Design of an effective swarming system for the pervasive monitoring of aquatic environments

Submitted by

KUAN Yoke Kong

Thesis Advisor(s)

Associate Professor Roland Bouffanais
Doctor David Mateo

Engineering Product Development

A thesis submitted to the Singapore University of Technology and Design in fulfilment of the requirement for the degree of Doctor of Philosophy (2018)
PhD Thesis Examination Committee

TEC Chair : Prof. David Rosen
Main Advisor : Assoc. Prof. Roland Bouffanais
Co-advisor : Dr. David Mateo
Internal TEC member 1 : Asst. Prof. Foong Shaohui
Internal TEC member 2 : Asst. Prof. Soh Gim Song

DECLARATION

I hereby confirm the following:

- I hereby confirm that the thesis work is original and has not been submitted to any other University or Institution for higher degree purposes.

- I hereby grant SUTD the permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created in accordance with Policy on Intellectual Property, clause 4.2.2.

- I have fulfilled all requirements as prescribed by the University and provided ONE copy of my thesis in PDF.

- I have attached all publications and award list related to the thesis (e.g. journal, conference report and patent).

- The thesis does not contain patentable or confidential information.

- I certify that the thesis has been checked for plagiarism via turnitin/ithenticate. The score is 14%

<table>
<thead>
<tr>
<th>Name &amp; Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thesis Candidate</td>
<td>Kuan Yoke Kong</td>
</tr>
</tbody>
</table>
Abstract

Human intervention and activities have affected the marine and inland water-bodies ecosystems thereby leading to an increase in the occurrence of environmental issues such as algal blooms, oil spills, and acidification. In order to study the human impact and understand the complex interplay between dynamic, physical, and biogeochemical processes in the aquatic environments, environmental scientists and oceanographers seek to access a wide array of measurements across a range of spatial and temporal scales: e.g. temperature, dissolved oxygen, pH, and biological data. However, there is a lack of efficacious technology for the robust monitoring and tracking of dynamic environmental features over the spatial scales of the order of few square kilometres, and with the necessary temporal resolution. Present monitoring techniques rely on either a single autonomous surface vehicle (ASV) or a fixed network of sensors that are unable to provide high temporal resolution over small spatial scales (0.1 km$^2$ – 1 km$^2$). This dissertation addresses the gap in terms of both technology and science related to a solution for the pervasive and permanent monitoring of water quality with adaptive resolution in both space and time. We study and discuss an innovative solution for the dynamic monitoring of aquatic environments through the design of a fleet of decentralised cooperative control ASVs that follow the swarm robotics paradigm.

First, we study the system-level design principles and develop a range of cooperative control strategies. Second, a robust fleet of ASVs is assembled based on the uncovered principles of dynamic collective behaviours. Finally, these cooperative control strategies are tested using the fleet of ASVs by performing collective behaviours such as flocking, navigation and dynamic area coverage. The topology and dynamics of the network of interaction play a key role in the effectiveness and responsiveness of swarming behaviours. This research also studies the effect of the number of interacting agents on the responsiveness of the system by using two distinct modelling approaches: a distributed linear leader-follower consensus protocol and an agent-based self-propelled particles model.

By studying the distinct distributed linear leader-follower consensus protocol and agent-based model, we uncover the important effect that the number of interacting agents has on the responsiveness of the collective. While increasing the number of interacting agents improves the system’s capacity to respond to slow changes, an excess of interactions can hinder the swiftness of its response to fast changes. Next, we investigate forced switching—which is the rewiring of the interaction network—with the same two models. For global forced switching, we uncover the existence of a trade-off between speed to consensus and the collective responsive of the system. In the case of local forced switching, we reveal that introducing a certain amount of forced switching improves the responsiveness of the system while keeping the number of interacting agents low. Finally, we have successfully developed and tested a fleet of 48 ASVs to perform...
a vast range of collective behaviours. This fleet of ASVs is the largest distributed multi-robot system of its kind reported to date. This research demonstrates that this swarm robotics system is able to achieve scalable deployment and dynamic monitoring in fully unstructured environments and in the absence of any supporting infrastructure. The robustness of the system is validated through the loss of multiple units while successfully performing different collective behaviours during the field experiments. A new metric is introduced to quantify coverage effectiveness for dynamic area monitoring. In addition, this research demonstrates the possibility of water quality monitoring of the system by measuring the water surface temperature and performing real-time temperature field reconstruction.

In conclusion, this research demonstrates that our system is capable of performing adaptive deployment, *in situ* measurement of water quality in the spatial scale of 1 km$^2$, and dynamic area coverage where the changes in area are in the order of one minute. Thus, this system provides an attractive and affordable solution to the complex and daunting problem of pervasive monitoring of aquatic environments. The study of interagent interaction shows that it has far-reaching implications for the design of the interaction network and provides insights for the design of artificial swarming systems.
List of Publications


Acknowledgements

Glory to God in the highest.

First and foremost, I would like to thank my advisor Associate Professor Roland Bouffanais whose patience, help, and guidance have made the completion of this thesis possible. Knowing that he has my best interests at heart has made the task much easier and enjoyable. I am very grateful for his support and motivation throughout the course of this PhD journey. I am also honoured to be his very first PhD student.

Special thanks to my co-advisor Dr. David Mateo whom I always go to for suggestions, simulation issues, and mathematical problems. He very patiently guided me through the coding process and explained the mathematics to me on many occasions.

I am grateful for the opportunity to collaborate with Professor Dick K. P. Yue’s group at MIT. They have provided me with a lot of guidance and technology know-hows in designing and building a seakeeping vessel. I thank Brandon Zoss and Grgur Tokić for countless insightful discussions on this field of research.

I would also like to extend my gratitude to my thesis examination committee members: Professor David William Rosen, Assistant Professor Foong Shaohui, and Assistant Professor Soh Gim Song. Their support and critique have played a valuable role in guiding and improving this dissertation.

I thank my fellow lab-mates: Wen Xin, Sreetej Lakkam, Nikolaj Horsevad Sørensen, Jabez Leong Kit, Francesco Vallegra, and Louis Goh. Without their help all the field experiments will be strenuous and might not be possible; my research and thesis would not have been completed. Without them, working in the lab would not have been as fun as it was.

This PhD journey would have been a lot tougher without the support and encouragement from new-found friends at SUTD. I would like to especially thank James Tan, Huang Zhixiang, George Chen, Hashina Parveen, Li Xiaqian, and Jessica Tan for that.

Most importantly, I would like to thank my parents and sisters for their unfailing love and belief in me. Their unwavering support and encouragement are one of the greatest gifts in life.

Last but not least, I would like to acknowledge National Supercomputing Centre for Aspire and SUTD-MIT IDC for Titan. Without their support, the hundreds of thousands of hours spent on simulations would not have been possible.
# Contents

Abstract vii

List of Publications viii

Acknowledgements ix

Contents xiii

List of Abbreviations xv

List of Figures xxvi

List of Tables xxvii

1 Introduction 1

1.1 Motivations ...................................... 1

1.1.1 Aquatic Environment Monitoring ...................... 2

1.2 Swarms ........................................ 5

1.2.1 Collective Behaviours ............................. 5

1.2.2 Swarm Intelligence .............................. 7

1.2.3 Biologically-inspired Algorithms ....................... 8

1.2.4 Swarm Robotics ................................ 10

1.3 Research Objectives .................................. 12

1.4 Thesis Outline ..................................... 14

2 Nuts and Bolts of Buoys 15

2.1 Buoy Design ...................................... 15

2.1.1 Conceptual Design ................................ 15

2.1.2 Hydrostatic Properties ............................ 18

2.1.3 Hydrodynamic Simulation .......................... 20

2.1.4 Propulsion System .............................. 21

2.1.5 Electronic Suite ................................ 23

2.1.6 Global Positioning System .......................... 25

2.1.7 Sensory Peripherals .............................. 27

2.2 Swarm-enabling Technology ............................. 30

2.2.1 XBee-Pro® Module ................................ 31

2.2.2 Distributed Communication .......................... 33

2.2.3 Buoy In-house Software .......................... 34
### Contents

2.2.4 Robot Operating System ................................................................. 36

2.3 Buoy Characterisations ................................................................. 38
  2.3.1 Speed Analysis ............................................................................. 38
  2.3.2 Battery Life Analysis ................................................................. 38
  2.3.3 Budgetary Consideration ............................................................ 40

2.4 Summary ......................................................................................... 40

3 Too Many Cooks Spoil the Broth .......................................................... 43
  3.1 Physical Approach to Swarming Systems ........................................ 43
  3.2 Network-Theoretic Approach ........................................................ 45
    3.2.1 Average Path Length ............................................................... 47
    3.2.2 Connectedness ......................................................................... 47
    3.2.3 Clustering Coefficient ............................................................. 48
    3.2.4 Laplacian Matrix ....................................................................... 49
    3.2.5 Small-World Network ............................................................... 50
  3.3 Static Network – Linear Consensus Protocol ..................................... 53
    3.3.1 State-space Analysis ................................................................. 55
  3.4 Dynamic and Switching Connectivity – Vicsek Model ....................... 56
    3.4.1 Neighbourhood of Interaction .................................................. 56
    3.4.2 Phase Transition ....................................................................... 58
    3.4.3 Correlations and Fluctuations ................................................... 59

3.5 Results and Discussion - Physics Approach ....................................... 60
  3.5.1 Distributed Linear Consensus with Varying Levels of Connectivity ... 61
  3.5.2 Vicsek Model with Varying Levels of Connectivity ....................... 63

3.6 Consensus with Forced Switching ...................................................... 65
  3.6.1 Distributed Linear Consensus with Watts and Strogatz Network .... 66
  3.6.2 Vicsek Model with Local Extended and Global Interaction ............ 66

3.7 Results and Discussion - Forced Switching ....................................... 68
  3.7.1 Distributed Linear Consensus with Forced Switching .................... 68
  3.7.2 Vicsek Model with Forced Switching Network .............................. 72

3.8 Summary ......................................................................................... 74

4 Where The Rubber Meets The Road ...................................................... 77
  4.1 Buoys’ Data-Flow ........................................................................... 77
  4.2 Cooperative Control Algorithms ..................................................... 78
    4.2.1 Basic Behavioural Rules ............................................................ 80
    4.2.2 Flocking Behaviour ................................................................. 83
    4.2.3 Collective Navigation Behaviour .............................................. 84
    4.2.4 Area Coverage Behaviour ....................................................... 84

  4.3 Field Experiments ......................................................................... 86
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1</td>
<td>Station–Keeping</td>
<td>87</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Avoidance</td>
<td>88</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Aggregation</td>
<td>89</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Leader-follower</td>
<td>90</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Dynamic Monitoring</td>
<td>91</td>
</tr>
<tr>
<td>4.4</td>
<td>Collective Data Analysis</td>
<td>95</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Numerical Method for Scalar Field Reconstruction</td>
<td>95</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Field Experiments - Scalar Field Reconstruction</td>
<td>99</td>
</tr>
<tr>
<td>4.5</td>
<td>Experimental Hindsight</td>
<td>102</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions</td>
<td>105</td>
</tr>
<tr>
<td>5.1</td>
<td>Key Accomplishments</td>
<td>105</td>
</tr>
<tr>
<td>5.2</td>
<td>Outlook And Perspectives</td>
<td>108</td>
</tr>
<tr>
<td>A</td>
<td>Supplementary Notes</td>
<td>111</td>
</tr>
<tr>
<td>A.1</td>
<td>Stability Calculation</td>
<td>111</td>
</tr>
<tr>
<td>A.2</td>
<td>Theoretical Analysis for Linear Consensus Protocol</td>
<td>112</td>
</tr>
<tr>
<td>B</td>
<td>Buoy GUI and Buoy Operating System Configurations</td>
<td>115</td>
</tr>
<tr>
<td>B.1</td>
<td>Configuration of MacOS computer for Buoy GUI</td>
<td>115</td>
</tr>
<tr>
<td>B.2</td>
<td>Configuration of BeagleBone Black for Buoy Operating System</td>
<td>118</td>
</tr>
<tr>
<td>C</td>
<td>Matlab Codes For Effect of Connectivity</td>
<td>121</td>
</tr>
<tr>
<td>D</td>
<td>Matlab Codes For Forced Switching</td>
<td>127</td>
</tr>
<tr>
<td>E</td>
<td>Code for Postprocessing of Scalar Field Reconstruction</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>147</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>ACO</td>
<td>Ant Colony Optimisation</td>
<td></td>
</tr>
<tr>
<td>ASC</td>
<td>Autonomous Surface Craft</td>
<td></td>
</tr>
<tr>
<td>ASV</td>
<td>Autonomous Surface Vehicle</td>
<td></td>
</tr>
<tr>
<td>AVA</td>
<td>Agri-Food and Veterinary Authority</td>
<td></td>
</tr>
<tr>
<td>BBB</td>
<td>BeagleBone Black</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>Cellular Automata</td>
<td></td>
</tr>
<tr>
<td>CAD</td>
<td>Computer-Aided Design</td>
<td></td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Research Projects Agency</td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>Dissolved Oxygen</td>
<td></td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
<td></td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
<td></td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
<td></td>
</tr>
<tr>
<td>HAB</td>
<td>Harmful Algal Bloom</td>
<td></td>
</tr>
<tr>
<td>HDPE</td>
<td>High-Density Polyethylene</td>
<td></td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
<td></td>
</tr>
<tr>
<td>LVC</td>
<td>Largest Voronoi Cell</td>
<td></td>
</tr>
<tr>
<td>NEA</td>
<td>National Environment Agency</td>
<td></td>
</tr>
<tr>
<td>NOAA</td>
<td>National Oceanic and Atmospheric Administration</td>
<td></td>
</tr>
<tr>
<td>NRF</td>
<td>National Research Foundation</td>
<td></td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
<td></td>
</tr>
<tr>
<td>pH</td>
<td>Potential of Hydrogen</td>
<td></td>
</tr>
<tr>
<td>PUB</td>
<td>Public Utilities Board</td>
<td></td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse-width Modulation</td>
<td></td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operation System</td>
<td></td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indicator</td>
<td></td>
</tr>
<tr>
<td>RTK</td>
<td>Real-time Kinematics</td>
<td></td>
</tr>
<tr>
<td>SI</td>
<td>Swarm Intelligence</td>
<td></td>
</tr>
<tr>
<td>SMART</td>
<td>Singapore-MIT Alliance for Research and Technology</td>
<td></td>
</tr>
<tr>
<td>SMI</td>
<td>Sparse Matrix Interpolation</td>
<td></td>
</tr>
<tr>
<td>SSN</td>
<td>Swarm Signaling Network</td>
<td></td>
</tr>
<tr>
<td>SPP</td>
<td>Self-Propelled Particles</td>
<td></td>
</tr>
<tr>
<td>TMSI</td>
<td>Tropical Marine Science Institute</td>
<td></td>
</tr>
<tr>
<td>TSP</td>
<td>Travelling Salesman Problem</td>
<td></td>
</tr>
<tr>
<td>VM</td>
<td>Vicsek Model</td>
<td></td>
</tr>
<tr>
<td>WS</td>
<td>Watts and Strogatz</td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

1.1 News clipping from a daily newspaper, Today, on 12 February 2014, reported on the losses of fish farmers along the Johor Straits due to plankton bloom. ........ 1
1.2 News clipping from a daily newspaper, The Newspaper, on 8 April 2013, illustration of Neptune buoy system, deployment locations and measuring parameters. Source: NEA, public domain. ................................................................. 3
1.3 News clipping from a daily newspaper, Today, on 5 March 2015, reported on plankton bloom that caused fish farmers along the Johor Straits to lose tonnes of fish. ................................................................. 4
1.4 Swarms in nature. (a) A swarm of ants foraging for food. (Credit: Zainichi Gaikokujuin, CC BY 3.0) (b) A flock of auklets exhibit swarm behaviour (Credit: U.S. Fish and Wildlife Service, D. Dibenski, public domain). (c) A school of fish swimming along with a ray (S.E.A Aquarium™, Singapore). (d) Lane formation of human crowds getting into train station after New Year celebration (Raffles Place MRT station, Singapore). ................................................................. 6
2.1 Mobile marine flooding platform developed by Professor Dick. K.P. Yue’s group at MIT (a) Platform with energy-harvesting ability (windmill). (b) Third prototype of mobile platform. ................................................................. 16
2.2 A computer-aided design (CAD) view of buoy. (a) Side-view of the buoy hull with main dimensions. (b) Semi-exploded rendered view of 3D buoy model showing the central parts of the buoy. Not all elements are shown (i.e. battery, XBee, GPS, hardware connections, etc). ................................................................. 17
2.3 The prototypes of autonomous buoy (a) First version of buoy made of styrofoam. (c) Fully assembled buoy no. 50 in the field, ready to be deployed. ............... 17
2.4 Centre of mass of buoys component measured from $Y = 0$ (values obtained from SOLIDWORKS) for calculating the buoy centre of gravity (red dot, $G$). The blue dot represents the centre of buoyancy ($B$). All dimensions are in millimetres. 19
2.5 Calculation of buoy stability at different heel angles (a) Shows the locations of metacentre, $M$, (green square and dots), centre of buoyancy, $B$, (blue square and dots) that rotated at the centre of gravity, $G$, (red square). Squares represent the initial positions of $M$, $B$, and $G$ when heel angle is 0°, and dots represent the position of $M$ and $B$ at different angles. (b) The buoy is able to self-right as the righting arm remains positive for $\phi$ in the range 0° - 180°. ......................... 19
2.6 Drag coefficients, $C_d$, of different attack angles obtained from the SOLIDWORKS flow simulation at different Reynolds numbers, $Re$. Inset: 5 different angles of attack used for flow simulation. ................................................................. 20
List of Figures

2.7 Hydrodynamic computational fluid dynamic (CFD) simulation of a buoy using the SOLIDWORKS flow simulation package. Streamlines of water flowing from lower left to upper right at 1 m/s (Reynolds number, $Re \approx 3.6 \times 10^5$) from 60° angle of attack, which generates highest drag coefficients. 21

2.8 Motor pods orientation and their control circuit. (a) Thrust pod orientation on buoy, each pod is offset by 120° from pod 1 (reference pod). The thrust direction is indicated by the arrows; the green arrow represents forward thrust and the red arrow indicates reverse thrust. (b) H-bridge diode configuration for motors (only two pods are shown for simplicity), which prevents the operation of forward and reverse motors at the same time within a pod. 22

2.9 Layout of our custom PCB design. Top: Back of the PCB where analogue to digital converter, EZO™ circuit and BeagleBone Black are mounted. Bottom: Top of the PCB where motor controller, IMU, GPS, XBee® are located. 23

2.10 Modular lid where the electronic suite is mounted on for fast switching when there are electronic issues during field experiments. 23

2.11 Block diagram of the electric components. All the sensors, controllers, and communication modules are connected to a central single board computer that runs the cooperative control algorithms. 24

2.12 BeagleBone Black (BBB) pinout diagram. 25

2.13 Adafruit Ultimate GPS module. 26

2.14 A buoy inside the transportation vehicle tracking the journey from the test site back to SUTD. 26

2.15 Emild’s Reach RTK module. 27

2.16 Tracking of half of the soccer field with the centre circle using Reach real-time kinematic (RTK) module. (a) Two communication systems were used, XBee® (represented by red line and dots) and WiFi (represented by blue line and dots). The jagged lines towards the end of the test was caused by the lost of RTK communication between the base station and the buoy. (b) Shows a zoom-in at the corner where centre circle and halfway line intercept. 28

2.17 All Atlas Scientific™ EZO™ circuit have UART and I²C interfaces, and footprint of a US quarter dollar coin. 28

2.18 Atlas Scientific™ PT1000 Temperature kit (a) Probe with standard EZO™ RTD circuit set up by Atlas Scientific™ (b) Probe mounted on buoy with 30 mm thermowell that extend probe outreach and also provide watertight seal. 29

2.19 Atlas Scientific™ dissolved oxygen kit (a) Probe with standard EZO™ circuit set up by Atlas Scientific™ (b) Probe mounted on buoy with probe pipe fitting to provide watertight seal. 29

2.20 Atlas Scientific™ pH kit (a) Probe with standard EZO™ circuit set up by Atlas Scientific™ (b) Probe mounted on buoy with probe pipe fitting to provide watertight seal. 30
2.21 XBee-Pro® Series 1, based on the DigiMesh® 2.4 protocol. 31
2.22 Xbee-Pro® received signal strength indicator (RSSI) measurement tests done at SUTD, East Coast Park (ECP), and Bedok reservoir (BR). The fit for data (purple line) is given by Equation (2.10), the empirical law mentioned in Kumar et al. [90]. 32
2.23 Sample of DigiMesh® network established by 11 buoys communicated through Xbee-Pro® in the field. The configuration of the buoys are quantitatively represented by the tag modules shown. The links between the modules denote direct communication between buoys, where blue links indicate strong signal and orange links indicate weak signal. Numbers adjacent to the links represent the transmitted and received signal strength in dB. The green module acts as an articulation node between two clusters by establishing a bridge that allows communication between them. This screenshot was obtained using Digi’s XCTU free software. 33
2.24 Experimental data of relatively successful communications established between two buoys separated by a certain distance (m) inside the fleet buoys, spread with a mean nearest neighbour distance of \( r_{ij} \). Data was measured during the dynamic area monitoring experiment (see section 4.3.5). 34
2.25 Graphical user interface (GUI) for monitoring movements of buoys and sending behavioural commands to buoys to perform certain tasks during field experiments. 35
2.26 Buoy finite state machine (FSM) diagram (left). Right shows the Shared Object where the Xbee module, GPS and AHRS nodes are connected to it. All states have access to the Shared Object. 38
2.27 Speed analysis from field experiment data. (a) Average speed analyzed from two runaway buoys during one of the field experiments that travelled approximately 320 metres (almost straight path). (b) Histogram of speed analysis of experiment data for collecting, expanding and leader-follower collective behaviours. 39
2.28 Balance of battery power for each buoys after 2.5 hours of field experiment performing various tasks continuously (50 buoys). 39
2.29 A fleet of 48 buoys lined up for a final check before deployment for field experiment. 41
3.1 Average path length for this simple network is 1.6. \( (d) = (d_{1\to2} + d_{1\to3} + d_{1\to4} + d_{1\to2} + d_{1\to5} + d_{2\to3} + d_{2\to4} + d_{2\to5} + d_{3\to4} + d_{3\to5} + d_{4\to5})/10 = 1.6 \). 47
3.2 The connectedness of a small network. (a) A connected undirected network, where there is a path between every pair of nodes. The green link is called a bridge that links both the left and right cliques together. (b) The absence of a bridge causes the network to be disconnected, resulting in two components. The nodes in these two components are connected to each others, \((1,2,3,4)\) and \((5,6,7,8)\). 48
3.3 The Watts and Strogatz model starting with ring lattice ($p = 0$), which can be turned by probability, $p$. Randomness of the networks increases with $p$, by rewiring the link according to $p$. Each link is rewired for maximum $p = 1$, where the network is completely random [156]. ........................................ 51

3.4 Characteristic average path length (PL) and average clustering coefficient (CC) for the Watts and Strogatz small-world network. PL and CC are normalized with the values of ring lattice for $p = 0$, $PL(0)$ and $C(0)$ respectively. This system has $N = 2048$ and the average degree is $k = 10$. .................................................. 53

3.5 Lattice (ring) topological network, with a leader (in green) and followers (in red), with $k = 2$ (left) and $k = 4$ (right), where $k$ is the set of neighbours of an agent. 54

3.6 An illustration of metric and topological neighbourhoods of interactions with two different densities. $R$ represents the radius of the metric neighbourhood, whereas $r$ is the radius of the topological neighbourhood with $k = 7$ (nearest neighbours, green triangles). Left: Shows a densely populated swarm, where $r$ is smaller than $R$ (in this case the metric neighbourhood has 14 neighbours, blue and green). Right: A less dense swarm, in this case $r$ is greater than $R$ in order to capture $k = 7$ neighbours (shown in green) and 4 neighbours for the metric interaction. In both cases, $R$ remains constant as it defines a metric zone around the agent. ............................................... 57

3.7 The average alignment, $\langle \varphi \rangle$, of the SPP in the steady state with varying noise, $\eta$ with $v_0 = 0.03$. Where N40R1D4 means; $N = 40$, where $N$ is the number of agents; $R = 1$, where $R$ is the radius of metric interaction; and $D = 4$, where $D$ is the density of the system. There are three different numbers of agents, $N = 40,100,4000$, that used metric interaction and same parameters as Ref [153]. The average connectivity for these three Ns is $\langle k \rangle = 12$. The N100k10D1 ($N = 100, k = 10$ and $\rho = 1$) represented by the green circle uses the topological interaction with connectivity, $k$, equal to 10. .................................................. 59

3.8 Responsiveness of a distributed leader-follower consensus protocol with $N = 2048$ agents following a single leader. (a) Total amplitude gain (normalized by the number of agents) as a function of connectivity, $k$, at different oscillating frequencies, $\omega$, where the leader input into the system. The solid colour lines correspond to a frequency, $\omega$, equal to the value specified in the legend. The dotted lines correspond to the frequencies evenly distributed between the solid lines. (b) At frequency, $\omega = 0.01$, the gain (non-normalized) starts to suffer beyond a certain threshold of connectivity, approximately $k = 20$, showing a detrimental effect related to connectivity. ........................................ 61
3.9 Normalized gain as a function of frequency at different number of connection, k. The vertical dotted lines split the spectrum of frequencies into four regions where the first region (low to high frequency) shows that k = N is the most favourable choice for the system, follow by k = 20, k = 10, and k = 2 for the rest of the regions respectively. ........................................ 62

3.10 Correlation in velocity fluctuations for VM, N = 2048 with topological interactions and number of neighbours k = 4, 22 and 34. The distance is measured in units of the computation box size L. .......................... 64

3.11 (a) The susceptibility, normalized by respective N, as a function of the number of neighbours, k, at N = 100, 512, 2048. All three systems show that there is a peak at a finite number of connections. (b) Curve fitting for the peaks, k_{peak}, corresponding to connectivity at different number of agents, k_{peak} = 10.48 \times \log(N) - 14.56. ........................................ 65

3.12 Forced switching method for the VM model. (a) Local extended forced switching, where k = 10 (green \(\triangle\) inside grey dotted circle) at previous iteration (left). Present iteration (right) \(k_s = 5\), the next 5 nearest neighbours are chosen from inside the black dotted circle (which is the maximum distance, \(k + k_s\)), \(k = k_p + k_s = 10\) (green \(\triangle\)). (b) Global (random) forced switching, starts from \(k = 10\) (left, green \(\triangle\) inside grey dotted circle) and moves to the next iteration (right) when \(k_s = 3\). These 3 agents are randomly chosen from the domain except the agents from previous set of nearest neighbours, \(k = k_p + k_s = 10\) (green \(\triangle\)). ... 68

3.13 Normalized gain as a function of probability, p, using the Watts and Strogatz network model for leader-follower linear consensus protocol (N = 2048). Left: effect of rewiring (randomly) on the system at low frequency, \(\omega = 0.0001\). It has a positive effect on the system, gain increases as the probability increases for all connectivity. Middle: system at mid-range frequency, \(\omega = 0.001\), there is no distinct observation whether a rewiring benefits the system or not prior to \(p = 0.01\) (dotted line). However, it is clear that rewiring has a detrimental effect on the gain beyond \(p = 0.01\). Right: at the frequency, \(\omega = 0.1\), random switching does not have any benefit to the system. After \(p = 0.01\) (dotted line), random switching leads to a reduction of the responsiveness of the system, i.e. a detrimental effect. ........................................ 70
3.14 Responsiveness of leader-follower consensus protocol with respect to connectivity and frequency with forced switching. (a) The total amplitude gain (normalized) for a system of $N = 2048$ as a function of connectivity at different oscillating frequencies, $\omega$, with varying probability of rewiring, $p$. Each color represents one frequency with the value specified on top of the line. Square markers indicate $p = 1$, circle markers represent $p = 0$, and dotted lines correspond to the probability of rewiring that evenly spread out between $p = 0$ and $p = 1$. (b) Responsiveness as a function of frequency, at $p = 0.029$ where switching occurs at $\omega = 0.008$. At this condition, the system performs equally well regardless of connectivity, beyond that the system benefits from low connectivity. .......................... 70

3.15 The influence of rewiring in terms of the second smallest eigenvalue, $\lambda_2$, of the graph Laplacian of the network. It shows that more random rewiring increases the speed to consensus for all connectivity. ....................... 71

3.16 Characteristics of the network shown by its properties at different values of connectivity, $k = 4, 20, 40$: average clustering coefficient (red □), CC, average path length (blue ○), PL, and second eigenvalue of the Laplacian matrix of the network (green △), $\lambda_2$. PL and CC are normalized with the values of ring lattice where $p = 0$, PL(0) and C(0) respectively. $\lambda_2$ is normalized with the maximum $\lambda_2$ as it increases with the probability of rewiring, $p$. .......................... 72

3.17 Effect of global random forced switching on collective response. (a) Susceptibility, $\chi$ as a function of forced switch, $k_s$, normalized by number of connection, $k$. (b) The network property of the second smallest eigenvalue of the Laplacian matrix was used to determine the speed to consensus, higher value signifies higher speed to consensus. .......................... 73

3.18 (a) Responsiveness of the system with varying connectivity, $k$, form up by combining number of connection and number of forced switching, $k_s, k + k_s$. The trend is the same as Figure 3.11a for $N = 2048$. (b) Effects of the local extended forced switching (with $k + k_s$) on the speed to consensus of a collective. .......................... 74

4.1 A diagrammatic representation of the date-flow template for microscopic flocking motion model. It details the main aspects for the generation of flocking motions via the five stages (boxes). The connections between the stages encode the data that propagates between them. In particular, the connections between the sensing stage and the flock member detection stage represents the raw sensor information from each sensor (e.g. GPS, camera, laser range-finder). The connection between the flock member detection and neighbour selection stage is the set of detected flock member. The neighbour selection stage passes at least one set of selected flock members to the motion computation stage which passes the next computed motion to the physical motion stage. .......................... 78
4.2 Simulation of leader-follower scenario in an imperfect communication. One of the buoys will act as a leader, by moving in a straight line at constant velocity while a certain fraction of the buoys “actively” follow that behaviour according to $G = \{\bar{g}^T, \bar{g}^R\}$. The rest of the buoys behave according to the default $G = \{\bar{g}^{AG}, \bar{g}^R\}$. The total ratio of followers is defined as the ratio of buoys that have the same velocity as the leader in the long-time asymptotic state. There are three typical outcomes for using this algorithm, shown by the insets: 1. When only $\sim 20\%$ of buoys is actively following the leader, the collective will split in two groups with the majority of buoys aggregating together and remaining stationary. 2. When half of the buoys are actively following the leader, the rest of the buoys will follow along but lag behind. 3. When a large fraction of the buoys ($\sim 80\%$) are actively following the leader, the remaining buoys will “passively” follow the leader even without knowledge of the leader’s position.

4.3 Simulation of collective avoidance behaviours in group while performing station-keeping task. A group of $N = 49$ buoys station-keep at target location with $r_0 = 100$ m. An additional buoy $i$ (lower right corner) is introduced and directed with Equation (4.9) towards a given goal (red cross mark) placed opposite the collective. Depending on the parameters, one can observe (a) yielding behaviour (zoom in), (b) bypassing behaviour, or (c) sneaking behaviour.

4.4 Equilibrium configurations of number of buoys, $N$, for dynamic area coverage behaviour following Equation (4.11) for the area given by Equation (4.12). The area changes with $\alpha$, $\alpha = 0$ (top row), $\alpha = 1$ (middle row), and $\alpha = 2$ (bottom row).

4.5 Initial deployment of buoys, done in batches. Collective buoys performing station-keeping task while waiting for new members to join in the collective. All together there were three batches. Red dots indicate the station-keep buoys while the blue dots indicate the joining buoys. Grey lines trailing the dots show the travel paths. Left: First batch station-keep while waiting for other buoys to join. Middle: Second batch joining the collective. Right: Third batch joining the collective formed by first and second batches.

4.6 Analysis of average drift while the fleet of buoys perform station-keeping task. All drift distances are calculated with respect to the initial point of each buoy and average were taken over all the buoys. Lines indicate average drift distance and the error bar denotes the standard deviation.

4.7 Avoidance behaviours in group navigation with yielding behaviour. (a) $N = 6$ with $r_0 = 10$ m, with an additional buoy (red star) passing through the collective performing station-keep task. (b) $N = 27$, $r_0 = 20$ m, one of the buoys (red star) was commanded to leave its position during station-keeping.
4.8 Reconstruction of aggregation behaviour of 43 buoys. (a) The aggregation paths were created by a sudden decrease of repulsion strength from $r_0 = 50$ m to 5 m as represented by black lines. Blue dots indicate the initial position and red dots represent the terminal locations for all the buoys. (b) Total surface covered by buoys during aggregation event and buoy density per m$^2$ at the center of collective. The red line with filled dots represent the surface covered data and the blue line with filled squares indicate the buoy density data. 90

4.9 Delaunay triangulation of 43 buoys’ positions during the aggregation, from initial position at $t = 0$ min to final position at $t = 8$ min. The Green square (20 m x 20 m) represents the centre region where all the buoys were heading towards. 90

4.10 Post field experiment reconstruction of leader-follower behaviour of 43 buoys with $r_0 = 5$, travelled for $\sim 400$ m. The path of the leader buoy is marked with a blue trail while the followers are marked with grey trails. Grey dots indicate the initial positions and red dots represent the termination of leader-follower behaviour. 91

4.11 Dynamic area coverage performed by a group of 20 buoys exhibiting dynamic scalability capability of the collective. Left: Evolution of the largest Voronoi cell’s area with respect to the total exploration area, $A_{LVC}$ (red), compared with the “ideal” case of $1/n$ (Blue). Right: Number of buoys, $n$, inside the exploration area (in this case a circle) at a particular time. 92

4.12 Correlation of largest Voronoi cell’s area, $A_{LVC}$, and number of buoys, $n$, for the same data as Figure 4.11. The fit of the experimental data, $A_{LVC} \sim n^{-1.05}$, shows that the efficiency of the system grows roughly linear with the number of buoys. 92

4.13 Pictures of the field experiment with 20 buoys performing dynamic area coverage following Equation (4.11), oscillating between a circle and a two-lobbed “dumbbell”. Insets: Post-experiment reconstruction of GPS location data, black dots represent buoy positions. Red dots and lines indicate the path of the buoys. 93

4.14 Flexibility of the collective by tasking 20 buoys to perform dynamic area coverage following Equation (4.11) that oscillating between a circle and a two-lobbed “dumbbell”. The buoys were covering an area of approximately 200 m$^2$. Evolution of the largest Voronoi cell’s area with respect to the total exploration area scaled with the number of buoys, $N \times A_{LVC}$. The grey shapes represent the dynamic exploration area. 94

4.15 Field experimental data of the fleet of buoys performing dynamic area coverage with the area changing from circle to dumbbell, at different frequency $\omega$. 94
4.16 Real time reconstruction of the temperature field during a field experiment extracted from control station graphical user interface (GUI). Dotted red lines indicate the selected interest of temperature contour. The slightly bigger dots with different colours indicate the present location of each buoy when the screen was captured. The blue regions are the lower temperature areas, whereas the red regions are the higher temperature areas. 100

4.17 Post experiment reconstruction of the temperature contours (blue line) based on the stored GPS data of 23 buoys. Grey dots indicate the starting position when contour-follow command was activated. Grey lines are the paths taken by different buoys toward the contour. Red dots are the final locations of the buoys. 100

4.18 Post experiment temperature field reconstruction of 43 collective buoys data. (a) The path taken by collectively moving buoys while mapping the temperature profile of the water. Spreading and aggregation paths are created by varying the repulsion strength from $r_0 = 10$ m to $50$ m and to $5$ m represent by grey line. Green dots indicate the initial position, while blue dots represent the maximum spread of the experiment. Red dots are the terminal locations for all the buoys. Reconstruction of the temperature field of the water based on collective data at (b) initial state, (c) maximum spread, and (d) terminal positions. Note: (c) and (d) share the same colorbar. 101

4.19 Pathway of data measurement for DO (represented by red line) and pH (represented by blue line) during the field experiment, the green dot represents the starting location of the buoy and the orange dot represents the ending location of the buoy. 102

4.20 A fleet of buoys stacked up in a van, ready to be transported to field test site. 103

5.1 Responsiveness of a distributed leader-follower consensus protocol with $N = 2048$ agents following a single leader. Total amplitude gain (normalized by the number of agents) as a function of connectivity, $k$, at different oscillating frequencies, $\omega$, which the leader inputs into the system. The solid colour lines correspond to a frequency, $\omega$, equal to the value specified in the legend. The dotted lines correspond to the frequencies evenly distributed between the solid lines. 106

5.2 Responsiveness of the system with varying connectivity, $k$, formed by combining number of connections and the number of forced switching, $k_s$, $k + k_s$. The trend is the same as Figure 3.11a for $N = 2048$. 107

5.3 An autonomous buoy that is equipped with water quality monitoring sensors can function alone or in group. 108

5.4 Picture of a field experiment with a fleet of buoys performing dynamic area coverage. 109
A.1 Stability calculation of a two-dimensional buoy. Waterline at the centre of the circle, radius is 16 cm. $M$ is the metacentre (in green), $B$ is the centre of buoyancy (in blue) and $G$ is the centre of gravity (in red) of the circle. Not drawn to scale. 111

A.2 A simple ring lattice with four nodes and two connections for each node using leader-follower consensus protocol. $u$ is the input or leader (represented by green dot), with three follower (represented by red and blue dots) and output (blue dot). 112
## List of Tables

2.1 Estimated mass and centre of gravity ($G$) of buoy components, where $G$ is measured from $Y = 0$ in Figure 2.4. ........................................... 18
2.2 Summary of Atlas Scientific™ probes measurement characteristics. ............. 30
2.3 List of components and cost for a buoy. ........................................... 40
Chapter 1

Introduction

1.1 Motivations

It all happened overnight...

