Effects of increased connectivity amongst leaders on the performance of a multi-agent system

Shamoil Khomosi

Reg. No. 190134204

May 2023

Supervisor: Dr Anton Selivanov

Department of Automatic Control and Systems Engineering

A dissertation submitted in partial fulfilment of the requirements for the degree of BEng Mechatronics and Robotics Engineering.

Contents

Abstract

The behaviour of natural swarm systems has long fascinated researchers and inspired the development of multi-agent systems, which are now being employed in emerging fields such as robotics, space systems, computer networks, and smart grids, to perform tasks such as formation control. However, achieving a desired formation with decentralised swarm systems poses control challenges, particularly when employing swarm agents with limited sensory awareness and inter-connectivity. This study investigates the impact of different levels of connectivity (or communication) amongst leader agents and proposes piecewise controllers that approximate the states of the follower agents. In addition, the study introduces a global controller that employs signal strength as an indicator for follower state approximation. The research also examines the challenges of implementing these controllers on e-puck robot hardware and the necessary modifications to their design and communication structure. It is found that greater leader connectivity provides substantial improvements in the settling performance only in the presence of a reduced number of leaders.

Abbreviations

MAS	Multi-Agent System
PDE	Partial Differential Equation
ODE	Ordinary Differential Equation
RSSI	Received Signal Strength Indicator
WBS	Work Breakdown Structure
PATS	Personal & Academic Tutoring System
RFID	Radio Frequency Identification
WSN	Wireless Sensor Network
ADR	Advection-Diffusion-Reaction
CRC	Cyclic Redundancy Check
UID	Unique Identifier
TOF	Time of Flight
GPS	Global Positioning System
LED	Light Emitting Diode
RAM	Random Access Memory
BLE	Bluetooth Low Energy
IMU	Inertial Measurement Unit

Symbols

N	Number of leader and follower agents.
N_l	Number of leader agents.
x_i	Position of the ith agent along the MAS deployment profile.
z_i	Position of the ith agent in the global coordinate system.
γ_i	Position of the target for the ith agent in the global coordinate system.
$\hat{z}_i$	Position estimate for the ith follower agent in the global coordinate system.
u_i	Local controller for the ith agent.
v_i	Global controller for the ith agent.
h	Separation between two adjacent agents.
l	Total length occupied by the multi-agent system; defined as $l = Nh$.
κ	Local controller gain.
ν	Global controller gain.
a	Coefficient describing first-order local dynamics of the agents.
ρ_j	Refers to the jth agent, which is generally a leader.
$L_x^{f_y}$	Lagrange's interpolating basis functions L_x for a given function f_y.
$\bar{\lambda}$	Signal strength normalised either against A_{max} or $\sum_{j=0}^{J} A_j$.
A_j	Strength of the signal received from the jth leader agent.
A_{max}	Maximum strength possible for any received signal for a given h.
$\mathbf{r}_j$	Euclidean distance between the jth leader agent and a given follower.
Λ	Degree of exponentiation for $\mathbf{r}$ used in an RSSI-based global controller.
J	Number of leaders that have a given follower in their transmission range.
r	Radius of the agents.
n_s	Number of infrared distance sensors placed on the periphery of each agent.
d_ι	Distance information captured by the ιth infrared distance sensor.
d_{max}	Maximum inter-agent separation distance for a given n_s, beyond which the agents may not be able to detect each other.
Δ_l	Displacement to the left neighbouring agent.
Δ_r	Displacement to the right neighbouring agent.
k	Step number for a discrete-time variable.

T_{sim} Webots simulation time-step.

n MATLAB simulation time-step.

w_l Rotational velocity of the e-puck robot's left wheel.

w_r Rotational velocity of the e-puck robot's right wheel.

Chapter 1

Introduction

A swarm of starlings flowing through the skies in unison produces scenic spectacles. The coordination amongst such swarms generates intricate patterns where the birds amass and their individual identities become indistinguishable. Inspired by the behaviour of natural swarm systems, multi-agent systems are now being employed in emerging fields such as robotics, space systems, computer networks, and smart grids.

One prominent example of such systems is drone light shows, where groups of illuminated and choreographed drones arrange themselves into various aerial formations. In fact, the largest drone show on record occurred on New Year's Eve of 2023 at the Sheikh Zayed Festival, where 5184 drones were simultaneously airborne [1]. Such events not only display the possibilities of modern technology but also demonstrate an environmentally friendlier alternative to traditional fireworks displays. The use of swarm systems, such as drone shows, has transitioned from being a topic of theoretical research to becoming a pragmatic solution in a multitude of scenarios. Multi-agent or swarm systems are being employed in a variety of fields, such as performing search and rescue operations [2], optimising wireless communication networks [3], deploying autonomous underwater vehicles [4], balancing loads on cloud-based servers [5], and operating smart grids [6].

Swarm systems usually comprise two types of agents, namely leaders and followers. Leaders, though limited in number, possess higher sensory and networking capabilities. In contrast, followers typically have limited sensory awareness and rely on communication with their neighbours and leaders to execute tasks. Utilising followers can reduce the overall cost by avoiding expensive equipment for all agents and decentralising the system to minimise the risks of single points of failure. However, doing so also poses control challenges due to the limited information available from their inexpensive sensors — thus requiring communication with their respective leader to deduce awareness of their state in the deployment environment.

Achieving a desired formation is a critical task when deploying a swarm, where prompt and accurate deployment of all agents to their respective targets is paramount for achieving the swarm's objective. Consequently, formation control is an active research area within the field of multi-agent systems (MAS). To enable the extension of formation control problems to large-scale multi-agent systems, swarms have been modelled as a continuum of agents, which are separated by infinitesimally small distances. By doing so, the movement of swarms can be compared to the behaviours observed in fluid flow, where partial differential equations (PDEs) are used for convenient modelling in the temporal and spatial domains. This approach allows an emphasis on the control aspects of formation control rather than indulging in the intricacies of real-world implementation.

Nevertheless, there are research gaps in utilising these models to evaluate the performance of MAS with different levels of connectivity between leaders and followers. Additionally, limited effort has been made to explore the practical application of such models on hardware or in virtual environment simulations. Investigating these areas could provide insights into the potential limitations and costs of increasing connectivity in MAS and its impact on the system's overall capabilities.

This study aims to investigate the factors that affect the performance of a MAS with varying sensing capabilities, separation between agents, inter-connectivity between leaders, and number of leaders in the swarm. The focus of the investigation is on global controllers for followers. By utilising different state approximation techniques, the global controllers compensate for the inability of a follower to sense its distance to the target curve. The paper also examines the challenges of implementing these controllers on hardware and the necessary modifications to the agents' design and communication structure. The performance of the MAS is evaluated through numerical analysis in MATLAB and virtual environment simulations in Webots, and the study briefly touches on the deployment of the swarm in two-dimensional spaces with non-zero targets.

1.1 Aim

Formation control is an active area of research within multi-agent systems (MAS). The effectiveness of a displacement-based formation control strategy may be dependent on various factors, including the sensing capabilities of the agents, range of communication, inter-agent connectivity and the number of leader agents. This research investigates the effects of increased

connectivity amongst the leaders, and finer piecewise approximations for controlling the followers, on the performance of a MAS modelled by a diffusion PDE. The performance of the MAS is assessed by studying the deployment of the agents to desired formations. Unlike ordinary differential equations (ODE), PDEs provide a method to model large-scale MAS whose complexity does not grow as the number of agents increases. The results will be validated via numerical analysis in MATLAB and virtual environment simulations in Webots.

1.2 Objectives

Basic Objectives:

1. Review the literature concerning PDE-based analysis and formation control of leader-follower MAS.
2. Develop a multi-agent interpretation of a PDE for the deployment of global controllers.
3. Design state feedback controllers for followers where the states are replaced by piecewise approximations in MATLAB.
4. Compare the performance of the MAS for controllers using piecewise constant, linear, and higher-order state approximations.
5. Implement non-zero target curve tracking under displacement-based formation control.

Advanced Objectives:

1. Emulate a MAS consisting of e-puck robots in Webots with varying leader connectivity and leader-follower control inputs.
2. Evaluate the performance of a received signal-strength based state approximation for the followers.

1.3 Project management

To ensure that the project was completed within the given time frame, a structured approach was taken with a work breakdown structure (WBS). The WBS was based on the objectives, which were then divided into smaller tasks. The progress of the project was tracked using a Gantt chart that was reviewed periodically to ensure that it aligned with current progress. This chart included key deadlines and considered other academic commitments to have maximum foresight into the upcoming workload. Next steps and any immediate results were tracked in an electronic logbook and on the university's PATS system. The project was able to achieve all

basic and advanced objectives.

The project began with an initial literature review, and rapid progress was made in achieving the basic objectives by Semester 1 Week 8. The scope of the project was broadened to facilitate maximum research in the time available, and appropriate training was completed with the Sheffield Robotics Lab. However, progress slowed in Weeks 10-12 due to increased academic workload. Semester 1 was mainly spent performing numerical simulations and analysis in MATLAB.

The project entered the virtual environment/hardware implementation phase during the Christmas break and the early weeks of Semester 2. To address potential hardware limitations, thorough research was conducted into e-pucks, and practical alternatives were investigated. During initial simulations in Webots, it was discovered that the e-puck's hardware was not suitable for this project due to the limited number and range of its proximity distance sensors. This risk was mitigated by spending more time on virtual environment (Webots) simulations to make the results a near-to-true reflection of the real world. The simulations were automated using a Python script for easy configuration changes. Only up to 40 agents could be simulated due to the limited processing capacity of the personal machine used. University machines were not available for simulation purposes due to IT restrictions, and remote work during Easter. Experimental configurations were adjusted to optimise the simulation results with limited agents.

Extensive simulations were conducted to analyse the controllers under various conditions. Doing so facilitated the exploration of hardware challenges that could be encountered upon the implementation of the controllers proposed in this study. The simulation-led approach led to the implementation of the signal-strength based controllers, which have not been previously explored by PDE-based multi-agent formation control research. The collation of all results was initiated in Semester 2 Week 7. This was followed by the dissertation being prepared over the Easter break to ensure that there was enough time for amendments before the submission deadline in Semester 2 Week 12.

1.3.1 Key changes

The implementation of the designed controllers has not been performed on e-pucks due to the reasons detailed in the previous section. Instead, the numerical and virtual environment simulations were extended to include scenarios of practical deployment for multi-agent systems, such

as target tracking and two-dimensional deployment. The objectives have been revised to reflect these changes. The originally defined objectives have also been provided in Appendix A.1.

1.3.2 Updated work breakdown structure

1. **Conduct a detailed literature review.** Initial literature review on multi-agent systems. Methods of modelling swarms of agents. PDE-based modelling and analysis of swarms. Investigation of consensus protocol. Formation control techniques based on sensory capabilities. Experimentation methods. Software and simulation tools utilised.

2. **Implement the controllers and perform numerical simulations.** Numerical solutions of discretised PDEs in MATLAB. Evaluation of piecewise constant control inputs; piecewise linear control inputs; piecewise higher-order control inputs; signal-strength derived control inputs.

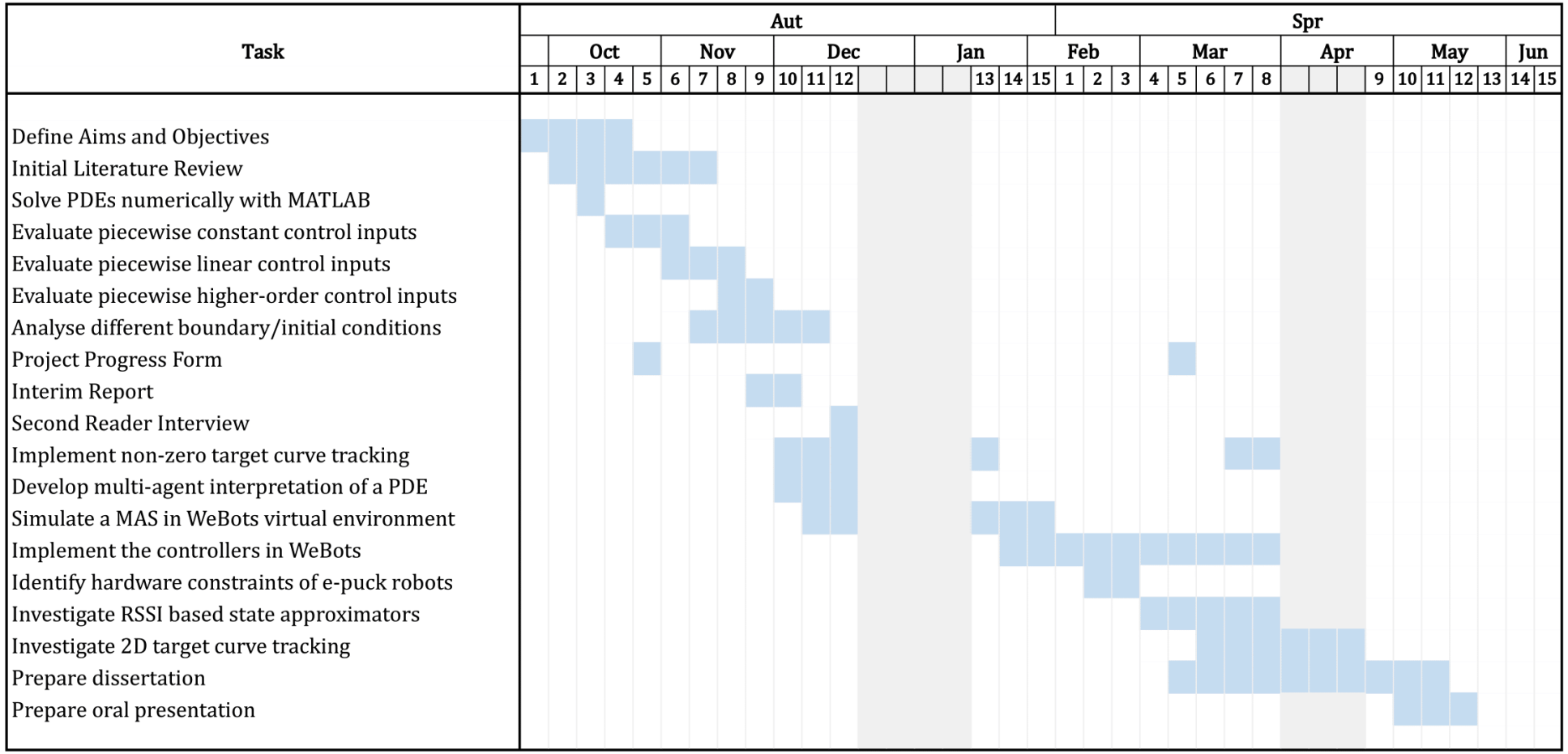
3. **Investigate the implementation of the controllers on hardware.** Sensors for the global controller (sensing method and number per agent). Sensors for the local controller (sensing method and number per agent). Communication modules for agents. Leader-follower communication topology. Leader-leader communication topology. Webots configuration for followers. Webots configuration for leaders.

4. **Assess the performance of the MAS under different test conditions.** Evaluation of different boundary and initial conditions. Evaluation of different numbers of agents, leaders, and agent separation. Evaluation of different controller gains. Deployment to non-zero target curves. Deployment to two-dimensional target curves. Hyper-parameters and gains for the controllers.

5. **Compare the performance of the controllers.** L_2 norm (error) analysis from MATLAB simulation. L_2 norm (error) analysis from Webots simulation. Measurement of performance indices (settling time, standard deviation, leader connectivity). Interpretation of effects of increased leader connectivity. Tabulation of results and simulation logs.

6. **Monitoring and reporting project progress.** Definition of aims and objectives. Termly project progress forms. Interim report. Second reader interview. Publication of simulation software and documentation. Dissertation writing and submission. Oral presentation.

1.3.3 Gantt chart



Chapter 2

Literature Review

2.1 Overview of multi-agent systems

A multi-agent system (MAS) refers to a network of interconnected components, referred to as agents, that coordinate and communicate with each other to achieve a collective goal. An example of the desired goal for a MAS is to track prescribed trajectories via formation control [7]. Based on the sensory capability and the degree of connectivity within the agents, problems concerning formation control can be broadly categorised into a position, displacement and distance-based topology [8], as shown in Figure 2.1.

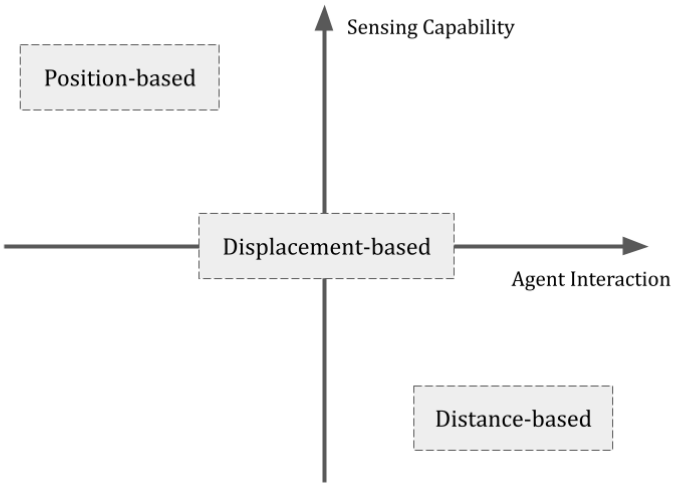


Figure 2.1: Formation control topologies. Source: Adapted from [8].

2.2 Formation control

In position-based formation control, all agents have the ability to sense their positions in the global coordinate system and hence actively control their distance to the target position to achieve the desired formation. On the contrary, in a distance-based control topology, the agents

can only sense their positions in a local coordinate system and hence control their distance relative to their neighbouring agents. Such a scheme is also referred to as a consensus protocol, where the agents only control their relative displacements to their neighbours to achieve a formation [9]. In the present study, *local control* is defined as the regulation of variables sensed in an agent's local coordinate system. Although position-based control is advantageous in terms of minimum inter-agent connectivity (or communication) requirements, distance-based control is beneficial for agents with limited sensing ability.

Problems concerning the consensus protocol have been studied widely in the control literature considering a variety of operating conditions, including fixed [10] and switching [11] network topology, discrete-time [12] and continuous-time agent dynamics [13], bidirectional and unidirectional communication, and constant and time-varying [14] communication delays.

