A lateral line sensor based mechanistic algorithm for emergent fish schooling behaviours in multi-agent swarms

By

VEDANG JOSHI

Department of Engineering Mathematics
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of MASTER OF ENGINEERING in the Faculty of Engineering.

25 APRIL 2022

Abstract

Fish use a specialist sensory system, known as the lateral line, to extract hydrodynamic data from the local environment. The lateral line sensor may be divided into the fluid acceleration-sensitive canal neuromast and the fluid velocity-sensitive superficial neuromast subsystems. Studies on the function of the lateral line have revealed the sensor to be crucial for the emergence of fish schools. However, the exact functional mechanisms governing this behaviour are unknown. Emulating such a sensory mechanism, coupled with an appropriate control mechanism, could aid the navigation of swarms of autonomous undersea vehicles, as they traverse the murky waters in the depths of the oceans. The vortices generated in the wake of the swimming fish are termed Kármán vortex streets (KVSs). We hypothesise that the fish are able to detect these KVSs by searching for the lowest pressure positions in their neighbourhood, based off the canal neuromast input, and navigate towards other fish. We further hypothesise that this behaviour leads to the emergence of a school of fish. The objectives of this study are twofold: we create a fluid environment, permitting free movement of agents within this region; we simulate a 2D fluid environment with the Reynolds number, $Re = 100$. The fish are modelled as circles. We observed the formation of the KVS behind the lead fish. We further propose a novel KD-Tree based control algorithm to allow the emergence of a formation of three fish in the domain. We tested the proposed algorithm based on a robustness to initial positions metric. We found that the Euclidean distances between the three fish converge to $\approx 0.2\ m$ when the initial positions of the fish are near to the front of the domain. This proposed algorithm has the potential to inform future work in decentralised control mechanisms for autonomous undersea vehicles.

Dedication and Acknowledgements

First, I would like to thank both my supervisors Dr. Sabine Hauert and Dr. Elliott Scott for their invaluable support throughout this year as I embarked upon this project. They stoked and developed my interests in bio-inspired robotics, multi-agent systems and computational fluid dynamics and for that I'm extremely grateful. In our regular meetings, they taught me how to think clearly and logically about problems, and patiently filled in the gaps in my knowledge. I would like to acknowledge Elliott, for giving up his time to give constructive feedback on my thesis.

I would like to thank Prof. Chris Gilligan's lab at Cambridge and Prof. Nick Jones' lab at Imperial College London for hosting me as a research assistant during my summers at university. These experiences helped me learn new analytical tools and techniques and overall greatly shaped my dissertation topic this year.

Besides the research aspect, I have been fortunate to make many life-long friends at university including my first and second year flatmates Mark, Alina and Elisha. Although I had to get over my disappointment by missing my chance to go abroad for my third year due to the pandemic, I thoroughly enjoyed my time at Clifton Hill House. A big shout out to Josh, Kim, Addison and numerous others for some great memories over the past few years.

I wish to express my deepest love and thanks to my girlfriend Sophia. I'm incredibly lucky to have met her. Thank you for introducing me to the magic of the theatre and musicals. I don't think I've laughed harder than when we went to see the Book of Mormon. Thank you for helping me take my mind off work when I've been stressed. Thank you especially for spending time reading through my thesis and pointing out strange sentence constructions and grammatical usages.

Finally, I want to thank my family for always supporting me, I couldn't have reached this far without all of you. Thank you Aaji and Aaba for encouraging me to pursue my dreams and to never give up. Thank you Ketan and Rucha for the numerous laughs when we're at home. A special thanks to Ketan for making me think more critically of my work and for our talks on the philosophy of science. Our discussions seem to illuminate a new perspective, one that I'm usually lacking, and for that I thank him. A big thank you to Aai for everything she's done for me over the years including listening to my seemingly endless frustrations when a piece of code wouldn't work, and her comforting reassurance that it will all work out. Thank you Baba for guiding my mathematical development ever since I was at school. When I failed to grasp a concept, he would patiently repeat his explanations until I did. I would never have studied maths at university and consequently written this thesis, had it not been for his efforts all those years ago. This work is dedicated to him.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: Vedang Joshi DATE: 25 April 2022

Table of Contents

LIST OF FIGURES

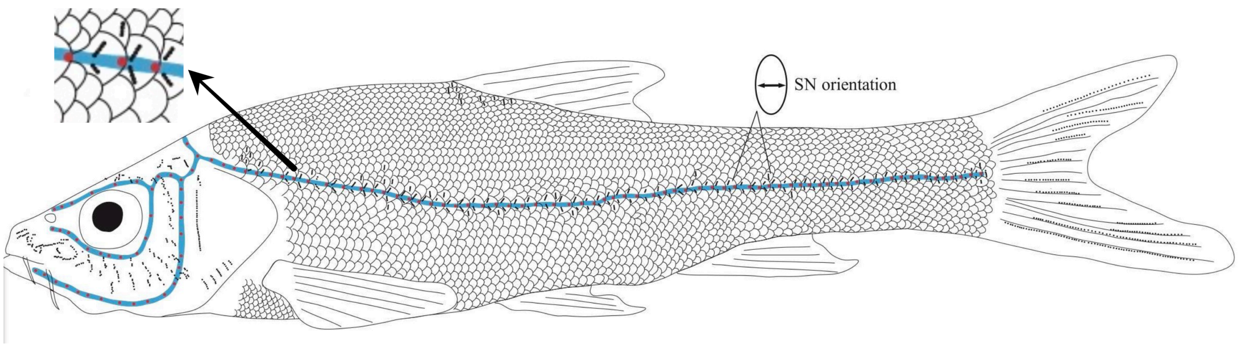
Chapter 1

Introduction

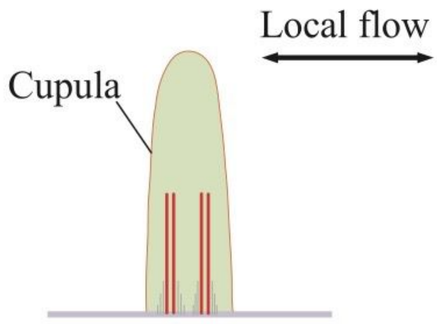
Fish comprise half of all extant vertebrate species on the planet [45]. Extensive research indicates most fish, along with some other aquatic animals, possess the ability to extract and interpret local hydrodynamic data through the lateral line sensor (see Figure 1.1). This sensory organ is capable of distinguishing between laminar and turbulent flow, so some species of fish have evolved to rely solely on the lateral line for navigation [31]. The lateral line consists of a series of hair-like cells covered by a cupula, which forms the neuromast structure. There are two different types of neuromasts: superficial neuromasts (SNs) and canal neuromasts (CNs). SNs are located on the skin of the fish while CNs are situated in hollow canals underneath the skin [13]. Kroese and Schellart [33] show that SNs are sensitive to external flow velocity variations while CNs detect external flow accelerations, which are influenced by local external pressure variations. This implies that SNs collect information on local external velocity profiles and CNs collect information on local pressure gradients. The lateral line system is imperative to help detect prey [14], communicate with other neighbouring fish [8], aid in rheotaxis (self-orientation in the direction of fluid flow) [2] and influence shoaling behaviours in fish [55].

1.1 Background

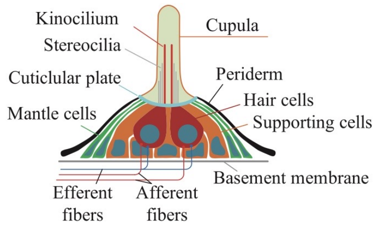
Shoals refer to any group of fish that remain together for social reasons, while ‘schools’ refer specifically to a synchronised group of fish demonstrating polarised swimming behaviour [54]. Individuals in a shoal suffer a minimised risk of predator capture through synchronised anti-predatory mechanisms including predator detection [41], predator evasion [27] and confusion [49]. Recent evidence indicates that all individuals also save energy regardless of their positions in the shoal [23]. Indeed, fish have been shown to slalom between experimentally generated vortices in water to decrease muscle activity leading to a decreased cost of locomotion [40]. Shoaling also has foraging benefits where the task of gathering food is distributed across individuals [42]. These shoaling behaviours display scalability, robustness and distributed self-organised principles. Note here, that there is no centralised control mechanism for these behaviours and that they have emerged out of simple interactions between neighbouring fish in a shoal. Research into this area has



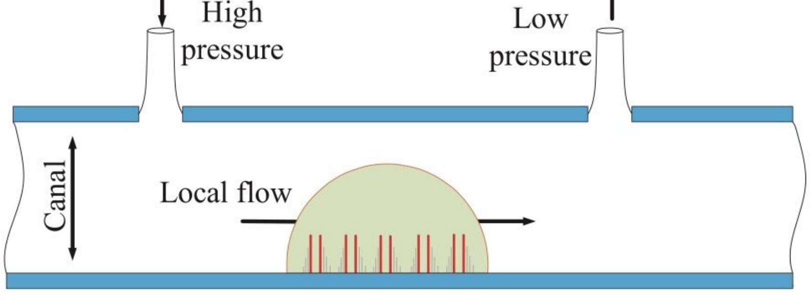
(a)



(b)



(c)



(d)

FIGURE 1.1. The fish lateral line sensor. (a) The distribution of superficial neuromasts (SNs) (black dots) and canal neuromasts (CNs) (red dots) across the fish body. (b) Diagram showing the SNs along with the cupula covering. (c) The anatomical description of the neuromast with the mantle and the supporting cells covered by the cupula. (d) Diagram showing the CNs connected to the local environment through pores on the canal surface. Figure reproduced with kind permission from the Institute of Physics, UK [32].

given rise to the field of swarm intelligence.

A review in swarm intelligence defined swarm robotics as the study of how to coordinate large groups of simple robots through the use of local rules, without any centralised control mechanism [48]. Swarm robotics takes inspiration from behaviours of societies of organisms that perform complex tasks well beyond the capabilities of a single organism. A recent study was influenced by bees and slime mould to form an integrated underwater swarm robotic exploration system [69]. Another study took inspiration from frog-call anti-phase synchronisation to allocate communication time slots in robot swarms [66]. Applying shoaling behaviour benefits to swarms of underwater robots has been difficult so far as a result of the computational expense of visual navigation in a salt water medium, which also obstructs radio waves for communication [51]. A lateral line based control system driving the shoaling behaviours in robot swarms could resolve this problem. Currently, there are no swarm robotics studies that primarily use the lateral line as a sensor input to control robot swarms. We believe this could be effected by first simulating a realistic fluid environment and creating a control algorithm using the input to the lateral line based off the vortices produced in the wake of individual fish in a shoal.

Vortices in fluids result from discrete areas of turbulence in the flow and cause local system perturbations, and are important to examine for long term fluid behaviour. A well-researched problem is the categorisation of vortices formed behind a cylinder in a uni-directional flow [73]. A well-known quantity in fluid mechanics to predict flow patterns in various fluid flow situations is the dimensionless Reynolds number (Re) defined as,

$$Re = \frac{\rho \cdot U_{mean} \cdot L}{\mu}, \tag{1.1}$$

where ρ is the density of the fluid, U_{mean} is the mean velocity in the environment, L is the characteristic length of the flow configuration and μ is the dynamic viscosity of the fluid. Vortices are shed periodically on alternating sides of the cylinder, when situated in a flow with $50 < Re < 3 \cdot 10^5$ [63]. At low Re the wake is laminar while it is turbulent for higher Re [50]. This periodic alternating vortex shedding in turbulent flows is called a Kármán vortex street (KVS). See Figure 1.2 for a graphical representation of the KVS. Reverse KVSs (KVSs with the direction of the vortices reversed) are also observed in the wake of swimming fish [75]. We hypothesise that the fish use these KVSs as a distinctive marker to allow fish to shoal together.

1.2 Motivation

This study aims to extend current work by Farrel Zulkarnaen, Elliott Scott and Sabine Hauert at the Hauert Lab, Bristol Robotics Laboratory. Scott developed an artificial lateral line sensor based on CNs which could be used in simple, low-cost robots for shoaling. Zulkarnaen further demonstrated that a single agent is able to navigate the flow using only pressure data obtained theoretically from the CN based lateral line sensor. Zulkarnaen trained a neural network on pressure data obtained from randomly generated points behind a cylinder in a fluid flow to predict a bearing for the fish in the flow. The pressure data was obtained from a pre-run simulation of a cylinder in a fluid created by Scott. This neural network is effectively a 'black-box' model, i.e. there is a lack of access to the inner workings and parameters of the model. To further this work, it is essential to create a simulation environment to allow a control algorithm to function in simulated 'real time'. In this context, we use 'real time' to refer to the simulation executing

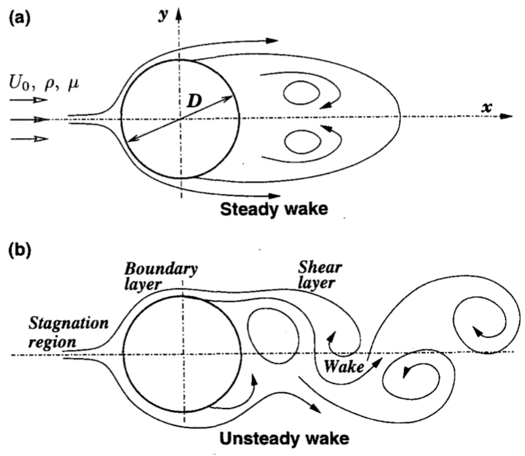


FIGURE 1.2. Formation of vortices behind a cylinder with $Re = 105$. (a) The influence of the adverse pressure gradients forms two free shear layers creating a region with two attached vortices behind the cylinder forming a steady wake. (b) The flow experiences primary wake instability inducing an unsteady wake with periodically alternating vortices. This forms the KVS. Figure reproduced from [56].

at the same rate as the 'wall clock' time. This thesis aims to create such a fluid simulation environment in which to investigate our hypothesis and allow for minimalistic fluid-structure interaction properties. Our secondary aim is to develop a mechanistic understanding of how the lateral line system influences a swarm of fish to make certain navigational decisions. This is achieved by taking local pressure data around the fish as an input to an in-house developed control algorithm, and output a direction vector to help navigate the fish, through the KVS, towards the lead fish. We hypothesise that control algorithms developed with the KVS characteristic inputs will show the emergence of shoaling behaviour in the modelled fish. The code for the environment and algorithm developed in this thesis is available at the project GitHub repository: `https://github.com/vedang-joshi/University_of_Bristol_masters_dissertation`

1.3 Thesis outline

This thesis shall be divided into six chapters. Chapter 1 is dedicated to introducing the problem. Chapter 2 reviews the experimental evidence for the necessity of the lateral line sensor. We discuss how the lateral line input may help fish identify KVSs. We also briefly discuss how this categorisation influences the fish to make certain decisions to position themselves in a formation around a lead fish. Chapter 3 details the creation of the simulation environment backed by the necessary mathematical theory with a description of the Navier-Stokes equations. We describe the different boundary conditions imposed on a domain, the weak formulation of Navier-Stokes equations and introduce the finite element method to discretise the domain. We show how minimalistic fluid-structure interaction can be achieved by the Arbitrary-Lagrange method. Chapter 4 describes the novel control algorithm proposed to demonstrate the emergence of schooling behaviours in swarms. Chapter 5 is dedicated to discussing the results obtained as well as limitations of the simulation environment and the proposed control algorithm. Chapter 6 concludes our findings and mentions several ways the current research may be extended.