Figure 1.1: News clipping from a daily newspaper, Today, on 12 February 2014, reported on the losses of fish farmers along the Johor Straits due to plankton bloom.

Fish farmers woke up to the sight of their fish floating belly up and did not have the opportunity to save them (see Figure 1.1). The cause was later confirmed by the Agri-Food and Veterinary Authority (AVA) to be plankton bloom which resulted in the depletion of dissolved oxygen in the sea water. This disaster cost the fish farmers millions of dollars in damages. This event was not the first of such cases—similar events happened on January 2010 and June 2013. The authorities and fish farmers realised that they would have been able to
minimise their losses if they had proper water monitoring and alert systems in place.

Algal blooms is a rapid multiplication of algae in coastal waters or inland water-bodies, and is recognised by the discoloration in the water from their pigments. It causes the level of oxygen in the water to deplete rapidly, clogging up delicate fish gills and blocking out sunlight for aquatic vegetation. Among these blooms, there are some that produce toxic or harmful effects on the ecosystem and mankind, these are called harmful algal blooms, or HABs. The toxins produced by HABs can kill fish, mammals and birds, and may cause human illness or even death in extreme cases [110]. It also contaminates drinking water and renders the water unsafe for any form of human activity. Although it is known that many factors contribute to algal bloom, how these factors create the “bloom” is still not well understood [110]. Some of these factors that contribute to the bloom are nutrients (in flow from agriculture or untreated sewage), thermal stratification, and tidal conditions. Research conducted by the National Oceanic and Atmospheric Administration (NOAA) of the United States of America (USA) showed that human activities play a role in the increased occurrence and intensity of some blooms. Algal bloom is not just an issue for Singapore, it happens worldwide. Many countries, such as the USA, China and Brazil, suffer from such “natural” disasters in large scale. Some of these blooms are large enough to be captured and monitored by satellite over hundreds of square kilometres, and some are in the order of less than few square kilometres.

A SGD4 million water monitoring system named Neptune was launched by the National Environment Agency (NEA) in 2013 and was operational by April 2014 (see Figure 1.2). This system consists of eight buoys stationed around the coast of Singapore and allows for near real-time coastal water monitoring. Neptune also has the ability to send alerts related to water quality through the NEA smartphone app, MyENV. However, just when the authorities and fish farmers thought they were ready for future algal blooms, the same disaster struck again (see Figure 1.3). A repeat of the previous year’s nightmare for fish farmers in the Straits of Johor happened again—thousands of fish had died. The total losses for all the fish farmers in 2015 exceeded those in 2014.

It was clear that the Neptune system, as technically impressive as it was, was not able to predict the catastrophe in time. Like many traditional in situ (fixed network) monitoring techniques, Neptune is unable to provide high temporal resolution over small spatial scales. We posit that a fleet of autonomous robots that are designed according to the swarm robotic paradigm can provide a solution for high spatio-temporal environmental monitoring.

1.1.1 Aquatic Environment Monitoring

Many government agencies, such as NEA and NOAA, are working on monitoring coastal water conditions. They also aim to provide advanced warning to the communities, so they can adequately plan for and deal with the developing algal blooms, oil spills and other threats.
1.1. Motivations

The Neptune system was deployed by the NEA to cover the coast of Singapore (in the order of 5 to 30 km). The NOAA has deployed many buoys around lakes in the USA, the coasts of the USA, in the North Atlantic Ocean, in the North Pacific Ocean and in the Indian Ocean with moored buoy arrays covering tens to few hundreds kilometres [111]. For monitoring large spatial scales (order of 1000 km in length), the NOAA and other organisations use satellite imaging techniques as the changes are large enough to be captured by satellites [2].

The National Research Foundation (NRF) of Singapore has funded research on a mobile wa-
Figure 1.3: News clipping from a daily newspaper, Today, on 5 March 2015, reported on plankton bloom that caused fish farmers along the Johor Straits to lose tonnes of fish.

ter monitoring kayak-like platform led by scientists from the Tropical Marine Science Institute (TMSI) at the National University of Singapore and the Singapore-MIT Alliance for Research and Technology (SMART) [86]. Many other research groups have also developed similar platforms such as the SCOUT autonomous surface craft (ASC) by MIT, the NUWC by DARPA (Defense Advanced Research Projects Agency), and SESAMO by Caccia et al. [30, 41, 101]. However, all these systems are “bulky”, monolithic, sophisticated, and expensive, as they are not designed for rapid deployment or re-deployment. The cost of deployment for such systems are in the order of thousands to tens of thousands of dollars with a complex logistic support. One disadvantage of these systems is the lack of robustness—if one of the vehicles fails the operation is hindered.

Present monitoring techniques rely on either a single ASV or a fixed network of sensors that is unable to provide high temporal resolution over small spatial scales (0.1 km$^2$ – 1 km$^2$). There is a lack of efficacious technology for the robust monitoring and tracking of dynamic environmental features over the spatial scales of the order of few square kilometres, and with the necessary temporal resolution. This is the region where some of the least understood physical dynamics occur in the ocean [82], and these physical dynamics with other biological factors
support the phytoplankton growth which leads to HAB \[100].

1.2 Swarms

Scientists have observed that swarms exhibit collective behaviours. In animal groups, these collective behaviours emerge from following simple rules and repeated local interactions between individual units [26]. Such emergent behaviours show that the whole is greater than the sum of individuals as the group develops swarm intelligence and uses it to achieve collective decisions. All swarms have somewhat similar features: lack of individual leader, local perception of the environment and other agents, and a high degree of adaptation to rapidly changing environments. For the rest of this dissertation we will use collective and swarm interchangeably, where the term—collective—refers to the biological swarm in this case.

1.2.1 Collective Behaviours

Biological swarming behaviours (or collective behaviours) can be seen in daily life. Examples include swarms of ants foraging for food, flocks of birds twisting in the evening, schools of fishes wincing at the thought of a predator and swarms of people (human crowds) (see Figure 1.4). These swarms exhibit self-organisation, a process by which simple rules produce complex patterns without the need for external influences [32]. Self-organisation occurs in nature when atoms and molecules get together spontaneously to form crystal, and crystals combine to form the intricate patterns of seashells [57]. Another example is given by wind blowing across the sand of the desert to produce elaborate patterns of dunes. It also happens when individual cells get together to form structures and patterns such as skin, kidney, heart and brain, i.e. in the general process of organogenesis.

The emergence of these structures and patterns in self-organisation requires the system’s components to either intercommunicate, interact or cooperate. These communications and interactions are typically local ones in the natural world. In order to identify emergence in a complex biological system a clear understanding of the following elements is essential: interactions among agents, informational exchanges, information processing and behavioural algorithms and responses [26,52].

- **Interactions among agents**: This can either be in physical or trophic form. Physical interactions are those that involve fundamental laws of physics, which are responsible for collective phenomena involving inanimate agents. Trophic interactions involve the flow of materials from one agent to another that have specific effects on the metabolism of the recipient.
• **Informational exchanges**: Exchanges of data or information from one agent to another either in unidirectional or bidirectional way that involve one or more sensory modalities. Such information exchanges are critical to artificial swarming systems.

• **Information processing**: The process of filtering and integration of salient signals from externally acquired information. For example, sudden changes in direction due to predator attack are swiftly picked up by fish in a school and birds in a flock.

• **Behavioural algorithm and response**: The behavioural response of an animal is influenced by the information that is acquired externally.

However, having a clear distinction between each of these elements is not always possible, as these elements might be simultaneous and/or interwoven (refer to Ref [52] for more details).

![Figure 1.4: Swarms in nature. (a) A swarm of ants foraging for food. (Credit: Zainichi Gaikokujin, CC BY 3.0) (b) A flock of auklets exhibit swarm behaviour (Credit: U.S. Fish and Wildlife Service, D. Dibenski, public domain). (c) A school of fish swimming along with a ray (S.E.A Aquarium™, Singapore). (d) Lane formation of human crowds getting into train station after New Year celebration (Raffles Place MRT station, Singapore).](image-url)

Collective behaviours are assumed to be purely about collective order in space or collective synchronization in time. However, statistical physics shows that systems with non-apparent order may develop a certain level of “hidden” organisation in interacting many-body systems [26]. These systems are typically characterised by long-range correlations in fluctuations of the
1.2. Swarms

state variables. For instance, Attanasi and co-authors have observed the existence of collective
behaviour in wild swarms of midges without any apparent collective order [5]. They suggested
that the true hallmark of collective behaviour in biological systems is correlation rather than
order. By comparing the interaction of birds with midges, it has been observed that midges
interact according to a metric distance [5] while birds in a flock interact topologically [9].

1.2.2 Swarm Intelligence

Swarm intelligence (SI) is a crucial emergent property of collective behaviours, which allows a
group to tackle and solve problems in a way that individuals within the group simply cannot
[57]. SI can be observed in ants foraging for food where they lay pheromone on their route in
order to find the shortest path to a food source [44,158]. It can also be observed with termites,
which cooperatively build complex nest structures without having any global knowledge of the
environment [76]. In the beehive, bees use sophisticated dances to recruit other members in the
swarm to follow them to a food source [155] and indicate the location of the nectar source [133].
Fish form schools [119] and birds gather in flocks [71] to have better chances of survival against
predators.

Sumpter has proposed a non-exhaustive list of principles for collective behaviour in ani-
mal societies [140]. This list is built on the basic principles of self-organisation: positive feed-
back, negative feedback, amplification of random fluctuations and multiple interactions [24,32].
Sumpter added the following principles to the list: individual integrity, response thresholds,
leadership, inhibition, redundancy, synchronization, and selfishness. These are the features that
can be observed in collective animal behaviours. We will briefly look at some of these principles
that are applicable and relevant to our artificial swarm design:

- **Positive feedback**: It amplifies events through recruitment or reinforcement [25]. For
  instance, when an ant finds a food source it will lay pheromone on the trail to recruit
  others to follow. Another example is when a group of people recommend a certain food
  stall, others will go and try it.

- **Negative feedback**: It stabilises the collective pattern by counterbalancing positive
  feedback in the form of saturation, exhaustion, or competition. For example, when the
  food source is too crowded for insects (bees or ants) to collect food, the new arrivals will
  search elsewhere.

- **Amplification of random fluctuations**: It includes events like random walks, errors,
  and random task-switching. Randomness enables the system to discover new solutions
  and fluctuation acts as seeds from which structures nucleate and grow. For instance,
  foragers may get lost due to some level of error in following the trails. This can lead to
  new food sources in unexplored areas.
• **Multiple interactions**: It provides the minimal density of mutually tolerant individuals that is required by self-organisation. This allows individuals to make use of the results of their own activities as well as others’ activities. For example, ant trail networks can self-organise and be use collectively when individuals use the pheromone laid down by others.

• **Leadership**: It indicates that some key individuals are the catalysts in insect societies, even though the notion of leadership seems incompatible with self-organisation [126]. Take for instance the shaking signal in honey bees; foraging begins in the morning, those bees that find food will shake the other bees deep in the hive [133] to start the foraging activity [132].

• **Redundancy**: It allows the system to continue to function even when faced with a major reduction in its workforce. For example, if some foragers in a honey bee colony are removed, the swarm will rapidly replace them by a new cohort of younger bees in order to survive [140].

### 1.2.3 Biologically-inspired Algorithms

In order to better understand the underlying mechanisms and dynamics of collective behaviours, scientists have sought and studied algorithms that mimic natural swarms. Some of these algorithms help us solve analogous engineering problems. There is a vast breadth of such algorithms to date, and without being exhaustive we will list a few examples below.

**Foraging**

Dorigo and co-authors introduced the first ant colony optimisation (ACO) algorithms, *The Ant System*, based on the ant foraging behaviour in finding the shortest paths from food source to their nest [48]. They have successfully solved the Travelling Salesman Problem (TSP) with *The Ant System*. The idea behind the TSP is to find the shortest route that allows a salesman to visit each city only once. Since then, ACO has been successfully applied to a variety of problems [49] such as scheduling [19], vehicle routing [58], DNA sequencing [20] and data mining [117].

**Cellular Automata**

Cellular automata (CA) originated from the study of self-reproducing automata by the mathematician John von Neumann. Self-reproducing automata were later picked up by John Conway and introduced as the “Game of Life” [39]. The concept of the “Game of Life” evolved into the cellular automata, which is still widely studied in the field of artificial life. CA offer a simple and
minimalist framework with local interacting agents approach that allows scientists to model, understand, and simulate dynamic collective behaviours [159].

Self-propelled Particles

The first widely-known flocking simulation (self-propelled particle model) was introduced by the computer graphics expert Craig Reynolds [125], who was primarily motivated by the visual appearance of a few dozen coherently flying objects [154]. He used small triangular objects called “boids” to simulate a flock of birds for movie animation. These boids move, dive and disappear into the distance in a manner highly reminiscent of flocks of real birds. Such lifelike behaviour would seem to require very complicated and sophisticated programming, however, these boids follow only three simple rules: collision avoidance, velocity matching, and flock centering. Each boid behaves in a manner whereby it keeps an “optimal distance” from its nearest neighbours in order to avoid collision. Reynold’s model shares features with an earlier simulation carried out by Aoki [3], who used the following rules in order to simulate the collective motion of fish: (a) avoidance, (b) parallel orientation movements and (c) approach. The speed and direction of the individuals were considered to be stochastic, but the direction of the units was related to the location and heading of the neighbours. Aoki’s study showed that collective motion can occur without a leader or the individuals having information regarding the movement of the entire school.

Another self-propelled particle (SPP) model similar to flocking was published by Vicsek and co-authors [153], which is widely referred to as “Vicsek Model” (VM) [154]. This simple model allows scientists to establish a quantitative interpretation of the behaviour of large flocks in the presence of perturbations (a statistical physics type of approach to flocking). VM considers only the alignment of the flocks and level of perturbations (noise) that affect the collective motion of the model, and the perturbations are taken into account by adding a random angle to the average direction of the flocks. This simple model also displays a second order type of phase transition from a disordered to an ordered (particles moving in same direction) state as the level of perturbations decreases (see section 3.4).

Consensus Problems

Consensus is an important problem in the area of cooperative control of multi-agent systems. The main idea of consensus is to develop distributed control strategies that enable a group of agents to reach an agreement on certain quantities of interest [93]. Consensus problems (or agreement problem)—convergence to a common value—have a long history in computer science [98]. Formal study of consensus problems in groups of experts originated in management science and statistics in 1960s [43]. The ideas of statistical consensus theory by DeGroot reappeared two decades later in aggregation of information with uncertainty obtained from multiple sensors [16]
and medical experts [157].

In multi-agent systems, a consensus protocol (or algorithm) is an interaction rule that specifies the information exchange between an agent and all of its neighbours in the system [114]. Existing consensus protocols can be roughly categorized into two classes: consensus without a leader and consensus with a leader. The case of consensus with a leader is also called leader-follower consensus or distributed tracking.

The study of the alignment problem involving reaching an agreement—without computing any objective functions—appeared in the work of Jadabaie et al. [81]. They provided a formal analysis of emergence of alignment in the simplified model of flocking, the VM. The theoretical framework for posing and solving consensus problems for networked dynamic systems was introduced by Olfati-Saber and Murray [115, 116] and building on the earlier work of Fax and Murray [54, 55]. The controllability of leader-follower multi-agent systems from a graph-theoretic perspective was considered by Rahmani et al. [123]. Distributed tracking control for multi-agent consensus with an active leader by using neighbour-based state estimators was addressed by Hong et al. [73].

Lessons learnt from the animal kingdom (including insect societies), scientific rules, and computer algorithms all enable us to better design a fleet of robots that swarm together to perform certain tasks. However, in order to operate the fleet of robots at the system level by issuing global objectives (such as aggregation, collective navigation, dynamic area coverage, mapping, sensing target area, etc), these objectives have to be mapped onto individual agent-specific commands. Such a process is known as cooperative control algorithm, and the particular form it takes determines the effectiveness of the large-scale collective behaviour of the system [40] (see section 4.2).

1.2.4 Swarm Robotics

Over the years, various expressions have been used to describe a group of simple physical robots that work together such as cellular robotics [17], collective robotics [89] and swarm robotics [47]. These terms are often overlapping and ambiguous. In this dissertation, we adopt the definition of swarm robotics by Şahin [130]:

*The study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment*

Based on this definition, one can easily differentiate swarm robotics systems from other multi-robot approaches [79]. Nevertheless, Şahin further emphasised a set of criteria for distinguishing swarm robotics research from multi-robot systems that do not have swarm behaviours [130]:
• **Autonomous robots**: The robots should have a physical embodiment and can physically interact with the world autonomously.

• **Large number of robots**: The minimum group size is 10 – 20 robots. However, the control algorithms must be able to accommodate for scalability.

• **Homogeneity**: The robots should be identical, at least at the level of interaction.

• **Simplicity**: The robots should be simple or relatively incapable on their own with respect to the task.

• **Local interaction**: The robots should have local sensing and communication capabilities.

Swarm robotics systems are often inspired by natural systems in which large numbers of simple agents perform complex collective behaviours through repeated local interactions between themselves and their environment, e.g. flock of birds and school of fish. It provides unique solutions in situations when different activities must be performed simultaneously, in high redundancy without a single point failure, and in situations where infrastructure for centralized control is technically infeasible. The central features of swarm robotics include: robustness, scalability and flexibility. It should have the robustness to cope with the loss of multiple individuals, as it is promoted by redundancy and lack of leader. In term of scalability, the robotics swarm must be able to perform equally well irrespective of the group size. The removal or addition of individuals should not result in drastic changes in the performance of the swarm. Flexibility is the ability to cope with a broad spectrum of different environments and tasks.

Here, we present three basic swarm robotics tasks that have been widely used by robotics scientists (refer to Ref [14,29] for the comprehensive lists of swarm robotics tasks):

• **Aggregation**: It is to group all the robots of a swarm in a region of the environment. Despite being a simple collective behaviour, aggregation is a very useful building block, as it allows a swarm of robots to get sufficiently close to one another so that they can interact.

• **Navigation**: A collective navigation is one where a robot with limited sensing and localization capabilities is able to reach a target location with the help of other robots [14]. Each robot in the swarm is sharing its knowledge with the rest of the robots.

• **Flocking**: This is when robots move in formation similarly to schools of fish or flocks of birds. For a group of autonomous robots, coordinated motion can be very useful as a way to navigate in an environment with limited or no collisions between robots and as a way to improve the sensing abilities of the swarm [83].
Swarm Robotics Systems

Through the inspiration from social animals, scientists aim to develop robotics systems that exhibit swarm intelligence features similar to those that characterise natural swarms. There are many research groups working to achieve this aim in order to use the robotic systems to solve complex and difficult engineering problems [29]. Most of these projects are based on ground robots, and we list a few of the them here: Swarm-bots [50], Kobot [150], TERMES [120], and Kilobot [128].

Recently, there have been some breakthroughs in aerial robotic swarms. Lindsey and co-authors demonstrated the construction of cubic structures with quadrotor teams [94]. Vásárhelyi and co-authors successfully implemented flocking strategies on a small flock of 10 quadcopters flying in an unstructured environment [152].

To the best of our knowledge, there was no true swarm robotics system for the monitoring of aquatic environments until Duarte and co-authors demonstrated a fleet of 10 commercial mono-hull boats modified to host a single-board computer [51]. Their experiments focused on developing an automatic design approach for the decentralised control strategy based on evolutionary computing techniques. Their control strategy shows promising results, however more progress is required in order to successfully apply it to collective operations in uncontrolled environments with a much larger number of platforms. Next, closest to swarm robotics system is the work by Valada et al. [151]. They have demonstrated a group of five robots performing distributed environmental monitoring missions. However, their robots need to be controlled by a central computer. Another swarm-like (not swarm) robotics system was presented by Jeffe et al. whereby 16 autonomous miniature underwater robot drifters were developed for water monitoring [82]. Their systems are more like distributed robots than swarm robots, as there is neither collective behaviour nor swarm intelligence. The rest of the water monitoring systems are traditional solutions which are typically costly, monolithic, bulky and incapable of providing spatial and temporal coverage [41,101,121].

1.3 Research Objectives

Although the motivation for this research originates from the rise in the occurrence of algal blooms, it is not limited to algal bloom monitoring. There is also a lack of efficacious technology for the robust monitoring and tracking of dynamic environmental features over the spatial scales of the order of few square kilometres, and with the necessary temporal resolution. Present monitoring techniques rely on either a single autonomous surface vehicle (ASV) or a fixed network of sensors that are unable to provide high temporal resolution over small spatial scales (0.1 km$^2$ – 1 km$^2$). Small spatial skills are the least understood regions in terms of their physical dynamics [82]. This dissertation addresses the gap in technology and science that is related to a
solution for the pervasive and permanent monitoring of water quality with adaptive resolution in both space and time. We study and discuss an innovative solution for the dynamic monitoring of aquatic environments through designing a fleet of decentralised cooperative control ASVs that follow the swarm robotics paradigm.

Through the literature review of the consensus problems in multi-agent systems in section 1.2.3, we realised that most of the studies focused on establishing the influence of the interaction network topology on the capacity of the collective to reach a consensus—its speed to consensus and stability. However, the question of how the network topology influences the system’s responsiveness when subjected to perturbation remains unanswered. Therefore, we study the relationship between the number of interactions and the responsiveness of the collective. In addition, we investigate the effect of forced switching on the responsiveness of the collective.

The goal of this research is to study the system-level design principles and develop a range of cooperative control strategies. Second, a fleet of ASVs is assembled based on the uncovered principles of dynamic collective behaviours. The cooperative control strategies are tested using this fleet of ASVs by performing collective behaviours such as flocking, navigation and dynamic area coverage. Finally, this research also studies the influence of the topology of the interaction network on the responsiveness of the system by using two distinct modelling approaches: a distributed linear leader-follower consensus protocol and the Vicsek Model. This research is part of a larger scale research initiative carried out at the Applied Complexity Group at SUTD [27], which is to have a swarm of ASV with distributed computing ability to monitor their environment and real-time field reconstruction. These are the two objectives for this research:

1. Study the influence of the topology of the interaction network on the responsiveness of the system by using a distributed linear leader-follower consensus protocol and Vicsek Model. First, we study the effect that the number of interactions has on the system. Second, we study the effect that forced switching (rewiring of link between agents) has on the system. These are the two hypotheses for this dissertation:
   - Increasing the number of interactions (connectivity) will lead to an increase in speed to consensus for the collective. However, if the system is overly connected, it will have detrimental effects on the responsiveness of the system.
   - Having access to global information (through forced switching), will lead to an increase in speed to consensus at the expense of the responsiveness of the system.

2. To develop system-level cooperative control strategies and design a fleet of low-cost and mobile floating platforms with the ability to perform swarming capabilities and exhibit collective behaviours—e.g. by performing canonical tasks as alignment, aggregation, avoidance, leader-following, and area coverage—according to the swarm robotics design principles [29,130]. These are the design criteria for the platforms: cost per platform less
than SGD1500, cost for total deployment (whole fleet) less than SGD1000, weight of each platform less than 10 kg, maximum size $0.5 \times 0.5 \times 0.5$ m ($L \times W \times H$).

### 1.4 Thesis Outline

The rest of thesis is organized as follows:

**Chapter 2** describes the design and building process of the swarm of robots. This chapter focuses on the autonomous robot at the unit level, from the individual robot design to the technology that enables all the units to function as swarm.

**Chapter 3** presents the study of the effect of the number of interactions on the collective by using two distinct modelling approaches: a distributed leader-follower linear consensus protocol and Vicsek’s self-propelled particle model. In addition, this chapter presents the investigation of the effect of forced switching, local extended interaction and global interaction, on the two models using network-theoretic approach. The network model by Watts and Strogatz is used for distributed leader-follower linear consensus protocol to provide global interaction. For Vicsek Model, we introduce a new forced switching technique for establishing the link between agent and its new neighbours.

**Chapter 4** introduces the implementation of the cooperative control algorithms and the basic behavioral rules for collective behaviours. It describes and analyses the field experiment of the fleet of robots performing various tasks. It also demonstrates the water quality monitoring capability of the fleet of robots.

**Chapter 5** concludes this dissertation with a short synthesis of achievements and findings. It also describes the contributions and provides suggestions for future work.
Chapter 2

Nuts and Bolts of Buoys

The journey of 50 ASVs from design, to construction, troubleshooting, and deployment starts here. This ASV is a unique buoy-like platform, which is simply referred to as buoy. This buoy is designed to be omni-directional, self-righting, robust and watertight. It is a homogenous group of buoys—all platforms have the same design, dimensions, electronic suite, propulsion system, operating system (software) and sensors (except two which have additional sensors, as mentioned in section 2.1.7).

The motivation of this research arises from environmental issues associated with the increase in ocean and global temperatures from global warming. However, the applications for the fleet of buoys are not limited to water quality monitoring as these buoys can also be used for other purposes, such as during oil spills, for search and report, and for tracking of dynamic environmental features.

This chapter discusses the design process and “the nuts and bolts” of the autonomous buoys. In addition, this chapter will discuss global positioning system (GPS), distributed communication, mesh network testing and buoy characterization. This chapter focuses on the autonomous buoy at the unit level, all details pertaining to the collective operation are dealt with in Chapter 4.

2.1 Buoy Design

The inspiration for the buoy design came from a collaboration with Professor Dick K. P. Yue’s group at the Massachusetts Institute of Technology (MIT) in the United States of America (see Figure 2.1). Together with Professor Yue’s group, we co-designed the fleet of 50 buoys. The final swarming buoy design was carefully thought of in terms of mass manufacturing, mass assembly, storage footprint and stacking up for transportation.

2.1.1 Conceptual Design

This new design is compact, stackable (small footprint) for transportation and storage, and allows for inexpensive batch manufacturing, fast switching of electronic suite, and stable oper-
Figure 2.1: Mobile marine flooding platform developed by Professor Dick. K.P. Yue’s group at MIT (a) Platform with energy-harvesting ability (windmill). (b) Third prototype of mobile platform.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image1.png}
\caption{Mobile marine flooding platform developed by Professor Dick. K.P. Yue’s group at MIT (a) Platform with energy-harvesting ability (windmill). (b) Third prototype of mobile platform.}
\end{figure}

The body is made of a truncated aluminum spherical shell with diameter 320 mm and a wall thickness of 5 mm (see Figure 2.2a). A flange is placed at the truncated top, 80 mm above the center of the sphere, in order to secure the modular lid. There are six mounting holes spread evenly on the side wall, well below the target waterline, where a range of water monitoring sensors can be attached. Figure 2.2b shows a semi-exploded computer-aided design (CAD) rendered view of the buoy with the major mechanical components, printed circuit board (PCB) (section 2.1.5) and vector propulsion system. The arrangement of motors (120° apart, discussed in section 2.1.4) allows the buoy to be driven in any direction (omni-directional) with high maneuverability and fast redeployment as the buoy does not have a bow or stern.

The first prototype of the buoy design was manually fabricated out of styrofoam to enable initial tests of complete assembly and integration of the mechanical design and electronics (see Figure 2.3a). The testings made us realize the advantages of a modular design with the electronics mounted below the top lid for quick inspection and troubleshooting both in the lab and in the field—e.g. we can rapidly address faulty electronics by simply swapping the lid. PCB mounts were 3D printed as a fast solution to ensure the PCB is mounted parallel to the lid to level the compass although creating more screw holes leads to potential leaks. Furthermore, having access from the top lid to all ports connected to the main electrical components (two USB connecting ports, a battery charging connector, and GPS and XBee® antennas) allows for charging and software inspection while keeping the platform fully assembled (see Figure 2.3b).

High-density polyethylene (HDPE) material was initially considered for the hull due to its light weight and good anti-biofouling properties. However, the cost of the mould for plastic injection process led us to consider a more cost-effective solution that is aluminum casting.
2.1. Buoy Design

Figure 2.2: A computer-aided design (CAD) view of buoy. (a) Side-view of the buoy hull with main dimensions. (b) Semi-exploded rendered view of 3D buoy model showing the central parts of the buoy. Not all elements are shown (i.e. battery, XBee, GPS, hardware connections, etc).

Figure 2.3: The prototypes of autonomous buoy (a) First version of buoy made of styrofoam. (c) Fully assembled buoy no. 50 in the field, ready to be deployed.

Metal casting process turned out to be a faster and cheaper way to fabricate 10 to 100 units of the body of our buoy, given the time and budget constrain. The lids were laser-cut from an acrylic sheet.

Upon reception of the first batch of 10 hulls, and after a complete assembly and initial testings, some previous design choices were modified for the subsequent batches: less side openings (future sensors placing) and less screw holes for securing the lid onto the hull, as fewer screws proved equally effective in keeping the integrity of the unit while reducing overall assembly time. The motor pod clamp was 3D printed initially, but it is impractical (slow manufacturing process), fairly expensive and not durable. To counter the costly inefficiency, aluminum casting was chosen for the pod clamps as well.
2.1.2 Hydrostatic Properties

Seakeeping—the ability of a ship to maintain its intended functions at sea in all conditions [149]—is a cornerstone for the success of the operation of our buoys. Therefore, it is important to ensure that our buoy design is able to support the weight of its components (such as hull, motors, battery, etc), to achieve stability, and to be self-righting. Based on Archimedes’ principle, we have calculated that our buoy is able to stay afloat with a load of 8.5 kg with waterline at the center of the spherical hull. The estimated total mass of our buoy with all components mounted on it is approximately 7.7 kg (see Table 2.1).

In order to have a stable and self-righting buoy, the centre of gravity ($G$) must be lower than the centre of buoyancy ($B$) (see Figure 2.4). To achieve that, all the heavy components are placed as low as possible to the base of the buoy. The centre of gravity for the buoy (including all major components: hull, motors and pod clamps, lid, battery, and electronic suite) is calculated by using the following:

$$ G_{\text{buoy}} = \sum \frac{G_i m_i}{m_i} $$  \hspace{1cm} (2.1)

where $G_i$ is the centre of gravity of component $i$ and $m_i$ is the mass of component $i$. Table 2.1 lists the $G$ and $m$ of each major component.

In general, the stability of a ship is dependent on the metacentric height ($GM$), distance between $G$ and the metacentre ($M$), of the ship and is calculated using the following [80]:

$$ GM = BM - BG $$  \hspace{1cm} (2.2)

where $BG$ is the distance from $B$ to $G$. $BM = I/V$ where $V$ is the volume of liquid displaced by the body and $I$ is the moment of inertia of the water-plane area about the longitudinal or transverse axis depending on whether rolling or pitching of the vessel is being considered (see section A.1). Positive $GM$ indicates that the ship is stable and capable of self-righting. The optimum stability of a ship design is when $G$ is below $B$ [63].

Table 2.1: Estimated mass and centre of gravity ($G$) of buoy components, where $G$ is measured from $Y = 0$ in Figure 2.4.

<table>
<thead>
<tr>
<th>Buoy components</th>
<th>Mass (kg)</th>
<th>$G$ (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hull</td>
<td>3.7</td>
<td>137</td>
</tr>
<tr>
<td>Motors and pod clamps</td>
<td>2.8</td>
<td>29</td>
</tr>
<tr>
<td>Lid</td>
<td>0.5</td>
<td>250</td>
</tr>
<tr>
<td>Battery and electronic suite</td>
<td>0.8</td>
<td>40</td>
</tr>
</tbody>
</table>

We did a two-dimensional estimation of the stability of the buoy at angles between $0^\circ – 180^\circ$, since the buoy has a cylindrical symmetry shape. For large angles of inclination, the righting arm, $GZ$, is considered instead of $GM$. $GZ$ is the horizontal distance between the lines of
2.1. Buoy Design

Figure 2.4: Centre of mass of buoys component measured from $Y = 0$ (values obtained from SOLIDWORKS) for calculating the buoy centre of gravity (red dot, $G$). The blue dot represents the centre of buoyancy ($B$). All dimensions are in millimetres.

buoyancy and gravity $[63]$, 

$$GZ_{\text{buoy}} = G M_{\text{buoy}} \sin \phi \quad (2.3)$$

where $\phi$ is the angle of inclination or heel angle in degree. Positive values of $GZ$ indicate that the buoy will be able to return back to its equilibrium position when tilted to an angle. Figure 2.5 shows that our buoy is able to maintain a favourable stability through very large angles of inclination. The instability angle for our design is exactly at 180°. However, a slight force or movement will push the buoy back to the stable regime.

Figure 2.5: Calculation of buoy stability at different heel angles (a) Shows the locations of metacentre, $M$, (green square and dots), centre of buoyancy, $B$, (blue square and dots) that rotated at the centre of gravity, $G$, (red square). Squares represent the initial positions of $M$, $B$, and $G$ when heel angle is 0°, and dots represent the position of $M$ and $B$ at different angles. (b) The buoy is able to self-right as the righting arm remains positive for $\phi$ in the range 0° - 180°.
2.1.3 Hydrodynamic Simulation

The SOLIDWORKS flow simulation package was used to analyze the fluid flow around the buoy and the pressure exerted on the surface of the external wall. The flow simulation is conducted with varying speed of fluid flow from 0.1 to 2 m/s (which is on the higher range of expected velocities) and 5 different angles of attack (0° to 60° with the increment of 15°, see Figure 2.6’s inset). These speeds correspond to a maximum Reynolds number \( Re \approx 6.4 \times 10^5 \) and Froude number \( Fr \approx 1.13 \) (see Figure 2.6).

The drag coefficient, \( C_d \), of the buoy is computed using the following:

\[
C_d = \frac{2F_d}{\rho v^2 A}
\]  

(2.4)

where \( \rho \) is the density of the fluid, \( v \) is the velocity of the fluid stream and \( A \) is the frontal area of the buoy. The drag force, \( F_d \), is estimated using SOLIDWORKS flow simulation. With these simulations (Figure 2.6), we are able to estimate the overall drag coefficient for our buoy, \( C_d = 0.6 \). Figure 2.7 shows the streamlines of flow at the angle of attack that generates maximum drag coefficients, 60°.

In addition, the simulations enable us to predict the power needed for maximum speed and understand the moment induced on the buoy by the drag force. This was used to empirically optimize motor pods placement such that the thrust moment would neutralise the drag moment resulting in level trim during steady-state translation.

![Figure 2.6: Drag coefficients, \( C_d \), of different attack angles obtained from the SOLIDWORKS flow simulation at different Reynolds numbers, \( Re \). Inset: 5 different angles of attack used for flow simulation.](image-url)
2.1. Buoy Design

Figure 2.7: Hydrodynamic computational fluid dynamic (CFD) simulation of a buoy using the SOLIDWORKS flow simulation package. Streamlines of water flowing from lower left to upper right at 1 m/s (Reynolds number, Re ≈ 3.6 × 10^5) from 60° angle of attack, which generates highest drag coefficients.

2.1.4 Propulsion System

An external vector propulsion system [78] is considered as it frees up space inside the buoy for additional battery and other electronic components. This propulsion system eliminates the need of having moving parts through the hull such as motor shaft, which poses a challenge for long term waterproofing. In addition, it increases the buoyancy and lowers the centre of gravity of the buoy. We have implemented a three-pod vector propulsion system that establishes the overall thrust vector [163] for the buoy (see Figure 2.8a). Each pod consists of two motors—one for forward motion and one for reverse motion—but it is specifically designed to allow only one of these two motors to be active at a time through the inline diodes (see Figure 2.8b). This configuration prevents any individual motor from producing thrust that would jeopardise the buoy’s desired motion. These six motors are controlled individually, using a Pololu MC33926 motor driver carrier. This carrier has a full H-bridge with an operating range of 5 – 28 V and delivers a continuous current of 3 A, which is sufficient to meet the motor needs.

An algorithm was developed to accurately control the individual thrust of the buoy for omnidirectional movement. This is achieved by solving two equations with the reference system depicted in Figure 2.8a. The first equation is

\[ \sum M_{Buoy} = \vec{r}_1 \times \vec{T}_{Pod1} + \vec{r}_2 \times \vec{T}_{Pod2} + \vec{r}_3 \times \vec{T}_{Pod3} = 0 \]  \hspace{1cm} (2.5)

where \( \vec{T}_{Podi} \) is the thrust vector produced by pod \( i \), and \( \vec{r} \) the moment radius. Equation (2.5) can be reduced to the following equation as \( \vec{r} \) is the same for all pods and there is no axial (z-axis) movement,

\[ \sum M_{Buoy_z} = T_1 + T_2 + T_3 = 0 \]  \hspace{1cm} (2.6)
Figure 2.8: Motor pods orientation and their control circuit. (a) Thrust pod orientation on buoy, each pod is offset by 120° from pod 1 (reference pod). The thrust direction is indicated by the arrows; the green arrow represents forward thrust and the red arrow indicates reverse thrust. (b) H-bridge diode configuration for motors (only two pods are shown for simplicity), which prevents the operation of forward and reverse motors at the same time within a pod.

where $M_{Buoy,z}$ is the moment about the $z$-axis (orthogonal to the base plane of the buoy or the lid). The next equation is the summation of all thrust vectors in the direction of the desired buoy velocity:

$$\sum \vec{F}_{Buoy} = \vec{T}_{Pod1} + \vec{T}_{Pod2} + \vec{T}_{Pod3} = \vec{T}_{thrust}$$

(2.7)

where $\vec{T}_{thrust}$ is the requested thrust vector to drive the buoy in the desired direction. By solving the following set of equations:

$$\begin{bmatrix} 0 & \cos \theta_1 & \cos \theta_2 \\ 1 & \sin \theta_1 & \sin \theta_2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} T_{thrust,x} \\ T_{thrust,y} \\ 0 \end{bmatrix}$$

(2.8)

where $\theta_1 = 210^\circ$, $\theta_2 = 330^\circ$, we obtain a solution for $T_1$, $T_2$, $T_3$ as follows:

$$T_1 = \frac{2}{3} T_{thrust,y}$$

$$T_2 = -\frac{1}{\sqrt{3}} T_{thrust,x} - \frac{1}{3} T_{thrust,y}$$

$$T_3 = \frac{1}{\sqrt{3}} T_{thrust,x} - \frac{1}{3} T_{thrust,y}$$

(2.9)

With this algorithm, the buoy is able to move in any direction depending on the heading calculated by the cooperative behavioural rules (see Chapter 4).
2.1.5 Electronic Suite

An in-house designed PCB is used to simplify the electronic assembly, while minimising the usage of cable connections (see Figure 2.9). All electronic components are mounted on the PCB except the EZO™ circuit, motor controller, XBee®, GPS sensor, and BeagleBone Black. As the boards have to go through the oven once for the components mounted on the top and then a second time for the ones on the bottom, therefore we placed all the sensitive components on the side that only goes through the reflow process once. The in-house designed PCB allows us to mount all electronic components on the lid for fast switching, therefore eliminating the need of field troubleshooting of electronics when it fails (see Figure 2.10).