Displacement-based formation control, which is the focus of this study, is a combination of position-based and consensus-based control and can be achieved via a leader-follower approach. In this approach, only the leader agents have an awareness of their position in the global coordinate system and hence can track their absolute position against the desired curve or trajectory. Awareness of position in an arbitrary space can, for example, be accomplished with time-of-flight distance sensors and visible markers that indicate the desired (target) curve.

On the other hand, the follower agents only have an awareness of their position and orientation relative to their neighbours. The follower agents control their displacements relative to their neighbouring agents in order to achieve the desired formation. These controllers can also be extended to actively drive the followers to the target curve based on their state approximations, rather than relying solely on controlling inter-agent displacements. The state approximation for each follower is specified based on the chosen control law and is relayed to the follower-agent by their corresponding leader. The choices available for the aforementioned control law are analysed and discussed in detail in this research. Displacement-based formation control for non-holonomic agents, which is the case for the e-puck robots in this study, was explored in [15]. In a practical implementation, local or relative positional awareness for the follower agents can be achieved via magnetic or infrared sensors [16].

2.3 Localisation and connectivity

An edge-following algorithm was developed in [17] for the self-assembly of hundreds of 'kilobot' robots onto tightly-packed complex two-dimensional shapes. Instead of leader agents, the study used pre-localised seed robots that defined the origin of the desired shape's coordinate system. [18] conducted experiments with heterogeneous robots to undertake mapping and deployment tasks in a large indoor environment. Even though a mixture of intelligent and simple robots was used in this study, it required the usage of expensive sensory equipment for localisation and a continuous flow of agents was not guaranteed whilst achieving a formation.

An acoustic localisation technique, namely Marco-Polo localisation, was first introduced in [19], where distances to neighbouring robots are estimated by measuring the time it takes their emitted audible-frequency sound to travel. This technique is particularly useful when the leaders are not aware of other leaders present in their proximity, and hence can be used as a precursor to initiating communication via other means with them. Similar experiments have also been performed with ultrasound [20], and acoustic beacons [21]. Acoustic localisation can also be implemented with a system of microphones and speakers on e-puck robots, as shown in [22], however, this technique has poor directional resolution and requires extensive bandwidth allocation for the agents making it less scalable. RFID-based localisation was used in [23], however, it suffered from loss of precision based on the geometry of the RFID grid and posed the additional cost of establishing and maintaining a grid of RFID tags in a physical environment. To bridge such precision constraints, [24] combines a Kalman-filter, that acts on dead-reckoning sensor data and Zig-bee-based WSN localisation, with RFID tags to eliminate any accumulated formation errors.

A more popular approach involves installing Aruko markers on top of the agents as a visual marker in [25], which would then be continually tracked by an overhead-mounted camera. The information would be fed into a computer to calculate the positions of each agent and then be relayed by a global communication system. Although this method offers high precision and minimal hardware and processing requirements for the agents, it is not scalable to large outdoor environments and is highly centralised as the majority of the processing is performed by an off-board computer. To decentralise the multi-agent system, onboard cameras could be used on agents for local visual identification, such as the use of colour-coded LEDs [26], however, this would be computationally demanding and cost-ineffective on a per-agent basis.

2.4 Mathematical models

MAS modelling has been traditionally approached via matrix and graph theory [9][27], which involves individually describing the dynamics and interconnectedness of agents via ODEs. As the number of agents in a given system increases, this method becomes numerically complex and computationally time-consuming [28]. PDE-based models of a MAS overcome this scalability problem, as the structure of a PDE is fundamentally independent of the number of agents in the system. Doing so not only reduces the numerical complexity of analysing a continuum of ODEs but also provides a wider range of mathematical tools to infer the stability and performance of a PDE-based MAS model under different control laws.

2.4.1 PDE-based analysis and control

Quantitative theories exploring the flocking behaviours of natural swarm formations, such as flocks of birds, were initially studied in [29][30]. The developed models were in many ways similar to PDEs such as Navier-Stokes and heat diffusion, which are frequently used in describing fluid flow. Multi-agent dynamics can, for instance, be described with the advection-diffusion-reaction (ADR) family of PDEs [31]. The diffusion term enforces cohesion amongst the agents by minimising inter-agent separations, thus forming the basis of the local controller in the present study (3.2). The reaction term drives the system to the target curve at steady-state and forms the basis of the global controller (3.4). Finally, the advection term is proportional to the gradient of the agents' states and can be used to model the local dynamics of the agents. A fluid-based PDE model for swarm agents was described in [32], which addressed pattern generation problems for multi-agent systems. Following the developments in PDE-based control of agents, the stability of large platoons of vehicles was studied using linear hyperbolic PDEs in [33]. PDE-based swarm models have also been used to study and model traffic flow [34].

A PDE-based approach for deploying swarm agents on a variety of planar curves was explored in [31]. Interestingly, two-dimensional planar deployments were achieved by leveraging two independent (orthogonal) one-dimensional deployments using the complex-valued Ginzburg–Landau PDE as a continuum model for the agents' two-dimensional dynamics. Anchor agents were used in this study to stabilise the desired formation, however, stabilisation in the case where virtual agents are present on the boundaries was not explored. Furthermore, the followers possessed position awareness in the global coordinate system and hence the system only required one leader and one anchor agent. Two-dimensional deployment manifolds

in three-dimensional space were also achieved in [35], where they attained open-loop stability for boundary control using the back-stepping method, with a Poisson kernel, on a disc parameterised by polar coordinates.

A distributed leader velocity estimator and finite-time distance-based control laws were devised in [36], however, they did not consider any degree of interconnectedness between the leaders and the hardware implications of the developed control laws. In [37][38], a displacement-based formation control algorithm was developed to study the deployment of agents onto a closed curve. However, further study needs to be conducted to evaluate the deployment under a varying number of leader agents, and different state approximations used for the control of follower agents. Displacement-based control was extended further in a recent study for the case where every agent measures their relative position to only one neighbour [39].

While the majority of studies have proposed parabolic PDE-based control methods with single-integrator agent dynamics, i.e. where the velocity of the agents is controlled, [40] proposed the use of a second-order linear wave PDE model to actively control the acceleration of the agents instead. On the other hand, a second-order non-linear PDE, namely the viscous Burger's equation, was used by [41] to describe corner-like and more complex shapes in deployment profiles. The aforementioned studies have proven the robustness of their proposed control laws, however, their applicability to non-point agents such as non-holonomic robots is yet to be determined.

Chapter 3

Modelling and Numerical Analysis

The following chapter introduces a PDE-based description of a multi-agent system. As highlighted in Section 2.4.1, the PDE-based model is used for developing the local and global controllers for the agents. The proposed controllers are simulated in MATLAB and the chapter concludes by analysing the simulation results and the settling performance of the multi-agent system.

3.1 PDE interpretation of a multi-agent system

A multi-agent system with $N+1$ agents can be represented by (3.1), which is an approximation of a second-order linear PDE when N is large [27],

$$\dot{z}_i(t) = f(t, z_i(t)) + u_i(t) + v_i(t), \quad i \in \{0, 1, \ldots, N\} \tag{3.1}$$

with state $z_i : (-\infty, \infty) \to \mathbb{R}$ for the ith agent [39]. The agents are modelled with single-integrator dynamics and are uniformly distributed in the space $x : [0, l] \to \mathbb{R}$ where they are placed $h = \frac{l}{N}$ distance apart. The dynamics of the ith agent, modelled as a first-order differential equation, are described as a function of its state $f(t, z_i(t)) = a z_i(t)$.

The local controller $u_i(t)$ that establishes the consensus protocol between the neighbouring agents is described in (3.2), where $\kappa \in \mathbb{R}$ is the controller gain. The consensus protocol enhances the cohesion of a swarm by iteratively communicating and updating the state of the agents based on the states of their neighbours.

$$u_i(t) = \kappa \frac{\partial^2 z_i}{\partial x^2} = \lim_{h \to 0} \kappa \frac{(z_{i-1}(t) - z_i(t)) + (z_{i+1}(t) - z_i(t))}{h^2} \tag{3.2}$$

The agents located at the boundary i.e. $i = 0, N$, can be either leaders, followers, or a combination of both. For the controllers discussed in the subsequent sections, it is assumed that the

boundary agents are leaders. To incorporate this assumption, the Dirichlet boundary condition is used in (3.3) to model these agents. In this case, the consensus protocol is implemented with only one neighbouring agent due to the limited connectivity of the boundary agents.

$$u_0(t) = \kappa \frac{(z_1(t) - z_0(t))}{h}$$
$$u_N(t) = \kappa \frac{(z_N(t) - z_{N-1}(t))}{h} \tag{3.3}$$

The global controller $v_i(t)$ is described in (3.4), where $\hat{z}_i$ is an approximation of the ith agent's state, and $\nu \in \mathbb{R}$ is the controller gain.

$$v_i(t) = -\nu \hat{z}_i(t) \tag{3.4}$$

As the leader agents ρ_j ($j \subset i$) have an awareness of their absolute position in the global coordinate system, their global controller can always be described with $\hat{z}_j(t) = z_j(t)$.

3.1.1 Choices for global controllers

When a **piecewise constant** function is used for the state approximation of the followers, the approximation in (3.5) can be used, where z_j is the state of the leader closest to the ith agent.

$$\hat{z}_i(t) = z_j(t) \tag{3.5}$$

When a **piecewise linear** function is used for the state approximation of the followers, (3.6) is obtained, where $x_i \in [0, l]$ is the spatial position of the ith agent in the global coordinate system. The implementation of a piecewise linear function within a global controller requires the leader ρ_j to communicate with the neighbouring leader ρ_{j-1} to obtain the state z_{j-1}. In general, greater connectivity amongst the leaders is required as the order of the state approximations increases.

$$\hat{z}_i(x,t) = z_j(t) + \frac{z_{j-1}(t) - z_j(t)}{x_{j-1} - x_j}(x_i - x_j) \tag{3.6}$$

When a **piecewise quadratic** function is used for the state approximation of the followers, an expression derived from Lagrange's interpolating polynomial can be obtained as follows:

$$\hat{z}_i(x,t) = z_{j+1}(t)L_0(x) + z_j(t)L_1(x) + z_{j-1}(t)L_2(x) \tag{3.7}$$

where L_0, L_1, and L_2 are Lagrange's interpolating basis functions defined as:

$$L_0(x) = \frac{(x_i - x_{j-1})(x_i - x_j)}{(x_{j+1} - x_{j-1})(x_{j+1} - x_j)}$$

$$L_1(x) = \frac{(x_i - x_{j-1})(x_i - x_{j+1})}{(x_j - x_{j-1})(x_j - x_{j+1})}$$

$$L_2(x) = \frac{(x_i - x_j)(x_i - x_{j+1})}{(x_{j-1} - x_j)(x_{j-1} - x_{j+1})}$$

A quadratic function may generate discontinuities in the state approximation for an agent that is equidistant from its two neighbouring leaders, as observed at $x = 400$ and 600 in Figure 3.2. Thus, in some cases, it may be beneficial to obtain a piecewise function which produces a weighted average of the quadratic functions based on the position of the agent with respect to its neighbouring leaders. Following this idea, when a **piecewise cubic (convex) function** is used for the state approximation of the followers, two different approximations, based on the location of the follower respective to its leader, need to be developed. The first approximation considers the case when the follower agent is situated between its leader ρ_j and the neighbouring leader ρ_{j-1}. The second approximation considers the case when the follower agent is between ρ_j and ρ_{j+1}. For brevity, only the latter case has been described in (3.8).

$$\hat{z}_i(x,t) = \frac{x_{j+1} - x_i}{x_{j+1} - x_j} f_1(x,t) + \frac{x_i - x_j}{x_{j+1} - x_j} f_2(x,t) \tag{3.8}$$

f_1 and f_2 are defined as Lagrange's interpolating polynomials for the states z_{j-1}, z_j, z_{j+1}, and z_{j+2}:

$$f_1(x,t) = z_{j+1}(t)L_0^{f_1}(x) + z_j(t)L_1^{f_1}(x) + z_{j-1}(t)L_2^{f_1}(x)$$

$$f_2(x,t) = z_{j+2}(t)L_0^{f_2}(x) + z_{j+1}(t)L_1^{f_2}(x) + z_j(t)L_2^{f_2}(x)$$

The **received signal strength indicator (RSSI)** of the data communicated by the leaders can also be used as an indicator for the state approximation of followers. The strength of a received signal is inversely proportional to the square of the transmission distance. This notion can be utilised to translate signal strength into an indication of the receiver's distance from the signal source, i.e., the transmitter. Consider the state approximation based on the RSSI defined in (3.9),

$$\hat{z}_i(x,t) = \sum_{j=0}^{J} \overline{\lambda}_j(x) z_j(t) \tag{3.9}$$

J is the number of leaders present in the multi-agent system and $\overline{\lambda}_j(x)$ is the normalised signal

strength defined in (3.10),

$$\overline{\lambda}_j(x) = \frac{A_j}{\sum_{m=0}^{J} A_m} \approx \frac{\frac{1}{\mathbf{r}_j^{\Lambda}}}{\sum_{m=0}^{J} \frac{1}{\mathbf{r}_m^{\Lambda}}}, \quad \Lambda = 2 \tag{3.10}$$

where $\mathbf{r}_j = \|x_i - x_j\|$ and $\mathbf{r}_m = \|x_i - x_m\|$ are the Euclidean distances between each follower (i) and the leaders (j,m) present in the swarm. A_j and A_m are the strengths of the signals transmitted by leader agents ρ_j and ρ_m respectively, and received by the ith (follower) agent.

The parameters that can be further tuned in (3.9) and (3.10) are:

1. Λ, which defines the degree of exponentiation of $\mathbf{r}$, can be interpreted as the order of the Minkowski distance for $\mathbf{r}$. When a value of $\Lambda >> 2$ is used, the normalised signal strength of the nearest leader becomes exponentially larger. Thus the state $\hat{z}_i$ converges to a piecewise constant state approximation. On the other hand, $\Lambda = 0$ produces an equal weighting $\overline{\lambda}_j$ for all signals received by the follower as $\overline{\lambda}_j$ becomes independent of $\mathbf{r}_j$. This may expose the follower to noisy measurements and increases the influence of leaders that are further away. $\Lambda = 2$ has been employed in this study as the received signal's strength can be directly substituted in (3.10).
2. J defines the number of neighbouring leaders a given follower can receive and process signals from. Limiting the value of J can be beneficial for reducing the influence of far away leaders on a follower's state, which may be particularly desirable for complex formations with a large variance or with a high ratio of leader-to-follower agents. Additionally, in the real world, a follower may not be able to receive signals from all leaders due to signal attenuation and limitations in the processing unit's ability to perform digital signal processing. Therefore, tuning the parameter J can also be used to increase the accuracy of the simulation. As the number of leaders $N_l << N$ in the present study, J is set equal to N_l for subsequent analysis.

The proposed RSSI-based controller is expected to be stable if the positions of the leaders are bounded. For a given follower system and fixed values of $\Lambda \in \mathbb{R}$ and $J \in \mathbb{N}$, the state approximation output of the RSSI controller will also remain bounded, since it is a linear combination of the state variables z_j and the normalised signal strengths $\overline{\lambda}_j$, which are also bounded by construction.

3.2 Simulation results

To model and simulate a PDE-based multi-agent system, a numerical solver is designed via the method-of-lines (or the explicit finite difference) technique, where the PDE is spatially discretised to provide multiple ODEs that span the temporal dimension. The resultant ODEs can be conveniently solved via the `odesolver` function in MATLAB (implementation details provided in Appendix A.2).

To evaluate the settling performance of the MAS, the ***L_2* error norm** is computed for the states of the agents as they converge to the target curve. The L_2 norm, which is an aggregation of the Euclidean distances between all agents and their respective targets, is defined as:

$$||e(\cdot,t)||_{L^2} = \sqrt{\int_0^l |e(x,t)|^2 \, dx} \tag{3.11}$$

Performance metrics of the controllers discussed in Section 3.1.1 vary against non-zero target curves based on the controller gains and the complexity, or the polynomial order, of the target curve. Hence, to reduce the dimensionality of the analysis, the simulations are performed under regulation control, i.e., control to the origin. This results in a simplification of the error term in (3.11) to $e(x,t) = z(x,t) - \gamma(x) = z(x,t)$. The simulation results are produced with an initial function $z_i(0) = 10\sin(\frac{2\pi x_i}{l})$, where $l = 1$. A sinusoidal function is chosen as it can be conveniently extended to describe a rich family of possible geometric curves, as shown via the deployment on Lissajous curves in [31], which are a sum of perpendicular and phase-shifted sinusoidal functions.

The controller gains and system dynamics for the MAS in (3.1) are arbitrarily defined as $\nu = 10, \kappa = 0.016$, and $a = 4$. The study maintains consistency in analysis by using the same arbitrarily defined gains across different controllers. The optimal selection of gains is not discussed in this study, however, the impact of the local and global controllers on MAS performance is described in Section 4.5.1. $N = 1000$ agents are placed at a step of $h = \frac{l}{N} = 0.001$ in the x-direction, and the time-step for the `odesolver` is chosen as $n = 0.001$. The number of leaders is set to be $N_l = 5$. The boundary agents are also considered to be leaders but they are excluded from the leader count N_l. To simplify the simulations, the boundary functions are set as $z_0(t) = z_N(t) = 0$. Numerical stability is guaranteed by satisfying the constraint $\frac{n\kappa^2}{h^2} \leq \frac{1}{2}$ [42]. The performance of the MAS under different state approximations is provided in Figure 3.1.