Chapter 2

Related Works

The goal of this chapter is to present the experimental evidence for our main claim: the lateral line is crucial for accurate shoaling behaviour in fish schools. We discuss the extent to which the lateral line can detect the presence of KVSs. Once the KVSs have been detected in the local environment, it is essential to navigate through this KVS. For a comprehensive discussion about the fish navigating through the environment, we formally define the fish swimming and interacting with the local environment, as a multi-agent system. We review the literature surrounding these areas. Finally, we mention the current work conducted by Scott and Zulkarnaen in lateral line input sensing and navigational control using pressure data.

2.1 Lateral line is essential for accurate schooling behaviour

It has long been determined that the lateral line system plays a vital role in schooling behaviour in fish. A study on saithe demonstrated that blindfolding fish had little effect on the position these fish took up with respect to their neighbours within a school. In contrast, the lateralis section resulted in a greater increase in the frequency of neighbours at 90° bearings to each other [55]. The authors report a significant quantitative difference between reaction times and schooling behaviour within the school of fish. These bearing measurements play a vital role in the accurate emergence of fish schools.

Faucher *et al.* [21] use a 'nearest distance to neighbours' metric to define schooling in their experiment, which aimed to determine the role of the lateral line for schooling purposes. In this study, the authors disable the trunk and head lateral line system as compared to the study in [55], where just the trunk lateral line is disabled. Pre-treatment, the authors report 95-98% of fish swim in the same direction. They also report that there were no collisions between the fish in the shoal. They report a small stable shoaling radius ($\approx$ 15 cm) indicating consistent robust shoaling behaviour. Post treatment, the authors report a 25% expansion in the shoal radius. They report only 63% of the fish swim in the same direction. The authors also observed higher collision rates between fish. The study is the seminal paper in demonstrating the crucial role of the entire lateral line system in shoaling behaviour in fish.

Mekdara *et al.* [45] studied how chemical ablation (i.e. chemical suppression) of the lateral line system affects schooling behaviour in giant danios. They reported that the fish could swim normally, exhibiting the same schooling behaviour immediately after treatment. However, after 1-2 weeks, the fish began to show difficulty in schooling. After 4-8 weeks of treatment, when the hair cells regenerated, the fish exhibited pre-treatment behaviour. The authors suggest that during the recovery process, the neurons connecting the lateral line system to the brain of the fish may make connections to new hair cells resulting in completely different inputs to the brain. The brain may take longer to adjust to these new inputs resulting in a longer recovery time. This definitively suggests that the lateral line system plays a vital role in fish schooling behaviour.

2.2 Lateral line input can identify KVS characteristics

Ren and Mohseni [60] developed an analytical model to show how the lateral line sensor input is influenced by the fish passing through a KVS. The authors demonstrate that such a model contains information to detect important parameters in the KVS, such as the vortex strength, the lateral and flow-wise spacings, the translational velocity and the orientation of the vortex street. They model the lateral line sensor as a cylinder, with pores running uniformly along its length. The pressure gradient between adjacent pores drives the flow in the canals. They suggest that a periodic pressure wave is expected at each pore as the KVS passes the fish. They note that the pressure gradient created by the KVS results in the input to the lateral line only containing low-frequency components. They show that the strength of the KVS is a function of the lateral line canal diameter, the viscosity of fluid in the canal, the characteristic velocity of fluid in the canal, the horizontal distance between vortices and the dimensionless velocity inside the canal. They comment that an accelerating fish in a school produces a KVS with an increase in vortex spacing and strength. They claim that a fish changing its direction will result in an alteration in the velocity distribution in the lateral line. This study indicates how artificial agents could be taught to detect a KVS. However, applying these ideas to a 'real time' control algorithm may not be feasible, as it requires fish to have an understanding of the the global vortex dynamics.

Zheng *et al.* [79] designed a robotic fish completely equipped with an artificial lateral line system primarily composed of pressure sensor arrays. The authors used it to detect a reverse KVS-like wake in adjacent robotic fish. The KVS results in hydrodynamic pressure variations in the flow field which the authors use along with the pressure sensor readings in their analysis. Using these readings, the oscillating frequency, amplitude, the relative vertical distance and the relative yaw, pitch and roll angles of adjacent fish can be effectively sensed. This study demonstrated the ability of fish to glean local flow information, and informs future projects involving close-range interaction and cooperation between underwater swarms of robots.

A similar experiment was carried out by Wang *et al.* [74] in which the authors also constructed a bio-inspired robotic fish. The motion was controlled by a linear Central Pattern Generator, fitted with a pressure sensor acting as the artificial lateral line system in the fish. The authors analyse the pressure data from the experiments conducted and extract the vortex intensity from this data to determine the distance between the pair of robots considered. The authors estimate a quadratic relationship between the vortex intensity at pressure 0, when the distance between the two fish considered is under 25 cm. The authors conclude that, at shorter distances between neighbouring fish, there exists a relationship between vortex intensity and distance between fish.

These are only two examples out of many studies on bio-inspired robotic fish. Artificial lateral lines with bionic neuromasts have been extensively studied and have been deployed on bio-inspired swimming robots [20][78][76]. Lagor *et al.* [34] describe one such artificial lateral line system capable of autonomous flow speed estimation and rheotaxis using only flow sensing information. DeVries *et al.* [17] present estimation and control strategies allowing a bio-inspired swimming robot to assimilate measurements from a multi-modal artificial lateral line system, allowing the robot to exhibit rheotaxis and station-holding behaviour.

2.3 Biological navigation in a KVS informing control algorithms

Liao *et al.* [40] investigated the movement of fish in a KVS and defined the response of the fish as a new Kármán gait. They report that the fish slalomed between the vortices within the KVS. The tail-beat frequency of the fish changed to match the vortex shedding frequency, along with a marked increase in the curvature and amplitude of the fish bodies. The authors found that the fish decrease their muscle activity to conserve energy through such a Kármán gait, which could influence fish-like robot motions through KVSs.

Chagnaud *et al.* [9] conducted experiments on the responses of the lateral line of goldfish in a unidirectional flow of water. The authors computed the flow velocity curves and analysed the frequency spectra of these curves at positions occupied by the fish during the experiment. They noted a sharp amplitude spike in the frequency spectra of the flow velocity curves at the vortex shedding frequency, when the fish navigated through a KVS. They also noted that this increase was more pronounced if the fish intercepted the edge of a KVS. These results are imperative while constructing control algorithms to determine how the interception of flow velocities affects fish navigation.

Venturelli *et al.* [70] conducted experiments with robotic fish with off-the-shelf pressure sensors to record data. They presented a pressure frequency spectrum analysis where a fast Fourier transform (FFT) was applied to the recorded pressure data to detect dominant frequencies from the frequency spectrum. The authors used a three second time window for the FFT analysis. They presented a turbulence intensity metric computed as the standard deviation of the flow velocity divided by the mean flow velocity at each point. The authors claim that the metric provided information on turbulent wakes through a spatial distribution analysis of the turbulence intensity.

Salumäe and Kruusmaa [61] conducted a study based on [70] to develop ‘real time’ algorithms for station holding in the KVS. They constructed a robotic self-propelled fish and recorded pressure data from the nose and sides of the robot. They applied the FFT to their data with a 30-50 second time window, and observed peak frequencies in pressure readings at the actuation and the vortex shedding frequencies. The authors concluded that although KVS characteristics were observed in the pressure input, the long time window for the FFT renders the two analyses in [70] unsuitable for ‘real time’ robot navigational control.

2.4 Current work in lateral line input sensing and navigation control

Scott [63] designed and developed a prototype of a lateral line sensor primarily focusing on the CN structure. The sensor was developed as a hollow curved tube, closely resembling a candy cane. The shorter end of the tube contains an artificial neuromast with a pin-like structure, which deflects in response to changing external pressures. The water pressure within the long (canal) side of the 'candy cane' remains neutral, so water is accelerated in or out of the mouth of the sensor (short side) in response to changing external pressures. The water accelerates inwards for high pressures and outwards for low pressures. This is low pressure reservoir. Scott used a static cylinder to generate vortices in a simulated fluid flow. He approximated the solutions to the Navier-Stokes equations (the robust model for fluid flows) using a Semi Implicit Method for Pressure Linked Equations (SIMPLE) numerical method, the Reynolds-Averaged Navier Stokes (RANS) solver for steady state approximations of the KVS and the $k-\omega$ transport model for turbulence computations. The oscillation of the neuromast generated by the changing velocities behind the cylinder was analysed by Farrel Zulkarnaen.

Zulkarnaen [80] constructed a learning algorithm with a neural network framework using the sensory readings provided from the simulations by Scott. The neural network framework learnt the non-parametric mapping between the sensory output signal and the position of the source of the signal, and provided an estimate of the position of the source of the KVS. Zulkarnaen extracted the sensory signals by randomising the positions of the sensor uniformly across a defined flow region. By sampling at these locations, he constructed a time series of the artificial neuromast oscillations. Zulkarnaen simplified the need for complicated neuromast oscillation simulations by assuming only the horizontal component of the exerted forces on the neuromast causes oscillation. Scott adjusted the design of the sensor to minimise frictional resistance in the flow. Thus, only the horizontal component of the flow velocity was considered as the driver for the oscillatory signal considered in the thesis. A final contribution of the thesis was the development of a simple controller based off the input from the neural network produced, which navigated a single agent towards the cylinder under consideration. This method lead to a significant delay as the neural network had to learn for 200 time units before the controller started navigating in the flow. This method is therefore unsuitable for 'real time' robot navigation.

2.5 Summary

We have highlighted the importance of the lateral line to ensure accurate fish schooling behaviour. We conducted an extensive literature review on the ability of the lateral line to identify KVS characteristics, but noted that only the global KVS characteristics may be detected. We recognise that individual agents may not have access to this data and that these insights cannot be applied to the development of control algorithms using only the local hydrodynamic data. We refer to the extensive work in the literature on experimental evidence for biological navigation in a KVS. We note that the current work on robotic fish navigation in a KVS cannot explain the mechanics of this capability. We refer to a study which shows that, although the vortex shedding frequency was sensed based off local pressure readings, the data collected was over a 30-50 second window. This is a significant delay and this approach cannot be used in 'real time' navigation. We briefly describe the current work in lateral line input sensing through the development of a simplistic lateral line sensor [63] and a proposed neural network [80] to navigate a single agent in a fluid flow. However, this framework can neither be extended to a multi-agent system, nor used for 'real time' navigation.

Chapter 3

Simulation environment

To be able to deploy autonomous undersea vehicles based on fish schooling behaviour, we require careful modelling of the surrounding environment i.e. a general fluid flow model. Fluids obey the general laws of mechanics, namely the conservation of momentum and the conservation of mass. These laws constitute the Navier-Stokes equations. Currently, proving the guaranteed existence of smooth solutions of the Navier-Stokes equations in d-dimensions is an open problem in mathematics [22]. The Navier-Stokes problem is well posed in the two-dimensional space, to the extent that if the initial conditions, boundary conditions and the forcing terms are smooth enough, then the solution along with its derivatives is continuous [57]. This is not always the case for the Navier-Stokes problem in three dimensions where the existence of solutions has only been proven for small time intervals [57]. For simplicity, we constrain all further analysis of the problem to two dimensions. This chapter aims to provide the necessary mathematical theory behind the Navier-Stokes equations in two dimensions. We derive the necessary reformulations required to apply numerical frameworks to create a suitable simulation environment for our multi-agent system. We provide the results of the simulations towards the end of the chapter.

3.1 Navier-Stokes equations

The Navier-Stokes equations are required to determine the fluid velocity field in a domain. These equations can be derived by applying the laws of conservation of mass and conservation of momentum using Newton's second law to a fluid element. We shall introduce some notation here for use throughout the thesis.

Consider a fluid flowing in a 2-dimensional Euclidean space, $\Omega \subset \mathbb{R}^2$, with a velocity field $\boldsymbol{u}\colon \Omega \times [0,T] \to \mathbb{R}^2$ and static pressure field $p\colon \Omega \times [0,T] \to \mathbb{R}$, where $T \in 0 < T \leq \infty$. The domain Ω is bounded with a piece-wise smooth boundary $\partial\Omega$. Boundary conditions exist on the boundary, $\partial\Omega = \partial\Omega_{\text{wall}} \cup \partial\Omega_{\text{inlet}} \cup \partial\Omega_{\text{outlet}}$, where $\partial\Omega_{\text{wall}}$, $\partial\Omega_{\text{inlet}}$, $\partial\Omega_{\text{outlet}}$ are the boundary conditions on the walls, inlet and outlet respectively. See Figure 3.1 for a diagrammatic representation of the domain and the boundaries. Under the assumption that the fluid is Newtonian and incompressible, the general Navier-Stokes equations are as follows,

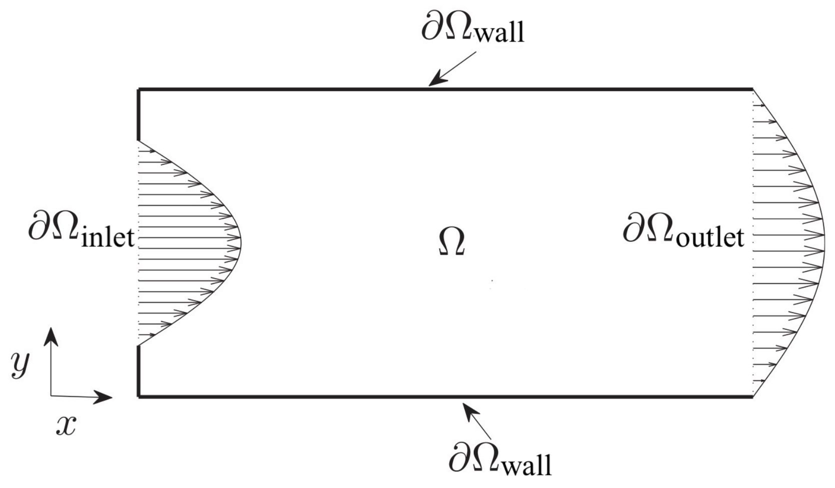

$\partial\Omega_{\text{wall}}$
$\partial\Omega_{\text{inlet}}$
Ω
$\partial\Omega_{\text{outlet}}$
y
x
$\partial\Omega_{\text{wall}}$

FIGURE 3.1. A schematic illustration of boundary conditions in the domain, Ω, under consideration. Figure adapted from [37].

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\mu}{\rho}\nabla^2 \boldsymbol{u} - \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nabla p + \boldsymbol{f}, \tag{3.1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{3.2}$$

where equation 3.1 is the momentum equation and equation 3.2 is the continuity equation. These equations together are the strong form of the Navier-Stokes equations. Here, μ is the dynamic viscosity, ρ is the fluid density and $\boldsymbol{f}$ refers to the effect by any external forces. For simplicity, we assume $\boldsymbol{f} = 0$. Here, ∇ is the gradient, $\nabla\cdot$ is the divergent operator and ∇^2 is the Laplace operator.