![Figure 2.9: Layout of our custom PCB design. Top: Back of the PCB where analogue to digital converter, EZO™ circuit and BeagleBone Black are mounted. Bottom: Top of the PCB where motor controller, IMU, GPS, XBee® are located.](image)

![Figure 2.10: Modular lid where the electronic suite is mounted on for fast switching when there are electronic issues during field experiments.](image)
A block diagram of all the electric subsystems and their relation is shown in Figure 2.11. The entire system is powered by a Revolectrix® Blend435 Black Label 3-cell Lithium Polymer (LiPo) battery with 4.3 V per cell and total capacity of 4600 mAh. This battery supplies direct 12 V to power all the motors and step down to 5 V and 3.3 V for all other electronic components respectively. A single-board computer (BeagleBone Black) is used as a processing unit for the buoy. XBee® and GPS modules are connected to the single-board computer via a serial connection (UART). The I²C communication protocol is used to connect all the sensors to the processing unit. Lastly, the speed and direction (forward or reverse) of the motor pods of the buoy are controlled based on the solution of Equation (2.9) by sending a pulse-width modulation (PWM) from the processing unit to the motor controllers.

![Figure 2.11: Block diagram of the electric components. All the sensors, controllers, and communication modules are connected to a central single board computer that runs the cooperative control algorithms.](image)

There are two sets of sensors on the buoy; one for navigation and one for characterization of the environment. These includes: Inertial Measurement Unit (IMU), GPS (see section 2.1.6), and Atlas Scientific™ sensors (temperature, pH, and dissolved oxygen, see section 2.1.7). The IMU module consists of a compact STMicroelectronics LSM303 sensor with triple-axis accelerometer and triple-axis magnetometer, and a STMicroelectronics low power three-axis digital L3GD20 gyroscope. LSM303 has a 16-bit linear acceleration, with full-scales of $\pm 2$ g (default) up to $\pm 16$ g. The magnetic field of LSM303 has a full-scale of $\pm 1.3$ up to $\pm 8.1$ Gauss. L3GD20 provides a full scale of $\pm 4.4$ up to $\pm 34.8$ rad/s with low-power feature and high shock survivability.

The BeagleBone Black (BBB) microcontroller is selected mainly for its 65 general-purpose
input/output (GPIO), up to 8 pulse-width modulators and 7 analog input ports that make it easy to expand and interface with other devices (see Figure 2.12) [38]. With the ability of generating PWM signal to control motors, it eliminates the need of having an extra microcontroller to interface with motors or other peripherals. This single-board computer BBB has a 1 Ghz AM3358 processor, 512 MB DDR3 RAM, and 4 GB of onboard flash memory. For our buoy, we run a non-native Ubuntu from a 16 GB SD-card as the operating system. This card also acts as an extra storage space for the data that is sent and received by the buoy.

![BeagleBone Black (BBB) pinout diagram.](image)

**Figure 2.12: BeagleBone Black (BBB) pinout diagram.**

### 2.1.6 Global Positioning System

The global positioning system (GPS) is used for localisation as the aquatic environments that we are interested in have access to GPS signal. GPS data is used to control the buoy whereabouts in order to achieve swarming behaviors such as avoidance, aggregation, target navigation and geofencing (see Chapter 4). The accuracy of the GPS module depends on type and cost of the module. We have chosen Adafruit Ultimate GPS, which is capable of tracking up to 22 satellites on 66 channels (up to 10 Hz) and high-sensitivity receiver (-165 dBm) (see Figure 2.13). The sensitivity of this GPS module is improved by using JDGA XP263 Mini Active GPS antenna (providing a gain of 25 dB), that allows us to mount on top of the lid for better reception. The low-power consumption (20 mA) for this module during navigation is ideal for compact mobile system. It has an accuracy of less than 3 metres which is sufficient for our applications.

We typically experience satellite acquisition within one minute following the power on cycle.
at our experimental sites (with clear skies). However, even in these conditions we observed high variability in the GPS localization. The estimated position of some buoys would in some occasions present short-lived pulse-like fluctuations of up to $\sim 10$ m. Since the cooperative control strategies rely on the positions reported by the GPS, these fluctuations would translate into sizable differences between the expected and observed collective behaviour (see Chapter 4). Figure 2.14 shows a buoy that was switched on accidentally inside the transportation vehicle and started tracking the journey from the test site back to SUTD. The GPS module is relatively accurate even when it was inside the vehicle and stacked among the rest of the buoys.

We tested Emild’s Reach real-time kinematics (RTK) global navigation satellite system (GNSS) that is capable of delivering centimetre level of accuracy (see Figure 2.15). It is an appropriate solution for further expansion of our buoy system where applications require centimetre range of accuracy. However, the RTK technology provides a single point failure for the swarm system, which is something we try to avoid in our buoy design. It requires a base station with accurate GPS location to measure the position error. This position error is in turn sent to the rover (in our case the buoy) for corrections. If the communication between base station and rover is down, the system accuracy is back to metre level ($\sim 2.5$ m), which is similar to Adafruit
2.1. Buoy Design

Ultimate GPS. In addition, a secondary communication channel is required for transmitting the information from the base station to each buoy (rover), which may affect the scalability of the system.

![Figure 2.15: Emild’s Reach RTK module.](image)

**Figure 2.15**: Emild’s Reach RTK module.

**Figure 2.16a** shows the results of using Reach RTK to track half of a soccer field on the SUTD campus with two different communication systems: XBee® (represented by red line and dots) and WiFi (represented by blue line and dots). The tracking started from the top left corner of the halfway line (near the stand) and moved toward the centre circle follow by circled around it. Then moved to the touch line (lower right sideline) to goal line and back to the start point. The jagged lines towards the end of the test was caused by the lost of communication (communication devices were switch off to simulate that). We have managed to obtain centimetre level accuracy for repeated trackings of the centre circle using Xbee® communication (see **Figure 2.16b**). In addition, the level of accuracy is less than a metre for two different communication systems.

2.1.7 Sensory Peripherals

The buoy platform is equipped with the capability of data acquisition thereby providing water monitoring for both research and real-world application this research with real world application. For water monitoring, the commonly measured water quality parameters are water surface temperature, potential of hydrogen (pH) value, dissolved oxygen (DO) and salinity. In order to keep the cost of these buoy platforms as low as possible these probes or sensors should be inexpensive, around SGD300 for pH, DO, and salinity setup and less than SGD100 for temperature setup. However, the accuracy and precision of the probes should not be compromised due to cost (usually pH, DO or salinity setup will cost more than SGD1000).

Atlas Scientific™ provides inexpensive probes and kits that meet our requirements. Their mission is “to convert devices that were originally designed to be used by humans into devices that are specifically designed to be used by robots” [96]. In addition, they are committed to producing high-quality components with high accuracy, repeatability and ease of use. Their kits provide the full solution from probe to EZO™ circuit that convert electrical read out from
sensor to respective reading unit, such as degree celsius or kelvin for temperature. These EZO™ circuits have two interfaces, UART and I²C, that communicate with the microcontroller (see Figure 2.17). Atlas Scientific™ also provides many sample codes for the microcontroller to control and acquire data from their variety of sensors.

We have decided to equip all buoys with a temperature probe (see Figure 2.18), as studies have shown that temperature is an important parameter for water quality monitoring [45, 95, 103]. The temperature kit from Atlas Scientific™ comes with a PT1000 temperature probe, EZO™ RTD circuit and thermowell that provide watertight seal. The size of thermowell fits nicely to the buoy circumferential ports 1/2” NTP fitting, that is designed for housing up to six sensors. The temperature range that EZO™ RTD circuit can read is -126 °C to 1254 °C. It has the accuracy of ±(0.10 °C + 0.0017 × T °C).
2.1. Buoy Design

Figure 2.18: Atlas Scientific™ PT1000 Temperature kit (a) Probe with standard EZO™ RTD circuit set up by Atlas Scientific™ (b) Probe mounted on buoy with 30 mm thermowell that extend probe outreach and also provide watertight seal.

We decided to purchase one kit for DO and one kit for pH, to demonstrate that our buoy is designed to be equipped with different sensors. Atlas Scientific™ also sells the probe pipe fitting that separates the circuit with the probe and provides a watertight seal. The DO probe is easier to maintain and use as compared to the pH probe. However, an electrolyte solution has to be added to the sensor if it has not been used for more than one year. The circuit setup and mounting are shown in Figure 2.19. There is one drawback for using a galvanic probe: the measurement is flow dependent as the probe will consume a small amount of oxygen it reads. In order to have accurate readings, water movement of approximately 60 ml/m is necessary. It can measure DO level from 0.01 to 100+ mg/L with an accuracy of \( \pm 0.05 \) mg/L.

Figure 2.19: Atlas Scientific™ dissolved oxygen kit (a) Probe with standard EZO™ circuit set up by Atlas Scientific™ (b) Probe mounted on buoy with probe pipe fitting to provide watertight seal.

Figure 2.20 shows the pH kit and buoy mounted with a probe sticking out from one of the
circumferential ports. The sensing part of the probe has to be capped up with a soaker bottle when the probe is not in use. In addition, the probe has to be calibrated (3-point calibration) with the solutions that come with the kit before data acquisition. This calibration procedure itself is typically hard to automate, and will be challenging when the size of the fleet increases (it is non-trivial to calibrate 20 - 50 probes before every deployment). This probe can read pH from 0.001 - 14.000 with a resolution of 0.001 and an accuracy of ± 0.002.

![Atlas Scientific™ pH kit](image)

Figure 2.20: Atlas Scientific™ pH kit (a) Probe with standard EZO™ circuit set up by Atlas Scientific™ (b) Probe mounted on buoy with probe pipe fitting to provide watertight seal.

Atlas Scientific™ has provided sensors with adequate range and accuracy for the measurement of the aquatic environments. Table 2.2 provides a brief overview of the probes that were purchased, see [97] for more details.

<table>
<thead>
<tr>
<th>Probe Kit</th>
<th>Units</th>
<th>Range</th>
<th>Accuracy</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>°C</td>
<td>-126 – 1254</td>
<td>±(0.10 + 0.0017 × T)</td>
<td>68.00</td>
</tr>
<tr>
<td>pH</td>
<td></td>
<td>0.001 – 14.000</td>
<td>±0.02</td>
<td>164.00</td>
</tr>
<tr>
<td>DO</td>
<td>mg/L</td>
<td>0.01 – 100+</td>
<td>±0.05</td>
<td>283.00</td>
</tr>
</tbody>
</table>

### 2.2 Swarm-enabling Technology

A swarm robotics system consists of autonomous robots with local sensing and communication capabilities, lacking centralized control or access to global information, situated in a possibly unknown environment performing a collective action (see section 1.2.4). In order for the autonomous buoys to swarm and exhibit collective behaviour, the buoy requires some level of interaction between individuals. Each buoy needs to have access to information about the nearby buoys. In the framework of distributed and swarm robotics, this interaction can be divided into “interaction via sensing”, “interaction via environment”, and “interaction via
communication” [15]. For our application of monitoring large bodies of water, interaction via communication is more appropriate so that each agent can actively broadcast a range of state variables to nearby agents.

An artificial swarm is capable of performing global collective actions under a wide range of group sizes (scalability), despite the possible sudden loss of multiple agents (robustness), and under unknown and dynamic circumstances (flexibility) [29]. To ensure the scalability, robustness, and flexibility of our collective system, each buoy should be able to communicate in a distributed mesh network where buoys can be added or subtracted during operations. To grant these features to the buoys, we equipped all the buoys with XBee-PRO® Series 1 modules that are capable of creating a distributed mesh network based on the DigiMesh® protocol.

2.2.1 XBee–Pro® Module

The XBee–Pro® Series 1 (Figure 2.21) provides an off-the-shelf, low cost, low power, and relatively long range wireless communication (750 m outdoor line-of-sight as advertised) solution for our buoy design. We coupled the XBee–Pro® with an external antenna Connectorized Quarter-Wave Monopoles (or Half-Wave Dipoles) by Linx Technologies®, which provides 1.1 dBi of gain in order to extend the communication range. It is capable of creating a distributed mesh network based on the DigiMesh® 2.4 GHz protocol, which automatically reconfigures the network as the agents move into or out of each other’s communication range.

![Figure 2.21: XBee-Pro® Series 1, based on the DigiMesh® 2.4 protocol.](image)

The field testing of XBee–Pro® shows that its communication range is approximately 300 m (instead of 750 m) (see Figure 2.22). The experiments were conducted in various environments, such as the carpark at East Coast Park, the soccer field at SUTD, and Bedok reservoir, using 2.4 GHz frequency band. We observed interferences from nearby devices and surrounding environments, with reservoir having the least interferences where a received signal strength indicator (RSSI) is no lower than -82 dBm at 300 m. The data points were obtained by using DigiMesh analysis tool between two XBee–Pro® modules over 200 iterations of 32-byte messages per point. Our results are in good agreement with the empirical formula RSSI = $-A - 10n \log_{10} D$.
discussed by Kumar et al. [90]. Fitting this formula to our experimental data, we obtained

\[
\text{RSSI} = -(17.8 \pm 2.5) - 10 \times (2.57 \pm 0.12) \log_{10} D ,
\]

where RSSI is expressed in dBm, and \( D \) is the distance between the modules measured in metres.

Figure 2.22: Xbee-Pro® received signal strength indicator (RSSI) measurement tests done at SUTD, East Coast Park (ECP), and Bedok reservoir (BR). The fit for data (purple line) is given by Equation (2.10), the empirical law mentioned in Kumar et al. [90].

The DigiMesh® offers a peer-to-peer mesh network with added network stability through self-healing and allows for tighter control of code space. It is a homogenous network where all nodes are configured as routers which allows them to route data. It has the flexibility to expand the network when more nodes are added in and the reliability where nodes come and go due to interference, damage or when they are out of range. DigiMesh® also provides a multi-hops function to ensure that the broadcast data reaches all nodes in the network.

Figure 2.23 shows an example of a typical communication mesh network obtained during a field test. At that given instant, the mesh network consists of three parts: an isolated node (top right) waiting for other nodes to come in its range, a cluster of three nodes (top left) strongly connected to each other, another cluster with highly connected nodes with an articulation node (green) that bridges the two clusters together. Using this mesh network, we can design temporary isolated buoy or buoys to work independently and not hinder the collective buoys operation—the XBee–Pro® modules allow for unconnected components to work independently, and for them to dynamically join other components or split into smaller groups. When the collective buoys are in communication with the isolated buoy, the collective will “pick up” the isolated one.
2.2. Swarm-enabling Technology

The DigiMesh® network is used for buoy communication, where the buoys use it to continuously broadcast their states—their current GPS coordinates, heading, behaviour, and environmentally sensed data—at a rate of 0.1 Hz. The maximum tested communication range is about 300 m (see section 2.2.1) and the modules are capable of relaying messages through multiple hops in the network (default is 7 hops). In principle, if the fleet of buoys operates within the effective communication range, they should be able to broadcast their state globally to all buoys (all-to-all connection). However, our field experiments show that in real and uncontrolled environments the effective communication range of the mesh network is significantly smaller.

Figure 2.24 shows the percentage of successful communications obtained during dynamic area monitoring field experiments (see section 4.3.5) as a function of the distance between buoys. These experimental results provide a measurement for effective communication range in a large and dynamic network of mobile XBee units. Interference between buoys will cause more messages to drop as the number of buoys increases (comparing \( N = 40 \) against \( N = 20 \) in Figure 2.24). This highlights that our distributed control algorithms should take this issue into consideration (drop messages with more buoys), in order to design a robust collective behaviours.

The swarming-inspired design principles and, in particular, the cooperative control strategies (see Chapter 4) are such that the successful collective operation of this system does not require a reliable, dynamic, and global communication network between all the buoys. As
Figure 2.24: Experimental data of relatively successful communications established between two buoys separated by a certain distance (m) inside the fleet buoys, spread with a mean nearest neighbour distance of $\langle r_{ij} \rangle$. Data was measured during the dynamic area monitoring experiment (see section 4.3.5).

the communication network is imperfect, with limited bandwidth, and limited communication range.

### 2.2.3 Buoy In-house Software

The buoy graphical user interface and operating system are written in Python, as there are many libraries and ready algorithms for controlling our electronic components such as Xbee®, GPS, motor, sensor, etc.

**Buoy Graphical User Interface**

Python offers a cross-platforms solution that allows us to build graphical user interface (GUI) (see Figure 2.25) using Tkinter for the field experiment control station regardless of the computer operating system (Windows, Linux or MacOS).

The GUI consists of two tabs (or pages): one for monitoring the fleet of buoys (see Figure 2.25), and the other is for the reconstruction of field data (see section 4.4). The data—e.g. location of each buoy, heading direction, distance to goal, scalar field information, and battery level—of the buoys are intercepted from the communication network where the buoys broadcast their information. In the fully autonomous case, no specific messages are sent from the GUI to the fleet of buoys with respect to buoy operations. However, in the development phase the GUI enables user to control a single buoy or multiple buoys remotely by sending commands from GUI to buoys. Additionally, the GUI allows for finer-grained control of the system during experimentation.
Figure 2.25: Graphical user interface (GUI) for monitoring movements of buoys and sending behavioural commands to buoys to perform certain tasks during field experiments.

The main GUI tab that is used for monitoring and controlling the swarm contains all the necessary button layouts and plots to give the passive observer a full understanding of fleet status (see Figure 2.25). It is divided into three sections:

- **Upper Section**: It is filled with buttons, and each considered for a specific task. These tasks are embedded with specific messages that can be easily sent to single or multiple buoys in order to direct behaviours as seen fit for the desired operation during the development phase.

- **Middle Section**: It provides a window into buoy communications with a live streaming of messages intercepted from the communication network. This data can be used to validate behavioral expectations or to capture message acknowledgments.

- **Lower Section**: It is devoted to graphically displaying information intercepted by the GUI. On the left, a real-time plot of buoys’ locations overlaying with Google map is shown. On the right are the plots of other parameters: thrust direction, range to goal, and scalar field gradient. It is useful to plot these parameters in order to validate proper operation of the buoys.
Buoy Operating System

The buoy operating system consists of four main threads (see Appendix B for the setup procedure of the operating system on the buoy): (1) GPS, (2) Xbee, (3) Buoy Data, and (4) Buoy thread. These threads are initialized and run simultaneously once the buoy is switched on. The functions of these threads are as follows:

- **GPS**: It is used to set up the GPS module to track buoy location for navigation purposes. It translates data in latitude and longitude into metre. In addition, it calculates the distance between two latitudinal and longitudinal points using the Haversine formula [127].

- **Xbee**: It is used to setup XBee-Pro® for mesh network communication. It encodes outgoing messages and broadcasts the messages. This thread also reads incoming messages and decodes the messages for the buoy to use.

- **Buoy Data**: It parses data received from other buoys and packs the data into a standard data stream. This data is stored in the memory (maximum storage 5000 lines) and in turn used for finding the nearest neighbours and temperature field reconstruction calculation. In addition, this thread prepares three output files (BuoyOUT, GPSOUT and TEST) and writes the data (buoy identification, time stamp, GPS, target location, sensed data and scalar field gradient) into the files according to a given format.

- **Buoy Thread**: Information from the GPS thread, Xbee thread, Buoy data thread, motor, battery and sensors are all passed to this thread. It uses the input information to compute the heading, goal direction and bearing, motor thrust and in turn control the buoy swarming behaviours. This thread is essentially the main program for controlling the action and communication of the buoy with other buoys or control station. In addition, it allows us to control individual buoys manually, which is useful for deploying and retrieving of the buoys.

With the aims of offering a platform-agnostic framework [36] and sharing our cooperative control algorithms with other artificial swarm researchers and practitioners, we looked into porting our Python algorithms into Robot Operation System (ROS) [122]—a commonly used operating system by robotic scientists and engineers. Porting our buoy operating system over to ROS was straightforward as ROS takes Python code, the whole process was completed within a few weeks with some modifications and addition of C++ codes.

### 2.2.4 Robot Operating System

In ROS, the packages form the atomic level of the operation system. Each package contains ROS runtime processes, libraries, and configuration files, that are used to create programmes
within ROS. A node is a process where computation is done. A typical robotic system will have many nodes that control different kinds of functions. This ROS filesystem ensures flexibility, modularity and reusability.

We have created the following ROS nodes (similar to buoy operating system) for the buoy’s operation:

- **GPS**: Initializes GPS module and processes incoming GPS messages (National Marine Electronics Association, NMEA, standard) for information such as latitude, longitude, date and time used for buoy operation.

- **AHRS**: Attitude and heading reference system (AHRS) is used for navigation purposes. It processes attitude and heading data from the IMU sensor according to ROS REP-103 standard using Madgwick AHRS algorithm [99].

- **FSM**: Finite state machine (FSM) is the core of the system, similar to Buoy thread in buoy operating system (Python codes), where all information is passed to it.

- **Scheduler**: Evaluates timed commands from local files and/or received over radio.

- **Data logging**: Logs sensed and received data to output files.

**Finite State Machine**

A Finite state machine (FSM) is used to conceptualise the three main stages of our buoys’ behaviours. These stages are sensing, decision-making, and movement. Each stage corresponds to a state in the FSM. We implemented the FSM in ROS using the ROS SMACH package developed by Bohren and Cousins [21].

The organization of our buoy FSM is shown in Figure 2.26. It starts with *Initialisation State* where the shared resources and communications are initialized. If the initialization fails the machine will repeat the initialization until it is successful. Once the initialization is successful, the machine will enter the main loop via the *Active State*, where the contents of the communications received are processed and “push” into the shared memory. Next, the machine will move to *Behaviour Choice State*, whereby it is a sub-machine by itself. It will generate the next waypoint for the buoy based on the behavioural rules. This waypoint is then passed down to the *Motor Control State* where it will calculate the propulsion thrust and control how much power to be sent to each motor-pair.

Inter-state communication is done through *Shared Object*, which is initialized at start-up and passed to every state. All the states are able to modify the shared memory based on new data from any ROS node. *Behaviour Choice*, a sub-FSM, contains states that correspond to basic rules, which is used to define the cooperative control strategies (see Chapter 4). This setup provides the flexibility of changing control strategies on-the-fly during operation.
2.3 Buoy Characterisations

2.3.1 Speed Analysis

One of the tests that we did was the speed test. The speed test was conducted in a large indoor water tank at the Singapore-MIT Alliance for Research and Technology (SMART) laboratory. It showed that our buoy is capable of traveling from 0.08 m/s (at 25% motor power) to 0.44 m/s (at 100% motor power) and has a maximum angular speed of \( \sim 4 \text{ rad/s} \). This speed range is further validated by field experiments (see Chapter 4). Figure 2.27a shows the analysis of speed from two runaway buoys (we lost control of these buoys after we deployed them on the test site) that travelled approximately 320 metres. However, these buoys provided us an opportunity to complement our speed test done in the laboratory at maximum motor power. The average speed for Buoy34 is estimated to be 0.45 m/s and 0.36 m/s for Buoy40, which is consistent with our laboratory test. Another speed analysis was performed on three different swarming behaviors: collecting, expanding (spreading) and leader-following. It shows that the buoy’s speed is between 0.05 m/s to 0.45 m/s during swarming operation (see Figure 2.27b).

2.3.2 Battery Life Analysis

Presently, the power consumption of the buoy is estimated based on the current drawn by the electronics and motors. We made a conservative fixed current assumption for the electronics to be 238 mA. The current drawn by the motors is obtained from the motor controller through its built-in feedback system. Figure 2.28 shows that our estimation is different from the actual balance of battery power after a 2.5 hours of field experiment performing various tasks. The average value from our battery estimation algorithm is \( \sim 27\% \) and the actual average of the
2.3. Buoy Characterisations

Figure 2.27: Speed analysis from field experiment data. (a) Average speed analyzed from two runaway buoys during one of the field experiments that travelled approximately 320 metres (almost straight path). (b) Histogram of speed analysis of experiment data for collecting, expanding and leader-follower collective behaviours.

balance power is \( \sim 14\% \). Despite the difference, we can safely assume that our buoys are able to operate for 2 hours while performing a range of tasks. However, if the buoys are continuously travelling at full speed then the battery life might drop to half the time (worst case scenario). Our present design allows for the addition of an extra battery to meet the needs of a power-demanding task or for extending operating time. We have a built-in safety measure to prevent overuse of battery that leads to degradation. In addition, our algorithm will automatically disable the motor when the battery power drops below 15%.
2.3.3  Budgetary Consideration

Building a low-cost ASV is one of the key objectives for the buoy design, because the interest of this research is to have a fleet of low-cost buoys rather than one bulky monolithic and expensive system. The availability of low-cost and high-performance electronics and low-cost manufacturing from China have made it possible for us to achieve this aim. Table 2.3 lists the costs for the main components that were used to build an autonomous buoy. We have built a fleet of autonomous buoys at the cost of slightly under SGD1500 each.

Table 2.3: List of components and cost for a buoy.

<table>
<thead>
<tr>
<th>Description</th>
<th>Unit Price (SGD)</th>
<th>Quantity</th>
<th>Total Price (SGD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hull</td>
<td>$270.40</td>
<td>1</td>
<td>$270.40</td>
</tr>
<tr>
<td>Pod clamp</td>
<td>$39.88</td>
<td>3</td>
<td>$119.64</td>
</tr>
<tr>
<td>Acrylic lid</td>
<td>$4.00</td>
<td>1</td>
<td>$4.00</td>
</tr>
<tr>
<td>LiPo battery</td>
<td>$81.25</td>
<td>1</td>
<td>$81.25</td>
</tr>
<tr>
<td>Motor controllers</td>
<td>$60.37</td>
<td>1</td>
<td>$60.37</td>
</tr>
<tr>
<td>Motor</td>
<td>$37.62</td>
<td>6</td>
<td>$225.72</td>
</tr>
<tr>
<td>Propeller and adaptor</td>
<td>$7.29</td>
<td>6</td>
<td>$43.74</td>
</tr>
<tr>
<td>GPS</td>
<td>$55.93</td>
<td>1</td>
<td>$55.93</td>
</tr>
<tr>
<td>GPS antenna</td>
<td>$91.00</td>
<td>1</td>
<td>$91.00</td>
</tr>
<tr>
<td>Xbee-Pro®</td>
<td>$47.60</td>
<td>1</td>
<td>$47.60</td>
</tr>
<tr>
<td>Xbee® antenna</td>
<td>$10.77</td>
<td>1</td>
<td>$10.77</td>
</tr>
<tr>
<td>Beaglebone black</td>
<td>$77.00</td>
<td>1</td>
<td>$77.00</td>
</tr>
<tr>
<td>LSM303 compass</td>
<td>$20.93</td>
<td>1</td>
<td>$20.93</td>
</tr>
<tr>
<td>Atlas Scientific™ temperature kits</td>
<td>$95.20</td>
<td>1</td>
<td>$95.20</td>
</tr>
<tr>
<td>PCB and parts</td>
<td>$150.00</td>
<td>1</td>
<td>$150.00</td>
</tr>
<tr>
<td>USB connector</td>
<td>$25.83</td>
<td>2</td>
<td>$51.66</td>
</tr>
<tr>
<td>Electrical connectors</td>
<td>$47.66</td>
<td>1</td>
<td>$47.66</td>
</tr>
<tr>
<td>Plug and cable connectors</td>
<td>$29.69</td>
<td>1</td>
<td>$29.69</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$1,482.56</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.4  Summary

We successfully designed and built 50 compact, omni-directional, self-righting and watertight autonomous buoys at a cost of under SGD1500 each (see Figure 2.29). The cost is considered low compared to all other options presently available. The vector propulsion system implemented on the buoy allows it to move in any direction at speeds of up to 0.45 m/s. The buoy shows an excellent seakeeping capability, as it can self-right when it was thrown into the water and continue with its allocated task. The total weight of the buoy is 7.5 kg, the waterline is slightly lower than the targeted one. In addition, every buoy is equipped with temperature sensor for water quality monitoring with only two buoys having additional sensors (pH and DO) mounted on them.
2.4. Summary

The GPS module affords each buoy with self-localizing capability which then allows it navigate. The XBee-pro® and its mesh communication network allows the buoys to exchange data and broadcast their state to buoys in theirs communication range. BeagleBone Black together with our buoy operation system (BOS) provide the buoy with enough computational power to integrate all components and process information on-board, and navigate the buoy to target locations autonomously. We have also successfully ported our BOS to ROS for flexibility, modularity, and further expansion (larger fleet). Through laboratory tests and field experiments, we managed to characterise the maximum speed (0.45 m/s) of individual buoys, battery lifespan (~ 2 hours) and the percentage of communication success with respect to number of buoys.

As there was an issue of less successful communication due to an increase in number of buoys, we looked into the effect of buoy-to-buoy interaction (also known as connectivity or number of connections) on the swarm. Can the collective system perform equally well with connections that are fewer in number than the buoys in the fleet? In addition, we are interested to investigate the effect of forced switching (rewiring of the interaction network) on the collective system. Will forced switching increase the speed to consensus and the responsiveness of the collective system? Is there a trade-off between them? The results of this study provides important insights about the network of interaction, and in turn help us better design the interacting network for the fleet of buoys. In addition, these results are platform-agnostic and can be implied to other artificial swarming robots.

Figure 2.29: A fleet of 48 buoys lined up for a final check before deployment for field experiment.
Chapter 3

Too Many Cooks Spoil the Broth

It is well known that an increase in connectivity increases the performance of a wide range of cooperative system. However, in the previous chapter, it was shown that an increase in the number of buoys in a swarm actually decreases the communication success rate. In this chapter, we study the effect of the number of interactions (agent-to-agent connections or number of connections) for collective decision-making using two distinct modelling approaches: a distributed linear leader-follower consensus and an agent-based self-propelled particles (SPP) model.

Studies have shown that biological swarms interact at the local level [4, 9], this is probably due to their biological and physiological limitations. The interconnection of all swarming agents is responsible for the effectiveness of the flow of behavioural information from one agent to another. This raises the question of whether having access to global information via connections with distant agents will benefit the collective system as a whole. There is no straightforward way for scientists to study this particular question in biological swarms. However, it is possible to implement some level of forced switching on an artificial robotic swarm such as our fleet of buoys. In this chapter, we also study the effects of forced switching on the collective decision-making with the same two models, but using network-theoretic approach.

3.1 Physical Approach to Swarming Systems

A wide range of swarming systems are characterized by relatively simple dynamical rules with agents interacting locally, yet producing excessively complex emergent collective behaviours. For many decades, scientists from rather diverse disciplines including animal behaviour [118], physics [153], biophysics [146], social sciences [67], and computer science [125] have been fascinated by the emergence of flocking, swarming, and schooling in groups of agents with local interactions. These local interactions are critical for swarms to perform an effective and coordinated response to changing environments. Here are some of the complex systems that scientists studied: flock of birds [4, 104], school of fish [72, 139], swarm of insects [5, 108], human crowd [106], and robotic swarm [85, 128]. The characteristics of the interaction network are known to strongly affect the swarm dynamics [88, 136, 162] and, in particular, its capacity to respond to local perturbations [13, 31, 35, 138].

In general, increasing the number of interactions between agents usually improves the per-
formance of the collective, but it is known that most natural swarms operate with a limited number of connections. For instance, flocking starlings interact on average with 6 to 7 number of conspecifics [9] and swarms of midges [6] regulate their nearest-neighbour distance depending on the size of the swarm. This limited interaction appears to be a behavioral feature and not a direct result of physical limitations of their sensing capabilities. These findings suggest that natural swarms may tune the amount of interaction or number of connections in order to increase their capacity to collectively respond to environmental changes.

Classic phenomenological models of collective motion feature a critical point at a certain number of connections that maximizes the integrated correlation in the swarm [153]. The observed collective dynamics of midges [6] provides experimental evidence that this swarm tunes the amount of interaction—inferred from density—in a way that maximizes correlations. This critical behaviour may help explain why different social organisms seem to self-limit the number of connections, assuming that large integrated correlations do enhance the collective response for the benefit of biological functions such as predator avoidance or foraging.

From the theoretical standpoint, some models of decision-making dynamics predict that over-reliance on social information can render a collective unresponsive to changing circumstances [84, 147]. Models of consensus in mobile communicating agents have also shown that consensus can be reached more efficiently with a limited interaction range [11].

Understanding the consequences of excessive connections is critical for achieving new functional predictions on collective animal behavior [138, 139], and for the study of spreading of behaviors in networked systems such as online communities. From a technological viewpoint, developing a predictive theoretical framework to understand under which circumstances these effects appear is of paramount importance for the emerging field of large-scale swarm robotics [85, 129].

In this dissertation, we investigate the effect of the number of interactions (agent-to-agent connections or number of connections) for collective decision-making using two distinct modelling approaches: a distributed linear leader-follower consensus protocol [102] and an agent-based SPP model known as the Vicsek Model [153], that well represent our buoys system and allows us to study the interaction network of the swarm without solving the complexity of the swarm system.

The distributed linear leader-follower consensus protocol is a linear time invariant (LTI) system with first order dynamics and static connectivity that allows us to solve the model analytically. With that we can subject the system to external perturbation by assigning a leader agent whose dynamics is oscillating—a time-varying input signal into the system—to simulate dynamic changes in environments. In a biological context, this dynamical leader may represent a member of a swarm with access to privileged information about a food source or a threat. The information propagates through the swarm via the agents’ behaviour, and the
collective response of the system measures its efficiency in propagating such information. For artificial multi-agent systems, introducing a leader can facilitate a range of formation control techniques [124]. In this scenario, having a responsive collective behaviour is crucial in the case that the target formation changes with time. Using elements borrowed from the state-space control theory we can solve the first-order differential equations of the dynamic system and study the gain (metric for responsiveness measurement), which is related to the transfer function at the agent level (see section 3.3).

Vicsek Model (VM) is a more realistic representation of a natural swarm as compared to the distributed linear leader-follower consensus protocol. It is a simplified model of SPP that only considers the alignment of agents and the level of perturbations with dynamic connectivity. This model allows us to study the responsiveness of a dynamic network system when it faces perturbation such as noise or a potential predator. Through predator attack simulations using VM, Mateo et al. showed that the predator avoidance ability of a swarm is closely related to its susceptibility [102]. Therefore, susceptibility of the collective motion (using VM) is used to indicate the responsiveness of the system subjected to the perturbation in this dissertation (see section 3.4). We used VM to study the effect of number of connections and forced switching on collective response through analysing the susceptibility of the system.

Lastly, these models allow us to control the number of connections which can range from two connections per agent to the total number of agents in the system (all-to-all connection). With that, these two models are able to provide a wide range of number of connections for the study of the effect of connectivity and forced switching on the systems.

3.2 Network-Theoretic Approach

There is an underlying network for all systems performing collective behaviour such as biological swarms, self-propelling particles, and many other self-organizing systems. This system level organization can be better understood using a network-theoretic approach, also known as “Network Science”. This approach provides an elegant and powerful framework—essentially grounded in graph theory—that links the local dynamics (and interaction) at the agent level with the global response at the swarm level. In fact, network models offer a natural way of describing how self-organization arises in complex systems, which in turn helps us gain insight into dynamical processes occurring on them. In addition, network science has a lot in common with statistical physics—e.g. percolation, scaling, order parameters, phase transitions, and critical exponents—and these similarities remain highly relevant for a network analysis [1,26]. With that, it helps to uncover the hidden structures emerging through self-organization. In addition, it helps to understand complex systems without solving the complex mathematics behind them.

Over the past decades, network science has emerged as a new interdisciplinary field and
gained popularity among scientists and researchers, as it allows them to represent and analyse the intricate underlying structure of connections among elements of complex systems \cite{114,156}. Network science builds on graph theory that can trace its roots back to pioneering work of Leonhard Euler in solving the Königsberg bridges problem \cite{10,12}. It also draws on theories and methods from statistical mechanics, data mining and information visualization, inferential modeling, and social structure. This new field offers novel tools and perspectives for a wide range of scientific problems from the analysis of social network to swarm and drug design.

In network science, a network consists of nodes or vertices and the direct interactions between them are called links or edges. The links of a network can be directed or undirected. It is important to emphasize that the network approach is intentionally limited to mapping the interactions between units, with all the attention focused on the global structure of the interactions within a system. That according to the “network thinking” paradigm, the detailed properties of each element or individual unit are ignored and not accounted for at the network level \cite{26}.

The study of consensus problem in multi-agent systems using network-theoretic approach has received a significant amount of attention in the literature \cite{88,113,116,135}. These studies focused on establishing the influence of the interaction network topology on the capacity of the collective to reach a consensus—its speed to consensus and stability. However, the question of how the network topology influences other desirable properties of distributed multi-agent systems—especially responsiveness to perturbation—remains open. In this dissertation, we used network-theoretic approach to investigate the relationship between forced switching of agents and the responsiveness of the collective using the same two models. In addition, we will use the following network properties to further study our models: average path length, connectedness, clustering coefficient, and Laplacian matrix.

The network that represents the distributed linear leader-follower consensus protocol is considered as an undirected network while the network for VM is a directed one. The degree of a node represents the number of links it has to other nodes. For directed networks, in-degree refers to the number of links pointing inward. Out-degree refers to the number of links pointing outward. For undirected networks, the in-degree is equal to the out-degree. All the links in a network can be represented in a matrix form known as the adjacency matrix, $A$. The elements of adjacency matrix is $a_{ij} = 1$ if there is a link pointing from node $i$ to node $j$, zero otherwise. The adjacency matrix for undirected network is symmetric while the directed one is asymmetrical. The notations \cite{12} for our calculation of directed networks are as follows: summing the columns of adjacency matrix gives the number of out-degrees of the node $i$, $k_{i}^{\text{out}} = \sum_{j=1}^{N} a_{ij}$ while summing the row gives the number of in-degrees of node $i$, $k_{i}^{\text{in}} = \sum_{j=1}^{N} a_{ji}$. 
3.2.1 Average Path Length

In a network, the distance between nodes does not represent the physical distance between the nodes in the physical space. In fact, physical distance is not relevant in the network. For example, two individuals who are interacting have a link although they are living at the opposite ends of the globe. However, any two individuals who work in the same office building might not know each other.