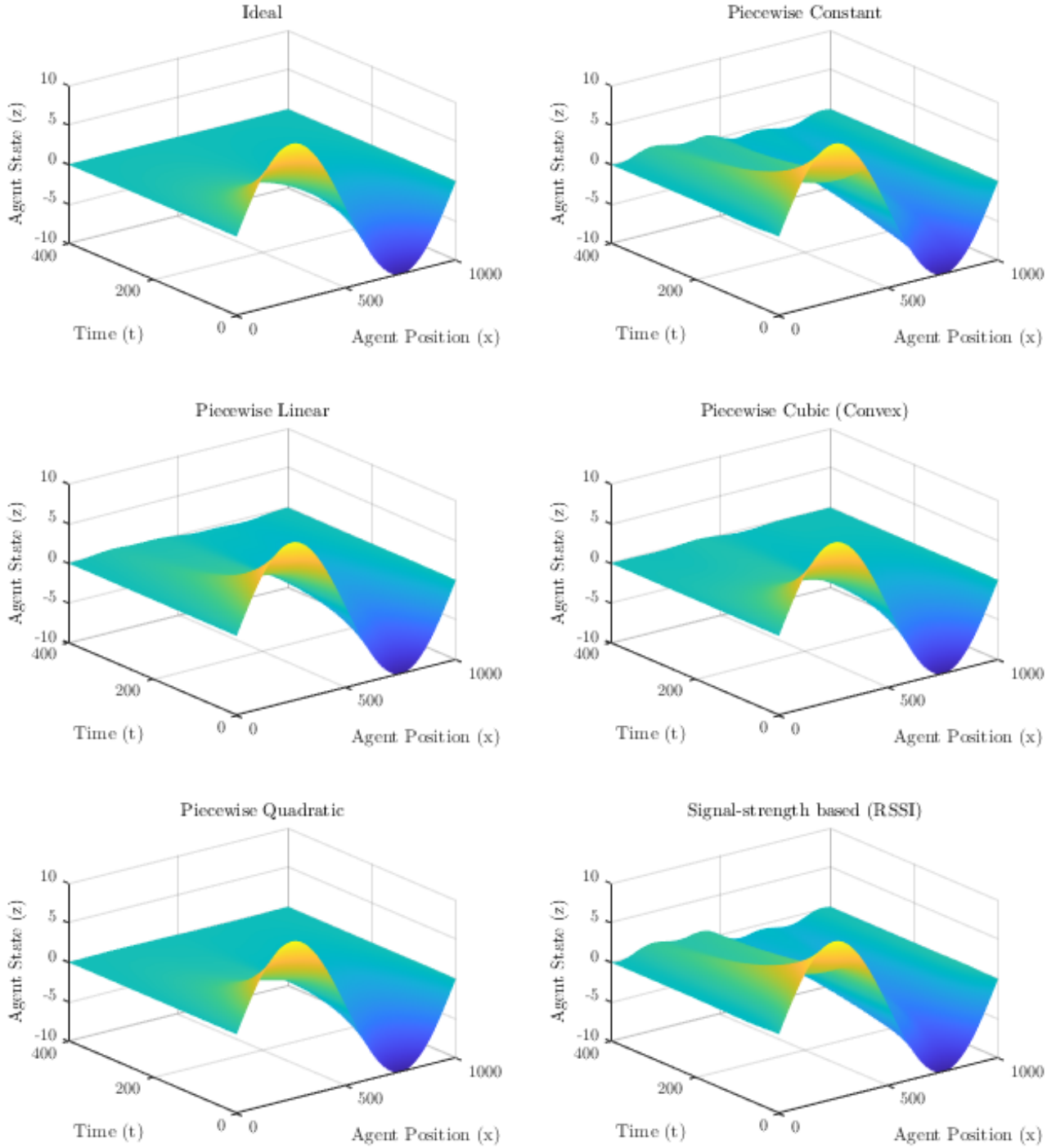


Figure 3.1: Settling performance of a MAS with different global controllers (state approximations) for the follower agents. $N = 1000$ agents are placed in a continuum on the Agent Position axis with an initial condition of $z_i(0) = 10\sin(\frac{2\pi x_i}{l})$. The collective decay of the agents towards zero ($z = 0$) is shown along the Time axis. $N_l = 5$ leader agents are evenly distributed along the Agent Position axis.

3.3 Analysis of controller performance

From Figure 3.2, it can be observed that the L_2 settling time and the cohesion of the MAS improves with higher-order piecewise state approximations. An ideal state-approximation case is also considered in the simulations where all agents are leaders, and hence have an awareness of their positions relative to the target curve. The ideal state-approximator is equivalent to the position-based control discussed in [8]. This provides a benchmark for comparing the state feedback controllers that have been developed.

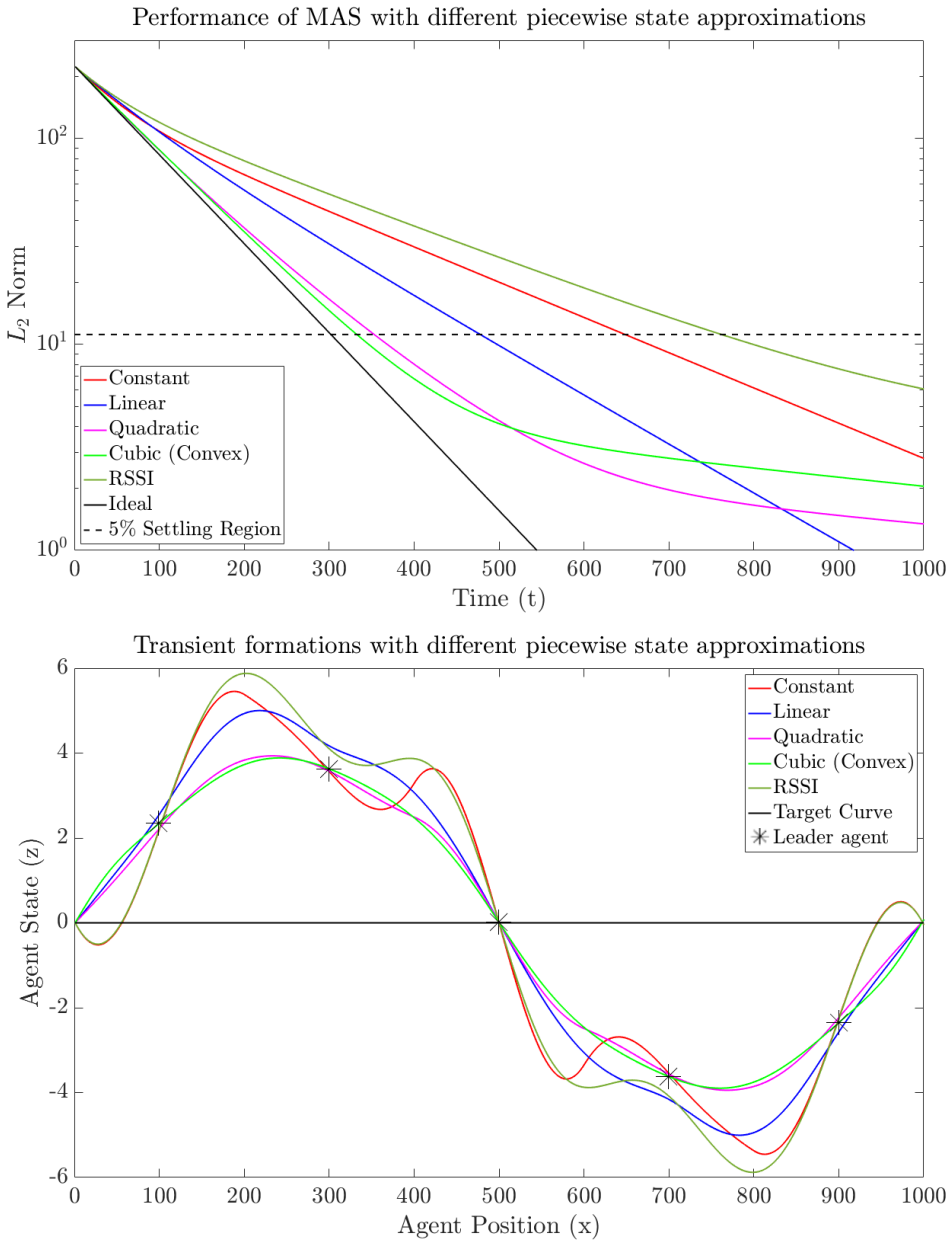


Figure 3.2: (a) L_2 performance with different piecewise state approximations for the agents, (b) Comparison of the states of the agents at t=100. These figures have been derived from the results in Figure 3.1.

Performance Index	Global Controller					
	Ideal	Constant	Linear	Quadratic	Cubic	RSSI
Leader Connectivity[1]	-	0	1	2	4	0
Settling Time[2]	304	650	480	356	334	764
Std Deviation[3]	-	0.7783	0.4903	0.1439	0.0863	0.9402
Simulation Time[4]	5.87	5.21	5.58	6.19	8.30	10.39

Table 3.1: MATLAB simulation performance indices.

[1] The number of neighbouring leaders a given leader agent is required to communicate with. It is assumed that the leaders have prior knowledge of the target curve and do not need to communicate their intentions with the rest of the leaders beforehand.
[2] Number of simulation steps required for the L_2 error norm to reduce to less than 95% of the initial error value.
[3] The maximum standard deviation of the agents from the transient performance achieved by an *ideal* controller — a lower value indicates greater cohesion amongst the agents.
[4] The time taken [seconds] to run the simulation measured using MATLAB's stopwatch timer function *tic toc* — provides an indication of the computational complexity of the controller.

The performance indices of the MAS have been tabulated in Table 3.1. The settling times and standard deviations reduce as higher-order controllers are used, however, the computational complexity also increases as a higher-order interpolating polynomial needs to be solved for each follower agent. The RSSI controller has a relatively large settling time and standard deviation of 764 steps and 0.9402 respectively, however, it may be preferred where hardware constraints limit the capabilities of the system (4.1.2). Additionally, it can be observed that the settling time is improved by 49% when a (piecewise) linear controller is used in place of a constant controller, and by 70% when a quadratic controller is used in place of a linear controller. Similar trends are also observed in the improvement of the standard deviation when the order of the piecewise controllers is increased. However, the benefits of the cubic controller lie in its ability to reduce discontinuities at agents equidistant from two neighbouring leaders, thus improving the overall cohesion of the swarm. Specifically, the cubic controller yields a standard deviation that is 40% lower than that of the quadratic controller, even though the difference in settling times between the two is only 22 simulation time steps.

The effect of increasing the proportion of leaders on the settling time of the formation can be observed in Figure 3.3. The simulations were performed with the number of leaders, $N_l = \{5, 8, 10, 20, 25, 40\}$, in a MAS consisting of $N = 1000$ agents. Mirroring the results analysed in the last section, the settling times are significantly influenced by the choice of

the global controller when $N_l = \{5, 8\}$. In these cases, controllers using higher-order piecewise state approximations demonstrate progressively better performance. On the other hand, the performance of the controllers converges to settling times within ±40 time steps after 10 leaders in the system. As the number of leaders increases, the controllers exhibit increasingly similar behaviour. Analogous to how a signal can be reconstructed more accurately with a higher number of sampled data points, an increase in the leader agents results in an increased awareness of the target curve by the MAS. Therefore the choice of the controller, and hence the connectivity required amongst leaders, is crucial to the performance of the MAS only when the number of leaders is below a certain threshold, i.e. $N_l \approx 10$ in this case. The correlation of this threshold with the inter-agent separation and the local dynamics of the agents requires further investigation.

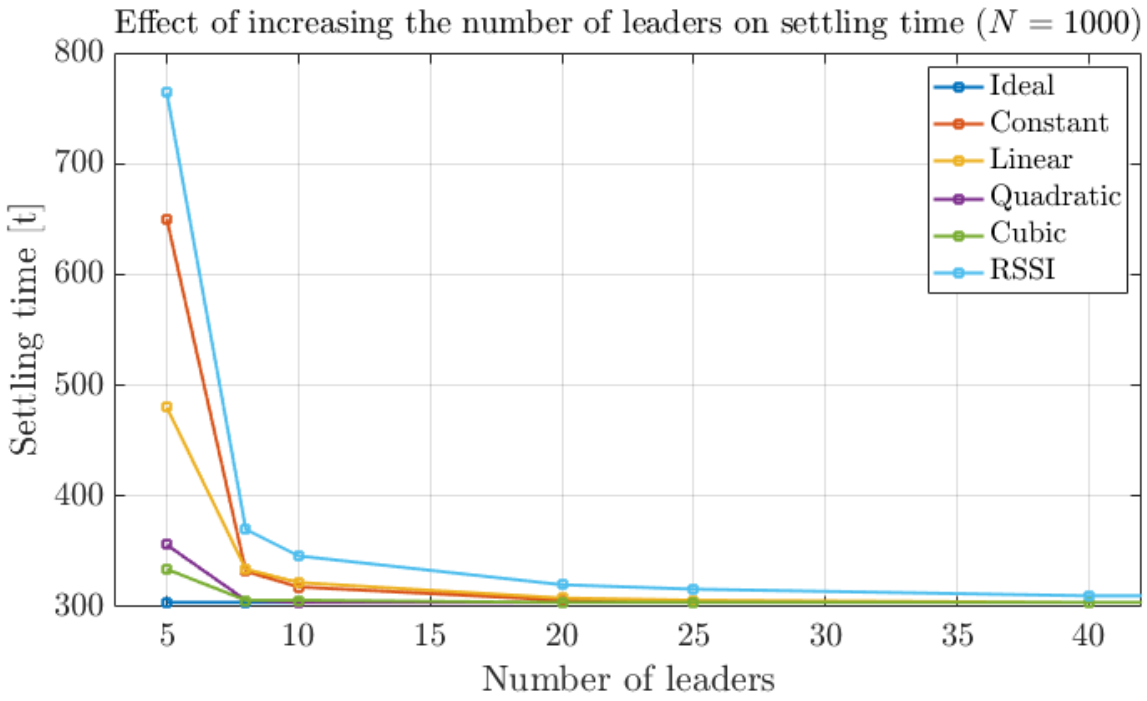


Figure 3.3: Effect of increasing the number of leaders and using different global controllers on MAS performance. The sinusoidal initial condition $z_i(0) = 10\sin(\frac{2\pi x_i}{l})$. Settling time reduces as the proportion of the leader agents increases. The total number of agents $(N) = 1000$.

The simulations of the MAS are also carried out under different initial conditions, namely a third-order initial condition, given by $z_i(0) = 10x_i^2(l - x_i)$, in Figure 3.4 and a second-order initial condition, given by $z_i(0) = 10x_i(l - x_i)$, in Figure 3.5. When a third-order initial condition is used, a cubic state approximation provides the actual states of the followers. Thus the piecewise cubic controller matches the performance of the ideal controller in Figure 3.4. Similarly, when a simpler, second-order, initial condition is used, both quadratic and cubic state approximations provide the actual states of the followers, as observed in Figure 3.5. Hence, the

order of the global controller required for faster settling times is directly correlated with the complexity of the initial/target formation. The impact of the formation's complexity becomes particularly apparent when non-zero target curves are considered, as studied in the following section.

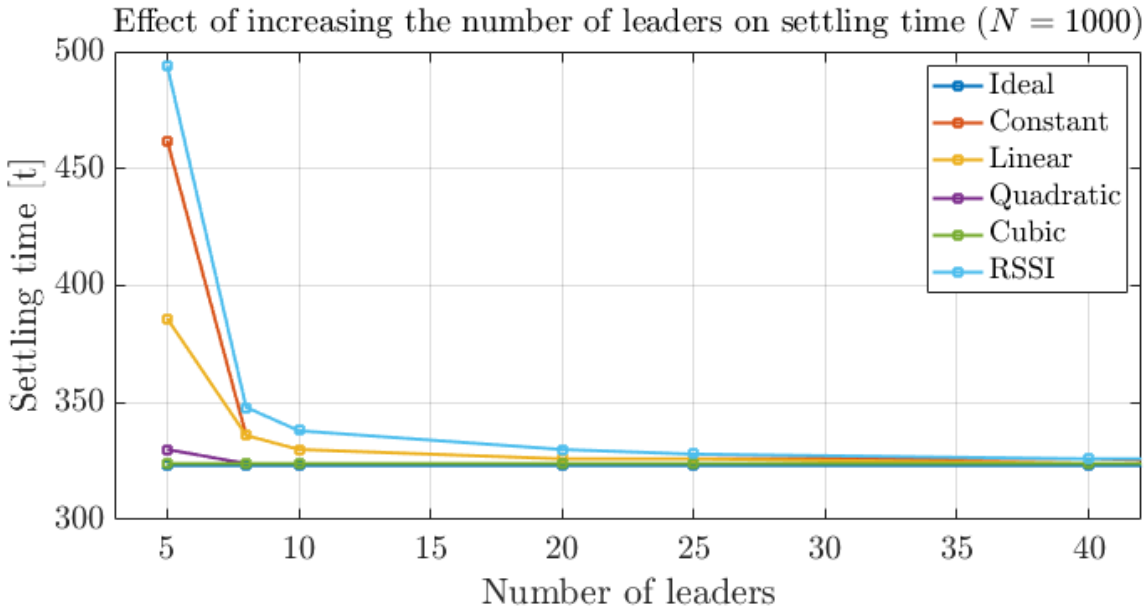


Figure 3.4: Effect of increasing the number of leaders and using different global controllers on MAS performance. The **third-order (cubic) initial condition**, $z_i(0) = 10x_i^2(l - x_i)$, is used.

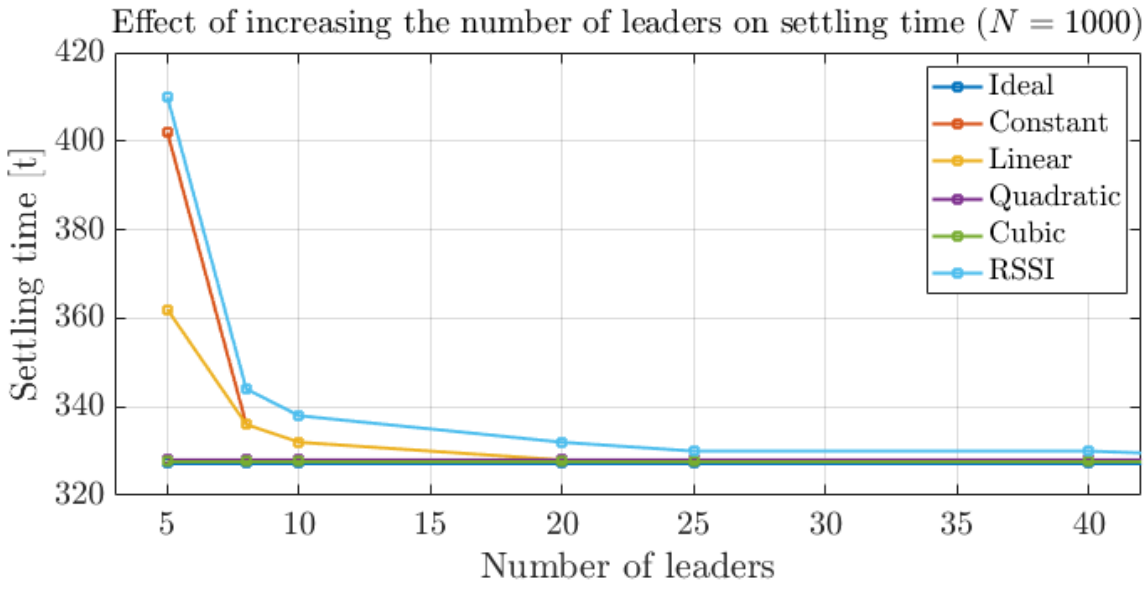


Figure 3.5: Effect of increasing the number of leaders and using different global controllers on MAS performance. The **second-order (parabolic) initial condition**, $z_i(0) = 10x_i(l - x_i)$, is used.

