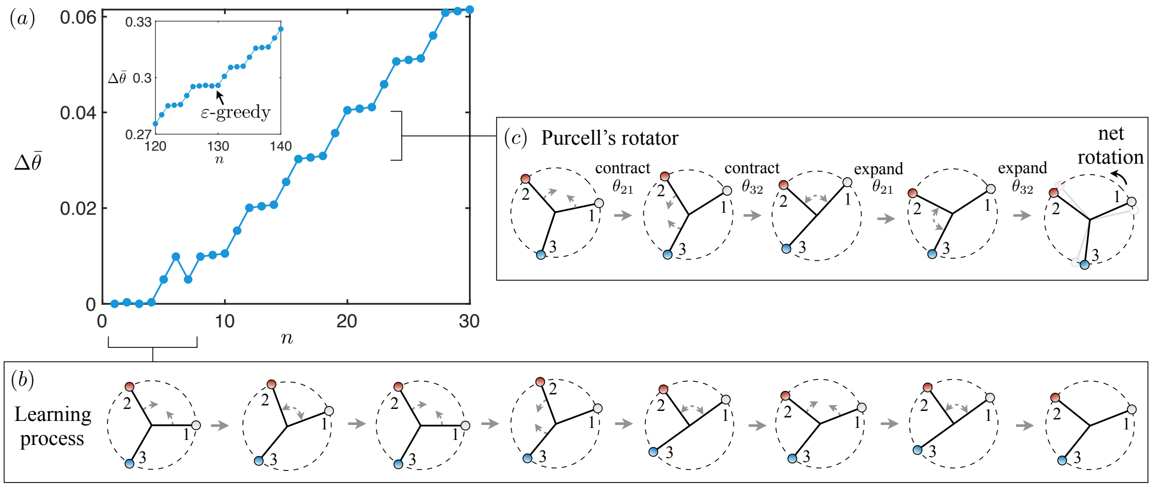


Figure 3.3 Reinforcement learning of a three-sphere ($N = 3$) rotator. (a) The net rotation of the machine, measured by the change of angular centroid, denoted as $\Delta\bar{\theta}$, generated by a series of actions at different learning steps n. (b) The rotator undergoes an initial learning stage by performing different actions to interact with the surrounding environment and learn from the resulting rewards. (c) Via reinforcement learning, the machine eventually repeats a sequence of cyclic motions that produce net rotation in the anti-clockwise direction. The strategy acquired through reinforcement learning here coincides with that used for Purcell's rotator by Dreyfus *et al.* [25]. Inset in (a): the ε-greedy scheme allows a small probability ε for the machine to act against the Q-matrix and perform a random action for exploration. Here we set $\phi = \pi/6$, $\gamma = 0.9$, $\epsilon = 0.05$, and $R/L = 0.1$. The rigid body rotation illustrated in panels (b)–(c) are magnified by twenty times for better visualization of the rotational motion.

3.3 Results and Discussion

3.3.1 3-sphere Rotator

We first consider a three-sphere configuration in this section. Instead of prescribing any sequence of strokes, we allow the rotator to take an action based on the Q-matrix (Section 3.2.3) and use the resulting reward to update the Q-matrix, informing the next action. We measure the net rotation of the machine $\Delta\bar{\theta} = \bar{\theta}_n - \bar{\theta}_0$ by comparing the angular centroid at the n-th learning step ($\bar{\theta}_n$) with the initial angular centroid ($\bar{\theta}_0$). Figure 3.3(a) shows a typical learning process of a 3-sphere rotator: the rotator takes the initial steps to explore the viscous environment (Figure 3.3(b)) without forming an effective rotational strategy yet. As the machine learns from its interaction with the environment progressively, it eventually repeats the same sequence of cyclic motions that produce net rotation in the anti-clockwise direction (Figure 3.3(c)). We note that the policy harvested by reinforcement learning here coincides with the mechanism proposed by Dreyfus *et al.* for Purcell's rotator [25]. As the analogue of the self-learning swimmer that produces net translation [99], our example here demonstrates the first use of reinforcement learning to generate net mechanical rotation in a low-Reynolds-number environment, without requiring prior knowledge of locomotion.

As a remark, even when the machine is informed by the Q-matrix to repeat the same sequence of strokes after sufficient learning steps (Figure 3.3a inset), the use of the ε-greedy selection scheme allows a small but non-zero probability ϵ for the machine to act against the Q-matrix and perform a random action for exploration. The sequence of strokes is therefore sometimes interrupted with random actions as shown in the inset. Such mechanism avoids being trapped around only locally optimal policies. For the 3-sphere configuration, the machine eventually returns to the Purcell's rotator sequence after the random actions. We will examine the effect of the magnitude of ε with more complex examples in the next section.

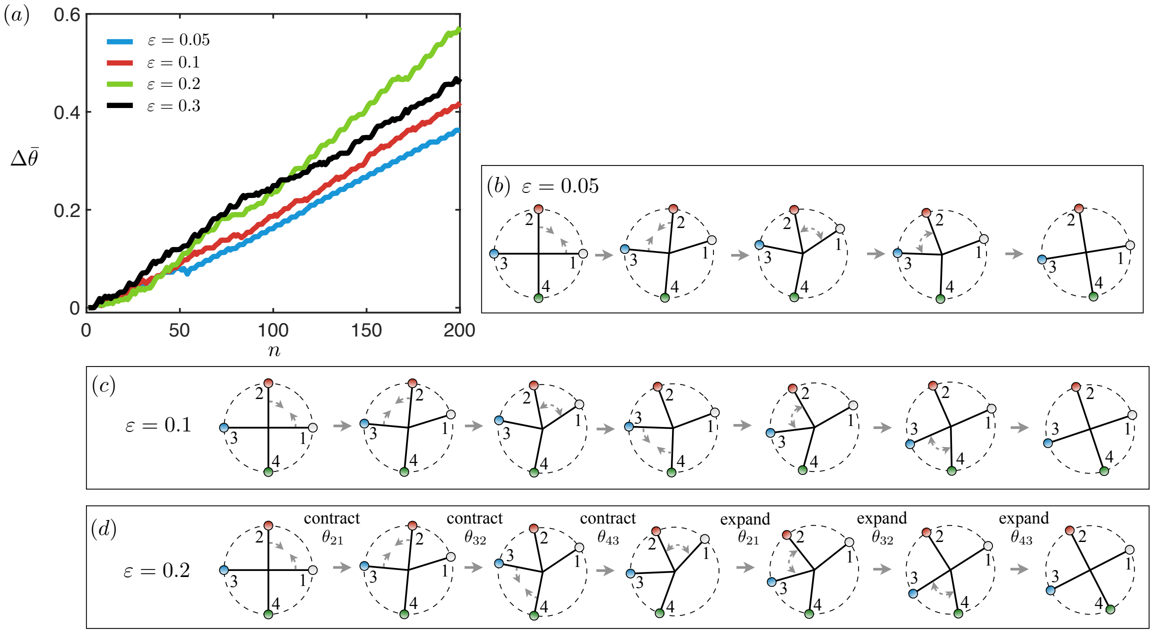


Figure 3.4 Reinforcement learning of a four-sphere ($N = 4$) rotator. (a) The net rotation of the machine, measured by the change of angular centroid $\Delta\bar{\theta}$, generated by a series of actions at different learning steps n. The value of ϵ in the ε-greedy scheme affects the policies acquired by the machine at the end of the learning process. (b) With $\epsilon = 0.05$, the machine has learned four-stroke cyclic motion same as that in the three-sphere rotator (Figure 3.3b), without utilizing the active angle θ_{43}. The angular displacement per cycle $\Delta\bar{\theta}_C = 0.008$; the angular displacement per cycle per stroke $\Delta\bar{\theta}_S = 0.002$. (c) With $\epsilon = 0.1$, the machine has learned an improved but sub-optimal six-stroke cyclic motion with $\Delta\bar{\theta}_C = 0.0161$ and $\Delta\bar{\theta}_S = 0.0027$. (d) With $\epsilon = 0.2$, the machine further improves the performance with another six-stroke cyclic motion with $\Delta\bar{\theta}_C = 0.0238$ and $\Delta\bar{\theta}_S = 0.004$. The motion involves a sequential contraction of all active angles $\theta_{i+1,i}$ from $i = 1$ to $i = 3$, followed by a sequential expansion of all active angles $\theta_{i+1,i}$ from $i = 1$ to $i = 3$. This policy, which consists of traveling waves of actuation propagating in the anti-clockwise direction, represents an extension of the strategy in Purcell's rotator to the case four spheres with all active angles utilized in the sequence. As a remark, the policy obtained with $\epsilon = 0.3$ is the same as that with $\epsilon = 0.2$; yet the more frequent interruptions by the random actions with $\epsilon = 0.3$ leads to a smaller net rotation overall compared with the case with $\epsilon = 0.2$ as shown in (a).

Effect of the Learning Parameters Here we show some simulations results with different α and γ values of the 3-sphere model. The effect of different α and γ become more apparent for systems with an increased number of spheres. To illustrate, we use the results for the case of $N = 9$ in Figure 3.5 below. We show the average net rotation of the machine, $\langle \Delta\bar{\theta} \rangle$, as a function of the learning steps n over 20 sample runs for different values of α (Figure 3.5(a) and γ (Fig. 3.5(b). The performance of the machine generally increases with α: at a small learning rate ($\alpha = 0.2$), the machine is unable to learn the traveling wave policy for the given number of training steps, resulting in the observed sub-optimal performance. The performance improves as α is increased to $\alpha = 0.6$, where the machine is able to learn the traveling wave policy in some but not all sample runs. With a maximized learning rate $\alpha = 1$, the machine acquires the traveling policy in all sample runs, leading to improved performance as shown in Figure 3.5 below.

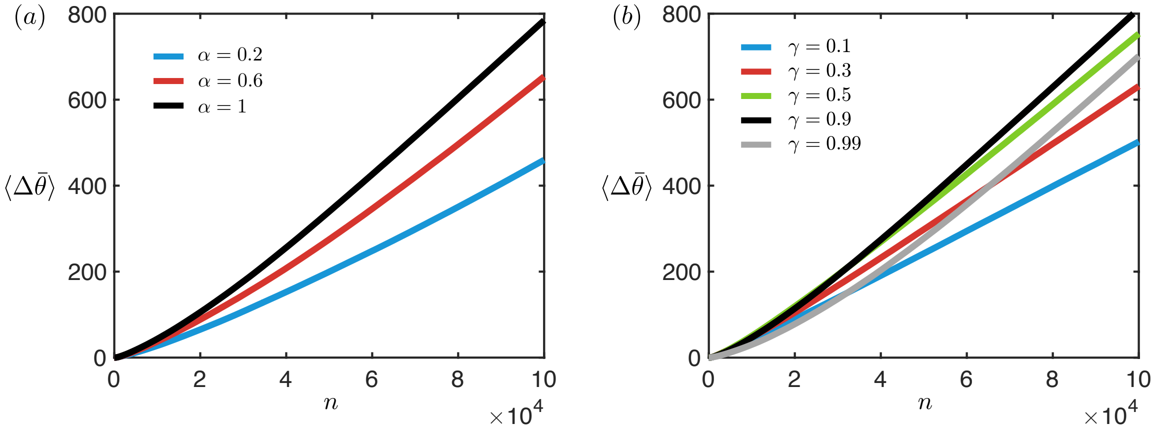


Figure 3.5 The effect of learning parameters α and γ for the case of $N = 9$. The average net rotation of the machine, $\langle \Delta\bar{\theta} \rangle$, as a function of the learning steps n for different values of (a) learning rate α, and (b) discount factor γ, over 20 sample runs. In these simulations, $\phi = \pi/18$, $\epsilon = 0.2$, and $R/L = 0.1$; $\gamma = 0.9$ in (a) and $\alpha = 1$ in (b).

In contrast, the effect of the discount factor γ on the overall performance is non-monotonic as shown in Figure 3.5(b). Similar to the case of translation, a sufficiently large γ (*e.g.*, $\gamma \geq 0.5$) is required for the machine to learn the traveling wave policy.

For a large value of $\gamma = 0.9$, the machine acquires the traveling wave policy in all sample runs. However, a further increase in γ (*e.g.*, $\gamma = 0.99$) leads to rotational strategies other than the traveling wave policy, hindering the overall performance. Based on these findings, we therefore set $\alpha = 1$ and $\gamma = 0.9$ in the simulations.

Effect of Model's Degree of Freedom Currently at each step, only one degree of freedom of the model is allowed to perform. For example, for the 3-sphere model, either θ_{32} changes or θ_{32} changes. We have performed additional simulations for a 3-sphere model, when the machine is allowed to change one or both angles in each learning step, the machines takes more learning steps to explore different actions due to the increase in sizes of the state space and the action space. This hinders the overall performance compared with the case when only one degree of freedom is allowed to change (Figure 3.6), but more rotational strategies emerge as a result. Interestingly, these new strategies are sub-optimal compared with the traveling wave policies. Given a sufficiently large number of learning steps, the machine still evolves to performing the traveling wave policy when both degrees of freedom are allowed to change. This suggests that the piecewise path of the shape deformations may be a good strategy, at least for the 3-sphere model. However, more extensive investigations are required to thoroughly address this interesting question, which we defer to a future study.

3.3.2 N-sphere Rotator

We next extend the analysis beyond the three-sphere configuration. For a configuration with N spheres, the description of the hydrodynamic force and velocity via the Oseen tensor on sphere i can be readily extended from Equation 3.1 as $\mathbf{F}_i = \sum_{j=1}^{N} \mathbf{H}_{ij}\mathbf{V}_j$. Similarly, the torque free condition now reads $\sum_{i=1}^{N} \mathbf{\Gamma}_i = \mathbf{0}$, where $\mathbf{\Gamma}_i = \mathbf{D}_i \times \sum_{j=1}^{N} \mathbf{H}_{ij}\mathbf{V}_j$. Similar to the case of three spheres, there are $N-1$ active elements that can contract or expand any one of the angles between two neighbouring spheres by an amount ϕ, except for the angle θ_{1N}, which only reacts passively to the

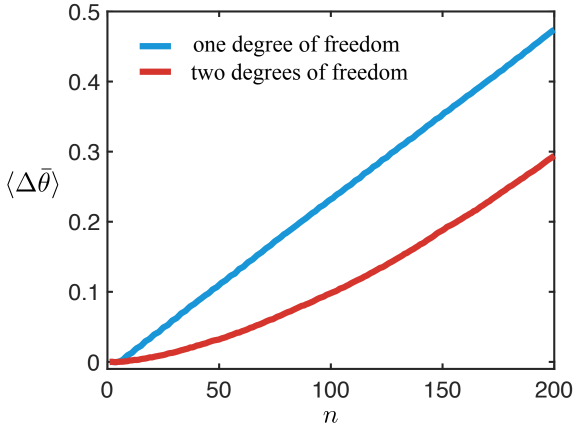


Figure 3.6 The average net rotation of the 3-sphere model when only one degree of freedom (blue) and when both degrees of freedom (red) are allowed to change in each learning step. The results are averaged over 30 sample runs. In these simulations, $\phi = \pi/6$, $\gamma = 0.9$, $\epsilon = 0.05$, and $R/L = 0.1$.

contraction and expansion of other angles. At each step, the Q-learning algorithm informs one pair of neighbouring spheres (*e.g.*, the i and $i+1$ spheres) to extend or contract their angle at a uniform rate ω: $\dot{\theta}_{i+1} - \dot{\theta}_i = \pm\omega$, while keeping other angles fixed (*i.e.*, $\dot{\theta}_j = \dot{\theta}_i$ for $j = 1, 2, ..., i-1$ and $\dot{\theta}_j = \dot{\theta}_{i+1}$ for $j = i+2, i+2, ..., N$). The goal is to learn effective strategies to generate net rotation based on the machine's interaction with the viscous environment.

We remark that as the number of sphere N in the machine increases, the angle between the spheres in its initial (equally spaced) configuration reduces accordingly as $\theta_e = 2\pi/N$. This also limits the angle of contraction (ϕ) allowed as the number of spheres increases in the machine. In order for ϕ to not exceed the maximum angle between the spheres (θ), we set $\phi = \theta_e/4 = \pi/(2N)$ in our simulations for a N-sphere system. In other words, the machine uses a fixed portion of θ_e for contraction. The machine, hence, has a smaller angle of contraction as the number of sphere increases. We note that only a small portion (1/4) of θ_e is used for contraction here to ensure that the spheres are sufficiently far apart for the hydrodynamic description via the Oseen tensor to be valid (see Section 3.2.2).

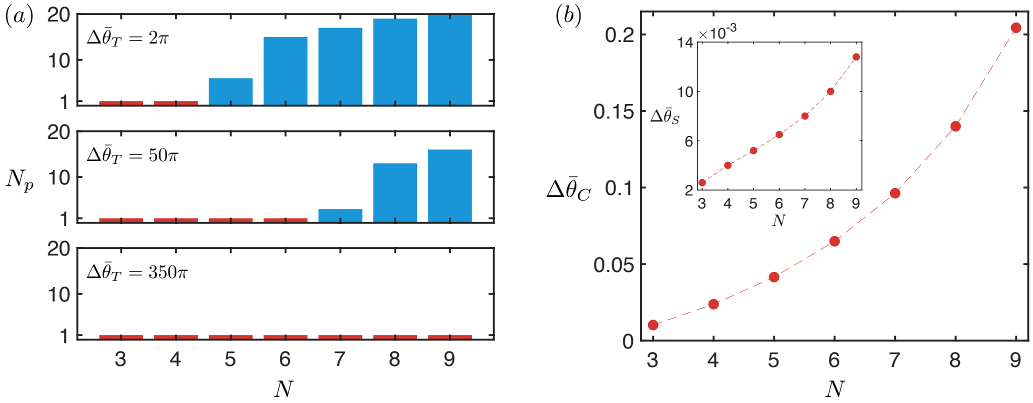


Figure 3.7 Mechanical rotation of a N-sphere rotator via reinforcement learning results. (a) The number of different policies N_p adopted by a N-sphere rotator when the learning process is stopped at different values of target angular displacement, $\Delta\bar{\theta}_T$, in 20 sample runs. For each run, the machine continues to learn until the net angular rotation $\Delta\bar{\theta}_n$ reaches $\Delta\bar{\theta}_T$. With a relatively short learning process ($\Delta\bar{\theta}_T = 2\pi$; top panel), the three-sphere and four-sphere rotators converge to a single policy in all runs (red bars), which correspond to the traveling wave policies. For $N > 4$, the machine adopts a wider variety of different policies as N increases (blue bars). With a longer training process ($\Delta\bar{\theta}_T = 50\pi$; middle panel), more rotators converge to the traveling wave policies at the end of the learning process (red bars), with a reduced number of policies for $N \geq 7$. With a sufficiently long learning process ($\bar{\theta}_T = 350\pi$; bottom panel)), all rotators converge to the traveling wave policies. (b) Characterization of the performance of the traveling wave policies of individual N-sphere rotators by the net angular displacement per cycle $\Delta\bar{\theta}_C$ and the net angular displacement per cycle per stroke (inset) $\Delta\bar{\theta}_S = \Delta\bar{\theta}_C/2(N-1)$, where $2(N-1)$ is the number of strokes in the traveling wave policies. Both $\Delta\bar{\theta}_C$ and $\Delta\bar{\theta}_S$ increase with N. In these simulations, $\phi = \pi/(2N)$, $\gamma = 0.9$, $\epsilon = 0.15$, and $R/L = 0.1$.

When we have a larger number of spheres N in the machine, the increased degree of freedom allows multiple effective strategies to emerge. The policy identified by reinforcement learning largely depends on different learning parameters, including the discount factor, the number of learning steps, and the value of ε in the ε-greedy scheme. We illustrate some general characteristics using a four-sphere ($N = 4$) configuration. Figure 3.4(a) shows that, for a fixed number of learning steps, a four-sphere machine evolves different rotational policies depending on the value of ε in the ε-greedy scheme. We can measure the performance of different policies by the angular displacement per cycle ($\Delta\bar{\theta}_C$) or the displacement per cycle per stroke ($\Delta\bar{\theta}_S = \Delta\bar{\theta}_C/N_s$); the latter measure divides the angular displacement per cycle by the number of strokes involved in the cycle, N_s, to account for the difference in the number of strokes in individual policies.