3.2 Boundary conditions

There are several types of boundary conditions which may be imposed for incompressible flows. The Dirichlet boundary conditions describe the velocity field on the whole or part of the boundary. The general Dirichlet boundary condition is given as,

$$\boldsymbol{u} = \boldsymbol{g}, \tag{3.3}$$

where $\boldsymbol{g}$ is a given velocity profile on the boundary.

3.2.1 Wall boundary condition

For the wall boundary, $\partial\Omega_{\text{wall}}$, we impose the following Dirichlet boundary condition,

$$\boldsymbol{u}(x,y,t)=\boldsymbol{0}, \tag{3.4}$$

which is a special case of the general Dirichlet boundary condition known as the no-slip boundary condition.

3.2.2 Inlet boundary condition

For the inlet boundary, $\partial\Omega_{\text{inlet}}$, we impose the following Dirichlet boundary condition,

$$\boldsymbol{u}(x,y,t)=\boldsymbol{g}(x,y,t), \tag{3.5}$$

with a velocity profile, $\boldsymbol{g}(x,y,t)$, specified

3.2.3 Outlet boundary condition

In numerical simulations, the natural boundary condition is often imposed. The Cauchy stress tensor is defined as $\sigma(\boldsymbol{u},p)=(-\frac{\mu}{\rho}\nabla\boldsymbol{u}+\ p\mathbb{I})$ where $\mathbb{I}$ is the identity matrix. This represents the internal forces in the flow. Let $\boldsymbol{n}$ be the unit vector normal to the boundary. For the outlet boundary, $\partial\Omega_{\text{outlet}}$, we impose the boundary condition where the normal stress, which equals the Cauchy stress tensor, vanishes at the boundary,

$$\sigma(\boldsymbol{u},p)\ \cdot\boldsymbol{n}=\boldsymbol{0}, \tag{3.6}$$

The simulation model presented in the thesis will assume that the flow computations continue into an 'imaginary channel' at the outlet, where the derivative of the velocity in the direction of the channel is zero. This boundary condition corresponds to a flow at the outlet with no significant changes downstream.

3.3 Numerical algorithms for the Navier-Stokes equations

It is important to understand the nature of partial differential equations (PDEs) to be able to have a comprehensive discussion about constructing numerical frameworks for PDEs. PDEs can be broadly categorised as elliptic, hyperbolic and parabolic. The Navier-Stokes equations may be categorised as a parabolic-hyperbolic PDE problem [4] which is prone to discontinuities in its solutions [30]. These discontinuities may occur due to small changes in the problem parameters or these solutions may only exist in some part of the domain or in time. To obtain a stable numerical framework, it is essential to obtain a 'well-posed' problem, where the term 'well-posed' refers to the guaranteed continuous existence of a unique solution for the PDE problem.

Rempfer [59] remarks that the pressure term (on the right hand side of equation 3.1) appears as a Lagrange multiplier to ensure that the velocity field remains incompressible at all times, thus satisfying equation 3.2. The interaction between the momentum and continuity equations causes a saddle-point stability problem. This saddle-point problem, due to the non-linearity in the pressure term, necessitates the development of an alternative set of PDEs to be solved.

The complex nature of the Navier-Stokes problem requires an approximation of the solutions on the domain. The finite element method is a powerful computational technique to break down complex domain geometries into simpler sub-domains. This method (particular to the Navier-Stokes problem) may be categorised by three features:

1. The domain may be represented by a collection of sub-domains called *finite elements*. The collection of finite elements is called a *finite element mesh*.
2. Over each finite element, the problem is approximated by functions (polynomials etc.), and algebraic equations may be developed at selective points on the finite elements. This is known as a *finite element model*. One procedure for obtaining such a set of finite elements is the weak formulation of the PDE problem, which is then solved by approximating the PDE problem as a system of linear equations.
3. The finite element model is then solved by obtaining an explicit equation for the pressure by breaking the coupling between the velocity and pressure in equations 3.1 and 3.2. One of the oldest 'splitting' methods proposed includes the one by Chorin [12] and Temam [67], known as Chorin's method. We shall consider a modified version of Chorin's method, called the incremental pressure correction scheme (IPCS), in our simulations as it provides improved accuracy at slightly extra computational cost [26].

The rest of the section will focus on an introduction to finite elements, the weak formulation of the Navier-Stokes problem, a description of the IPCS and a derivation of the weak formulation of the IPCS. We shall explore briefly the approximation of the Navier-Stokes problem using iterative methods for linear systems.

3.3.1 Introduction to finite elements

We can describe Ω using a set of vertices. We can produce a set of nodes in Ω depending on these vertices. We define triplets of these nodes to be *finite elements* with the condition that these elements may only exist between nearby nodes, thereby creating small, closed shapes on the domain. The finite elements are connected to each other through these nodes. This discretisation of space across the domain creates a *finite element mesh*. The finite element method states that the solution on the continuous domain may be approximated by the solutions obtained on these discretised finite elements [10]. There are many types of elements spanning a wide range of spatial dimensions. There are some element types that permit 'mid-side' nodes i.e. elements with nodes positioned midway between the vertices. These mid-side nodes are found in finite elements in higher order geometries. Some commonly used finite elements for 2D and 3D geometries are shown in Figure 3.2. We use a triangular finite element in our implementation of the finite element mesh (see Figure 3.3).

3.3.2 Weak formulation of the general Navier-Stokes equations

We restate the strong form (equations 3.1 and 3.2) of the Navier-Stokes equations in an integral form. This is known as the weak form of the Navier-Stokes equations. The idea is to frame the strong Navier-Stokes equations in such a way that the solution to the weak formulation is sought in a trial space, for which

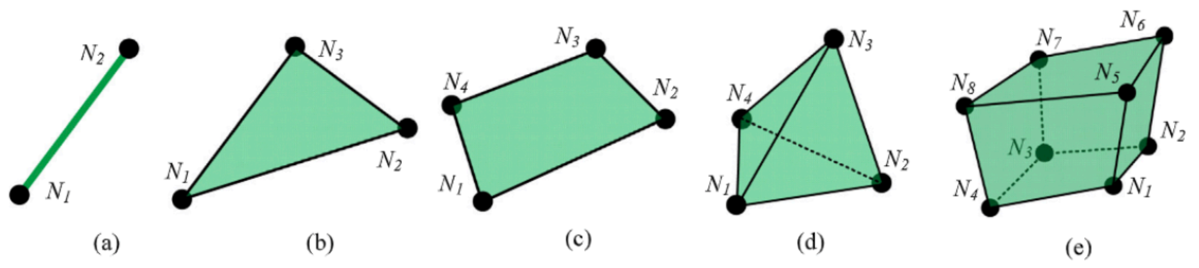


FIGURE 3.2. Commonly used finite element types. a) Line element (2D). b) Triangular element (2D). c) Quadrilateral element (2D). d) Tetrahedron element (3D). e) Hexahedron element (3D). Figure reproduced from [58].

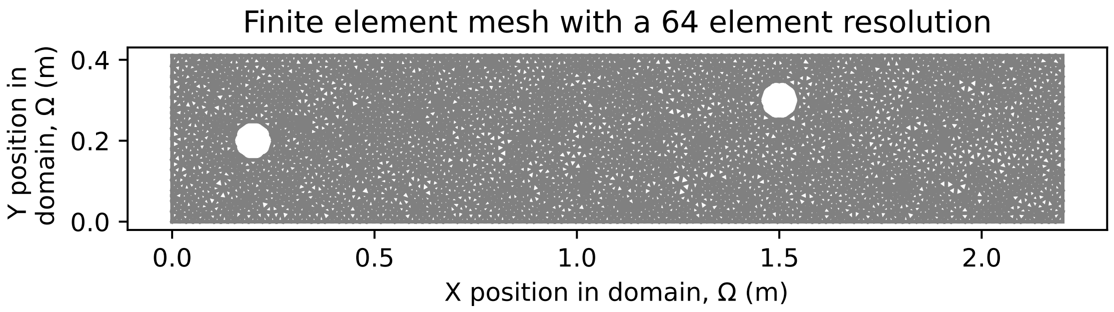

Finite element mesh with a 64 element resolution
Y position in domain, Ω (m)
0.0
0.2
0.4
0.0
0.5
1.0
1.5
2.0
X position in domain, Ω (m)

FIGURE 3.3. An illustration of a mesh constructed on a domain Ω, with two circular objects initialised at (0.2, 0.2) and (1.5, 0.3). The mesh resolution is 64 triangular elements.

the reformulation is satisfied for all functions of a test space. For the sake of completeness, we derive the reformulation of the equations 3.1 and 3.2. The presentation of this section mostly follows [57]. We multiply equation 3.1 by a test function, $\boldsymbol{v} \in \boldsymbol{V}$, where $\boldsymbol{V}$ is a suitable space, and integrate by parts over the domain, Ω. Rearranging the terms slightly,

$$\begin{aligned}\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{v}\, dx - \int_\Omega \frac{\mu}{\rho} \nabla^2 \boldsymbol{u} \cdot \boldsymbol{v}\, dx + \int_\Omega [\boldsymbol{u} \cdot (\nabla \boldsymbol{u})] \cdot \boldsymbol{v}\, dx \\ + \int_\Omega \nabla p \cdot \boldsymbol{v}\, dx = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\, dx,\end{aligned} \tag{3.7}$$

We apply an integration by parts to the second and fourth terms in equation 3.7 to yield,

$$-\int_\Omega \frac{\mu}{\rho} \nabla^2 \boldsymbol{u} \cdot \boldsymbol{v}\, dx = \int_\Omega \frac{\mu}{\rho} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v}\, dx - \int_{\partial\Omega} \frac{\mu}{\rho} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \cdot \boldsymbol{v}\, ds, \tag{3.8}$$

$$\int_\Omega \nabla p \cdot \boldsymbol{v}\, dx = -\int_\Omega p \nabla \cdot \boldsymbol{v}\, dx + \int_{\partial\Omega} p \boldsymbol{v} \cdot \boldsymbol{n}\, ds, \tag{3.9}$$

Substituting the right hand side of equations 3.8 and 3.9 in equation 3.7 and collecting terms, we obtain,

$$\int_\Omega \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{v}\, dx + \int_\Omega \frac{\mu}{\rho} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v}\, dx + \int_\Omega [\boldsymbol{u} \cdot (\nabla \boldsymbol{u})] \cdot \boldsymbol{v}\, dx - \int_\Omega p \nabla \cdot \boldsymbol{v}\, dx - \int_{\partial\Omega} (\frac{\mu}{\rho}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n}) \cdot \boldsymbol{v}\, ds = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\, dx, \tag{3.10}$$

We conduct a similar procedure on equation 3.2, where we multiply the equation by a test function, $q \in \boldsymbol{Q}$, where $\boldsymbol{Q}$ is a suitable space, and integrate over the domain, Ω. We get,

$$\int_\Omega q \nabla \cdot \boldsymbol{u}\, dx = 0. \tag{3.11}$$

Equations 3.10 and 3.11 together describe the weak formulation of the general Navier-Stokes equations. This weak problem is a continuous problem with the solutions $\boldsymbol{u}$ and p existing in an infinite dimensional space. These infinite dimensional spaces are replaced by finite dimensional function spaces $\boldsymbol{V}$ and $\boldsymbol{Q}$ consisting of continuous piece-wise linear functions [36]. The function spaces are chosen in such a way that the test functions disappear on the boundary when Dirichlet boundary conditions are imposed on Ω. The function spaces $\boldsymbol{V}$ and $\boldsymbol{Q}$ are known as Hilbert spaces. It is beyond the scope of this thesis to provide a mathematically rigorous introduction to Hilbert spaces, and it is sufficient to understand that we can impose certain trial and test functions, which test the approximation of the solution of the PDE problem on suitable vector and scalar spaces. The main idea of the reformulation step is to consider an averaged version of the equation satisfied on Ω instead of the equation being satisfied continuously on Ω. We shall, however, clarify the idea of a 'suitable' space here.

As discussed earlier, the construction of any numerical framework entails a well-posed PDE problem, along with the resolution of the saddle-point problem. The Ladyzhenskaya-Babuska-Brezzi (LBB) theorem provides criteria for when the discretisation of the saddle-point problem has stable solutions [25]. A complete proof of the theorem may be found in [25]. This implies the basis vectors on the $\boldsymbol{V}$ and $\boldsymbol{Q}$ Hilbert spaces, defined for the velocity and pressure solutions, cannot be chosen arbitrarily. Choosing an unstable pair of basis vectors may lead to undesirable oscillations in the pressure solution [35]. To avoid this problem, there exist many pairs of stable basis vectors satisfying the LBB criteria, including the well-known Taylor-Hood family of elements. These include a pair of continuous piece-wise quadratic elements for the velocity and continuous piece-wise linear elements for the pressure. These are the basis vectors for the velocity and pressure that will be used for the simulations in this thesis.

3.3.3 Incremental Pressure Correction scheme (IPCS)

The IPCS may be broken down into three steps. The pressure gradient is treated explicitly in the one step, and implicitly corrected in another step. In such a scheme, one step is always reserved for the projection of a vector field onto a divergence free space [28]. We employ a splitting method by considering equations 3.1 and 3.2 separately. We assume a backward Euler scheme is used for the time integration. We compute a *tentative velocity* ($\bar{\boldsymbol{u}}^{n+1}$) by advancing equation 3.1 by a midpoint finite difference scheme in time,

$$\frac{\bar{\boldsymbol{u}}^{n+1} - \boldsymbol{u}^n}{\Delta t} = \frac{\mu}{\rho} \nabla^2 \bar{\boldsymbol{u}}^{n+1} - \bar{\boldsymbol{u}}^{n+1} \cdot \nabla \bar{\boldsymbol{u}}^{n+1} - \nabla p_*^{n+1} + \boldsymbol{f}^{n+1}, \tag{3.12}$$

where p_*^{n+1} is a guess for p^{n+1} and $\boldsymbol{u}^n$ is the previous computed velocity vector. After $\bar{\boldsymbol{u}}^{n+1}$ is obtained from equation 3.12 we may represent the difference between the actual pressure and the guessed pressure as δp^{n+1} which is obtained from,

$$\frac{\boldsymbol{u}^{n+1}-\bar{\boldsymbol{u}}^{n+1}}{\Delta t}=-\nabla\delta p^{n+1}, \tag{3.13}$$

Here, the corrected velocity ($\boldsymbol{u}^{n+1}$) is still unknown, so we use the incompressibility condition (refer to equation 3.2) applying to the corrected velocity, but not the tentative velocity. By taking the divergence of equation 3.13, we get the pressure Poisson equation, which gives us δp^{n+1},

$$\nabla^2\delta p^{n+1}=\frac{\nabla\cdot\bar{\boldsymbol{u}}^{n+1}}{\Delta t}, \tag{3.14}$$

Now the pressure correction step is performed.

$$p^{n+1}=p_*^{n+1}+\delta p^{n+1}, \tag{3.15}$$

Finally we perform the velocity correction,

$$\boldsymbol{u}^{n+1}=\bar{\boldsymbol{u}}^{n+1}-\Delta t\nabla\delta p^{n+1}, \tag{3.16}$$

and obtain the corrected velocity.