3.4 Non-zero target curve tracking

The previous sections focused on regulation control, where the agents' states converge to zero at steady-state. In order to track a non-zero target, where the desired states of all agents may not be the same, the states of the leaders can be replaced with their relative positions to the desired curve, as shown in (3.12), so that $\lim_{t\to\infty} z_i(t) = \gamma_i$. $\gamma_i \in \mathbb{R}$ is defined as the prescribed position on a time-invariant target curve $\gamma \in \mathbb{R}^2$ for the ith agent.

$$z_i(t) \to z_i(t) - \gamma_i \tag{3.12}$$

Thus, the multi-agent system described in (3.1) can be re-defined as (3.13). The time dependency of $z_i(t)$ has been omitted for brevity.

$$\dot{z}_i = \kappa \frac{[(z_{i-1} - z_i) - (\gamma_{i-1} - \gamma_i)] + [(z_{i+1} - z_i) - (\gamma_{i+1} - \gamma_i)]}{h^2} + a z_i + \nu(\hat{z}_i - \gamma_i) \tag{3.13}$$

The simulation results depicted in Figure 3.6 and 3.8 consider the initial conditions defined as $z_i(0) = 0$ for a target curve $\gamma_i = 10\sin(\frac{2\pi x_i}{l})$. Analysis of the L_2 norm performance for the five non-ideal global controllers in Figure 3.9 reveals that they cannot achieve convergence to the target curve with zero steady-state error. Such limitations arise because the follower agents do not have an awareness of their respective target positions and the approximated state measurements used in their global controllers are not sufficient to drive the formation accurately to the target curve.

When considering a sinusoidal target curve with $\gamma_i = 10\sin(\frac{2\pi x_i}{l})$ in Figure 3.9, the piecewise constant, linear, and quadratic controllers with $N_l = 5$ interpolating points provide progressively better approximations. The steady-state L_2 error of the quadratic and cubic controllers is 4.0 and 8.1, respectively, and they are the only controllers to settle within 95% of the target curve. The steady-state errors of the other controllers, in order of increasing magnitude, are 38.4 for the linear controller, 47.4 for the piecewise constant controller, and 62.4 for the RSSI-based controller.

Thus, for a MAS with a target formation (i) of polynomial order higher than the order of the global controller, and (ii) with interpolating points larger than the number of leader agents, there will be a residual error due to inaccuracies arising from the state approximations. Figure 3.7 depicts this discrepancy between a target curve and its piecewise approximations.

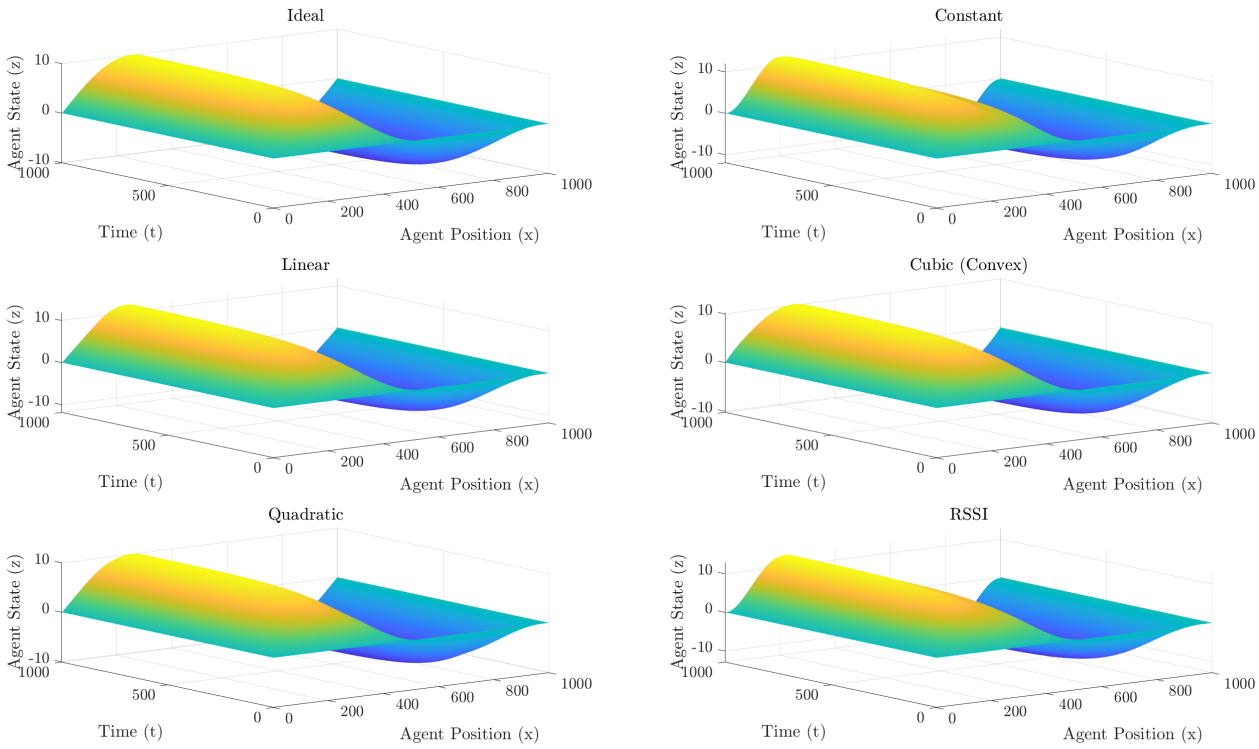


Figure 3.6: Performance of a MAS state feedback with different piecewise state approximations for the agents, $\gamma_i = 10\sin(\frac{2\pi x_i}{l})$ and $z_i(0) = 0$. $N_l = 5$ leader agents are evenly distributed along the Agent Position axis.

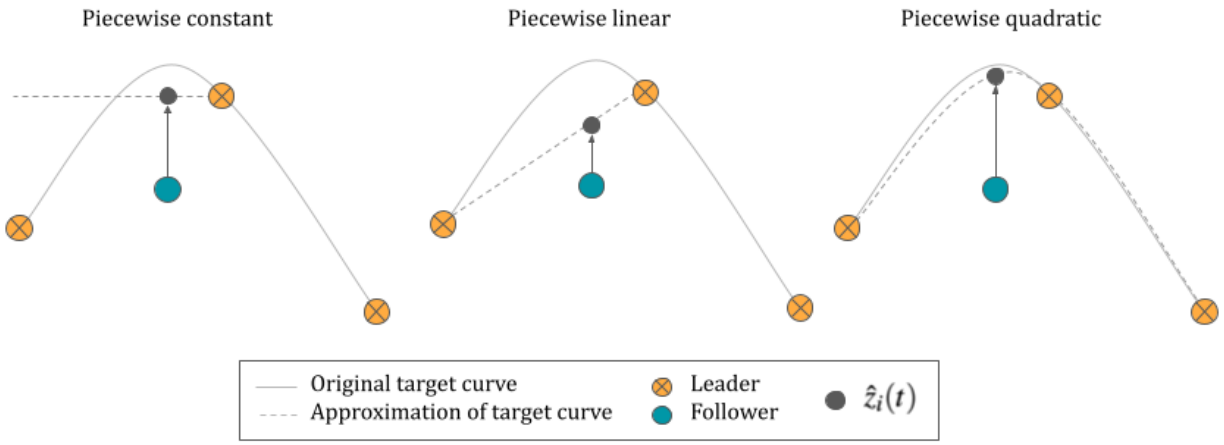


Figure 3.7: Illustration of the target curve approximation obtained when different piecewise state approximation functions are used for the followers.

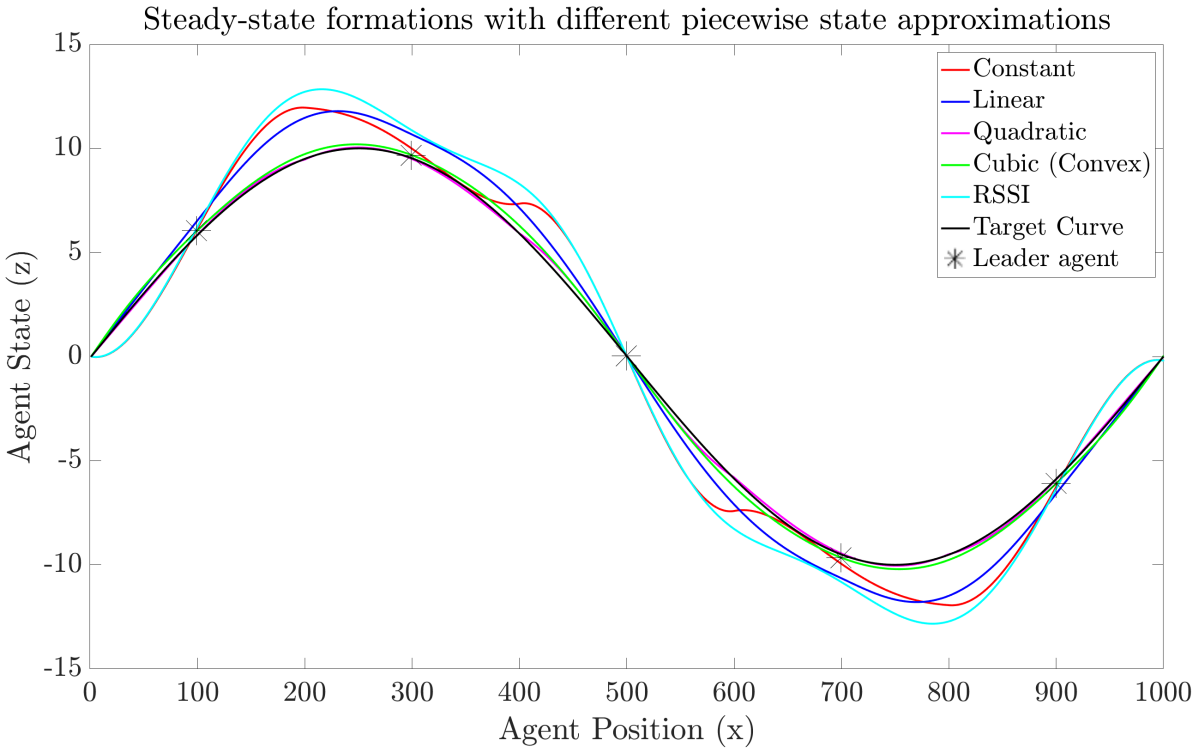


Figure 3.8: Comparison of the steady-state formations with different piecewise state approximations for the agents at t=1000; $\gamma_i = 10\sin(\frac{2\pi x_i}{l})$ and $z_i(0) = 0$. $N_l = 5$ leader agents are evenly distributed along the Agent Position axis.

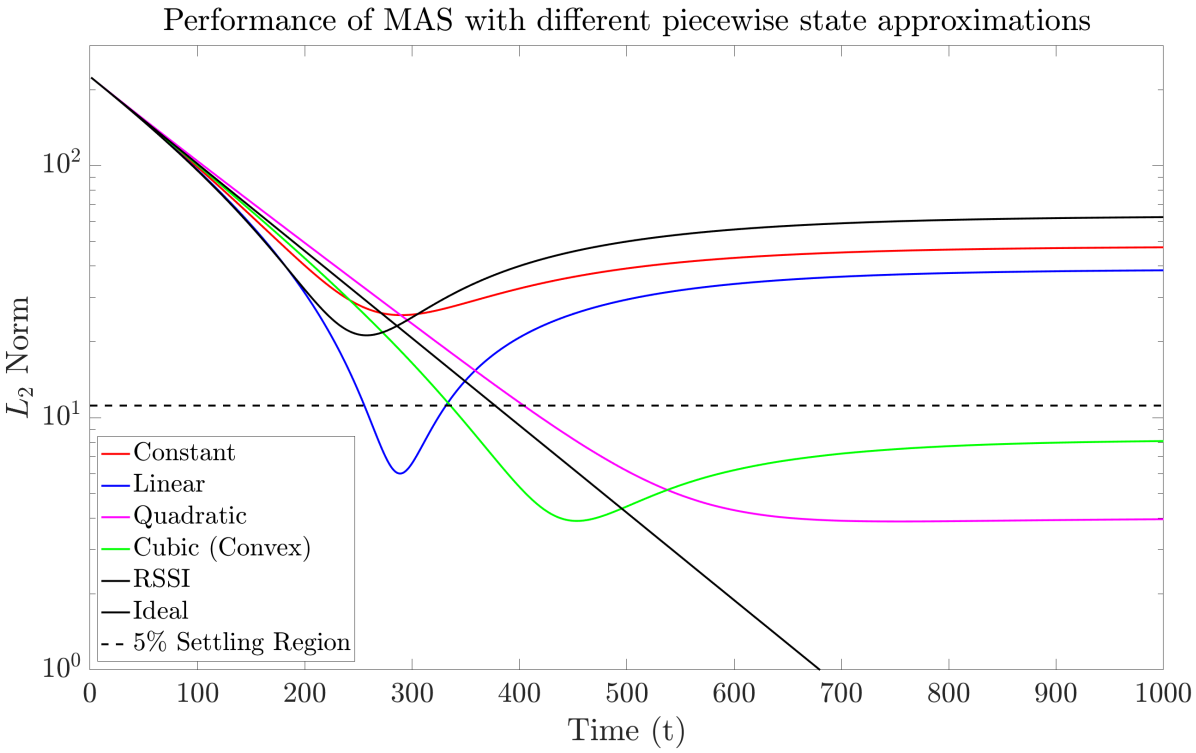


Figure 3.9: L_2 performance with different piecewise state approximations for the agents, $\gamma_i = 10\sin(\frac{2\pi x_i}{l})$ and $z_i(0) = 0$.

Chapter 4

Virtual Environment Simulations

The following chapter describes the implementation of the controllers described in Section 3.1.1 within a virtual environment, with a focus on discussing the implications of hardware constraints and leader connectivity on the performance of the MAS. E-puck robots are used in this study to model and simulate the swarm agents within virtual environment simulations. Developed originally for teaching purposes at École Polytechnique Fédérale de Lausanne, these miniature mobile robots are also available for research use at the Sheffield Robotics Laboratory. The hardware and software of the e-puck are entirely open-source, providing low-level access to electronic peripherals and offering sensory extension possibilities.

The e-puck robots, as shown in Figure 4.1, have a diameter of 70mm, a height of 55mm, and weigh 150g. They are equipped with a 32-bit STM32F407 168 MHz microcontroller with 192KB RAM and 1024KB Flash. Communication with the robots is possible via USB, Bluetooth 2.0, BLE, WiFi, and infrared transceivers. The robots use a differential drive and have two stepper motors with a 50:1 reduction gear. The periphery of the robot includes a 9-axis IMU, 10 LEDs, a speaker, and 4 microphones, amongst other features [43].

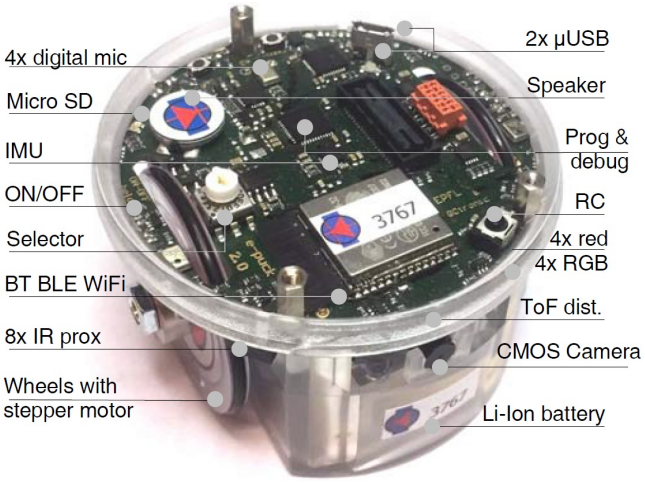


Figure 4.1: The e-puck mobile robot. [43]

The virtual environment simulations are performed using Webots software. Webots is an open-source 3D robot simulator used widely for industrial, educational, and research purposes. It provides a complete model of the e-puck robots' sensors and actuators [44], allowing researchers to test MAS algorithms in a virtual environment before deploying them on real robots. The simulations are executed on a MacOS machine with a 1.8 GHz Intel Core i5 processor and 1600 MHz 8 GB RAM, with a time-step of 64 ms (implementation details provided in Appendix A.2). A render of the simulation environment is shown in Figure 4.2.

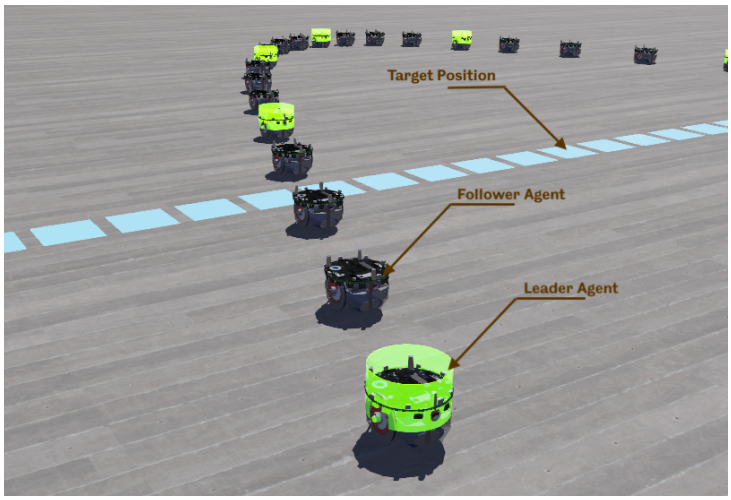


Figure 4.2: Rendering of the swarm agents (e-puck robots) and the target position in Webots. The leader agents are highlighted in green for illustration.