Similar to the case for translation [99], the value of ε in the ε-greedy scheme plays an important role in the learning process. When there is not any exploration scheme ($\varepsilon = 0$), the machine frequently gets trapped going back and forth between two states, resulting in reciprocal motion that does not yield net rotation [99]. With a small $\varepsilon = 0.05$ (blue line in Figure 3.4(a)), the machine is able to identify an effective but sub-optimal policy for net rotation (Figure 3.4(b)); indeed the four-stroke policy follows the same sequence of strokes as a 3-sphere Purcell's rotator in Figure 3.3(c), with the angle θ_{43} not participating in the gait at all (sphere 4 thus acts essentially like a passive cargo). The angular displacement per cycle for this policy is given by $\Delta\bar{\theta}_C = 0.008$ and $\Delta\bar{\theta}_S = \Delta\bar{\theta}_C/4 = 0.002$ on a per stroke basis. As we increase the exploration rate ($\varepsilon = 0.1$, red line in Figure 3.4(a)), the machine learns an improved six-stroke policy (Figure 3.4(c)) with larger $\Delta\bar{\theta}_C = 0.0161$ and $\Delta\bar{\theta}_S = \Delta\bar{\theta}_C/6 = 0.0027$. For $\varepsilon = 0.2$ (green line in Figure 3.4(a)), the machine acquires another six-stroke policy as shown in Figure 3.4(d) with further improved $\Delta\bar{\theta}_C = 0.0238$ and $\Delta\bar{\theta}_S = \Delta\bar{\theta}_C/6 = 0.004$. This policy here consists of contraction of all active angles in a sequential manner starting from θ_{21} in the anti-clockwise direction, followed by expansion of all active angles again in a sequential manner starting from θ_{21}. More generally, we define such type of policies, namely a sequential contraction of angles $\theta_{i+1,i}$ from $i = 1$ to $i = N - 1$ followed by a sequential expansion of angles $\theta_{i+1,i}$ from $i = 1$ to $i = N - 1$, traveling wave policies, because the sequence of action corresponds to a propagation of traveling wave of actuation in the anti-clockwise direction. These traveling wave policies therefore consists of $2(N - 1)$ strokes; indeed the sequence of strokes in Purcell's rotator ($N = 3$) in Figure 3.3(c) and the $N = 4$ policy in Figure 3.4(d) are both traveling wave policies. As a remark, the machine also learns the traveling wave policy with an even higher exploration rate ($\varepsilon = 0.3$, black line in Figure 3.4(a)); yet the overall displacement of the angular centroid is less compared with the case with $\varepsilon = 0.2$ (green line) because the sequence of actions is frequently interrupted by the random actions at the higher value of ε.

Next, we further increase the number of spheres in the system up to $N = 9$ and examine the number of different policies obtained by reinforcement learning for different values of N. The policy eventually adopted by the machine largely depends on the number of learning steps allowed. In Figure 3.7(a), we examine the policy adopted by the machine when its rotation has reached a certain target angular displacement, $\Delta\bar{\theta}_T$. For instance, when the machine is allowed to learn up to a target angular displacement of $\Delta\bar{\theta}_T = 2\pi$ (top panel, Figure 3.4(a)), all trials for $N = 3$ and $N = 4$ machines converge to a single policy – the traveling wave policy. However, increasingly more policies emerge in the trials for machines with a larger number of spheres. When more learning is allowed by increasing the target angular displacement to $\Delta\bar{\theta}_T = 50\pi$ (middle panel in Figure 3.7(a)), more configurations converge to the traveling wave policies ($N = 3$ to $N = 6$) with lower number of policies appearing in the trials for $N > 7$. Finally, when sufficient amount of learning is allowed (*e.g.*, $\Delta\bar{\theta}_T = 350\pi$, bottom panel in Figure 3.7(a)), all configurations considered converge to a single policy, namely the traveling wave policy. These results demonstrate that the larger the target angular displacement, the more chance the the machine can learn to converge to the traveling wave policy, suggesting its optimality in generating net rotation at low Reynolds number. We also note that the same trend applies to swimmers consisting of linear chains of spheres for net translation [99]: given sufficient amount of learning, the swimmers with different numbers of spheres all converge to the same type of traveling wave policy via reinforcement learning.

In Figure 3.7(b), we quantify the performance of the traveling wave policy for different values of N in terms of the angular displacement per cycle $\Delta\bar{\theta}_C$ and the angular displacement per cycle per stroke $\Delta\bar{\theta}_S$ (inset). As the number of sphere N increases, the traveling wave policy generates more displacement per cycle $\Delta\bar{\theta}_C$. Even though the number of strokes in the traveling wave policy also increases as 2(N-1), machine with a higher number of spheres still produce a larger displacement per cycle per stroke, $\Delta\bar{\theta}_S = \Delta\bar{\theta}_C/2(N-1)$, as shown in the inset.

3.4 Concluding Remarks

In this work, we demonstrate the first use of reinforcement learning to generate mechanical rotation at low Reynolds numbers. This alternative approach diverges from the conventional way of prescribing a pre-defined sequence of strokes based on knowledge of locomotion; instead we exploit a simple reinforcement learning algorithm (Q-learning) to enable a machine to identify effective rotational policies based on its interaction with the surroundings, without requiring prior knowledge of locomotion. When the machine has the minimum degrees of freedom for net rotation ($N = 3$), it recovers the strategy identified by Dreyfus *et al.* for Purcell's rotator, which shares similarity with the conformational changes of some molecular motors undergoing ATP- or photochemically-driven rotational movements [25, 57, 58]. For an increased number of spheres ($N > 4$), the machine is capable of identifying multiple effective policies for net rotation, depending on different learning parameters in the system. However, when sufficient learning steps are allowed, the machine eventually evolves to a single policy – the traveling wave policy. The traveling wave policy enables the machine to generate net rotation by a sequential contraction (and then expansion) of active angles in the machine. The sequence of strokes in Purcell's ratotor is a special case of this family of traveling wave policies. As a remark, the change in the angular centroid is used as the reward in reinforcement learning here based on the goal to maximize net rotation of the machine. Rewards accounting for energy consumption due to different actions may also be considered in future work for optimization based on energetic considerations. Recent works have also suggested traveling wavelike deformations to be energy-optimal strokes for locomotion [1, 4, 27, 61].

The alternative approach in this work is particularly desirable when a machine explores an environment with unknown properties or when the knowledge of locomotion remains incomplete in more complex environments. The approach based on reinforcement learning bypasses the challenging of designing locomotory gaits in advance in these situations. As a proof of concept, we adopt a standard Q-learning algorithm for its simplicity and expressiveness. There exists a vast potential in the use of more advanced machine learning approaches [69, 86–88, 90, 96] for locomotion problems involving more complex maneuvers in future works.

CHAPTER 4

DEEP REINFORCEMENT LEARNING FOR GAIT SWITCHING AND TARGETED NAVIGATION OF MICROSWIMMERS

4.1 Background and Related Work

The design of artificial microswimmer arises many recent interests because of its potential biomedical application. Smart artificial microswimmers capable of adapting their locomotion behaviors in response to surrounding environments offer exciting opportunities for biomedical applications. Swimming microorganisms have evolved versatile navigation strategies by switching their locomotory gaits in response to their surroundings [62]. Their navigation strategies typically involve switching between translation and rotation modes such as run-and-tumble and reverse-and-flick in bacteria [12, 52, 93, 106], as well as run-stop-shock and run-and-spin in eukaryotes [98, 102]. One fundamental challenge for applications of smart artificial microswimmers is to achieve targeted navigation towards specific targets. Here we employ a deep reinforcement learning algorithm to enable a reconfigurable microswimmer to self-learn a set of locomotory gaits as well as the corresponding gait-switching mechanisms for performing targeted navigation. Interestingly, the navigation strategies learnt by the swimmer via artificial intelligence is reminiscent to the gait-switching behaviors observed in biological cells due to natural selection. Our results demonstrate the potential of using artificial intelligence to develop smart artificial microswimmers that can adapt to complex biological environments similar to biological cells. Such an adaptive, multimodal gait-switching ability is particularly desirable for biomedical applications of artificial microswimmers such as targeted drug delivery and microsurgery [15,32,39,51,110], which require navigation towards target locations in biological media with uncontrolled and/or unpredictable environmental factors [14, 67, 73].

Pioneering works by Purcell and subsequent studies demonstrated how simple reconfigurable systems with ingenious locomotory gaits can generate net translation and rotation, given the stringent constraints for locomotion at low Reynolds numbers [79]. Yet, the design of locomotory gaits becomes increasingly intractable when more sophisticated maneuvers are required or environmental perturbations are present. Existing microswimmers are therefore typically designed with fixed locomotory gaits and rely on manual interventions for navigation [22, 32, 48, 76, 78, 101]. It remains an unresolved challenge in developing microswimmers with adaptive locomotory strategies similar to that of biological cells that can navigate complex environments autonomously. Modular microrobotics and the use of soft active materials [49, 50] have been proposed to address the challenge.

More recently, the rapid development of artificial intelligence (AI) and its applications in locomotion problems [13,36,38,54,82,100] have opened different paths towards designing the next generation of smart microswimmers [17, 97]. Various machine learning approaches have enabled the navigation of active particles in the presence of background flows [2, 18], thermal fluctuations [70, 85], and obstacles [108]. As minimal models, the microswimmers are often modeled as active particles with prescribed self-propelling velocities and certain degrees of freedom for speed variation and re-orientation. However, the complex adjustments in locomotory gaits required for such adaptations are typically not accounted for. Recent studies have begun to examine how different machine learning techniques enable reconfigurable microswimmers to evolve effective gaits for self-propulsion [99] and chemotactic repsonse [45].

Here, we combine reinforcement learning (RL) with artificial neural network to enable a simple reconfigurable system to perform complex maneuvers in a low-Reynolds-number environment. We show that the deep RL framework empowers a microswimmer to adapt its locomotory gaits in accomplishing sophisticated tasks including targeted navigation and path tracing, without being explicitly programmed. The multimodel gait switching strategies are reminiscent of that adopted by swimming microorganisms. Furthermore, we examine the performance of these locomotion strategies against perturbations by background flows. The results showcase the versatility of AI-powered swimmers and their robustness in media with uncontrolled environmental factors.

4.2 Formulation

4.2.1 Geometric Setup

We consider a simple reconfigurable system consisting of three spheres with radius R and centers $\mathbf{r}_i$ $(i = 1, 2, 3)$ connected by two arms with variable lengths and orientations as shown in Figure 4.1(a)). This setup generalizes previous swimmer models proposed by Najafi and Golestanian [71] and Ledesma-Aguilar *et al.* [63] by allowing more degrees of freedom. The interaction between the system and the surrounding viscous fluid is modeled by low Reynolds number hydrodynamics, imposing stringent constraints on the locomotive capability of the system. Unlike the traditional paradigm where the locomotory gaits are prescribed in advance [5, 8, 41, 63, 71, 103], here we exploit a deep RL framework to enable the system to self-learn a set of locomotory gaits to swim along a target direction, θ_T. We employ a deep neural network based on the Actor-Critic structure and implement the Proximal Policy Optimization (PPO) algorithm [54, 88] to train and update the agent (*i.e.*, AI) in charge of the decision making process (Figure 4.1b)). The deep RL framework here extends previous studies from discrete action spaces to continuous action spaces [18, 64, 70, 99], enhancing the swimmer's capability in developing more versatile locomotory gaits for complex navigation tasks.

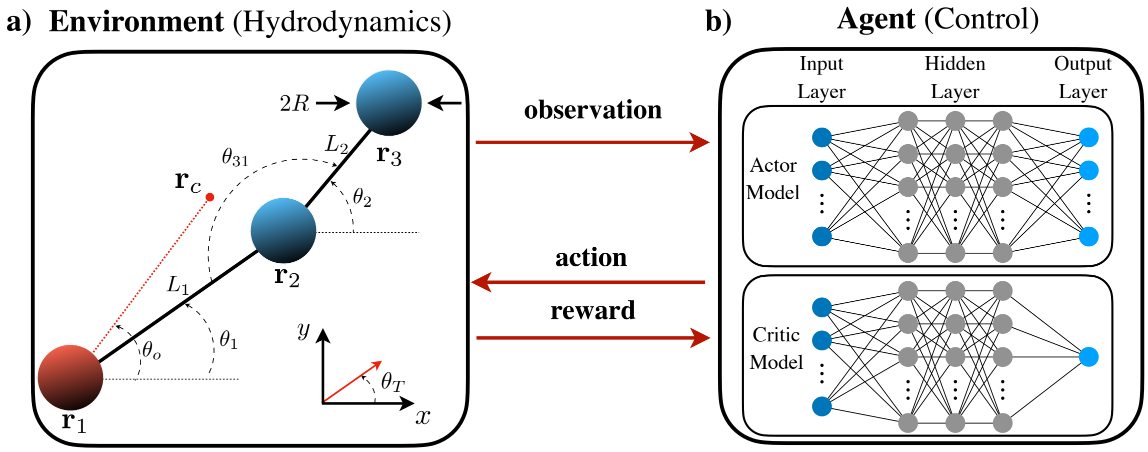


Figure 4.1 Schematics of the model microswimmer and the deep neural network with Actor-Critic structure. a) Schematic of the model microswimmer consisting of three spheres with raidus R and centers $\mathbf{r}_i (i = 1, 2, 3)$. We mark the leftmost sphere $\mathbf{r}_1$ as red and the other two spheres $\mathbf{r}_2, \mathbf{r}_3$ as blue to indicate the current orientation of the swimmer. The spheres are connected by two arms with variable lengths L_1, L_2 and orientations θ_1, θ_2, where θ_{31} is the intermediate angle between two arms. The swimmer's orientation θ_o is defined based on the relative position between the swimmer's centroid $\mathbf{r}_c = \sum_i \mathbf{r}_i/3$ and $\mathbf{r}_1$ as $\theta_o = \arg(\mathbf{r}_c - \mathbf{r}_1)$. The swimmer is trained to swim along a target direction θ_T. b) Schematic of Actor-Critic neural networks. Both networks consist of three sets of layers (input layer, hidden layer, and output layer). Each layer is composed of neurons (marked as nodes). The weights of the neural network are illustrated as links in between the nodes. The input layer has the same dimension as the observation. The three linear hidden layers have the dimension of $64, 32, 32$, respectively. The output layer dimension of the actor network is the same as the action space dimension, whereas the output layer of the actor network has only 1 neuron.

We discuss the general idea as follows: based on the current observation, a reinforcement learning agent decides the next action using the Actor neural network. The next action is then evaluated by the Critic neural network to guide the training process. The swimmer performs the action advised by the agent and interacts with the hydrodynamic environment, leading to movements that constitute the next observation and reward. Both the Actor and Critic neural network are updated periodically to improve the overall performance. See more details in the Deep Reinforcement Learning section.

4.2.2 Hydrodynamic Interactions.

The interaction between the spheres and their surrounding fluid is governed by the Stokes equation ($\nabla p = \mu \nabla^2 \mathbf{u}$, $\nabla \cdot \mathbf{u} = 0$). Here, p, μ and $\mathbf{u}$ represent, respectively, the pressure, dynamic viscosity, and velocity field. In this low Reynolds number regime, the velocities of the spheres $\mathbf{V}_i$ and the forces $\mathbf{F}_i$ acting on them can be related linearly as

$$\mathbf{V}_i = \mathbf{G}_{ij}\mathbf{F}_j, \tag{4.1}$$

where $\mathbf{G}_{ij}$ is the Oseen tensor [23, 44, 56] given by

$$\mathbf{G}_{ij} = \begin{cases} \frac{1}{6\pi\mu R}\mathbf{I}, \\ \frac{1}{8\pi\mu|\mathbf{r}_i - \mathbf{r}_j|}(\mathbf{I} + \hat{\mathbf{r}}_{ij}\hat{\mathbf{r}}_{ij}). \end{cases} \tag{4.2}$$

Here, $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{r}}_{ij} = (\mathbf{r}_i - \mathbf{r}_j)/|\mathbf{r}_i - \mathbf{r}_j|$ denotes the unit vector between spheres i and j. The torque acting on the sphere i is calculated by $\mathbf{T}_i = \mathbf{r}_i \times \mathbf{F}_i$. The rate of actuation of the arm lengths $\dot{L}_1$, $\dot{L}_2$ and the intermediate angle $\dot{\theta}_{31}$ can be expressed in terms of the velocities of the spheres $\mathbf{V}_i$. The kinematics of the swimmer is fully determined upon applying the force free ($\sum_i \mathbf{F}_i = 0$) and torque-free ($\sum_i \mathbf{T}_i = 0$) conditions. The Oseen tensor hydrodynamic description is valid when the spheres are not in close proximity ($R \ll L$). We therefore constrain the arm and angle contractions such that $0.6L \leq L_1, L_2 \leq L$ and $2\pi/3 \leq \theta_{31} \leq 4\pi/3$.

The actuation rate of the arm lengths $\dot{L}_1, \dot{L}_2$ can be expressed in terms of the relative velocities of the spheres parallel to the arm orientations:

$$(\mathbf{V}_2 - \mathbf{V}_1) \cdot \hat{\mathbf{r}}_{21} = \dot{L}_1, \tag{4.3}$$

$$(\mathbf{V}_3 - \mathbf{V}_2) \cdot \hat{\mathbf{r}}_{32} = \dot{L}_2, \tag{4.4}$$

The actuation rate of the intermediate angle $\dot{\theta}_{31}$ can be expressed in terms of the relative velocities of the spheres perpendicular to the arm orientations:

$$(\mathbf{V}_2 - \mathbf{V}_1) \cdot \frac{d\hat{\mathbf{r}}_{21}}{d\theta_1} = L_1\dot{\theta}_1, \tag{4.5}$$

$$(\mathbf{V}_3 - \mathbf{V}_2) \cdot \frac{d\hat{\mathbf{r}}_{32}}{d\theta_2} = L_2\dot{\theta}_2, \tag{4.6}$$

$$\dot{\theta}_1 - \dot{\theta}_2 = \dot{\theta}_{31}, \tag{4.7}$$

where $\dot{\theta}_1$ and $\dot{\theta}_2$ are the arm rotation speeds. Together with the Oseen tensor description of the hydrodynamic interaction between the spheres, Equations (4.1)–(4.2) in the main text, and the overall force-free and torque-free conditions, the kinematics of the swimmer is fully determined.

In presenting our results, we scale lengths by the fully extended arm length L, velocities by a characteristic actuation rate of the arm V_c, and hence time by L/V_c and forces by $\mu L V_c$. We nondimensionalize lengths by the fully extended arm length L, velocities by a characteristic actuation rate of the arms V_c. This results in the characteristic time scale of $T = L/V_c$, force scale of $\mu L V_c$, and torque scale of $\mu L^2 V_c$. Here we impose a maximum possible actuation rate of the arms as $4V_c$. We use asterisks ($*$) to denote the dimensionless variables: the arm lengths L_1^* and L_2^* vary in the range of $[0.6, 1.0]$; the actuation rate of arms $\dot{L}_1^*$ and $\dot{L}_2^*$ vary in the range of $[-4, 4]$; and the actuation rate of intermediate angle $\dot{\theta}_{31}^* \in [-2\pi/3, 2\pi/3]$. We further set $R^* = 0.1$ to keep the spheres far apart. Following those scales, we have the dimensionless rotlet flow strength $\gamma^* = \gamma / L^2 V_c$.

In our simulations, we assume uniform actuation rates of $\dot{L}_1^*, \dot{L}_2^*, \dot{\theta}_{31}^*$ during each action step. To determine the proper actuation rate, the swimmer first receives a suggested action by the PPO agent and clips the given action based on the physical constraints described above. The clipped action is then used to determine the swimmer's kinematics. We set the time duration for each action step as $\Delta t^* = 0.1$. In the main text, we drop the asterisks for simplicity and only present dimensionless variables.

4.2.3 Deep Reinforcement Learning

Reinforcement learning is a branch of machine learning that deals with how to learn control strategies to interact with a complex environment. It is a framework for learning how to interact with the surroundings from experience and it is inspired by how biological systems. By trial and error through experience, through positive and negative rewards and feedback, the agent learn how to interact with their environment. Q-learning is a foundational algorithm in reinforcement learning. In this paradigm, an agent can perceive its state and perform actions. After each action, a numerical reward is given. The goal of the agent is to maximize the total reward it receives over time. The experience gained by the agent is stored in a Q-matrix, $Q(s_n, a_n)$, which is an state-action-value function that captures the expected long-term reward for taking the action a_n at the given state s_n.The Q-learning algorithms involve estimating state-action value functions that indicate how good it is to be in a given state (in terms of total expected reward in the long term), or how good it is to perform a particular action in a certain state. The most basic way to build this value function consists in updating a table that contains a value for each state (or each state-action pair), but this approach is not practical for large scale problems and lack of computational efficiency. In order to deal with tasks that have a very large number of states, it is necessary to use the generalization capabilities of function approximators. The neural networks are a particular case of such function approximators that can be used in combination with Q-learning which can be used to map input states to (action, Q-value) pairs. The aim of the Deep Q-Network (DQN) implementation is to improve the computational time of simple Q-learning, find optimal hyperparameters and apply the framework to the problems with large degrees of freedom.