Frequently, the path is used to represent the route that runs along the links between any two nodes instead of a physical distance in network science. Therefore, path length is the number of links (or edges) that the path contains. Any pair of nodes might have more than one path between them, the path with the least number of links is known as the shortest path length (SP) or geodesic path, $d_{ij}$. If there is no link between two nodes, then the shortest path length is defined as $d_{ij} = \infty$. The shortest path length for an undirected network is symmetric, $d_{ij} = d_{ji}$. However, for a directed network SP does not coincide in general, $d_{ij} \neq d_{ji}$.

The average path length (or average shortest path length) is an effective metric for the network size. It is the average distance between all pairs of nodes, $\langle d \rangle$, (see Figure 3.1). The average path length for a directed network of $N$ nodes is [10],

$$\langle d \rangle = \frac{1}{N(N-1)} \sum_{i,j=1, i \neq j}^{N} d_{ij}$$  

(3.1)

Note that when $i = j$, $d_{ij} = 0$ and $\langle d \rangle$ measures node pairs that are in the same component. If the network is disconnected the average path length will be infinite.

![Figure 3.1: Average path length for this simple network is 1.6. \(\langle d \rangle = (d_{1\rightarrow2} + d_{1\rightarrow3} + d_{1\rightarrow4} + d_{1\rightarrow5} + d_{2\rightarrow3} + d_{2\rightarrow4} + d_{2\rightarrow5} + d_{3\rightarrow4} + d_{3\rightarrow5} + d_{4\rightarrow5})/10 = 1.6.\)](image)

3.2.2 Connectedness

The connectedness of a network indicates the reachability of any chosen pair of nodes. A network is connected if there is a path between any pair of nodes (node $i$ and $j$) in the network (see Figure 3.2a). If such a path does not exist, the network is considered disconnected, $d_{ij} = \infty$ (see Figure 3.2b). For disconnected networks, the sub-networks are called components or clusters—a subset of nodes in the network where all pairs of nodes have a path that links them. For the
Figure 3.2: The connectedness of a small network. (a) A connected undirected network, where there is a path between every pair of nodes. The green link is called a bridge that links both the left and right cliques together. (b) The absence of a bridge causes the network to be disconnected, resulting in two components. The nodes in these two components are connected to each others, (1,2,3,4) and (5,6,7,8).

two networks that are considered in this research, there is a minimum number of connections to ensure the connectedness of the network.

For the VM, in order to ensure the connectedness, \( k^{out} \) must be equal or greater than the critical degree, \( k_c \), defined as follows [8, 88]

\[
k_c = c \ln N
\]  

(3.2)

where \( c \) is the coefficient, and the smallest value found so far is 0.9967 [8] for a large number of \( N \). Komareji and Bouffanais studied the values of \( N \) smaller than 1000, and showed that the average value of the coefficient, \( c \), is 1.15 [88]. This value tends to decrease with increasing values of \( N \), which is consistent with the value 0.9967 found in Ref [8] for large \( N \).

For the Watts and Strogatz random network model [156] (used in distributed linear leader-follower consensus protocol), the following is required

\[
N \gg k \gg \ln N \gg 1
\]  

(3.3)

where \( k \gg \ln N \) guarantees that a random network will be connected [22].

### 3.2.3 Clustering Coefficient

Network clustering refers to the tendency of a network to form cliques. In other words, “to what proportion of the neighbours of node \( i \) are also neighbours of each other.” The clustering of a network can be quantitatively measured by means of the clustering coefficient. The global clustering coefficient (also known as transitivity) gives an overall clustering view of the networks [137]. It is defined as [23, 107]

\[
C = \frac{3 \times (\text{Number of triangles})}{(\text{Number of connected triples})}
\]  

(3.4)
Here a “connected triple” means three nodes $uvw$ with edges $(u, v)$ and $(v, w)$ (the edge $(u, w)$ can either be present or not). This global clustering coefficient definition could be applied to either directed or undirected networks.

Watts and Strogatz introduced a local measure of the clustering coefficient of node $i$ with degree $k_i$ for undirected networks, known as the local clustering coefficient [156]:

$$C_i = \frac{2L_i}{k_i(k_i - 1)} \quad (3.5)$$

where $L_i$ represents the number of links between $k_i$ neighbours of node $i$ and $0 \leq C_i \leq 1$. The local clustering coefficient, $C_i = 1$ means that the neighbours of node $i$ are all linked to each other. On the other hand, $C_i = 0$ means that none of the neighbours of node $i$ are connected to one another. When $k_i \leq 1$, $C_i$ is defined as follows $C_i \equiv 0 [12]$. Equation (3.5) can be extended to directed graphs [137]:

$$C_i = \frac{L_i}{k_i(k_i - 1)} \quad (3.6)$$

The clustering coefficient for the entire network is captured by averaging $C_i$ over all nodes, and is called the average clustering coefficient, $\langle C \rangle$, [10,156]:

$$\langle C \rangle = \frac{1}{N} \sum_{i=1}^{N} C_i \quad (3.7)$$

The following definition for the clustering coefficient of a directed network by Fagiolo is used in this dissertation [53]:

$$C_i^{\text{out}} = \frac{(A^2A^T)_{ii}}{k_i^{\text{out}}(k_i^{\text{out}} - 1)} \quad (3.8)$$

This equation allows one to track the clustering coefficient of the out-degree, $C_i^{\text{out}}$ while varying out-degree, $k_i^{\text{out}}$.

### 3.2.4 Laplacian Matrix

Classically, the speed of convergence to consensus can be assessed via the spectral properties of the graph Laplacian. The Laplacian matrix of a graph is defined as

$$L = (D - A) \quad (3.9)$$

where $D$ is the degree matrix and $A$ is the adjacency matrix of the graph. Degree matrix of a graph is a diagonal matrix where all diagonal elements are equal to the number of connected neighbours, $k_i$ of node $i$ and the rest are zeros; $d_{ii} = k_i$ where $d_{ij} = 0$ for all $i \neq j$. The Laplacian matrix has all row sums equal to zero, therefore there exists a zero eigenvalue corresponding to the right eigenvector with identical nonzero elements $(1, 1, ...)$.

This indicates that the rank($L$) ≤
n – 1, for connected undirected network there is an isolated eigenvalue at zero [65], and this is known as the trivial eigenvalue of $L$. For an undirected network, $L$ is a symmetric matrix and has real eigenvalues with ascending order [46]:

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq 2k_{\text{max}}$$

(3.10)

where $k_{\text{max}}$ is the largest degree in the network.

Olfati-Saber et al. defined the graph Laplacian associated with the digraph as $L = D_{\text{out}} - A$ (where $D_{\text{out}}$ is obtained by taking the out-degree of the nodes) and showed that the Laplacian of a strongly connected digraph has an isolated eigenvalue at zero too [116]. Strongly connected digraphs mean that for any two distinct nodes on the graph there exist a path that link them together. In addition, the number of zeros in eigenvalue of a Laplacian matrix indicates the number of sub-graphs in the network for disconnected networks.

When comparing the eigenvalues between graphs, it is often more useful to use a normalised Laplacian matrix of the graph [92]. A normalised Laplacian matrix for undirected network can be defined as [37, 134]:

$$\bar{L} = D^{-1/2}(D - A)D^{-1/2}$$

(3.11)

For directed networks, the normalised Laplacian matrix is defined as [116]

$$\bar{L}_{\text{out}} = D_{\text{out}}^{-1}(D_{\text{out}} - A)$$

(3.12)

where $D_{\text{out}}$ is the out-degree matrix.

The second smallest eigenvalue, $\lambda_2$, of graph Laplacian is used to quantify the speed of convergence for undirected and directed graphs, ($|\lambda_2|$ is used for directed graphs). It is also called the algebraic connectivity, and or known as the Fiedler eigenvalue [92]. It is well-known that $\lambda_2$ for undirected graph depends on the denseness of the graph. For sparse graphs, $\lambda_2$ is relatively small, whereas for dense graph $\lambda_2$ is relatively large [65]. Olfati-Saber et al. showed that for digraphs, $\lambda_2$ also has the same effect as undirected graph on linear consensus protocol [116]. Shang et al. proved that $k \sim 10$ significantly speeds up the rate of convergence to consensus level which is similar to an all-to-all connection (optimal level) for self-propelled particles (Shang and Bouffanais called it swarm signaling network, SSN) [136].

### 3.2.5 Small-World Network

Small-world phenomenon, also known as the six degrees of separation, states that anybody anywhere on Earth is separated by only six other people from any other human individual. There are six acquaintances that connect us to the president of United States of America. Hungarian writer Frigyes Karinthy was the first person to describe the concept of small world property in
his short story, “Chains”, in 1929. Stanley Milgram, an American psychologist, carried out the first experiment to test the small-world phenomena and showed six degree of separation. In his experiment, Milgram randomly selected residents of Wichita and Omaha, and instructed them by letter to forward the letter to his target, a stock broker in Boston, Massachusetts. They were only allowed to send directly to the target if they are acquainted, otherwise they would have to send to someone whom they think is mostly likely an acquaintance of the targeted stock broker [148]. This concept was later popularized by playwright John Guare with his play, titled “Six Degrees of Separation”. Network scientists believe that the degree of separation has decreased after the invention of internet and out-spring of social media. Backstrom et al. showed that is it now four degrees of separation between any two individuals on this planet. They looked at the world scale social network—the entire Facebook network of active users (approximately 721 million users and 69 billion friendship links in 2012) [7].

A small-world network refers to a network that has the properties of small world effect of social and technological network, small path length and high level of clustering [156]. Regular networks, such a ring lattice, tend to be clustered but are not small world. Random networks or Erdős and Rényi model are able to capture the small path length of real-world networks. However, they do not have a high level of clustering. Watts and Strogatz (WS) proposed a network that successfully exhibits the coexistence of high clustering and small path length [156]. The WS model allows users to “tune” the network between regularity (ring lattice) and disorder (random network), by randomly rewiring each link with probability $p$, where $0 < p < 1$ (see Figure 3.3).

Figure 3.3: The Watts and Strogatz model starting with ring lattice ($p = 0$), which can be turned by probability, $p$. Randomness of the networks increases with $p$, by rewiring the link according to $p$. Each link is rewired for maximum $p = 1$, where the network is completely random [156].

The random rewiring procedure for interpolating between a regular ring lattice and a random network, without altering the number of nodes or links in the network, starts with an undirected ring lattice. This ring lattice has $N$ nodes, each connected to its $k$ nearest neighbours. A node and one of its links are chosen to be switched based on a given probability, $p$, ($0 \leq p \leq 1$). If the probability is lesser than what is specified by the user, it will be reconnected to another node chosen uniformly at random over the entire ring. In addition, duplication of a link is forbidden,
if that happens the link remains. These steps are repeated in clockwise direction around the
node on the ring until all nodes and links have been considered once (see Listing 3.1). As there
are $n \times k/2$ links in the entire network, the rewiring process will last for $k/2$ laps. Figure 3.3
shows the three realizations of this process, for different values of $p$. For $p = 0$, the original ring
lattice is unchanged; as $p$ increases, the network becomes increasingly disordered until $p = 1$,
at which all the links are rewired randomly. The intermediate value of $p$ yields a small-world
network: highly clustered like a regular ring lattice, yet with a small characteristic path length,
like a random network.

Listing 3.1: Algorithm for Watts and Strogatz Model

```matlab
function H = WattsStrogatz(N,K,beta)
% H = WattsStrogatz(N,K,beta) returns a Watts-Strogatz model graph, H,
% (aka small work network) with N nodes, N*K edges, mean node degree 2*k
% rewiring probability beta.
%
% Probability of rewiring, beta = 0 is a ring lattice, and
% beta = 1 is a random graph.
%
% Connect each node to its K next and previous neighbours. This constructs
% indices for a ring lattice.
s=repelem((1:N)',1,K);
t=s+repmat(1:K,N,1);
t=mod(t-1,N)+1;

% Rewire the target node of each edge with probability beta
for source=1:N
    switchEdge = rand(K, 1) < beta;
    newTargets = rand(N, 1);
    newTargets(source) = 0;
    newTargets(s(t==source)) = 0;
    newTargets(t(source, ~switchEdge)) = 0;
    [~, ind] = sort(newTargets, 'descend');
    t(source, switchEdge) = ind(1:nnz(switchEdge));
end
H=graph(s,t);
end
```

With a rather small amount of rewiring $p = 0.01$ at $N = 2048$, a network is able to feature
small-world properties of small average path length with relatively high average clustering
coefficient (see Figure 3.4). The average path length starts to decrease when $p < 0.01$ and
tapers off slowly for $p > 0.01$. On the contrary, the average clustering coefficient remains rather
constant for $p < 0.01$ and starts to decrease towards zero after $p = 0.01$. This model has
allowed scientists to study a continuum of networks that are between the two extremes—a
regular lattice and a completely random network.
Figure 3.4: Characteristic average path length (PL) and average clustering coefficient (CC) for the Watts and Strogatz small-world network. PL and CC are normalized with the values of ring lattice for $p = 0$, $PL(0)$ and $C(0)$ respectively. This system has $N = 2048$ and the average degree is $k = 10$.

3.3 Static Network – Linear Consensus Protocol

Here, we consider a set of $(N + 1)$ identical agents (plus one leading agent) performing a distributed linear consensus protocol on their scalar state-variable $x_i(t)$, through a connected and undirected network. The system is characterized by the global state vector $X(t) = \{x_i(t); i = 0, \ldots, N\}$ and the adjacency matrix of the underlying graph $A = \{a_{ij}; i, j = 0, \ldots, N\}$, where $a_{ij} = 1$ if agent $i$ is connected to $j$ and 0 otherwise. Given a certain connectivity graph, the state of the system evolves according to

$$\frac{dx_i}{dt} = \frac{\omega_0}{k_i} \sum_{j=0}^{N} a_{ij} (x_j(t) - x_i(t))$$

$$= \sum_{j=0}^{N} w_{ij} x_j(t) \quad (3.13)$$

where $\omega_0$ is the natural response frequency of our identical agents, and $k_i = \sum_{j=0}^{N} a_{ij}$ is the degree of agent $i$, its number of neighbours in the network sense. The quantity $w_{ij} = \omega_0 (a_{ij}/k_i - \delta_{ij})$—where $\delta_{ij}$ is a Kronecker delta—is introduced for the sake of a compact notation for the governing dynamical equations. Note that, by definition, $w_{ii} = -1$ and $\sum_j w_{ij} = 0$ for all $i$. As is classical with many swarming systems, these dynamics involve relative output information of neighbouring agents [114]. For simplicity, we used a static undirected one-dimensional lattice network (see Figure 3.5) with different topologies that are obtained by changing the number of connections, $k$.

We model the process of distributed transfer of information (response of the system) by considering a leader-follower consensus dynamics. This model is implemented by assigning one
Figure 3.5: Lattice (ring) topological network, with a leader (in green) and followers (in red), with \( k = 2 \) (left) and \( k = 4 \) (right), where \( k \) is the set of neighbours of an agent.

of the agents in the network to be the leader and following an arbitrary trajectory, \( x_0(t) = u(t) \), instead of Equation (3.13), while the rest of the agents follow Equation (3.13). This single control input has a direct effect on the dynamics of the leader’s, \( k_0 \), neighbouring agents. It also has an indirect effect on the dynamics of other agents in the network through the coupled set of dynamical equations (Equation (3.13)). In the presence of a single leader, Equation (3.13) can be rewritten as

\[
\frac{dx_i}{dt} = \sum_{j=1}^{N} w_{ij} x_j(t) + w_{i0} u(t)
\]

for \( i = 1, \ldots, N \).

Within this leader-follower scheme, one can characterise the response of the multi-agent system by measuring its capacity to follow the leader’s trajectory, \( u(t) \). Specifically, with an input signal oscillating at the frequency \( \omega \), i.e. \( u(t) = u_0 e^{j\omega t} \), the state of all agents at long times becomes proportional to \( u(t) \) with a factor given by the transfer function,

\[
H(\omega) = \lim_{t \to \infty} \frac{X(t)}{u(t)} = (j\omega I - W_F)^{-1} W_L
\]

where \( X(t) \) is the global state vector, \( I \) is the identity matrix of dimension \( N \), \( W_F = \{w_{ij}\} \) is the \( N \times N \) consensus protocol matrix between the follower agents, and \( W_L = \{w_{i0}\} \) is the \( N \times 1 \) consensus protocol matrix between the followers and the leader. The elements of \( H = \{h_i\} \) correspond to the frequency response of each individual agent, with \( |h_i(\omega)| \leq 1 \) for all \( i \) and \( \omega \) [112]. As it is clear from Equation (3.15), the response functions have a nontrivial dependency on the topology of the agents’ connectivity through the entries of \( W_F \) and \( W_L \). The collective response of the system can be characterised by performing a singular value decomposition of \( H \), giving a single singular value \( \sigma^2 = \sum_i |h_i|^2 = H^2 \). Throughout this research, we will use \( H^2(\omega) \) (or its normalised form, \( H^2/N \)) as a measure of the collective frequency response of multi-agent systems.
3.3.1 State-space Analysis

The leader-follower consensus dynamics can be modeled using linear time invariant (LTI) state-space approach [114], commonly adopted in control theory, where the leader’s dynamic is considered as the input of the system and the responsiveness of the multi-agent system could be evaluated by the output (see section A.2). The state-space equations are formulated as [112]

\[ \dot{x} = Ax + Bu \]
\[ y = Cx + Du \]

where \( A \) is the state matrix (also known as adjacency matrix in our network-centric approach), \( B \) is the input matrix, and \( C \) is the output matrix. \( D \) is the “feedthrough matrix”, however in our case there isn’t any direct feedthrough therefore \( D \) is the zero matrix. Note that \( A, B, C, D \) are static (constant) matrices in our study.

While this idealised model may be too simple to accurately reproduce the behavior of real systems, it contains the essential ingredients of a linear first order dynamical system. This system is said to be controllable if a signal \( u(t) \) is able to drive the system from a given state \( x_0 \) to a certain \( x_1 \). However, in order for the system to have an effective response to a changing environment, it is just as important to consider how fast the system reaches \( x_1 \). The transfer function can be computed analytically for some specific geometries of the static connectivity of the network. For this, we limited ourselves to the one-dimensional regular ring lattice and the transfer function is

\[ H(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \]

where \( D = 0 \) for our network-centric approach. We evaluated the system output at different frequencies by substituting \( s \) with \( j\omega \) and formed the equation below:

\[ H(j\omega) = C(j\omega I - A)^{-1}B \]

This distributed linear leader-follower consensus model is a good idealisation of the process of social information transference occurring in a swarm, despite the static nature of the topology of interaction. For instance, when one individual (one of the birds in a flock or one of the fish in a school) has privileged information about a potential threat or other kind of external perturbation, it will act as a temporary leader and trigger a wave of agitation that propagates strikingly fast through the swarm [72]. Such waves of agitation are initiated by very rapid changes in the leading agent’s state, which are then very effectively propagated to all other swarming agents [4].
3.4 Dynamic and Switching Connectivity – Vicsek Model

Vicsek and co-authors introduced a simple self-propelled particles (SPP) model that captures some key features of the emergence of self-ordered motion in systems of particles [153]. Over the years, several more elaborate or complex versions of this model have been considered to generate more realistic and specific dynamics [154]. However, the generality and simplicity of Vicsek Model (VM) was kept for our study on the effect of connectivity.

In VM, each particle moves with constant absolute velocity, \( v_0 \), in a two-dimensional periodic space, size \( L \times L \). Every particle aligns its direction of motion with its neighbours’ mean orientation (including itself). The particles update their position according to [153]

\[
\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \Delta t
\]

where \( \Delta t \) is the time interval between two updates, which is classically set to \( \Delta t = 1 \). The velocity of a particle is \( \mathbf{v}_i(t) \). Here, the velocity vector \( \mathbf{v}_i = v_0 \hat{\mathbf{\theta}}_i \), where \( \hat{\mathbf{\theta}}_i = \cos \theta_i \hat{i} + \sin \theta_i \hat{j} \) has a constant magnitude \( v_0 \), therefore the direction \( \theta_i \) of each particle is obtained from the expression

\[
\theta_i(t + \Delta t) = \arg \left( \sum_{j \sim i} \mathbf{v}_j(t) \right) + 2\pi \eta_i(t)
\]

where \( \arg() \) gives the orientation of a vector, and the sum \( j \sim i \) is performed over the neighbours of \( i \) including \( i \) itself. The noise (or perturbation), \( \eta_i(t) \), is a random number uniformly distributed between \( [-\eta/2, \eta/2] \). This extremely simple model allows the simulation of many thousands of flocking agents and displays a continuous phase transition from a disordered to an ordered (particles moving in parallel) state as the level of perturbations decrease Figure 3.7.

3.4.1 Neighbourhood of Interaction

The uncoordinated local interactions between agents guide us in understanding the basic mechanistic functioning of a swarm in terms of collective motion [26]. The central importance of these local interactions have led researchers and scientists to experiment different local interaction rules [135], in order to replicate the swarming behaviours of different species [141, 154]. In general, there are two main groups of local interaction rules: metric neighbourhood of interaction and topological neighbourhood of interaction. Both of these interaction are based on the distance between agent and its nearest neighbours Figure 3.6.

The metric neighbourhood of interaction was first considered [153] and has drawn a significant amount of attention [154]. Agents in the metric neighbourhood framework exchange information with all other agents located within their fixed and given radius—assumed to be the same for all [68–70]. The metric distance was only recently challenged following the anal-
Figure 3.6: An illustration of metric and topological neighbourhoods of interactions with two different densities. $R$ represents the radius of the metric neighbourhood, whereas $r$ is the radius of the topological neighbourhood with $k = 7$ (nearest neighbours, green triangles). Left: Shows a densely populated swarm, where $r$ is smaller than $R$ (in this case the metric neighbourhood has 14 neighbours, blue and green). Right: A less dense swarm, in this case $r$ is greater than $R$ in order to capture $k = 7$ neighbours (shown in green) and 4 neighbours for the metric interaction. In both cases, $R$ remains constant as it defines a metric zone around the agent.

Analysis of empirical data for the dynamics of flocks of starlings [9] and human crowds [64, 105]. Ballerini et al. claimed that interactions based on metric distance are unable to reproduce the density changes, typical of bird aggregations, because one would expect cohesion to be lost when mutual distances become too large compared to the interaction range [9].

The topological neighbourhood of interaction was introduced following the challenges of metric neighbourhood framework. The results from Ref [9, 64, 105] showed that in some systems agents only interact with a fixed number of neighbours, e.g. six to seven of them in the particular case of starlings [9]. Therefore, for topological neighbourhood its agents only interact with specified number of neighbours, example, $k = 7$ would mean that each agent interacts with 7 nearest neighbours regardless of the distance separating them, shown in Figure 3.6 green triangles.

In addition, scientists have introduced many other neighbourhoods of interactions, e.g. hybrid metric-topological distance [109, 135], Voronoi based [61], fixed set (might not be the nearest neighbours) [144], bounding box [66] and angle based [66] (See Ref [56] for a comprehensive list and detailed explanation of respective interactions). Nevertheless, we used a topological neighbourhood of interaction for all our simulations as it facilitates our control of the exact number of neighbours that an individual agent interacts with. Mateo and co-authors have shown that the uncovered phenomenology is largely independent of the specific nature of the interaction, metric or topological [102].
3.4.2 Phase Transition

In a general sense, phase transition is a process, during which a system, consisting of a large number of interacting particles, undergoes a change (or transition) from one phase to another as a function of one or more external parameters [154]. One of the well-known examples of phase transition is water to steam or ice when temperature increases above the boiling point or drops below freezing point respectively. In this case, the temperature is the external or “control” parameter.

Phase transitions are defined by the change of one or more specific system variables, called “order” parameters. Order parameter comes from the observation that phase transitions usually involve an abrupt change in a *symmetry property* of the system. For example, in the solid state of matter, the atoms have a well-defined average position on the sites of an ordered crystal lattice, whereas positions in the liquid and gaseous phases are disordered and random. Accordingly, the order parameter refers to the degree of symmetry that characterise a phase—this value is usually zero in one phase (in the disordered phase) and non-zero in the other (in the ordered phase). First order phase transition occurs when the order parameter changes discontinuously at the critical state, e.g. water’s volume changes to ice when it freezes. In contrast, second order (or continuous) phase transition involves continuous changes from an ordered to a disordered phase by the change of one or more specific system variables (order parameters) [154].

In the case of a swarm with collective order, the order parameter is the alignment of the collective, the chosen order parameter (also known as polarization) is the average normalized velocity, $\varphi$ [153],

$$\varphi = \frac{1}{Nv_0} \left| \sum_{i=1}^{N} v_i \right|$$

(3.21)

where $v_i$ is the velocity of agent $i$, $N$ is the total number of agents, and $v_0$ is the constant speed of the unit in the system. This measure of the swarm order approaches unity if all agents move in almost the same direction and is exactly one if they are perfectly aligned. On the contrary, when agents fail to align themselves, the order parameter moves toward zero. Vicsek et al. uncovered the existence of a continuous second order phase transition with respect to two control parameters: the level of noise, $\eta$, and the density, $\rho = N/L^2$ [42,153].

Figure 3.7 shows similar results given in Ref [153] as the same parameters and conditions for $N = 40, 100, 4000$. In the absence of noise, $\langle \varphi \rangle$ is at 1 (fully aligned), and low noise region $\langle \varphi \rangle$ approaches unity. Conversely, $\langle \varphi \rangle$ approaches zero with increasing noise. In addition, Figure 3.7 shows that for very low noise (less than 0.1) and high noise (more than 0.9) regions, $\langle \varphi \rangle$ for metric interaction (represented by blue +) and topological interaction (represented by green circle) are the same for $N = 100$, (and green circle). This shows that in the low noise region, $\langle \varphi \rangle$ is largely independent of the nature of interaction, which is in line with the study by Mateo
et al. [102]. By referring to Figure 3.7, we can ensure that the system used for simulation is in the high level of order (with low level of noise) and far away from the criticality.

Figure 3.7: The average alignment, $\langle \phi \rangle$, of the SPP in the steady state with varying noise, $\eta$ with $\nu_0 = 0.03$. Where $N40R1D4$ means; $N = 40$, where $N$ is the number of agents; $R = 1$, where $R$ is the radius of metric interaction; and $D = 4$, where $D$ is the density of the system. There are three different numbers of agents, $N = 40, 100, 4000$, that used metric interaction and same parameters as Ref [153]. The average connectivity for these three Ns is $\langle k \rangle = 12$. The N100k10D1 ($N = 100, k = 10$ and $\rho = 1$) represented by the green circle uses the topological interaction with connectivity, $k$, equal to 10.

### 3.4.3 Correlations and Fluctuations

Correlation is the expression of an indirect information transfer mediated by the direct interaction between the individuals: two agents that are outside their range of direct interaction (be it visual, acoustic, touch or any other) may still be correlated if information is transferred from one to another through the intermediate interacting agents. The absence of direct interaction in the group implies the absence of correlation [5]. The correlation length—the spatial span of the correlation—can be significantly larger than the interaction range, depending on the level of noise in the system. Correlation measures how the behavioral changes of one agent influence those of other agents across the group [33].

In statistical physics, fluctuation is the deviation (of an individual agent) with respect to the mean behaviour of the system [34]. For collective behaviour, it is useful to measure how similar the fluctuations are in a system, and is called connected correlation function. The connected correlation function measures how similar the deviation of agent $i$ with respect to the mean behavior of the system is to the deviation of agent $j$. These fluctuations are the inherent collective response of the swarm to external random perturbations, i.e. the noise [26].
In our study using VM, we consider the fluctuations in the direction of travel and follow the framework introduced by Attanasi et al. [5]. The dimensionless velocity fluctuation is defined as

$$\delta \varphi_i = \frac{v_i - \langle v \rangle}{\sqrt{\sum_{k=1}^{N} \|v_k - \langle v \rangle\|^2 / N}}$$  \hspace{1cm} (3.22)$$

where $$\langle v \rangle = \sum_{i=1}^{N} v_i / N$$ is the average velocity.

The connected correlation function is then given by

$$C(r) = \frac{\sum_{i \neq j} \delta \varphi_i \cdot \delta \varphi_j \delta(r - r_{ij})}{\sum_{i \neq j} \delta(r - r_{ij})}$$  \hspace{1cm} (3.23)$$

where $$r_{ij} = \|r_i - r_j\|$$ is the distance between agents $$i$$ and $$j$$, and $$\delta(r - r_{ij})$$ is a smoothed Dirac distribution (see Figure 3.10).

The collective response of the swarm depends crucially on two factors: how distant in space the behavioural change of one agent affects that of another agent (spatial span of the correlation) and how strong this effect is (intensity of the correlation) [5]. To combine these two factors in one single observable one can calculate the cumulative correlation up to scale $$r$$. For finite-size systems, one can use the maximum of the cumulative correlation as an estimation of the total correlation in the system

$$\chi \equiv \max_{r_0} \left( \int_{r < r_0} C(r) \, dr \right)$$  \hspace{1cm} (3.24)$$

In statistical physics, $$\chi$$ is exactly equal to the susceptibility, namely the response of the system to an external perturbation [18,77]. While in collective animal behavior one cannot formally relate this integrated correlation to the response of the system, several studies [5,33] have shown a phenomenological relation between $$\chi$$ defined as Equation (3.24) and the way the group responds collectively to a perturbation.

### 3.5 Results and Discussion - Physics Approach

In order to characterise the effect that varying levels of connectivity have on the collective response of the system, we calculated the gains of the system for distributed linear leader-follower consensus protocol and computed the correlation function and susceptibility for VM (see Appendix C for MATLAB codes).
3.5. Results and Discussion - Physics Approach

3.5.1 Distributed Linear Consensus with Varying Levels of Connectivity

Significant attention has been dedicated to the problem of convergence to consensus [136] and controllability of multi-agent dynamics [88] in the presence of complex network topologies—possibly switching—with directed or undirected information flow [114]. Here, given the simple topology of the static network, both convergence to consensus and controllability are guaranteed. Our focus lies with the overall responsiveness of the collective in adapting to fast changes in the dynamics of the single leader.

We used a collective composed of \( N = 2048 \) agents with varying levels of connectivity, \( k \), and input oscillation frequencies, \( \omega \), spanning four orders of magnitude. Figure 3.8a shows the total amplitude gain (normalized by the number of agents) as a function of the number of connections, \( k \). The total amplitude gain can be used to interpret the extent to which the agents are able to follow the perturbation induced by the leader at variable frequencies. Note that the natural response frequency is one, \( \omega_0 = 1 \), when it is not stated. At low frequency, \( \omega \lesssim 10^{-3} \), the system responsiveness to perturbations increases with the number of connections. Conversely, at high frequency, \( \omega > 0.1 \), the addition of more connections to the system results in a reduction of the performance of the system. We have observed that there is an interesting intermediate frequency, where \( \omega \sim 0.01 \) (see Figure 3.8b). The responsiveness of the system

Figure 3.8: Responsiveness of a distributed leader-follower consensus protocol with \( N = 2048 \) agents following a single leader. (a) Total amplitude gain (normalized by the number of agents) as a function of connectivity, \( k \), at different oscillating frequencies, \( \omega \), where the leader input into the system. The solid colour lines correspond to a frequency, \( \omega \), equal to the value specified in the legend. The dotted lines correspond to the frequencies evenly distributed between the solid lines. (b) At frequency, \( \omega = 0.01 \), the gain (non-normalized) starts to suffer beyond a certain threshold of connectivity, approximately \( k = 20 \), showing a detrimental effect related to connectivity.
reaches a peak at a finite number of connections, approximately $k = 20$, thereafter it reduces with more connectivity. This intermediate frequency regime shows that we can double the amount of agents following the leader by setting the connectivity to its optimal value.

Next we studied the normalized gain as a function of frequency at different numbers of connections. We observed that the whole spectrum of frequencies (that our calculations covered) can be divided into four regions (see Figure 3.9). For the region where $\omega \lesssim 0.003$, the maximum number of connections gives the best performance of the system—when responding to perturbations. When the system is operating at $0.003 < \omega \lesssim 0.018$ region, $k = 20$ yield a better performance as compared to an all-to-all connectivity, as observed previously. In the region $0.018 < \omega \lesssim 0.1$, $k = 10$ outperformed the rest of the connectivities. Lastly, if a system operates in the region $\omega > 0.1$, the result showed that the basic ring topology, $k = 2$, has the best responsiveness. This result may appear counter-intuitive.

![Figure 3.9: Normalized gain as a function of frequency at different number of connection, k. The vertical dotted lines split the spectrum of frequencies into four regions where the first region (low to high frequency) shows that $k = N$ is the most favourable choice for the system, follow by $k = 20$, $k = 10$, and $k = 2$ for the rest of the regions respectively.](image)

From the standpoint of designing artificial swarm, this analysis highlights that the appropriate connectivity for a system depends on the pace of the perturbation the system is subjected to. When subjected to slow-changing perturbations, the system’s effectiveness always benefits from a higher level of connectivity. On the other hand, fast perturbations inevitably reduce the system’s effectiveness with increasing interagent connectivity. It is essential for the performance of systems conducting distributed consensus to be able to react efficiently to perturbations in the appropriate time scale. For example, ants performing collective transport of food rely on transiently informed peers to locate their nest [62]. These informed “leaders” forget their knowledge after a time of joining the collective action, and thus provide a changing signal with a certain characteristic time scale to the swarm. Successful transport depends both on a high
3.5. Results and Discussion - Physics Approach

This analytical study provides guidelines for the design of the interconnecting network of an artificial swarm using the distributed linear leader-follower consensus protocol subjected to external perturbations (in term of oscillating frequencies). Figure 3.9 can be used as a guideline to determine the optimal number of connections that is most suitable for a swarm design at various frequencies. For instance, we can tune the interaction network of our fleet of buoys to better respond to the changes in the dynamic environment by varying the number of connections (based on the frequency of the change).

3.5.2 Vicsek Model with Varying Levels of Connectivity

A natural starting point to characterize the responsiveness of the collective motion is to study the connected correlation in fluctuations of the velocity [108]. We computed the correlation function, \( C(r) \), and susceptibility, \( \chi \), for a swarm composed of \( N = 2048 \) SPPs while varying the number of neighbours, \( k \) (with the following parameters setting: \( v_0 = 0.04, \eta = 0.05 \) and \( \rho = 1 \)). In order to investigate the behaviour of swarms displaying a high degree of alignment, we perform the calculations in the low noise regime. From Figure 3.7, we can be certain that our system at \( \eta = 0.05 \) is in the high order state. With \( N = 2048 \), this ensures that the system is large enough for statistical analyses and represents a large swarm, as most collective motions range from dozens to a few thousands and rarely more [154]. In addition, this size allows us to study the system at a wider range of connectivity while still being within a connected network (see section 3.2.2).

In order to determine the values of the correlation function \( C(r) \) and susceptibility, \( \chi \), we computed the histogram of the correlations in the system every 1000 iterations throughout the \( 8 \times 10^5 \) iterations, after discarding the first \( 1 \times 10^5 \) iterations as transient dynamics. This was followed by numerically integrating the correlation \( C(r) \) until \( r = r_0 \) to obtain \( \chi \) (see Figure 3.10). The integrated correlation \( \chi \) (Equation (3.24)) is a measure of the total amount of correlation present in the system [5]. In the collective behaviour literature, this quantity is commonly referred to as susceptibility in reference to its analogue in equilibrium statistical physics, where the dissipation-fluctuation theorem establishes that the response of the system is proportional to this quantity. We find that a collective of SPPs following the Vicsek Model in the ordered phase can exhibit a large integrated correlation if the number of neighbours, \( k \), is set to an appropriate level (see inset of Figure 3.10).

The integrated correlation depends both on the span of correlations (how far in space the behavior of one agent influences another) and the intensity of correlations (how strong this influence is) [5]. There is an intrinsic trade-off between the span of correlation and the intensity of correlation; an increase in the number of connections allows information to travel farther through the swarm—increased correlation length—but subjects each agent to more
Correlation

\[ k = 4 \]

\[ k = 22 \]

\[ k = 34 \]

**Figure 3.10:** Correlation in velocity fluctuations for VM, \( N = 2048 \) with topological interactions and number of neighbours \( k = 4, 22 \) and 34. The distance is measured in units of the computation box size \( L \).

information. This additional information decreases the relevance of each signal thus decreasing the correlation strength. **Figure 3.10** shows the correlation function for three different values of the number of neighbours, \( k \), illustrating this trade-off. For small values of \( k \) (e.g. \( k = 4 \)), correlations are large but confined to short distances. As \( k \) increases, so does the spread of correlations and thus \( \chi \).

This analysis shows that there is a certain optimal number of connections, \( k^* \approx 20 \) for \( N = 2048 \), for our configuration of VM. As the increase in spatial spread is unable to compensate for the reduction in correlation strength, it therefore led to a decrease in \( \chi \) with a further increase in connectivity. Agents are better at escaping a predator’s attack when the collective has a higher \( \chi \) [102]. At low number of connections (e.g. below \( k = 20 \)), the increase in correlation length effectively allows the information to propagate faster through the interaction network, thus more agents are capable of responding to the presence of the threat. On the contrary, at higher numbers (e.g. beyond \( k = 20 \)) of connections, this increase in the correlation length is accompanied by a drastic reduction in correlation strength, thus severely reducing the responsiveness of agents in the vicinity of the threat.