3.3.4 The weak formulation of the IPCS

To pose a weak problem for the Navier-Stokes solver to be implemented, we require a weak formulation of the IPCS. This reformulation shall be discussed in this section. We shall combine the general weak formulation of the Navier-Stokes equations as presented in section 3.3.2 with the general IPCS as discussed in section 3.3.3. We can accomplish this in three steps as shown below.

***Step* 1**: We shall combine the finite midpoint advanced momentum equation (equation 3.12) with the weak formulation of the momentum equation (equation 3.10) to obtain,

$$\begin{aligned}\int_\Omega \frac{\bar{\boldsymbol{u}}^{n+1}-\boldsymbol{u}^n}{\Delta t}\cdot\boldsymbol{v}\,dx+\int_\Omega\frac{\mu}{\rho}\nabla\boldsymbol{u}^n\cdot\nabla\boldsymbol{v}\,dx+\int_\Omega[\boldsymbol{u}^n\cdot(\nabla\boldsymbol{u}^n)]\cdot\boldsymbol{v}\,dx-\int_\Omega p^n\nabla\cdot\boldsymbol{v}\,dx\\ -\int_{\partial\Omega}(\frac{\mu}{\rho}\frac{\bar{\boldsymbol{u}}^{n+1}-\boldsymbol{u}^n}{\Delta t}-p^n\boldsymbol{n})\cdot\boldsymbol{v}\,ds=\int_\Omega\boldsymbol{f}\cdot\boldsymbol{v}\,dx,\end{aligned} \tag{3.17}$$

***Step* 2**: We subtract equation 3.1 expressed in terms of the tentative velocity and guessed pressure, $\bar{\boldsymbol{u}}^{n+1}$ and p_*^{n+1} from equation 3.1 expressed in terms of the corrected velocity and pressure, $\boldsymbol{u}^{n+1}$ and p^{n+1},

$$\frac{\partial\boldsymbol{u}^{n+1}}{\partial t}-\frac{\mu}{\rho}\nabla^2\boldsymbol{u}^{n+1}+\boldsymbol{u}^{n+1}\cdot\nabla\boldsymbol{u}^{n+1}+\nabla p^{n+1}-\boldsymbol{f}-\frac{\partial\bar{\boldsymbol{u}}^{n+1}}{\partial t}+\frac{\mu}{\rho}\nabla^2\bar{\boldsymbol{u}}^{n+1}-\bar{\boldsymbol{u}}^{n+1}\cdot\nabla\bar{\boldsymbol{u}}^{n+1}-\nabla p_*^{n+1}+\boldsymbol{f}=0, \tag{3.18}$$

which reduces to,

$$\frac{\partial\boldsymbol{u}^{n+1}}{\partial t}-\frac{\partial\bar{\boldsymbol{u}}^{n+1}}{\partial t}-\frac{\mu}{\rho}\nabla^2\boldsymbol{u}^{n+1}+\frac{\mu}{\rho}\nabla^2\bar{\boldsymbol{u}}^{n+1}+\boldsymbol{u}^{n+1}\cdot\nabla\boldsymbol{u}^{n+1}-\bar{\boldsymbol{u}}^{n+1}\cdot\nabla\bar{\boldsymbol{u}}^{n+1}+\nabla p^{n+1}-\nabla p_*^{n+1}=0. \tag{3.19}$$

We resolve the sum of the third and fourth terms in equation 3.19 as follows,

$$\begin{aligned}-\frac{\mu}{\rho}\nabla^2\boldsymbol{u}^{n+1}+\frac{\mu}{\rho}\nabla^2\bar{\boldsymbol{u}}^{n+1} &= -\frac{\mu}{\rho}\nabla^2[\boldsymbol{u}^{n+1}-\bar{\boldsymbol{u}}^{n+1}]\\ &= -\frac{\mu}{\rho}\nabla^2[-\Delta t\nabla\delta p^{n+1}] \text{ from equation 3.13}\\ &= \frac{\mu}{\rho}\nabla\Delta t[\nabla^2\delta p^{n+1}] \text{ commutativity of } \nabla \text{ and } \nabla^2\\ &= \frac{\mu}{\rho}\nabla\Delta t[\frac{\nabla\cdot\bar{\boldsymbol{u}}^{n+1}}{\Delta t}] \text{ from equation 3.14}\\ &= 0, \text{ from equation 3.2}\end{aligned} \tag{3.20}$$

to obtain,

$$\frac{\boldsymbol{u}^{n+1}-\bar{\boldsymbol{u}}^{n+1}}{\Delta t}+\boldsymbol{u}^{n+1}\cdot\nabla\boldsymbol{u}^{n+1}-\bar{\boldsymbol{u}}^{n+1}\cdot\nabla\bar{\boldsymbol{u}}^{n+1}+\nabla p^{n+1}-\nabla p_*^{n+1}=0. \tag{3.21}$$

As the gradient of a vector is always orthogonal to that vector, and the dot product of two orthogonal vectors is zero, equation 3.21 reduces to,

$$\frac{\boldsymbol{u}^{n+1}-\bar{\boldsymbol{u}}^{n+1}}{\Delta t}+\nabla p^{n+1}-\nabla p_*^{n+1}=0. \tag{3.22}$$

Taking the divergence of equation 3.22 and from equations 3.13 and 3.14 we obtain,

$$-\frac{\nabla\cdot\bar{\boldsymbol{u}}^{n+1}}{\Delta t}+\nabla^2 p^{n+1}-\nabla^2 p_*^{n+1}=0. \tag{3.23}$$

Rearranging the terms in equation 3.23, we may write down the weak formulation of the pressure correction step as follows,

$$\int_\Omega \nabla p^{n+1}\cdot\nabla q\,dx=\int_\Omega \nabla p_*^{n+1}\cdot\nabla q\,dx+\frac{1}{\Delta t}\int_\Omega \nabla\cdot\bar{\boldsymbol{u}}^{n+1}\cdot q\,dx, \tag{3.24}$$

***Step* 3**: A simple rearrangement of equation 3.22 gives the weak formulation of the velocity correction step,

$$\int_\Omega \boldsymbol{u}^{n+1}\cdot\boldsymbol{v}\,dx=\int_\Omega \bar{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{v}\,dx-\Delta t\int_\Omega \nabla[p^{n+1}-p_*^{n+1}]\cdot\boldsymbol{v}\,dx, \tag{3.25}$$

Equations 3.17, 3.24 and 3.25 give the weak formulation of the IPCS. We may now solve the Navier-Stokes system of equations through a sequence of three linearised weak problems.

3.3.5 Linearisation of PDEs using finite element methods

The linearisation of the weak formulation of the IPCS for the Navier-Stokes system yields a system of equations of the form,

$$A\boldsymbol{x}=\boldsymbol{b}, \tag{3.26}$$

where $A \in \mathbb{R}^{n\times n}$ is the coefficient matrix of the system $\boldsymbol{x}$ and $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$. Assuming the matrix A is invertible, we may easily write the solutions of the linear system as,

$$\boldsymbol{x} = A^{-1}\boldsymbol{b}, \tag{3.27}$$

This is generally not the case in the Navier-Stokes problem, as matrix A is sparse with $O(n)$ non-zero entries, while A^{-1} has $O(n^2)$ non-zero entries [7]. Therefore, the conventional methods to compute the inverse of matrix A, such as Gaussian elimination, are extremely inefficient on such systems (they require $O(n^3)$ operations). We require an iterative method by guessing the solution $\boldsymbol{x}_0$, and computing a sequence of k guesses $(\boldsymbol{x}_k)$ until the solution converges at $\boldsymbol{x}$. A well-studied iterative solver for this problem includes the Biconjugate Gradient Stabilised (BiCGStab) algorithm [65], which will be initially used for the three linear problems derived. This particular configuration may lead to a failure of convergence of the solution of a non-symmetric system linear system. A popular choice for non-symmetric systems is the Generalized Minimal Residual (GMRES) method, alongside the Incomplete LU factorisation (ILU) preconditioner. The ILU preconditioner has been shown to perform well for a broad range of test problems including turbulent flows using the Navier–Stokes equations discretized on stretched meshes [53]. This is a close description of the system under consideration and we may make use of this configuration in the case that the earlier considered configuration fails. We have thus introduced a family of solver-preconditioner configurations to solve the assembled linear system of equations.

So far, we have discussed the mathematical requirements to create an environment where we may only model stationary objects surrounded by a constant fluid flow. For a dynamic system, where the objects are permitted to move freely through the domain, we require a method to describe the fluid-structure interactions. The next section briefly introduces the Arbitrary Lagrangian-Eulerian method to allow objects to move in a meshed space.

3.4 The Arbitrary Lagrangian-Eulerian (ALE) finite element method for fluid-structure interactions

We have succeeded in creating a theoretical framework for our simulations, which will include generating a mesh to initialise three fish at certain positions in Ω. To simulate a fluid environment allowing fish to interact with each other, requires a minimalistic implementation of the fluid-structure interactions. To this end, we require an appropriate kinematical description of the fluid flow that allows distortions in Ω. As we have approximated the domain as a finite element mesh, this choice effectively determines the relationship between the distorting domain and the equivalent deformation of the enmeshed space. Two classical descriptions of motion in continuum mechanics include the Lagrangian and Eulerian descriptions.

1. Lagrangian algorithms: These algorithms allow a powerful mapping between individual nodes on a mesh with the associated moving particle in the domain. However, these algorithms cannot handle large deformations in the mesh without resorting to frequent re-meshing techniques.

2. Eulerian algorithms: Here, the finite element mesh is fixed and the domain moves with respect to this mesh. The advantages include the ability of such methods to handle large distortions, but at the cost of lower resolution of the fluid flow features.

One technique that, to some extent, succeeds in combining the best features of both kinematical descriptions of motion is the Arbitrary Lagrangian-Eulerian method (ALE). The technique allows larger freedom in moving the mesh: greater distortions are managed due to the Eulerian description with greater resolution due to the Lagrangian description [18]. Meshes with larger resolutions require the mesh to be smoothed after the ALE method is applied. There is a trade off between the number of times a mesh is smoothed over after the ALE application and the computational expense. We found that the mesh when smoothed over 1000 times for each application of the ALE method was a reasonable computational overhead for our application.

Although the ALE method provides an improved ability for distortions within the mesh, the distortion limit is still inadequate for the nature of the simulation being created. A complete distortion of the mesh is observed when a fish is moved 0.13 m (see Figure 3.4) calculated from the centre of the fish. To resolve this issue, we require a process, known as re-meshing, where a completely new mesh is generated, with the updated fish coordinates and the function spaces interpolated from the old mesh to the new one. In this context, we define an interpolation space as an intermediate space between two Hilbert spaces. The two Hilbert spaces are continuously embedded in a particular topological vector space, such that a mapping exists between the two spaces and the family of intermediate spaces [3].

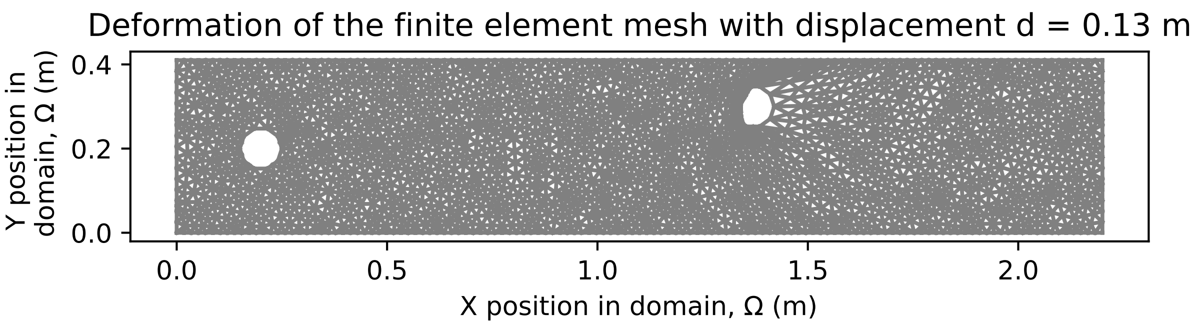

Deformation of the finite element mesh with displacement d = 0.13 m
Y position in domain, Ω (m)
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
X position in domain, Ω (m)

FIGURE 3.4. An illustration of the deformation of the mesh constructed on a domain Ω. Two fish are initialised at (0.2, 0.2) and (1.5, 0.3), with the latter fish displaced 0.13 m in the negative x-direction, creating a deformation in the mesh. The mesh resolution has been set to 64 triangular elements.

We generate a new mesh after the fish has moved 0.05 m, and use the fenicstools [46] library to interpolate between non matching meshes. This is achieved by projecting the solution functions for the pressure and velocity on the previous distorted mesh to new pressure and velocity solution spaces on the non-deformed mesh using the interpolation function in [46].

3.5 The environment setup

We present the fluid simulation environment programmed in the Python language (version 3.7.6). We use the FEniCS (version 2019.1.0) [1] platform to set up the Navier-Stokes problem. The DOLFIN [1] library implements the meshes, function spaces and finite element assembly. The in-built Unified Form

Language in FEniCS allows us to express the finite element weak formulation of the IPCS using notation similar to the mathematical notation used in this chapter. The three linear weak problems (as defined in section 3.3.4) are solved using the BoomerAMG [77] preconditioner and the biconjugate gradient stabilized method. All preconditioners and solver methods applied are available as functions in the FEniCS suite. All simulations were executed on a Lenovo nx360 m5 compute node, with a 2.4 GHz Intel E5-2680 v4 CPU, as part of the BlueCrystal Phase 4 compute cluster at the University of Bristol, UK.

One of the simplest models of animal motion was presented by Vicsek [71], where animals are described as point particles with constant speed, and whose direction is updated at discrete time increments. Our fish model deviates from this proposed description by describing the fish as circles in 2D, naturally extending to cylinders in 3D. All fish are set to have a radius, $r = 0.05\ m$. This means that the fish is 10 cm long. Denil [15] has reported the speed of a 35 cm long trout (*Salmo fario*) to be 3.5 ms^{-1}. This translates to speeds of 10 body lengths per second. As trouts are known to shoal in the oceans, we use the trout as a model to set the fish speeds in our environment. We set the constant speed of the fish to be 1 ms^{-1}. We allow the fish to move in any direction in the domain with the obvious domain boundary constraints.