Proximal Policy Optimization Policy gradient methods are a type of reinforcement learning techniques that rely upon optimizing parameterized policies with respect to the expected return (long-term cumulative reward) by gradient descent. A large amount of theory behind RL lies under the assumption of The Reward Hypothesis which in summary states that all goals and purposes of an agent can be explained by a single scalar called the reward. *The Reward Hypothesis*: That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward). The agent must formally work through a theoretical framework known as a Markov Decision Process which consists of a decision (what action to take?) to be made at each state. This gives rise to a sequence of states, actions and rewards known as a trajectory, $S_0, A_0, R_1, S_1, A_1, R_2, \ldots$ and the objective is to maximize this set of rewards.

To improve training stability, need to avoid parameter updates that change the policy too much at one step. To solve this, the Proximal Policy Optimization (PPO) is easy to tune and has better sample efficient. Different from Deep Q Network, PPO doesn't use a replay buffer to store past experiences (Deep Q Network has experience replay. When the agent interact with the environment with policy π, it will store transition experience (s, a, r, s') in replay buffer. When learning with SGD, the agent sample batch-experience from replay buffer, learning batch by batch). Instead, it learns directly from what the agent encounters in the env and once a batch of experience used to do a gradient update the experience is then discarded and the policy moves on and this also means that policy gradient methods are typically less sample efficient than queue learning methods because they only use the collected experience once for doing an update.

We use Proximal Policy Optimization Algorithm to train our RL agent for the stationary environment and partial models in the non-stationary environment. The agent's motion control is managed with a neural network with an Actor-Critic structure. The Actor network can be considered as a stochastic control policy $\pi_\phi(a_t|o_t)$, where it generates an action a_t given an observation o_t following a Gaussian distribution. Here, ϕ represents all the parameters of the actor neural network. The Critic network is used to compute the value function V_φ by assuming the agent starts at an observation o and acts according to a particular policy π_ϕ. The parameters in the critic network is represented as φ.

To effectively train the swimmer, we divide the total training process into episodes. Each episode can be considered as one round, which terminates after a fixed amount of training steps ($N_L = 100$). To ensure fully exploration of the observation space, we randomly initialize the swimmer's geometric configurations (L_1^*, L_2^*) and the target direction (θ_T) at the beginning of each episode.

At time t, the agent receives its current observation o_t and samples action a_t based on the policy π_ϕ. Given a_t, the swimmer interacts with its surrounding and calculates the next state s_{t+1} and reward r_t. The next observation o_{t+1} extracted from s_{t+1} is sent to the agent for the next iteration. All the observations, actions, rewards and sampling probabilities are stored for the agent's update. The update process begins after running fix amount of episodes $N_E = 20$ (Total training steps of an update is therefore: $N_s = N_E * N_L = 2000$). The goal for the update is to optimize ϕ so that the expected long term rewards $J(\pi_\phi) = \mathbf{E}[R_{t=0}|\pi_\phi]$ is maximized.

The expectation is taken with respect to each running episode, τ. Here, we use the infinite-horizon discounted returns $r_t = \sum_{t'}^{\infty} \gamma^{t'-t} r_{t'}$, where γ is the discount factor measuring the greediness of the algorithm. We set $\gamma = 0.99$ ensuring its farsightedness. To solve this optimization problem, we use the typical policy gradient approach estimation: $\nabla_\phi J(\pi_\phi)$. More specifically, we implemented the clipped advantage PPO algorithm to avoid large changes in each gradient update. We estimated the surrogate objective $J(\pi_\phi)$ by clipping the probability ratio $r(\phi)$ times the advantage function $\hat{A}_t$. The probability ratio measures the probability of selecting an action for the current policy over the old policy ($r(\phi) = \frac{\pi_\phi(a|o)_{N_s \times 1}}{\pi_{\phi_{\text{old}}}(a|o)_{N_s \times 1}}$). The advantage function $\hat{A}_t$ describes the relative advantage of taking an action a based on an observation o over a randomly selected action and is calculated by subtracting the value function $V_{N_s \times 1}$ from the discounted return $R_{N_s \times 1}$ ($\hat{A}_t = R_{N_s \times 1} - V_{N_s \times 1}$).

We then update the parameters ϕ, φ via a typical gradient descent algorithm: Adam optimizer. The full detail for our implementation in combination with high confidence change point detection method is included in the Algorithm 1 and 2. Here, K_e is the total epoch number. N_L is the number of steps in one episode, and N_s is the total number of steps for each update. The PPO algorithm uses fixed-length trajectory segments τ. During each iteration, each of N_A parallel actors collect T time steps of data, then we construct the surrogate loss on these $N_A T$ time steps of data, and optimize it with Adam for K_e epochs.

High Confidence Change Point Detection Method We first introduce joint probability distribution p_{θ_k} associated with a partial model k, which is a multivariate Gaussian distribution parameterized by an ensemble of N neural networks predicting the next observation and reward conditioned on the current observation and action through the mean and covariance:

$$p_{\theta_k^n}(o_{t+1}, r_t | o_t, a_t) = \mathcal{N}(\mu_{\theta_k^n}(o_t, a_t), \Sigma_{\theta_k^n}(o_t, a_t)) \tag{4.8}$$

The weights of each neural network n is parameterized by θ_k^n with $\mu_{\theta_k^n}(o_t, a_t), \Sigma_{\theta_k^n}(o_t, a_t)$ the neural network outputs. We periodically update the network's weighs by minimizing the negative log prediction likelihood loss function:

$$\mathcal{J}(\theta, D) = E_{(o_t, a_t, r_t, o_{t+1})\ D}[-\log p_\theta(o_{t+1}, r_t | o_t, a_t)], \tag{4.9}$$

where D is the dataset contains all the simulation experience. Using the joint probability distribution, we can calculate the log likelihood ratio at time t:

$$L_{k,t} = \log(p_{\theta_k}(o_{t+1}, r_t | o_t, a_t) / p_{\theta_{z_t}}(o_{t+1}, r_t | o_t, a_t)) \tag{4.10}$$

where $E[L_{k,t}] > 0$ if partial model k seems more probable than the current partial model. Lastly, the high confidence change point detection is realized by MCUSUM statistics,

$$W_{k,t} \leftarrow \max(0, W_{k,t-t} + L_{k,t}),\ k \in [1, 2] \cup [\text{new}] \tag{4.11}$$

, which can be interpreted as a quality signal inferring if a known partial model might be better suited for the current environment or a new partial model should be created. We further set a threshold h to determine the next partial model z_t:

$$z_t \leftarrow \begin{cases} \text{argmax}_k W_{k,t}, & \text{if } \exists k \in [1, 2] \cup [\text{new}] \text{ s.t.} W_{k,t} > h \\ z_{t_1}, & \text{else} \end{cases} \tag{4.12}$$

The detail of our implementation is included below in the Algorithm 1 and 2. A complete description of the method can be found in [3].

Algorithm Pseudocode

Algorithm 1 Context Detection RL

```
Input: Non-stationary environment $E$, threshold $h$, episode length $N_e$, policy update step $N_s$, context detection update step $F$.
$z_0 \leftarrow 1$;$K \leftarrow 1$; $W_{z_0,0} \leftarrow 0$; $W_{\text{new},0} \leftarrow 0$
Initialize model $p_{\theta_{z_0}}$, policy $\pi_{\phi_{z_0}}$, dataset $D_{z_0}$, initial state $s_0$
$M \leftarrow \{p_{\theta_{z_0}}\}$
for time step $t = 0, 1, ...$ do
    if $\mathrm{mod}(t, N_e) = 0$ then
        Reset state $s_t$
    end if
    Sample action $a_t$ from policy $\pi_{\phi_{z_t}}$
    Evaluate the next observation $o_{t+1}$ and reward $r_t$ following the swimmer's hydrodynamics in non-stationary environment $E$
    Update MCUSUM statistics $W_{k,t}$
    Update $z_t$
    if $z_t \neq z_{t-1}$ then
        Reset MCUSUM statistics
        if $z_t = \text{new}$ then
            $K \leftarrow K + 1$; $z_t \leftarrow K$
            Initialize model $p_{\theta_{z_t}}$ and policy $\pi_{\phi_{z_t}}$
            $M \leftarrow M \cup \{p_{\theta_{z_t}}\}$
        end if
    end if
    $D_{z_t} \leftarrow D_{z_t} \cup \{(o_t, a_t, r_t, o_{t+1})\}$
    if $\mathrm{mod}(t, F) = 0$ then
        $\theta_{z_t} \leftarrow \theta_{z_t} - \lambda_p \nabla J_p(\theta_{z_t}, D_{z_t})$
    end if
    if $t = 0$ or $\mathrm{mod}(t, N_s) \neq 0$ then
        append observation $o_{t+1}$, action $a_t$, reward $r_t$ and action sampling probability $\pi_{\phi_{z_t}}(a_t|o_t)$ to lists $o_{N_s\times 2}$, $a_{N_s\times 2}$, $R_{N_s\times 1}$, and $\pi_{\phi_{\text{old}}}(a_t|o_t)_{N_s\times 1}$
    else
        Update the policy using Algorithm 2
    end if
end for
```

Algorithm 2 PPO, Actor-Critic, Update the Agent

```
Input: Initial policy parameter $\phi$, initial value function parameter $\varphi$
for $i = 0, 1, 2, ...K_e$ do
    Compute infinite-horizon discounted returns $R_{N_s\times 1}$
    Evaluate expected returns $V_{N_s\times 1}$ using observations $o_{N_s\times 2}$ and value function $V_\varphi$
    Compute the advantage function: $\hat{A}_t = R_{N_s\times 1} - V_{N_s\times 1}$.
    Evaluate the probability for policy $\pi_\phi$ using observations $o_{N_s\times 2}$ and actions $a_{N_s\times 2}$, store the probability to $\pi_\phi(a|o)_{N_s\times 1}$
    Compute the probability ratio: $r(\phi) = \frac{\pi_\phi(a|o)_{N_s\times 1}}{\pi_{\phi_{\text{old}}}(a|o)_{N_s\times 1}}$
    Compute the clipped surrogate loss function: $L^{\text{CLIP}}(\phi) = E[\min(r(\phi)\hat{A}_t, \mathrm{clip}(r(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$
    Compute the value-function loss: $L^{\text{VF}}(\varphi) = \frac{1}{2}E[(R_{N_s\times 1} - V_{N_s\times 1})^2]$
    Compute the entropy loss: $L^{\text{S}} = \alpha S[\pi_\phi]$
    Compute the total loss: $L(\phi, \varphi) = -L^{\text{CLIP}}(\phi) + L^{\text{VF}}(\varphi) - L^{\text{S}}$
    Optimize surrogate $L$ wrt $(\phi, \varphi)$, with $K$ epochs and minibatch size $M \leq N_a T$
    $\phi_{old} \leftarrow \phi$, $\varphi_{old} \leftarrow \varphi$
end for
```

4.3 Results and Discussion

4.3.1 Targeted Navigation.

We first use the deep RL framework to train the model system in swimming along a target direction θ_T, given any arbitrary initial swimmer's orientation θ_o. The swimmer's orientation is defined based on the relative position between the swimmer's centroid $\mathbf{r}_c = \sum_i \mathbf{r}_i/3$ and $\mathbf{r}_1$ as $\theta_o = \arg(\mathbf{r}_c - \mathbf{r}_1)$ (Figure 4.1).

In the RL algorithm, the state $s \in (\mathbf{r}_1, L_1, L_2, \theta_1, \theta_2)$ of the system is specified by the sphere center $\mathbf{r}_1$, arm lengths L_1, L_2, and arm orientations θ_1, θ_2. The observation $o \in (L_1, L_2, \theta_{31}, \cos\theta_d, \sin\theta_d)$ is extracted from the state, where θ_{31} is the intermediate angle and $\theta_d = \theta_T - \theta_o$ is the difference between the target direction θ_T and the swimmer's orientation θ_o; note that the angle difference is expressed in terms of $(\cos\theta_d, \sin\theta_d)$ to avoid discontinuity in the orientation space. The AI decides the swimmer's next action based on the observation using the Actor neural network: for each action step Δt, the swimmer performs an action $a \in (\dot{L}_1, \dot{L}_2, \dot{\theta}_{31})$ by actuating its two arms, leading to swimmer displacement. To quantify the success of a given action, the reward is measured by the displacement of the swimmer's centroid along the target direction, $r_t = (\mathbf{r}_{c_{t+1}} - \mathbf{r}_{c_t}) \cdot (\cos\theta_T,\ \sin\theta_T)$.

We divide the training process into a total of N_e episodes, with each episode consisting of $N_t = 150$ learning steps. To ensure a full exploration of the observation space o, both the initial swimmer state s and the target direction θ_T are randomized in each episode. Based on the training results after every 20 episodes, the Critic neural network updates the AI to maximize the expected long-term rewards $\mathbf{E}[R_{t=0}|\pi_\theta]$, where π_θ is the stochastic control policy, $R_t = \sum_{t'}^{\infty} \gamma^{t'-t} r_{t'}$ is the infinite-horizon discounted future returns, and γ is the discount factor measuring the greediness of the algorithm [88,95]. A large discount factor $\gamma = 0.99$ is set here to ensure farsightedness of the algorithm. As the episodes proceed, the Actor-Critic structure progressively trains the AI and thereby enhances the performance of the swimmer.

In Figure 4.2, we visualize the navigation of a trained swimmer along a target direction θ_T, given a substantially different initial orientation, θ_o. The swimmer's targeted navigation is accomplished in three stages: (1) in the initial phase (blue curve and regime), the swimmer employs "steering" gaits primarily for re-orientation, followed by (2) "transition" phase (red curve and regime) in which the swimmer continues to adjust its direction while self-propelling, before reaching (3) the "translation" phase (green curve and regime), in which the re-orientation is complete and the swimmer simply self-propels along the target direction. This example illustrates how an AI-powered reconfigurable system evolves a multimodal navigation strategy without explicitly programmed or relying on any prior knowledge of low-Reynolds-number locomotion. We next analyze the locomotory gaits in each mode in the evolved strategy.

4.3.2 Multimodal Locomotory Gaits.

Here we examine the details of the locomotory gaits acquired by the swimmer for targeted navigation in the steering, transition, and translation modes. We distinguish these gaits by visualizing their configurational changes in the three-dimensional (3D) configuration space of the swimmer (L_1, L_2, θ_{31}) in Figure 4.3. Here we utilize an example of a swimmer navigating towards a target direction with $|\theta_d| > \pi/2$ to illustrate the switching between different locomotory gaits (Figure 4.3a)). The swimmer needs to re-orient itself in the counter-clockwise direction in this example; an example for the case of clockwise rotation is included in the Appendix. The dots in Figure 4.3a) represent configurations at different action steps. The configurations for the steering (blue dots), transition (red dots), and translation (green dots) gaits are clustered in different regions in the configuration space. A representative sequence of configurational changes for each mode of gaits are shown as solid lines to aid visualization (Figure 4.3a)).

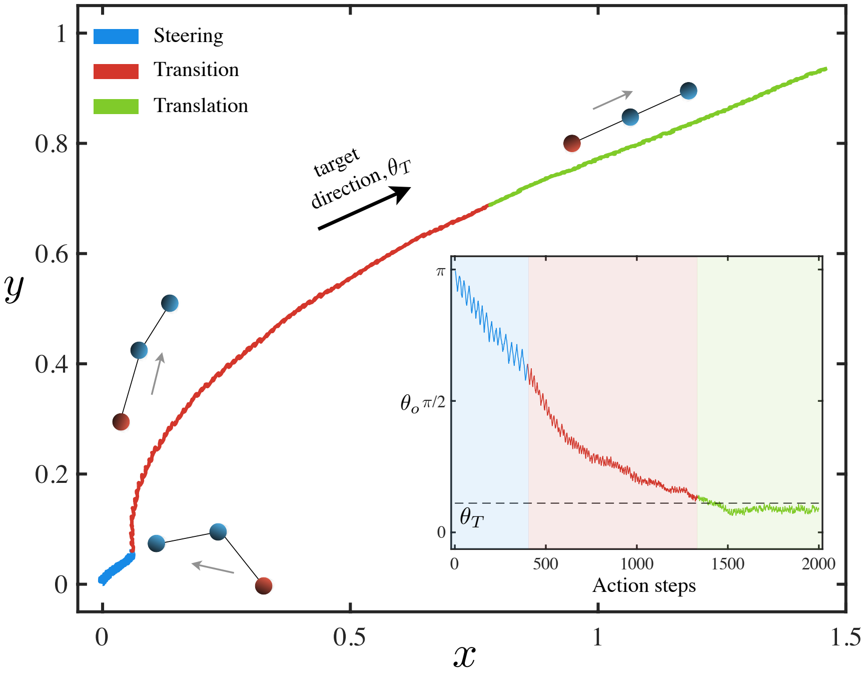


Figure 4.2 Example of target navigation utilizing three distinct locomotory gaits. The Artificial Intelligence powered swimmer switches between distinct locomotory gaits (steering, transition, translation) advised by the reinforcement learning algorithm to steer itself towards a specified target direction θ_T (black arrow) and swim along the target direction afterwards. Different parts of the swimmer's trajectory are colored to represent the locomotion due to different locomotory gaits, where the steering, transition, and translation gaits are marked as blue, red, green, respectively. Schematics of the swimmer configurations (not-to-scale) are shown for illustrative purpose, where the leftmost sphere is marked as red and other two spheres marked as blue to indicate the swimmer's current orientation (grey arrows). The inset shows the change in swimmer's orientation θ_o over action steps.

We further examine the evolution of L_1, L_2, and θ_{31} using the representative sequences of configurational changes identified in Figure 4.3a) for each mode of gaits. For the steering gaits (Figure 4.3b)), blue lines and Figure 4.3d), blue box, the swimmer repeatedly extends and contracts L_2 and θ_{31}, but keeps L_1 constant (the left arm rests in the fully contracted state). The steering gaits thus reside in the L_2-θ_{31} plane in Figure 4.3a) (blue line). The large variation in θ_{31} generates net rotation, substantially re-orientating the swimmer orientation with a relatively small net translation (Figure 4.3c)). For the transition gaits (Figure 4.3b)), red lines and Figure 4.3d), red box), the swimmer repeatedly extends and contracts all L_1, L_2 and θ_{31}, leading to significant amounts of both net rotation and translation (Figure 4.3c)). In the configuration space (Figure 4.3a)), the transition gaits tilt into the L_1-L_2 plane with an average θ_{31} less than π (red line). Compared with the steering gaits, the variation of θ_{31} becomes more restricted (Figure 4.3b)), resulting in smaller net rotation for fine tuning of the swimmer's orientation in the transition phase. Finally, for the translation gaits (Figure 4.3b)), green lines and Figure 4.3d), green box), the swimmer's orientation is aligned with the target direction ($\theta_d \approx 0$); the swimmer repeatedly extends and contracts L_1 and L_2, while keeping θ_{31} close to π (*i.e.*, all three spheres of the swimmer are aligned), resembling the swimming gaits of Najafi-Golestanian swimmers [40, 71]. In the configuration space (Figure 4.3a)), the translation gaits reside largely in the L_1-L_2 plane with an approximately zero average θ_{31}, generating the maximum net translation with minimal net rotation (Figure 4.3c)). The details of gaits categorization are summarized under Supplementary methods.