We observed that for \( N = 100 \) and \( N = 512 \), both systems also exhibit a peak at certain connectivity. All three sets of simulations (\( N = 100, N = 512, N = 2048 \)) have the same conditions, except for the size of the swarm (see **Figure 3.11a**). This shows that VM exhibits a large susceptibility if the number of neighbours, \( k \), is set to an appropriate level. **Figure 3.11b** shows the best fitted curve for estimating the optimal number of neighbours, \( k_{\text{peak}} \), based on
3.6 Consensus with Forced Switching

\[ N = 100, 256, 512, 1024, 2048, \]

\[ k_{\text{peak}} = 10.48 \pm 0.9 \times \log(N) - 14.56 \pm 2.4 \quad (3.25) \]

Figure 3.11: (a) The susceptibility, normalized by respective \( N \), as a function of the number of neighbours, \( k \), at \( N = 100, 512, 2048 \). All three systems show that there is a peak at a finite number of connections. (b) Curve fitting for the peaks, \( k_{\text{peak}} \), corresponding to connectivity at different number of agents, \( k_{\text{peak}} = 10.48 \times \log(N) - 14.56 \).

This study provides an insight for the design of the interaction network for an artificial swarm. We can operate the fleet of buoys with much lesser number of connections than the number of agents. By using Equation (3.25) we can estimate the \( k_{\text{peak}} \) of the swarm and tune our fleet of buoys to achieve a more effective dynamical response. In addition, this study also highlights that an excess number of connections will lead to a less responsive system, and at the same time incur extra costs for establishing connections and transmitting information between buoys.

3.6 Consensus with Forced Switching

The study from the previous sections (section 3.3, section 3.4 and section 3.5) using models with agents interacting at the local level, similar to natural swarms [4, 9]. Here, we study the effect of having access to global information (through forced switching techniques) in swarms on collective response, as it has yet to be discovered. The possibility of implementing some level of forced switching on artificial robotic swarm allows us to study the effect of forced switching on collective response and decision-making. This study uses the same two models by subjecting them to forced switching.
3.6.1 Distributed Linear Consensus with Watts and Strogatz Network

The Watts and Strogatz Network modeling approach of random rewiring (forced switching) is used for distributed linear leader-follower consensus protocol. The analysis starts with a ring lattice with a fixed number of connections, \( k \), it goes through Watts and Strogatz model algorithm (see Listing 3.1) for forced switching (probability greater than zero) before calculating the gain of the system (see section 3.3).

We ran a series of calculations varying the number of connections, \( k \), with varying the probability of random rewiring (forced switching), \( p \), for the leader-follower linear consensus protocol. We kept the same collective of size \( N = 2048 \) agents and with one single leader driven at varying frequency, \( \omega \), spanning four orders of magnitude (see Appendix D for MATLAB codes).

3.6.2 Vicsek Model with Local Extended and Global Interaction

There is no ready network model (for forced switching) like the Watts and Strogatz model for VM. Therefore, we developed our own forced switching method (for local extended and global switching). In general, VM agents connect with their specified (in the topological sense) number of nearest neighbours, \( k \), at the beginning and seldom switch neighbours once the system is stable, especially at low noise. We introduced a forced switching term, \( k_s \), into the system, where \( k_s \) is the number of neighbouring connections that are to be switched. The \( k_s \) varies between zero to the number of connections, \( k \) (max \( k_s = k \)). When \( k_s = 0 \), it means that the VM is in its fundamental state (very small amount of switching or no switching at low noise) of interagent interaction. The number of remaining (unchanged) connections, \( k_p \), is equal to number of connections, \( k \), minus the number of forced switching; \( k_p = k - k_s \). This method only has the memory of the previous iteration (network connections), therefore it does not take up a large amount of memory and yet has an effect on the system. By using this technique, we are able to rewire the links between agents of the interaction network (see Appendix D for MATLAB codes).

For local extended forced switching (or local extended interaction), the agents forced switch locally by linking to neighbouring agents (from the set of next nearest neighbours). For example, if \( k = 10 \) and \( k_s = 5 \), the agents will link with their next 5 nearest neighbours (agent 11, 12, 13, 14 and 15) but within the maximum reach of \( k + k_s \) (see Figure 3.12a). Listing 3.2 shows the algorithm for local extended forced switching method.

Listing 3.2: Algorithm for local extended forced switching

```matlab
function [tind] = FS_nn(kn,ks,iter,tind,prevadjmat)
% This function force the node/agent to switch the number neighbour in each
% iteration specified by ks (degree switch), using topological method.
```
For the global random forced switching (or global interaction), the agents forced switch by randomly linking to any agent including very distant ones, e.g if $k_s = 3$, the agent will randomly link with any 3 agents in the system (that are not previously connected to it) (see Figure 3.12b). Listing 3.3 shows the algorithm for global forced switching method.

Listing 3.3: Algorithm for global forced switching

```matlab
function [tind] = FS_rand(kn, ks, iter, tind, prevadjmat)
% This function force the node/agent to switch the number neighbour in each
% iteration specified by ks (degree switch), using topological method.
% kn = out-degree,
% ks = no. force switching,
% iter = iteration,
% tind = new set of neighbour, and
% prevadjmat = the previous adjacency matrix.

lnn = tind(2:kn+1);
pnn = find(squeeze((prevadjmat(iter,:))));
fdiff = setdiff(lnn,pnn);
rmolddiff = setdiff(pnn,lnn);
tind_temp = tind(kn+2:end);
for irmo = 1:length(rmolddiff)
tind_temp(tind_temp==rmolddiff(irmo)) = [];
end
lfdiff = length(fdiff);
if lfdiff < ks
    lnn = setdiff(lnn,fdiff);
    switchpos = randperm(length(lnn),ks-length(fdiff));
    for iswitch = 1:length(switchpos)
        tind(tind==lnn(switchpos(iswitch))) = tind_temp(iswitch);
    end
end
end
```
Figure 3.12: Forced switching method for the VM model. (a) Local extended forced switching, where $k = 10$ (green $\Delta$ inside grey dotted circle) at previous iteration (left). Present iteration (right) $k_s = 5$, the next 5 nearest neighbours are chosen from inside the black dotted circle (which is the maximum distance, $k + k_s$), $k = k_p + k_s = 10$ (green $\Delta$). (b) Global (random) forced switching, starts from $k = 10$ (left, green $\Delta$ inside grey dotted circle) and moves to the next iteration (right) when $k_s = 3$. These 3 agents are randomly chosen from the domain except the agents from previous set of nearest neighbours, $k = k_p + k_s = 10$ (green $\Delta$).

3.7 Results and Discussion - Forced Switching

The network-theoretic approach is used to study the effect of forced switching on the collective response for both models. For distributed linear consensus, other than studying the gain, we also analysed the effect of forced switching on speed to consensus (algebraic connectivity) and other network properties: average path length and clustering coefficient. We used the metrics of susceptibility and algebraic connectivity to study of the effect of forced switching on VM.

3.7.1 Distributed Linear Consensus with Forced Switching

To characterise the effect of the varying probability of rewiring (or forced switching as known in the dissertation), $p$, on the collective response, we studied the normalized gain as a function
of \( p \) (see Figure 3.13). Additionally, we studied the effect of random rewiring on the speed to consensus for the distributed linear leader-follower consensus protocol (see Figure 3.15).

### Effect of Forced Switching on Normalized Gain

In the effect of forced switching on the collective response (gain), we observed that at low frequency, \( \omega = 0.0001 \) (left figure on Figure 3.13), the responsiveness of the system increases as \( p \) increases, for lower values of the connectivity, \( k \). For high values of \( k \), it is expected that an increase in the random rewiring will not have much effect on the system, as the system is near an all-to-all connectivity (span of reach is wide enough to reach many far-away agents). This result shows that some level of random rewiring is beneficial to the collective response of the system at low frequency. It shows that we can improve the responsiveness of the system by having some level of randomness in a connected network. This is, however, only valid for \( \omega \lesssim 0.001 \).

For a frequency in the range of \( 0.001 < \omega < 0.1 \) (centre figure of Figure 3.13), there is no obvious observation whether rewiring will benefit the system at \( p < 0.01 \) (dotted line indicates \( p = 0.01 \)). However, it is clear that a rewiring has detrimental effects on gain for \( p > 0.01 \). We termed this frequency range a transitional (changing-over) range, for which the benefit of rewiring starts to have a negative effect on the system (\( p > 0.01 \)) and higher values of connectivity (\( k = 80 \)) starts to respond in poorer manner compared to lower values (\( k = 10 \)). Nevertheless, systems that operate in this frequency range can still benefit from some level of rewiring up to a certain value, \( p \lesssim 0.001 \).

In the high frequency regime, \( \omega \gtrsim 0.1 \), a further increase in random rewiring beyond \( p = 0.01 \) has a detrimental effect on the responsiveness of the system. In fact, it showed that there is no advantage in rewiring agents of the system in this high frequency regime (right figure of Figure 3.13).

The results will be discussed via a different perspective where the normalized gain (responsiveness) is a function of connectivity (see Figure 3.14a). It is clear that at low frequency (red lines, \( \omega \lesssim 10^{-4} \)), increasing random rewiring in a network improves its responsiveness when subjected to external perturbations. When \( p = 1 \) (represented by \( \square \)), the responsiveness of the system improves by one order of magnitude as compared to the ring lattice, \( p = 0 \) (represented by \( \bigcirc \)). However, this effect tapers off as the number of connections increases to a certain \( k \) value, \( k = 128 \). As the frequency increases, we see that the ring lattice performs better than a fully random network. At frequency, \( \omega \approx 0.04 \), we noticed that \( p = 1 \) decreased quickly to its worst responsiveness (lowest gain) with just a few increments.

For systems that follow the distributed linear leader-follower consensus protocol similar to ours, we observed that the effect of having more connectivity on the system switches in a particular condition (value of \( p \) and \( \omega \)). In this condition, \( p = 0.029 \) and \( \omega \approx 0.008 \), the system
Figure 3.13: Normalized gain as a function of probability, $p$, using the Watts and Strogatz network model for leader-follower linear consensus protocol ($N = 2048$). Left: effect of rewiring (randomly) on the system at low frequency, $\omega = 0.0001$. It has a positive effect on the system, gain increases as the probability increases for all connectivity. Middle: system at mid-range frequency, $\omega = 0.001$, there is no distinct observation whether a rewiring benefits the system or not prior to $p = 0.01$ (dotted line). However, it is clear that rewiring has a detrimental effect on the gain beyond $p = 0.01$. Right: at the frequency, $\omega = 0.1$, random switching does not have any benefit to the system. After $p = 0.01$ (dotted line), random switching leads to a reduction of the responsiveness of the system, i.e. a detrimental effect.

Figure 3.14: Responsiveness of leader-follower consensus protocol with respect to connectivity and frequency with forced switching. (a) The total amplitude gain (normalized) for a system of $N = 2048$ as a function of connectivity at different oscillating frequencies, $\omega$, with varying probability of rewiring, $p$. Each color represents one frequency with the value specified on top of the line. Square markers indicate $p = 1$, circle markers represent $p = 0$, and dotted lines correspond to the probability of rewiring that evenly spread out between $p = 0$ and $p = 1$. (b) Responsiveness as a function of frequency, at $p = 0.029$ where switching occurs at $\omega = 0.008$. At this condition, the system performs equally well regardless of connectivity, beyond that the system benefits from low connectivity.
performs equally well regardless of the connectivity. Beyond this frequency, the positive effect (higher in gain) of having a greater number of connections switches to a negative effect (lower in gain) therefore, we call this the cross-over condition. Other than this condition, the effect of connectivity is split into several regions, shown in Figure 3.9. For artificial swarm design, this implies that if a system is subjected to external perturbation at low frequency, it is beneficial to increase the randomness of the interaction network of the system.

**Effect of Forced Switching on Algebraic Connectivity**

We used the network property of second smallest eigenvalue of the Laplacian matrix (see section 3.2.4) to characterise the effect of forced switching on the speed to consensus for the distributed linear leader-follower consensus protocol (see Figure 3.15). This network property is not a function of frequency and is not affected by the frequency (same for average clustering coefficient and average path length).

Olfati-Saber et al. showed that $\lambda_2$ increases as the probability of rewiring increases for all values of the connectivity, which yields the so-called ultra-fast speed to consensus [113]. Shang and Bouffanais also showed a similar observation [135]. Our analytical result shows that some level of random rewiring makes the system “smaller” (small-world effect) and thus information is transferred faster throughout the system (see Figure 3.15), which is in line with Ref [113,135]. It seems that this is the result of a reduced average path length, which has been widely studied

![Figure 3.15: The influence of rewiring in terms of the second smallest eigenvalue, $\lambda_2$, of the graph Laplacian of the network. It shows that more random rewiring increases the speed to consensus for all connectivity.](image)

instead of average clustering coefficient [74,143,160]. However, Figure 3.16 shows that $\lambda_2$ is highly affected by the average clustering coefficient—more than the average path length for all values of connectivity (except $k$ that is very near to number of agents, $N$). This implies that a smaller clustering in the network with short path length leads to ultra-fast consensus.
Therefore, depending on the region of frequencies the systems are subjected to, systems can be designed to have high speed to consensus and high responsiveness (in low frequencies region, where changes in the dynamical environments take more than 2.5 hours).

![Network Characteristics](image)

Figure 3.16: Characteristics of the network shown by its properties at different values of connectivity, $k = 4, 20, 40$: average clustering coefficient (red □), CC, average path length (blue ○), PL, and second eigenvalue of the Laplacian matrix of the network (green △), $\lambda_2$. PL and CC are normalized with the values of ring lattice where $p = 0$, PL(0) and C(0) respectively. $\lambda_2$ is normalized with the maximum $\lambda_2$ as it increases with the probability of rewiring, $p$.

### 3.7.2 Vicsek Model with Forced Switching Network

We have seen in section 3.5.2 how the number of connections affects the susceptibility (responsiveness), $\chi$, of the system. Here, we further the study by looking at the effects of global interaction and local extended interaction on the collective response. We ran a series of simulations with the same method and conditions as in section 3.5.2: $N = 2048$, $v_0 = 0.04$, $\eta = 0.05$ and $\rho = 1$, but now with a forced switching network.

#### Global Interaction

The global random forced switching has a negative effect on the collective system; any amount of forced switching (even 1 global connection) reduces the responsiveness, $\chi$, of the system (see Figure 3.17a). This effect occurs for any number of connections. This implies that as the span of correlation increases, information travels farther, therefore subjecting agents to more irrelevant information (section 3.5.2).

The directly proportional relationship of speed to consensus and number of connection for the SPP model has been shown by Shang and Bouffanais [135,136]: an increase in connectivity leads to higher speed to consensus. We observed this relationship in our global random forced switching case (see Figure 3.17b). Our result shows that an increase in global random forced
3.7. Results and Discussion - Forced Switching

Figure 3.17: Effect of global random forced switching on collective response. (a) Susceptibility, $\chi$, as a function of forced switch, $k_s$ normalized by number of connection, $k$. (b) The network property of the second smallest eigenvalue of the Laplacian matrix was used to determine the speed to consensus, higher value signifies higher speed to consensus.

switching leads to an increase in speed to consensus for all values of $k$, with $k_s = k$ yielding the maximum $\lambda_2$. When analysing Figure 3.17a side by side with Figure 3.17b, we uncovered the existence of a trade-off between speed to consensus and the responsiveness of the system with global interaction. Therefore, global interaction is good for systems seeking to increase speed to consensus (collective motion) at the expense of responsiveness. We can apply global interaction on our fleet of buoys network interaction when speed to consensus is the top priority for tasks such as flocking and leader-follower.

Local Extended Interaction

A certain amount of local extended forced switching improves the responsiveness (increase in susceptibility) of the system (Figure 3.18a). When $k = 10$ (low connectivity), $\chi$ is directly proportional to the combination of the number of connections and number of local extended forced switching, $k + k_s$. Conversely, at high connectivity $k = 35$, $\chi$ is inversely proportional to $k + k_s$. As for $k = 13$ and $k = 20$ the system exhibits a peak at certain $k + k_s$. This result forms a curve that is similar to Figure 3.11a for $N = 2048$ with varying number of connections. This implies that the susceptibility, $\chi$, of the collective can be increased (towards the peak) without increasing the number of connections for $k + k_s \leq k_{peak}$ (where $k_{peak}$ is the optimal number of connections that yields the peak in $\chi$).

This study provides an important insight for the design of the interaction network of our fleet of buoys, as we can tune the system to perform at the highest level of responsiveness without increasing the number of connections. By using the local extended forced switching
too many cooks spoil the broth. In practice, a communication channel is used to establish this interaction network, such that bandwidth and communications are limited.

Figure 3.18: (a) Responsiveness of the system with varying connectivity, $k$, form up by combining number of connection and number of forced switching, $k_s$, $k + k_s$. The trend is the same as Figure 3.11a for $N = 2048$. (b) Effects of the local extended forced switching (with $k + k_s$) on the speed to consensus of a collective.

Local extended forced switching also shows similar trade-off between speed to consensus and responsiveness as global forced switching for higher $k$ values, $k \gtrsim 22$ (see Figure 3.18b). However, for $k = 10$, $\chi$ and $\lambda_2$ increase with the increase of $k + k_s$. This similar trend has been observed for $k = 15$, but only for $k + k_s \leq k_{\text{peaks}}$. This result implies that a system can operate at the high level of susceptibility and improve speed to consensus with $k + k_s \leq k_{\text{peaks}}$ for local extended forced switching.

3.8 Summary

Through the study of two bio-inspired models, we have provided evidence suggesting that it is beneficial for an artificial swarm to limit the amount of interaction in a system when the responsiveness of the collective is the main priority (e.g. when our fleet of buoys is performing dynamic area coverage). The analysis of frequency-response in a distributed linear leader-follower consensus protocol and VM reveals a decrease in responsiveness associated with an excess of connectivity. The optimal connectivity depends on how fast the environment changes. Since these models are relatively general and simple, it suggests that this observation may be a general feature for a wide range of swarm system involving distributed consensus.
This study not only sheds new light on our understanding of collective behaviour, it also has clear implications for the design of the interaction network of our buoys. It may be more desirable to limit the number of neighbours in order to achieve better performance of our buoys in terms of dynamical response and depending on the pace or dynamics of the specific environmental feature being tracked. The agents of these models mostly interact locally even with the increase of connectivity.

Next we investigated the effect of forced switching on the responsiveness of the system. The distributed linear leader-follower consensus protocol with random switching provides a number of important insights about the collective response. At low frequency, $\omega \lesssim 0.001$, the system benefits from random forced switching; the responsiveness of the system increases with more random rewiring (best result with total randomness, $p = 1$). Switching from positive effect (increase in gain) with random rewiring to negative effect (decrease in gain) happens at $0.001 < \omega < 0.1$, the transitional range. There is a condition for which the switch happens immediately at $p = 0.029$ and $\omega \approx 0.008$. Systems that operate in this frequency range can still benefit from some level of rewiring up to a certain value, $p \lesssim 0.001$. At high frequency, the network topology has no effect on the system when $p < 0.01$, beyond this point ($p > 0.01$) it leads to a detrimental effect in the responsiveness of the system. The speed to consensus increases with more randomness in the network for all $k$ values except values near the all-to-all connectivity. Therefore, these insights provide guidelines for designing the collective system.

The global random forced switching has a negative effect on the collective system, any amount of forced switching leads to a reduction in the responsiveness. However, increases in global forced switching increase the speed to consensus. This implies that there exists a trade-off between speed to consensus and responsiveness of the system when subjected to global interaction. For the local extended forced switching there are slight variations, however, at high $k$ values ($k \gtrsim 22$ for $N = 2048$) the system still exhibits the same trade-off trend as global interaction. For lower $k$ values ($k + k_s \leq k_{\text{peak}}$), the responsiveness of the system and speed to consensus increase with the increase of $k + k_s$. This interesting observation enables one to tune the system to operate at the maximum level of responsiveness while using the same number of connections.

These two studies provide a number of important insights for the design of interaction network for our fleet of buoys. First, we can operate the fleet of buoys with much lower number of connections (optimal connectivity) than the number of agents. This optimal connectivity leads to better performance of the fleet of buoys when tracking dynamic environmental features. By using Equation (3.25) we can estimate the $k_{\text{peak}}$ of the swarm (optimal connectivity) and tune our fleet of buoys to achieve a more effective dynamical response, and excess connectivity leads to reduced responsiveness of the system while incurring extra costs (for establishing connections and transmitting information between buoys). In practice, a communication channel is used to establish this interaction network, such that bandwidth and communications are limited.
Finally, the global forced switching technique allows us to achieve a fast rate of convergence for the fleet of buoys (when responsiveness of the system not important). By using the local extended forced switching technique, we can tune the fleet of buoys to operate at the highest level of responsiveness without increasing the number of connections; this therefore, reduces the bandwidth requirement in the communication network as the number of buoys increases. Next, we look at the implementation of cooperative control algorithms and basic behavioural rules on our buoys system followed by running a series of field experiments.
Self-organization is a process whereby some form of overall order arises from local interactions between agents of an initially disordered system without the need for external influences [32]. We have developed a set of algorithms, with the insights and understanding from nature and previous chapters, to control the buoys in performing certain self-organising behaviours. Here, the theory and algorithms are put to test just like where the rubber meets the road.

We demonstrate the self-organization of our buoys by implementing the following five basic behavioural rules: avoidance, alignment, aggregation, target navigation and geofencing. With the combination of these rules, we can program the buoys to display a range of collective behaviours like flocking, collective navigation and dynamic area coverage. These behaviours are validated through field experiments at two waterbodies (both approximately 0.32 km\(^2\)) in Singapore, we labelled these two experiment sites as Reservoir and Quarry. In addition, we applied this collective buoys for real world applications of water monitoring, by measuring temperature, DO, and pH of the waterbodies.

4.1 Buoys’ Data-Flow

For ease of identifying and addressing the key aspects for the production of flocking motion and how these aspects are related (for our fleet of buoys), we adopted the data-flow template (DT) introduced by Fine and Shell [56]. This five-stage DT template (see Figure 4.1) aids in designing and presenting complete microscopic flocking motion model for our fleet of buoys. Each of the five stages—sensing, flock member detection, neighbour selection, motion computation, and physical motion—of the DT represents the key aspects for the generation of flocking motions. The five stages of the DT are connected by the information that is passed between them [56]:

- **Sensing**: This stage translates the environment from the individual sensor’s (e.g. GPS, laser range-finder, temperature sensor) reference frame into usable input for the later stages.

- **Flock member detection**: This stage uses the raw sensor information provided by the sensing stage and outputs the set of all detected flock members.
Figure 4.1: A diagrammatic representation of the date-flow template for microscopic flocking motion model. It details the main aspects for the generation of flocking motions via the five stages (boxes). The connections between the stages encode the data that propagates between them. In particular, the connections between the sensing stage and the flock member detection stage represents the raw sensor information from each sensor (e.g. GPS, camera, laser range-finder). The connection between the flock member detection and neighbour selection stage is the set of detected flock member. The neighbour selection stage passes at least one set of selected flock members to the motion computation stage which passes the next computed motion to the physical motion stage.

- **Neighbour selection**: This stage takes the set provided by the flock member detection stage and outputs at least one subset of the set (e.g. k-nearest neighbours).

- **Motion computation**: This stage uses the set(s) generated by the neighbour selection stage to calculate the next motion of the flock member.

- **Physical motion**: This stage takes the computed motion from the motion computation stage and translates it into a form that can be realized in either a simulated or physical robot.

In our work, GPS and temperature (including DO and pH for some buoys) data are sensed by all the buoys at the sensing stage. This data together with individual buoy identification are broadcasted to other buoys through the communication network. With this data, each buoy is able to detect its members (flock member detection stage), in this case, the members are those buoys that are within the XBee® mesh network (see section 2.2). Since the buoys sense other buoys exclusively via radio-communication, the neighbourhood of a given buoy will always be a subset of the buoys present in its communication range. We have not imposed any selection rule for neighbours (neighbour selection stage), therefore the neighbours of a given buoy are by default all the buoys detected by it. For the motion computation stage, each buoy uses its sensed data and the information of its neighbours to calculate the next motion (see section 4.2). The output data of the motion computation stage is passed to the physical motion stage, which in turn uses to control the motors’ speed and direction (see section 2.1.4).

### 4.2 Cooperative Control Algorithms

Ideally, one would like to operate the collective buoys at the system level by issuing global objectives such as aggregation, collective navigation, dynamic area coverage, mapping, sensing
4.2. Cooperative Control Algorithms

target area, collective sensing, distributed search and rescue, etc. These global collective objectives have to be mapped into individual agent-specific commands, and such a process is known as cooperative control algorithm. The form of this algorithm will determine the effectiveness of the large-scale collective behaviour of the system [40].

In order to ensure certain degree of scalability and robustness we imposed spatial locality on the algorithm, meaning that the action of an agent is solely determined by the information gathered on a certain neighbourhood of its location. Likewise, flexibility to changing environment benefits from imposing temporal locality, meaning that the action is purely determined by the current state of the neighbourhood. Therefore, new information has to be updated to the agents (at every time step). We implement these conditions by considering the iterative algorithms (or update rules) that control the trajectory of an individual agent by updating its target velocity according to

\[
v_i(t + \Delta T) = \tilde{F}(t, \vec{r}_i(t), \{\vec{r}_j(t)\}_{j \sim i}, \{\vec{v}_j(t)\}_{j \sim i})
\]  

(4.1)

where \(i\) is the agent index, \(\Delta T\) is the sampling time at which this rule is applied, \(\vec{v}_i\) the velocity of agent, \(\vec{r}_i\) its position, and \(j \sim i\) is the set of agents in the neighbourhood of \(i\)—its “neighbours”, excluding \(i\). For our fleet of buoys, we do not impose any rule on how the neighbourhood is constructed (refer to Ref [56] for examples of how to construct neighbourhood). We let it “evolve” through radio-communication since our buoys communicate through Xbee® mesh network. Therefore, the neighbours of a given buoy are those within its communication range. In other words, all the buoys that it can detect. Since the set of sensed agents depends on their motion, the set of neighbours will change in time with dynamics coupled to the motion of the agents [26,88].

Next we will discuss the basic behavioural rules and three collective behaviours—flocking, navigation, and area coverage—implemented in our buoys and tested in field experiments. These behaviours are constructed as a superposition of non-dimensional basic behavioural rules—repulsion, alignment, aggregation, target navigation, and geofencing—used as building blocks, so that

\[
v_i^0 = v_0 \sum_{\vec{g}_i \in G_i} \vec{g}_i
\]  

(4.2)

where \(v_0\) is a constant speed of the buoy, and \(G_i\) is the set of elementary dynamical behavioural rules considered for agent \(i\). For conciseness we omitted the explicit time-dependence term, that is \(\vec{v}_i = \vec{v}_i(t)\) and \(\vec{v}_i^0 = \vec{v}_i(t + \Delta T)\). We would like to highlight that these behavioural rules can be agent-specific, in that not all agents follow the same rules at any given time. However, we are able to collectively deploy a large fleet of ASVs on aquatic surface using these behaviours. These behaviours are also used to control buoy motion and position them optimally to collectively monitor arbitrary shapes (see Appendix B for buoy operating system algorithms). Additionally, these basic behavioural rules and collective behaviours are not new (ref to Ref [56] and [154]...
for a comprehensive review of flocking motion models and collective motion, respectively) but the formulas are novel.

4.2.1 Basic Behavioural Rules

In order for the individual agent—in this case the buoy—to swarm together and move in a collective manner, a set of decentralized control algorithms is required at the agent level. This set of decentralized control algorithms consists of five basic behavioural rules: avoidance, alignment, aggregation, target navigation, and geofencing [164]. This is not an exhaustive set of rules, additional rules can be added depending on the desired tasks (ref to Ref [14] for a comprehensive review of tasks). This section presents each rule and discusses their implementation in detail. These rules require the GPS position data to be sent by other buoys through Xbee® communication network. The update rate of the default positions for each buoy is set at a frequency of 0.1 Hz (this value can be adjusted to suit the desired settings of the experiment).

4.2.1.1 Repulsion

We model the agent-to-agent avoidance (or repulsion) based on a two-body repulsion force model as follows:

\[
g^R_i = -\sum_{j \neq i} \frac{\alpha^d_R \tilde{r}_{ij}}{r^d_{ij} r_{ij}}
\]

where \( \tilde{r}_{ij} = \tilde{r}_j - \tilde{r}_i \) and \( r_{ij} = ||\tilde{r}_{ij}|| \). The position, \( \tilde{r}_i \), of agent \( i \) and \( \tilde{r}_j \) is the position of its neighbours. The repulsion strength is determined by \( \alpha_R \) and \( d > 1 \) which controls the repulsion decreasing with distance. The value \( d = 2 \) is used for flocking and navigation behaviours, and \( d = 3 \) is used for dynamic area coverage or monitoring.

The repulsion strength can also be used to control the distance between agent and its neighbouring agents. When \( \alpha_R = r_0/\sqrt{3} \) the agents are separated by a distance of \( \approx r_0 \). However, for the dynamic area coverage we would like the agents to spread as much as possible in the area. In such a scenario, we scale the repulsion strength with the total area to explore, \( S \), per agent, i.e. \( r^2_0 \propto S/N \). we set \( r_0 = 1.3R_0/\sqrt{n_i} \), where \( n_i \) is the number of neighbours of each agent.

4.2.1.2 Alignment

In swarming or collective behaviour, agents typically move together with the same orientation and align to each other. This provides a “smoother” collision avoidance behaviour and allows the swarm to move while the agents stay in formation. The following velocity alignment is
implemented:
\[
\vec{g}_i^{AL} = \frac{1}{n_i} \sum_{j \sim i} \vec{v}_j \tag{4.4}
\]
where \(n_i\) is the number of nearest neighbours. If the behaviour of the agents is dominated by Equation (4.4), the swarm will perform heading consensus and the agents will align themselves in a common direction of travel.

### 4.2.1.3 Aggregation

This dynamical behavioural rule leads all the agents to collectively undergo an aggregation or clustering process, in a collective manner. It is implemented as
\[
\vec{g}_i^{AG} = \frac{1}{n_i} \sum_{j \sim i} \vec{r}_{ij} \tag{4.5}
\]
Setting \(G_i = \{\vec{g}_i^R, \vec{g}_i^{AG}\}\) (Equation (4.5) and Equation (4.3)) in Equation (4.2) will ensure that the collective perform a collision-free rendezvous in space at an unspecified location. However, for our fleet of buoys it is as true as the accuracy of GPS data (< 3 m).

The sum of normalized distances is used instead of the center of mass because simulations showed that this yields an aggregation process where the sum of all the agents’ displacements is typically lower than if one were to use \(\vec{g}_i^{AG} \propto \sum \vec{r}_{ij}\). Thus the present form (Equation (4.5)) is less demanding on the total power consumed by the collective.

### 4.2.1.4 Target Navigation

In order to deploy the collective to a specific target location \(\vec{T}\) on the surface of water, the following behavioural rule is set in the direction of the target destination:
\[
\vec{g}_i^T = \frac{\vec{T} - \vec{r}_i}{\|\vec{T} - \vec{r}_i\|} \tag{4.6}
\]
This behavioural rule is useful for initial deployment, however, it is also useful for re-deployment in terms of directing agents to a specific location \(\vec{T}\). For the initial deployment of the buoys, we send a target location \(\vec{T}\) to all buoys (usually centre of the waterbody) before putting them on the waterbody. However, due to the imperfect communication some of the buoys (3-4 out of 50 buoys) may not receive the target location. To overcome this issue, the buoys are set to perform flocking by default, only a fraction of them need to explicitly use \(\vec{g}_i^T\) in order for the whole system to move towards target location and aggregate around \(\vec{T}\).

This is validated through a simulation of a leader-follower scenario (see Figure 4.2). A simulation of 51 agents in a leader-follower scenario was simulated, where the target \(\vec{T}\) was set to the
position of a particular buoy, “leader”. All the buoys were placed in their equilibrium configuration corresponding to $\mathcal{G} = \{\vec{g}^{AG}, \vec{g}^R\}$ in the initial state. A particular buoy was then tasked to move in a straight line at constant speed, while a fraction of the buoys become “active” followers by changing from $\vec{g}^{AG}$ to $\vec{g}^T$. The so called active followers were buoys that received direct target destination from the leader (which was the position of the leader). The simulation shows that the entire fleet of buoys is able to follow the leader with only $\sim 50\%$ of the buoys as active followers.

Figure 4.2: Simulation of leader-follower scenario in an imperfect communication. One of the buoys will act as a leader, by moving in a straight line at constant velocity while a certain fraction of the buoys “actively” follow that behaviour according to $\mathcal{G} = \{\vec{g}^T, \vec{g}^R\}$. The rest of the buoys behave according to the default $\mathcal{G} = \{\vec{g}^{AG}, \vec{g}^R\}$. The total ratio of followers is defined as the ratio of buoys that have the same velocity as the leader in the long-time asymptotic state. There are three typical outcomes for using this algorithm, shown by the insets: 1. When only $\sim 20\%$ of buoys is actively following the leader, the collective will split in two groups with the majority of buoys aggregating together and remaining stationary. 2. When half of the buoys are actively following the leader, the rest of the buoys will follow along but lag behind. 3. When a large fraction of the buoys ($\sim 80\%$) are actively following the leader, the remaining buoys will “passively” follow the leader even without knowledge of the leader’s position.

4.2.1.5 Geofencing

To perform dynamic area coverage, “geofencing” behavioural rule is defined. It attracts the agents towards the area $A < 0$ and is formulated as follows:

$$\vec{g}_i^{GF} = \frac{1}{1 + \exp\left(-A(\vec{r})\right)} \frac{\vec{\nabla} A}{\|\vec{\nabla} A\|}$$  \hspace{1cm} (4.7)
The direction of this term \( -\nabla A \) causes the agents to move towards decreasing \( A \), that is towards the interior of the surface. The scaling term, \( [1/(1 + \exp (-A))] \), will only affect the agent significantly when it is outside the region of interest and “attract” it back. Therefore, \( \|\vec{g}^{GF}\| \simeq 1 \) when outside region of interest and \( \|\vec{g}^{GF}\| \simeq 0 \) when inside.

### 4.2.2 Flocking Behaviour

Flocking is a behaviour observed in nature (in many bird species), which form large groups of individuals moving together toward a common target location. This behaviour emerges at the collective level in a distributed manner, as a consequence of local interactions between autonomous agents. It has received particular attention from several scientific communities—first by the computer graphics community [125], then followed by physicists [154]. Subsequently, the control community established a formal framework [81, 114], which has been put into practice and expanded in the context of multi-robot systems and swarm robotics [29, 150].

The classical “flocking” model, a staple of collective motion studies was introduced by Reynolds [125]. It assumes that the motion of an agent is composed of three terms: avoidance, alignment and attraction. This model and many implementations have been discussed elsewhere [154]. Recently, this flocking model has been successfully implemented by Vásárhelyi et al. for a flock of 10 quadcopters [152].

To achieve a cohesive flocking behaviour with our autonomous buoys, we implement the rule

\[
\vec{v}_i = v_0 \sum_{j=1}^{n_i} \left\{ \left( \frac{1}{n_i} - \frac{r_{ij}^2}{3r_{ij}^2} \right) \frac{\vec{r}_{ij}}{r_{ij}} + \frac{1}{n_i} \vec{v}_j \right\}
\]

where \( n_i \) is the number of neighbours, \( \vec{r}_{ij} = \vec{r}_j - \vec{r}_i \), \( r_{ij} = \|\vec{r}_{ij}\| \), and \( r_0 \) is a free parameter that controls the equilibrium nearest-neighbour distance. If there is no neighbour available, the buoy will perform its last known task or initial task (assigned during initial deployment). The first part of the equation aggregates the buoys towards the center and repels them from nearby buoys. The second part aligns the buoys’ velocity to move in a cohesive fashion. The factor \( 1/3 \) in the repulsion term is introduced phenomenologically so that a large number of agents following this rule will navigate in a lattice-like formation where each agent is approximately at a distance \( r_0 \) from its nearest neighbour (see section 4.2.1.1).

Flocking is the default behaviour for our fleet of buoys and it grants the system an extra layer of robustness against failures in the communication network. As discussed in section 4.2.1.4, the system can perform several collective actions even if fewer than all agents receive the appropriate command. This is made possible due to the “uninformed” agents passively mimicking the motion of nearby agents performing the same collective action the informed agents are actively performing.
4.2.3 Collective Navigation Behaviour

The elementary way to control a team of ASVs is to set a target location $\mathbf{T}$ for the group to reach. Naturally, we do not want all buoys to reach the same point, but for the collective to gather around the target location. Therefore, the following behaviour rule is implemented for navigating the buoys to a specific location

$$\mathbf{\dot{v}}_i = v_0 \left\{ \frac{\mathbf{T} - \mathbf{r}_i}{\| \mathbf{T} - \mathbf{r}_i \|} - \sum_{j \neq i} \frac{r_0^2}{3r_{ij}^2} \mathbf{r}_{ij} \right\}$$

This collective behaviour is similar to flocking, except that the buoys aggregate at a specified location or station-keep at that location. The target $\mathbf{T}$ can either be a fixed location or a time-dependent path. For leader-follower scheme implementation, a particular buoy’s position is set as the target location, $\mathbf{T}(t) = \mathbf{r}_I(t)$. This implementation allows an external controller to manually navigate the “leader” agent $I$ and have the collective to follow its motion.

In principle, deploying the buoys by using this rule requires information to be transmitted globally to every single buoy, as they all need to know the value of $\mathbf{T}$. When operating a large number of buoys in open, uncontrolled environments, it is extremely difficult to guarantee that a particular message will reach all the buoys, so one would assume that this mode of control is not very scalable nor robust. However, because of the default flocking behaviour, only a fraction of the buoys need to explicitly follow the target navigation behavioural rule in order for all of them to have a cohesive collective behaviour where they assemble around the target (see section 4.2.1.4). This is a good example of an “indirect” flow of information, where an agent can react to an environmental change even if it is unable to detect the change itself. For instance, the simulations discussed in section 4.2.1.4 show how some agents can “passively” follow a leader without having explicit information on the leader’s location.