We take inspiration from the benchmark study on computations of a flow around a cylinder [62], and add two extra circles (modelled fish) initialised further along in the domain, in the x-direction. We construct a domain, $\Omega = [0,2.2]\times[0,0.41]$ (m^2), with two fish initialised at random positions, with the lead fish holding its station at (0.2,0.2) - refer to Figure 3.5, for a template of the geometry. We assume the fluid density, $\rho = 1\ kgm^{-3}$, and the dynamic viscosity, $\mu = 0.001\ Nsm^{-2}$. We set the effect of the external forces, $\boldsymbol{f} = 0\ N$. On the walls, we impose the no-slip Dirichlet boundary condition, $\partial\Omega_{\text{wall}}|_{\boldsymbol{u}=0}$. On the outlet, we impose the natural boundary condition, $\partial\Omega_{\text{outlet}}|_{\boldsymbol{u}=0}$. On the inlet boundary, we impose the Dirichlet boundary condition, $\partial\Omega_{\text{inlet}}|_{\boldsymbol{u}=\boldsymbol{g}}$ where $\boldsymbol{g}$ is the parabolic inflow profile,

$$\boldsymbol{g}(0,y) = \left(\frac{4\boldsymbol{U}y(0.41-y)}{0.41^2}, 0\right), \tag{3.28}$$

where $\boldsymbol{U} = 1.5\ ms^{-1}$ is the maximum velocity. We consider a time-step of 0.001 s, with the total number of time steps as 5000, for our simulations.

With the maximum velocity $\boldsymbol{U} = 1.5$, the mean velocity is, $U_{mean} = \frac{2}{3}\cdot\boldsymbol{U} = 1$. The characteristic length of the flow configuration, $L = 2\cdot r = 2\cdot 0.05 = 0.1$. We may then calculate the Reynolds number (Re) of the flow,

$$Re = \frac{1\cdot 1\cdot 0.1}{0.001} = 100. \tag{3.29}$$

We generate the mesh corresponding to the geometric representation as shown in Figure 3.5 and declare the velocity inflow profile. We define the trial and test functions, then use these to define the weak reformulation, as presented in section 3.3.4. We write a timestepping loop for each small increment in time to compute the velocity and pressure solutions on the Hilbert spaces, as defined earlier. Within the timestepping loop, we use the in-built weak reformulation assembly function in DOLFIN to assemble the matrices for the three linear weak problems. We only assemble the right hand side vectors and apply the boundary conditions. We use FEniCS' *solve* function, with the BiCGStab algorithm as the argument to the function to obtain the velocity vectors and pressure solutions. We save these pressure and velocity values as CSV files to be processed before the data is used as input to a navigational control algorithm. We

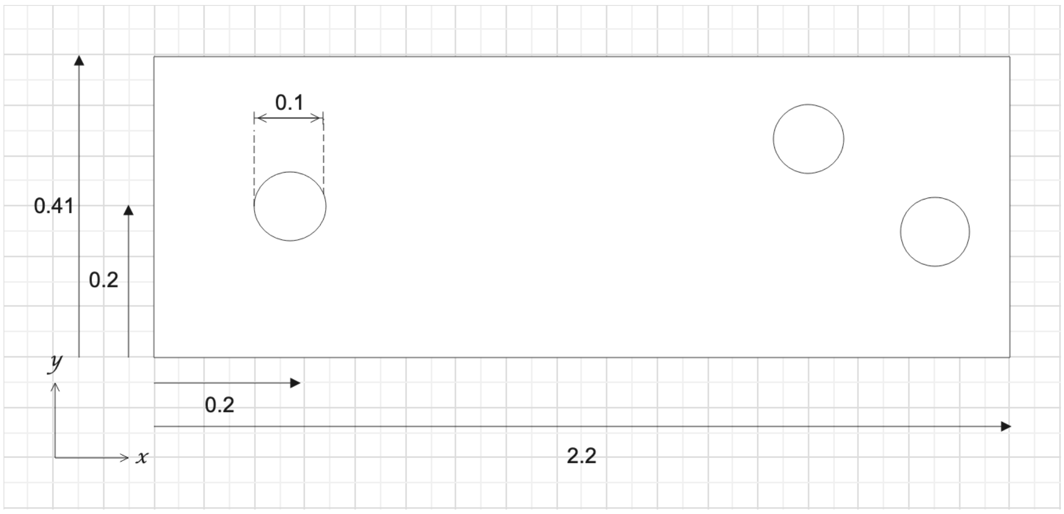


FIGURE 3.5. The geometric set-up of the simulation environment. The lead fish is initialised at (0.2, 0.2) with the following fish initialised at random positions in the domain. All values are in metres.

catch any collisions between the lead fish and the following fish, or collisions between the following fish by setting future displacement values to zero when the Euclidean distance is below 0.2 m.

3.6 Results

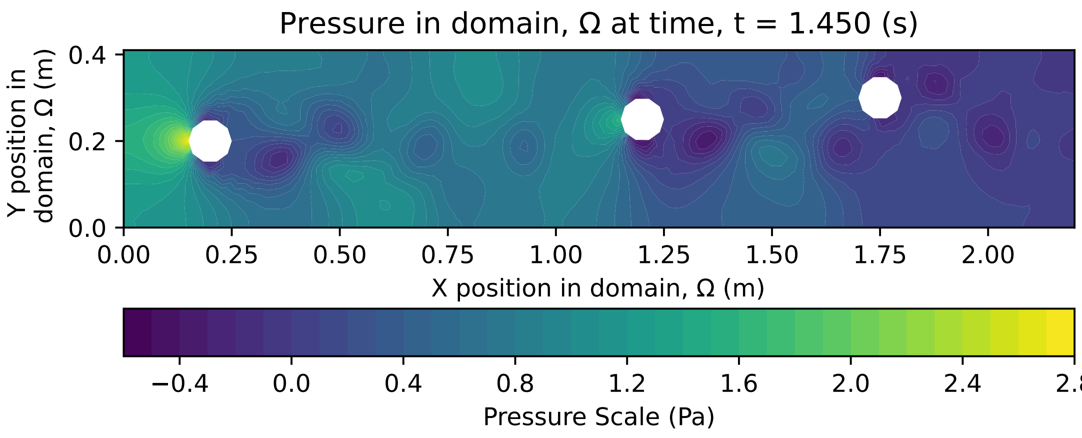


FIGURE 3.6. The pressure plot at time $t = 1.45$ s. The KVS is formed in the domain Ω, with three stationary circular fish. The lead fish is positioned at (0.2, 0.2).

During the initialisation of the simulation environment, we first notice two symmetrical vortices being formed immediately behind the fish on each side of the wake. With gradual increases in time, these structures progressively elongate downstream, quickly at first and slowly thereafter, creating a re-circulation region. Qualitatively, this phenomenon is in good agreement with past experimental studies [16]. We observe a steady pattern approximately starting from $t = 200$ time steps. This steady pattern lasts till $t = 1230$ when the first separation occurs. It is interesting to observe that this separation is not absolute and, within a few time steps, the separation is absorbed back into the steady wake. At $t = 1280$, the re-circulation region is completely destabilised and the alternating vortices are formed. We notice that the KVSs are generated without any external perturbations after a long period of simulated time, wherein the wake remains steady. A flow with such a Reynolds number exhibits a periodic KVS shedding as observed at time, $t = 1.45\ s$ in Figure 3.6. We also observe the alternating vortices being generated behind the following fish. For each time step, we generate the velocity and pressure values across the domain and record this data in individual CSV files. The files are further used as input to our control algorithm. All data generated must be processed before use. We discuss the pre-processing step in brief in the next section.

3.6.1 Pre-processing the pressure data

We include the velocity CSV files as a multi-dimensional dataset, available from the project repository for applications in future work. For this thesis, however, we constrain our analysis to the pressure data obtained through our simulation framework. To be able to conduct any meaningful analysis on the data, we load the dataset $D \in \mathbb{R}^{n\times m}$ from each individual CSV files to a Pandas [44] dataframe. Each timestep generates a dataset D with $n = 2404$ rows and $m = 3$ columns. The columns refer to the x, y coordinates and the corresponding pressure values.

3.7 Summary

We have successfully simulated our fluid environment with a comprehensive discussion on the mathematical formulations required for the the linearisation of the Navier-Stokes problem. We demonstrate the formulation of vortices in the domain and compare the results qualitatively to previous numerical studies for validation. The next chapter focuses on the in-house control algorithm we designed for multiple agents in the simulated environment.

Chapter 4

Control algorithms for multi-agent systems

The goal of this chapter is to present a new navigational control algorithm developed for the fish in our environment. The aim is to write a set of rules allowing the fish to interact with the simulated environment and make decisions based off the local pressure inputs. We inspect Figure 3.6 closely to find that the individual vortices formed in the KVS have a higher vortex strength closer to the source and get weaker further away from the source. We find that the individual vortices correspond to local low pressure systems, where the centre of the vortex has the lowest pressure in the local system, with the pressure incrementally increasing outward. We aim to construct algorithms where the following fish navigate towards the centre of these local low pressure systems on each iteration, and gradually navigate themselves towards the lead fish.

4.1 A KD-Tree based lowest pressure point search algorithm

We allow the Navier-Stokes system to be simulated for the first 1450 time steps to initialise the simulation environment and generate the KVSs. The control shall be applied only after these first 1450 time steps. We obtain the pressure data for all points in the domain and save the data for each time step to a CSV file, as detailed in the pre-processing step earlier. Here, we require an efficient algorithm to obtain a sample of points closest to the following fish to use as input to the theoretical lateral line sensor. This is closely related to the well-studied k-nearest neighbours problem, encountered in many machine learning domains such as classification and regression.

One such algorithm proposed for the k-nearest neighbours problem is the KD-Tree algorithm [43], which we implement through the SciPy [72] library. The algorithm creates a query point tree, which lists the nearest points given a specific set of coordinates, within a given maximum distance to the coordinates. Using this query point tree, we inspect the locus of points within 0.06 m of the centre of the following fish for the nearest coordinates with the lowest pressure. As the radius of the fish is set to 0.05 m, setting the maximum distance to search for the nearest neighbours to 0.06 m restricts the fish to only receive inputs in the local environment. This is realistic to the extent that fish in nature do not have access to

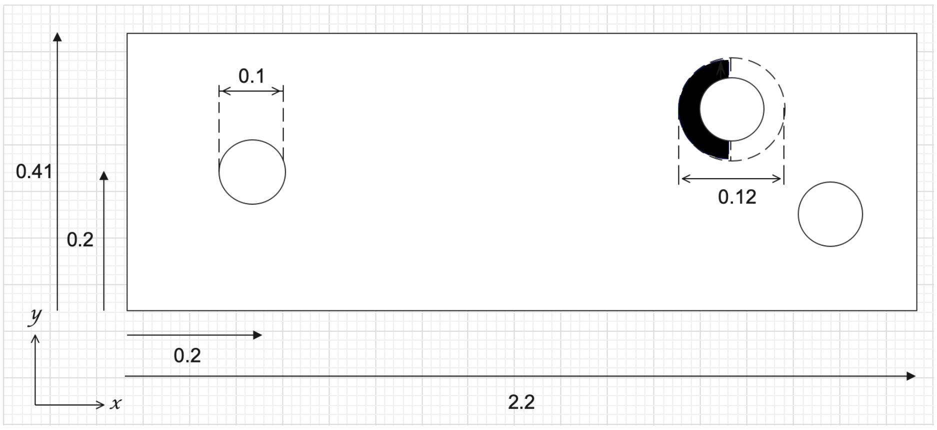


FIGURE 4.1. The cKD-Tree algorithm implemented for the front half of the fish. The area shaded in black forms the region of pressure points serving as input to the fish. All values are in metres.

```
Algorithm 1 An algorithm to check if a fish is in the domain
Input: Dataframe of pressure values in the neighbourhood of the following fish, current fish position
Output: boolean check_fish_in_domain
Initialise check_fish_in_domain == False
while check_fish_in_domain is False do
    Locate min pressure in dataframe and set the fish destination to the x and y coordinates corresponding
      to the minimum pressure
    fish_destination_position = [min_pressure_xcoord, min_pressure_ycoord]
     vector_translation = fish_destination_position - current_fish_position
     Need to check if new fish position still inside domain. If yes, proceed, if no, need to choose next minimum.
     Check if circle fully intersects rectangle i.e. check whether circle still in domain.
     Compute area_of_intersection
     If area of intersection ≠ area of circle then some part of circle exists outside domain
     check_fish_in_domain = boolean(area_of_intersection == ($\pi \cdot radius\_of\_fish^2$)
end
return boolean check_fish_in_domain
```

global hydrodynamic data and their decision making process is also based off local inputs. When the fish detects the nearest point with the lowest pressure in its neighbourhood, it computes the vector bearing towards this point. We implement an in-house algorithm here to check whether the fish would still be in the domain if it moved to this new position (see Algorithm 1). If the fish is still inside the domain at this new position on each time step, the fish moves 0.001 m towards this point (constant speed of $1m \cdot s^{-1}$) until it has reached the new position. If this condition is not satisfied, the algorithm keeps searching for the next lowest pressure position within the domain. We assume that the fish does not wait to reach the new position before triggering the control algorithm. A new KD-Tree is calculated based on the new position and the entire process is repeated. In this work, the KD-tree based policy solely controls the two following fish, where the lead fish remains stationary. In an accelerating frame of reference, we could view these dynamics as the following fish accelerating towards the lead fish, while the lead fish maintains a constant velocity.

In order to evaluate the robustness of the proposed algorithm, we rely on the robustness with respect to initial conditions metric. We analyse whether the fish takes finite time to station itself within a Euclidean distance of 0.2 m of the lead fish. We randomise the positions of each fish for every simulation run. Due to time and computational expense constraints, we simulate the system and show results from 3 randomly picked runs to demonstrate the robustness of our algorithm with changing initial conditions.

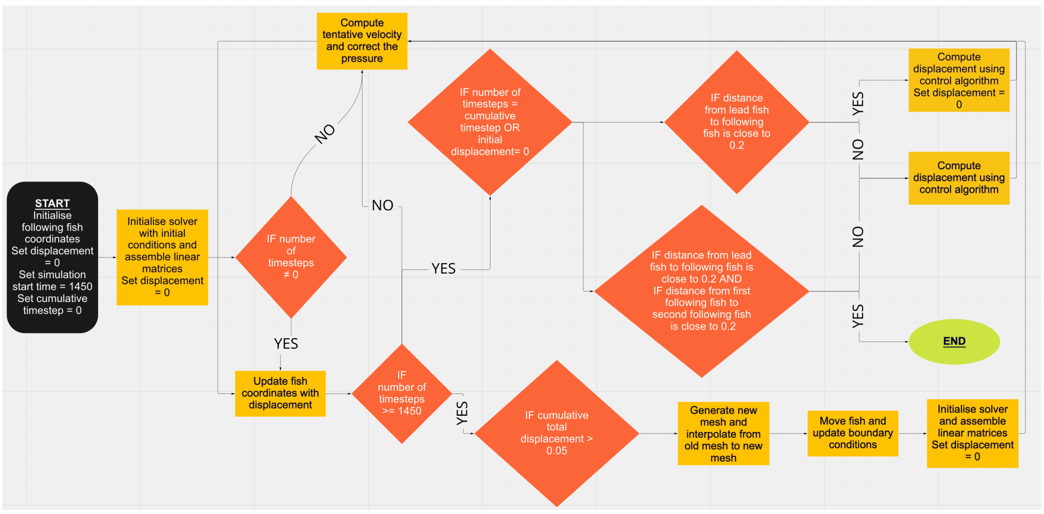

START
Initialise following fish coordinates
Set displacement = 0
Set simulation start time = 1450
Set cumulative timestep = 0
Initialise solver with initial conditions and assemble linear matrices
Set displacement = 0
IF number of timesteps ≠ 0
NO
YES
Compute tentative velocity and correct the pressure
NO
Update fish coordinates with displacement
IF number of timesteps >= 1450
YES
IF number of timesteps = cumulative timestep OR initial displacement= 0
YES
IF distance from lead fish to following fish is close to 0.2
YES
NO
Compute displacement using control algorithm
Set displacement = 0
Compute displacement using control algorithm
IF distance from lead fish to following fish is close to 0.2 AND
IF distance from first following fish to second following fish is close to 0.2
NO
YES
END
IF cumulative total displacement > 0.05
Generate new mesh and interpolate from old mesh to new mesh
Move fish and update boundary conditions
Initialise solver and assemble linear matrices
Set displacement = 0

FIGURE 4.2. A flowchart of the integration of the simulation environment and the control algorithm applied to the following fish in the domain.