The swimming gaits can again be separated into three gaits: steering, transition, and translation. For the steering gait, the swimmer prioritizes on rotation and only actuates its right arm and intermediate angle θ_{31}. Its left arm stays fully contracted. For the transition gait, the swimmer utilizes all mechanical devices and performs periodical actuation. The configuration space tilts in the upward direction. For

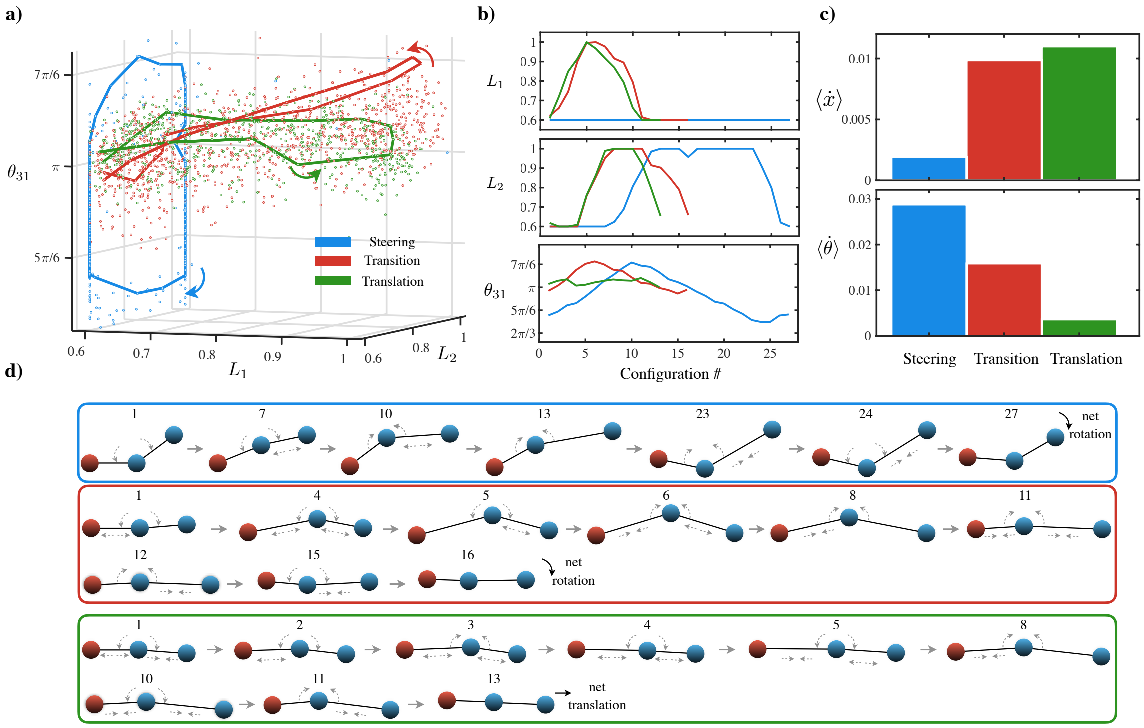


Figure 4.3 Analysis of configurational changes revealing three distinct modes of locomotory gaits. The steering, transition, and translation gaits are marked as blue, red, green, respectively. a) A 3D configuration plot for a typical simulation which the swimmer aligns with the target direction via a counterclockwise rotation, where L_1, L_2 are the arm lengths and θ_{31} is the intermediate angle. Each dot represents one specific configuration of a locomotory gait. The solid lines mark an example cycle of each locomotory gait. b) The changes in the arm lengths L_1 and L_2 and the intermediate angle θ_{31} with respect to the configuration number for each locomotory gait. c) The average translational velocity $\langle\dot{x}\rangle$ and rotational velocity $\langle\dot{\theta}\rangle$ are calculated by averaging the centroid translation along the target direction θ_T and the change of swimmer's orientation θ_o over the total number of action steps for each locomotory gaits. d) Representative configurations labelled with the configuration number are displayed to illustrate the configurational changes for each selected sequence of locomotory gaits for the steering (blue box), transition (red box), and translation (green box) modes. The leftmost sphere of the swimmer is marked as red and other two spheres are marked as blue to indicate the swimmer's current orientation. The grey arrows indicate the contraction/extension of the arms and the intermediate angle. For illustration, the reference frame of the configurations are rotated consistently such that the left arm of the first configuration is aligned horizontally in each sequence.

the translation gait, the swimmer focuses more on the arm actuation and limits its angle rotation. The geometric configuration for the clockwise rotation can be roughly viewed as a reflection around the $\hat{\mathbf{r}}_{21}$ axis of the counterclockwise rotation for the steering and transition gaits.

It is noteworthy that the multi-modal navigation strategy emerges solely from the AI without relying on prior knowledge of locomotion. The switching between rotation, transition, and translation gaits is analogous to the switching between turning and running modes observed in bacterial locomotion [12, 52]. These results demonstrate how an AI-powered swimmer, without being explicitly programmed, self-learns complex locomotory gaits from rich action and configuration spaces and undergoes autonomous gait switching in accomplishing targeted navigation.

4.3.3 Gaits Categorization

Through observing simulation results, we notice distinct locomotory gaits based on the orientation of the swimmer relative to the target direction. For the purpose of gait analysis, we define each gaits below: The steering gait is mainly used when the target direction is oriented far away from the swimmer's orientation: $|\theta_T - \theta_o| > \pi/2$. The transition gait gets activated once the swimmer's orientation reaches around $\pi/2$ of the target: $|\theta_T - \theta_o| \leq \pi/2$. The translation gait occurs after the swimmer first roughly aligns with the target orientation: $|\theta_T - \theta_o| \leq 5\pi/180$.

4.3.4 Performance Evaluation.

Here we investigate the improvement of swimmer's performance with increased number of training episodes N_e. At initial stage of training with a small N_e, the swimmer may fail to identify the right sets of locomotory gaits to achieve targeted navigation due to insufficient training. Continuous training with increased number of episodes would enable the swimmer to identify better locomotory gaits to complete

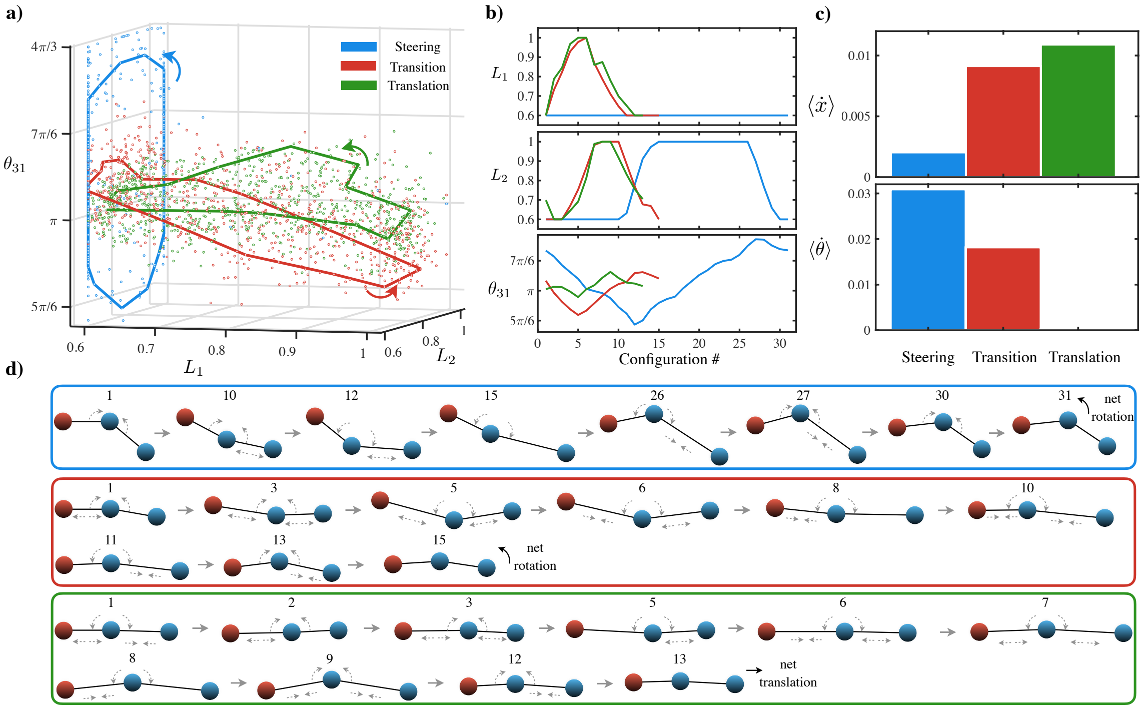


Figure 4.4 Analysis of configurational changes revealing three distinct modes of locomotory gaits. The steering, transition, and translation gaits are marked as blue, red, green, respectively. (a) A 3D configuration plot for a typical simulation which the swimmer aligns with the target direction via a clockwise rotation (b) The changes in the arm lengths L_1 and L_2 and the intermediate angle θ_{31} with respect to the configuration number for each locomotory gait. (c) The average translational velocity $\langle \dot{x} \rangle$ and rotational velocity $\langle \dot{\theta} \rangle$ are calculated by averaging the centroid translation along the target direction θ_T and the change of swimmer's orientation θ_o over the total number of action steps for each locomotory gaits. (d) Representative configurations labelled with the configuration number are displayed to illustrate the configurational changes for each selected sequence of locomotory gaits for the steering (blue box), transition (red box), and translation (green box) modes.

navigation tasks. Here we measure the improvement of swimmer's performance with increased N_e by three locomotion tests: (1) Random target test: the swimmer is assigned a target direction selected randomly from a uniform distribution in $[0, 2\pi]$; (2) Rotation test: the swimmer is assigned a targeted direction with a large angle of difference with swimmer's orientation (*i.e.*, $\theta_d = \pm\pi/2$); (3) Translation test: the swimmer is assigned a target direction equal to the swimmer's orientation (*i.e.*, $\theta_d = 0$). A test is considered to be successful if the swimmer travels along the target direction for a distance of 5 *unit* in 10000 action steps. These tests ensure that the trained swimmer acquires a set of effective locomotory gaits to swim along any specified direction with robust rotation and translation.

We consider the success rates of the three tests over 100 trials (Figure 4.5). For $N_e = 3 \times 10^4$, success rates of around 90% are obtained for the three tests. When N_e is increased to 9×10^4, the swimmer masters translation with a 100% success rate but still needs more training for rotation. When N_e is increased further to 15×10^4, the swimmer obtains 100% success rates for all tests. This result demonstrates the continuous improvement in the robustness of targeted navigation with increased N_e up to 15×10^4. As we further increase N_e, we found the relationship between N_e and performance to be non-monotonic. For a total training episodes much greater than $N_e = 15 \times 10^4$, the overall success rate will begin to drop and eventually fluctuate around 95%. We selected the trained result at $N_e = 15 \times 10^4$ for the best overall performance.

To better understand the swimmer's training process, we also varied the number of steps in each episodes, N_l. For a range from 100 to 300 and a fixed total episodes N_e, we found $N_l = 150$ provides the most efficient way to balance translation and rotation and require least amount of action steps to complete both the rotation and translation tests. We remark that, when $N_l = 100$, the swimmer was only able to translate but not to rotate, indicating the significant role N_l plays in learning.

Lastly, we remark that the swimmer appears to require more training, both in N_e and N_l, to learn rotation compared to translation. This may be attributed to the inherit complexity of rotation gaits, where the swimmer needs to actuate its intermediate angle in addition to the actuation of the two arms required in translation gaits.

4.3.5 Path Tracing–"SWIM".

Next we showcase the swimmer's capability in tracing complex paths in an autonomous manner. To illustrate, the swimmer is tasked to trace out the English word "SWIM" (Figure 4.6). We note that the hydrodynamic calculations required to design the locomotory gaits to trace such complex paths become quickly intractable as the complexity increases. Here, instead of explicitly programming the gaits of the swimmer, we only select target points ($\mathbf{p}_i$, $i = 1, 2, ..., 17$, red spots in Figure 4.6) as landmarks and require the swimmer to navigate towards these landmarks with its own AI, with the target directions at action step $t+1$ given by $\theta_{T_{t+1}} = \arg(\mathbf{p}_i - \mathbf{r}_{c_t})$. The swimmer is assigned with the next target point $\mathbf{p}_{i+1}$ when its centroid is within a certain threshold (0.1 of the fully extended arm length) from $\mathbf{p}_i$. The completion of these multiple navigation tasks sequentially enables the swimmer to successfully trace out the word "SWIM" with a high accuracy (Figure 4.6). In accomplishing this task, the swimmer switches between the three modes of locomotory gaits autonomously to swim towards individual target points and turn around the corners of the path based on the AI-powered navigation strategy. It is noteworthy that the swimmer is able to navigate around some corners (*e.g.*, at target points 4 and 6) without activating the steering gaits, which are employed for corners with more acute angles (*e.g.*, at target points 8, 14, and 16). While past approaches based on detailed hydrodynamic calculations, manual interventions, or other control methods may also complete such tasks, here we present reinforcement learning as an alternative approach in accomplishing these complex maneuvers in a more autonomous manner.

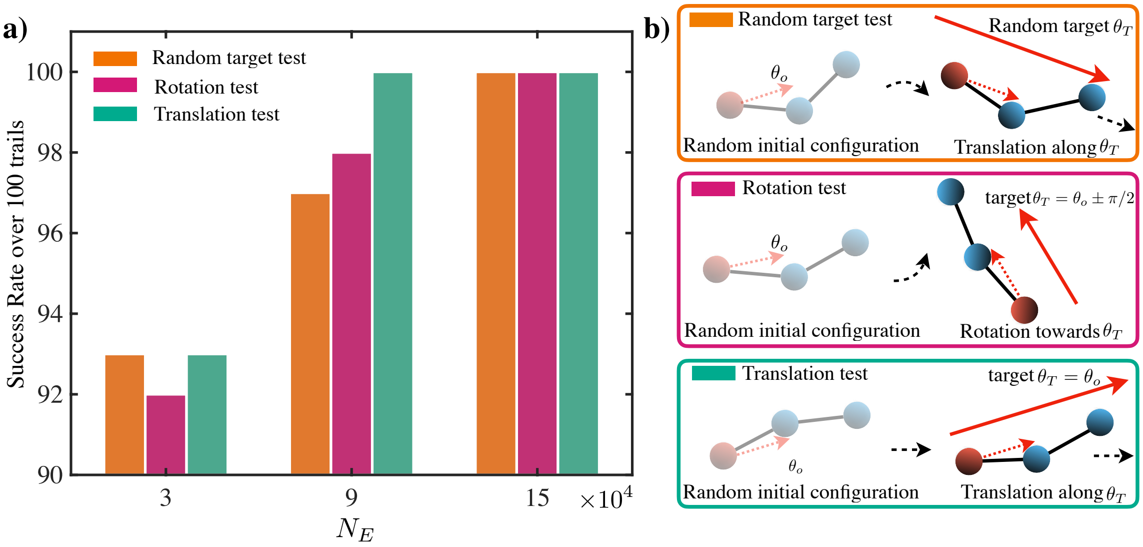


Figure 4.5 Analysis of the swimmer's performance with increasing number of episodes. Number of episodes N_e indicates the total training time of the swimmer. Each episode during training contains a fixed amount of action steps $N_l = 150$. a) We used three tests (random target test, rotation test and translation test) to measure the swimmer's performance in a fixed number of training steps $N_l = 150$. For all tests, the swimmer starts with a random initial configuration to ensure a full exploration of the observation space. A total of 100 trials are considered for each test with swimmers trained at different N_e. A swimmer with insufficient training (3×10^4 episodes) may occasionally fails in the three tests (success rate $\approx 90\%$). At $N_e = 9 \times 10^4$, the swimmer masters translation and improves its rotation ability. When N_e increases to 1.5×10^5, the swimmer obtains a 100% success rate in all tests. b) Schematics of the random target test, rotation test, and translation test. The leftmost sphere is marked as red and other spheres are marked as blue to indicate the swimmer's orientation θ_o (red dashed arrows). Given a random initial configuration, we test the swimmer's ability to translate along or rotate towards a target direction θ_T (solid red arrows). The black dashed arrows indicate the swimmer's intended moving direction.

4.3.6 Robustness Against Flows.

Simulations of Background Flow We consider the motion of a swimmer under the influence of a background flow due to a rotlet at the origin

$$\mathbf{u}^*_\infty(\mathbf{r}^*) = -\frac{\boldsymbol{\gamma}^* \times \mathbf{r}^*}{r^{*3}}, \tag{4.13}$$

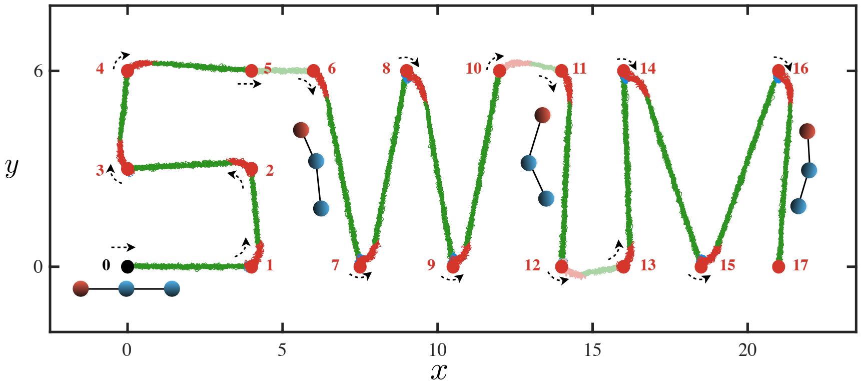


Figure 4.6 Demonstration of complex navigation capability of Artificial Intelligence powered swimmer. The Artificial Intelligence powered swimmer switches between various locomotory gaits autonomously in tracing a complex trajectory "SWIM". The trajectory of the central sphere of the swimmer is colored based on the mode locomotory gaits: steering (blue), transition (red), and translation (green). The swimmer is given a lists of target points (1-17) with one target point at a time. The black arrows at each point indicate the intended direction of the swimmer. From the current target point, the swimmer determines the target direction for the next action step $t+1$, $\theta_{T_{t+1}}$ and adapts the locomotory gaits based on its AI in navigating towards that direction. Schematics of the swimmer configurations (not-to-scale) are shown for illustrative purposes, where the leftmost sphere is marked as red and other two spheres are marked as blue to indicate the swimmer's current orientation.

where $\boldsymbol{\gamma}^* = \gamma^* \mathbf{e}_z$ prescribes the dimensionless rotlet strength γ^* in the z-direction. To account for the background flow $\mathbf{u}^*_\infty(\mathbf{r}^*)$, the mobility relation is adjusted with the hydrodynamic forces and torques on the spheres given by [23]

$$\mathbf{F}^*_i = \sum_{j=1}^{3} \mathbf{M}^*_{ij} \cdot \mathbf{V}^*_j - 6\pi R^* \mathbf{u}^*_\infty(\mathbf{r}^*_i) + \sum_{j=1, i\neq j}^{3} 36\pi^2 R^{*2} \mathbf{G}^*_{ij} \cdot \mathbf{u}^*_\infty(\mathbf{r}^*_i), \tag{4.14}$$

$$\mathbf{T}^*_i = \mathbf{r}^*_i \times \mathbf{F}^*_i - 4\pi R^{*3} \nabla^* \times \mathbf{u}^*_\infty(\mathbf{r}^*_i) + \sum_{j=1, i\neq j}^{3} 24\pi^2 R^{*4} \nabla^* \times \mathbf{G}^*_{ij} \cdot \mathbf{u}^*_\infty(\mathbf{r}^*_i), \tag{4.15}$$

where $\mathbf{M}^*_{ij}$ is the inverse of $\mathbf{G}^*_{ij}$ and $\mathbf{r}^*_i$ denotes the position of sphere i.

Last, we examine the performance of targeted navigation under the influence of flows (Figure 4.7a),b)). In particular, to determine to what extent the AI-powered swimmer is capable of maintaining its target direction against flow perturbations, we use the same AI-powered swimmer trained without any background flow, and impose a rotational flow generated by a rotlet at the origin [44, 56], $\mathbf{u}_\infty = -\boldsymbol{\gamma} \times \mathbf{r}/r^3$, where $\boldsymbol{\gamma} = \gamma \mathbf{e}_z$ prescribes the strength of the rotlet in the z-direction, $r = |\mathbf{r}|$ is the magnitude of the position vector $\mathbf{r}$ from the origin (see Simulations of background flow under Supplementary methods). Here the AI-powered swimmer is tasked to navigate towards the positive x-direction under flow perturbations due to the rotlet. We examine how the swimmer adapts to the background flow when performing this task. For comparison, we contrast the resulting motion of the AI-powered swimmer with that of an untrained swimmer (*i.e.,* a Najafi-Golestanian (NG) swimmer that performs only fixed locomotory gaits without any adaptivity [71]). Without the background flow, both swimmers self-propel with the same speed. Both swimmers are initially placed close to the rotlet with $\mathbf{r}_c = -5\mathbf{e}_x$ and we sample their performance with three different initial orientations: $\theta_{o_0} = -\pi/3$, 0, and $\pi/3$, under different flow strengths. Under a relatively weak flow ($\gamma = 0.15$, Figure 4.7a)), the AI-powered swimmer is capable of navigating towards the positive x-direction regardless of its initial orientations against flow perturbations. In contrast, the trajectories of the NG swimmer are largely influenced by the rotlet flow passively depending on the initial orientation of the swimmer. For an increased flow strength ($\gamma = 1.5$, Figure 4.7b)), the NG swimmer completely loses control of its direction and is scattered by the rotlet into different directions again due to the absence of any adaptivity. Under such a strong flow, the AI-powered swimmer initially circulates around the rotlet but eventually manages to escape from it, navigating to the positive x-direction successfully with similar trajectories for all initial orientations. We note that the vorticity experienced by the swimmer in this case is comparable with typical re-orientation rates of the AI-powered swimmer. We also remark that when navigating under flow perturbations, the AI-powered swimmer adopts the transition gaits to constantly re-orient itself towards the positive x-direction and self-propels along that direction eventually. These results showcase the AI-powered swimmer's capability in adapting its locomotory gaits to navigate robustly against flows.