4.1 Hardware considerations

A physical implementation of the control laws discussed in Section 3.1.1 introduces a number of hardware constraints, which necessitates consideration of the agents' sensor, actuator, and overall system dynamics.

In order to achieve reliable MAS deployment, a low-latency and high-fidelity communication network is essential for implementing a leader-follower approach amongst agents. The communicated signals, however, will inevitably incur a time delay due to signal propagation and onboard signal processing requirements. Additionally, emitters and receivers have a limited aperture and range which bounds the region surrounding the agents within which the signals can be transmitted or received. Errors are also introduced due to multi-path signal propagation and interference. Consequently, if the communication topology is not well structured and channelised, it can lead to cross-talk between leaders and followers, eventually causing unpre-

dictable behaviour. Receivers may also have poor directional resolution and may be prone to sensory artefacts due to multiple reflections or grazing rays.

The external forces acting on the agents, such as friction, are non-linear, and the local dynamics of the robot may be second-order or higher, leading to modelling errors. The stepper motors on e-pucks have a maximum rotational velocity of 1200 steps/sec, i.e. 7.536 rad/sec, which can lead to a saturation of large input commands. Since the study primarily focuses on connectivity amongst agents, it does not take into account the impact of modelling, observation, and actuation errors on the design of multi-agent controllers. However, these constraints have been modelled in Webots, where appropriate, to minimise simulation errors.

Simulation of the agents is achieved by utilising a dimensionally and weight-accurate model of e-pucks, as provided by the Webots library. Contact friction is modelled with a static coefficient of 0.9 and a kinetic coefficient ranging between 0.5 to 0.8 for the frictional interaction between rubber (i.e., the e-puck tires) and dry asphalt (i.e., the deployment surface) [45]. The choice of rubber as the tire material is based on the use of rubber O-rings on e-pucks' wheels. Emitted signals are modelled to be omni-directional (infinite aperture), with a baud rate of 2 Mbit/s. The strength of a received signal is equal to the inverse squared distance between the emitter and the receiver. Furthermore, the robot body is modelled as a `bounded` object to realistically deflect infrared rays. The wheels are modelled to be perfectly rigid (without any suspension) and do not slip or skid.

4.1.1 Design modifications

In order to implement the local controller, every agent requires awareness of its local position, i.e. its separation from the neighbouring agents. The original e-puck robots have eight infrared proximity (distance) sensors installed symmetrically on the perimeter of the robot's chassis. These proximity sensors can only measure distances up to 5 cm accurately. Moreover, the usage of only eight sensors would introduce *blind-spots* if the separation distance between two robots was greater than 2.18 cm (4.1). Blind-spots refer to regions in the proximity of a robot where it cannot detect any objects, even if the objects are within the range of the proximity sensor (Figure 4.4). In (4.1), r is the radius of the e-pucks, n_s is the number of proximity sensors, and d_{max} is the maximum inter-agent distance before blind-spots appear. In the case of $n_s = 8$, it meant that if any e-pucks were separated by more than 2.18 cm, the local controller would no longer be operational.

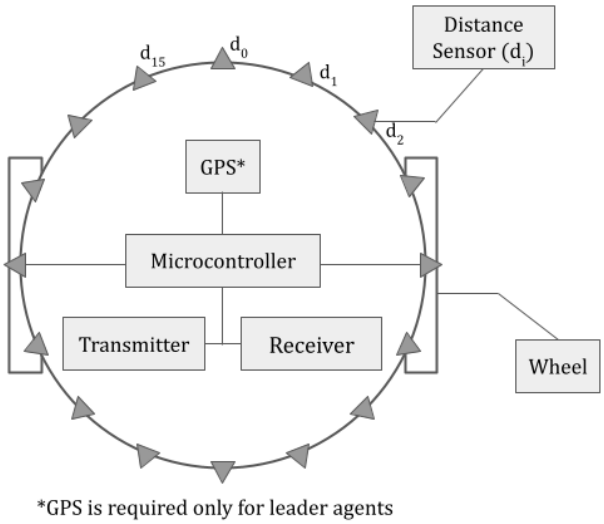


Figure 4.3: Top-view block-diagram of the modified e-puck robot. The robots were modified in Webots to include sixteen distance sensors on the perimeter to facilitate the implementation of local controllers.

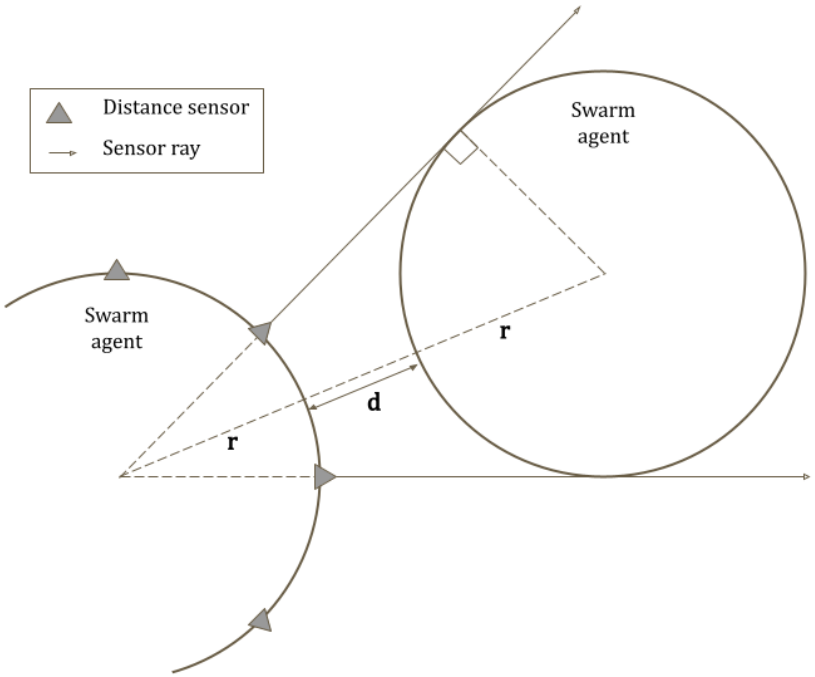


Figure 4.4: Example illustration for the case $n_s = 8$, where d (d_{max}) is the maximum separation distance; if the agents move further away, they will not detect each other's presence.

$$\sin\frac{\pi}{n_s} = \frac{r}{d_{max}+2r} \implies d_{max} = r\left(\csc\frac{\pi}{n_s} - 2\right) \tag{4.1}$$

Thus, as the usage of the sensors in their original configuration was not suited for the appli-

cations of this research, the proximity sensors are replaced with sixteen distance sensors, that can measure up to 25 cm and are placed symmetrically on the robot's perimeter as shown in Figure 4.3. A suitable hardware choice for such a sensor may be SHARP GP2Y0AF30, which can measure up to 24 cm [46]. With sixteen sensors, the maximum separation distance before blind-spots appear increases to 11.10 cm (4.1). The calculations are performed assuming that surface grazing and deflection of the infrared rays do not take place. It is worth noting that in order to maintain a positive value of d_{max} in (4.1), n must be greater than six. Therefore, a minimum of six proximity distance sensors is always required irrespective of the radius r.

4.1.2 Communication structure

A communication link between the agents can be established using infrared transceivers or Bluetooth Low Energy (BLE) modules, both of which are present on e-puck robots. BLE is suitable for applications that require low power consumption, high data rates of up to 2 Mbit/s, and long-range communication of up to 100 m, whereas an infrared transceiver is suitable for applications that require low-cost, and interference-free communication over short distances with direct line-of-sight. As a direct line-of-sight may not be available to all intended recipients (followers) of a message sent by the leader agents, BLE communication is the preferred method in this study. The communication between all agents occurs on the same frequency bandwidth, i.e. 2.4 GHz in the case of BLE. Although this increases the likelihood of interference and cross-talk, it demands a simpler hardware design rather than requiring the transmission and reception of signals on different bandwidths. Errors introduced in the data packets due to interference can be detected using software-based check-sum solutions, such as cyclic redundancy checks (CRC), Hamming codes and parity bits.

In order to differentiate leaders and followers agents, every agent in the system is assigned a unique identification number (UID). The agents only process the payload of the data packets addressed to them, i.e. packets that include their UID. Thus, every data packet consists of three components, the receiver's UID, the payload (agent's state information), and error correction bit(s). The method described is equivalent to a decentralised publish/subscribe messaging pattern commonly used in distributed systems, where agents publish messages to a channel, and other agents subscribe to the same channel to receive the messages [47].

The communication topology for the various global controllers has been depicted in Figure 4.5. In order to implement a piecewise constant controller, the leaders do not need to communicate

with any neighbouring leaders - they simply advertise their measured states to their follower agents. On the other hand, piecewise linear, quadratic, and cubic (convex) require the leaders to (i) communicate with one, two, or four neighbouring leaders respectively, (ii) estimate the state of each follower via the interpolation equations described in Section 3.1.1, and (iii) relay the interpolated state to their respective followers. All piecewise controllers require the leaders to be aware of the UID of all follower agents assigned to them.

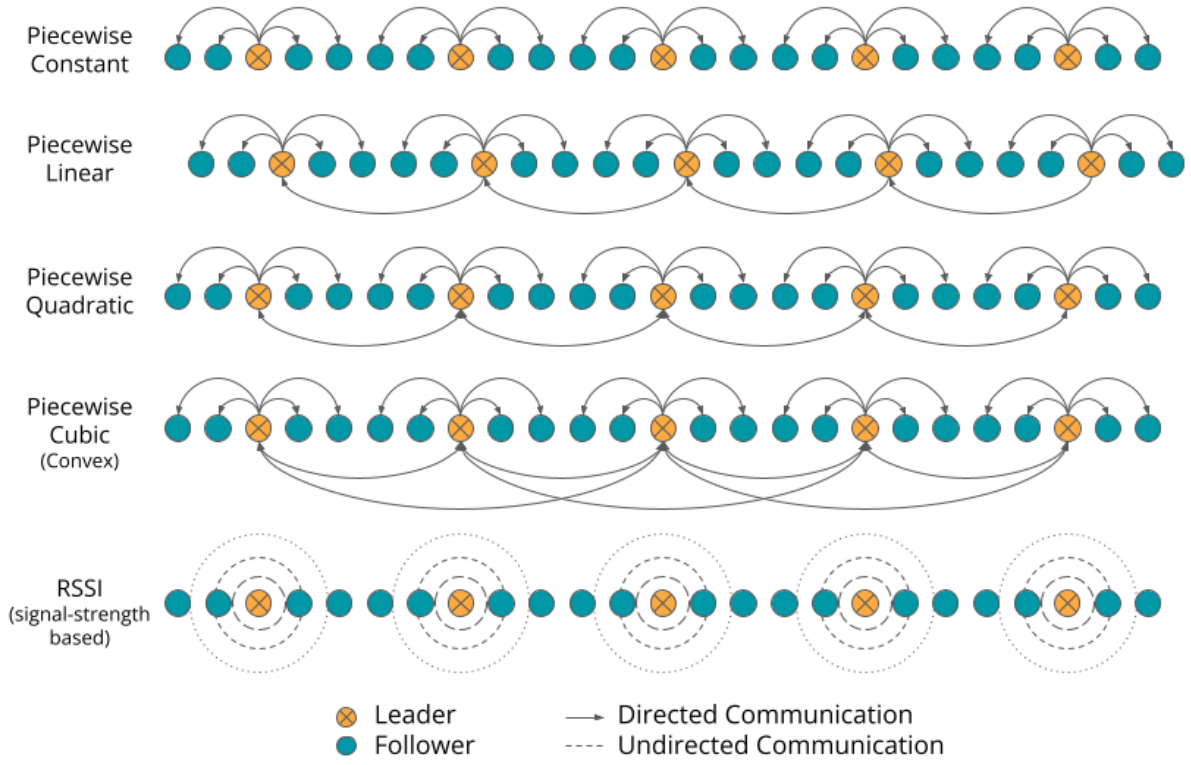


Figure 4.5: Communication topology required for the piecewise and signal-strength based state approximation global controllers.

Communication processing overheads in multi-agent systems are directly proportional to the number of communication links maintained by each agent. The time interval to establish a new connection is approximately 20 ms and is required every time an agent switches to a different leader or transitions between transmitting and receiving data [48]. Thus, using higher-order piecewise controllers which demand wider connectivity with the neighbouring leaders, would have adverse consequences on the minimum controller time-step and the communication latency for the agents and, consequently, on the settling time of a formation. Interested readers may refer to [49] for improved management of coexisting BLE connections. Communication overheads, including the time required for transitioning between connections, have not been modelled in this study in order to isolate the impact of the global controller's state approxima-

tion from the processing overheads on the performance of a MAS.

The RSSI-based controller, in contrast to the piecewise controllers, simplifies the communication structure by requiring undirected (broadcast) communication. The leader agent broadcasts its current state and the follower agents listen for any incoming data packets. The followers continually update their state approximation based on the signal strength of the data packets received as described in (3.9) and (4.2). The signal strength (-dBm) is included in the BLE protocol's advertisement data, as a signed 8-bit integer value, when a BLE Scan is performed. Although the accuracy of RSSI as an indicator of distance may be affected by environmental conditions, the RSSI-based controller may remain suitable because the state approximation in (3.9) relies on the ratio of the received signal strengths — effectively cancelling out any fluctuations in the RSSI values.

Thus, every data packet in RSSI-based controllers consists solely of two components, the payload and error correction bit(s). This eliminates the need for each agent to possess a UID, and for the leaders to have knowledge of the number or type of agents (leaders or followers) in their proximity. Furthermore, this method is fault-tolerant, as it can be extended to enable the followers to broadcast their estimated positions to other followers in the absence of sufficient leaders, or if a leader ceases to operate.

4.2 Implementation of the controllers

The following section describes the implementation of the global and the local controllers defined in (3.1) utilising the sensors present on the modified e-puck robot.

4.2.1 Global controller for leaders

The e-puck robot is equipped with a time-of-flight (TOF) distance sensor and a Global Positioning System (GPS) module, which enable the measurement of every leader agent's distance to the target curve.

Positioned at the front of the robot, the infrared TOF sensor can measure distances up to 2m with a line-of-sight measurement error of $<$ 1cm. Lower power consumption (0.05W) and simpler software implementation make it beneficial for agents with limited battery life and processing capabilities. However, this type of sensor has limitations, as it requires the target to be

physically detectable by infrared rays and to be within the sensor's measurement range. The limited aperture also restricts any rotational movements for the leader, which could result in the target not being captured by the sensor. Additionally, TOF sensors cannot differentiate targets from obstacles or other agents in the environment, which may lead to erroneous target tracking. As a result, visual sensors such as cameras may be required to accurately identify the target's location. The TOF sensor can, however, be used in conjunction with GPS to provide accurate position information, especially when the leader is in close proximity to the target and when environmental obstacles impede GPS signals.

The GPS module is used in this study to provide the leaders with positional awareness in the global coordinate system due to the aforementioned reasons. The relative displacement to the target is obtained by setting $z_i(t)$ with the leader's current GPS coordinates and γ_i with the target coordinates for the leader in (3.12). If formation control within 2-D space is required, the relative orientation to the target also needs to be included in the global controller, as shown in (4.6).

4.2.2 Global controller for followers

The discrete-time equivalents of the controller choices discussed in Section 3.1.1 are used to approximate the relative position of the followers to the target. Here t is replaced with kT_{sim}, where T_{sim} is the simulation update period used in Webots (64 ms), and $k \in \mathbb{N}$ represents the current simulation step. It is important to note that the application of these controllers requires the communication structures discussed in Section 4.1.2 to be in place. If a leader does not receive one or more of the required positions from its neighbouring leaders (due to the signal being noise corrupted, processor overloading, inability to manage multiple connections or other reasons), it reverts to a lower-order (piecewise constant or linear) approximation to avoid any ill-conditioned interpolation.

To implement the RSSI-based global controller for follower agents in a real-time environment, where leaders broadcast data packets asynchronously and in no particular order, (3.9) can be modified to the equivalent form presented in (4.2). This modification allows the followers to update their state estimate as they receive signals from neighbouring leaders instead of waiting to receive a signal from all leaders, the number of which they may not be aware of.

$$\hat{z}_i[k] = \bar{\lambda}_j z_j + (1 - \bar{\lambda}_j)\hat{z}_i[k-1], \quad \hat{z}_i[0] = 0 \tag{4.2}$$

$\bar{\lambda}_j$ is the ratio of the received signal strength A_j, for the state z_j of a given neighbouring leader ρ_j, to the maximum received signal strength A_{max} that can be possible based on inter-agent distances, as defined in (4.3).

$$\bar{\lambda}_j = \frac{A_j}{A_{max}}, \quad A_{max} = \frac{1}{h^2} \tag{4.3}$$

(4.2) is equivalent to the ideal RSSI equation defined in (3.9) with $\sum_{m=0}^{J} A_m$ being replaced with A_{max} as a normalisation factor for A_j, and the summation of the received signals over $j \in J$ being replaced with a step update equation, to update the state approximation for the followers as and when new signals are received.

4.2.3 Local controller

The local controller implementation is identical for both leaders and followers. Implementation of the local controller in (3.2) requires the calculation of displacement of each agent to its neighbours. Once the position sensors on the e-puck robot are modified as described in Section 4.1.1, these displacements can be obtained using (4.4) for the ith agent.

$$\begin{aligned} \Delta_{r_i}[k] &= z_{i+1}[k] - z_i[k] = \min\left(d_\iota[k]\sin\left(\frac{2\pi\iota}{n_s}\right)\right), \quad \iota = 0, 1, \ldots, \frac{n_s}{2} - 1 \\ \Delta_{l_i}[k] &= z_{i-1}[k] - z_i[k] = \min\left(d_\iota[k]\sin\left(\frac{2\pi\iota}{n_s}\right)\right), \quad \iota = \frac{n_s}{2}, \frac{n_s}{2} + 1, \ldots, n_s - 1 \end{aligned} \tag{4.4}$$

Δ_{r_i} and Δ_{l_i} are the perpendicular displacements to the right and the left neighbouring agents respectively. Here d_ι, i.e. the distance measured by the ιth proximity sensor, is projected onto the axis perpendicular to the axis of deployment of the agents.