4.2 Initial testing and bug fixes

We tested the algorithm on set initial fish coordinates to determine the extent of the robustness. We found four major bugs that were caught and debugged accordingly. There is one major bug that has not yet been resolved. We present a brief description of all five issues along with the fixes in this section.

Initial tests of this algorithm (tests conducted for the first 50 ms of the simulation) indicated that the lowest pressure point was behind the following fish, causing the fish to move further and further away from the lead fish. We require the control algorithm to demonstrate the emergence of shoals in fish and a close grouping of fish is expected. We propose a simple fix to this problem: restricting the cKD-Tree algorithm to create a query point tree. Using the coordinates found only in the front half of the fish (see Figure 4.1 for a graphical representation) - we choose the lowest pressure coordinates from this list. We observe that each iteration gradually reduces the Euclidean distance between the lead and following fish.

We conducted a test to allow the fish to change their decisions every 0.02 s. We initialised the simulations till $t = 20$ timesteps, after which the control algorithm was deployed. We observed that the BiCGStab-BoomerAMG solver-preconditioner configuration threw errors after 1320 iterations with this configuration, implying that the solution failed to converge before the fish made a new decision. This was attributed to the short time between decisions taken. We allow each decision to be completed before a new decision is taken, allowing extra time for the Navier-Stokes approximated linear system to converge.

Even with this fix, we observe a failure of convergence of the BiCGStab-BoomerAMG solver-preconditioner configuration after $t = 1520$ timesteps when the fish are initialised further along in the domain. We traced the failure of convergence problem to the rounding errors generated during the update of the current position of the fish. We rely on the NumPy [29] library for arithmetic operations which leads to an inexact representation of the true values. When the fish coordinates are updated with these inexact representations, we observe a truncation error, which we believe is the main cause of convergence failure. We aim to mitigate these issues by inspecting the vector bearings generated by the control algorithm and perform the rounding to an appropriate number of decimal places.

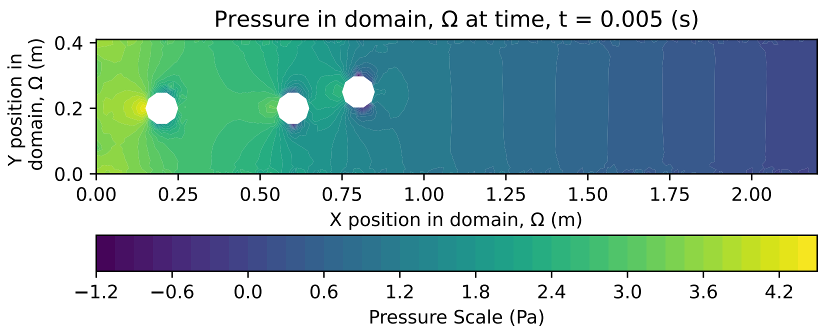
The truncation issue was resolved, yet the program fails to converge if the distance between the lead fish and the initial positions of the following fish are more than 0.4 m. We attribute this to a potential bug in the implementation of the remeshing, contributing to a solver convergence failure. For shorter Euclidean distances, our program works sufficiently as demonstrated in section 4.3.

We initially set a 'hold station' condition wherein the fish position was not updated when the Euclidean distance between the lead fish and the following fish was less that 0.2 m. After these fixes, we observed that the 'hold station' condition imposed on the fish did not work and the program crashed when the following fish got too near and intersected the lead fish. We compute the Euclidean distance between the lead fish and the first following fish, and, similarly, between the first following fish and the second following fish. We set a 'hold station' condition for the first and second following fish with a threshold value of 0.2 m. This second condition was devised to end the algorithm as the definition of a formation was attributed to existence of a constantly maintained distance between each fish. We attributed the errors, obtained in the implementation of the condition, to the lack of absolute tolerance set on the threshold value. We set the absolute tolerance to 0.001, so the distance between fish is not exactly 0.2 m which resembles the slight discrepancies in distances between individuals in actual fish shoals. We have successfully managed to integrate the control algorithm with the implemented Navier-Stokes solver and we present the complete working of our implementation in Figure 4.2.

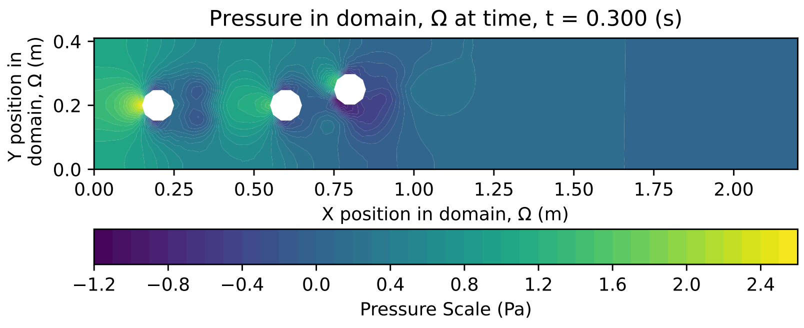
4.3 Results

This section is dedicated to demonstrating the implementation of the algorithm and the observed emergent schooling behaviour in the three agents considered. Figure 4.3 shows snapshots of the four different stages in the execution of the simulation with the following fish initialised at $(0.6, 0.2)$ and $(0.8, 0.25)$. The uppermost figure shows the state of the system at the start of the simulations at time $t = 0.005$ s. The next figure shows a steady wake with the re-circulation zone at time $t = 0.3$ s behind the lead fish. Here, the control algorithm has not yet been applied. The third figure from the top shows the beginning of the formation of the KVS. Note the alternating periodic vortices formed behind the lead fish. These low pressure pockets will inform the fish navigation once the control algorithm is applied at time $t = 1.45$ s. The bottom figure shows the state of the system at time $t = 1.684$ s, where the three fish have navigated through the fluid and have come to rest behind the lead fish.

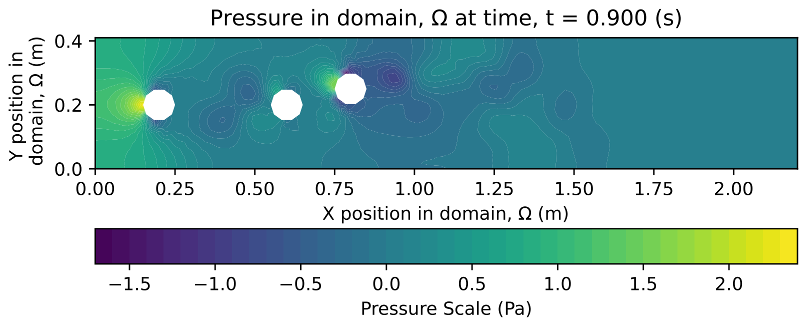
We apply the robustness to initial positions metric to test that the simulation terminates in finite time. We simulate the system and demonstrate the reduction in the Euclidean distance between the lead and first following fish. This is also shown for the distances between the first and second following fish (see Figure 4.4 to see the convergence of the distances to ≈ 0.2 m for 3 randomly picked runs). We simulate the system till $t = 1450$ time steps which is represented by the flat horizontal line till time $t = 1.45$ s in Figure 4.4. Right after the control algorithm is applied, we observe a linear decrease between the lead and first fish, and a rapid decrease between the two following fish. On occasional runs, this is reversed after some time, with the distance between the following fish increasing sharply before dropping and converging to



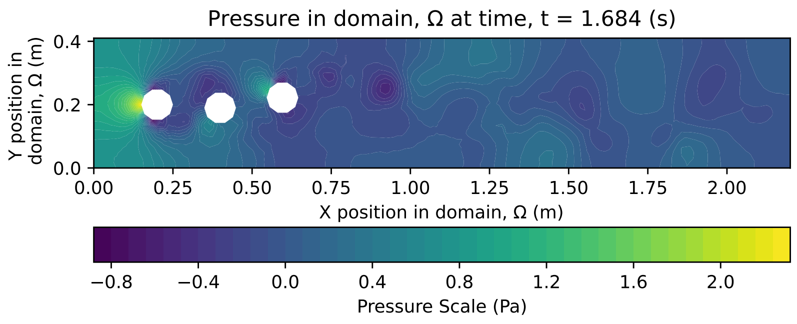
(a)



(b)



(c)



(d)

FIGURE 4.3. Snapshots of the system through time with the following fish initialised at (0.6, 0.2) and (0.8, 0.25). a) The initial state of the system at $t = 0.005\ s$. b) The formation of the steady wake with the re-circulation zone at $t = 0.3\ s$. c) The alternating periodic vortices formed at $t = 0.9\ s$. d) The final state of the system with the emergence of a fish shoal at $t = 1.684\ s$.

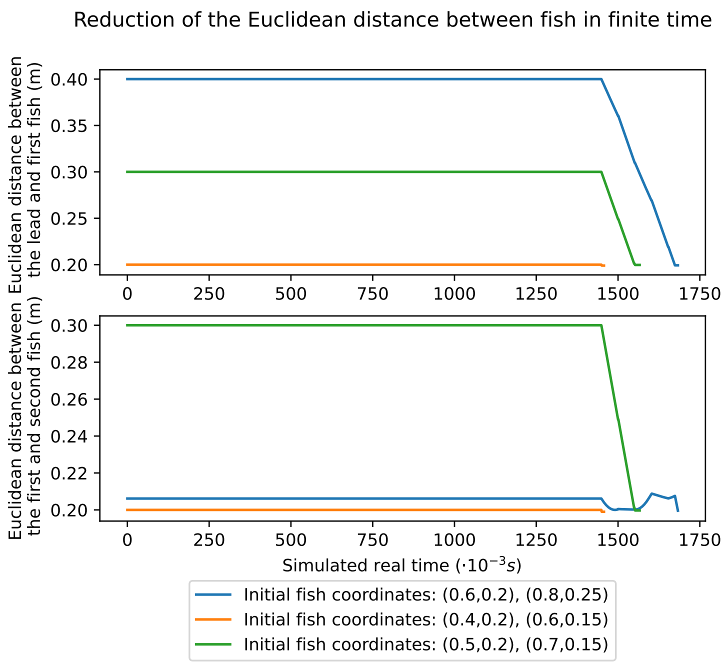


FIGURE 4.4. The convergence of the Euclidean distance between the fish to 0.2 m in finite time. The upper plot shows the reduction in the Euclidean distance between the lead fish and the first following fish. The lower plot shows the reduction in the Euclidean distance between the first and second following fish. This convergence is shown for various initial fish positions.

$\approx$0.2 m. We note here that the emergence of the shoals may not take place at the same point in time for each run. This behaviour is to be expected, as fish schools depend on the initial distance between fish. The larger the distance between the fish, the longer it takes for the shoal to form. We observe that this expected behaviour only holds true for the distance reduction between the lead and first following fish. There appears to be no such correlation between the first and second following fish. We may attribute this to the no-collision rule applied to the fish, where ≈ 0.2 m distance is maintained between the fish at all times. This rule appears to supersede the control applied to each individual fish in the fluid flow. It is important to note that the following fish also produce vortices, which may theoretically be detected by other following fish. This allows our model to be scalable across a large multi-agent system.

4.4 Summary

This chapter demonstrated the construction and implementation of a novel KD-Tree based control algorithm for multi-agents swarms based off local pressure inputs. We show that this algorithm can be integrated satisfactorily in our fluid environment. We highlight the bugs found during the initial testing of the algorithm and potential fixes for these issues. We show that the Euclidean distance between the three fish converges to ≈ 0.2 m in finite time for 3 randomly simulated runs for different initial fish coordinates. The fact that the following fish still produce vortices after the control algorithm is applied allows the scalability of our algorithm over a large multi-agent system.

Chapter 5

Discussion

The contributions of this thesis, especially the work on integrating our simplistic control algorithm with a ‘real time‘ simulation environment, are intended to be foundational work for future lateral line sensor based navigation research. This chapter serves as a comprehensive discussion of the results obtained and addresses the limitations of the simulation environment and the proposed control algorithm.

5.1 Simulation environment

The construction of an appropriate simulation environment permitted us to formulate a novel control algorithm allowing individual fish to shoal together. This section summarises the simulation construction and attempts to justify the various modelling decisions involved. We discuss the results obtained from the environment construction. We comment on the various limitations of the environment.

5.1.1 Construction of the environment

The environment was setup in a 2D Euclidean space with boundary conditions on the walls, inlet and the outlet, where the fluid was assumed to be Newtonian and incompressible. This assumption allowed us to formulate a well-accepted fluid flow model through the Navier-Stokes equations. There are many different numerical frameworks in the literature which may be applied to approximate the Navier-Stokes problem. Two of the main techniques are the finite element method and the finite difference method. The finite element method has some key advantages over the finite difference method. Some of these key advantages include: the ease of application of irregular geometries to the problem, the construction of non-uniform meshes (where appropriate), and the suitable application of boundary conditions. With the finite element method, the structure and its finite element closely resemble one another. The body under consideration in the fluid flow may have an arbitrary shape, load and support conditions [19]. These factors influenced the choice of this method for spatial discretisation of the domain, Ω.

Within the finite element method, a key modelling decision is the choice of the particular finite element. Figure 3.2 shows the commonly used 2D and 3D finite element types: the line, triangle, quadrilateral, tetrahedron and hexahedron elements. We restrict our choices to the 2D elements, due to the 2D nature of the domain under consideration. Sevilla *et al.* [64] conducted a comparative study on different finite element types and their effects on the solution in a compressible fluid flow. The authors reported that the linear elements, employed for the mesh on their domain, generated an entropy layer which warped the solution in the vicinity of the bluff body, situated in the flow. As the computational expense increases with higher order elements, and there have been reports of the Navier-Stokes approximated solutions being warped in the neighbourhood of the bluff bodies in the fluid, a triangular finite element was the ideal choice for the finite element method employed in this thesis.