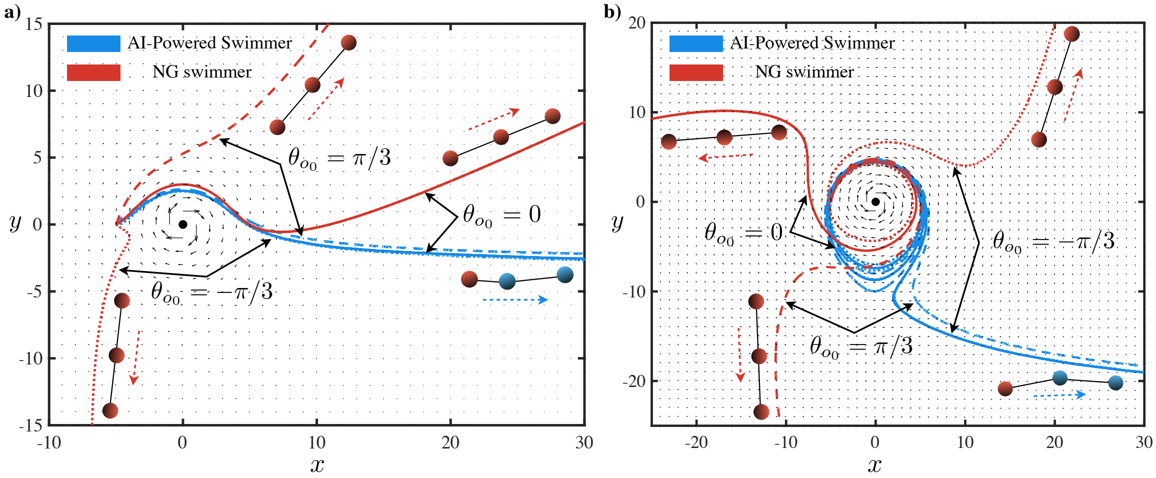


Figure 4.7 Analysis of the performance of targeted navigation under the influence of flows. a) The Artificial Intelligence powered swimmer and the Najafi-Golestanian (NG) swimmer escape from a relatively weak rotlet flow, $\mathbf{u}_\infty = -\boldsymbol{\gamma} \times \mathbf{r}/r^3$, where $\boldsymbol{\gamma} = \gamma \mathbf{e}_z$ prescribes the strength of the rotlet in the z-direction, $r = |\mathbf{r}|$ is the magnitude of the position vector $\mathbf{r}$ from the origin ($\gamma = 0.15$). The leftmost sphere of the AI-powered swimmer is marked as red and other spheres are marked as blue to indicate the swimmer's current orientation (blue dashed arrow). The NG swimmer is colored red with its orientation marked as red dashed arrows. Three sets of trajectories (dashed, dotted, and solid lines) are shown with different initial swimmer orientation θ_{o_0}. The AI-powered swimmer travels to the right regardless of its initial orientation whereas the trajectory for the NG swimmer is highly affected by the rotlet flow. b) We compare the trajectories of the AI-powered swimmer and the NG swimmer in a strong rotlet flow ($\gamma = 1.5$). The NG swimmer completely loses control in the flow, while the AI-powered swimmer maintains its orientation towards the positive x-direction, with similar trajectories for different initial orientations.

4.4 Concluding Remarks

In this work, we present a deep RL approach to enable navigation of an artificial microswimmer via gait switching advised by the AI. In contrast to previous works that considered active particles with prescribed self-propelling velocities as minimal models [18, 70, 85] or simple one-dimensional swimmers [45, 64, 99], here we demonstrate how a reconfigurable system can learn complex locomotory gaits from rich and continuous action spaces to perform sophisticated maneuvers. Through RL, the swimmer develops distinct locomotory gaits for a multimodal (*i.e.*, steering, transition, and translation) navigation strategy. The AI-powered swimmer can adapt its locomotory gaits in an autonomous manner to navigate towards any arbitrary directions. Furthermore, we show that the swimmer can navigate robustly under the influence of flows and trace convoluted paths. Instead of explicitly programming a swimmer to perform these tasks in the traditional approach, the swimmer is advised by the AI to perform complex locomotory gaits and autonomous gait switching in accomplishing these navigation tasks. The multimodal strategy employed by the AI-powered swimmer is reminiscent of the run-and-tumble in bacteria [12, 52]. Taken together, our results showcase the vast potential of this deep RL approach in realizing adaptivity similar to that of biological organisms for robust locomotive capabilities. Such adaptive behaviors are crucial for future biomedical applications of artificial microswimmers in complex media with uncontrolled and/or unpredictable environmental factors.

We finally discuss several possibilities for subsequent investigations based on this deep RL approach. While we demonstrate only planar motion in this work, the approach can be readily extended to three-dimensional navigation by allowing out-of-plane rotation the swimmer's arms with expanded observation and action spaces for the additional degrees of freedom. Moreover, the deep RL framework is not tied to any specific swimmers; a simple multi-sphere system is used in this work for illustration, and the same framework applies to other reconfigurable systems. We also remark that the AI-powered swimmer is able to overcome some influences of flows even though such flows were absent in the training. Subsequent investigations including the flow perturbation in the training may lead to even more powerful AI that could exploit the flows to further enhance the navigation strategies. Another practical aspect to consider is the effect of Brownian noise [26, 46]. Specifically, the characterization of the effect of thermal fluctuations in both the training process of the swimmer and its resulting navigation performance is currently underway. In addition to flow and thermal fluctuations, other environmental factors, including the presence of physical boundaries and obstacles, may be addressed in similar manners in future studies. The deep RL approach here opens an alternative path towards designing adaptive microswimmers with robust locomotive and navigation capabilities in more complex, realistic environments.

CHAPTER 5

LINEAR 3-SPHERE DEVICE IN A NON-STATIONARY ENVIRONMENT VIA REINFORCEMENT LEARNING CONTEXT DETECTION

5.1 Background and Related Work

Swimming microorganisms live in a world where viscous force dominates. In this low Reynolds number limit, due to their small scale, inertia effects become negligible comparing to its viscous counterpart. Under such stringent constraints, microorganisms have developed many versatile swimming strategies including using flagella rotary motors [11, 62] and molecular motors [84]. Motivated by potential biomedical and environmental interest [34, 74], recent studies have focused on developing artificial microswimmers in complex environments.

Purcell's leading work illustrates the challenges of developing microswimmers from the perspective of low Reynolds number fluid dynamics [79]. Many subsequent studies have demonstrated how a simple system can generate net locomotion [71] or mechanical rotation [25]. However, the design of complex systems with sophisticated maneuvers soon become intractable. More recently, reinforcement learning techniques have been used in studies of various aspects of fluid locomotion problems [17,97], including three sphere micro-swimmer locomotion [45,64,99], active particle navigation [2, 19, 85], flow navigation [18, 43], and fish and bird like motions [37,38,54,82].

5.2 Non-Stationary Environment Formulation

We consider a simple system composed of three identical spheres of radius R connected by two arms with variable arm length L_1, L_2 (Figure 5.1 bottom). This system generalizes the one-dimensional swimmer proposed by Najafi and Golestanian [71] by

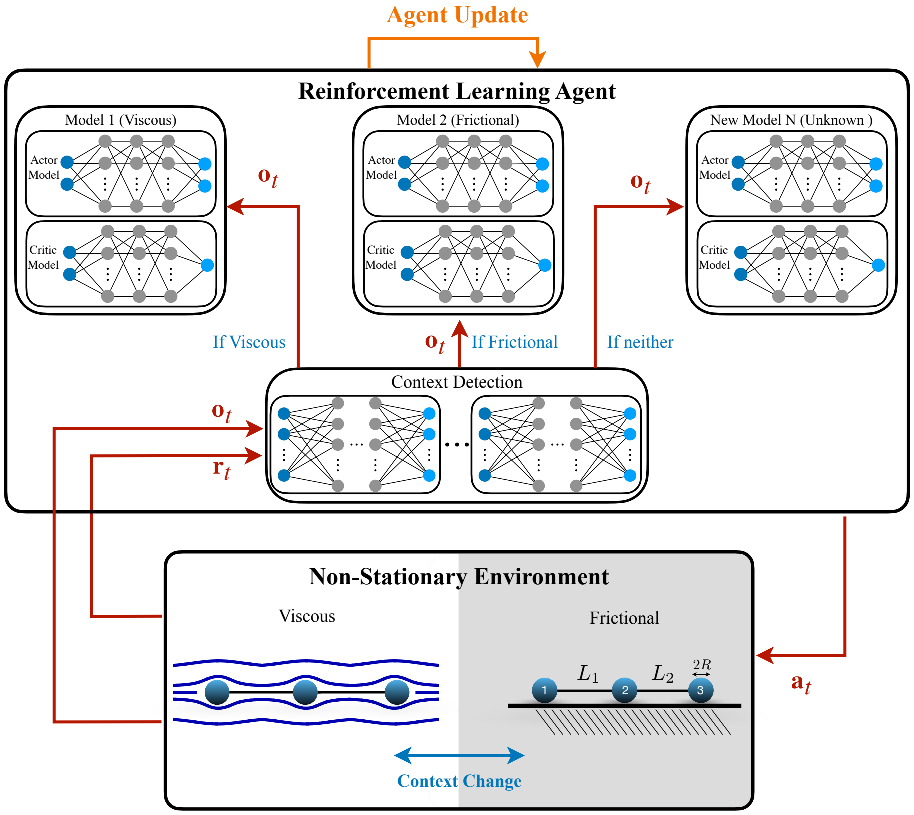


Figure 5.1 Schematic of the Reinforcement Learning Context Detection algorithm. The swimmer performs the given action and interacts with the non-stationary environment (bottom), where the current environment is unknown to the swimmer. Reward and observation is calculated based on the net centroid displacement and geometric configuration change. Receiving the reward and the next observation, the reinforcement learning agent utilizes its context detection method to determine which model best fits the current environment and advises the next action to the swimmer (Top). During training, the agent periodically updates its policy models and context detection method.

allowing continuous arm actuation. The dynamics of sphere interaction is described by a non-stationary environment consisting of two distinct environments: (1) *Viscous* environment. The interactions between the spheres and the surrounding fluid is governed by the Stokes equation. In this low Reynolds number regime, the viscous force dominates. (2) *Frictional* environment. The 3-sphere device is placed upon a horizontal frictional surface, where its movement is determined by the arm driving force and the isotropic Coulomb friction. We model the non-stationary environment by constantly switching from one environment to another, where the change of environment happens instantaneously in the eyes of the device. The instantaneous environment change can be considered as the limiting case of the concentration signal sensing problem [66], where the concentration signal stays constant until a sudden jump due to the change of environment. The only difference is that our device has no direct access to the concentration signal. We now describe the environment dynamics below.

5.2.1 Low Reynolds Number Hydrodynamics

We consider a low Reynolds number environment (*Viscous*), where the interaction between the device and the surrounding newtonian fluids is governed by the Stokes equation ($\nabla p = \mu \nabla^2 \mathbf{u}$), where p, μ, $\mathbf{u}$ are the pressure, dynamic viscosity and velocity field, respectively. Due to the linearity of the Stokes equation, we write down the velocity and force coupling in the x direction as:

$$V_{x_i} = \mathbf{G}_{ij} F_{x_j} \tag{5.1}$$

where $\mathbf{G}_{ij}$ is the one-dimension Oseen tensor [23, 44, 56] given by

$$\mathbf{G}_{ij} = \begin{cases} \frac{1}{6\pi\mu R}, \\ \\ \frac{1}{4\pi\mu|r_{x_i} - r_{x_j}|}. \end{cases} \tag{5.2}$$

Here, r_{x_i} is the x position of the sphere i, and $|r_{x_i} - r_{x_j}|$ is distance between spheres i and j. We then express constant arm actuation rate $\dot{L}_i$ in terms of the sphere's horizontal velocities V_{x_i}:

$$V_{x_2} - V_{x_1} = \dot{L}_1 \tag{5.3}$$

$$V_{x_3} - V_{x_2} = \dot{L}_2 \tag{5.4}$$

Applying the force free condition $\sum_i F_{x_i} = 0$, the swimmer's kinematic is fully determined. We constrain the arm length ($0.6L \leq L_1, L_2 \leq L$) to ensure the validity of the Oseen tensor approximation ($R \ll L$). In this work, we scale the length by fully extended arm length L and the velocity by a characteristic arm actuation velocity V_c. This results a time scale of $T = L/V_c$. For the Viscous environment, the forces are scaled by $\mu L V_c$.

5.2.2 Dry Friction Dynamics

Consider the same one dimension three sphere device placed on a horizontal flat frictional surface (*Frictional* environment), and denote the sphere mass m, acceleration $\ddot{x}$, and the frictional coefficient μ_f. We can write down the equations of motions of each spheres by accounting the driving forces $\mathbf{F}_{d_i}$ induced by actuation of arm L_i on sphere i and the corresponding friction forces $\mathbf{F}_{f_i}$:

$$\ddot{x}_i = \frac{1}{m}(\mathbf{F}_{d_i} - \mathbf{F}_{d_{i-1}} + \mathbf{F}_{f_i}) \tag{5.5}$$

with $\mathbf{F}_{d_0} = \mathbf{F}_{d_3} = \mathbf{0}$. The friction forces on each sphere is written as:

$$\mathbf{F}_{f_i} = \begin{cases} -F_c(\dot{x}_i), & \text{if } |\mathbf{F}_{d_i} - \mathbf{F}_{d_{i-1}}| > F_c \\ -(\mathbf{F}_{d_i} - \mathbf{F}_{d_{i-1}}), & \text{if } |\mathbf{F}_{d_i} - \mathbf{F}_{d_{i-1}}| \leq F_c \end{cases} \tag{5.6}$$

where $F_c = \mu_f mg$ is the magnitude of the sliding friction exerted by the surface on the sphere; a static friction equals to the sliding friction is assumed. We further assume each arm applies only the minimum driving forces needed to satisfy the arm actuation condition for constant rate at each action step, resulting the acceleration on each sphere $\ddot{x}_i = 0$. We note that under our assumption at least one sphere will stay stationary during the actuation period. Summing over equation [5.5] for all spheres i, we have $\sum_i \mathbf{F}_{f_i} = 0$. We can then determine the driving forces along the x direction $F_{d_{x_i}}$ based on the values of $\dot{L}_i$:

$$F_{d_{x_1}}, F_{d_{x_2}} = \begin{cases} F_c, F_c & \text{if } \dot{L}_1, \dot{L}_2 < 0 \\ F_c, 0 & \text{if } \dot{L}_1 < 0, \dot{L}_2 = 0 \\ 0, F_c & \text{if } \dot{L}_1 = 0, \dot{L}_2 < 0 \\ -F_c, -F_c & \text{if } \dot{L}_1, \dot{L}_2 > 0 \\ -F_c, 0 & \text{if } \dot{L}_1 > 0, \dot{L}_2 = 0 \\ 0, -F_c & \text{if } \dot{L}_1 = 0, \dot{L}_2 > 0 \\ 0, F_c & \text{if } \dot{L}_1 \geq 0, \dot{L}_2 \leq 0, |\dot{L}_1| < |\dot{L}_2| \\ -F_c, 0 & \text{if } \dot{L}_1 \geq 0, \dot{L}_2 \leq 0, |\dot{L}_1| > |\dot{L}_2| \\ -F_c/2, F_c/2 & \text{if } \dot{L}_1 \geq 0, \dot{L}_2 \leq 0, |\dot{L}_1| = |\dot{L}_2| \\ 0, -F_c & \text{if } \dot{L}_1 < 0, \dot{L}_2 > 0, |\dot{L}_1| < |\dot{L}_2| \\ F_c, 0 & \text{if } \dot{L}_1 < 0, \dot{L}_2 > 0, |\dot{L}_1| > |\dot{L}_2| \\ F_c/2, -F_c/2 & \text{if } \dot{L}_1 < 0, \dot{L}_2 > 0, |\dot{L}_1| = |\dot{L}_2| \end{cases} \tag{5.7}$$

This is equivalent as determining the sphere that are made to translate during an actuation. We similarly scale the length by L, velocity by V_c, and forces by mg.

5.2.3 Reinforcement Learning Context Detection

We first focus on a deep RL approach [45, 54, 88] training the swimmer in a single environment to produce positive net locomotion in the x direction. In the RL

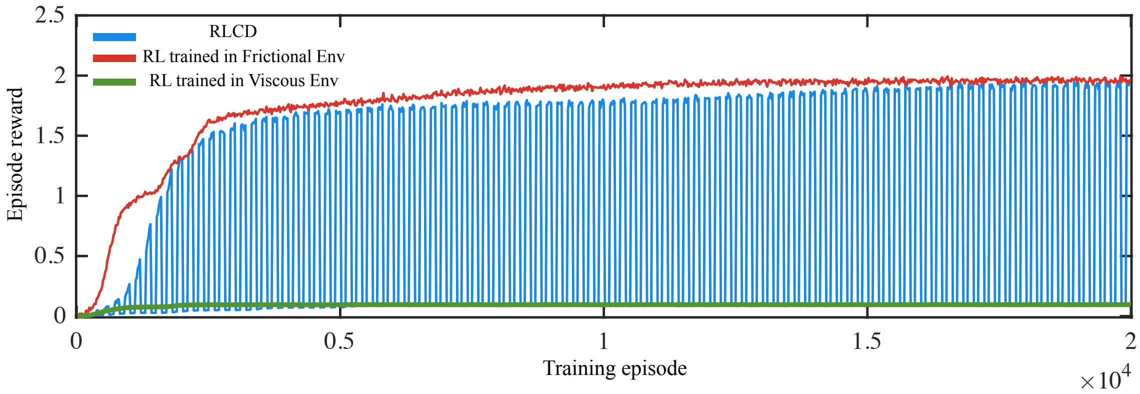


Figure 5.2 Episode reward increases as the total training episode increases. We plot the average episode reward across five agents and compare the training performance of RLCD (blue), RL trained in frictional (red) and viscous (green) environment. Because of the simple setup of our system, RL agents trained in frictional and viscous environment learn effective policies quickly. With sufficient training, RLCD, trained under a non-stationary environment, reaches approximately the same performance in both environment.

algorithm, the state $s \in (r_{x_1}, r_{x_2}, r_{x_3})$ contains all the x positions of the spheres. The observation $o \in (L_1, L_2)$ is extracted from the state as geometric configurations. The RL agent determines the next action based on the current observation through the Actor neural network. The swimmer then performs the action $a \in (\dot{L}_1, \dot{L}_2)$ by actuating both of its arms for the duration of one action step, $\Delta t = 0.1$. The RL agent evaluates the success of the action by measuring the net centroid displacement: $r_t = r_{c_{t+1}} - r_{c_t}$, where the centroid $r_c = \sum_i r_{x_i}$. The training process is divided into N_e total episodes, with each episode containing $N_t = 100$ action steps. We randomly initialize a state s_0 at the beginning each episode to ensure full exploration of the observation space. The Actor and Critic network is further updated for every 20 episodes by maximizing the expected long-term rewards $E[R_{t=0}|\pi_\phi]$. Here, π_ϕ is the stochastic control policy, $R_t = \sum_{t'}^{\infty} \gamma^{t'-t} r_{t'}$ is the infinite-horizon discounted future returns, and γ is the discount factor measuring the greediness of the algorithm. We set $\gamma = 0.99$ ensuring the farsightedness of the RL algorithm.

The RL framework described above has shown to be effective for generating locomotory gaits for a single environment [54]. To adapt this RL framework to the non-stationary environment, a straightforward approach is applying it on each individual environment with a context change detection algorithm. We therefore employ a recent high confidence change point detection method developed by Alegre *et al* [3] and abbreviate this combination generally as "RLCD" (not to be confused with the "RL-CD" algorithm [21]). Here, the term "context" indicates a type of dynamics in a specific environment; hence, a context change corresponds to a change in the environment in which the device is immersed. RLCD enables the agent to quickly detect a change of environment and train a set of partial models, each specialized for different environment dynamics (Figure 5.1).