Figure 4.3 shows an example of how a buoy following Equation (4.9) reaches a target when a group of aggregating buoys is in the way. If this buoy has $r_0$ comparable to that of the group members, they will move around their equilibrium position to open up space for the buoy as it travels through the group (Figure 4.3(a)). If the group members are fixed at their positions, the dynamics Equation (4.9) allows the moving buoy to circumvent the group and reach the goal (Figure 4.3(b)). Alternatively, if the moving buoy has a $r_0$ considerably smaller than the fixed group members, it will simply “sneak” through the collective to reach its goal (Figure 4.3(c)).

4.2.4 Area Coverage Behaviour

The intended use of this large-scale networked array of mobile sensing buoys is to monitor and characterize aquatic environments in regions of interest, which may vary depending on the application. For example, the networked array may be deployed in a specified area in a harbor...
4.2. Cooperative Control Algorithms

Figure 4.3: Simulation of collective avoidance behaviours in group while performing station-keeping task. A group of $N = 49$ buoys station-keep at target location with $r_0 = 100$ m. An additional buoy $i$ (lower right corner) is introduced and directed with Equation (4.9) towards a given goal (red cross mark) placed opposite the collective. Depending on the parameters, one can observe (a) yielding behaviour (zoom in), (b) bypassing behaviour, or (c) sneaking behaviour.

to assist in marine operations by monitoring key environmental and flow parameters. More interestingly, the area to monitor might not be specified externally or in advance, but instead be defined dynamically by the collective of buoys itself. By local processing of the sensed data, the buoys may determine the shape in which to self-deploy in order to track a range of biological markers, a particular temperature profile, or oil spill.

Thus the applications or the features being tracked determine the way the buoys are being deployed. There are two common ways to deploy the collective for an efficient monitoring of a given region—“blanket coverage” [60] and “barrier coverage”. Blanket coverage is used when buoys are tasked to sense a scalar field (such as temperature), where they are required to spread as uniformly as possible across the region of interest (ROI). By contrast, barrier coverage is used for tracking how a substance spreads, such as an algal bloom or an oil spill scenario. The buoys will detect the spread and position themselves uniformly along the contour of the region. Here, we describe the blanket coverage algorithm implemented on the fleet of buoys for dynamic area exploration.

Very often, the shape of the region to monitor will evolve with some arbitrary, unknown dynamics. Therefore, the fleet of buoys should have a responsive behaviour that allows them to dynamically spread across arbitrary areas, and adapt to changes in a timely manner. To obtain dynamic area coverage, we define a control algorithm that follows Equation (4.1), which is inspired by the potential-field approach to area coverage [75]. It also consists of a term aggregating the buoys towards the interior of the area and a term repelling the buoys from each other.
Given an arbitrary region described by

$$A(r) < 0,$$  \hspace{2cm} (4.10)

where $A$ is a signed distance function that increases monotonically outside the region, we define the area coverage behavioural rule as

$$\vec{v}_i' = v_0 \left\{ \frac{1}{1 + \exp\left(-\vec{v} \cdot r\right)} \cdot \frac{\vec{r}_i A}{\|\vec{v} A\|} - \sum_{j \sim i} \frac{r^3_{ij}}{r_{ij}^3} \right\}$$  \hspace{2cm} (4.11)

where the attraction term (proportional to $-\vec{v} A$) is scaled in such a way that it is $\simeq 1$ outside the area to cover ($A > 0$) and $\simeq 0$ inside of it ($A < 0$). Figure 4.4 shows the illustration of the equilibrium distribution of buoys following this behaviour for number of buoys, $N = 10, 20$ and 50 at different shapes of the area given by Equation (4.12).

![Figure 4.4](image)

Figure 4.4: Equilibrium configurations of number of buoys, $N$, for dynamic area coverage behaviour following Equation (4.11) for the area given by Equation (4.12). The area changes with $\alpha$, $\alpha = 0$ (top row), $\alpha = 1$ (middle row), and $\alpha = 2$ (bottom row).

### 4.3 Field Experiments

A series of field experiments was conducted, deploying up to 48 buoys on a calm body of water (Reservoir or Quarry) to test the efficacy of the buoys in performing station-keeping, avoidance, aggregation, leader-follower, and dynamic monitoring under real-world conditions. The experiments were performed under usual weather conditions of an equatorial climate—including rain showers, with average temperatures of around 29°C, dew point of 25°C, winds of 1.8 m/s, and pressures around 1009 hPa.

These experiments confirmed that, at the unit level, the buoys are capable of goal seeking, station keeping, and distributed communication. At the system level, the tests provided clear
evidence that the system can efficiently perform large-scale collective behaviour in the face of distributed, fragmented communications. In what follows, we show that the collective behaviour of the system and its typical response times are in good agreement with the predictions from simulations, thereby demonstrating that the mesh network strategy and the decentralized control algorithms provide a robust framework that can be scaled up to these system sizes.

### 4.3.1 Station–Keeping

It is not a trivial task to deploy all 40 - 50 buoys at once, therefore we dispatched them batch by batch. We set a specific location for every batch of buoys before deploying them on the body of water (usually centre of the experimental site). Figure 4.5 shows the deployment process at Quarry with the following weather conditions: wind speed of 2.3 m/s from south or southeast direction, water was calm and temperature of 31.2 °C. The first batch of 19 buoys were sent out and station-kept at the target location (indicate by red dots) while waiting for the next batch of 8 buoys (blue dots) to arrive. Upon arrival of the 8 buoys, the first and second batches swarmed together and continued to perform station-keeping task while waiting for third batch of 6 buoys to arrive. Once the third batch arrived, all three batches (33 buoys) swarmed together to perform the station-keeping task. This experiment validates that Equation (4.9) is able to navigate the buoys to the specified location and exhibit collective behaviour. In addition, this experiment also demonstrates that our fleet of buoys is scalable and robust, as they continued to perform the station-keeping task even as more buoys are added to the collective. The collective accommodated additional buoys, as other buoys joined the group, by increasing the spread area while maintaining the spread distance, \( r_0 = 5 \) (for this case).

![Figure 4.5: Initial deployment of buoys, done in batches. Collective buoys performing station-keeping task while waiting for new members to join in the collective. All together there were three batches. Red dots indicate the station-keep buoys while the blue dots indicate the joining buoys. Grey lines trailing the dots show the travel paths. Left: First batch station-keep while waiting for other buoys to join. Middle: Second batch joining the collective. Right: Third batch joining the collective formed by first and second batches.](image-url)

We analysed the accuracy of station-keeping by looking at the average drift distance of all the
buoys with respect to time (see Figure 4.6) over three experiments. All three experiments were performed on different days at 2 different locations (Reservoir and Quarry). Experiment 1 was conducted with 19 buoys and was performed at Quarry with the following weather conditions: wind speed of 2.1 m/s and calm water. Experiment 2 was conducted with 43 buoys at Quarry with the following weather conditions: wind speed of 2.3 m/s and ripple on water. Experiment 3 was conducted with 24 buoys was perform at Reservoir with the following weather conditions: wind speed of 2.4 m/s and ripple on water. The fleet of buoys was able to drive itself almost back to the initial position after 2 – 3 minutes, with an average drift of 2 – 3 m (including GPS accuracy < 3 m). This amount of drift is acceptable for our application as we are looking at mapping area in the order of 100 m to few kilometres. The result also shows that the wind speed has a direct impact on the drifting distance for our fleet of buoys, just as we would have expected.

Figure 4.6: Analysis of average drift while the fleet of buoys perform station-keeping task. All drift distances are calculated with respect to the initial point of each buoy and average were taken over all the buoys. Lines indicate average drift distance and the error bar denotes the standard deviation.

4.3.2 Avoidance

Experiments of buoys yielding to another buoy that passed through the swarm are shown in Figure 4.7. Figure 4.7a shows how a buoy (red star) following Equation (4.9) reaches a target when a group of aggregating buoys is in the way. The other buoys will move around their equilibrium position to open up a space for the buoy to pass through (blue line), as it has $r_0$ comparable to that of the collective buoys. Another experiment from a different day, Figure 4.7b, shows a buoy (red star) leaving the collective buoys (similar $r_0$ for all buoys) and those on its way yield for the buoy to travel through (blue line). This shows that the collective buoys that follow Equation (4.9) are able to avoid each other by relying on GPS data. These
experimental results are in agreement with the simulation result (see Figure 4.3(a)).

![Figure 4.7: Avoidance behaviours in group navigation with yielding behaviour. (a) N = 6 with \( r_0 = 10 \) m, with an additional buoy (red star) passing through the collective performing station-keep task. (b) N = 27, \( r_0 = 20 \) m, one of the buoys (red star) was commanded to leave its position during station-keeping.]

4.3.3 Aggregation

The purpose of our fleet of buoys is to measure scalar fields such as temperature, wave height, or to map waterbodies. The accuracy of reconstruction of these scalar fields depends on the spread of the buoys and their density in the area of interest. Therefore, we deployed our buoys in a loose configuration for large area coverage and a compact configuration for a small area in order to have high resolution over it.

To illustrate the situation, we performed an aggregation field experiment following Equation (4.9), with a group of 43 buoys at Quarry. The initial state of the experiment was when all the buoys were spread out at the centre of the Quarry and formed their equilibrium lattice arrangement. Figure 4.8a shows the trajectory of this experiment after a sudden reduction of \( r_0 \) from 50 m down to 5 m. The entire aggregation event took approximately 8 minutes, during which the area covered by the buoys decreased from 104,150 m\(^2\) to 1170 m\(^2\). The average speed for buoys on the outer ring was 0.4 m/s, and the ones at the inner rings aggregated at proportionally lower speeds of 0.25 to 0.15 m/s. Figure 4.8b shows the surface covered and density varying with time, where density is measured using Delaunay tessellation field estimator [131]. Figure 4.9 shows the change in the total surface covered by the buoys (through Delaunay tessellation plot) from \( t = 0 \) min at the initial state of the aggregation to the ending state, \( t = 8 \) min.
4.3.4 Leader-follower

The leader-follower behaviour [40] was validated by manually controlling (setting waypoints) one of the buoys, “leader”. While the rest of the buoys following the “leader”, by setting leader position as their target, $\mathbf{T}$ in Equation (4.9). The group traversed $\sim 400$ m in calm water with $r_0 = 5$. The trajectories of the buoys show that the collective is capable of maintaining a tight formation with a slight “tail” while following a leader even during sharp turns in its trajectory. This slight “tail” is also exhibited in the simulation result for $\sim 80\%$ active followers (see inset of Figure 4.2).
4.3. Field Experiments

Figure 4.10: Post field experiment reconstruction of leader-follower behaviour of 43 buoys with \( r_0 = 5 \), travelled for \( \sim 400 \) m. The path of the leader buoy is marked with a blue trail while the followers are marked with grey trails. Grey dots indicate the initial positions and red dots represent the termination of leader-follower behaviour.

4.3.5 Dynamic Monitoring

To validate the buoys’ capability to perform dynamic area coverage Equation (4.11), we deployed 20 buoys and tasked them with the following area covering

\[
A_{\alpha,\epsilon}(\hat{r}) = \frac{r^2 - R_{\alpha,\epsilon}(\hat{r})^2}{10R_0^2}
\]  

(4.12)

where

\[
R_{\alpha,\epsilon}(\hat{r}) = \frac{R_0}{2} \frac{2 - \alpha + 3\alpha(\hat{r} \cdot \hat{\epsilon})^2}{\sqrt{1 + \alpha/2 + 11\alpha^2/32}}
\]  

(4.13)

\( R_0 = 25 \) m, and initial \( \alpha = 0 \). Where \( \hat{\epsilon} \) is the principal axis. The buoys were deployed sequentially into the area of interest (circle) over a span of 8 minutes and covered the circle homogeneously by spreading out.

We used Voronoi tessellation to measure the quality of deployment of our buoys by partitioning the exploration area where the buoys were, into cells. Each buoy has its associated cell and can be considered as its designated monitoring area. Ideally, each of the cell is \( 1/N \) of the total area for each buoy to cover. With Voronoi tessellation, we quantified the efficiency of area coverage by the buoys using the area of the largest Voronoi cell (LVC) as a measure of how solicited the most solicited buoy is. The LVC area relative to the total exploration area, \( A_{LVC} \), is presented in Figure 4.11. The size of cell decreases as more buoys were deployed into the exploration area with time. By fitting the experimental results of \( A_{LVC} \) to a power-law of
the number of buoys, $n$, currently deployed (Figure 4.12), we find that

$$A_{LVC}(n) = (1.56 \pm 0.02)n^{-1.05\pm0.01}$$ (4.14)

which corroborates the scalability of the system as $A_{LVC}$ scales roughly as $1/n$, for $n$ up to $\sim 20$.

Figure 4.11: Dynamic area coverage performed by a group of 20 buoys exhibiting dynamic scalability capability of the collective. Left: Evolution of the largest Voronoi cell’s area with respect to the total exploration area, $A_{LVC}$ (red), compared with the “ideal” case of $1/n$ (Blue). Right: Number of buoys, $n$, inside the exploration area (in this case a circle) at a particular time.

Figure 4.12: Correlation of largest Voronoi cell’s area, $A_{LVC}$, and number of buoys, $n$, for the same data as Figure 4.11. The fit of the experimental data, $A_{LVC} \sim n^{-1.05}$, shows that the efficiency of the system grows roughly linear with the number of buoys.

Next we tested the flexibility of our collective buoys by varying the shape of the exploration area between a circle and a two-lobbed “dumbbell” with time. We did that by oscillating
α in Equation (4.13) between 0 and 2, and ensuring that the buoys are able to successfully adapt to these changes. We also ascertain that the buoys are able to re-position themselves to the changes accordingly (see Figure 4.13). Figure 4.14 shows the time evolution of $A_{LVC}$.

Figure 4.13: Pictures of the field experiment with 20 buoys performing dynamic area coverage following Equation (4.11), oscillating between a circle and a two-lobbed “dumbbell”. Insets: Post-experiment reconstruction of GPS location data, black dots represent buoy positions. Red dots and lines indicate the path of the buoys.

for flexibility experiment with 20 buoys. The experimental results (redline) show that $A_{LVC}$ oscillates between 5.6% ($N \times A_{LVC} = 1.12$) and 12.3% ($N \times A_{LVC} = 2.46$), with an average value slightly below 8% which follows closely to the simulation (blue line). We observed that experimental $A_{LVC}$ has large and short-lived pulse-like oscillations that occurred at $t = 9, 17, 20, \text{ and } 26 \text{ minutes}$. These “pulses” are the results of many perturbations we faced when operating a multi-robot system in a real world situation. Examples of these perturbations are inaccuracies of GPS localization and limited communication rate between buoys (currently set at 0.1 Hz). However, these pulses do not accumulate over time, thereby showing that our system is stable. This is due to the high responsiveness of the individual buoy and robustness of the collective, which allows the system to quickly recover from deviation.

Next we did other experiments to determine the responsiveness of our fleet of buoys by subjecting them to changes in area (at different frequencies). We increased the frequency, $\omega$, by oscillating from circle to dumbbell, and determined the area coverage captured by the buoys, (see Figure 4.15). The result shows that the responsiveness of our fleet of buoys decreases with
Figure 4.14: Flexibility of the collective by tasking 20 buoys to perform dynamic area coverage following Equation (4.11) that oscillating between a circle and a two-lobbed “dumbbell”. The buoys were covering an area of approximately 200 m². Evolution of the largest Voronoi cell’s area with respect to the total exploration area scaled with the number of buoys, \( N \times A_{LVC} \). The grey shapes represent the dynamic exploration area.

Increasing frequency, with a specific cutoff frequency, \( \omega_c \approx 0.02 \). This implies that our fleet of buoys is able to monitor changes in area in the order of one minute. Beyond this limit (less than one minute), the fleet of buoys is unable to keep up to the changes. Although our buoys are omni-directional, we did not design them to have a fast propulsion system (average speed = 0.45 m/s). Nevertheless, this response speed is sufficient for algal bloom or oil spill monitoring as changes during such events usually take more than one minute.

Figure 4.15: Field experimental data of the fleet of buoys performing dynamic area coverage with the area changing from circle to dumbbell, at different frequency \( \omega \).
4.4 Collective Data Analysis

With a fleet of buoys and quality sensors (see section 2.1.7) mounted on them, we are able to conduct real-world applications of water monitoring. Collective data is collected whenever the buoys are deployed, with every buoy broadcasting its sensed data to other buoys at time intervals specified by the user. This transmission is done through the distributed Xbee® mesh network (see section 2.2.1). The collectively sensed temperature data is used for a temperature field reconstruction by individual buoys on-the-fly. The computation is done whenever new data is received, with a default time interval of ten seconds.

4.4.1 Numerical Method for Scalar Field Reconstruction

Presently, all buoys are computing the same domain with practically the same amount of data (not all buoys will receive all the data sent by other buoys owing to the nature of the mesh network) in a non-distributed way. Thus far, we have yet to achieve a fully distributed computation with buoys in the same area (sub-group or cluster) sharing their computational information and task (sub-domains). Therefore, we need to look for an efficient numerical method for scalar field reconstruction on small, low-power portable computational platforms (BeagleBone Black). Hermite functions coupled with a Sparse Matrix Interpolation technique [163] meet the requirements for our application. The Hermite polynomial is recursive, orthogonal or orthonormal, and has having an underlying polynomial definition that may closely resemble the physics of the scalar fields one wishes to uncover.

Hermite Functions and Polynomials

Hermite functions are of interest because of their recursive relation and orthogonality. It is weighted by the Gaussian distribution, the limits are smoothed to blend well with the estimated far field values [163]. In addition, the orthogonality of this function makes scalar field reconstruction easier and faster when computing coefficients as compared to other methods (e.g. non-linear least square method). The orthogonality property of the normalized Hermite functions $\psi_n$ is

$$\int_{-\infty}^{\infty} \psi_m(x) \psi_n(x) dx = \delta_{mn}$$

(4.15)

where $\delta_{mn}$ is Kronecker, $\delta$, which is one when $m = n$ and zero otherwise [28]. The Hermite functions can be expressed as follows

$$\psi_n(x) = (2^n n! \sqrt{\pi})^{\frac{1}{2}} e^{-\frac{x^2}{2}} H_n(x)$$

(4.16)
where \( H_n(x) \) is the \( n \)-th Hermite polynomial. The recursive relation for Hermite functions are

\[
\psi_0(x) = \pi^{-\frac{1}{4}} e^{-\frac{x^2}{2}}
\]

\[
\psi_1(x) = \pi^{-\frac{1}{4}} \sqrt{2x} e^{-\frac{x^2}{2}}
\]

\[
\psi_{n+1}(x) = \sqrt{\frac{2}{n+1}} x \psi_n(x) - \sqrt{\frac{n}{n+1}} \psi_{n-1}(x)
\]

These recursive terms are very useful for finding the coefficients for approximating the target function.

\( H_n(x) \) is the \( n \)-th order Hermite polynomial [142] (also known as physicist’s version of Hermite polynomials), where Rodrigues [59] defines it as follows:

\[
H_n(x) = (-1)^n x^n \frac{d^n}{dx^n} e^{-x^2}
\]

and the recursive formula [28]

\[
H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)
\]

where \( H_0(x) = 1 \) and \( H_1(x) = 2x \). The algorithm, Listing 4.1, is for establishing a Hermite function with \( n \) polynomial with locations of evaluation.

**Listing 4.1: Algorithm for establishing a Hermite Function**

```python
def _poly(N, xi):
    """
    Evaluates Hermite Function of order N
    Inputs: N, xi
    N: Order of Hermite Polynomial
    xi: Evaluated location
    Output: Yn
    Yn: Hermite Function
    """
    if N == 1:
        return pi**(-1./4.)*exp((-1./2.)*xi**2)
    Yn = ones((N, xi.size))
    Yn[0] = pi**(-1./4.)*exp((-1./2.)*xi**2)
    Yn[1] = sqrt(2.)*pi**(-1./4.)*xi*exp((-1./2.)*xi**2)
    if N == 2:
        return Yn
    for n in range(2, N):
        Yn[n] = xi*sqrt(2./n)*Yn[n-1] - sqrt((n-1)/n)*Yn[n-2]
```

With the Hermite functions and polynomials described above, we can represent any function in any domain \( \in [-\infty, \infty] \) as

\[
f(x) = \sum_m c_m \psi_m(x)
\]

(4.22)

where \( c_m \) are the coefficients of the function. For water quality monitoring, we will be using a two-dimensional form,

\[
f(x, y) = \sum_m \sum_n c_{mn} \psi_m(x) \psi_n(y)
\]

(4.23)

A rapid and accurate method for determining the polynomial coefficients is obtained by applying an inner product to the following integral [163]

\[
c_{mn} = \iint f(x, y) \psi_m(x) \psi_n(y) dx dy
\]

(4.24)

Once the coefficients are obtained, we can use Equation (4.23) to approximate the best fit for our target function. Listing 4.2 is the algorithm that is used for calculating the coefficients of a 2D function.

Listing 4.2: Algorithm for calculating the coefficients of a 2D function

```python
from numpy import ones, array, pi, sqrt, exp, linspace

def fit_2d(fn, shp, **kwargs):
    """
    Using Orthonormal properties, this algorithm calculates the inner product of the target function against the Hermite Functions in order to return the coefficients which represent the best fit 2D series.
    """
    M, N = shp
    ax = kwargs.get('ax', 5.)
    bx = kwargs.get('bx', 5.)
    ay = kwargs.get('ay', 5.)
    by = kwargs.get('by', 5.)
    is_fn = isfunction(fn)
    x_num, y_num = (50, 50) if is_fn else fn.shape
    x = linspace(ax, bx, num=x_num)
    y = linspace(ay, by, num=y_num)
    return Yn
```

Where The Rubber Meets The Road

Sparse Matrix Interpolation

For this research, we would like to approximate large spatial regions with a limited number of buoys. Therefore, connecting information from distant buoys is desirable for reconstruction. In order to overcome this issue, higher order polynomials are used. With higher polynomial degree, we will be able to capture features along the far field and have better prediction in the region where the buoys are deployed. However, this method is computationally costly—higher polynomial degree means longer computational time.

We introduced Sparse Matrix Interpolation (SMI) to overcome this shortcoming. This method uses a spline interpolated sparse matrix as a base for evaluating the inner product to obtain good fits with lower sample size (collected data). For our application, a basic cubic spline interpolation will be sufficient to infer scalar field on the locations (or grid) that have no collective data [163]. By doing so, we increase the sample size of the data (collected data and interpolated data) and this in turn allows us to feed this grid data to Hermite functions with a lower polynomial degree to approximate the target field. With SMI we are able to converge on a near exact reconstruction with as little as 10% sampled area [163]. In summary, we leveraged on the advantages of Hermite functions and to overcome its shortcoming with SMI for scalar field reconstruction on the buoys.

The Hermite functions coupled with a Sparse Matrix Interpolation technique (mentioned above) can be used for any 2D scalar field reconstruction—such as temperature, pH, or DO—from the data collected by the fleet of buoys. First, the collective data and the grid locations are fed to SMI algorithm to fill up location without data using cubic spline interpolation.
Next, the increased data (collected data together with interpolated data) is input to Hermite expansion to estimate the coefficients. Lastly, the coefficients are used to estimate the best fit for the scalar field reconstruction. This process is repeated whenever the buoys receive new information (default 10 s interval).

4.4.2 Field Experiments - Scalar Field Reconstruction

Temperature data is the only collective data among all the three parameters that were measured, as all buoys have a temperature probe. There is only one buoy with a DO probe and a pH probe mounted on it. Temperature data is collected by the buoys as they swarm around in the Reservoir or Quarry. It is broadcasted together with the rest of the information by the buoys. Every buoy is capable of storing up to 5000 data points of temperature measurements.

Temperature Field Reconstruction

Every buoy performs real-time temperature field reconstruction based on the collective data (individual sensed data and data sensed by other buoys). Figure 4.16 shows the real-time temperature field reconstruction superimposed on satellite image of the Reservoir displayed by the GUI (screen captured of an instance during one of the experiments) on the control station (computer) during an experiment. This reconstruction was based on the data that was intercepted and extracted by the control station since the initial deployment until the very instant when a screen capture was done. The intercepted data closely matches the data held on each buoy, but the GUI might not have exactly the same amount of data as each buoy deal to the imperfect communication network. Dotted red lines are the selected temperature contours within the reconstructed field, and the buoys are represented by coloured dots with outline. The blue regions are the lower temperature areas, whereas the red regions are the higher temperature areas. Although, we have yet to achieve distributed computation for temperature field reconstruction, our fleet of buoys is considered as an advanced system compared to a system that does temperature field reconstruction on centralised computer (near real-time) [87] or a system that does post experiment temperature field reconstruction [51].

We can direct the buoys to follow one of the contour lines in Figure 4.16 by sending commands from the control station. Figure 4.17 shows the buoys collectively position themselves on one of the red dotted lines in Figure 4.16. Alternatively, this behaviour can be programmed into the buoys to detect anomalies and to follow the abnormal high or low temperature contours.

Another water surface temperature monitoring experiment was conducted at the reservoir around 14:00 – 16:00 hours (see Figure 4.18). Weather conditions from a nearby meteorological centre were as follows: temperature was approximately 31.2 °C, mean wind speed of 2.3 m/s from South or Southeast. 43 out of 48 buoys that were deployed on a moderately calm body of water
100 Where The Rubber Meets The Road

Figure 4.16: Real time reconstruction of the temperature field during a field experiment extracted from control station graphical user interface (GUI). Dotted red lines indicate the selected interest of temperature contour. The slightly bigger dots with different colours indicate the present location of each buoy when the screen was captured. The blue regions are the lower temperature areas, whereas the red regions are the higher temperature areas.

Figure 4.17: Post experiment reconstruction of the temperature contours (blue line) based on the stored GPS data of 23 buoys. Grey dots indicate the starting position when contour-follow command was activated. Grey lines are the paths taken by different buoys toward the contour. Red dots are the final locations of the buoys.

managed to complete the assigned task. Two buoys went astray, two lost control/communication during the spreading process and one malfunctioned. For this experiment, all the buoys were sent to the centre of the reservoir and to perform a maximum spread followed by aggregation. These series of tasks were created by varying the repulsion strength, $r_0$, (see section 4.2.1.1) from 10 m to 50 m and to 5 m (see Figure 4.18a). Post experiment temperature field reconstructions
show that the buoys become better at mapping the water surface temperature as more data is collected (refer to Appendix E for scalar field reconstruction algorithms and sample data). This is in line with one of the requirements of Hermite functions. Figure 4.18b shows the reconstruction at initial state, and Figure 4.18c display the temperature filed where all the buoys are at maximum spread. Lastly Figure 4.18d shows the final reconstruction from all the data collected from the start to the end.

Figure 4.18: Post experiment temperature field reconstruction of 43 collective buoys data. (a) The path taken by collectively moving buoys while mapping the temperature profile of the water. Spreading and aggregation paths are created by varying the repulsion strength from \( r_0 = 10 \text{ m} \) to 50 m and to 5 m represent by grey line. Green dots indicate the initial position, while blue dots represent the maximum spread of the experiment. Red dots are the terminal locations for all the buoys. Reconstruction of the temperature field of the water based on collective data at (b) initial state, (c) maximum spread, and (d) terminal positions. Note: (c) and (d) share the same colorbar.

Temperature difference for the two experiments shown in Figure 4.16 and Figure 4.18 are approximately 2 °C. The typical variation of surface temperature in Singapore reservoirs is approximately 1 – 2 °C [91, 161]. Our experiments show that we have successfully designed a fleet of buoys that is able to measure water surface temperature and perform real-time scalar field reconstruction.
Other Sensory Data

Given our low cost approach, we only purchased one DO probe and one pH probe for measurement and proof of concept. Figure 4.19 shows the pathways of data measurement for DO (represented by red line) and pH (represented by blue line) during the field experiment. The average measurement for DO is 10.7 mg/L and pH is 8.1. The DO and pH measurement of another (Kranji) reservoir done by Te et al. show that DO could range from 3 mg/L – 14.4 mg/L, with the average DO at 7 mg/L, and the average pH value at 8.5, but could range from 7.1 – 10.35 [145]. Our DO and pH measurement fall within the range mentioned by Ref [145]. This show that our buoy is versatile as it can incorporate different type of sensors for water quality monitoring. To the best of our knowledge, our buoy is the only system that is able to perform real-time DO and pH measurement and has the ability to perform real-time scalar field reconstruction.

Figure 4.19: Pathway of data measurement for DO (represented by red line) and pH (represented by blue line) during the field experiment, the green dot represents the starting location of the buoy and the orange dot represents the ending location of the buoy.

4.5 Experimental Hindsight

Performing field experiments with a relatively large fleet of buoys requires addressing a number of challenges: scout for field test locations, organize transportation, setup and deployment of the system, troubleshoot and recover, post-process, and analyze the data.

The size of the system imposes serious limitations on the waterbodies adequate for testing and experimentation in Singapore. Small-scale experiments were carried out on land by moving the units on trolleys to manually simulate the thruster directions. The data gathered from such
tests is qualitative at best and they are hard to scale up because they require about as many researchers as buoys.

Transportation of a fairly large number of units to the field test site is a logistical challenge. A design of the hull that allows us to easily stack many units in a compact formation proved to be of paramount importance (see Figure 4.20). However, having to transport these in a van further limits the set of waterbodies eligible for testing.

![Figure 4.20: A fleet of buoys stacked up in a van, ready to be transported to field test site.](image)

Initialisation and deployment of the buoys in the field poses challenges that should not be underestimated as the fleet of the buoys get larger. With the fleet of buoys, the longer the time elapsed between activating the first buoy and deploying the last buoy onto the waterbody, the shorter the testing time will be. We found that this setup time could be shortened if we first spread the buoys around the waterbody (once they are initialised) by using a kayak, instead of injecting them from a single point.

In order to maximize the usefulness of field tests during development, special attention was granted to having a fast recovery of faulty units and on-site debugging. For this reason, we developed a graphical user interface (GUI) for data processing and monitoring of the fleet of buoys, which gathers the messages broadcast by the buoys to report the state of the whole system (estimated locations, current behaviour, battery levels, etc.) that would allow us to identify potential problems of faulty units in situ (see section 2.2.3). Having instant feedback through the GUI would, in many instances, be enough to be able to make an educated guess on whether a particular observed issue was related to the mechanics, the electronics, or the software.

Post-processing and data analysis of the experimental results can introduce considerable overhead time if not dealt with properly. For instance, post-processing can be automated by having the units connect to a wireless network in the lab and uploading their logged data to a
computer that processes them to generate a video of the reconstructed GPS trajectories along with any relevant metric. In our experience, we found that having post-processed data in under two hours from the completion of the experiment translates into a responsive research as it allows us to tune the field tests based on the feedback, especially when several field tests were performed on consecutive days.

4.6 Summary

Having a large fleet of small, low-cost, autonomous buoys with sensing capabilities offers a great potential for pervasive and persistent monitoring of aquatic environments. In this chapter, we demonstrated the possibilities of using these buoys for pervasive monitoring by deploying up to 48 buoys over an area approximately 0.32 km² in an inland waterbody. The autonomous buoys were tasked with operations relevant for monitoring, such as station-keeping, aggregation, leader-follower, and dynamic area coverage.

We have successfully operated the collective buoys to perform collective flocking, navigation, and area coverage at the system level by implementing cooperative control algorithms with new expressions for the local update rules. These collective behaviours have been tested in a series of open waterbodies field experiments that allow us to characterize the performance of the system under real-world conditions and imperfect distributed mesh communications network.

A novel metric to quantify the scalability and flexibility of the deployment for area coverage is introduced. With this metric, we show that the performance of the system scales approximately linearly with the number of buoys deployed. By testing the collective buoys with changing target area, we find that the flexibility is in good agreement with predictions from simulations. There is no metric for robustness in our case, however it is shown through the ability to continue swarming even though some of the buoys encountered malfunctions, went astray, behaved erratically, or stopped functioning (due to flat battery) altogether.

We have demonstrated pervasive water monitoring by measuring the water surface temperature, DO, and pH using blanket coverage method (water quality monitoring). The buoys are able to use the sensed data (temperature) to perform real-time temperature field reconstruction albeit not computed in a fully distributed way.

The collective buoys with pervasive monitoring capabilities were developed using swarm robotics design principles of scalability, robustness, and flexibility. The field experiments allow us to quantify the scalability of the system up to the group sizes considered and its flexibility to adapt to intrinsically dynamic environments. Given the system design, its efficiency is expected to remain for a significantly larger number of buoys with scalable distributed communication strategy.
Chapter 5

Conclusions

Given the considerable challenges in understanding and studying a range of environmental problems in aquatic environments, there is a clear need for pervasive and permanent monitoring with adaptivity in both space and time, especially in the submesoscales (0.1 km – 2 km) to mesoscale (2 km – 20 km). This dissertation pushes the frontiers of scientific understanding and technological ability required for developing a system to monitor aquatic environments pervasively. From the theoretical standpoint specifically, we have studied the system-level design principles and developed a range of cooperative control strategies. We have also studied the influence that the interaction network topology has on the responsiveness of the system by using two distinct modelling approaches: a distributed linear leader-follower consensus protocol and an agent-based self-propelled particles model, the Vicsek Model. Subsequently, this theoretical knowledge and the uncovered principles of dynamic collective behaviours are used to assemble a robust fleet of ASVs. Finally, these cooperative control strategies are tested and validated experimentally using our fleet of ASVs by performing collective behaviours such as flocking, navigation and dynamic area coverage.

5.1 Key Accomplishments

This research has achieved four key accomplishments and contributed to the fields of multi-agent system dynamics, autonomous robotic swarm, and design of swarming systems. The key accomplishments of this dissertation can be summarised as follows:

- **Effect of number of connections on collective response:**
  We have provided evidence that it is optimal for artificial swarms to limit the amount of interaction through the analysis of two models (distributed linear leader-follower consensus protocol and Vicsek Model). These two models show a similar phenomenology on how the number of connections affect the effectiveness in the group-level response. The optimal connectivity depends on the dynamics of the perturbation that the system faces (see Figure 5.1). The system requires an increase in number of connections when subjected to slow perturbations, and the opposite is true for fast perturbations. However, for the intermediate frequency range there is a certain number of connections at which the responsiveness peaks, and a further increase in connections has a detrimental effect
on the responsiveness of the system. Through Vicsek Model (VM), we have provided an estimation for the optimal number of neighbours, $k_{\text{peak}}$, in systems, like ours, following VM.

Given that these models are relatively general and unadorned, we suggest that this non-trivial relation between responsiveness and connectivity may be a general feature of a wide range of complex systems involving distributed consensus. Besides shedding new light on our understanding of collective behaviour, these observations have far-reaching implications for the design of the interaction network and artificial swarms.

![Graph showing the responsiveness of a distributed leader-follower consensus protocol with $N = 2048$ agents following a single leader. Total amplitude gain (normalized by the number of agents) as a function of connectivity, $k$, at different oscillating frequencies, $\omega$, which the leader inputs into the system. The solid colour lines correspond to a frequency, $\omega$, equal to the value specified in the legend. The dotted lines correspond to the frequencies evenly distributed between the solid lines.](image)

Figure 5.1: Responsiveness of a distributed leader-follower consensus protocol with $N = 2048$ agents following a single leader. Total amplitude gain (normalized by the number of agents) as a function of connectivity, $k$, at different oscillating frequencies, $\omega$, which the leader inputs into the system. The solid colour lines correspond to a frequency, $\omega$, equal to the value specified in the legend. The dotted lines correspond to the frequencies evenly distributed between the solid lines.

- **Effect of forced switching on collective response:**
  
  Through a network-theoretic analysis of the two paradigmatic models, we uncovered that an increase in the level of random forced switching increases the speed to consensus. However, the responsiveness of the system depends on forced switching and the dynamics of the perturbations.

  For systems modelled by the distributed linear leader-follower consensus protocol, we observed that random forced switching (or rewiring) increases the responsiveness of the system at low frequency. At high frequency, the amount of random forced switching with $p < 0.01$ does not have an effect on the responsiveness of the system, but for $p > 0.01$ it has a detrimental effect on the collective response. There is a transitional range (which is also known as the middle frequency range) when the positive effect—the responsiveness of the system increases with more random rewiring—of random forced switching on the collective
response switches to a negative effect—the responsiveness of the system decreases with more random rewiring.

For VM with global forced switching, we identified that any amount of forced switching leads to a reduction in the responsiveness but an increase in speed to consensus. For the system with local extended forced switching, it also exhibits a similar trade-off for a high number of connections (beyond the optimal connectivity). However, a certain amount of local extend forced switching for low number of connections (before the optimal connectivity) increases the responsiveness of the system (see Figure 5.2). This result shows that one can tune its system to operate at a high level of responsiveness while using the same number of connections.

![Figure 5.2: Responsiveness of the system with varying connectivity, k, formed by combining number of connections and the number of forced switching, ks, k + ks. The trend is the same as Figure 3.11a for N = 2048.](image)

This work thus provides design insights and guidelines for artificial swarms and their interaction networks.

- **Autonomous buoy:**
  We have designed and assembled a low-cost (SGD1500), compact, omni-direction, and autonomous buoy with water quality monitoring ability (see Figure 5.3). The original design enables high manoeuvrability and effective operations as a standalone autonomous surface vehicle.

  The ease of fast deployment and re-deployment allows users to do a quick survey of target and/or problematic aquatic environment at low cost before deploying highly sophisticated and expensive equipment requiring complex logistic supports. This autonomous buoy can be driven by scheduled way-points or manually controlled by the user via radio to any location on the waterbody for measurement.
Figure 5.3: An autonomous buoy that is equipped with water quality monitoring sensors can function alone or in group.

Presently, measurements of water quality are either measured from the bank or by driving a boat to the designated area. With our autonomous buoy, this data can be gathered in a faster and safer manner for a fraction of the cost.

- Swarm of buoys:
  We have developed novel cooperative control strategies and successfully tested them using a fleet of 48 buoys in fully unstructured environments of approximately 0.32 km$^2$ and in the absence of any supporting infrastructure. To date, our fleet of buoys is the largest such system reported in the literature. This fleet of buoys is able to perform flocking, collective navigation, dynamic area coverage, pervasive water quality monitoring (by measuring surface temperature, DO and pH), and real-time scalar field reconstruction.