Then, we rewrote the Navier-Stokes equations into the weak formulation and combined this reformulation with the IPCS. We have already discussed the spatial discretisation techniques applied. We now discuss the temporal discretisation techniques necessary to formulate our simulation environment. Two main temporal discretisation techniques are the backward differencing schemes (e.g. backwards Euler scheme) (BDSs) and the Crank-Nicholson scheme. The BDSs work by approximating all terms in the momentum equation (equation 3.1) at the $k+1$ time-step. The Crank-Nicholson scheme, however, discretises time by considering the time-step $k+\frac{1}{2}$ between times k and $k+1$. A main advantage of using the BDSs over the Crank-Nicholson scheme is the unconditional stability of the scheme with respect to growing or oscillatory solutions. Any Δt used in the BDSs will lead to unconditionally stable solutions while only specific upper bounds on the value of Δt permit stable solutions in the Crank-Nicholson scheme. This existence of stable solutions, regardless of bounds placed on the value of Δt, influenced the decision to choose the BDS (the backwards Euler scheme in particular) as the preferred temporal discretisation technique.

This leads to the temporally discretised set of Navier-Stokes equations. This system could be solved by setting up a large, single coefficient matrix and solving for the velocity and pressure simultaneously. This ‘coupled’ approach, although known to be robust and stable, suffers from a large demand on computer memory. This necessitated the development of segregated methods, where the velocity and pressure coupling is split up and an iterative method is applied. A main family of splitting methods is the incremental and non-incremental pressure or velocity correction schemes. We chose the incremental pressure correction scheme, due to its improved accuracy over the other splitting schemes, with the higher computational expense trade-off.

5.1.2 Simulation environment results

We observed two symmetrical vortices formed behind the lead fish at $t = 200$ time steps, with the recirculation region destabilised at $t = 1280$ time steps. Persillon and Braza [52] report similar observations in their study and add that the generated vortices are schematic representations of rotational motion superimposed on a stretched ellipse. Note the topological similarities between the geometries of the vortices in Figures 5.1 and 3.6, which validates our simulation environment.

We refer back to the choice of a triangular finite element over a linear element. From the observations in [64], the observed concentric geometries for the alternating vortices may not have been produced at all, or may contain substantial warps, such that individual vortices may not be distinguished in the flow. As the existence of these vortices are crucial to the implementation of the proposed navigational control algorithm, this further validates our choice of finite elements.

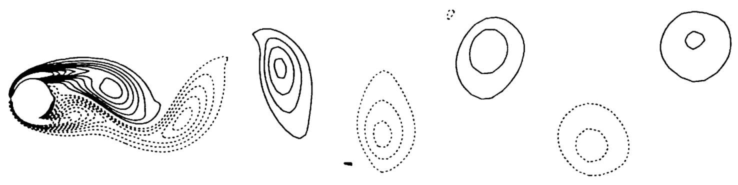


FIGURE 5.1. Vorticity plot for Re = 100. Figure adapted from the results from the numerical simulations of the Navier-Stokes equations in [52].

5.1.3 Limitations of the environment

We highlighted the importance of creating a suitable environment to test navigational algorithms in 'real time' conditions. We determine three main assumptions and limitations of the simulation environment, including the modelling assumptions in the Navier-Stokes solver, the uncertainty around the numerical stability conditions and the unreliability of the algorithm to correctly classify and distinguish between vortices produced by different sources.

We have previously discussed the role of the Navier-Stokes equations in a fluid modelling setting. It is vital to understand the modeller's decisions concerning the free parameters in these equations. We assumed the effect of the external forces on the fluid element and the modelled fish to be zero. This naturally implies that the weight of the fish equals the buoyant force experienced. This is a significant assumption, as the net force experienced by a body largely contributes towards its velocity and acceleration. The effects of a fluid element and objects with non-zero net force would need to be explored in a future study.

The stability of the numerical computation techniques applied depends on many factors. One of the most important factors is the Re. We have set the $Re = 100$ as a constant due to the set parabolic velocity inflow profile. Changing the Re changes the dynamics of the fluid under consideration. However, studies report a higher Re may result in numerical divergence in the discretised Navier-Stokes problem [11]. The numerical solutions stabilise for low dt and Re and larger dx values. The trade off is that the simulations run much slower with smaller dts, while larger dx values lead to errors in the simulation [11]. Future work could include the simulation environment being tested for multiple inflow profiles, leading to various tested Re values. This could also be considered a separate metric to test the robustness of the proposed algorithm. This unpredictability of the behaviour of the numerical stability conditions is exacerbated by the lack of detection of obstacles and the incapacity of the fish to distinguish between vortices from different sources.

We have proposed an extremely simple simulation environment with the only interactions between different fish. This is an incomplete model of a fluid environment as other obstacles exist, which may also produce vortices leading to low pressure zones. We believe this would lead to conflicting inputs to the lateral line, potentially driving the fish away from the shoal. This has been mitigated by studies considering the vortex shedding frequencies requiring the study of the velocity in the neighbourhood of individual fish [39]. Fish possess the ability to detect and distinguish between different vortex shedding frequencies [39]. This is in agreement with current biological understanding where the input to the fish brain consists of CN along with SN inputs, as discussed earlier. Although the problem of the in-feasibility of access to the global hydrodynamic data still exists, this ability to distinguish between different vortices could be a valuable addition to the proposed control algorithm.

5.2 KD-Tree based control algorithm

This section discusses the construction of the algorithm, the modelling decisions that were involved and brief justifications for the choice or certain parameter values or methods. We briefly discuss the results obtained, before discussing the limitations of the algorithm.

5.2.1 Construction of the algorithm

We allow the simulation environment to run to allow a distinct KVS to be formed in the domain, before the control algorithm is applied. The key concept of the algorithm is the ability to navigate the fish based solely off the neighbourhood pressure values. We relate this to the well-studied k-nearest neighbour problem. For agent-based simulations with interactions between multiple agents, fast response times are essential and brute force methods are often inefficient. Space-partitioning methods resolve this issue by requiring less parameters to be tuned [47]. These space-partitioning algorithms for the k-nearest neighbour problems operate by dividing the space recursively at each iteration, by employing a splitting rule. Hence, the KD-Tree space partitioning algorithm was the base algorithm chosen to list the nearest pressure values within a given radius.

5.2.2 Results from the algorithm implementation

The initial tests of the algorithm raised several fundamental issues with the construction of the algorithm. Notably, the lowest pressure point in the vicinity of the fish was behind the fish, and the implementation of the algorithm meant the fish would move in the positive x-direction of the domain away from the lead fish, which is the opposite of the expected following fish behaviour. This was resolved by restricting the KD-Tree to construct a list of points on the front half of the fish. This decision has been backed by studies on bio-inspired robots, where the pressure sensor is towards the front end of the fish and the authors claim that the arrangement detects KVS characteristics [17, 34, 76, 78]. Upon implementation of the algorithm with this fix amongst others, we observe a sharp decrease in the Euclidean distance between the lead and first following fish. We observe a sinusoidal curve in the Euclidean distance between the first and second following fish. This is attributed to the non-collision rule applied between fish - i.e., there must be a ≈ 0.2 m gap between two agents at all times. We observe that once the first following fish has held station, the second following fish sharply follows.

5.2.3 Limitations of the algorithm

The preceding chapter presented a new unsupervised learning based navigational approach for multi-fish systems in a fluid environment. The algorithm aims to navigate fish through a fluid environment, depending on the low pressure vortices created in the wake of a swimming fish as an input to a pressure sensor. The algorithm searches for the minimum pressure in the vicinity of each fish and directs the fish towards this low pressure point. Given the assumptions presented, including a finite domain and constant non-fluctuating fish speeds, the algorithm is guaranteed to end in finite time. Unfortunately, it is impossible to guarantee whether the algorithm will always complete its intended goal of navigating multiple fish towards a lead fish and hold station once a distance threshold has been reached.

This is, in part, attributed to the small scale of the domain under consideration. Numerous test runs have shown that the system crashes once a low pressure region begins to build up near the boundaries of the domain. This is an unavoidable occurrence, due to the nature of the boundary conditions imposed on the domain. To guarantee that the fish correctly determines the presence of this boundary, the fish would need to navigate itself along the boundary, periodically checking for equal low pressure conditions over a small time frame. We believe that this extra distance to be checked would be on the order of the control algorithm's positional uncertainty threshold. In effect, the fish would need to be certain of the 'overlap' between different boundary regions to be sure that it should definitely navigate away from this region. Currently, this is conflicting with the simple pressure input required for the control algorithm to function. A similar problem occurs when the following fish positions itself near the lead fish, as the boundary conditions on the domain are also imposed on the fish themselves. In this case, the fish would need to traverse a small distance around the lead fish to make sure that the goal has been reached. This would be mitigated in a larger environment as the fish would no longer need to distinguish between other fish and the boundary.

Considering the nature of the KD-Tree algorithm, we note the limitations of the computational time required to search for the nearest neighbour points. On average the computational costs are $O(\log n)$ [24]. If our simulation model were to be extended to higher dimensions, the curse of dimensionality requires the KD-Tree algorithm to search for many more branches for points as compared to a lower dimensional space, reducing the efficiency comparable to an exhaustive search [68]. For real-time applications in hardware, a widely accepted modification is an approximate nearest neighbour search. This modification sets an upper bound on the number of points to examine in a tree and as the best point is not searched for exhaustively, it offers significant speed increases. A well-known modification is the point-to-point Iterative Closest Point algorithm with computational costs of $O(1)$ [6].

In the results, we observe the emergence of a linear formation of the three fish. The term 'emergent behaviour' refers to the collective behaviour of a multi-fish system that is not attributed to any single fish, but is a result of coordination between many fish [38]. If we accept this definition of emergent behaviour, we highlight that the behaviour of the three fish in the domain fulfils the above stated criteria. We note that the individual fish are not told about the location of the lead fish in advance, nor are they specifically programmed to move towards the coordinates of the lead fish. The only rule is to find the minimum pressure position in the vicinity of a single fish and allow that fish to navigate towards the point. As the vortices travel rightwards in the domain, the fish are exposed to more localised low pressure systems and, gradually, a specific arrangement of fish is formed.

5.3 Summary

This chapter has focused on discussing the limitations of the fluid environment and the proposed control algorithm for navigation. We highlight the effects of the modeller's chosen parameter values (including effects from external forces and the value of Re) on the simulation properties. We note the lack of obstacles in the fluid potentially generating vortices and creating noise in the input to the current fish. We mention the computational costs associated with the current implementation of the KD-Tree and propose a modification in the form of set upper bounds on the number of points searched - a well known example being the Iterative Closest Point algorithm with costs of the order $O(1)$ compared to $O(\log n)$.

Chapter 6

Conclusion and Future Work

We have succeeded in creating an in-house fluid environment to test decentralised control algorithms in ‘real time’. We have adequately succeeded in our secondary aim of presenting a novel algorithm for ’real time’ navigation through vortices in this fluid environment. This chapter concludes the work conducted throughout the thesis and comments on possible directions to further this work.

6.1 Summary of Contributions

First, we draw attention to the fluid environment presented in Chapter 3. We have succeeded in introducing the Navier-Stokes equations - a widely accepted model of incompressible fluid flow dynamics. We introduce the idea of discretising the domain under consideration using finite element methods to create a mesh on the domain. We explain how the Navier-Stokes equations may be discretised using the weak formulation of the system. We present several solver-preconditioner configurations to solve this linearised system on the vertices of the above constructed mesh. We provide derivations for the weak formulation of a numerical scheme for the discretised Navier-Stokes system. This is then implemented in Python using the FEniCS platform for PDEs. To be able to move objects in the mesh without solver convergence failure, we require the ALE method to move objects and then smooth the deformed mesh multiple times. We introduce the method and its implementation in our simulation framework.

This is a crucial step in developing navigational algorithms for use in robotics, as the fluid in which the robot operates is not static. It requires the control algorithm to take input in ‘real time’ with changing fluid features and act accordingly. The ability to control the motion of agents in a fluid, along with simulating the environment simultaneously, is not currently supported in commercial computational fluid dynamics software. This thesis creates such an integrated environment.

Chapter 4 details the proposed control algorithm for this integrated environment, along with the supporting algorithms required for minimalistic collision detection and boundary avoidance. We take inspiration from the KD-Tree algorithm to detect the nearest neighbours, implemented using the SciPy library in Python. It creates a query point tree, searching for points within a given radius. We set the radius to 0.06 m, which is slightly more than the radius of the fish agent, so as to only include local pressure data as inputs to the control algorithm. We evaluate this proposed algorithm by testing whether a swarm is created starting off with different initial positions. A swarm is defined as a collection of agents within 0.2 m of each other. We show the gradual decrease of the Euclidean distance between the lead fish and the following fish eventually reaching the assigned 0.2 m threshold, at which point, the simulation terminates. This is a notable step in furthering the current body of knowledge in the field of swarm emergence through simple, local interactions. We have managed to demonstrate this behaviour in a dynamic environment, using only local pressure data for each fish. This work has significant implications for navigational control strategies concerning swarms of autonomous undersea vehicles.

We discuss our findings in Chapter 5 and briefly comment on the limitations of the environment modelling process, as well as the limitations of the proposed control algorithm. We highlight the Iterative Closest Point algorithm as a potential modification for our implementation, due to the computational costs being several orders of magnitude lower than the KD-Tree approach.

6.2 Future Work

We propose that future studies could vary the Re and the external forces applied on the fluid elements, and analyse the change in fluid dynamics. To demonstrate the scalability of the shoal, we require studies with many more agents. Other comparative studies could be conducted on the differences between the simulations and real world robotic applications. We have already developed the mathematics to allow the simulation to be implemented in 3D. Future studies could focus on this transition from 2D to 3D and conduct analyses on the robustness on the simulation framework and control algorithm.

Studies should also be focused on furthering the fluid-structure interaction capabilities of the current environment. We chose a simplistic model of a circle to generate the vortices in the domain due to time and computational constraints. A more realistic fish model could take the form of an oscillating aerofoil, which would produce the reverse KVS observed in the wake of swimming fish. Recent work on fluid-structure interaction is done in the turtleFSI [5] library, where an ALE frame of reference is created. Integrating these capabilities in our fluid environment would significantly inform the development of new control algorithms. These new improvements would contribute to the existing body of knowledge on swarm decision-making processes in fish schools.

Bibliography

[1] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. Rognes, and G. Wells, *Archive of numerical software: The fenics project version 1.5*, University Library Heidelberg, (2015).

[2] C. Baker and J. Montgomery, *The sensory basis of rheotaxis in the blind mexican cave fish, astyanax fasciatus*, Journal of Comparative Physiology A, 184 (1999), pp. 519–527.

[3] C. Bennett and R. C. Sharpley, *Interpolation of operators*, Academic press, 1988.

[4] M. Bercovier, O. Pironneau, and V. Sastri, *Finite elements and characteristics for some parabolic-hyperbolic problems*, Applied Mathematical Modelling, 7 (1983), pp. 89–96.

[5] A. Bergersen, A. Slyngstad, S. Gjertsen, A. Souche, and K. Valen-Sendstad, *turtleFSI: A robust and monolithic FEniCS-based fluid-structure interaction solver*, Journal of Open Source Software, 5 (2020), p. 2089.