4.2.4 State update

The input command for the rotational speed of the e-puck's left and right wheels at time step k can be obtained by combining the global (3.1.1) and local (4.4) controllers, resulting in (4.5).

$$w_{l_i}[k] = w_{r_i}[k] = \kappa\frac{\Delta_{l_i}[k] + \Delta_{r_i}[k]}{h^2} - \nu||\hat{z}_i[k] - \gamma_i|| \tag{4.5}$$

$w_{l_i}[k]$ and $w_{r_i}[k]$ are the left and right wheel speed commands in radians per second for the ith agent respectively. An illustration of the symbols used in (4.5) is provided in Figure 4.6. The state measurements and approximations used in the controllers are prone to measurement and observation errors such as quantisation, grazing, and fluctuating signal strength. To reduce the

introduction of erratic outputs on the agent's states, a moving average filter with a window size of 3 is also applied to w_{l_i} and w_{r_i}.

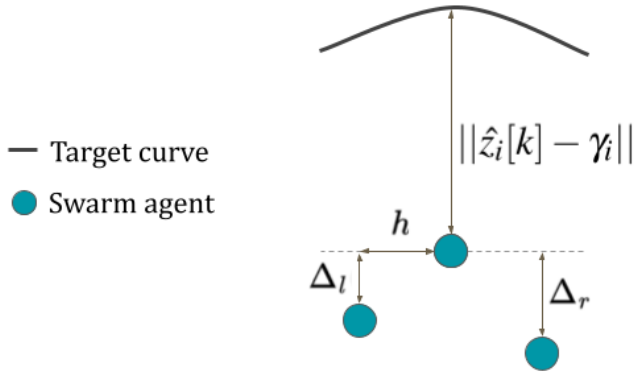


Figure 4.6: An illustration of the symbols used for the state update equation in (4.5).

4.3 Deployment in two-dimensional manifolds

Movements of the agents in two-dimensional Euclidean space can be achieved by extending (4.5) to also include a target bearing for each agent along with the straight-line distance to the target position. This results in separate commands for the left and the right wheel speeds, as shown in (4.6). The updated speed command controls the distance as well as the orientation to the target position for each agent. In (4.6), γ_i and $\hat{z}_i[k]$ are 2-D vectors representing the position of the target and the position of the ith agent in $\mathbb{R}^2$ respectively. A derivation of the global controller defined in (4.6) using polar transformation can be found in [50].

$$
\begin{aligned}
w_{l_i}[k] &= \kappa \frac{\Delta_{l_i}[k] + \Delta_{r_i}[k]}{h^2} - \nu \left(||\hat{z}_i[k] - \gamma_i|| - \tan^{-1} \frac{\hat{z}_{i_x}[k] - \gamma_{i_x}}{\hat{z}_{i_y}[k] - \gamma_{i_y}} \right) \\
w_{r_i}[k] &= \kappa \frac{\Delta_{l_i}[k] + \Delta_{r_i}[k]}{h^2} - \nu \left(||\hat{z}_i[k] - \gamma_i|| + \tan^{-1} \frac{\hat{z}_{i_x}[k] - \gamma_{i_x}}{\hat{z}_{i_y}[k] - \gamma_{i_y}} \right)
\end{aligned}
\tag{4.6}
$$

An illustration of two-dimensional deployment in Webots is provided in Figure 4.7. The transient performance of the various global controllers and deployment on 2-D targets has also been compiled in a simulation footage that can be found in [51].

4.3.1 Collision avoidance

Avoidance of inter-agent collision is of particular importance when swarm agents are deployed onto two-dimensional planar curves and thus is an active area of research. A widely used

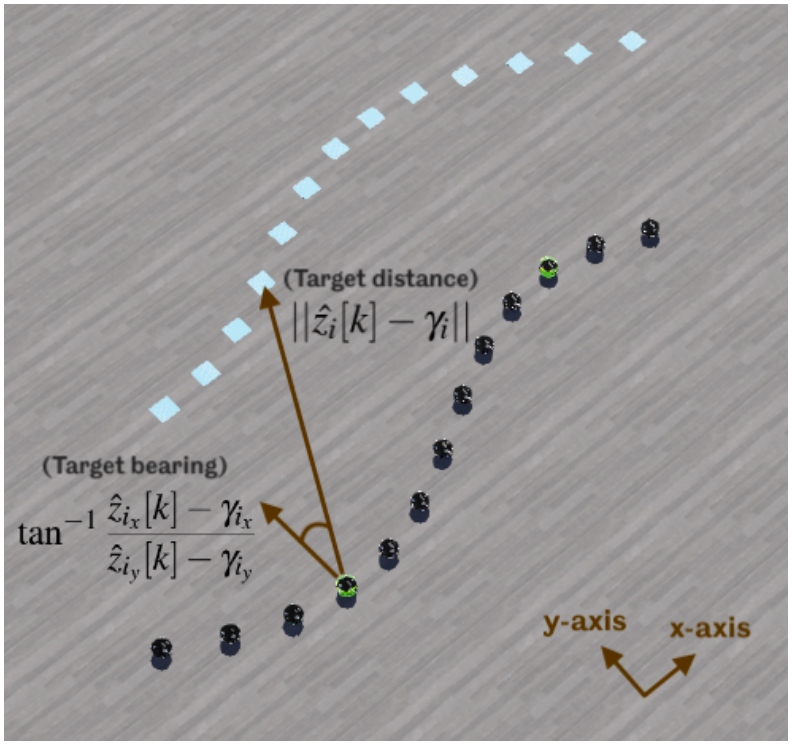


Figure 4.7: The initial state of the MAS in Webots; the desired position (target) for each agent i is illustrated with blue squares in the simulation arena.

potential-fields-based collision avoidance technique for MAS was developed in [52]. The state update command for a given agent was derived from the gradient of its potential field, which is computed as the negative gradient of the sum of attractive potentials (towards the target) and repulsive potentials (away from nearby agents). Other popular techniques of collision avoidance include velocity obstacle [53], model predictive control of inter-agent distances [54], and reciprocal velocity obstacle [55].

Due to its ease of implementation, a vector inversion strategy is used for obstacle avoidance in this study. The agents detect obstacles using their infrared distance sensors and then compute a vector in a direction opposite to the detected obstacles to move away from them. This is done by replacing the term defining the target bearing, $\tan^{-1}\left(\frac{\hat{z}_{i_x}[k]-\gamma_{i_x}}{\hat{z}_{i_y}[k]-\gamma_{i_y}}\right)$, in (4.6) with the collision avoidance bearing defined as $\frac{2\pi i}{n_s} + \pi$ where $i = \{\arg\ \min_i\ d_i \quad | \quad d_i < h_{collision}\}$, i.e. the index of the proximity distance sensor that detects a distance less than $h_{collision}$. $h_{collision}$ is defined as the inter-agent distance threshold below which the collision avoidance bearing takes priority over the target bearing. $h_{collision} = 0.01$m has been used for the simulations in this study.

4.4 Simulation results

The following section describes the simulation configuration in Webots, followed by an analysis of the results obtained. The agents are placed at a step of $h = 0.1$m in the x-direction. The number of leaders is tested with N_l being 2, 5, and 10. The total number of agents is set to $N = 40$. The system dynamics $f(t, z_i(t))$ in (3.1) are modelled by Webots as described in Section 4.1. The simulation results in all subsequent figures are produced with an initial function $z_i(0) = 3.5\sin(\frac{2\pi x_i}{l})$, where $l = hN$. The amplitude of the initial function is chosen such that the maximum separation between the agents is within the range of the distance sensors, so that the local controller remains functional. The controller gains for the MAS in (3.1) are arbitrarily defined as $\nu = 20, \kappa = 0.5$. Increasing κ in an attempt to improve the swarm's cohesion may seem reasonable, however, doing so would amplify the measurement noise from the proximity distance sensors. The effect of varying the controller gains is further discussed in Section 4.5.1.

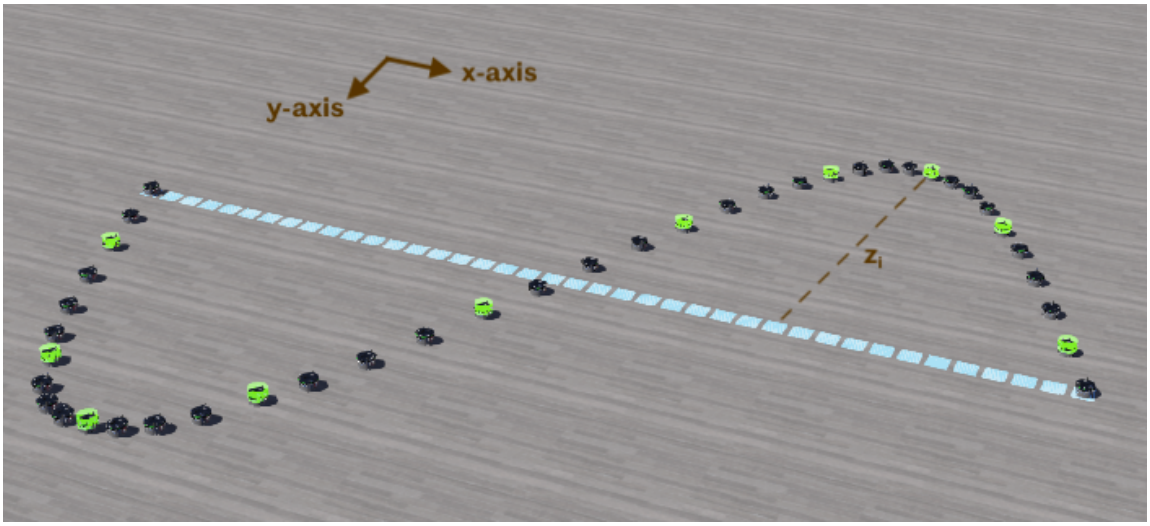


Figure 4.8: The initial state of the MAS in Webots; the desired position (target) for each agent i is illustrated with blue squares in the simulation arena.

The cubic controller has not been implemented because it requires four leader agents in the proximity of each leader agent. Such a simulation would not be representative with the limited number of agents that have been simulated in Webots. Additionally, a cubic approximation requires the reception of all four neighbouring leaders' positions to perform interpolation for the followers. If any neighbour's position is not received, it can make the interpolation equation singular and lead to unpredictable state approximations.

The performance indices from Figure 4.9 have been summarised in Table 4.1.

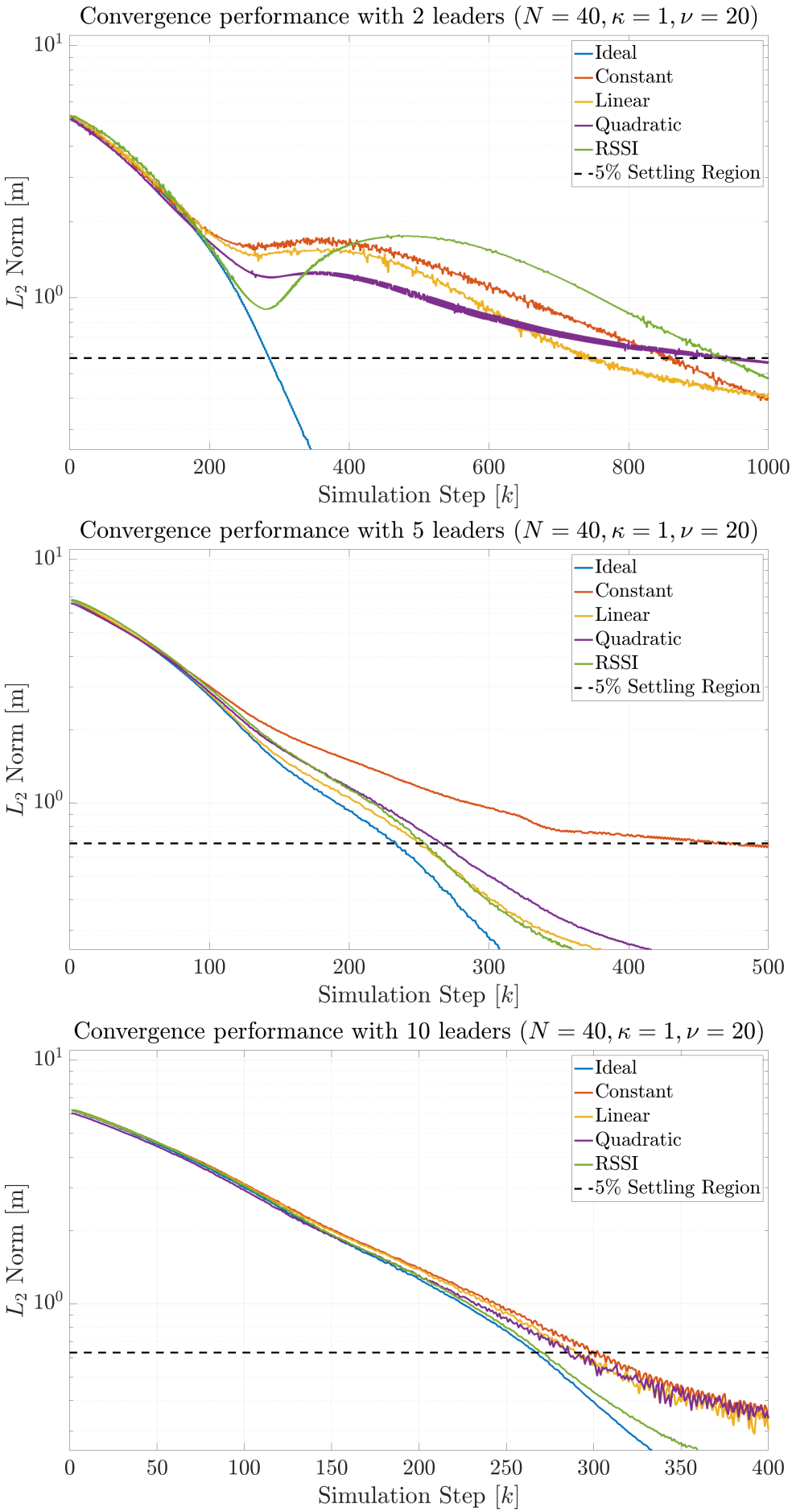


Figure 4.9: L_2 performance of the MAS simulated in Webots with different global controllers and number of leaders.

Performance Index	Global Controller				
	Ideal	Constant	Linear	Quadratic	RSSI
Leader Connectivity[1]	-	0	1	2	0
Settling Time[2] ($N_l = 2$)	284	847	742	931	928
Settling Time[2] ($N_l = 5$)	230	457	249	254	251
Settling Time[2] ($N_l = 10$)	267	297	289	286	271
Simulation Step Time [ms][3]	6.682	5.301	7.184	7.699	7.263

Table 4.1: Webots simulation performance indices.

[1] The number of neighbouring leaders a given leader agent is required to communicate with. It is assumed that the leaders have prior knowledge of the target curve and do not need to communicate their intentions with the rest of the leaders beforehand.
[2] Number of simulation steps required for the L_2 error norm to reduce to less than 95% of the initial error value.
[3] The average simulation step time [ms] for executing the controllers for the agents.

4.5 Analysis of controller performance

When the swarm has only two leaders, it initially converges quickly for all piecewise and RSSI controllers, exhibiting a similar rate to the ideal controller. However, the convergence slows down after $k = 200$, as observed by the plateauing of the L_2 norm in Figure 4.9. This can be attributed to poor state estimation by the followers' global controllers, resulting in undershooting and overshooting of the target. At this point, the local controller's dominance increases, where it continues driving the MAS to zero steady-state error. To reduce these plateaus, the cohesion of the swarm can be increased by raising the local controller gain, although this may result in longer settling times (4.5.1). Settling times for the case of two leaders in Figure 4.9 also differ based on the controllers, with the linear controller settling 105 time-steps faster than the constant controller, while the RSSI-based and quadratic controllers have similar settling times.

A significant improvement in settling time is observed for all controllers when the number of leaders increased to $N_l = 5$. The constant, linear, quadratic, and RSSI controllers show a reduction of 59.7%, 95.9%, 96.3%, and 96.7%, respectively, when compared to the ideal controller's settling time. However, the constant controller still performs poorly due to the poor state approximation of the followers. As the system converges to the target, the discontinuities in the piecewise-constant state-approximation of the followers increase, further slowing down the swarm's convergence [51]. As the number of leaders increases to 10, the performance rel-

ative to the ideal controller improves further, and all controllers perform similarly against the target curve. This trend is similar to the results observed in Figure 3.3, where the settling times of all controllers converge to similar values as the number of leaders increases.

It can be observed that higher-order piecewise approximations perform successively better, with the linear controller outperforming the constant and the linear and quadratic controllers showing similar performance. In contrast, the performance of the RSSI controller gradually improves and eventually outperforms the constant, linear, and quadratic controllers as more leaders are added. This is because the estimation of the follower's state is influenced by more leaders, resulting in a higher-order signal-strength based approximation. Notably, the settling time of the RSSI controller differs from the ideal controller only by 4 time steps when 10 leaders are present.