Consider the non-stationary environment consisting of a list of environments $E_1 ... E_K$. Let C be a random environment change point switching from E_i to E_j. A proper detection method should consider two things: i) time it takes to detect the change point C, ii) false detection before C occurs. RLCD minimizes both by computing quality signals based on the experience (i.e., the action performed, state transition, and reward) of the device. The quality signals, $W_{k,t} = \max(0, W_{k,t-1} + L_{k,t})$, are computed for each partial model k at every action step t utilizing a multivariate variant of cumulative sum (MCUSUM) method [3], where $L_{k,t}$ is the log-likelihood ratio indicating how likely a particular model k becomes a better fit than the current model. The algorithm will activate the partial model with the highest quality signal that surpasses threshold h and thereby enable the device to adapt its locomotory gaits in response to the change of environment. Furthermore, in the context detection algorithm, a new partial model will be generated when all other partial models become ineffective in describing the current environment, allowing the swimmer to explore unlimited distinct environments (Figure 5.1 top).

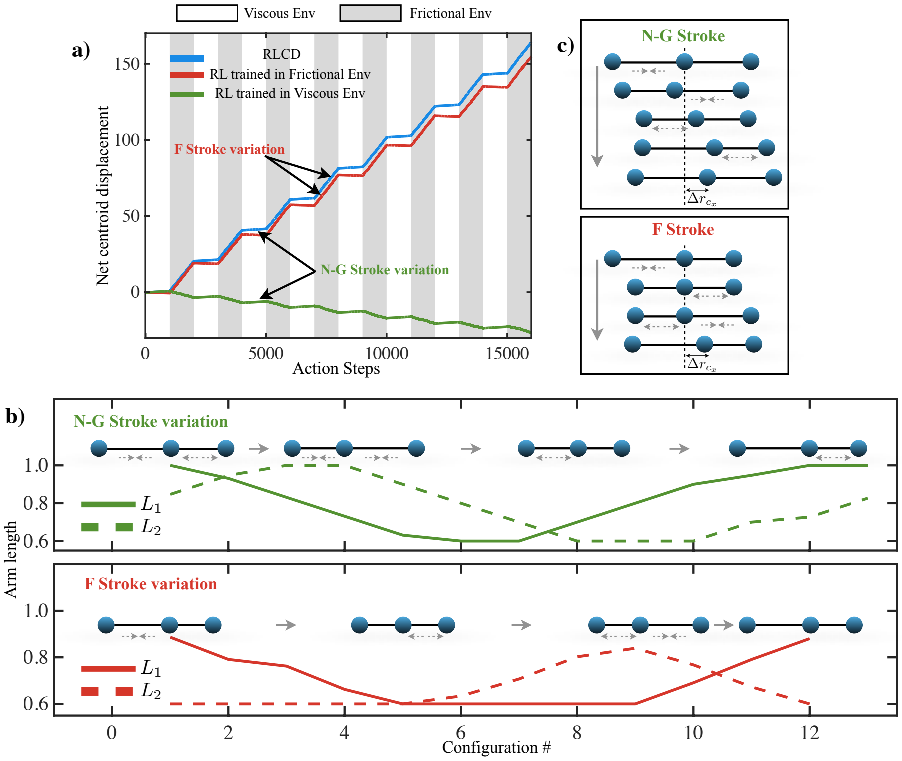


Figure 5.3 Analysis the performance of RLCD (blue), RL module trained in frictional (red) and viscous (green) environment in a non-stationary environment. a) The non-stationary environment periodically switches from viscous (white) to frictional (gray) environment. We test each agent's ability to translate to the right. Both RL trained in viscous and frictional environment performs well in their trained environment, but fails to translate in the foreign environment. On the other hand, RLCD masters both environment and detects a context change quickly, resulting the most net centroid displacement. b) A sample cycle of N-G stroke and F stroke variations are plotted alongside with geometric configurations. The grey dashed arrows indicate the arm actuation. c) Stroke sequence for the discrete N-G stroke and F stroke. We mark the net centroid displacement after each cycle as Δr_{c_x}.

In Figure 5.2, we illustrate an average training result of RLCD agents in a non-stationary environment (blue) with RL agents trained separately in viscous (green) and frictional (red) environment. In the training process, RLCD agent gradually builds up experience of the current environment. The experience is then used to improve both its ability to accurately detect an environment change and develops the locomotory strategies of the two partial models. Here, we periodically switch between viscous and frictional environment every 100 episodes, corresponding to the frequent oscillation shown in Figure 5.2 blue. We observe RLCD quickly acquires effective locomotory strategies around 6×10^3 episodes for the viscous environment (green).

Given sufficient time (roughly 2×10^4 episodes), RLCD learns a set of locomotory gaits that generate similar rewards as the RL agents in both environment as well as detecting a context change. We remark that the environment change detection usually takes around 3 to 7 action steps, a very minimum time comparing to its total training action steps, 2×10^6.

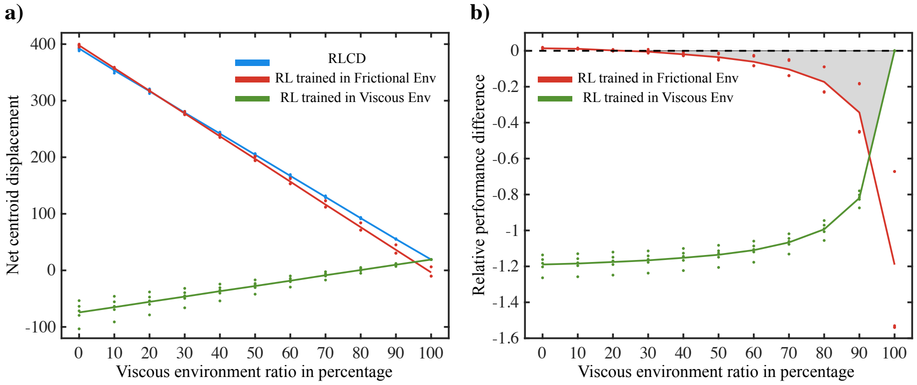


Figure 5.4 We evaluate the performance of RLCD (blue) and RL trained in viscous (green) and frictional (red) across various ratio of the amount of time stays in viscous environment over the total simulation. a) We plot the average net centroid displacement as solid lines and each agent's displacement as dots, with a total of 5 agents for each algorithm. At the viscous ratio of 20 percent and above, RLCD starts to outperform all other algorithms. RLCD also has the most consistent result with a smaller range across the 5 agents. b) We use RLCD's average result as a benchmark to examine the relative performance of other RL agents. A negative relative difference indicates a worse performance than RLCD. For the most of the scenario, RL agents have a relative difference below 0 indicating RLCD's superior performance.

5.3 Results and Discussion

5.3.1 Performance Comparison and Effective Locomotory Policies

We next compare our RLCD agents with RL agents trained in the viscous or frictional environment by placing them in a non-stationary environment. We illustrate their performance difference by showing their net centroid displacement over a set amount of action steps (Figure 5.3a)). We simulate the non-stationary environment by periodically switching between the viscous (white shaded area) and frictional (grey shaded area) environment for every 1000 action steps. We remark that even though the environment is switched periodically, no agents have information on when the switch would occur. In addition, since our context detection only takes around three to seven action steps, a randomly generated context change on the same scale wouldn't significantly affect the performance. Overall, the RLCD agents (blue) demonstrate their superior performance by properly detecting the change of the environment and utilizing the corresponding gait. In contrast, both RL agents trained in viscous (green) and frictional (red) environment perform negatively in the other environment as indicated by the slopes of the curves. We also observe a large performance discrepancy between the RL agent trained in viscous and frictional environment due to the inherent reward difference of the two environments. This reward difference can be more clearly observed in Figure 5.2. Here, the effectiveness of one particular RL agent greatly depends on the environment it was trained in. We now visualize the actuation policies RLCD agent used in the viscous and frictional environment by plotting the dimensionless arm length of a representative stroke sequence. For the viscous environment (Figure 5.3b) green), RLCD utilizes a policy similar to a discrete case optimal strategy studied by Najafi and Golestanian [71] (Figure 5.3c) top). For the frictional environment (Figure 5.3b) red), RLCD develops an entirely different policy identified as a variation of the F-stroke [29] (Figure 5.3c) bottom). This example demonstrates RLCD's ability to acquire effective locomotory gaits in distinct environment and successfully navigate non-stationary environment.

5.3.2 Performance across Various Non-Stationary Envrionment

Last, we examine the RLCD's performance by varying the viscous environment ratio in a non-stationary environment. We define the ratio as the amount time device spends in the viscous environment over the total time of the simulation. Figure 5.3a) represents a case of viscous environment ratio equals 50 in percentage. (A ratio of 0% or 100% indicate a pure frictional or viscous environment, respectively). Furthermore, we model the non-stationary environment by periodically switching between viscous and frictional environment for a total of 19 times in Figure 5.4a). Here, we illustrate the performance of RLCD (blue) and RL (red and green) agents by plotting the average net centroid displacement (solid lines) with individual agents' displacement (dots). RLCD agents begin to outperform the other RL agents as the viscous ratio rises above 20%. We observe, interestingly, a monotonic increase in the spread of the RL agents (dots) as the ratio moves towards a foreign environment to the agents. This happens because even though all RL agents have reached an almost identical policy in their trained environment, the slight difference between each agent is greatly amplified in a foreign environment, which impacts the consistency of RL agents in a non-stationary environment. In contrast, RLCD always selects the proper policy for the corresponding environment and makes the overall spread almost indistinguishable. In Figure 5.4 b), we set the average RLCD net centroid displacement as a baseline and compare the relative performance difference of the other RL agents. A relative difference of 0 indicating the same performance as RLCD, whereas a negative value signifies a worse performance. We shade the region where RLCD demonstrates superior performance over the RL agents for better illustration; we observe clearly that RLCD outperforms the other RL agents for viscous environment ratio between 20% to 100%. Those results showcase both the consistent performance and the wide range effectiveness of the RLCD algorithm in a non-stationary environment.

5.4 Concluding Remarks

In this work, we present a context change detection method in combination with RL algorithm allowing for developing effective locomotory gaits in distinct environments. In contrast to previous works that utilize RL for a single environment [64, 99], we focus on the device's ability to navigate non-stationary environments without prior knowledge of when an environment change would occur. We demonstrate that the RLCD enables the device to master specialized locomotory gaits for each environment and detect a context change quickly. In presenting the result, we leave out the comparison of RL agent trained in a non-stationary environment, which is observed to perform worse than RL agent trained in the frictional environment. Last, we briefly comment on the algorithm's limitation. This context change detection method assumes that the individual environments are distinct, meaning either the effective locomotory strategies (observation transition function) or the reward (function) is significantly different. For an environment with very similar observation transition and reward function, the detection mechanism very likely would fail. In such case, a RL agent might be sufficiently effective in the considered environment. Taken together, our result demonstrates both robustness and superior performance over a wide range of viscous environment ratio when comparing to the conventional RL agent trained in a single environment.

We now discuss several possible future investigations based on this context detection and deep RL approach. In this work, we primarily limit ourselves to two distinct environments each required relatively "little" training, while our approach can be employed with "infinite" numbers of distinct environments. A subsequent investigation would focus on designing a complex non-stationary environment involving more environments and advanced navigation capability. Moreover, we assume an instantaneous environment change, which can be considered as a limiting case of a chemical sensing problem. A future work would incorporate the transition phase between two environments in order to provide a complete picture of navigating the non-stationary environment. The context detection and deep RL approach present here offers a new avenue for designing artificial devices in navigating complex fast changing environments.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This dissertation addresses four interrelated problems in in the design of artificial micro-swimmers: train a Stokesian swimmer via Q-learning, mechanical rotation in low Reynolds number fluids, gait switching and targeted navigation of microswimmers in Stokes flow and linear three-sphere device in a non-stationary environment. First, this dissertation presents Q-learning method to generate one dimensional translation in the low Reynolds number regime. Second, this dissertation shows the Q-learning method can be applied to generate net rotation in the low Reynolds number fluids. Third, this dissertation presents deep reinforcement learning combined with proximal policy optimization method for two-dimensional swimmers that is able to navigate to the target. Finally, this dissertation presents a generative adversarial network, named reinforcement learning context detection, for generating effective policies in the non-stationary environment.

This dissertation develops new reinforcement learning methods and deep neural network to help develop the effective gaits for the artificial microswimmer. In the future work, the intrinsic curiosity module with the advanced deep reinforcement learning methods will be investigated and extended to more swimmer related tasks, such as artificial swimmers in biomedical and environmental applications. The curiosity driven method will be developed based on the all types of environments, where those rewards will be collected based on a inverse and forward model. The efficiency of the curiosity driven method will be studied and used to improve the performance of the training process.

APPENDIX A

SUPPLEMENTARY METHODS

This appendix summarizes the numerical results of the three-sphere rotator.

A.1 Three-sphere Rotator

A.1.1 Problem Setup for Three-sphere Rotator

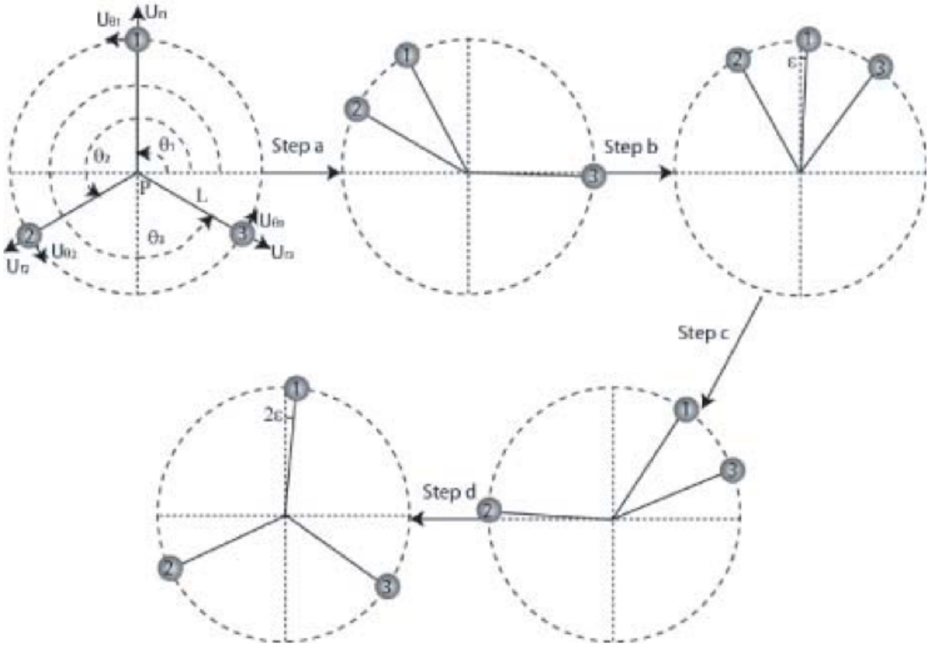


Figure A.1 Complete 4-step cycle of the proposed non-reciprocal motion of a rotational motor. The device experiences a net rotation after completion of a cycle [24].

The complete cycle of rotation can be divided into four steps with no translational motion allowed:

1. Step a: fix the angle between spheres 1 and 3, reduce the angle between spheres 1 and 2 with a constant angular velocity, the angular position of each sphere is define as $\theta_1, \theta_2, \theta_3$.

2. Step b: fix the angle between spheres 1 and 2, reduce the angle between spheres 1 and 3 with the same constant angular velocity ω.

3. Step c: fix the angle between spheres 1 and 3, increase the angle between spheres 1 and 2 with the same constant angular velocity ω to $2\pi/3$.

4. Step d: fix the angle between spheres 1 and 2 ($2\pi/3$), increase the angle between spheres 1 and 3 with the same constant angular velocity ω to $2\pi/3$, now the rotator is in the original configuration.

5. compute the net angular displacement 2ε after the four steps: $2\varepsilon = \sum_{i=a}^{d} \theta_{1i}$.

A.1.2 Simplification of the Torque Exert on Each Sphere

At low Reynolds number, the governing equations are the Stokes equation and the incompressibility condition. Since both equations are linear, and if we denote by $\vec{r_i}$ the position vector of sphere i measured from the center P , then there is a linear tensor relation between the forces acting on each sphere $\vec{F_i}$ and their velocity $\vec{V_i}$ of the form,

$$\vec{F_i} = \sum_{j=1}^{3} H_{ij} \vec{V_j}, \text{ with } \begin{cases} H_{ii} = -6\pi\eta R \mathbf{I} \\ H_{ij} = -6\pi\eta R \frac{3R}{4R_{ij}} (\mathbf{I} + \hat{R}_{ij}\hat{R}_{ij}) \\ \hat{R}_{ij} = \frac{\vec{r_i} - \vec{r_j}}{\vec{r_i} - \vec{r_j}} \\ R_{ij} = \vec{r_i} - \vec{r_j} \end{cases} \tag{A.1}$$

where $\mathbf{I}$ is the identity tensor and η is the viscosity of the fluid. These equations account for the leading-order hydrodynamic influences and for interactions among each of the spheres treated as point forces ($R/L \ll 1$). If we denote $\vec{D_i}$ as the position of the center of sphere i measured from a fixed point in the laboratory frame, an additional linear tensor relation between the torques acting on each sphere $\vec{\Gamma}_i$ and the velocities of each sphere can be derived in the form:

$$\vec{\Gamma}_i = \vec{D}_i \wedge \sum_{j=1}^{3} H_{ij} \vec{V_j}. \tag{A.2}$$

Since no external force is present, the system is force-free and torque-free:

$$\sum_{i=1}^{3} \vec{F_i} = \vec{0} \text{ and } \sum_{i=1}^{3} \vec{\Gamma}_i = \vec{0}. \tag{A.3}$$

The system was parameterized as follows: a local cylindrical coordinates $\vec{U}_{r_i}, \vec{U}_{\theta_i}$ is defined for each sphere, so that the velocity of each sphere in the laboratory frame is given by

$$\vec{V_i} = \vec{V_p} + L\dot{\theta}_i \vec{U}_{\theta_i}, \tag{A.4}$$

where $\vec{V_p}$ is the translation velocity of the centre P. During each step of the cycle, two other constraints are added; for example during the first step we set

$$\dot{\theta}_2 - \dot{\theta}_1 = -\omega, \tag{A.5}$$

and

$$\dot{\theta}_1 = \dot{\theta}_3. \tag{A.6}$$

To simplify expression of the torque exert on each sphere:

1. let θ_i be the angular displacement of sphere i.

2. let $\vec{R_i}$ be the position of sphere i in Cartesian coordinates (points from P to center of sphere i)

$$\vec{R_i} = L \begin{vmatrix} -\sin(\theta_i)\dot{\theta}_i \\ \cos(\theta_i)\dot{\theta}_i \end{vmatrix},$$

with $\dot{\theta}_i = \theta_i t$.