[6] P. J. Besl and N. D. McKay, *Method for registration of 3-d shapes*, in Sensor fusion IV: control paradigms and data structures, vol. 1611, Spie, 1992, pp. 586–606.

[7] K. H. Brox, *Comparison of some preconditioners for the coupled navier-stokes equations*, Master's thesis, 2015.

[8] J. M. Butler and K. P. Maruska, *The mechanosensory lateral line is used to assess opponents and mediate aggressive behaviors during territorial interactions in an african cichlid fish*, Journal of Experimental Biology, 218 (2015), pp. 3284–3294.

[9] B. P. Chagnaud, H. Bleckmann, and M. H. Hofmann, *Kármán vortex street detection by the lateral line*, Journal of Comparative Physiology A, 193 (2007), pp. 753–763.

[10] S. Chakraverty and K. K. Pradhan, *Vibration of functionally graded beams and plates*, Academic Press, 2016.

[11] J. X. Chen, N. Lobo, C. E. Hughes, and J. M. Moshell, *Real-time fluid simulation in a dynamic virtual environment*, IEEE Computer Graphics and Applications, 17 (1997), pp. 52–61.

[12] A. J. Chorin, *Numerical solution of the navier-stokes equations*, Mathematics of computation, 22 (1968), pp. 745–762.

[13] S. Coombs, J. Janssen, and J. F. Webb, *Diversity of lateral line systems: evolutionary and functional considerations*, in Sensory biology of aquatic animals, Springer, 1988, pp. 553–593.

[14] S. Coombs and P. Patton, *Lateral line stimulation patterns and prey orienting behavior in the lake michigan mottled sculpin (cottus bairdi)*, Journal of Comparative Physiology A, 195 (2009), pp. 279–297.

[15] G. Denil, *La mecanique du poisson de riviere: les capacites mecaniques de la truite et du saumon*, in Annales des travaux publics de Belgique, vol. 38, 1937, pp. 412–423.

[16] S. C. R. Dennis and G.-Z. Chang, *Numerical solutions for steady flow past a circular cylinder at reynolds numbers up to 100*, Journal of Fluid Mechanics, 42 (1970), pp. 471–489.

[17] L. DeVries, F. D. Lagor, H. Lei, X. Tan, and D. A. Paley, *Distributed flow estimation and closed-loop control of an underwater vehicle with a multi-modal artificial lateral line*, Bioinspiration & biomimetics, 10 (2015), p. 025002.

[18] J. Donea, A. Huerta, J.-P. Ponthot, and A. Rodríguez-Ferran, *Arbitrary l agrangian–e ulerian methods*, Encyclopedia of computational mechanics, (2004).

[19] I. Erhunmwun and U. Ikponmwosa, *Review on finite element method*, Journal of Applied Sciences and Environmental Management, 21 (2017), pp. 999–1002.

[20] Z. Fan, J. Chen, J. Zou, D. Bullen, C. Liu, and F. Delcomyn, *Design and fabrication of artificial lateral line flow sensors*, Journal of micromechanics and microengineering, 12 (2002), p. 655.

[21] K. Faucher, E. Parmentier, C. Becco, N. Vandewalle, and P. Vandewalle, *Fish lateral system is required for accurate control of shoaling behaviour*, Animal Behaviour, 79 (2010), pp. 679–687.

[22] C. L. Fefferman, *Existence and smoothness of the navier-stokes equation*, The millennium prize problems, 57 (2000), p. 67.

[23] F. E. Fish, *Swimming strategies for energy economy*, Fish swimming: an etho-ecological perspective, 90 (2010).

[24] J. H. Friedman, J. L. Bentley, and R. A. Finkel, *An algorithm for finding best matches in logarithmic expected time*, ACM Transactions on Mathematical Software (TOMS), 3 (1977), pp. 209–226.

[25] V. Girault and P.-A. Raviart, *Finite element methods for Navier-Stokes equations: theory and algorithms*, vol. 5, Springer Science & Business Media, 2012.

[26] K. Goda, *A multistep technique with implicit difference schemes for calculating two-or three-dimensional cavity flows*, Journal of computational physics, 30 (1979), pp. 76–95.

[27] J.-G. J. Godin, *Antipredator function of shoaling in teleost fishes: a selective review.*, Naturaliste Canadien, 113 (1986), pp. 241–250.

[28] J.-L. Guermond, P. Minev, and J. Shen, *Error analysis of pressure-correction schemes for the time-dependent stokes equations with open boundary conditions*, SIAM Journal on Numerical Analysis, 43 (2005), pp. 239–258.

[29] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.

[30] D. Hoff, *Discontinuous solutions of the navier-stokes equations for compressible flow*, Archive for rational mechanics and analysis, 114 (1991), pp. 15–46.

[31] Y. Jiang, J. Fu, D. Zhang, and Y. Zhao, *Investigation on the lateral line systems of two cavefish: Sinocyclocheilus macrophthalmus and s. microphthalmus (cypriniformes: Cyprinidae)*, Journal of Bionic Engineering, 13 (2016), pp. 108–114.

[32] Y. Jiang, Z. Ma, and D. Zhang, *Flow field perception based on the fish lateral line system*, Bioinspiration & biomimetics, 14 (2019), p. 041001.

[33] A. Kroese and N. Schellart, *Velocity-and acceleration-sensitive units in the trunk lateral line of the trout*, Journal of Neurophysiology, 68 (1992), pp. 2212–2221.

[34] F. D. Lagor, L. D. DeVries, K. Waychoff, and D. A. Paley, *Bio-inspired flow sensing and control: Autonomous rheotaxis using distributed pressure measurements*, Journal of Unmanned System Technology, 1 (2013), pp. 78–88.

[35] H. P. Langtangen and A. Logg, *Solving pdes in minutes–the fenics tutorial volume i*, (2016).

[36] M. G. Larson and F. Bengzon, *The finite element method: theory, implementation, and applications*, vol. 10, Springer Science & Business Media, 2013.

[37] Y. Li, J.-I. Choi, Y. Choic, and J. Kim, *A simple and efficient outflow boundary condition for the incompressible navier–stokes equations*, Engineering Applications of Computational Fluid Mechanics, 11 (2017), pp. 69–85.

[38] Z. Li, C. H. Sim, and M. Y. H. Low, *A survey of emergent behavior and its impacts in agent-based systems*, in 2006 4th IEEE international conference on industrial informatics, IEEE, 2006, pp. 1295–1300.

[39] J. C. Liao and O. Akanyeti, *Fish swimming in a kármán vortex street: kinematics, sensory biology and energetics*, Marine technology society journal, 51 (2017), p. 48.

[40] J. C. Liao, D. N. Beal, G. V. Lauder, and M. S. Triantafyllou, *Fish exploiting vortices decrease muscle activity*, Science, 302 (2003), pp. 1566–1569.

[41] A. Magurran, W. Oulton, and T. Pitcher, *Vigilant behaviour and shoal size in minnows*, Zeitschrift für Tierpsychologie, 67 (1985), pp. 167–178.

[42] A. Magurran and T. Pitcher, *Foraging, timidity and shoal size in minnows and goldfish*, Behavioral Ecology and Sociobiology, 12 (1983), pp. 147–152.

[43] S. Maneewongvatana and D. M. Mount, *Analysis of approximate nearest neighbor searching with clustered point sets*, arXiv preprint cs/9901013, (1999).

[44] W. McKinney et al., *Data structures for statistical computing in python*, in Proceedings of the 9th Python in Science Conference, vol. 445, Austin, TX, 2010, pp. 51–56.

[45] P. J. Mekdara, M. A. Schwalbe, L. L. Coughlin, and E. D. Tytell, *The effects of lateral line ablation and regeneration in schooling giant danios*, Journal of Experimental Biology, 221 (2018), p. jeb175166.

[46] M. Mortensen and M. Kuchta, *fenicstools*. `https://github.com/mikaem/fenicstools`, (2019).

[47] Y. Narasimhulu, A. Suthar, R. Pasunuri, and V. Venkaiah, *Ckd-tree: An improved kd-tree construction algorithm*, in International Semantic Intelligence Conference, (2021), pp. 211–218.

[48] I. Navarro and F. Matía, *An introduction to swarm robotics*, International Scholarly Research Notices, 2013 (2013).

[49] S. Neill and J. M. Cullen, *Experiments on whether schooling by their prey affects the hunting behaviour of cephalopods and fish predators*, Journal of Zoology, 172 (1974), pp. 549–569.

[50] L. Olander, L. Conroy, I. Kulmala, R. P. Garrison, M. Ellenbecker, B. Biegert, B. Fletcher, H. D. Goodfellow, G. Rosén, B. Ljungqvist, et al., *Local ventilation*, Industrial Ventilation Design Guidebook, (2001), pp. 807–1023.

[51] F. Oleari, D. L. Rizzini, F. Kallasi, J. Aleotti, and S. Caselli, *Issues in high performance vision systems design for underwater interventions*, in IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2016, pp. 945–950.

[52] H. Persillon and M. Braza, *Physical analysis of the transition to turbulence in the wake of a circular cylinder by three-dimensional navier–stokes simulation*, Journal of Fluid Mechanics, 365 (1998), pp. 23–88.

[53] P.-O. Persson and J. Peraire, *Newton-gmres preconditioning for discontinuous galerkin discretizations of the navier–stokes equations*, SIAM Journal on Scientific Computing, 30 (2008), pp. 2709–2733.

[54] T. J. Pitcher, *Functions of shoaling behaviour in teleosts*, in The behaviour of teleost fishes, Springer, 1986, pp. 294–337.

[55] T. J. Pitcher, B. L. Partridge, and C. Wardle, *A blind fish can school*, Science, 194 (1976), pp. 963–965.

[56] V. Przulj, *Computational modelling of vortex shedding flows*, PhD thesis, City University London, 1998.

[57] A. Quarteroni and S. Quarteroni, *Numerical models for differential problems*, vol. 2, Springer, 2009.

[58] M. M. Rahman, T. T. Nahar, and D. Kim, *Feview: Finite element model (fem) visualization and post-processing tool for opensees*, SoftwareX, 15 (2021), p. 100751.

[59] D. REMPFER, *On boundary conditions for incompressible navier-stokes problems*, (2006).

[60] Z. REN AND K. MOHSENI, *A model of the lateral line of fish for vortex sensing*, Bioinspiration & biomimetics, 7 (2012), p. 036016.

[61] T. SALUMÄE AND M. KRUUSMAA, *Flow-relative control of an underwater robot*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 469 (2013), p. 20120671.

[62] M. SCHÄFER, S. TUREK, F. DURST, E. KRAUSE, AND R. RANNACHER, *Benchmark computations of laminar flow around a cylinder*, in Flow simulation with high-performance computers II, Springer, 1996, pp. 547–566.

[63] E. SCOTT, *A Biologically-inspired Artificial Lateral line*, PhD thesis, University of Bristol, 2021.

[64] R. SEVILLA, O. HASSAN, AND K. MORGAN, *An analysis of the performance of a high-order stabilised finite element method for simulating compressible flows*, Computer Methods in Applied Mechanics and Engineering, 253 (2013), pp. 15–27.

[65] G. L. SLEIJPEN, H. A. VAN DER VORST, AND D. R. FOKKEMA, *Bicgstab (l) and other hybrid bi-cg methods*, Numerical Algorithms, 7 (1994), pp. 75–109.

[66] D. SUTANTYO AND P. LEVI, *A bio-inspired tdma scheduling algorithm for underwater robotic swarms*, in 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2013, pp. 1107–1112.

[67] R. TEMAM, *Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (i)*, Archive for Rational Mechanics and Analysis, 32 (1969), pp. 135–153.

[68] C. D. TOTH, J. O'ROURKE, AND J. E. GOODMAN, *Handbook of discrete and computational geometry*, CRC press, 2017.

[69] J. C. VARUGHESE, R. THENIUS, P. LEITGEB, F. WOTAWA, AND T. SCHMICKL, *A model for bio-inspired underwater swarm robotic exploration*, IFAC-PapersOnLine, 51 (2018), pp. 385–390.

[70] R. VENTURELLI, O. AKANYETI, F. VISENTIN, J. JEŽOV, L. D. CHAMBERS, G. TOMING, J. BROWN, M. KRUUSMAA, W. M. MEGILL, AND P. FIORINI, *Hydrodynamic pressure sensing with an artificial lateral line in steady and unsteady flows*, Bioinspiration & biomimetics, 7 (2012), p. 036004.

[71] T. VICSEK, A. CZIRÓK, E. BEN-JACOB, I. COHEN, AND O. SHOCHET, *Novel type of phase transition in a system of self-driven particles*, Physical review letters, 75 (1995), p. 1226.

[72] P. VIRTANEN, R. GOMMERS, T. E. OLIPHANT, M. HABERLAND, T. REDDY, D. COURNAPEAU, E. BUROVSKI, P. PETERSON, W. WECKESSER, J. BRIGHT, S. J. VAN DER WALT, M. BRETT, J. WILSON, K. J. MILLMAN, N. MAYOROV, A. R. J. NELSON, E. JONES, R. KERN, E. LARSON, C. J. CAREY, İ. POLAT, Y. FENG, E. W. MOORE, J. VANDERPLAS, D. LAXALDE, J. PERKTOLD, R. CIMRMAN, I. HENRIKSEN, E. A. QUINTERO, C. R. HARRIS, A. M. ARCHIBALD, A. H. RIBEIRO, F. PEDREGOSA, P. VAN MULBREGT, AND SCIPY 1.0 CONTRIBUTORS, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.

[73] S. VOGEL AND J. GOLLUB, *Life in moving fluids*, Physics Today, 48 (1995), p. 84.

[74] W. WANG, X. ZHANG, J. ZHAO, AND G. XIE, *Sensing the neighboring robot by the artificial lateral line of a bio-inspired robotic fish*, in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2015, pp. 1565–1570.

[75] D. WEIHS, *Semi-infinite vortex trails, and their relation to oscillating airfoils*, Journal of Fluid Mechanics, 54 (1972), pp. 679–690.

[76] Y. XU AND K. MOHSENI, *A pressure sensory system inspired by the fish lateral line: Hydrodynamic force estimation and wall detection*, IEEE Journal of Oceanic Engineering, 42 (2016), pp. 532–543.

[77] U. M. YANG ET AL., *Boomeramg: A parallel algebraic multigrid solver and preconditioner*, Applied Numerical Mathematics, 41 (2002), pp. 155–177.

[78] Y. YANG, N. NGUYEN, N. CHEN, M. LOCKWOOD, C. TUCKER, H. HU, H. BLECKMANN, C. LIU, AND D. L. JONES, *Artificial lateral line with biomimetic neuromasts to emulate fish sensing*, Bioinspiration & biomimetics, 5 (2010), p. 016001.

[79] X. ZHENG, C. WANG, R. FAN, AND G. XIE, *Artificial lateral line based local sensing between two adjacent robotic fish*, Bioinspiration & biomimetics, 13 (2017), p. 016002.

[80] F. ZULKARNAEN, *Fish-inspired sensing and control: Localisation for swarms of autonomous underwater robots*, Master's thesis, University of Bristol, 2021.