  We have introduced a novel metric to quantify the flexibility and scalability of the deployment for the particular collective behaviour of dynamic area coverage (see Figure 5.4). With this metric, we found that the performance of the system scales approximately linearly with the number of buoys deployed. By testing the fleet of buoys with a changing target area, we find that the flexibility is in good agreement with predictions from simulations. We have demonstrated that our fleet of buoys is able to respond to changes in dynamic environments in the order of one minute, therefore showcasing high temporal flexibility.

  This fleet of buoys can be used for predicting the appearance of algal bloom by mounting the appropriate sensors on the system.

5.2 Outlook And Perspectives

This research work has made a big leap in our team’s ultimate goal of designing and assembling a large fleet of buoys with distributed computing ability to monitor the environment and process
5.2. Outlook And Perspectives

data collected by other buoys on-the-fly for aquatic environments. There is of course still more work to be done and suggestions for future research are as follows:

- **Theories validation:**
  We need to validate our findings from the two consensus models for artificial swarm design by implementing and testing them on different platforms, e.g. swarm of land robots or aerial robots.

- **Algorithms:**
  We need to develop a set of algorithms that is able to assimilate collected data and distribute it across all buoys to achieve a truly distributed computing system. This set of algorithms should have the ability to predict the changes of the environments (at least few steps ahead) based on mathematical models. For example, if the swarm is used to monitor oil spill, this set of algorithms should be able to predict the area of spill and spread based on advection or diffusion theory instead of searching the area and boundary “blindly”.

Another set of algorithms for efficient data distribution has to be developed to keep up with the large number of data collected from the buoys. This set of algorithms should be smart enough to determine which data is useful for the fleet of buoys, and which buoys should the data be sent to—through the communication network in the most efficient way.

With the integration of robots with different features (e.g. some robots with faster motor speed or larger battery capacity), or a swarm that is heterogeneous in nature, there is a need to develop a set of algorithms for decision-making. For example, if one buoy that is far from the swarm has detected an “interesting” feature (e.g. large temperature gradient), this set of algorithms will then decide which buoy (a buoy with faster motor speed will be preferred over one with low motor speed) to send for verification.

Figure 5.4: Picture of a field experiment with a fleet of buoys performing dynamic area coverage.
• Physical Design:
We could extend the sensing abilities of our buoys by installing a low cost Lidar sensor. In that case, our buoys will have a true obstacle avoidance ability instead of being entirely based on GPS. This will improve the capability of the buoys for obstacle avoidance (e.g. collision with ships or large floating object such as lost cargo) and for re-routing their path to destination. In addition, we could use this for search and identification applications, such as looking for missing aircraft parts that crashed into the ocean.

Another worthwhile expansion is heterogeneous swarming in which some of the units may have different properties and/or features. For instance, some of the buoys may have bigger platforms that have the ability to harvest energy or contain large battery capacity to provide charging services to smaller buoys. Different sensors could be mounted on some units of this heterogeneous swarm of buoys to perform different sensing. For example, some could have sensors that can extend deep into the waterbodies to measure water quality at different depths. This could further reduce the cost of implementing expensive sensors like DO, pH, and salinity sensors, where only a fraction of the buoys have these sensors.

Finally, a vision for our group is to have heterogeneous swarming capabilities that consist of aerial drones, surface vehicles and underwater vehicles. Aerial drones will provide a bird’s eye view or overview of the situation and enhance communication of data among the buoys. The buoys will then use this data to explore the water surface environment and map changes of the dynamic environments. Lastly, the buoys would provide “guidance” to underwater vehicles for underwater exploration and measurement.

This is certainly an exciting research area with tremendous potential to provide innovative solutions for monitoring and exploring aquatic environments. It provides an avenue to minimise the cost of damage for the fish farmers due to algal bloom (similar to the events that happened in 2014 and 2015) and allows scientists to learn more about the ocean.
A.1 Stability Calculation

Buoy stability calculation using a two-dimensional estimation for simplicity, as the sphere has the same shape (circle) in all directions. To calculate the centre of buoyancy we used the centroid formula for semi-circle, with $R = 16$ cm:

\[
B = \frac{4R}{3\pi} = \frac{4 \times 0.16}{3\pi} = 0.068 \, m
\]

Thus the distance from $B$ to $G$, $BG$, is

\[
BG = 0.068 - \frac{10}{100} = -0.032 \, m
\]
Now, we calculate the distance between $B$ and $M$, $BM$, for a circle

\[ BM = \frac{I}{V} \]

\[ I = \frac{\pi R^4}{4} \]

\[ V = \frac{\pi R^2}{2} \]

\[ BM = \frac{R^2}{2} \]

\[ BM = 0.005 \text{ m} \]

Where $I$ is the moment of inertia of the waterplane area and $V$ is the volume of liquid displaced by the body. Next, we find the distance between $G$ and $M$, $GM$,

\[ GM = BM - BG \]

\[ GM = 0.005 + 0.032 \]

\[ GM = 0.037 \text{ m} \]

\[ GM = 3.7 \text{ cm} \]

With the metacentric height at 3.7 cm, it shows that our buoy is stable (since $GM$ is positive) [63].

### A.2 Theoretical Analysis for Linear Consensus Protocol

Here, we show the theoretical analysis for distributed linear leader-follower consensus protocol using a simplified ring lattice with four nodes and two connections for each node. Using

![Figure A.2: A simple ring lattice with four nodes and two connections for each node using leader-follower consensus protocol. u is the input or leader (represented by green dot), with three follower (represented by red and blue dots) and output (blue dot).](image-url)
Equation (3.14), we can get the following first order equations for all nodes, $x_1$, $x_2$, and $x_3$:

$$\frac{dx_1}{dt} = \frac{1}{2}(u + x_2) - x_1$$

$$\frac{dx_2}{dt} = \frac{1}{2}(x_1 + x_3) - x_2$$

$$\frac{dx_3}{dt} = \frac{1}{2}(u + x_2) - x_3$$

With these equations ($\dot{x}_i = \frac{dx_i}{dt}$) we can re-arrange them to fit into Equation (3.16) and evaluate $x_2$ as an output.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -1 & \frac{1}{2} & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} \\ 0 & \frac{1}{2} & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} [u] \quad (A.1)$$

$$y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + 0 [u] \quad (A.2)$$

Taking Laplace transform on both Equation (A.1) and Equation (A.2), and substituting them together we get the final gain equation:

$$G(s) = \frac{Y(s)}{U(s)} = C [sI - A]^{-1} B$$

$$\frac{Y(s)}{U(s)} = \frac{1}{2s^2 + 4s + 1} \quad (A.3)$$

With that we can calculate the gain by substituting $s$ with frequency, $j\omega$, into equation Equation (A.3).
Appendix B

Buoy GUI and Buoy Operating System Configurations

B.1 Configuration of MacOS computer for Buoy GUI

The completion of these three steps is necessary to run the buoy GUI on computer with Mac OS:

1. Ensure the following programs are installed on the computer:
   - Python 3
   - Tcl/Tk
   - pyserial
   - pyscipy
   - matplotlib

2. Configure serial connection (via USB) between Xbee and the computer using xctu (it can be downloaded from the following URL: https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu). After the successful installation of xctu, connect the Xbee to the computer’s USB port via XBee Explorer USB connector and follow the instructions given by xctu for establishing the communication between the XBee and the computer (ref to https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu).

3. Store all the GUI source codes (can be obtained by requesting from SUTD Applied Complexity Group, URL: http://people.sutd.edu.sg/acg) in a folder anywhere on the computer and create an “Output” folder for storing output files of the received data.

After completing the above three steps, navigate to the folder where GUI source codes are stored in the terminal window and ensure that the Xbee is connected to the computer. Then, run Buoy_Main.py (see Listing B.1) on the terminal and the GUI will appear (see Figure 2.25).

Listing B.1: GUI_Main.py

```python
#!/usr/local/bin/python
```
from XBee_API import XBeeAPI
from GUI_Data import BuoyData
from XB_Finder import serial_ports
import GUI_Math as BM
import matplotlib
matplotlib.use('TkAgg')
from GUI_app import Buoy_GUI
from GUI_Plots import animate_fig, update_history

try:
    # Python 2
    from Tkinter import *
except:
    # Python 3
    from tkinter import *

from GUI_Math import Buoy_GUI
from GUI_Plots import animate_fig, update_history

 XB = XBeeAPI(serial_ports())
 BD = BuoyData()

 def subLoop():
     
     XB.read()
 while XB.RxFrames:
     sender, line = XB.exe_RxFrame()
 
     if sender and line:
         
         try:
             line = bytearray.fromhex(line).decode()
         except UnicodeDecodeError:
             continue

         line = line.replace('
', '')
         line = line.replace('}', '')

         line = sender + ', ' + line

         # Write Incoming line to Output
```
BD.write(str(time()) + ',' + line + '\n')

# Parse Incoming line and execute calculations
line = BD.parse(line)
BD.Calcs = BM.calc(BD.Data)

# Use the Buoy Name if available, if not use hex ID
split = line.split(',,')
try:
    split[0] = BD.BoBID[split[0]]['name']
except KeyError as E:
    pass

# Update Streaming Serial
stream_box = app.frames['GUI_Main'].stream_box
stream_box.insert(0.0, ','.join(split) + '\n')

app.after(50, subLoop)

def subsubLoop():
    app.frames['GUI_Main'].set_buoyIDs()
    app.after(10000, subsubLoop)

# Load BD.BoBID from file
names_raw = loadtxt('Buoy.names', delimiter='\t', skiprows=1, dtype=bytes)
BD.BoBID = {}
for row in names_raw.astype(str):
    BD.BoBID[row[2]] = {'name': row[1], 'IP': row[3]}
app = Buoy_GUI(BD=BD, XB=XB)
app.after(10, subLoop)
app.after(5000, subsubLoop)

GM = app.frames['GUI_Main']
anim_kw = {'fargs': (BD, GM), 'interval': 1*10**3}
fig = app.frames['GUI_Main'].figs[0]
main = animation.FuncAnimation(fig, animate Fig, **anim_kw)

BH = app.frames['Buoy_History']
update_kw = {'fargs': (BD, BH), 'interval': 10*10**3}
fig = app.frames['Buoy_History'].figs[0]
history = animation.FuncAnimation(fig, update_history, **update_kw)
```

B.2 Configuration of BeagleBone Black for Buoy Operating System

These are the following steps for configuring BeagleBone Black (BBB) to auto-load buoy operating system (BOS) upon power on:

1. Update image on BBB, the latest image can be downloaded from the following http://beagleboard.org/latest-images (ensure that it is for Flashing eMMC).

2. Run Buoy_Setup.sh (see Listing B.2) to configure BBB for BOS: update operating system (Debian), install the necessary programmes to run BOS, create folder to store BOS source codes, and configure Debian to load Buoy_Main.py on startup.

3. Copy BOS source codes (can be obtained by requesting from SUTD Applied Complexity Group, URL: http://people.sutd.edu.sg/acg) from computer to BBB to the following directory: home/debian/Python/.

Upon completing the steps above, BBB is ready to run BOS on startup.

Listing B.2: Buoy_Setup.sh

```bash
#!/bin/bash

# All the commands used to install are necessary

# Update and upgrade, which might take a while
echo "***Installing System\_Update***"
sudo apt-get update -y
sudo apt-get upgrade -y

# Sync the time
echo "***Synchronizing Date and Time***"
sudo ntpdate -u us.pool.ntp.org

# Python Tools
echo "***Installing\_essential\_Python\_tools***"
sudo apt-get install --y build-essential python python-dev python-setuptools
sudo apt-get install --y python-pip python-smbus
sudo pip -H install --upgrade pip

echo "***Installing\_Adafruit\_BBIO\_library***"
sudo pip -H install --upgrade Adafruit_BBIO

echo "***Installing\_scipy\_and\_matplotlib***"
sudo pip -H install --upgrade scipy numpy matplotlib pynmea2

# Setup Service Files
echo "***Making\_File\_Directories***"
sudo mkdir /home/debian/Python
```
B.2. Configuration of BeagleBone Black for Buoy Operating System

```bash
sudo mkdir /home/debian/Python/Output

echo "***Establishing Temporary_Buoy_Main.py for service allocation***"
sudo touch /home/debian/Python/Buoy_Main.py
printf '#!/usr/bin/python

"Temporary_Buoy_Main
Used ONLY to establish target file for service
"
print("Temporary_Buoy_Main.py created")
> /home/debian/Python/Buoy_Main.py
sudo chmod +x /home/debian/Python/Buoy_Main.py

echo "***Establishing Service File for Boot Load***"
sudo touch /lib/systemd/system/Buoy.service
printf '[Unit]
Description=Buoy_Init_Service

[Service]
WorkingDirectory=/home/debian/Python/
ExecStart=/usr/bin/python_Buoy_Main.py
StandardOutput=null
Type=idle
Restart=on-failure
RestartSec=30

[Install]
WantedBy=multi-user.target
Alias=Buoy.service
> /lib/systemd/system/Buoy.service

echo "***Enabling Boot Service***"
sudo systemctl enable Buoy.service
sudo systemctl start Buoy.service
```
Appendix C

Matlab Codes For Effect of Connectivity

Listing C.1: Leader-follower Linear Consensus Protocol

```matlab
function WSnetwork(Nagents, kn)

% This function generate graph using Watts and Strogatz small work network
% method (where P=0 is normal ring lattice) and node switch. It also find the
% gain and cut-off frequency (normalised norm2 gain = 0.7071) for the
% state space system (A,B,C,D). probks is the probability of switching,
% dagent is the driving agent. This function allows input of different probks.
% It is for single or multiple input and evaluate all the point for output
% excluding the leaders. Notes: Node switch for P > 0 (0 <= P <= 1)

clear
close all
tic
% Add path for file access at sub-folder
addpath(genpath(fileparts(pwd))) %if file is not in ykkodes, eg subfolder.

%*******************Input Parameters*********************************
Nagents = 1000; % number of agents
kn = 5; % interaction radius for neighbors for metric

%*******************End of Input Parameters***************************

%*******************Parameters setting*******************************
ssize = 1000; % statistical size
prob = [0, 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1];
foldername = [’LCWS/WS_k’,num2str(kn)];
if exist(foldername,’dir’) ~= 7
disp(‘not exist’)
mkdir(foldername)
end

%*******************End of Parameters setting*************************

for iprob = 1:length(prob)
    fprintf(’Probability = %.4fn’, prob(iprob));
    concount = 0;
    disconcount = 0;
    for issize = 1:ssize
        % Generation of Graph (Network)
    end
end
```

% Programme written by YKK from SUTD EPD on 2017, feel free to use it for your
% own research.
Listing C.2: Vicsek Model - SPP

```matlab
function psi = VM_topo(Nagents, kn, eta)

%% This algorithm runs Vicsek Model using topological method
%% and also look for correlation function of the fluctuation and
%% susceptibility (ref Cavagna et al 2010 and Attanasi et al 2014).
%% Not tracking the previous (iteration = i - 1) agents’ connection.
%% Programme written by YKK from SUTD EPD on 2017, feel free to use it for your
%% own research.

tic

%***Add path for file access at sub-folder
addpath(genpath(fileparts(pwd))) % if file is not in ykkcodes, eg subfolder.

%%%%%%%%%%%%%%%%%Input Parameters%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nagents = 100; % number of agents
kn = 3; % interaction radius for neighbors for metric
eta = 0.05; % noise 0 to 1

% Generate Watts-Strogatz network and ensure is linked

gnetwork = WattsStrogatz(Nagents, kn, prob(iprob));

%% Calculate clustering coefficient, ave path length, and laplacian
adjmatrix = full(adjacency(gnetwork));
aveCC(iprob, issize) = clustCoeff(adjmatrix); % looking for clustering coefficient using MIT’s code
avecc(iprob, issize) = cluster_coeff(adjmatrix); % YK code
avePL(iprob, issize) = ave_path_length(adjmatrix); % looking for ave path length
Lev(iprob, issize) = Leig_value2(adjmatrix); % looking for laplacian eigenvalues

%% check for connectedness
if sum(sum(isinf(distances(gnetwork)),2))>0
    %disp('Disconnected ')
    disconcount = disconcount + 1;
else
    %disp('Connected ')
    concount = concount + 1;
end

if rem(issize, sssize/10) == 0
    fprintf(' Iteration %i\n', issize);
end
connected_prob(iprob, issize) = concount/issize;

%****************************Saving the workspace*******************************
fname = ['N', num2str(Nagents), 'ss', num2str(issize), 'k', num2str(kn), 'Allprob.mat'];
foldnfname = [foldername, '/', fname];
save(foldnfname);

%**************************toc**********************************************
toc

%semilogx(prob, mean(aveCC,2)/mean(aveCC(1,:)), '*')
```

Matlab Codes For Effect of Connectivity

```matlab
% Calculate clustering coefficient, ave path length, and laplacian
adjmatrix = full(adjacency(gnetwork));
aveCC(iprob, issize) = clustCoeff(adjmatrix); % looking for clustering coefficient using MIT’s code
avecc(iprob, issize) = cluster_coeff(adjmatrix); % YK code
avePL(iprob, issize) = ave_path_length(adjmatrix); % looking for ave path length
Lev(iprob, issize) = Leig_value2(adjmatrix); % looking for laplacian eigenvalues

% check for connectedness
if sum(sum(isinf(distances(gnetwork)),2))>0
    %disp('Disconnected ')
    disconcount = disconcount + 1;
else
    %disp('Connected ')
    concount = concount + 1;
end

if rem(issize, sssize/10) == 0
    fprintf(' Iteration %i\n', issize);
end
connected_prob(iprob, issize) = concount/issize;

%****************************Saving the workspace*******************************
fname = ['N', num2str(Nagents), 'ss', num2str(issize), 'k', num2str(kn), 'Allprob.mat'];
foldnfname = [foldername, '/', fname];
save(foldnfname);

%**************************toc**********************************************
toc

%semilogx(prob, mean(aveCC,2)/mean(aveCC(1,:)), '*')
```

Matlab Codes For Effect of Connectivity
fprintf('noise = %.4f\n', eta);

%****************************End of Input Parameters******************************

%************************Parameters setting***************************************
Density = 1;  % rho = Nagent/L^2;
L = sqrt(Nagents/Density);  % size of space
v = 0.04;  % speed
dt = 1;  % time interval between 2 updating of the directions and positions
qscale = 0.3;  % quiverscale
vis = 0;  % visualization 1 = show, any other not show
niter = 100000;  % number of iterations
pinterval = niter/10;

% Creating folder for saving workspace
foldername = ['VM_topo/VMsus_k', num2str(kn)];
if exist(foldername, 'dir') == 0
    disp('not exist')
    mkdir (foldername)
end

%****************************End of Parameters setting***************************

% randomly generate locations (x and y coordinate) and angles for agent
[xpos, ypos, theta] = gen_random_location(Nagents, L, v, qscale, vis);

%******************************Phase transition*************************************
for i = 1 : niter % run the Model for transient period
    % Phase transition
    psi(i) = sqrt((sum(cos(theta))).^2+(sum(sin(theta))).^2)/Nagents;
    % looking for the avetheta using topological method and track the rewiring
    [avetheta] = Topo_PBC(Nagents, xpos, ypos, theta, kn, L);
    % updating angle for theta base on nearest neighbor
    xi = 2*pi*(rand(Nagents,1)-0.5); % white noise, xi (-pi to pi)
    theta = avetheta + eta * xi; % updating angle of agent
    % updating position of Xi(t+1) = Xi(t) + vi(t) * delta t
    vxpos = v.*cos(theta); % forward update
    vypos = v.*sin(theta);
    xpos = xpos + vxpos.*dt;
    ypos = ypos + vypos.*dt;
    % periodic BC when position lesser than zero and greater than L
    [xpos, ypos] = PBC(xpos, ypos, L);
    if rem(i, pinterval) == 0
        fprintf('Iteration %i\n', i);
    end
end

%****************************End of Simulation*************************************
toc

%*******************************Saving the workspace********************************
fname = ['N\n', num2str(Nagents), '\i\n', num2str(niter), 'noise\n', num2str(eta), 'k\n', num2str(kn), \n']
Matlab Codes For Effect of Connectivity

Listing C.3: Connected Correlation Function

```matlab
function [totalcorr, count] = fluctuation (Nagents, L, xpos, ypos, vxpos, vypos, nbins)

% This algorithm determine ....

% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.

% This algorithm uses Attanasi.....

% Writing function for calculating fluctuation – correlation and
% susceptibility.

maxdist = L/sqrt(2);
bindist = nbins/(maxdist);

% looking for distance between ri and rj
xx = xpos' - xpos';
yy = ypos' - ypos';

% look for x-distance between ri and rj
%xx = repmat(xpos, 1, Nagents) - repmat(xpos', 1, Nagents, 1);
% look for y-distance between ri and rj
%yy = repmat(ypos, 1, Nagents) - repmat(ypos', 1, Nagents, 1);

% Periodic Boundary Condition – Distance between vi and vj
xx(xx > L/2) = xx(xx > L/2) - L;
xx(xx < -L/2) = xx(xx < -L/2) + L;
yy(yy > L/2) = yy(yy > L/2) - L;
yy(yy < -L/2) = yy(yy < -L/2) + L;
distr = sqrt(xx.^2 + yy.^2);

% Distance between agent ri and rj

% Fluctuation
avevxpos = mean(vxpos); % look for average velocity for x component;
avevypos = mean(vypos); % look for average velocity for y component;
vel4norm2 = sum((vxpos - avevxpos).^2 + (vypos - avevypos).^2) / Nagents; % value that normalize velocity fluctuation

totalcorr = zeros(nbins, 1);
count = zeros(nbins, 1);
for vi = 1:Nagents
  for vj = vi+1:Nagents
    bin = ceil(bindist * distr(vi, vj));
    count(bin) = count(bin) + 1;
    totalcorr(bin) = totalcorr(bin) + ((vxpos(vi) - avevxpos) * (vxpos(vj) - avevxpos) +
                                            (vypos(vi) - avevypos) * (vypos(vj) - avevypos));
  end
end
totalcorr = totalcorr / vel4norm2;
end
```

Listing C.4: Generation of Agents with Random Location

```matlab
function [xpos,ypos,theta]=gen_random_location(Nagent,L,v,qsacle,vis);

% This algorithm generate random location for nodes, could be use for any
% particles simulation, eg vicsek model.
% Input number of agent => Nagent, size of space => L and
% vis for visualization (default no visualization) vis = 1 for
% visualization

% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.

% randomly generate locations (x and y coordinate) and angles for agent
xpos = rand(Nagent,1)*L;
ypos = rand(Nagent,1)*L;
theta = pi*2*rand(Nagent,1); % from 0 to 2 pi
vxpos = v.*cos(theta);
vypos = v.*sin(theta);

if vis == 1 % plot initial position and direction if vis ==1
    figure
    axis([0 L 0 L])
    axis('square')
    hold on
    quiver(xpos,ypos,vxpos,vypos,qsacle,'r');
    plot(xpos,ypos,'r.');
    pause(0.1)
end
```

Listing C.5: Topological Interaction with Periodic Boundary Condition

```matlab
function [avetheta,adjmat]=TopoPBC(Nagents,xpos,ypos,theta,kn,L)

% This function look for the nearest neighbours of particle swarm using
% topological method, kn = number of nearest neighbours, eg kn=7 mean the 7
% nearest nodes of node i will be its neighbours.
% The function also account for periodic boudary condition while looking
% for the nearest neighbours.

% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.

% Looking for nearest neighbours using topological method
adjmat = zeros(Nagents);
for k = 1:Nagents
    %Find any distance that is more than L/2 or less than –L/2 for Periodic Boundary
    Condition
    [dx, dy] = PBC_L2(xpos,ypos,L,k);
    %Look for the nearest distance neighbours, depend on kn (number of nearest distance)
    distnn = dx.^2 + dy.^2; % maxi dist is L/sqrt(2)
    [~, tind]=sort(distnn);
    adjmat(k,tind(2:kn+1)) = 1;
nnb = theta(tind(1:kn+1));
    avetheta(k,1) = atan2(mean(sin(nnb)),mean(cos(nnb)));
end
end
```
Listing C.6: Maximun Distance in Periodic Boundary Condition

```matlab
function [dx, dy] = PBC_L2(xpos, ypos, L, k)

% This algorithm sets xpos and ypos for determining nearest distance in
% PBC. x distance between 2 points is L/2 and same goes for y. where xpos
% is the x-position and ypos is y-position of agents. L is the length of
% the box (of periodic boundary condition) and k is the number of
% connnections.

% Programme written by YKK from STUD EPD on 2018, feel free to use it for your
% own research.

% Periodic BC when position lesser than -L/2 and greater than L/2
dx   = xpos(k) - xpos;
dx(dx>L/2) = dx(dx>L/2) - L;
dx(dx<-L/2) = dx(dx<-L/2) + L;
dy   = ypos(k) - ypos;
dy(dy>L/2) = dy(dy>L/2) - L;
dy(dy<-L/2) = dy(dy<-L/2) + L;
end
```
Appendix D

Matlab Codes For Forced Switching

Listing D.1: Average Path Length

function APL = ave_path_length(adjmat)

% This algorithm look for average path length for static network.
% Calculating average path length by summing all the distance and divide
% by max number of link. adjmat is the adjacency matrix.

% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.

% for normalising Path length
n = length(adjmat);
if adjmat == transpose(adjmat)
    G = graph(adjmat); % undirected graph
else
    G = digraph(adjmat); % directed graph
end
dG = distances(G); % finding the distance between each node
APL = sum(sum(dG, 2)) / (n * (n - 1));
end

Listing D.2: Average Clustering Coefficient

function ave_cc = cluster_coeff(adjmat)

% Function takes a graph G (adjmat is adjacency matrix) as input and
% returns the average clustering coefficient, cc.
% This is done by the method described in:
% Giorgio Fagiolo 2007 clustering in complex directed networks (PRE)
% Using out-degree for digraph and is also for undirected graph
% cci = (A^2 * A^T / (degout * (degout - 1)))
% if degout = 1 cci is equivalent to 0

% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.

% n = length(adjmat);
deg_out = sum(adjmat, 2);
call = (diag((adjmat*adjmat*transpose(adjmat))/(deg_out.*(deg_out-1))));
if any(isnan(call))
call(isnan(call))=0;
end
ave_cc = mean(call);
end

Listing D.3: Second Smaller Eigenvalue

function Lev2 = Leig_value2(adjmat)
% Function takes a graph G as input and returns the laplacian eigenvalues,
% General form :
% L = D - A,
% where L is laplacian matrix, D is degree of node i and A is (adjmat)
% adjacency matrix, return the value of 2nd eigenvalue. Sum rows for in
% degree, i elements and sum columns for out degree, j elements. This code
% works for directed and undirected graph.
% Normalised form :
% undirected graph:  Lnorm = D^-(1/2)*(D - A)*D^-(1/2)
% ref Chung 1996 Spect
% directed graph:   Lnorm = D^-1*(D - A)
% ref Roland et al 2016
% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.
% L = degk - adjmat; % non normalize

degk = diag(sum(adjmat,2));

if adjmat == transpose(adjmat) % check whether A is directed or undirected ,
    disp('undirected')
    if ~isnan(degk^(1/2)) % check to see is the matrix disconnected
        Lnorm = degk^-(1/2)*(degk - adjmat)*degk^-1*(1/2); % normalized L undirected graph
        Lev = eig(Lnorm);
    else
        L = (degk - adjmat); % undirected graph and can't be normalised
        Lev = eig(L);
    end
else
    disp('directed')
    Lnorm = degk^-1*(degk - adjmat); % directed graph
    Lev = sort(eig(Lnorm));
end

Lev2 = Lev(2);

Listing D.4: Watts and Strogatz Network

function h = WattsStrogatz(N,K,beta)
% H = WattsStrogatz(N,K,beta) returns a Watts-Strogatz model graph
% (aka small world network) with N nodes, N*K edges, mean node degree 2*k
% rewiring probability beta.
% Probability of rewiring, beta = 0 is a ring lattice, and
% beta = 1 is a random graph.
% This code is obtain from
% Connect each node to its K next and previous neighbors. This constructs
% indices for a ring lattice.
s = repelem((1:N)',[1,K]);
t = s + repmat(1:K,N,1); 
return

% Rewire the target node of each edge with probability beta
for source=1:N
    switchEdge = rand(K, 1) < beta;
    newTargets = rand(N, 1);
    newTargets(source) = 0;
    newTargets(t==source) = 0;
    newTargets(t(source, switchEdge)) = 0;
    [~, ind] = sort(newTargets, 'descend');
    t(source, switchEdge) = ind(1:nnz(switchEdge));
end
end

Listing D.5: Local Extended Force Switching

function [tind] = FS_nn(kn, ks, iter, tind, prevadjmat)
% This function forced the node/agent to switch the number neighbour in each
% iteration specified by ks (degree switch), using topological method. It replaces the
% forced switch link by the NEAREST ONE DIRECTLY after kn, kn = number of nearest
% neighbours, eg if kn=7, the 8th nearest node of node i will be its neighbour.
% The function also account for periodic boundary condition while looking
% for the nearest neighbours.
% kn is out degree, ks is no. force switching, iter is the iteration,
% tind is new set of neighbour, and prevadjmat is the previous adjacency matrix.
% Programme written by YKK from STUD EPD on 2017, feel free to use it for your
% own research.

lnn = tind(2:kn+1);
pnn = find(squeeze((prevadjmat(iter,:))));
fdiff = setdiff(lnn,pnn);
rmolddiff = setdiff(pnn,lnn);
tind_temp = tind(kn+2:end);
for irmo = 1:length(rmolddiff)
tind_temp(tind_temp=rmolddiff(irmo)) =[];
end
lfdiff = length(fdiff);
if lfdiff < ks
    lnn = setdiff(lnn,fdiff);
    switchpos = randperm(length(lnn),ks-length(fdiff));
    for iswitch = 1:length(switchpos)
        tind(tind==lnn(switchpos(iswitch))) = tind_temp(iswitch);
    end
end

Listing D.6: Global Force Switching

function [tind] = FS_rand(kn, ks, iter, tind, prevadjmat)
% This function force the node/agent to switch the number neighbour in each
% iteration specified by ks (degree switch), using topological method. Replace the
% forced switch link by RANDOMLY PICKING a nodes in the space to link with.
% The function also account for periodic boundary condition while looking
% for the nearest neighbours.
% kn is out degree, ks is no. force switching, iter is the iteration,
% tind is new set of neighbour, and prevadjmat is the previous adjacency matrix.
%
% Programme written by YKK from STUD EPD on 2017, feel free to use it for your own research.

lnn = tind(2:kn+1);
pnn = find(squeeze((prevadjmat(iter,:))));
fdiff = setdiff(lnn,pnn);
rmolddiff = setdiff(pnn,lnn);
tind_temp = tind(kn+2:end);
for irmo = 1:length(rmolddiff)
    tind_temp(tind_temp==rmolddiff(irmo)) =[];
end
lfdiff = length(fdiff);
if lfdiff < ks
    lnn = setdiff(lnn,fdiff);
    switchpos = randperm(length(lnn),ks-length(fdiff));
    % randomly select a node from anywhere in the space to be switched
    switchposrand = randperm(length(tind_temp),length(switchpos));
    for iswitch = 1:length(switchpos)
        tind(tind==lnn(switchpos(iswitch))) = tind_temp(switchposrand(iswitch));
    end
end
Appendix E

Code for Postprocessing of Scalar Field Reconstruction

Listing E.1: Code for Postprocessing of Temperature Field reconstruction

```python
#!/usr/local/bin/python3
'''
MIT-SUTD 2015–2018
Authors: Yoke Kong Kuan
'''
import matplotlib.pyplot as plt
from matplotlib import rcParams, animation
from numpy import zeros, ones, array, average, size, reshape
from numpy import cumsum, sin, cos, pi, mgrid
from numpy import meshgrid, sqrt, exp, mean
from numpy import linspace, diff, ndarray, arange
from numpy import average, absolute, power, subtract
from numpy import loadtxt
import numpy as np

from Area_Check import area_check
from Google_Map import Google_Map
import time
import Hermite as herm

# main program

plotted = False
M, N = 12, 12

""
Read file from Matlab output xpos ypos and temperature, and
save it in array or list (not sure). These can be use to look
for temperature interpolation using hermite polynomials and function
""

dxpos = []
dypos = []
dtemp = []
t = -1

with open("lonlatTempAg0615TQ.txt", 'r') as f:
    xytempdata = f.read()
    xytemplist = xytempdata.split('\n')

for line in xytemplist:
    xytsplit = line.split('t')
```
if len(xytsplit) is 2:
    t += 1
if len(xytsplit) is 3:
    # print("len 3")
    tempx, tempy, tempT = xytsplit[:3]
dxpos.append(float(tempx))
dypos.append(float(tempy))
dtemp.append(float(tempT))
t += 1

dxpos = reshape(dxpos, (t, int(len(dxpos)/t)))
dypos = reshape(dypos, (t, int(len(dypos)/t)))
dtemp = reshape(dtemp, (t, int(len(dtemp)/t)))

""
From here onwards the code is setting up data and interpolate
temperature field for the area.
""

# for lat lon t 0615TQ
miny, maxy = 1.3495, 1.3535
minx, maxx = 103.9205, 103.9253
xi = linspace(minx, maxx)
yi = linspace(miny, maxy)
xx, yy = meshgrid(xi, yi)

for i in range(t):  # t is the max time step
    if i is 0:
        x = array(dxpos[i])
        y = array(dypos[i])
        T = array(dtemp[i] - 273.15)
        print("i am zero")
    else:
        print("I am not zero")
        addx = array(dxpos[i])
        addy = array(dypos[i])
        addT = array(dtemp[i] - 273.15)
        x = np.append(x, addx)
        y = np.append(y, addy)
        T = np.append(T, addT)

    print("time=", i)
    print(x.size)
    print(y.size)

c_mn, H_kw = herm.Spmfit_2d(T, x, y, (M, N))  # compute coefficients
T_fit = herm.eval_2d(xx, yy, c_mn.reshape(M, N), **H_kw)  # curve fit
u, v = herm.grad(xx, yy, c_mn, **H_kw)

# Set font for latex
plt.rcParams['text.latex preamble'] = ['\usepackage{DejaVuSans}']
params = {'text.usetex': True,
          'font.size': 11,
          'font.family': 'DejaVu_Sans',
          'text.latex.unicode': True,
          }
plt.rcParams.update(params)
fig = plt.figure(figsize=(8, 7))
# plt.figure()
lon_range = array([minx, maxx])
lat_range = array([miny, maxy])
center = array([mean(lat_range), mean(lon_range)])
gm = 0  # 0 = no, 1 = yes

plot_kw = {'vmin': T.min(), 'vmax': T.max(),
            'cmap': 'hot', 'alpha': 1}
CSF = plt.contourf(xx, yy, T.fit, 10, **plot_kw)
plt.clf()
ax = plt.axes(xlim=lon_range, ylim=lat_range, aspect=1.00)
ax.tick_params(labelsize=16, direction='in ')
ax.set_xlabel('Longitude ', fontsize=20)
ax.set_ylabel('Latitude ', fontsize=22)

if g is 1:
    try:
        GMap = Google_Map(center, size=640, scale=2,
                          lat_rng=lat_range, lon_rng=lon_range)
        gmap = GMap.Map
        ext = GMap.extents
        plt.imshow(plt.imread(gmap), extent=ext, zorder=0, alpha=0.3)
    except:
        print('Cannot download Map, sian\_liao ')
        area, area_axis = area_check(lat_range, lon_range)
        if area:
            img = plt.imread(area)
            plt.imshow(img, zorder=0, extent=area_axis, alpha=0.3)

CB = plt.colorbar(CSF, shrink=0.8, extend='both')
CS = plt.contour(xx, yy, T.fit, 10, **plot_kw)
plt.scatter(addx, addy, c=(55/255, 126/255, 184/255), s=10, zorder=1, alpha=1)
plt.savefig('TempCont56.pdf', dpi=1200, bbox_inches='tight', transparent='True')

Listing E.2: Code for SMI, Hermite Function and Curve Fitting
def _asarray(x):
    """If the target variable is an array, return it directly. If it is not an array, make it so and return it."
    if isinstance(x, ndarray):
        return x
    return array(x, dtype=float)

def _norm(x):
    """Returns normalized values to a boundary of [-10 10] in order to obtain good Hermite fit relation"
    x = _asarray(x)
    a, b = x.min(), x.max()
    xn = x - (a+b)/2.
    xn /= (b-a)/2.
    return xn

def _poly(N, xi):
    """Evaluates Hermite Polynomial Function of order N"
    Inputs: N, xi
    N: Order of Hermite Polynomial
    xi: Evaluated location
    Output: Yn
    Yn: Hermite Polynomial Function
    """
    xi = _asarray(xi)
    xi = xi.flatten()
    if N == 1:
        return pi**(-1./4.)*exp((-1./2.)*xi**2)
    Yn = ones((N, xi.size))
    Yn[0] = pi**(-1./4.)*exp((-1./2.)*xi**2)
    Yn[1] = sqrt(2.)*pi**(-1./4.)*xi*exp((-1./2.)*xi**2)
    if N == 2:
        return Yn
    for n in range(2, N):
        Yn[n] = xi*sqrt(2./n)*Yn[n-1] - sqrt((n-1)/n)*Yn[n-2]
    return Yn

def eval_2d(x, y, c_mn, **kwargs):
Evaluate 2-D Hermite series based on coefficient
\( C_{mn} \) on linspace \( x \) and \( y \)

Inputs: \( x, y, C_{mn} \)
Outputs: evaluated value on grid \( x, y \)

kwargs:
- \( x \): shift in \( x \)
- \( y \): shift in \( y \)
- \( L_x \): scaling in \( x \)
- \( L_y \): scaling in \( y \)
- \( ff \): far field shift

```python
x = asarray(x)
y = asarray(y)
c_mn = array(c_mn, dtype=float)
shp = x.shape
x = x.flatten()
y = y.flatten()

x = kwargs.get('_x', 0.)
y = kwargs.get('_y', 0.)
L_x = kwargs.get('_L_x', 1.)
L_y = kwargs.get('_L_y', 1.)
ff = kwargs.get('_ff', 0.)

M, N = c_mn.shape
Y_m = poly(M, (x-x)/L_x)
Y_n = poly(N, (y-y)/L_y)
f = dot(c_mn, Y_n)*Y_m
f = sum(f, axis=0) + ff
return f.reshape(shp)
```

def _der_c(c):
    Evaluate Hermite derivative based solely on coefficients of Hermite Function Polynomials

Inputs: \( C_n \)
Outputs: \( C_n \_dx \)

```python
c = _asarray(c)
hrm_der = zeros_like(c)
N = c.size

if N == 1:
    return hrm_der

n = arange(1, N+1)
a = sqrt((n+1.)/2.)*c
b = sqrt(n/2.)*c
hrm_der[0] = sqrt(1./2.)*c[1]
```
hrm_der[1:N-1] = a[2:] - b[:N-2]
hrm_der[-1] = -sqrt((N-1)/2.)*c[N-2]
return hrm_der

def grad(x, y, c, **kwargs):
    
    Evaluates the Gradient of a Hermite series based on Hermite Coefficient Derivatives on grid x, y
    
    Inputs: x, y, C
    Output: u, v
    
    kwargs are passed through for the evaluation phase
    
    c = asarray(c)
    M, N = c.shape
    
    kwargs.pop('_ff', 0.)
    c_dx = zeros_like(c)
    c_dy = zeros_like(c)
    
    for i in range(N):
        c_dx[:, i] = _der.c(c[:, i])
    for i in range(M):
        c_dy[i, :] = _der.c(c[i, :])
    
    u = eval_2d(x, y, c_dx, **kwargs)
    v = eval_2d(x, y, c_dy, **kwargs)
    return u, v

def Hess(x, y, c, **kwargs):
    
    Evaluates the Second Partial derivative at a point for a Hermite series based on Hermite Coefficient Derivatives on grid x, y
    
    Inputs: x, y, C
    Evaluates: | dxx  dxy |
    det | |
    | dyx  dyy |
    
    Output: 1 = local minimum
    0 = local maximum or saddle
    
    Used to determine whether to go with or against the gradient in order to find the local extrema
    
    kwargs are passed through for the evaluation phase
    
    c = asarray(c)
    M, N = c.shape
    
    kwargs.pop('_ff', 0.)
cx = array([_*der_c(1[0]) for l in c[:, None].T])
cxx = array([_*der_c(_der_c(1[0])) for l in c[:, None].T])
cyy = array([_*der_c(_der_c(1[0])) for l in c[:, None]])
cxy = array([_*der_c(1[0]) for l in cx.T[:, None]])

fxx = eval_2d(x, y, cxx.T, **kwargs)
fyy = eval_2d(x, y, cyy, **kwargs)
fxy = eval_2d(x, y, cxy, **kwargs)

D = fxx*fyy - fxy**2

H = zeros_like(D)

""" If testing a singular location, don't attempt
to access elements in a 0-d array. """

if not H.ndim:
    if D > 0. and fxx > 0.:
        return 1.
    return 0.

mn = (D > 0.) & (fxx > 0.)
mx = (D > 0.) & (fxx < 0.)