3. the point-to-point distance between two spheres is,

$$R_{ij} = 2L\sin(\frac{\theta_i - \theta_j}{2}).$$

4. the unit vector between two spheres is,

$$\hat{R_{ij}} = \frac{\vec{R_{ij}}}{R_{ij}} = \frac{1}{2\sin(\frac{\theta_i - \theta_j}{2})} \begin{vmatrix} \cos(\theta_i) - \cos(\theta_j) \\ \sin(\theta_i) - \sin(\theta_j) \end{vmatrix}.$$

5. the force $\vec{F_{ij}}$ exerts on each sphere i by sphere j is,

$$\vec{F_{ij}} = H_{ij}\vec{v_j},$$

with

$$H_{ij} = -6\pi\eta R\frac{3R}{4R_{ij}}(I + \hat{R_{ij}}\hat{R_{ij}}),$$

then

$$\vec{F_{ij}} = -6\pi\eta R, \frac{3R}{8L\sin(\frac{\theta_i-\theta_j}{2})}(\vec{v_j} + \hat{R_{ij}}(\hat{R_{ij}}\cdot\vec{v_j})),$$

$\vec{v_j} + \hat{R_{ij}}(\hat{R_{ij}}\cdot\vec{v_j})$ can be expressed as follows:

$$\begin{aligned} &L\dot{\theta}_j(\begin{vmatrix}-\sin(\theta_j)\\ \cos(\theta_j)\end{vmatrix} + \frac{\sin(\theta_i)\cos(\theta_j)-\cos(\theta_i)\sin(\theta_j)}{4\sin(\frac{\theta_i-\theta_j}{2})^2}\begin{vmatrix}\cos(\theta_i)-\cos(\theta_j)\\ \sin(\theta_i)-\sin(\theta_j)\end{vmatrix}) \\ &= L\dot{\theta}_j(\begin{vmatrix}-\sin(\theta_j)\\ \cos(\theta_j)\end{vmatrix} + \frac{\sin(\theta_i-\theta_j)}{4\sin(\frac{\theta_i-\theta_j}{2})^2}\begin{vmatrix}\cos(\theta_i)-\cos(\theta_j)\\ \sin(\theta_i)-\sin(\theta_j)\end{vmatrix}) \end{aligned}$$

therefore

$$\begin{aligned} \vec{F_{ij}} &= -6\pi\eta R\frac{3R}{8\sin(\frac{\theta_i-\theta_j}{2})}\dot{\theta}_j(\begin{vmatrix}-\sin(\theta_j)\\ \cos(\theta_j)\end{vmatrix} + \frac{\sin(\theta_i-\theta_j)}{4\sin(\frac{\theta_i-\theta_j}{2})^2}\begin{vmatrix}\cos(\theta_i)-\cos(\theta_j)\\ \sin(\theta_i)-\sin(\theta_j)\end{vmatrix}) \\ &= -6\pi\eta R\frac{3R}{8\sin(\frac{\theta_i-\theta_j}{2})}\dot{\theta}_j[\begin{vmatrix}-\sin(\theta_j)+\frac{1}{2}\cot(\frac{\theta_i-\theta_j}{2})(\cos(\theta_i)-\cos(\theta_j))\\ \cos(\theta_j)+\frac{1}{2}\cot(\frac{\theta_i-\theta_j}{2})(\sin(\theta_i)-\sin(\theta_j))\end{vmatrix}] \end{aligned}$$

with $\frac{\sin(\theta_i-\theta_j)}{4\sin(\frac{\theta_i-\theta_j}{2})^2}$ can be simplified as

$$\frac{\sin(\theta_i-\theta_j)}{4\sin(\frac{\theta_i-\theta_j}{2})^2} = \frac{2\sin(\frac{\theta_i-\theta_j}{2})\cos(\frac{\theta_i-\theta_j}{2})}{4\sin(\frac{\theta_i-\theta_j}{2})^2} = \frac{1}{2}\cot(\frac{\theta_i-\theta_j}{2})$$

$$\vec{F_{ii}} = -6\pi\eta RL\dot{\theta}_j\begin{vmatrix}-\sin(\theta_j)\\ \cos(\theta_j)\end{vmatrix}$$

6. the system is torque-free, compute the torque of each sphere (pointing inward or outward the x-y plane in the z-axis)

$$\vec{\Gamma_i} = \vec{R_i}\vec{F_i}$$

$$
\begin{aligned}
\vec{R}_i\vec{F}_i &= \frac{9\pi\eta LR^2\dot{\theta}_j}{4\sin(\frac{\theta_i-\theta_j}{2})}[-\cos(\theta_i-\theta_j)-\frac{1}{2}\frac{\cos(\frac{\theta_i-\theta_j}{2})}{\sin(\frac{\theta_i-\theta_j}{2})}2\sin(\frac{\theta_i-\theta_j}{2})\cos(\frac{\theta_i-\theta_j}{2})] \\
&= \frac{9\pi\eta LR^2\dot{\theta}_j}{4\sin(\frac{\theta_i-\theta_j}{2})}[-1+2\sin(\frac{\theta_i-\theta_j}{2})^2-\cos(\frac{\theta_i-\theta_j}{2})^2] \\
&= \frac{9\pi\eta LR^2\dot{\theta}_j}{4\sin(\frac{\theta_i-\theta_j}{2})}[3\sin(\frac{\theta_i-\theta_j}{2})^2-2]
\end{aligned}
$$

Thus, the governing equations and torque-free constraint can be expressed as follows:

$$\Gamma_1 = -6\pi\eta L^2R\dot{\theta}_1 + \frac{9\pi\eta LR^2(3\sin(\frac{\theta_1-\theta_2}{2})^2-2)}{4\sin(\frac{\theta_1-\theta_2}{2})}\dot{\theta}_2 + \frac{9\pi\eta LR^2(3\sin(\frac{\theta_1-\theta_3}{2})^2-2)}{4\sin(\frac{\theta_1-\theta_3}{2})}\dot{\theta}_3 \tag{A.7}$$

$$\Gamma_2 = \frac{9\pi\eta LR^2(3\sin(\frac{\theta_1-\theta_2}{2})^2-2)}{4\sin(\frac{\theta_1-\theta_2}{2})}\dot{\theta}_1 - 6\pi\eta L^2R\dot{\theta}_2 + \frac{9\pi\eta LR^2(3\sin(\frac{\theta_2-\theta_3}{2})^2-2)}{4\sin(\frac{\theta_2-\theta_3}{2})}\dot{\theta}_3 \tag{A.8}$$

$$\Gamma_3 = \frac{9\pi\eta LR^2(3\sin(\frac{\theta_1-\theta_3}{2})^2-2)}{4\sin(\frac{\theta_1-\theta_3}{2})}\dot{\theta}_1 + \frac{9\pi\eta LR^2(3\sin(\frac{\theta_2-\theta_3}{2})^2-2)}{4\sin(\frac{\theta_2-\theta_3}{2})}\dot{\theta}_2 - 6\pi\eta L^2R\dot{\theta}_3 \tag{A.9}$$

$$0 = \Gamma_1 + \Gamma_2 + \Gamma_3 \tag{A.10}$$

with

1. Γ_i is torque in z-direction on sphere i;
2. L is the length from P to center of sphere i in the x-y plane;
3. η is the fluid viscosity;
4. R is the radius of each sphere;
5. $\dot{\theta}_i$ is the counterclockwise angular velocity of sphere i;
6. θ_i is the angular position of sphere i;
7. Θ is the internal angular change.

APPENDIX B

ADDITIONAL NUMERICAL RESULTS

Figure B.1 shows the total angle of rotation of the Purcell's rotator during a complete four-stroke cycle as a function of the internal angular change for different values of the ratio $\frac{R}{L}$ when the device cannot have translational motion. The results indicate the net rotational motion depends significantly on the ratio of $\frac{R}{L}$.

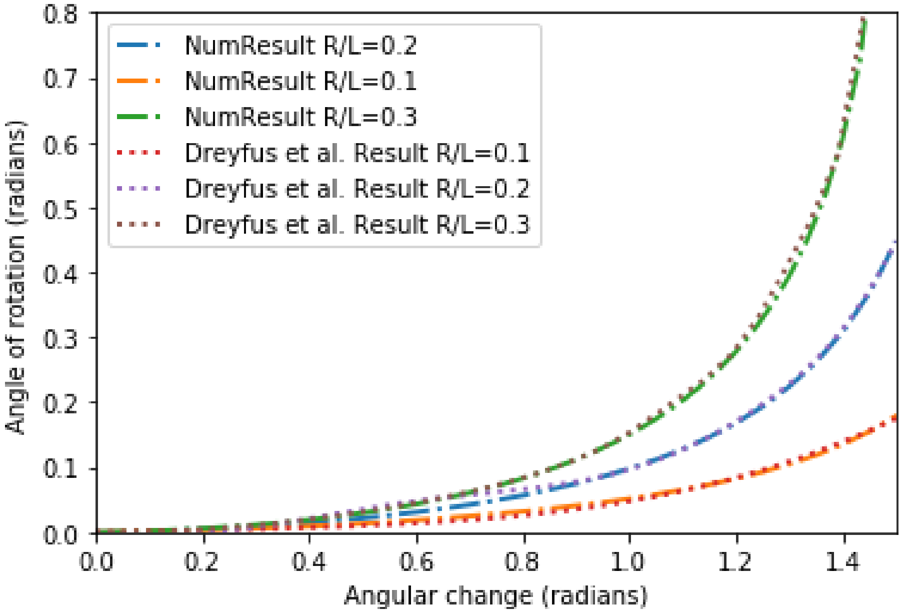


Figure B.1 Total angle of rotation as a function of the internal angular change when the centre P is fixed (no translational motion). The different curves correspond to different ratios of R/L, the dotted lines correspond to Fig. 2. in Dreyfus *et al.* paper, the dash-dotted lines correspond to my numerical results for $R/L = 0.1, 0.2, 0.3$. The graph shows our numerical results are consistent with the results presented in Dreyfus's paper.

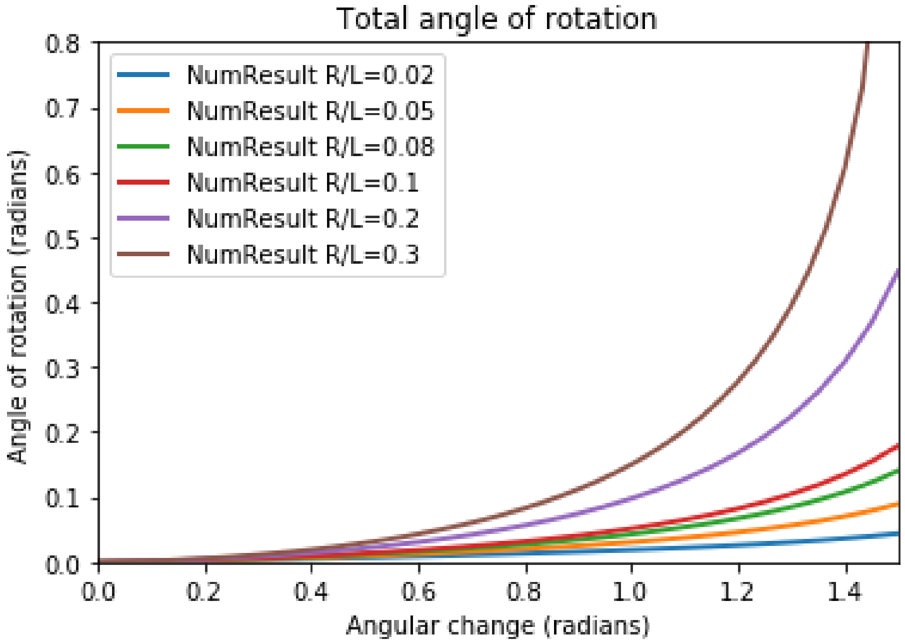


Figure B.2 Total angle of rotation as a function of the internal angular change when the centre P is fixed (no translational motion). The different curves correspond to the ratio of $R/L = 0.02, 0.05, 0.08, 0.1, 0.2, 0.3$, the result indicates the net rotational motion depend significantly on the ratio of $\frac{R}{L}$.

Q-Learning Results for Self-learning Three-sphere Rotator

Problem setup

1. Goal: learn how to act by interacting with the fluid environment;

2. Agent(the 3-sphere rotator): the rotator consists of three spheres of radius R placed on an imaginary circle, each sphere is connected to a rod of length L and the rods are connected at the centre P of the circle;

3. State(s_n): configuration of the rotator, 3-sphere rotator has 4 configurations and each configuration can transition to 2 configurations by moving the spheres closer or further apart;

4. Action(a_n): enlarge/contract angle between spheres;

5. Reward(r_n): certain rotation angle of the angle centroid of the rotator(c_n);

$$r_n = \hat{e}\Delta c_n$$

(a) $\hat{e}$: desired direction (set clockwise as the positive direction);

(b) angle centroid of the swimmer is defined as : $c_n := \frac{\sum_{i=1}^{3} a_i(n)}{3}$, e.g. at state 1: set the angle of sphere 1 as 0 and clockwise as the positive direction:

$$\theta_1 = 0 \text{ rad}$$
$$\theta_2 = -2.0944 \text{ rad}$$
$$\theta_3 = 0.5944 \text{ rad}$$
$$c_1 = \frac{\theta_1 + \theta_2 + \theta_3}{3} = -0.5 \text{ rad}$$

(c) $\theta_i(n)$: angle position of i-th sphere;

(d) Δc_n: rotation between states displaces c_n, $\Delta c_n = c_{n+1} - c_n$

6. D: cumulative angular displacement of the sphere centroid

$$D = \sum_n \hat{e} \Delta c_n$$

update $Q(s_n, a_n)$

$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha[R_n + \gamma \max Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)] \quad \text{(B.1)}$$

with $\alpha = 1, \gamma = 0.8, \epsilon = 0.05$

To implement the algorithm, let's start by recollecting the states define earlier and map each of the states to numbers:

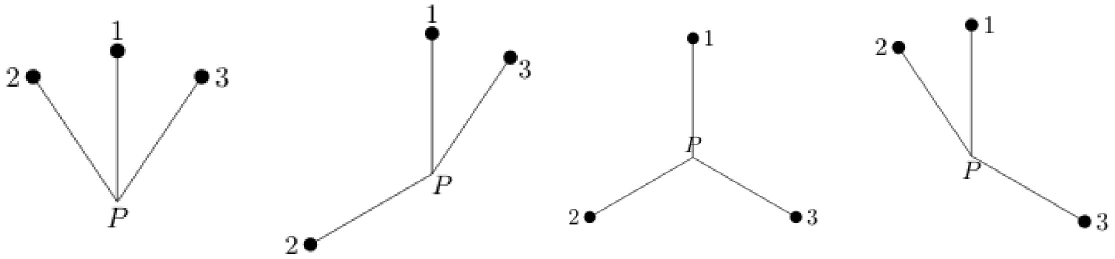


Figure B.3 From left to right are configurations of Purcell's rotator from state 0 to state 3 respectively.

The three-sphere rotator has four different configurations. In each training step, the device transit from one configuration to another and updates the corresponding entry in the Q-matrix according to Equation (B.1). Figure B.4 depicts a typical self-learning process: the rotator initially struggles to rotate in the clockwise direction thus rotate clockwise and counterclockwise with D remains close to 0. The device continues exploring the surrounding environment by performing different actions and adapting its propulsion policy. After accumulating enough experiences, the device develops an effective rotation policy that repeats the same sequence of actions except with ϵ probability when random action is chosen and rotate with increasing D. The rotation gaits obtained by Q-learning algorithm for the Purcell's rotator is consistent with Dreyfus's device [24].

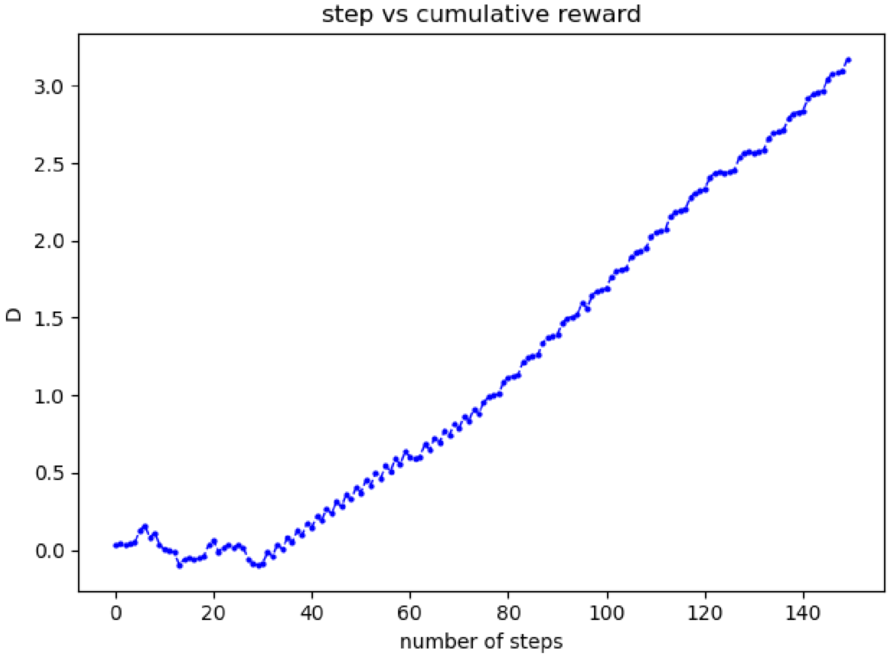


Figure B.4 A typical learning process of a self-learning three-sphere rotator as translation not allowed. The learning outcome is consistent with Dreyfus *et al.*'s device.

Sudden change in the Q-entry verification Increase the number of learning steps and check the evolution of the differences of entries in the Q-matrix and check the evolution of each Q-entry. According to the update scheme of Q-learning and deterministic reward matrix(R),

$$R = \left[\begin{array}{lcccc} & action_0 & action_1 & action_2 & action_3 \\ state_0 & -Inf & -0.010681 & 0.010681 & -Inf \\ state_1 & 0.010681 & -Inf & -Inf & -0.001486 \\ state_2 & -0.010681 & -Inf & -Inf & 0.001486 \\ state_3 & -Inf & 0.001486 & -0.001486 & -Inf \end{array}\right] \qquad \text{(B.2)}$$

Figures B.5 and B.6 show when the learning step is greater than 230, the difference of entries of each row in the Q-matrix remains the same, Figure 19 is the plot of each Q-entry evolves over learning step, the learning step of sudden change is the same as the Q-entry difference. If we consider 0.01 as the immediate reward, for the corresponding state, the action with 0.01 reward is always preferred and gets updated, thus the cumulative future reward can be expressed as 0.01 +0.01 +0.01 +..., which cause the sudden change in the Q-entry when apply the $\epsilon-$greedy scheme, the less preferred action with lower reward is performed and lead to the sudden change in the Q-entry as shown in Figure B.6.

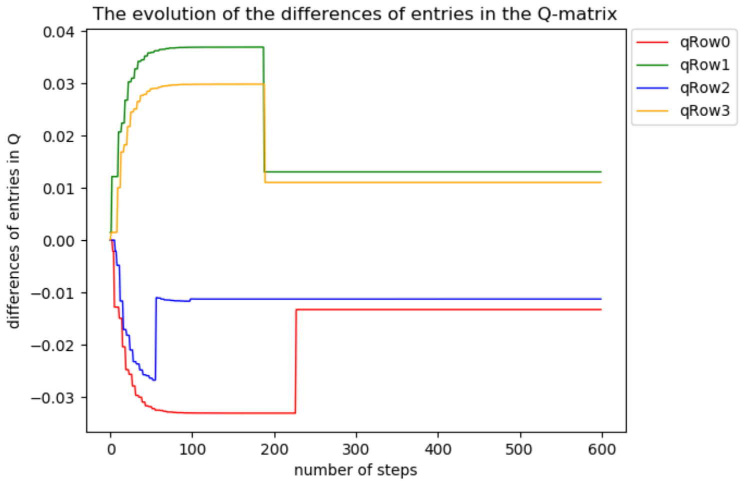


Figure B.5 The evolution of the differences of entries in the Q-matrix.

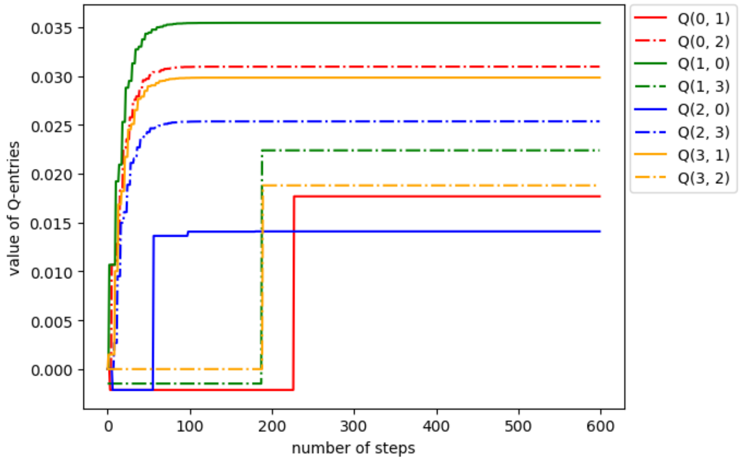


Figure B.6 The evolution of the Q-entries in the Q-matrix.

The total number of learning step is 600 with $\gamma = 0.8$, $\epsilon = 0.05$. To verify the sudden change of difference in the entries of Q-matrix based on the immediate reward matrix:

1. for Q[0:](row 0 in the Q-matrix), to check the magnitude of the sudden change: check the change of Q-entry at training step 77, Q[0, 1] change from -0.002 to 0.018; Q[0, 2] remains the same as 0.031; the magnitude of the sudden change is 0.020; the typical scale of the reward is from approximately 0.01 to 0.001, according to the update scheme of Q-learning, if we use 0.01 as the immediate reward, the cumulative future reward can be expressed as $0.01 + \gamma 0.01 + \gamma 0.01 + ...$, thus, the sudden change in the Q-entry is in the range of future reward. This reason also holds for the following conditions;

2. Q[1:](row 1 in the Q-matrix), to check the magnitude of the sudden change: check the change of Q-entry at training step 156, Q[1, 3] change from 0 to 0.022, Q[1, 0] remains the same as 0.035, the magnitude of the sudden change is 0.022;

3. for Q[2:](row 2 in the Q-matrix) , to check the magnitude of the sudden change: check the change of Q-entry at training step 234, Q[2, 0] change from 0 to 0.014, Q[2, 3] remains the same as 0.025, the magnitude of the sudden change is 0.014;

4. for Q[3:](row 3 in the Q-matrix), to check the magnitude of the sudden change: check the change of Q-entry at training step 105, Q[3, 1] change from 0 to 0.018, Q[3, 2] remains the same as 0.030, the magnitude of the sudden change is 0.018.