4.5.1 Impact of the local controller

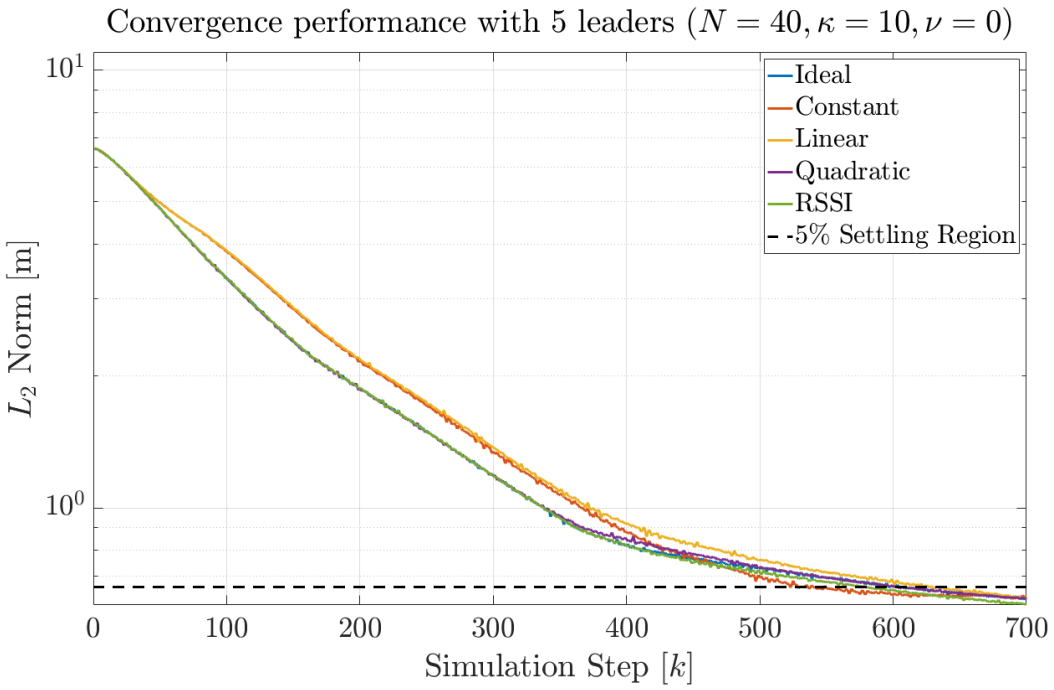


Figure 4.10: L_2 performance of the MAS simulated in Webots in the presence of only a local controller.

Figure 4.10 depicts the convergence of the L_2 norm of the MAS in the presence of only a local controller, i.e., when the global controller gain is set to zero. Doing so helps in comprehending the impact of a local controller on the performance of the swarm. This topology is equivalent to distance-based control described in [8], where only inter-agent distances are controlled. The

MAS configuration consists of 5 leaders and 40 agents. The convergence achieved is slower compared to the case with 5 leaders in Figure 4.9, but zero steady-state error is still achieved. The settling times achieved by all the controllers in this configuration are larger than the worst settling time (from piecewise-constant) observed in Figure 4.9 for $N_l = 5$. Additionally, the settling times for all controllers are within 600 ± 50 time steps because the local controller of the agents is independent of the type of global controller used. The primary contribution of the local controller is to ensure cohesion and open-loop stability of the swarm by minimising discontinuities in inter-agent separation.

4.5.2 Impact of the global controller

Figure 4.11 depicts the convergence of the L_2 norm of the MAS in the presence of only a global controller, i.e., when the local controller gain is set to zero. This topology is equivalent to position-based control described in [8], where only the agents' distances to the targets are controlled. The MAS configuration consists of 5 leaders and 40 agents. The initial convergence achieved with only a global controller ($k < 150$) is similar to the case with 5 leaders in Figure 4.9. However, zero steady-state error is not achieved in this case since cohesion (continuity amongst inter-agent distances) is not enforced. Therefore, global controllers offer faster transience but at the cost of errors in state estimation becoming evident at steady-state. It is observed that higher leader connectivity results in a lower steady-state error, with the performance increment reducing substantially beyond a linear controller. Additionally, the RSSI controller achieves a steady-state error lower than the quadratic controller in this scenario.

The boundary conditions also contribute significantly to the steady-state error of the piecewise controllers. Leaders on the boundary, who lack neighbouring leaders on one side, always resort to a constant controller, regardless of the global controller used for leaders present away from the boundary. This results in poor state approximation for the global controllers of follower agents near the boundary. While adding more leaders can mitigate formation errors at the boundary, it may not always be feasible during a deployment.

Figure 4.12 illustrates the steady-state formation error in the presence of a constant controller and no local controller, highlighting the importance of the local controller in driving steady-state errors to zero. The figure also shows the influence of initial conditions on steady-state performance, where the followers maintain the shape of the initial curve near their respective leader due to inaccuracies in the piecewise constant controller's approximation.

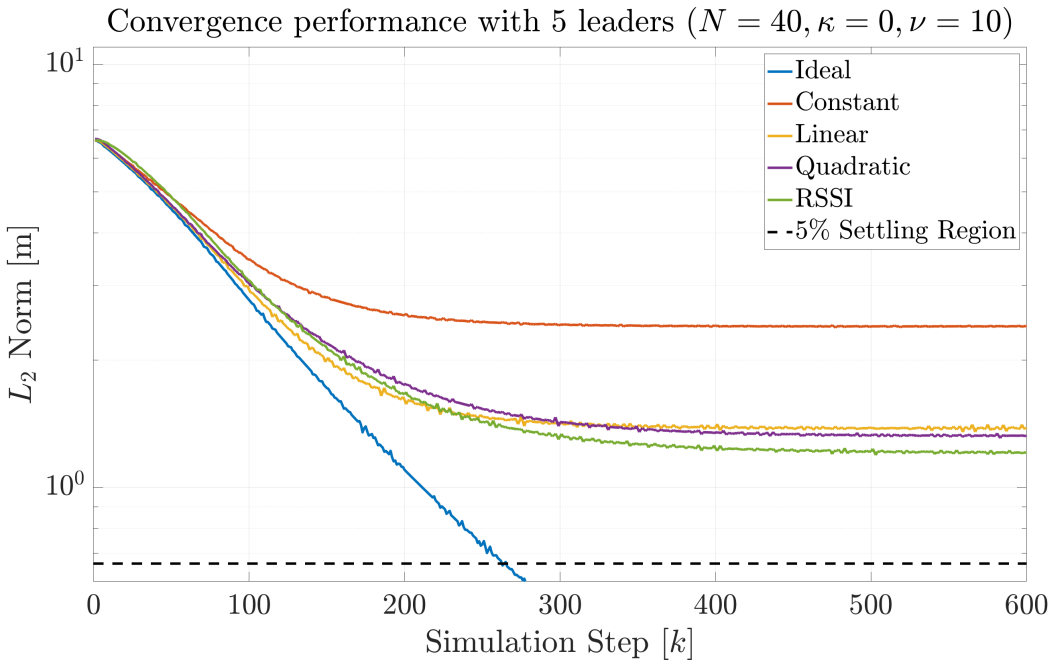


Figure 4.11: L_2 performance of the MAS simulated in Webots in the presence of only a global controller.

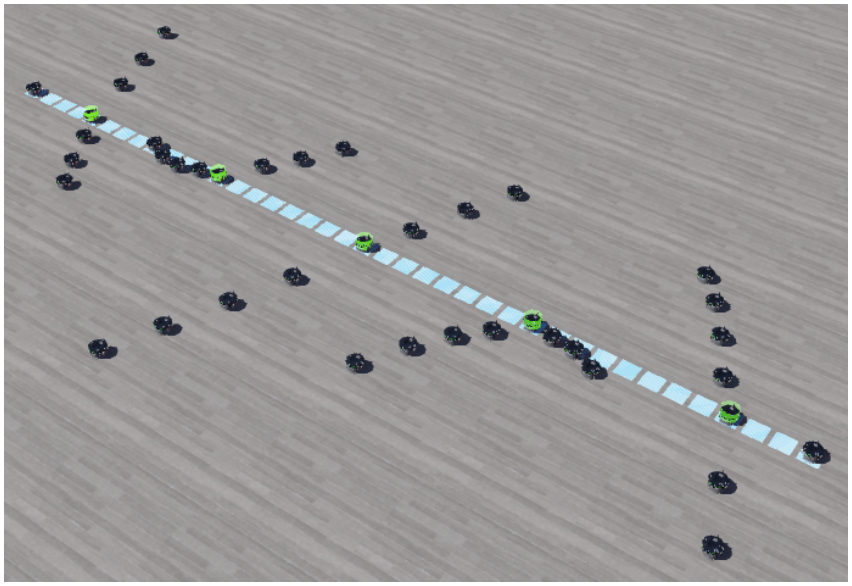


Figure 4.12: Steady-state formation with a piecewise constant global controller and no local controller. Initial conditions were equivalent to those in Figure 4.8 and the target curve is highlighted by the blue squares. The follower agents maintain the shape of the sinusoidal initial condition in their leader's vicinity due to the shortcoming of the piecewise constant global controller. The usage of a local controller would enforce cohesion amongst the agents by minimising inter-agent separations (discontinuities) and driving the steady-state error to zero.

Chapter 5

Conclusion

This study proposes two classes of global controllers, namely piecewise and signal-strength based (RSSI), and analyses their performance for a MAS using MATLAB and Webots. The piecewise controller is evaluated in multiple polynomial orders, including piecewise constant, linear, quadratic, and cubic. Results show that higher-order piecewise controllers improve performance successively, but the overall performance gains beyond a linear controller are minimal — especially when considering controller complexity and the cost of implementation. The RSSI controller performs similarly to the constant controller for a few leaders but surpasses higher-order piecewise controllers as the number of leaders increases. Therefore, if communication and controller complexity are to be minimised, an RSSI controller is generally preferred, whereas a higher-order piecewise controller should be preferred to improve settling performance with a minimum number of leaders. This has been illustrated in Figure 5.1.

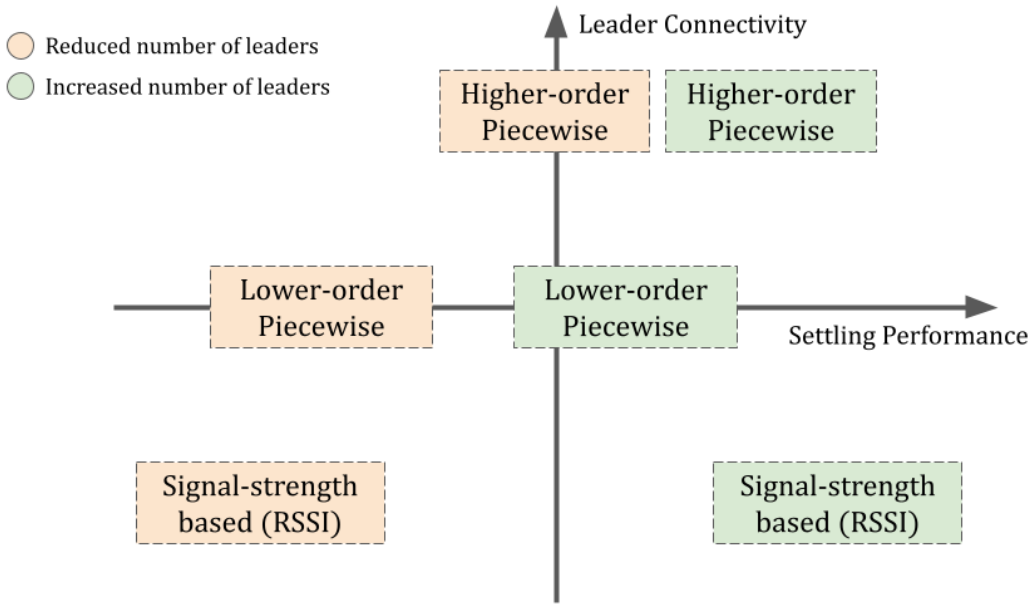


Figure 5.1: Settling performance of a MAS for different leader connectivity (global controllers) and number of leaders.

Furthermore, the performance of all controllers converges beyond a certain number of leaders, and identifying this threshold through simulations, in a deployment scenario, can reduce the impact of implementing a given controller. Finally, while the global controller helps achieve faster transience, the local controller maintains cohesion within the swarm and drives steady-state errors to zero. Using a larger gain for the local controller may be particularly prudent in order to minimise formation discontinuities when using lower-order piecewise controllers.

5.1 Summary

The study surveys the literature concerning formation control of multi-agent systems, with a focus on PDE-based models and experimental approaches. Global controllers requiring varying levels of leader connectivity — namely the piecewise constant, linear, quadratic, cubic and signal-strength based approximation — are proposed for estimating the states of the followers. The proposed controllers are subsequently implemented in MATLAB and Webots and their associated hardware and software-based constraints are discussed. The study analyses performance indices such as the L_2 error norm, formation settling time, and simulation complexity. Performance evaluations are conducted under various initial conditions, target curves, number of leaders, and controller gains. The study concludes by highlighting the inter-connectivity required amongst leaders based on the complexity of the target curve, the initial conditions, the controller gains, and the number of leaders in the MAS.

5.2 Future Work

There are several avenues for future work based on the findings of this study. Firstly, the extension of the proposed controllers to time-varying targets could be explored. Additionally, the optimal selection of global and local controller gains should be investigated to improve settling performance. In order to reduce a mismatch between the e-puck robots and their PDE-based model, the advection term of the PDE proposed in (3.1) could be extended to model higher-order and time-delayed dynamics of the agents. Furthermore, the PDE could be treated mathematically to quantitatively verify the closed-loop stability of the proposed controllers. The implementation of the proposed controllers on physical e-pucks and the correlation of the results with this study's simulation-based analysis should also be investigated. Such experimental studies may also consider exploring cost-efficient alternatives to using a high number of proximity distance sensors for implementing the local controller.

References

[1] Hazem Hussein. “Sheikh Zayed Festival breaks four Guinness World Records welcoming New Year of 2023”. In: *Emirates News Agency - WAM* (Jan. 2023). URL: https://wam.ae/en/details/1395303115517 (Accessed: 09/05/2023).

[2] Daniel S. Drew. “Multi-Agent Systems for Search and Rescue Applications”. In: *Current Robotics Reports* 2 (2 Mar. 2021), pp. 189–200. ISSN: 2662-4087. DOI: 10.1007/s43154-021-00048-3. URL: https://link.springer.com/10.1007/s43154-021-00048-3.

[3] Quoc-Viet Pham et al. “Swarm intelligence for next-generation networks: Recent advances and applications”. In: *Journal of Network and Computer Applications* 191 (Oct. 2021), p. 103141. ISSN: 10848045. DOI: 10.1016/j.jnca.2021.103141. URL: https://linkinghub.elsevier.com/retrieve/pii/S1084804521001582.

[4] Linling Wang et al. “A survey of underwater search for multi-target using Multi-AUV: Task allocation, path planning, and formation control”. In: *Ocean Engineering* 278 (June 2023), p. 114393. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2023.114393. URL: https://linkinghub.elsevier.com/retrieve/pii/S0029801823007771.

[5] J. Octavio Gutierrez-Garcia and Kwang Mong Sim. “Agent-based Cloud bag-of-tasks execution”. In: *Journal of Systems and Software* 104 (June 2015), pp. 17–31. ISSN: 01641212. DOI: 10.1016/j.jss.2015.02.039. URL: https://linkinghub.elsevier.com/retrieve/pii/S016412121500045X.

[6] Ghezlane Halhoul Merabet et al. “Applications of Multi-Agent Systems in Smart Grids: A survey”. In: IEEE, Apr. 2014, pp. 1088–1094. ISBN: 978-1-4799-3824-7. DOI: 10.1109/ICMCS.2014.6911384. URL: http://ieeexplore.ieee.org/document/6911384/.

[7] Hai Do et al. “Formation Control Algorithms for Multiple-UAVs: A Comprehensive Survey”. In: *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems* 8 (June 2021), p. 170230. DOI: 10.4108/eai.10-6-2021.170230.

[8] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. “A survey of multi-agent formation control”. In: *Automatica* 53 (Mar. 2015), pp. 424–440. ISSN: 00051098. DOI: 10.1016/j.automatica.2014.10.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109814004038.

[9] J. Alexander Fax and Richard M. Murray. “GRAPH LAPLACIANS AND STABILIZATION OF VEHICLE FORMATIONS”. In: *IFAC Proceedings Volumes* 35 (1 2002), pp. 55–60. ISSN: 14746670. DOI: 10.3182/20020721-6-ES-1901.00090. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667015385116.

[10] H.G. Tanner, A. Jadbabaie, and G.J. Pappas. “Stable flocking of mobile agents. I. Fixed topology”. In: vol. 2. IEEE, Dec. 2003, pp. 2010–2015. ISBN: 0-7803-7924-1. DOI: 10.1109/CDC.2003.1272910. URL: http://ieeexplore.ieee.org/document/1272910/.

[11] R. Olfati-Saber and R.M. Murray. “Consensus Problems in Networks of Agents With Switching Topology and Time-Delays”. In: *IEEE Transactions on Automatic Control* 49 (9 Sept. 2004), pp. 1520–1533. ISSN: 0018-9286. DOI: 10.1109/TAC.2004.834113. URL: http://ieeexplore.ieee.org/document/1333204/.

[12] David Angeli and Pierre-Alexandre Bliman. “Stability of leaderless discrete-time multi-agent systems”. In: *Mathematics of Control, Signals, and Systems* 18 (4 Oct. 2006), pp. 293–322. ISSN: 0932-4194. DOI: 10.1007/s00498-006-0006-0. URL: http://link.springer.com/10.1007/s00498-006-0006-0.

[13] L. Moreau. “Stability of continuous-time distributed consensus algorithms”. In: vol. 4. IEEE, Dec. 2004, 3998–4003 Vol.4. ISBN: 0-7803-8682-5. DOI: 10.1109/CDC.2004.1429377. URL: http://ieeexplore.ieee.org/document/1429377/.

[14] Pierre-Alexandre Bliman and Giancarlo Ferrari-Trecate. “Average consensus problems in networks of agents with delayed communications”. In: *Automatica* 44 (8 Aug. 2008), pp. 1985–1995. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2007.12.010. URL: https://www.sciencedirect.com/science/article/pii/S0005109808000344.

[15] Zhiyun Lin, B Francis, and M Maggiore. “Necessary and sufficient graphical conditions for formation control of unicycles”. In: *IEEE Transactions on Automatic Control* 50 (1 Jan. 2005), pp. 121–127. ISSN: 0018-9286. DOI: 10.1109/TAC.2004.841121. URL: http://ieeexplore.ieee.org/document/1381658/.

[16] Michael J. Caruso. “Applications of magnetic sensors for low cost compass systems”. In: *Record - IEEE PLANS, Position Location and Navigation Symposium* (2000), pp. 177–184. DOI: 10.1109/plans.2000.838300.

[17] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. “Programmable self-assembly in a thousand-robot swarm”. In: *Science* 345 (6198 Aug. 2014), pp. 795–799. ISSN: 0036-8075. DOI: 10.1126/science.1254295. URL: https://www.science.org/doi/10.1126/science.1254295.

[18] Andrew Howard, Lynne E. Parker, and Gaurav S. Sukhatme. “Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection”. In: *The International Journal of Robotics Research* 25 (5-6 May 2006), pp. 431–447. ISSN: 0278-3649. DOI: 10.1177/0278364906065378. URL: http://journals.sagepub.com/doi/10.1177/0278364906065378.