H[mn] = 1.

return H

def fit_2d(fn, shp, **kwargs):

    """ Using Orthonormal properties, this function calculates the inner product of the target function against the Hermite Functions in order to return the coefficients which represent the best fit 2D series. """

c = int_*fn(x,y)*Ym(x)*Yn(y)dxdy

M, N = shp

ax = kwargs.get('ax', -5.)
bx = kwargs.get('bx', 5.)
ay = kwargs.get('ay', -5.)
by = kwargs.get('by', 5.)
is_fn = isfunction(fn)

x_num, y_num = (50, 50) if is_fn else fn.shape

x = linspace(ax, bx, num=x_num)
y = linspace(ay, by, num=y_num)

xx, yy = meshgrid(x, y)

L_x = kwargs.get('L_x', (x.max()-x.min())/10.)
L_y = kwargs.get('L_y', (y.max()-y.min())/10.)
Code for Postprocessing of Scalar Field Reconstruction

```python
fvals = fn(xx, yy) if is_fn else fn
fvals = fvals.flatten()

_x = kwgars.get('x', (x.min() + x.max()) / 2.)
_y = kwgars.get('y', (y.min() + y.max()) / 2.)

fvals = fvals.flatten()

ff = kwargs.get('ff', fvals.mean())
x = kwgars.get('x', (x.min() + x.max()) / 2.)
y = kwgars.get('y', (y.min() + y.max()) / 2.)

Hx = _poly(M, (xx.flatten() - _x) / Lx)
Hy = _poly(N, (yy.flatten() - _y) / Ly)

c = (Hx[:, None] * Hy * fvals).reshape(M, N, *xx.shape)
c = trapz(trapz(c, (x - _x) / Lx, axis=2), (y - _y) / Ly, axis=2)

fit_kw = {'x': _x, 'y': _y, 'Lx': Lx, 'Ly': Ly, 'ff': ff}

return c, fit_kw

def Sp_fit_2d(dat, xs, ys, shp):
    """Using Orthonormal properties, this function calculates the inner product of the target function against the Hermite Functions given as parameters in order to return the coefficients which represent best fit 2D series."
    M, N = shp
    pts = array([xs, ys], dtype=float).T
dat = asarray(dat)
x = asarray(xs)
y = asarray(ys)

    num = pts.shape[0] if pts.shape[0] < 50 else 50
    rx = linspace(xs.min(), xs.max(), num=num)
    ry = linspace(ys.min(), ys.max(), num=num)
    rxx, ryy = meshgrid(rx, ry)

    grid_kw = {'method': 'cubic', 'fill_value': dat.mean()}
sGd = griddata(pts, dat, (rxx, ryy), **grid_kw)

    kw = {'ax': rx.min(), 'bx': rx.max(),
          'ay': ry.min(), 'by': ry.max()}
    return fit_2d(sGd, (M, N), **kw)

def MC_fit_2d(dat, x, y, shp, **kwgars):
    """Using Orthonormal properties, this function calculates the inner product of the target function against the Hermite Functions given a Monte Carlo style integration in order to return coefficients which represent best fit 2D series."
```

Listing E.3: Sample Data of Temperature and Location

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Temperature</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.387003</td>
<td>-9.647688</td>
<td>305.596000</td>
</tr>
<tr>
<td>26.152799</td>
<td>23.562627</td>
<td>304.634000</td>
</tr>
<tr>
<td>-35.993251</td>
<td>-8.905557</td>
<td>304.110000</td>
</tr>
<tr>
<td>9.088599</td>
<td>0.742130</td>
<td>304.380000</td>
</tr>
<tr>
<td>2.596806</td>
<td>-2.226390</td>
<td>304.589000</td>
</tr>
<tr>
<td>1.112885</td>
<td>-6.493637</td>
<td>304.573000</td>
</tr>
<tr>
<td>-1.669328</td>
<td>-27.273275</td>
<td>304.954000</td>
</tr>
<tr>
<td>-12.798180</td>
<td>-13.172806</td>
<td>305.269000</td>
</tr>
<tr>
<td>-40.805714</td>
<td>5.937043</td>
<td>304.603000</td>
</tr>
<tr>
<td>-24.483477</td>
<td>-37.106495</td>
<td>305.012000</td>
</tr>
<tr>
<td>15.765910</td>
<td>8.348962</td>
<td>304.535000</td>
</tr>
<tr>
<td>-23.741515</td>
<td>-18.182182</td>
<td>304.672000</td>
</tr>
<tr>
<td>-35.993253</td>
<td>-23.377091</td>
<td>304.494000</td>
</tr>
<tr>
<td>25.039920</td>
<td>-35.065638</td>
<td>304.466000</td>
</tr>
<tr>
<td>28.193164</td>
<td>10.389821</td>
<td>304.547200</td>
</tr>
<tr>
<td>-13.169105</td>
<td>-26.160080</td>
<td>304.426000</td>
</tr>
<tr>
<td>-18.733529</td>
<td>15.584729</td>
<td>304.482000</td>
</tr>
<tr>
<td>37.652686</td>
<td>24.119226</td>
<td>304.565000</td>
</tr>
<tr>
<td>23.185183</td>
<td>-11.503013</td>
<td>305.442000</td>
</tr>
<tr>
<td>-2.782213</td>
<td>-4.638313</td>
<td>304.596000</td>
</tr>
<tr>
<td>5.007984</td>
<td>-17.996650</td>
<td>304.717000</td>
</tr>
<tr>
<td>-10.572408</td>
<td>24.119224</td>
<td>305.634000</td>
</tr>
<tr>
<td>15.951429</td>
<td>-17.996650</td>
<td>304.334000</td>
</tr>
<tr>
<td>1.298403</td>
<td>36.364367</td>
<td>304.262000</td>
</tr>
<tr>
<td>-6.306275</td>
<td>-2.968519</td>
<td>304.306000</td>
</tr>
<tr>
<td>27.080280</td>
<td>-0.371063</td>
<td>304.549000</td>
</tr>
<tr>
<td>8.717674</td>
<td>32.468185</td>
<td>305.093000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>29</td>
<td>4.637059</td>
<td>7.977897</td>
</tr>
<tr>
<td>30</td>
<td>-20.402858</td>
<td>-5.565974</td>
</tr>
<tr>
<td>31</td>
<td>-27.265687</td>
<td>1.669794</td>
</tr>
<tr>
<td>32</td>
<td>-25.596356</td>
<td>27.644342</td>
</tr>
<tr>
<td>33</td>
<td>-12.241735</td>
<td>12.801742</td>
</tr>
<tr>
<td>34</td>
<td>-9.088600</td>
<td>-28.386470</td>
</tr>
<tr>
<td>35</td>
<td>-31.902632</td>
<td>20.402858</td>
</tr>
<tr>
<td>36</td>
<td>-27.265684</td>
<td>25.232420</td>
</tr>
<tr>
<td>37</td>
<td>-34.870366</td>
<td>7.792364</td>
</tr>
<tr>
<td>38</td>
<td>9.088599</td>
<td>2.782987</td>
</tr>
<tr>
<td>39</td>
<td>2.967731</td>
<td>2.040857</td>
</tr>
<tr>
<td>40</td>
<td>1.669328</td>
<td>6.864701</td>
</tr>
<tr>
<td>41</td>
<td>-0.370925</td>
<td>-24.119222</td>
</tr>
<tr>
<td>42</td>
<td>-11.314259</td>
<td>-10.389819</td>
</tr>
<tr>
<td>43</td>
<td>-40.249272</td>
<td>8.64705</td>
</tr>
<tr>
<td>44</td>
<td>-23.370592</td>
<td>-35.62236</td>
</tr>
<tr>
<td>45</td>
<td>17.064313</td>
<td>10.204288</td>
</tr>
<tr>
<td>46</td>
<td>-22.443111</td>
<td>-16.141325</td>
</tr>
<tr>
<td>47</td>
<td>-35.055886</td>
<td>-22.263896</td>
</tr>
<tr>
<td>48</td>
<td>25.410955</td>
<td>-33.395846</td>
</tr>
<tr>
<td>49</td>
<td>29.306049</td>
<td>12.059614</td>
</tr>
<tr>
<td>50</td>
<td>-12.056220</td>
<td>-24.119222</td>
</tr>
<tr>
<td>51</td>
<td>-17.806162</td>
<td>16.697924</td>
</tr>
<tr>
<td>52</td>
<td>38.209128</td>
<td>24.304759</td>
</tr>
<tr>
<td>53</td>
<td>24.112550</td>
<td>9.833220</td>
</tr>
<tr>
<td>54</td>
<td>10.015966</td>
<td>3.154053</td>
</tr>
<tr>
<td>55</td>
<td>3.895098</td>
<td>1.855324</td>
</tr>
<tr>
<td>56</td>
<td>4.266134</td>
<td>6.308104</td>
</tr>
<tr>
<td>57</td>
<td>0.741961</td>
<td>22.078365</td>
</tr>
<tr>
<td>58</td>
<td>10.201374</td>
<td>8.534493</td>
</tr>
<tr>
<td>59</td>
<td>18.548012</td>
<td>3.339584</td>
</tr>
<tr>
<td>60</td>
<td>26.338208</td>
<td>37.106498</td>
</tr>
<tr>
<td>61</td>
<td>24.668878</td>
<td>29.499666</td>
</tr>
<tr>
<td>62</td>
<td>6.677311</td>
<td>14.471534</td>
</tr>
<tr>
<td>63</td>
<td>11.128850</td>
<td>10.204288</td>
</tr>
<tr>
<td>64</td>
<td>31.160783</td>
<td>26.160080</td>
</tr>
<tr>
<td>65</td>
<td>14.281989</td>
<td>15.955796</td>
</tr>
<tr>
<td>66</td>
<td>3.338655</td>
<td>39.518419</td>
</tr>
<tr>
<td>67</td>
<td>28.564090</td>
<td>1.113196</td>
</tr>
<tr>
<td>68</td>
<td>10.015965</td>
<td>34.880108</td>
</tr>
<tr>
<td>69</td>
<td>5.007983</td>
<td>9.647690</td>
</tr>
<tr>
<td>70</td>
<td>-18.548012</td>
<td>-3.339584</td>
</tr>
<tr>
<td>71</td>
<td>-26.338208</td>
<td>3.710652</td>
</tr>
<tr>
<td>72</td>
<td>-24.668878</td>
<td>29.499666</td>
</tr>
<tr>
<td>73</td>
<td>-6.677311</td>
<td>14.471534</td>
</tr>
<tr>
<td>74</td>
<td>-11.128850</td>
<td>37.106498</td>
</tr>
<tr>
<td>75</td>
<td>10.387003</td>
<td>-26.160080</td>
</tr>
<tr>
<td>76</td>
<td>-31.160783</td>
<td>15.955796</td>
</tr>
<tr>
<td>77</td>
<td>-14.281989</td>
<td>4.638313</td>
</tr>
<tr>
<td>78</td>
<td>17.435238</td>
<td>1.113195</td>
</tr>
<tr>
<td>79</td>
<td>12.798180</td>
<td>-5.937039</td>
</tr>
<tr>
<td>80</td>
<td>28.378569</td>
<td>27.273277</td>
</tr>
<tr>
<td>81</td>
<td>-33.015520</td>
<td>-6.879167</td>
</tr>
<tr>
<td>82</td>
<td>10.015966</td>
<td>3.154053</td>
</tr>
<tr>
<td>83</td>
<td>3.895098</td>
<td>-1.855324</td>
</tr>
<tr>
<td>84</td>
<td>4.266134</td>
<td>-6.308104</td>
</tr>
<tr>
<td>85</td>
<td>0.741961</td>
<td>-22.078365</td>
</tr>
<tr>
<td>86</td>
<td>-10.201374</td>
<td>-8.534493</td>
</tr>
<tr>
<td>87</td>
<td>-39.507422</td>
<td>7.606836</td>
</tr>
<tr>
<td>88</td>
<td>-21.515746</td>
<td>-33.766911</td>
</tr>
<tr>
<td>89</td>
<td>18.177198</td>
<td>12.987275</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>89</td>
<td>-20.588377</td>
<td>-14.842598</td>
</tr>
<tr>
<td>90</td>
<td>-33.943000</td>
<td>-21.150701</td>
</tr>
<tr>
<td>91</td>
<td>25.967398</td>
<td>-31.911585</td>
</tr>
<tr>
<td>92</td>
<td>30.975377</td>
<td>13.358341</td>
</tr>
<tr>
<td>93</td>
<td>-10.386892</td>
<td>-22.449430</td>
</tr>
<tr>
<td>94</td>
<td>-17.064201</td>
<td>17.811119</td>
</tr>
<tr>
<td>95</td>
<td>39.136495</td>
<td>24.90291</td>
</tr>
<tr>
<td>96</td>
<td>25.225435</td>
<td>-8.534492</td>
</tr>
<tr>
<td>97</td>
<td>30.975377</td>
<td>13.358341</td>
</tr>
<tr>
<td>98</td>
<td>-10.386892</td>
<td>-22.449430</td>
</tr>
<tr>
<td>99</td>
<td>-17.064201</td>
<td>17.811119</td>
</tr>
<tr>
<td>100</td>
<td>39.136495</td>
<td>24.90291</td>
</tr>
<tr>
<td>101</td>
<td>25.225435</td>
<td>-8.534492</td>
</tr>
<tr>
<td>102</td>
<td>30.975377</td>
<td>13.358341</td>
</tr>
<tr>
<td>103</td>
<td>-10.386892</td>
<td>-22.449430</td>
</tr>
<tr>
<td>104</td>
<td>-17.064201</td>
<td>17.811119</td>
</tr>
<tr>
<td>105</td>
<td>39.136495</td>
<td>24.90291</td>
</tr>
<tr>
<td>106</td>
<td>25.225435</td>
<td>-8.534492</td>
</tr>
<tr>
<td>107</td>
<td>30.975377</td>
<td>13.358341</td>
</tr>
<tr>
<td>108</td>
<td>-10.386892</td>
<td>-22.449430</td>
</tr>
<tr>
<td>109</td>
<td>-17.064201</td>
<td>17.811119</td>
</tr>
<tr>
<td>110</td>
<td>39.136495</td>
<td>24.90291</td>
</tr>
<tr>
<td>111</td>
<td>25.225435</td>
<td>-8.534492</td>
</tr>
<tr>
<td>112</td>
<td>30.975377</td>
<td>13.358341</td>
</tr>
<tr>
<td>113</td>
<td>-10.386892</td>
<td>-22.449430</td>
</tr>
<tr>
<td>114</td>
<td>-17.064201</td>
<td>17.811119</td>
</tr>
<tr>
<td>115</td>
<td>Time(sec)=</td>
<td>30</td>
</tr>
<tr>
<td>116</td>
<td>13.725658</td>
<td>-4.638311</td>
</tr>
<tr>
<td>117</td>
<td>29.676972</td>
<td>28.757537</td>
</tr>
<tr>
<td>118</td>
<td>-30.975267</td>
<td>-5.937037</td>
</tr>
<tr>
<td>119</td>
<td>10.572409</td>
<td>4.081715</td>
</tr>
<tr>
<td>120</td>
<td>4.266134</td>
<td>-0.371065</td>
</tr>
<tr>
<td>121</td>
<td>5.749944</td>
<td>-6.308104</td>
</tr>
<tr>
<td>122</td>
<td>2.040364</td>
<td>-20.965170</td>
</tr>
<tr>
<td>123</td>
<td>-8.903082</td>
<td>-7.421298</td>
</tr>
<tr>
<td>124</td>
<td>-38.579944</td>
<td>8.163433</td>
</tr>
<tr>
<td>125</td>
<td>-19.660899</td>
<td>-32.097119</td>
</tr>
<tr>
<td>126</td>
<td>19.475489</td>
<td>14.842600</td>
</tr>
<tr>
<td>127</td>
<td>-18.733531</td>
<td>-13.914936</td>
</tr>
<tr>
<td>128</td>
<td>-32.644597</td>
<td>-20.037506</td>
</tr>
<tr>
<td>129</td>
<td>26.709247</td>
<td>-30.798390</td>
</tr>
<tr>
<td>130</td>
<td>32.830111</td>
<td>14.100471</td>
</tr>
<tr>
<td>131</td>
<td>-8.903082</td>
<td>-20.591416</td>
</tr>
<tr>
<td>132</td>
<td>-16.136834</td>
<td>18.367716</td>
</tr>
<tr>
<td>133</td>
<td>40.434898</td>
<td>24.861357</td>
</tr>
<tr>
<td>134</td>
<td>26.523838</td>
<td>-7.606830</td>
</tr>
<tr>
<td>135</td>
<td>5.564433</td>
<td>7.606833</td>
</tr>
<tr>
<td>136</td>
<td>8.532157</td>
<td>-12.987273</td>
</tr>
<tr>
<td>137</td>
<td>-7.790195</td>
<td>28.572003</td>
</tr>
<tr>
<td>138</td>
<td>19.104567</td>
<td>-14.657065</td>
</tr>
<tr>
<td>139</td>
<td>5.379018</td>
<td>43.785667</td>
</tr>
<tr>
<td>140</td>
<td>-2.596695</td>
<td>0.742130</td>
</tr>
<tr>
<td>141</td>
<td>32.459188</td>
<td>3.39587</td>
</tr>
<tr>
<td>142</td>
<td>14.096581</td>
<td>39.332888</td>
</tr>
<tr>
<td>143</td>
<td>7.419271</td>
<td>11.688547</td>
</tr>
<tr>
<td>144</td>
<td>-15.580392</td>
<td>-1.113194</td>
</tr>
<tr>
<td>145</td>
<td>-23.741513</td>
<td>7.050236</td>
</tr>
<tr>
<td>146</td>
<td>-22.628625</td>
<td>31.726056</td>
</tr>
<tr>
<td>147</td>
<td>-3.524062</td>
<td>17.811119</td>
</tr>
<tr>
<td>148</td>
<td>-8.717562</td>
<td>39.703952</td>
</tr>
</tbody>
</table>
Code for Postprocessing of Scalar Field Reconstruction

149 13.169217  -24.304755  305.046000
150  -29.491455  18.533250  304.421000
151     -8.903081  9.647690  304.732000
152  20.402969   3.710651  305.009000
153  Time(sec)=  40
154  15.580393   -4.452779  305.646000
155  31.531817   29.685200  304.627000
156  -29.305940  -5.565973  304.123000
157  12.241737   4.452780  304.384000
158  4.266134   -0.185532  304.565000
159  29.305940  -5.751506  304.579000
160  17.620647   -6.308104  305.271000
161  -37.281652   7.421302  304.603000
162  -17.620647  -30.427272  305.015000
163  21.330335  16.141327  304.539000
164  -17.620646  -13.358338  304.667000
165  -30.975269  -19.295376  304.500000
166  27.451208  -30.241793  304.477000
167  35.241399   14.100472  304.104000
168  -7.790197  -19.109845  304.409000
169  -15.209355  18.924314  304.492000
170  42.660669   24.490291  304.599000
171  27.822130  -7.235765  305.453000
172  1.669328    7.977897  304.615000
173  9.645042  -12.801741  304.729000
174  -6.491792   29.128601  305.627000
175  20.588377  -13.914936  304.332000
176  6.306385   44.713329  304.193000
177  -1.298292  -0.556597  304.344000
178  34.684958   2.968522  304.546000
179  16.322350   40.446083  305.079000
180  9.088599   12.059612  304.694000
181  -12.983586   0.000000  304.984000
182  -21.886667   7.606833  304.620000
183  -21.701258   32.097121  304.355000
184  -1.298292   17.996651  304.367000
185  -7.790195   40.075017  305.202000
186  14.096584  -23.936930  305.058000
187  -28.749494   19.295381  304.427000
188  -8.161121   9.462157  304.735000
189  22.443221   4.267249  305.000000
190  Time(sec)=  50
191  17.435238  -4.638311  305.652000
192  33.386551   29.499667  304.625000
193  -26.709245  -6.122571  304.117000
194  14.282100   3.339585  304.397000
195  5.379019   -1.113195  304.552000
196  7.790196   -6.308104  304.583000
197  4.266134  -22.263898  304.957000
198  -5.935351  -5.937039  305.267000
199  -35.983249   5.380445  304.593000
200  -15.951319  -29.499665  305.009000
201  24.112548  15.584730  304.541000
202  -15.209357  -13.729403  304.672000
203  -28.935016   -19.109844  304.498000
204  28.378575  -29.870728  304.484000
205  37.281651  12.987277  304.109000
206  -5.935351  -18.553248  304.405000
207  -13.725546  17.625587  304.489000
<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.104566</td>
<td></td>
</tr>
<tr>
<td>34.499437</td>
<td></td>
</tr>
<tr>
<td>-24.854399</td>
<td></td>
</tr>
<tr>
<td>15.580392</td>
<td></td>
</tr>
<tr>
<td>6.120868</td>
<td></td>
</tr>
<tr>
<td>8.532157</td>
<td></td>
</tr>
<tr>
<td>5.007984</td>
<td></td>
</tr>
<tr>
<td>-5.564426</td>
<td></td>
</tr>
<tr>
<td>-35.426807</td>
<td></td>
</tr>
<tr>
<td>-14.281991</td>
<td></td>
</tr>
<tr>
<td>25.225433</td>
<td></td>
</tr>
<tr>
<td>-14.096472</td>
<td></td>
</tr>
<tr>
<td>-27.451095</td>
<td></td>
</tr>
<tr>
<td>29.120536</td>
<td></td>
</tr>
<tr>
<td>38.580055</td>
<td></td>
</tr>
<tr>
<td>-4.636948</td>
<td></td>
</tr>
<tr>
<td>-13.169103</td>
<td></td>
</tr>
<tr>
<td>45.257364</td>
<td></td>
</tr>
<tr>
<td>31.346304</td>
<td></td>
</tr>
<tr>
<td>4.266134</td>
<td></td>
</tr>
<tr>
<td>12.056331</td>
<td></td>
</tr>
<tr>
<td>-6.120868</td>
<td></td>
</tr>
<tr>
<td>22.628741</td>
<td></td>
</tr>
<tr>
<td>7.233752</td>
<td></td>
</tr>
<tr>
<td>0.741961</td>
<td></td>
</tr>
<tr>
<td>37.467171</td>
<td></td>
</tr>
<tr>
<td>19.104563</td>
<td></td>
</tr>
<tr>
<td>12.056330</td>
<td></td>
</tr>
<tr>
<td>-12.241737</td>
<td></td>
</tr>
<tr>
<td>19.289972</td>
<td></td>
</tr>
<tr>
<td>-20.031931</td>
<td></td>
</tr>
<tr>
<td>0.927478</td>
<td></td>
</tr>
<tr>
<td>-6.677310</td>
<td></td>
</tr>
<tr>
<td>16.136837</td>
<td></td>
</tr>
<tr>
<td>-25.967282</td>
<td></td>
</tr>
<tr>
<td>-4.636947</td>
<td></td>
</tr>
<tr>
<td>25.039916</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>269</td>
<td>33.942994</td>
</tr>
<tr>
<td>270</td>
<td>-24.483475</td>
</tr>
<tr>
<td>271</td>
<td>14.653025</td>
</tr>
<tr>
<td>272</td>
<td>6.120869</td>
</tr>
<tr>
<td>273</td>
<td>8.161232</td>
</tr>
<tr>
<td>274</td>
<td>4.822577</td>
</tr>
<tr>
<td>275</td>
<td>-5.749833</td>
</tr>
<tr>
<td>276</td>
<td>-35.241289</td>
</tr>
<tr>
<td>277</td>
<td>-13.540030</td>
</tr>
<tr>
<td>278</td>
<td>24.668991</td>
</tr>
<tr>
<td>279</td>
<td>14.096472</td>
</tr>
<tr>
<td>280</td>
<td>27.080171</td>
</tr>
<tr>
<td>281</td>
<td>28.564093</td>
</tr>
<tr>
<td>282</td>
<td>38.209130</td>
</tr>
<tr>
<td>283</td>
<td>-5.564426</td>
</tr>
<tr>
<td>284</td>
<td>13.169103</td>
</tr>
<tr>
<td>285</td>
<td>45.813807</td>
</tr>
<tr>
<td>286</td>
<td>31.717229</td>
</tr>
<tr>
<td>287</td>
<td>3.153249</td>
</tr>
<tr>
<td>288</td>
<td>12.241738</td>
</tr>
<tr>
<td>289</td>
<td>-7.604678</td>
</tr>
<tr>
<td>290</td>
<td>22.999666</td>
</tr>
<tr>
<td>291</td>
<td>6.306385</td>
</tr>
<tr>
<td>292</td>
<td>1.298403</td>
</tr>
<tr>
<td>293</td>
<td>37.652689</td>
</tr>
<tr>
<td>294</td>
<td>18.733639</td>
</tr>
<tr>
<td>295</td>
<td>10.943445</td>
</tr>
<tr>
<td>296</td>
<td>-13.540029</td>
</tr>
<tr>
<td>297</td>
<td>-19.289972</td>
</tr>
<tr>
<td>298</td>
<td>-20.402856</td>
</tr>
<tr>
<td>299</td>
<td>-0.741849</td>
</tr>
<tr>
<td>300</td>
<td>-6.491905</td>
</tr>
<tr>
<td>301</td>
<td>3.153249</td>
</tr>
<tr>
<td>302</td>
<td>-25.225322</td>
</tr>
<tr>
<td>303</td>
<td>-4.822465</td>
</tr>
<tr>
<td>304</td>
<td>25.039916</td>
</tr>
<tr>
<td>305</td>
<td>Time(sec)=80</td>
</tr>
<tr>
<td>306</td>
<td>17.991681</td>
</tr>
<tr>
<td>307</td>
<td>32.830110</td>
</tr>
<tr>
<td>308</td>
<td>-25.596360</td>
</tr>
<tr>
<td>309</td>
<td>12.983698</td>
</tr>
<tr>
<td>310</td>
<td>5.749944</td>
</tr>
<tr>
<td>311</td>
<td>6.491905</td>
</tr>
<tr>
<td>312</td>
<td>3.153249</td>
</tr>
<tr>
<td>313</td>
<td>-6.491793</td>
</tr>
<tr>
<td>314</td>
<td>-36.539692</td>
</tr>
<tr>
<td>315</td>
<td>-14.096473</td>
</tr>
<tr>
<td>316</td>
<td>22.443221</td>
</tr>
<tr>
<td>317</td>
<td>-15.765800</td>
</tr>
<tr>
<td>318</td>
<td>-28.563981</td>
</tr>
<tr>
<td>319</td>
<td>27.822133</td>
</tr>
<tr>
<td>320</td>
<td>36.910727</td>
</tr>
<tr>
<td>321</td>
<td>-7.233754</td>
</tr>
<tr>
<td>322</td>
<td>-14.096471</td>
</tr>
<tr>
<td>323</td>
<td>45.628289</td>
</tr>
<tr>
<td>324</td>
<td>30.975380</td>
</tr>
<tr>
<td>325</td>
<td>1.298403</td>
</tr>
<tr>
<td>326</td>
<td>10.943446</td>
</tr>
<tr>
<td>327</td>
<td>-8.903081</td>
</tr>
<tr>
<td>328</td>
<td>21.886781</td>
</tr>
<tr>
<td>329</td>
<td>5.564425</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td>330</td>
<td>-0.185407</td>
</tr>
<tr>
<td>331</td>
<td>36.910729</td>
</tr>
<tr>
<td>332</td>
<td>17.806160</td>
</tr>
<tr>
<td>333</td>
<td>8.717674</td>
</tr>
<tr>
<td>334</td>
<td>-14.652914</td>
</tr>
<tr>
<td>335</td>
<td>14.467509</td>
</tr>
<tr>
<td>336</td>
<td>22.999664</td>
</tr>
<tr>
<td>337</td>
<td>9.459524</td>
</tr>
<tr>
<td>338</td>
<td>9.274118</td>
</tr>
<tr>
<td>339</td>
<td>3.895099</td>
</tr>
<tr>
<td>340</td>
<td>8.717564</td>
</tr>
<tr>
<td>341</td>
<td>18.733642</td>
</tr>
<tr>
<td>342</td>
<td>33.572069</td>
</tr>
<tr>
<td>343</td>
<td>-29.676865</td>
</tr>
<tr>
<td>344</td>
<td>14.838543</td>
</tr>
<tr>
<td>345</td>
<td>9.459524</td>
</tr>
<tr>
<td>346</td>
<td>9.274118</td>
</tr>
<tr>
<td>347</td>
<td>3.895099</td>
</tr>
<tr>
<td>348</td>
<td>8.717564</td>
</tr>
<tr>
<td>349</td>
<td>40.991233</td>
</tr>
<tr>
<td>350</td>
<td>15.951319</td>
</tr>
<tr>
<td>351</td>
<td>22.628739</td>
</tr>
<tr>
<td>352</td>
<td>-18.733532</td>
</tr>
<tr>
<td>353</td>
<td>-32.30116</td>
</tr>
<tr>
<td>354</td>
<td>28.564094</td>
</tr>
<tr>
<td>355</td>
<td>41.918710</td>
</tr>
<tr>
<td>356</td>
<td>-7.604679</td>
</tr>
<tr>
<td>357</td>
<td>-16.322421</td>
</tr>
<tr>
<td>358</td>
<td>49.894422</td>
</tr>
<tr>
<td>359</td>
<td>31.160786</td>
</tr>
<tr>
<td>360</td>
<td>0.927478</td>
</tr>
<tr>
<td>361</td>
<td>11.128853</td>
</tr>
<tr>
<td>362</td>
<td>-10.572408</td>
</tr>
<tr>
<td>363</td>
<td>25.410955</td>
</tr>
<tr>
<td>364</td>
<td>6.862828</td>
</tr>
<tr>
<td>365</td>
<td>-3.895099</td>
</tr>
<tr>
<td>366</td>
<td>41.362270</td>
</tr>
<tr>
<td>367</td>
<td>19.661005</td>
</tr>
<tr>
<td>368</td>
<td>10.387002</td>
</tr>
<tr>
<td>369</td>
<td>-18.919048</td>
</tr>
<tr>
<td>370</td>
<td>-22.814146</td>
</tr>
<tr>
<td>371</td>
<td>-26.152799</td>
</tr>
<tr>
<td>372</td>
<td>-4.639877</td>
</tr>
<tr>
<td>373</td>
<td>-10.572408</td>
</tr>
<tr>
<td>374</td>
<td>18.362608</td>
</tr>
<tr>
<td>375</td>
<td>-26.709243</td>
</tr>
<tr>
<td>376</td>
<td>-9.830448</td>
</tr>
<tr>
<td>377</td>
<td>25.225434</td>
</tr>
<tr>
<td>378</td>
<td>24.112550</td>
</tr>
<tr>
<td>379</td>
<td>37.096242</td>
</tr>
<tr>
<td>380</td>
<td>-34.313924</td>
</tr>
<tr>
<td>381</td>
<td>18.919048</td>
</tr>
<tr>
<td>382</td>
<td>8.161232</td>
</tr>
<tr>
<td>383</td>
<td>10.757928</td>
</tr>
<tr>
<td>384</td>
<td>4.451541</td>
</tr>
</tbody>
</table>

Time (sec) = 90

Time (sec) = 100
<table>
<thead>
<tr>
<th>Row</th>
<th>x</th>
<th>y</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>389</td>
<td>-11.499777</td>
<td>-15.770260</td>
<td>305.235000</td>
</tr>
<tr>
<td>390</td>
<td>-45.442774</td>
<td>2.411926</td>
<td>304.574000</td>
</tr>
<tr>
<td>391</td>
<td>-17.991572</td>
<td>-40.260548</td>
<td>304.991000</td>
</tr>
<tr>
<td>392</td>
<td>26.894761</td>
<td>12.616211</td>
<td>304.515000</td>
</tr>
<tr>
<td>393</td>
<td>-22.628630</td>
<td>-26.345611</td>
<td>304.634000</td>
</tr>
<tr>
<td>394</td>
<td>-36.725214</td>
<td>-27.087740</td>
<td>304.459000</td>
</tr>
<tr>
<td>395</td>
<td>31.902750</td>
<td>-39.332884</td>
<td>304.459000</td>
</tr>
<tr>
<td>396</td>
<td>46.555769</td>
<td>10.946421</td>
<td>304.093000</td>
</tr>
<tr>
<td>397</td>
<td>-9.644932</td>
<td>-29.870729</td>
<td>304.396000</td>
</tr>
<tr>
<td>398</td>
<td>-19.104453</td>
<td>15.213665</td>
<td>304.458000</td>
</tr>
<tr>
<td>399</td>
<td>54.345963</td>
<td>20.594111</td>
<td>304.609000</td>
</tr>
<tr>
<td>400</td>
<td>34.870478</td>
<td>-16.141324</td>
<td>305.444000</td>
</tr>
<tr>
<td>401</td>
<td>0.741961</td>
<td>8.163430</td>
<td>304.567000</td>
</tr>
<tr>
<td>402</td>
<td>13.169217</td>
<td>-25.603483</td>
<td>304.709000</td>
</tr>
<tr>
<td>403</td>
<td>-12.612660</td>
<td>30.612860</td>
<td>305.650000</td>
</tr>
<tr>
<td>404</td>
<td>29.676978</td>
<td>-28.572000</td>
<td>304.302000</td>
</tr>
<tr>
<td>405</td>
<td>6.677310</td>
<td>50.833901</td>
<td>304.266000</td>
</tr>
<tr>
<td>406</td>
<td>-4.822465</td>
<td>-4.638311</td>
<td>304.317000</td>
</tr>
<tr>
<td>407</td>
<td>45.813810</td>
<td>-2.597451</td>
<td>304.525000</td>
</tr>
<tr>
<td>408</td>
<td>20.773890</td>
<td>45.084395</td>
<td>305.004000</td>
</tr>
<tr>
<td>409</td>
<td>13.354622</td>
<td>10.018755</td>
<td>304.674000</td>
</tr>
<tr>
<td>410</td>
<td>-22.443111</td>
<td>-8.348961</td>
<td>304.949000</td>
</tr>
<tr>
<td>411</td>
<td>-26.894650</td>
<td>3.154054</td>
<td>304.599000</td>
</tr>
<tr>
<td>412</td>
<td>-30.604338</td>
<td>34.323512</td>
<td>304.319000</td>
</tr>
<tr>
<td>413</td>
<td>-6.862717</td>
<td>18.738782</td>
<td>304.338000</td>
</tr>
<tr>
<td>414</td>
<td>-12.798177</td>
<td>44.342265</td>
<td>305.200000</td>
</tr>
<tr>
<td>415</td>
<td>21.330339</td>
<td>-41.002678</td>
<td>305.008000</td>
</tr>
<tr>
<td>416</td>
<td>-30.418822</td>
<td>17.440056</td>
<td>304.421000</td>
</tr>
<tr>
<td>417</td>
<td>-12.798179</td>
<td>3.525118</td>
<td>304.716000</td>
</tr>
<tr>
<td>418</td>
<td>30.789860</td>
<td>-0.556596</td>
<td>304.998000</td>
</tr>
</tbody>
</table>