REFERENCES

[1] D. Agostinelli, F. Alouges, and A. DeSimone. Peristaltic waves as optimal gaits in metameric bio-inspired robots. *Frontiers in Robotics and AI*, 5:99, 2018.

[2] J. K. Alageshan, A. K. Verma, J. Bec, and R. Pandit. Machine learning strategies for path-planning microswimmers in turbulent flows. *Physical Review E*, 101:043110, Apr 2020.

[3] L. N. Alegre, A. L. C. Bazzan, and B. C. da Silva. Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, pages 97–105, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[4] F. Alouges, A. DeSimone, L. Giraldi, Y. Or, and O. Wiezel. Energy-optimal strokes for multi-link microswimmers: Purcell's loops and Taylor's waves reconciled. *New Journal of Physics*, 21(4):043050, Apr 2019.

[5] F. Alouges, A. DeSimone, L. Giraldi, and M. Zoppello. Self-propulsion of slender micro-swimmers by curvature control: N-link swimmers. *International Journal of Non-Linear Mechanics*, 56:132–141, 2013.

[6] F. Alouges, A. DeSimone, L. Heltai, A. Lefebvre-Lepot, and B. Merlet. Optimally swimming Stokesian robots. *Discrete and Continuous Dynamical Systems - Series B*, 18:1189, 2012.

[7] F. Alouges, A. DeSimone, and A. Lefebvre. Optimal strokes for low Reynolds number swimmers: An example. *Journal of Nonlinear Science*, 18(3):277–302, Jun 2008.

[8] J. E. Avron, O. Kenneth, and D. H. Oaknin. Pushmepullyou: an efficient microswimmer. *New Journal of Physics*, 7(1):234, 2005.

[9] O. R. Barclay. The mechanics of amphibian locomotion. *Journal of Experimental Biology.*, 23(2):177–203, 1946.

[10] L. E. Becker, S. A. Koehler, and H. A. Stone. On self-propulsion of micro-machines at low Reynolds number: Purcell's three-link swimmer. *Journal of Fluid Mechanics*, 490:15–35, 2003.

[11] H. C. Berg. The rotary motor of bacterial flagella. *Annual Review of Biochemistry*, 72(1):19–54, 2003.

[12] H. C Berg and D. A. Brown. Chemotaxis in escherichia coli analysed by three-dimensional tracking. *Nature*, 239(5374):500–504, 1972.

[13] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, and K. Gustavsson. Zermelo's problem: Optimal point-to-point navigation in 2D turbulent flows using reinforcement learning. *Chaos*, 29(10):103138, 2019.

[14] J. P. Celli, B. S. Turner, N. H. Afdhal, S. Keates, I. Ghiran, C. P. Kelly, R. H. Ewoldt, G. H. McKinley, P. So, S. Erramilli, and R. Bansil. Helicobacter pylori moves through mucus by reducing mucin viscoelasticity. *Proceedings of the National Academy of Sciences*, 106(34):14321–14326, 2009.

[15] H. Ceylan, I. C. Yasa, O. Yasa, A. F. Tabak, J. Giltinan, and M. Sitti. 3D-Printed biodegradable microswimmer for drug delivery and targeted cell labeling. *bioRxiv*, 2018.

[16] U. K. Cheang, F. Meshkati, H. Kim, K. Lee, H. C. Fu, and M. J. Kim. Versatile microrobotics using simple modular subunits. *Scientific Reports*, 6:30472, 2016.

[17] F. Cichos, K. Gustavsson, B. Mehlig, and G. Volpe. Machine learning for active matter. *Nature Machine Intelligence*, 2(2):94–103, 2020.

[18] S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale. Flow navigation by smart microswimmers via reinforcement learning. *Physical Review Letters*, 118:158004, Apr 2017.

[19] S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale. Smart inertial particles. *Physical Review Fluids*, 3:084301, Aug 2018.

[20] B. Colvert, M. Alsalman, and E. Kanso. Classifying vortex wakes using neural networks. *Bioinspiration Biomimetics*, 13:025003, 2018.

[21] B. C. da Silva, E. W. Basso, A. L. C. Bazzan, and P. M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 217–224, New York, NY, USA, 2006. Association for Computing Machinery.

[22] B. Dai, J. Wang, Z. Xiong, X. Zhan, W. Dai, C.-C. Li, S.-P. Feng, and J. Tang. Programmable artificial phototactic microswimmer. *Nature Nanotechnology*, 11(12):1087–1092, 2016.

[23] J. K. G. Dhont. *An introduction to dynamics of colloids*. Amsterdam, Netherlands: Elsevier, 1996.

[24] R. Dreyfus, J. Baudry, M. L. Roper, M. Fermigier, H. A. Stone, and J. Bibette. Microscopic artificial swimmers. *Nature*, 437(7060):862–865, 2005.

[25] R. Dreyfus, J. Baudry, and H. A. Stone. Purcell's "rotator": mechanical rotation at low Reynolds number. *European Physical Journal B*, 47(1):161–164, Sep 2005.

[26] J. Dunkel and I. M. Zaid. Noisy swimming at low reynolds numbers. *Physical Review E*, 80:021903, Aug 2009.

[27] D. J. Earl, C. M. Pooley, J. F. Ryder, I. Bredberg, and J. M. Yeomans. Modeling microscopic swimmers at low Reynolds number. *Journal of Chemical Physics*, 126:064703, 2007.

[28] S. J. Ebbens and J. R. Howse. In pursuit of propulsion at the nanoscale. *Soft Matter*, 6:726–738, 2010.

[29] B. Elder, Z. Zou, S. Ghosh, O. Silverberg, T. E. Greenwood, E. Demir, V. S.-E. Su, O. S. Pak, and Y. L. Kong. A 3D-Printed self-learning three-linked-sphere robot for autonomous confined-space navigation. *Advanced Intelligent Systems*, 3(9):2100039, 2021.

[30] J. Elgeti, R. G. Winkler, and G. Gompper. Physics of microswimmers âÂâÂ– single particle motion and collective behavior: a review. *Reports on Progress in Physics*, 78(5):056601, 2015.

[31] L. J. Fauci and R. Dillon. Biofluidmechanics of reproduction. *Annual Review of Fluid Mechanics*, 38:371–394, 2006.

[32] W. Gao, D. Kagan, O. S. Pak, C. Clawson, S. Campuzano, E. Chuluun-Erdene, E. Fullerton, L. Zhang, E. Lauga, and J. Wang. Cargo-towing fuel-free magnetic nanoswimmers for targeted drug delivery. *Small*, 8:460–467, 2012.

[33] W. Gao, S. Sattayasamitsathit, K. M. Manesh, D. Weihs, and J. Wang. Magnetically powered flexible metal nanowire motors. *Journal of the American Chemical Society*, 132:14403–14405, 2010.

[34] W. Gao and J. Wang. The environmental impact of micro/nanomachines: A review. *ACS Nano*, 8:3170–3180, 2014.

[35] W. Gao and J. Wang. Synthetic micro/nanomotors in drug delivery. *Nanoscale*, 6:10486–10494, 2014.

[36] R. Gautam, A. Celani, T. Sejnowski, and M. Vergassola. Learning to soar in turbulent environments. *Proceedings of the National Academy of Sciences*, 113:E4877 – E4884, 2016.

[37] M. Gazzola, B. Hejazialhosseini, and 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.

[38] M. Gazzola, A. A. Tchieu, D. Alexeev, A. de Brauer, and P. Koumoutsakos. Learning to school in the presence of hydrodynamic interactions. *Journal of Fluid Mechanics*, 789:726–749, 2016.

[39] A. Ghosh and P. Fischer. Controlled propulsion of artificial magnetic nanostructured propellers. *Nano Letters*, 9(6):2243–2245, 2009.

[40] R. Golestanian and A. Ajdari. Analytic results for the three-sphere swimmer at low Reynolds number. *Physical Review E*, 77:036308, Mar 2008.

[41] R. Golestanian and A. Ajdari. Stochastic low reynolds number swimmers. *Journal of Physics: Condensed Matter*, 21(20):204104, 2009.

[42] R. Golestanian, T. B. Liverpool, and A. Ajdari. Propulsion of a molecular machine by asymmetric distribution of reaction products. *Physical Review Letters*, 94:220801, 2005.

[43] K. Gustavsson, L. Biferale, A. Celani, and S. Colabrese. Finding efficient swimming strategies in a three-dimensional chaotic flow by reinforcement learning. *European Physical Journal E*, 40(12):110, Dec 2017.

[44] J. Happel and H. Brenner. *Low Reynolds number hydrodynamics: with special applications to Particulate Media*. Noordhoff International Publishing, Netherlands, 1973.

[45] B. Hartl, M. Hübl, G. Kahl, and A. Zöttl. Microswimmers learning chemotaxis with genetic algorithms. *Proceedings of the National Academy of Sciences*, 118(19), 2021.

[46] J. R. Howse, R. A. L. Jones, A. J. Ryan, T. Gough, R. Vafabakhsh, and R. Golestanian. Self-motile colloidal particles: From directed propulsion to random walk. *Physical Review Letters*, 99:048102, Jul 2007.

[47] C. Hu, S. Pané, and B. J. Nelson. Soft micro- and nanorobotics. *Annual Review of Control, Robotics, and Autonomous Systems.*, 1(1):53–75, 2018.

[48] W. Hu, G. Z. Lum, M. Mastrangeli, and M. Sitti. Small-scale soft-bodied robot with multimodal locomotion. *Nature*, 5554:81–85, 2016.

[49] H.-W. Huang, M. S. Sakar, A. J. Petruska, S. Pané, and B. J. Nelson. Soft micromachines with programmable motility and morphology. *Nature Communications*, 7(26):12263, 2016.

[50] H.-W. Huang, F. E. Uslu, P. Katsamba, E. Lauga, M. S. Sakar, and B. J. Nelson. Adaptive locomotion of artificial microswimmers. *Science Advances*, 5(1), 2019.

[51] T.-Y. Huang, M. Sakar, A. Mao, A. Petruska, F. Qiu, X.-B. Chen, S. Kennedy, D. Mooney, and B. Nelson. 3D printed microtransporters: Compound micromachines for spatiotemporally controlled delivery of therapeutic agents. *Advanced materials (Deerfield Beach, Fla.)*, 27, 09 2015.

[52] E. P. Ipiña, S. Otte, R. Pontier-Bres, D. Czerucka, and F. Peruani. Bacteria display optimal transport near surfaces. *Nature Physics*, 15:610–615, 2019.

[53] H. Jeckel, E. Jelli, R. Hartmann, P. K. Singh, R. Mok, J. F. Totz, L. Vidakovic, B. Eckhardt, J. Dunkel, and K. Drescher. Learning the space-time phase diagram of bacterial swarm expansion. *Proceedings of the National Academy of Sciences*, 116(5):1489–1494, 2019.

[54] Y. Jiao, F. Ling, S. Heydari, N. Heess, J. Merel, and E. Kanso. Learning to swim in potential flow. *Physical Review Fluids*, 6:050505, May 2021.

[55] M. I Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[56] S. Kim and S. J. Karrila. *Microhydrodynamics: principles and selected applications.* New York, NY: Dover, 2005.

[57] K. Kinosita, R. Yasuda, H. Noji, and K. Adachi. A rotary molecular motor that can work at near 100% efficiency. *Philosophical Transactions of the Royal Society B*, 355(1396):473–489, 2000.

[58] N. Koumura, R. W. J. Zijlstra, R. A. van Delden, N. Harada, and B. L. Feringa. Light-driven monodirectional molecular rotor. *Nature*, 401:152–155, 1999.

[59] J. Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[60] E. Lauga. Bacterial hydrodynamics. *Annual Review of Fluid Mechanics*, 48(1):105–130, 2016.

[61] E. Lauga. Traveling waves are hydrodynamically optimal for long-wavelength flagella. *Physical Review Fluids*, 5:123101, Dec 2020.

[62] E. Lauga and T. R. Powers. The hydrodynamics of swimming microorganisms. *Reports on Progress in Physics*, 72:096601, 2009.

[63] R. Ledesma-Aguilar, H. Löwen, and J. Yeomans. A circle swimmer at low Reynolds number. *The European Physical Journal E*, 35:1–9, 2012.

[64] Y. Liu, Z. Zou, A. C. H. Tsang, O. S. Pak, and Y.-N. Young. Mechanical rotation at low Reynolds number via reinforcement learning. *Physics of Fluids*, 33(6):062007, 2021.

[65] R. D. Maladen, Y. Ding, C. Li, and D. I. Goldman. Undulatory swimming in sand: Subsurface locomotion of the sandfish lizard. *Science*, 325(5938):314–318, 2009.

[66] H. H. Mattingly, K. Kamino, B. B. Machta, and T. Emonet. E. coli chemotaxis is information-limited. *bioRxiv*, 2021.

[67] S. A. Mirbagheri and H. C. Fu. Helicobacter pylori couples motility and diffusion to actively create a heterogeneous complex medium in gastric mucus. *Physical Review Letters*, 116:198101, 2016.

[68] M. Mirzakhanloo, S. Esmaeilzadeh, and M.-R. Alam. Active cloaking in stokes flows via reinforcement learning. *Journal of Fluid Mechanics*, 903, Sep 2020.

[69] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[70] S. Muiños-Landin, A. Fischer, V. Holubec, and F. Cichos. Reinforcement learning with artificial microswimmers. *Science Robotics*, 6(52), 2021.

[71] A. Najafi and R. Golestanian. Simple swimmer at low Reynolds number: Three linked spheres. *Physical Review E*, 69:062901, Jun 2004.

[72] B. Nasouri, A. Vilfan, and R. Golestanian. Efficiency limits of the three-sphere swimmer. *Physical Review Fluids*, 4:073101, Jul 2019.

[73] X. Nassif, S. Bourdoulous, E. Eugène, and P.-O. Couraud. How do extracellular pathogens cross the blood–brain barrier? *Trends in Microbiology*, 10(5):227–232, 2002.

[74] B. J. Nelson, I. K. Kaliakatsos, and J. J. Abbott. Microrobots for minimally invasive medicine. *Annual Review of Biomedical Engineering*, 12(1):55–85, 2010.

[75] G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. van Rees, and P. Koumoutsakos. Synchronisation through learning for two self-propelled swimmers. *Bioinspiration Biomimetics*, 12(3):036001, Mar 2017.

[76] C. Ohm, M. Brehmer, and R. Zentel. Liquid crystalline elastomers as actuators and sensors. *Advanced Materials*, 22(31):3366–3387, 2010.

[77] O. S. Pak, W. Gao, J. Wang, and E. Lauga. High-speed propulsion of flexible nanowire motors: Theory and experiments. *Soft Matter*, 7:8169–8181, 2011.

[78] S. Palagi, A. G. Mark, S. Y. Reigh, K. Melde, T. Qiu, H. Zeng, C. Parmeggiani, D. Martella, A. Sanchez-Castillo, N. Kapernaum, F. Giesselmann, D. S. Wiersma, E. Lauga, and P. Fischer. Structured light enables biomimetic swimming and versatile locomotion of photoresponsive soft microrobots. *Nature Materials*, 15:647, 2016.

[79] E. M. Purcell. Life at low Reynolds number. *American Journal of Physics*, 45:3–11, 1977.

[80] J. Qiu, W. Huang, C. Xu, and L. Zhao. Swimming strategy of settling elongated micro-swimmers by reinforcement learning. *Science China Physics, Mechanics Astronomy*, 63(8), Apr 2020.

[81] G. Reddy, A. Celani, T. J. Sejnowski, and M. Vergassola. Learning to soar in turbulent environments. *Proceedings of the National Academy of Sciences*, 113(33):E4877–E4884, 2016.

[82] G. Reddy, J. Wong-Ng, A. Celani, T. J Sejnowski, and M. Vergassola. Glider soaring via reinforcement learning in the field. *Nature*, 562(7726):236–239, 2018.

[83] D. Saintillan. Rheology of active fluids. *Annual Review of Fluid Mechanics*, 50(1):563–592, 2018.

[84] P. Sartori, F. V. Geyer, A. Scholich, F. Jülicher, and J. Howard. Dynamic curvature regulation accounts for the symmetric and asymmetric beats of chlamydomonas flagella. *eLife*, 5:e13258, 2016.

[85] E. Schneider and H. Stark. Optimal steering of a smart active particle. *EPL (Europhysics Letters)*, 127(6):64003, Nov 2019.

[86] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1889–1897, 07–09 Jul 2015.

[87] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[88] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[89] S. Sengupta, M. E. Ibele, and A. Sen. Fantastic voyage: Designing self-powered nanorobots. *Angewandte Chemie*, 51(34):8434–8445, 2012.

[90] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. *Proceedings of the 31st International Conference on Machine Learning*, 32(1):387–395, 22–24 Jun 2014.

[91] O Silverberg, E Demir, G Mishler, B Hosoume, N Trivedi, C Tisch, D Plascencia, O S Pak, and I E Araci. Realization of a push-me-pull-you swimmer at low Reynolds numbers. *Bioinspiration Biomimetics*, 15(6):064001, Sep 2020.

[92] J. E. Simons and S. D. Olson. *Sperm motility: models for dynamic behavior in complex environments*, pages 169–209. Springer International Publishing, Cham, 2018.

[93] R. Stocker, J. R. Seymour, A. Samadani, D. E. Hunt, and M. F. Polz. Rapid chemotactic response enables marine bacteria to exploit ephemeral microscale nutrient patches. *Proceedings of the National Academy of Sciences*, 105:4209–4214, 2008.

[94] H. A. Stone and A. D. T. Samuel. Propulsion of microorganisms by surface distortions. *Physical Review Letters*, 77:4102–4104, 1996.

[95] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, 1998.

[96] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 20:1057–1063, 2000.

[97] A. C. H. Tsang, E. Demir, Y. Ding, and O. S. Pak. Roads to smart artificial microswimmers. *Advanced Intelligent Systems*, 2(8):1900137, 2020.

[98] A. C. H. Tsang, A. T. Lam, and I. H. Riedel-Kruse. Polygonal motion and adaptable phototaxis via flagellar beat switching in the microswimmer euglena gracilis. *Nature Physics*, 14:1216 – 1222, 2018.

[99] A. C. H. Tsang, P. W. Tong, S. Nallan, and O. S. Pak. Self-learning how to swim at low reynolds number. *Physical Review Fluids*, 5:074101, Jul 2020.

[100] S. Verma, G. Novati, and P. Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854, 2018.

[101] A. von Rohr, S. Trimpe, A. Marco, P. Fischer, and S. Palagi. Gait learning for soft microrobots controlled by light fields. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6199–6206, 2018.

[102] K. Y. Wan and R. E. Goldstein. Time irreversibility and criticality in the motility of a flagellate microorganism. *Physical Review Letters*, 121:058103, Aug 2018.

[103] Q. Wang. Optimal strokes of low Reynolds number linked-sphere swimmers. *Applied Sciences*, 9(19), 2019.

[104] Q. Wang and H. G. Othmer. Analysis of a model microswimmer with applications to blebbing cells and mini-robots. *Journal of Mathematical Biology*, 76:1699–1763, 2018.

[105] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[106] L. Xie, T. Altindal, Suddhashil Chattopadhyay, and Xiao lun Wu. Bacterial flagellum as a propeller and as a rudder for efficient chemotaxis. *Proceedings of the National Academy of Sciences*, 108:2246 – 2251, 2011.

[107] L. Yan, X. Chang, R. Tian, N. Wang, L. Zhang, and W. Liu. A numerical simulation method for bionic fish self-propelled swimming under control based on deep reinforcement learning. *Proceedings of the Institution of Mechanical Engineers, Part C*, 234(17):3397–3415, 2020.

[108] Y. Yang, M. A. Bevan, and B. Li. Micro/nano motor navigation and localization via deep reinforcement learning. *Advanced Theory and Simulations*, 3(6):2000034, 2020.

[109] J. M. Yeomans, D. O. Pushkin, and H. Shum. An introduction to the hydrodynamics of swimming microorganisms. *European Physical Journal Special Topics*, 223(9):1771–1785, Sep 2014.

[110] L. Zhang, J. J. Abbott, L. Dong, K. E. Peyer, B. E. Kratochvil, H. Zhang, C. Bergeles, and B. J. Nelson. Characterizing the swimming properties of artificial bacterial flagella. *Nano Letters*, 9(10):3663–3667, 2009.