[19] F. Dellaert, F. Alegre, and E.B. Martinson. “Intrinsic Localization and Mapping with 2 applications: Diffusion Mapping and Macro Polo localization”. In: vol. 2. IEEE, 2003, pp. 2344–2349. ISBN: 0-7803-7736-2. DOI: 10.1109/ROBOT.2003.1241943. URL: http://ieeexplore.ieee.org/document/1241943/.

[20] Ling Mao et al. “Relative Localization Method of Multiple Micro Robots Based on Simple Sensors”. In: *International Journal of Advanced Robotic Systems* 10 (2 Feb. 2013), p. 128. ISSN: 1729-8814. DOI: 10.5772/55587. URL: http://journals.sagepub.com/doi/10.5772/55587.

[21] Satoki Ogiso et al. “Self-localization method for mobile robot using acoustic beacons”. In: *ROBOMECH Journal* 2 (1 Dec. 2015), p. 12. ISSN: 2197-4225. DOI: 10.1186/s40648-015-0034-y. URL: http://robomechjournal.springeropen.com/articles/10.1186/s40648-015-0034-y.

[22] Dušan Nemec et al. “Mutual acoustic identification in the swarm of e-puck robots”. In: *International Journal of Advanced Robotic Systems* 14 (3 May 2017), p. 172988141771079. ISSN: 1729-8814. DOI: 10.1177/1729881417710794. URL: http://journals.sagepub.com/doi/10.1177/1729881417710794.

[23] J. Hvizdos, J. Vascak, and A. Brezina. “Object identification and localization by smart floors”. In: IEEE, Sept. 2015, pp. 113–117. ISBN: 978-1-4673-7939-7. DOI: 10.1109/INES.2015.7329649. URL: https://ieeexplore.ieee.org/document/7329649.

[24] Han Wu et al. “Precise Localization and Formation Control of Swarm Robots via Wireless Sensor Networks”. In: *Mathematical Problems in Engineering* 2014 (2014). Ed. by Guangming Xie, pp. 1–12. ISSN: 1024-123X. DOI: 10.1155/2014/942306. URL: http://www.hindawi.com/journals/mpe/2014/942306/.

[25] Gerhard Freudenthaler and Thomas Meurer. “PDE-based multi-agent formation control using flatness and backstepping: Analysis, design and robot experiments”. In: *Automatica* 115 (May 2020), p. 108897. ISSN: 00051098. DOI: 10.1016/j.automatica.2020.108897. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109820300959.

[26] Vahid Rahmani and Vahid Rostami. “Adaptive Color Mapping for NAO Robot Using Neural Network”. In: *Advances in Computer Science : an International Journal* 3 (2014), pp. 66–71. ISSN: 2322-5157. URL: https://www.academia.edu/8648665/Adaptive_Color_Mapping_for_NAO_Robot_Using_Neural_Network.

[27] Giancarlo Ferrari-Trecate, Annalisa Buffa, and Mehdi Gati. “ANALYSIS OF COORDINATION IN MULTI-AGENT SYSTEMS THROUGH PARTIAL DIFFERENCE EQUATIONS. PART I: THE LAPLACIAN CONTROL”. In: *IFAC Proceedings Volumes* 38 (1 2005), pp. 203–208. ISSN: 14746670. DOI: 10.3182/20050703-6-CZ-1902.02086. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667016380983.

[28] Samuel Coogan and Murat Arcak. “Scaling the size of a formation using relative position feedback”. In: *Automatica* 48 (10 Oct. 2012), pp. 2677–2685. ISSN: 00051098. DOI: 10.1016/j.automatica.2012.06.083. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109812003536.

[29] John Toner and Yuhai Tu. “Flocks, herds, and schools: A quantitative theory of flocking”. In: *Physical Review E* 58 (4 Oct. 1998), pp. 4828–4858. ISSN: 1063-651X. DOI: 10.1103/PhysRevE.58.4828. URL: https://link.aps.org/doi/10.1103/PhysRevE.58.4828.

[30] Craig W. Reynolds. “Flocks, herds and schools: A distributed behavioral model”. In: *ACM SIGGRAPH Computer Graphics* 21 (4 Aug. 1987), pp. 25–34. ISSN: 0097-8930. DOI: 10.1145/37402.37406. URL: https://dl.acm.org/doi/10.1145/37402.37406.

[31] Paul Frihauf and Miroslav Krstic. “Leader-Enabled Deployment Onto Planar Curves: A PDE-Based Approach”. In: *IEEE Transactions on Automatic Control* 56 (8 Aug. 2011),

pp. 1791–1806. ISSN: 0018-9286. DOI: 10.1109/TAC.2010.2092210. URL: http://ieeexplore.ieee.org/document/5635320/.

[32] Luciano C. A. Pimenta et al. “Control of swarms based on Hydrodynamic models”. In: IEEE, May 2008, pp. 1948–1953. ISBN: 978-1-4244-1646-2. DOI: 10.1109/ROBOT.2008.4543492. URL: http://ieeexplore.ieee.org/document/4543492/.

[33] P. Barooah, P.G. Mehta, and J.P. Hespanha. “Mistuning-Based Control Design to Improve Closed-Loop Stability Margin of Vehicular Platoons”. In: *IEEE Transactions on Automatic Control* 54 (9 Sept. 2009), pp. 2100–2113. ISSN: 0018-9286. DOI: 10.1109/TAC.2009.2026934. URL: http://ieeexplore.ieee.org/document/5208241/.

[34] E.W. Justh and P.S. Krishnaprasad. “Steering laws and continuum models for planar formations”. In: vol. 4. IEEE, Apr. 2004, pp. 3609–3614. ISBN: 0-7803-7924-1. DOI: 10.1109/CDC.2003.1271708. URL: http://ieeexplore.ieee.org/document/1271708/.

[35] Jie Qi, Rafael Vazquez, and Miroslav Krstic. “Multi-Agent Deployment in 3-D via PDE Control”. In: *Automatic Control, IEEE Transactions on* 60 (Apr. 2015), pp. 891–906. DOI: 10.1109/TAC.2014.2361197.

[36] Pablo Hernández-León et al. “Distance-Based Formation Maneuvering of Non-Holonomic Wheeled Mobile Robot Multi-Agent System”. In: *21st IFAC World Congress* 53 (2 July 2020). 21st IFAC World Congress, pp. 5665–5670. ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.12.1588. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896320321844.

[37] Jieqiang Wei et al. “Multi-agent deployment under the leader displacement measurement: a PDE-based approach”. In: IEEE, June 2019, pp. 2424–2429. ISBN: 978-3-907144-00-8. DOI: 10.23919/ECC.2019.8796132. URL: https://ieeexplore.ieee.org/document/8796132/.

[38] Jieqiang Wei, Emilia Fridman, and Karl Henrik Johansson. “A PDE approach to deployment of mobile agents under leader relative position measurements”. In: *Automatica* 106 (Aug. 2019), pp. 47–53. ISSN: 00051098. DOI: 10.1016/j.automatica.2019.04.040. URL: https://linkinghub.elsevier.com/retrieve/pii/S0005109819302006.

[39] Anton Selivanov and Emilia Fridman. “PDE-Based Deployment of Multiagents Measuring Relative Position to One Neighbor”. In: *IEEE Control Systems Letters* 6 (2022),

pp. 2563–2568. ISSN: 2475-1456. DOI: 10.1109/LCSYS.2022.3169999. URL: https://ieeexplore.ieee.org/document/9762356/.

[40] Shu-Xia Tang, Jie Qi, and Jing Zhang. “Formation tracking control for multi-agent systems: A wave-equation based approach”. In: *International Journal of Control, Automation and Systems* 15 (Dec. 2017), pp. 2704–2713. DOI: 10.1007/s12555-016-0562-0.

[41] G. Freudenthaler and T. Meurer. “PDE-based tracking control for multi-agent deployment”. In: *10th IFAC Symposium on Nonlinear Control Systems* 49 (18 Aug. 2016), pp. 582–587. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.10.228. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896316318080.

[42] Gordon Everstine. *Analytical Solution of Partial Differential Equations*. The George Washington University, Apr. 2012, pp. 103–109. URL: https://www.yumpu.com/en/document/read/5503140/analytical-solution-of-partial-differential-equations-gordon-everstine.

[43] GCtronic WikiSysop. “e-puck2”. In: *GCtronic wiki* (Mar. 2023). URL: https://www.gctronic.com/doc/index.php/e-puck2 (Accessed: 09/05/2023).

[44] Cyberbotics. “Webots User Guide: GCTronic’ e-puck”. In: *Cyberbotics* (2023). URL: https://cyberbotics.com/doc/guide/epuck (Accessed: 09/05/2023).

[45] *Engineering ToolBox. Friction - Friction Coefficients and Calculator*. 2004. URL: https://www.engineeringtoolbox.com/friction-coefficients-d_778.html (Accessed: 09/05/2023).

[46] *Distance Measuring Sensor Unit. GP2Y0D21YK0F*. SHARP Corporation. Dec. 2006. URL: https://docs.rs-online.com/c43f/0900766b80d1bdd1.pdf (Accessed: 09/05/2023).

[47] Ivan Livaja et al. “A distributed geospatial publish/subscribe system on Apache Spark”. In: *Future Generation Computer Systems* 132 (July 2022), pp. 282–298. ISSN: 0167739X. DOI: 10.1016/j.future.2022.02.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X22000577.

[48] Microchip. “Bluetooth® Low Energy Connection Process”. In: *Developer Help* (2021). URL: https://www.microchipdeveloper.com/wireless:ble-link-layer-connections (Accessed: 09/05/2023).

[49] Antonio Del Campo et al. “Analysis and Tools for Improved Management of Connectionless and Connection-Oriented BLE Devices Coexistence”. In: *Sensors (Basel, Switzerland)* 17 (4 Apr. 2017), p. 792. ISSN: 1424-8220. DOI: 10.3390/s17040792. URL: http://www.mdpi.com/1424-8220/17/4/792.

[50] Autonomous Systems Lab and ETH Zurich. “Mobile Robots Kinematics, Lecture Notes: Autonomous Mobile Robots”. In: *Robotics and Intelligent Systems* (2018). URL: https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/asl-dam/documents/lectures/autonomous_mobile_robots/spring-2018/kinematics_2.pdf.

[51] Shamoil Khomosi. “Webots Demonstration: Multi-agent System Control”. In: *YouTube* (Apr. 2023). URL: https://youtu.be/h-m6VeHH6ec (Accessed: 09/05/2023).

[52] O. Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: vol. 2. Institute of Electrical and Electronics Engineers, 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247. URL: http://ieeexplore.ieee.org/document/1087247/.

[53] Paolo Fiorini and Zvi Shiller. “Motion Planning in Dynamic Environments Using Velocity Obstacles”. In: *The International Journal of Robotics Research* 17 (7 July 1998), pp. 760–772. ISSN: 0278-3649. DOI: 10.1177/027836499801700706. URL: http://journals.sagepub.com/doi/10.1177/027836499801700706.

[54] Run Mao, Hongli Gao, and Liang Guo. “A Novel Collision-Free Navigation Approach for Multiple Nonholonomic Robots Based on ORCA and Linear MPC”. In: *Mathematical Problems in Engineering* 2020 (June 2020), pp. 1–16. ISSN: 1024-123X. DOI: 10.1155/2020/4183427. URL: https://www.hindawi.com/journals/mpe/2020/4183427/.

[55] Jur van den Berg et al. “Reciprocal n-Body Collision Avoidance”. In: *Robotics Research* (2011). Ed. by Roland, Hirzinger Gerhard Pradalier Cédric, and Siegwart, pp. 3–19. DOI: 10.1007/978-3-642-19457-3_1. URL: http://link.springer.com/10.1007/978-3-642-19457-3_1.

Appendix A

Appendix

A.1 Original aims and objectives

Formation control is an active area of research within multi-agent systems (MAS). The effectiveness of a displacement-based formation control strategy may be dependent on various factors, including sensing capabilities, range, connectivity and the number of individual agents. This research aims at investigating the effects of increased connectivity amongst the leaders, and finer piecewise approximations for controlling the followers, on the performance of such a MAS that is modelled by a diffusion PDE. Unlike ordinary differential equations (ODE), partial differential equations (PDE) provide a method to model large-scale MAS whose complexity does not grow as the number of agents increases. The results will be validated via software simulation and hardware implementation.

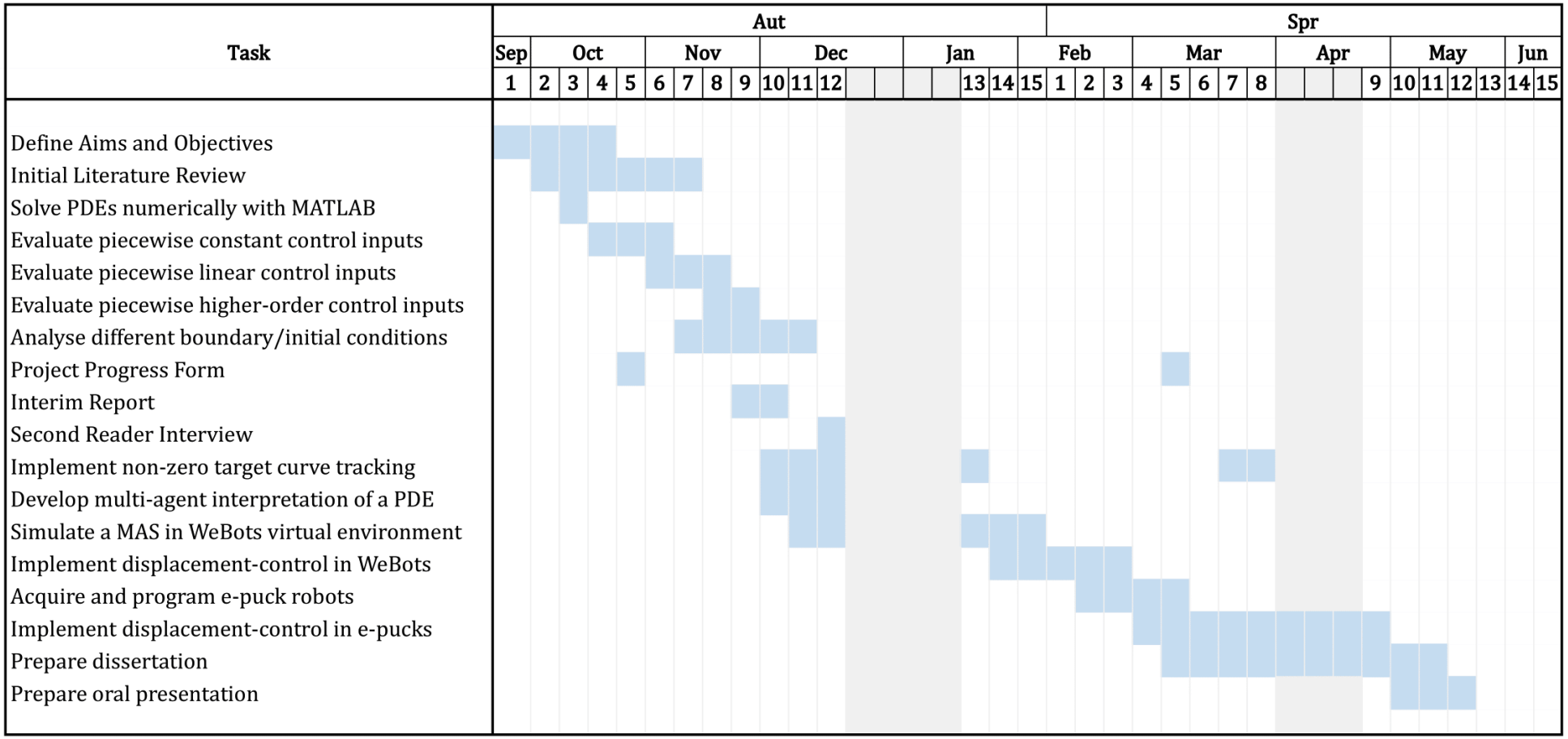
Basic Objectives:

1. Review the literature concerning PDE-based analysis and formation control of leader-follower MAS.
2. Develop a MATLAB program to numerically solve PDEs via the method of lines.
3. Design a MATLAB program with control inputs for followers corresponding to piecewise constant and piecewise linear functions.
4. Compare the performance of the MAS with increased connectivity between the leaders for different initial and boundary conditions. leader-follower control inputs.

Advanced Objectives:

1. Design a MATLAB program with control inputs for followers corresponding to higher-order piecewise functions.
2. Emulate a MAS consisting of e-puck robots in Webots Simulator with varying leader connectivity and leader-follower control inputs.
3. Implement the above simulations with e-puck robot hardware in the Sheffield Robotics Lab.

Original Gantt chart



A.2 Simulation framework

The simulation framework developed for this dissertation is available in a Git repository at https://github.com/shamoilkhomosi/multi-agents. The framework uses MATLAB and Webots to implement and simulate multi-agent systems using various communication and control strategies. The MATLAB code simulates the control algorithms for the agents, while the Webots code provides the physics and graphics engine for the simulation environment.

This framework facilitates the simulation of different operating scenarios to demonstrate the performance of the multi-agent system when the number of agents, number of leaders, global controller topology, local and global controller gains, initial and boundary conditions, and the target curve are varied.

The numerical simulations in this study are carried out using a central MATLAB script. The multi-agent system is simulated by solving discretised ODE functions with a fixed time step. The ODE functions for different global controllers are stored in separate files and called by the central script as needed. The solutions to the ODEs, which correspond to the agents' states, are then used to calculate various performance metrics, such as settling times, standard deviations, and L_2 norms. These metrics are also plotted to visualise the results, and they are used throughout this dissertation to analyse the performance of the system.

The Webots simulations are sequenced and executed autonomously via a Python script where the user can enter the desired simulation parameters. Based on the user's input, the script generates the Webots world file, updates the controllers of the agents, runs the simulation via Webots' command line interface, and stores the performance logs and simulation results in the working directory. These log files can be loaded and analysed using MATLAB.

The simulation scenarios available in the framework can be modified and extended to suit different multi-agent system applications. To use the simulation framework, interested readers may follow the instructions provided in the Git repository.